*Proceedings of the 2004 International Conference on Algorithmic Mathematics and Computer Science (AMCS'04) pp. 317-320, Las Vegas, Nevada, June, 2004*

# On the Conversion Between Number Systems

Houssain Kettani
Department of Computer Science
Jackson State University
Jackson, MS 39217
houssain.kettani@jsums.edu

## Abstract

*In this paper, we revisit the problem of conversion between number systems. We present a conversion function that relates each digit in a base $b$ system to the decimal value that is equal to the base $b$ number in question. Thus, each base $b$ digit of the related base $b$ number can be obtained directly from the corresponding decimal number without the requirement of knowing any other base $b$ digit.*

## 1 Introduction

We represent an unsigned $b$-ary, radix $b$, or base $b$ number as the following string of digits:

$$(d_n d_{n-1} \dots d_0 . d_{-1} d_{-2} \dots d_{-m+1} d_{-m})_b,$$

where the integer $b \geq 2$, $d_j \in \{0, 1, \dots, b-1\}$ for $j = n, n-1, \dots, -m+1, -m$, and "." is the radix point. The term to the left of the radix point is referred to as the integer part, while that to the right of the radix point is referred to as the fractional part. The everyday number system that we use is the decimal (base 10). Other number systems that are used in computer work are binary (base 2), octal (base 8), and hexadecimal (base 16).

The classical way to convert a base $b_1$ number to another number in base $b_2$, so that both have the same decimal value, is to first convert base $b_1$ to decimal then convert from the latter to base $b_2$ — for example, see [3].

A more convenient conversion method is adopted when $b_1 = b_2^p$ where $p$ is a positive decimal integer. Then, to convert from base $b_2$ to base $b_1$, we gather $p$ base $b_1$ digits (from right to left for the integer part, and from left to right for the fraction part) into one base $b_2$ digit so that both have the same decimal value. On the other hand, to convert from base $b_1$ to base $b_2$, we expand one base $b_2$ digit into $p$ base $b_1$ digits so that both have the same decimal value. In other words, the relation between base $b$ and base $b^p$ is given by

$$(\dots d_1 d_0 . d_{-1} \dots)_b = (\dots d'_1 d'_0 . d'_{-1} \dots)_{b^p},$$

where

$$\begin{aligned} d'_j &= (d_{pj+p-1} \dots d_{pj+1} d_{pj})_b \\ &= \sum_{i=0}^{p-1} d_{pj+i} b^i. \end{aligned}$$

To illustrate this point, consider conversion between binary and hexadecimal. In this case we have $b_2 = 2$ and $p = 4$. As an illustrative numerical example, we have $(00100101)_2 = (25)_{16}$.

Suppose now we would like to write

$$(d)_{10} = (d_n d_{n-1} \dots d_0 . d_{-1} d_{-2} \dots d_{-m+1} d_{-m})_b,$$

where the right hand side is a decimal number and the left hand side is a base $b$ number. Then, by definition, to convert from base $b$ to decimal , we write

$$d = \sum_{j=-m}^{n} d_j b^j. \quad (1)$$

Alternatively, the classical way to convert from decimal to base $b$, is to repeatedly divide the integer part of $d$ by $b$ and record the remainders. Then, starting with the first recorded one, we write down these remainders from right to left starting to the left of the radix point. Thus, we obtain the integer part of the corresponding base $b$ number.

On the other hand, the fraction part is obtained as follows. We repeatedly multiply the fraction part of $d$ by $b$ and record the resultant integer. Then, starting with the first recorded one, we write down these integers from left to right starting to the right of the radix point. Thus, we obtain the fraction part of the corresponding base $b$ number — see [3] for details and illustrative examples.

However, is there any simple function that we can use to convert from decimal to base $b$? Moreover, what if we are only interested in the value of $d_j$ for some $j \geq 0$? Then based on the repeated division algorithm, we need to compute all the values $d_k$ for $k = 0, 1, \dots, j$. Also, what if we are only interested in the value of $d_j$ for some $j < 0$? Then based on the repeated multiplication algorithm, we need to compute all the values $d_k$ for $k = -1, -2, \dots, j$.

Section 2 answers the above questions and presents a very simple and useful result. A summary of this paper and further research directions are presented in Section 3.

# 2 Conversion between Decimal and Base b Numbers

The following theorem answers the questions posed in the introduction and presents a very simple and useful result. By this theorem, the $d_j$'s are directly accessible without the requirement of computing any other $d_j$ except the one of interest. In this theorem, we use $\lfloor \cdot \rfloor$ to indicate the floor function. We note here that the result presented here is stronger than that presented in [1]. In the latter, the only if result was presented.

