**LABORATORY WORK №1**

**Subject: «Number systems. Transfer numbers from one number system to another. Arithmetic in the number systems.»**

**Purpose:** To acquire the skills of operations in different number systems.

There are many methods or techniques which can be used to convert numbers from one base to another. We'll demonstrate here the following

* Decimal to Other Base System
* Other Base System to Decimal
* Other Base System to Non-Decimal
* Shortcut method - Binary to Octal
* Shortcut method - Octal to Binary
* Shortcut method - Binary to Hexadecimal
* Shortcut method - Hexadecimal to Binary

**Decimal to Other Base System**

Steps

* **Step 1** - Divide the decimal number to be converted by the value of the new base.
* **Step 2** - Get the remainder from Step 1 as the rightmost digit (least significant digit) of new base number.
* **Step 3** - Divide the quotient of the previous divide by the new base.
* **Step 4** - Record the remainder from Step 3 as the next digit (to the left) of the new base number.

Repeat Steps 3 and 4, getting remainders from right to left, until the quotient becomes zero in Step 3.

The last remainder thus obtained will be the most significant digit (MSD) of the new base number.

**Example**

Decimal Number: 27310

Calculating Binary Equivalent:

 546 : 2 = 273, **Remainder 0**

 273 : 2 = 136, **Remainder 1**

 136 : 2 = 68, **Remainder 0**

 68 : 2 = 34, **Remainder 0**

 34 : 2 = 17, **Remainder 0** Binary Number 546(10) = 1000100010(2)

 17 : 2 = 8, **Remainder 1**

 8 : 2 = 4, **Remainder 0**

 4 : 2 = 2, **Remainder** **0**

 2 : 2 = **1**, **Remainder 0**

Decimal Number: 27310 = Binary Number: 10001000102

.

**Other base system to Decimal System**

Steps

* **Step 1** - Determine the column (positional) value of each digit (this depends on the position of the digit and the base of the number system).
* **Step 2** - Multiply the obtained column values (in Step 1) by the digits in the corresponding columns.
* **Step 3** - Sum the products calculated in Step 2. The total is the equivalent value in decimal.

**Example**

Binary Number: 111012

Calculating Decimal Equivalent:

|  |  |  |
| --- | --- | --- |
| **Step** | **Binary Number** | **Decimal Number** |
| Step 1 | 111012 | ((1 x 24) + (1 x 23) + (1 x 22) + (0 x 21) + (1 x 20))10 |
| Step 2 | 111012 | (16 + 8 + 4 + 0 + 1)10 |
| Step 3 | 111012 | 2910 |

Binary Number: 111012 = Decimal Number: 2910

**Other Base System to Non-Decimal System**

Steps

* **Step 1** - Convert the original number to a decimal number (base 10).
* **Step 2** - Convert the decimal number so obtained to the new base number.

**Example**

Octal Number: 258

Calculating Binary Equivalent:

**Step 1: Convert to Decimal**

|  |  |  |
| --- | --- | --- |
| **Step** | **Octal Number** | **Decimal Number** |
| Step 1 | 258 | ((2 x 81) + (5 x 80))10 |
| Step 2 | 258 | (16 + 5 )10 |
| Step 3 | 258 | 2110 |

Octal Number: 258 = Decimal Number: 2110

**Step 2: Convert Decimal to Binary**

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Operation** | **Result**  | **Remainder** |
| Step 1 | 21 / 2 | 10 | 1 |
| Step 2 | 10 / 2 | 5 | 0 |
| Step 3 | 5 / 2 | 2 | 1 |
| Step 4 | 2 / 2 | 1 | 0 |
| Step 5 | 1 / 2 | 0 | 1 |

Decimal Number: 2110 = Binary Number: 101012

Octal Number: 258 = Binary Number: 101012

**Shortcut method - Binary to Octal**

Steps

* **Step 1** - Divide the binary digits into groups of three (starting from the right).
* **Step 2** - Convert each group of three binary digits to one octal digit.

