Simple Conversion Algorithms by H. Ramil Alvarez.

H. RAMIL ALVAREZ

Robson Cassiano - Translator

SIMPLE CONVERSION ALGORITHMS

$p \to p-1$ and $p \to p+1$

Conversions of numbers from a p-ary number system to a (p+1)-ary system (conversion $p \to p+1$) and to a (p-1)-ary system (conversion $p \to p-1$) are of interest in connection with the use of the ternary system in digital computers [1] and for transmitting information in numerical form, for which the use of ternary code is more economical than binary [2].

The conversions that arise in this case are either already conversions of the type under consideration (e.g., conversions 2-3 and 3-2), or can be reduced to these conversions (e.g., conversions 10-3 and 3-10 can be reduced to conversions 10-9 and 9-10).

We will consider positional number systems with base numbers from 0 to $p-1$, where p is the base of the system.

The representation of a number N in the p-ary system is $(a_n...a_0.d_1...d_m)p$,

where $0 \leq a_k \leq p-1$ and $0 \leq d_i \leq p-1$. Here, $(a_n...a_0)p$ is the integer part of the number, and $(.d1...d_m)p$ is the fractional part.

For integers A and m ($m > 1$), let's introduce: $\{A\}m$ - the least non-negative residue modulo m; $[A]m$ is determined from the following relation: $A = [A]m \cdot m + \{A\}m$.

The rules for converting integers are based either on calculating the quotient and remainder from dividing the integer by the new base in the old number system, or on calculating the product of the integer and the old base in the new system.

The conversion of fractional numbers is based on calculating the integer and fractional parts of the product of the fraction and the new base in the old system.

Therefore, let's consider the algorithms for division and multiplication of integers by $p-1$ and $p+1$ in the p-ary system.

---

1. Algorithms for division by $p-1$ and $p+1$

Let's consider the rules for obtaining the quotient and remainder from dividing the integer A, having the representation $(a_n...a_0)p$, by $p-1$ and $p+1$.

1.1. Algorithm for division by $p-1$.

We will show that if $a_n < p-1$, then $[(a_n...a_0)p]_{p-1} = (b_{n-1}...b_0)_p$ $\{(a_n...a_0)_p\}_{p-1} = c_0$

where $b_i (i = n-1...0)$ and c_0 are determined by the following recurrence formulas: $b_i = c_{i+1} + [a_i + c_{i+1}]_{p-1}$ $c_i = \{a_i + c_{i+1}\}_{p-1}$ (1) $c_n = a_n$

(Page 2)

For c_i, defined by formulas (1), the inequality $c_i < p-1$ holds; hence $[a_i + c_{i+1}]_{p-1} \leq 1$ and $b_i < p$.

(Page 2)

For c_i, defined by formulas (1), the inequality $c_i < p-1$ holds; hence $[a_i + c_{i+1}]_{p-1} \leq 1$ and $b_i < p$.

Consequently, $(b_{n-1}...b_0)$ is the representation of some number in the number system with base p.

We will prove the correctness of formulas (1) by induction on n. For $n=1$ we have $a_1p + a_0 = a_1(p-1) + (a_0 + a_1) = = a_1(p-1) + (p-1)[a_0 + a_1]_{p-1} + \{a_0 + a_1\}_{p-1} = = (p-1)(a_1 + [a_0 + a_1]_{p-1}) + \{a_0 + a_1\}_{p-1}$

From this, $[(a_1a_0)]_{p-1} = a_1 + [a_0 + a_1]_{p-1}$ and $\{(a_1a_0)_p\}_{p-1} = \{a_0 + a_1\}_{p-1}$, i.e., for $n=1$ formulas (1) are correct.

Assume formulas (1) are correct for $n < k$, we will show that they are correct for $n = k$: $\sum_{i=k}^{0} a_i p^i = a_k p^k + a_{k-1}p^{k-1} + \sum_{i=k-2}^{0} a_i p^i = p^{k-1}(a_k p + a_{k-1}) + + \sum_{i=k-2}^{0} a_i p^i = (p-1)(a_k + [a_{k-1} + a_k]_{p-1})p^{k-1} + \{a_{k-1} + a_k\}_{p-1}p^{k-1} + \sum_{i=k-2}^{0} a_i p^i$.

Since formulas (1) are correct for $n < k$, then for $\{a_{k-1} + a_k\}_{p-1}p^{k-1} + \sum_{i=k-2}^{0} a_i p^i$ we have $\{a_{k-1} + a_k\}_{p-1}p^{k-1} + \sum_{i=k-2}^{0} a_i p^i = (p-1)\sum_{i=k-2}^{0} b_i p^i + c_0$,

where $b_i (i = k-2...0)$ and c_0 are determined by formulas (1) and $c_{k-1} = \{a_{k-1} + a_k\}_{p-1}$. Thus, we have $\sum_{i=k}^{0} a_i p^i = (p-1)(a_k + [a_{k-1} + a_k]_{p-1})p^{k-1} + + (p-1)\sum_{i=k-2}^{0} b_i p^i + c_0 = (p-1)((a_k + [a_{k-1} + a_k]_{p-1})p^{k-1} + \sum_{i=k-2}^{0} b_i p^i) + c_0$.

Denoting $c_k = a_k$ and $b_{k-1} = c_k + [a_{k-1} + c_k]_{p-1}$ we get $c_{k-1} = \{a_{k-1} + c_k\}_{p-1}$ and $\sum_{i=k}^{0} a_i p^i = (p-1)\sum_{i=k-1}^{0} b_i p^i + c_0, \text{ where } b_i (i = n-1...0) \text{ and } c_0 \text{ are determined by formulas } (1)$, i.e., formulas (1) are also correct for $n = k$.