## 2.1 Theorem

*Let $d \geq 0$ be a decimal number and $b \geq 2$ be a decimal integer. Let also $d_j \in \{0, 1, \dots, b-1\}$ for $j = n, n-1, \dots, -m+1, -m$. Suppose that the decimal number $d$ is expressible as*

$$(d)_{10} = (d_n d_{n-1} \dots d_0 . d_{-1} \dots d_{-m+1} d_{-m})_b. \quad (2)$$

*Then, (2) is equivalent to*

$$d_j = \lfloor db^{-j} \rfloor - b \lfloor db^{-j-1} \rfloor$$

## 2.2 Proof of Theorem

In this section, we prove Theorem 2.1. To prove the if part, we have

$$\begin{aligned} \sum_{j=-m}^{n} d_j b^j &= \sum_{j=-m}^{n} (\lfloor db^{-j} \rfloor - b \lfloor db^{-j-1} \rfloor) b^j \\ &= \sum_{j=-m}^{n} \lfloor db^{-j} \rfloor b^j - \sum_{j=-m}^{n} \lfloor db^{-j-1} \rfloor b^{j+1} \\ &= \lfloor db^m \rfloor b^{-m} - \lfloor db^{-n-1} \rfloor b^{n+1}. \end{aligned}$$

Now, note that the second term is equal to zero since $0 \leq db^{-n-1} \leq 1$. This is the case since otherwise we need an extra digit to the left to represent $d$. Note also that the term $db^m$ is an integer. This is also the case since otherwise we need an extra digit to the right to represent $d$. Thus,

$$\sum_{j=-m}^{n} d_j b^j = d,$$

and this concludes the proof of the first part of the theorem.

Before proceeding to prove the second part of the theorem, we adopt the following generalization of the divisibility concept.

**Generalized Divisibility:** *Let $a_1$ and $a_2$ be two real numbers. We say $a_1$ is divisible by $a_2$ if the ratio $a_1/a_2$ is an integer.*

Now, to prove the only if part, we first let $y = db^{-j-1}$, and consider two cases:

**Case 1:** $y \in \mathbb{N}$**.** This means that $d$ is divisible by $b^{j+1}$. This in turn, forces $d_k = 0$ for $k \leq j$. Accordingly, $\lfloor by \rfloor - b\lfloor y \rfloor = 0$.

**Case 2:** $y \notin \mathbb{N}$**.** Then we can write $y = x + \epsilon$, where $x \in \mathbb{N}$ and $\epsilon \in (0, 1)$. Thus,

$$\lfloor by \rfloor - b\lfloor y \rfloor = \lfloor b\epsilon \rfloor.$$

On the other hand, we can write

$$d = xb^{j+1} + \epsilon b^{j+1}.$$

Now, the term $xb^{j+1}$ would modify $d_k$ for some $k > j$. To see the effect on $d_j$, we need to observe the term $\epsilon b^{j+1}$. To do this, we partition the interval $(0, 1)$ into $b$ subintervals of equal length. Thus, we consider the following cases $\epsilon \in [\frac{b-i}{b}, \frac{b-i+1}{b})$, $i = b, b-1, \ldots, 1$. Therefore, we have $(b-i)b^j \leq \epsilon b^{j+1} < (b-i+1)b^j$, which implies that $d_j = b-i$. Accordingly, $\lfloor b\epsilon \rfloor = b-i$. This concludes the proof.

### 2.3 Remarks

The minimum value of index $n$ for which (2) holds can be easily expressed from (1) and is given by

$$n_{min} = \lfloor log_b d \rfloor.$$

The minimum value of index $m$, on the other hand, is not as elegantly expressible. In fact, $m$ may diverge to infinity. For example, consider the decimal number $0.1$, then we have

$$(0.1)_{10} = (0.0001100110011\ldots)_2.$$

In this case, although the decimal fractional part has finite number of digits, the corresponding binary fractional part does not have finite number of bits.

Finally, we note that it can easily be shown that the values of the $d_j$'s in the theorem are also equal to $\lfloor b(db^{-j-1} - \lfloor db^{-j-1} \rfloor) \rfloor$ and $\lfloor b^{-j} mod(d, b^{j+1}) \rfloor$.

## 3 Summary and Further Research

We have discussed conversion between any two number systems. Unlike the classical conversion from decimal to base $b$ system, we have presented a functional conversion that expresses each digit in base $b$ in terms of the decimal number in question. This in turn, provides us with a direct access to each base $b$ digit instead of the need of knowing any previous base $b$ digits.

A general version of number systems suggests that $b$ can be any nonzero number (real or imaginary) and that the $d_j$'s can be chosen from any specific set of numbers so that any number is uniquely expressible in this base. In such case, very interesting results and properties can arise. For more information and historical prospective on such general case see [2, pp. 195 – 213]. Thus, it would be of interest to generalize the result of this paper to the case when $b \in \mathbb{C}^*$ and $d_j$'s are chosen from any appropriately defined set.

Finally, we note that the result of this paper may have a strong impact on many areas of research. For example, consider the problem of converting a very large decimal number to binary. Then, we can use the method suggested in this paper together with parallel computing to optimize with respect to time and space. In

other words, we can compute each bit or set of bits independently of the others, then the final result is constructed from the results of the subproblems. The latter is a nice example of a divide-and-conquer algorithm.

## References


[1] H. Kettani (2004), "A Functional Conversion from Decimal to Base b Numbers." *Proceedings of the 3rd Hawaii International Conference on Statistics, Mathematics and Related Fields,* Honolulu, Hawaii, June, 2004.

[2] D. E. Knuth (1998), "*The Art of Computer Programming, Volume 2: Seminumerical Algorithms,*" third edition, Addison-Wesley.

[3] M. M. Mano and C. R. Kime (2001), "*Logic and Computer Design Fundamentals,*" second edition, Prentice Hall.