**Example**

Binary Number: 101012

Calculating Octal Equivalent:

|  |  |  |
| --- | --- | --- |
| **Step** | **Binary Number** | **Octal Number** |
| Step 1 | 101012 | 010 101 |
| Step 2 | 101012 | 28 58 |
| Step 3 | 101012 | 258 |

Binary Number: 101012 = Octal Number: 258

**Shortcut method - Octal to Binary**

Steps

* **Step 1** - Convert each octal digit to a 3 digit binary number (the octal digits may be treated as decimal for this conversion).
* **Step 2** - Combine all the resulting binary groups (of 3 digits each) into a single binary number.

**Example**

Octal Number: 258

Calculating Binary Equivalent:

|  |  |  |
| --- | --- | --- |
| **Step** | **Octal Number** | **Binary Number** |
| Step 1 | 258 | 210 510 |
| Step 2 | 258 | 0102 1012 |
| Step 3 | 258 | 0101012 |

Octal Number: 258 = Binary Number: 101012

**Shortcut method - Binary to Hexadecimal**

Steps

* **Step 1** - Divide the binary digits into groups of four (starting from the right).
* **Step 2** - Convert each group of four binary digits to one hexadecimal symbol.

**Example**

Binary Number: 101012

Calculating hexadecimal Equivalent:

|  |  |  |
| --- | --- | --- |
| **Step** | **Binary Number** | **Hexadecimal Number** |
| Step 1 | 101012 | 0001 0101 |
| Step 2 | 101012 | 110 510 |
| Step 3 | 101012 | 1516 |

Binary Number: 101012 = Hexadecimal Number: 1516

**Shortcut method - Hexadecimal to Binary**

Steps

* **Step 1** - Convert each hexadecimal digit to a 4 digit binary number (the hexadecimal digits may be treated as decimal for this conversion).
* **Step 2** - Combine all the resulting binary groups (of 4 digits each) into a single binary number.

**Example**

Hexadecimal Number: 1516

Calculating Binary Equivalent:

|  |  |  |
| --- | --- | --- |
| **Step** | **Hexadecimal Number** | **Binary Number** |
| Step 1 | 1516 | 110 510 |
| Step 2 | 1516 | 00012 01012 |
| Step 3 | 1516 | 000101012 |

Hexadecimal Number: 1516 = Binary Number: 101012

**Translation of proper fractions of a decimal number system to another number system.**

To transfer the right to another decimal number system is necessary to multiply this fraction series on the basis of the system into which it is translated . This is multiplied by the fractional part only . Fraction in the new notation is written in the form of whole parts of the work, starting with the first . For example, the number 0322(10) to translate into other number system.

**Example**

0,322\*2=0,644 **0** 0,322\*8=2,576 **2** 0,322\*16=5,152 **5**

0,644\*2=1,288 **1** 0,576\*8=4,608 **4** 0,152\*16=2,432 **2**

0,288\*2=0,576 **0** 0,608\*8=4,864 **4** 0,432\*16=9,912 **9**

0,576\*2=1,152 **1** 0,864\*8=6,912 **6** 0,912\*16=14,592 **F**,

0,152\*2=0,304 **0**

$0,322\_{(10)}$= $0,0101\_{(2) }$= $0,2446\_{(8)}$= $0,529F\_{(16)}$.

**Tasks**

 **Complete the table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variant**  | **Decimal number** | **Binary Number** | **Octal Number** | **Hexadecimal Number** |
| **1** | 358.95 |  |  |  |
|  | 1101101011 |  |  |
|  |  | 547 |  |
|  |  |  | 164A |
| **2** | 634.67 |  |  |  |
|  | 1101111010101 |  |  |
|  |  | 472 |  |
|  |  |  | 7AC |
| **3** | 582.02 |  |  |  |
|  |  | 364 |  |
|  |  |  | 1F6E |
|  | 1000111101010 |  |  |
| **4** | 369.025 |  |  |  |
|  | 100001111101 |  |  |
|  |  | 641 |  |
|  |  |  | 4D61 |
| **5** | 468.15 |  |  |  |
|  | 11010001011 |  |  |
|  |  | 734 |  |
|  |  |  | 2D4A |
| **6** | 654.27 |  |  |  |
|  | 1100001010101 |  |  |
|  |  | 632 |  |
|  |  |  | 5AD |
| **7** | 286.52 |  |  |  |
|  | 1011111101011 |  |  |
|  |  | 274 |  |
|  |  |  | 1D8E |
| **8** | 492.025 |  |  |  |
|  | 1111011001101 |  |  |
|  |  | 375 |  |
|  |  |  | 4C61 |
| **9** | 417.75 |  |  |  |
|  | 110011101011 |  |  |
|  |  | 737 |  |
|  |  |  | 952F |
| **10** | 744.67 |  |  |  |
|  | 1101101010101 |  |  |
|  |  | 267 |  |
|  |  |  | 4D67 |

**Amdahl's law**

In [computer architecture](https://en.wikipedia.org/wiki/Computer_architecture), Amdahl's law gives the theoretical [speedup](https://en.wikipedia.org/wiki/Speedup) in [latency](https://en.wikipedia.org/wiki/Latency_%28engineering%29%22%20%5Co%20%22Latency%20%28engineering%29)of the execution of a task at fixed [workload](https://en.wikipedia.org/wiki/Workload) that can be expected of a system whose resources are improved. It is named after computer scientist [Gene Amdahl](https://en.wikipedia.org/wiki/Gene_Amdahl), and was presented at the [AFIPS](https://en.wikipedia.org/wiki/American_Federation_of_Information_Processing_Societies) Spring Joint Computer Conference in 1967.

Amdahl's law can be formulated the following way:



where

Slatency is the theoretical speedup in latency of the execution of the whole task;

s is the speedup in latency of the execution of the part of the task that benefits from the improvement of the resources of the system;

p is the percentage of the execution time of the whole task concerning the part that benefits from the improvement of the resources of the system before the improvement.

Furthermore,



show that the theoretical speedup of the execution of the whole task increases with the improvement of the resources of the system and that regardless the magnitude of the improvement, the theoretical speedup is always limited by the part of the task that cannot benefit from the improvement.

Amdahl's law is often used in [parallel computing](https://en.wikipedia.org/wiki/Parallel_computing) to predict the theoretical speedup when using multiple processors. For example, if a program needs 20 hours using a single processor core, and a particular part of the program which takes one hour to execute cannot be parallelized, while the remaining 19 hours (p = 0.95) of execution time can be parallelized, then regardless of how many processors are devoted to a parallelized execution of this program, the minimum execution time cannot be less than that critical one hour. Hence, the theoretical speedup is limited to at most 20 times (1/(1 − p) = 20).

Example 1

If 30% of the execution time may be the subject of a speedup, p will be 0.3; if the improvement makes the affected part twice faster, s will be 2. Amdahl's law states that the overall speedup of applying the improvement will be



Example 2

We are given a serial task which is split into four consecutive parts, whose percentages of execution time are p1 = 0.11, p2 = 0.18, p3 = 0.23, and p4 = 0.48 respectively. Then we are told that the 1st part is not sped up, so s1 = 1, while the 2nd part is sped up 5 times, so s2 = 5, the 3rd part is sped up 20 times, so s3 = 20, and the 4th part is sped up 1.6 times, so s4 = 1.6. By using Amdahl's law, the overall speedup is



Notice how the 20 times and 5 times speedup on the 2nd and 3rd parts respectively don't have much effect on the overall speedup when the 4th part (48% of the execution time) is sped up only 1.6 times.