For the case $a_n = p-1$, one can consider the representation $(0a_n...a_0)_p$, and formulas (1) are already applicable to it. Therefore, in the general case, we have $[(a_n...a_0)_p]_{p-1} = (b_n...b_0)_p$, $\{(a_n...a_0)_p\}_{p-1} = c_0$ where $b_i (i = n...0)$ and c_0 are determined by the recurrence formulas $b_i = c_{i+1} + [a_i + c_{i+1}]_{p-1}$, $c_i = \{a_i + c_{i+1}\}_{p-1}$, $c_{n+1} = 0$. (2)

(Page 3)

Note that $0 \leq a_i + c_{i+1} \leq 2p-3$. Therefore $0 \leq [a_i + c_{i+1}]_{p-1} \leq 1$ and we have $[a_i + c_{i+1}]_{p-1} = \begin{cases} 0, & \text{if } a_i + c_{i+1} < p-1, \\ 1, & \text{if } a_i + c_{i+1} \geq p-1; \end{cases}$ $\{a_i + c_{i+1}\}_{p-1} = \begin{cases} a_i + c_{i+1}, & \text{if } a_i + c_{i+1} < p-1, \\ a_i + c_{i+1} - p + 1, & \text{if } a_i + c_{i+1} \geq p-1. \end{cases}$ Let's consider the use of the division by $p-1$ algorithm for converting the number 3973 from the decimal system to the nonary (base-9) system.

(Table showing the calculation for 3973)

Thus, $(3973)_{10} = (5404)_9$.

1.2. Algorithm for division by $p+1$.

Note that the number $p+1$ in the p-ary number system has the form $(11)_p$, so $k(p+1)$, where k is a base number, has the form $(kk)_p$.

For a two-digit number $(a_1a_0)_p$ we have $a_1p + a_0 = a_1(p+1) + (a_0 - a_1) = = a_1(p+1) + (p+1)[a_0 - a_1]_{p+1} + \{a_0 - a_1\}_{p+1} = = (p+1)(a_1 + [a_0 - a_1]_{p+1}) + \{a_0 - a_1\}_{p+1}$.

Note that $-p < a_0 - a_1 < p$. Therefore $-1 \leq [a_0 - a_1]_{p+1} \leq 0$; and $[a_0 - a_1]_{p+1} = \begin{cases} -1, & \text{if } a_0 < a_1, \\ 0, & \text{if } a_0 \geq a_1. \end{cases}$ In this case, $[a_0 - a_1]_{p+1} = -1$ if $a_1 > 0$, i.e., $0 \leq a_1 + [a_0 - a_1]_{p+1} < p$. From this we have $[(a_1a_0)_p]_{p+1} = a_1 + [a_0 - a_1]_{p+1}$, $\{(a_1a_0)_p\}_{p+1} = \{a_0 - a_1\}_{p+1}$

(Page 4)

Using induction on n, it can be shown that $[(a_n...a_0)_p]_{p+1} = (b_{n-1}...b_0)_p$ $\{(a_n...a_0)_p\}_{p+1} = c_0$, where $b_i (i = n-1...0)$ and c_0 are determined by the following recurrence formulas: $b_i = c_{i+1} + [a_i - c_{i+1}]_{p+1}$, $c_i = \{a_i -$

(Page 4)

Using induction on n, it can be shown that $[(a_n...a_0)_p]_{p+1} = (b_{n-1}...b_0)_p$ $\{(a_n...a_0)_p\}_{p+1} = c_0$, where $b_i (i = n-1...0)$ and c_0 are determined by the following recurrence formulas: $b_i = c_{i+1} + [a_i - c_{i+1}]_{p+1}$, $c_i = \{a_i - c_{i+1}\}_{p+1}$ $c_n = a_n$ (3)

Note that $-(p+1) < a_i - c_{i+1} < p$. Therefore $-1 \le [a_i - c_{i+1}]_{p+1} \le 0$ and we have $[a_i - c_{i+1}]_{p+1} = \begin{cases} 0, & \text{if } a_i \ge c_{i+1}, \\ -1, & \text{if } a_i < c_{i+1}; \end{cases}$ $\{a_i - c_{i+1}\}_{p+1} = \begin{cases} a_i - c_{i+1}, & \text{if } a_i \ge c_{i+1}, \\ p + 1 + a_i - c_{i+1}, & \text{if } a_i < c_{i+1}; \end{cases}$ Let's consider the use of the division by $p+1$ algorithm for converting the number 3276 from the octal (base-8) system to the nonary (base-9) system.

(Table showing the calculation for 3276)

Thus, we have $(3276)_8 = (2327)_9$.

1.3. General form of algorithms for division by $p-1$ and $p+1$.

The division algorithms for $p-1$ and $p+1$ can be written in a general form as follows: $[(a_n...a_0)_p]_{p\pm1} = (b_n...b_0)_p$, $\{(a_n...a_0)_p\}_{p\pm1} = c_0$, where $b_i (i = n...0)$ and c_0 are determined by the recurrence formulas $b_i = c_{i+1} + [a_i \pm c_{i+1}]_{p\pm1}$, $c_i = \{a_i \pm c_{i+1}\}_{p\pm1}$, $c_{n+1} = 0$.

1.4. Other applications of the division algorithms for $p-1$ and $p+1$.

One of the methods for constructing error-detecting codes is modular checking, in which each number is considered along with its residue modulo some base r. Work [3] provides values of r for decimal and binary systems and rules for obtaining residues. The application of the division algorithms for $p-1$ and $p+1$ allows in some cases to simplify the rules for obtaining residues, as well as to expand in the case of the binary system the range of r values ($r = 9$ and 17).

(Page 5)

Let's consider, for example, the rule given in [3] for obtaining the residue modulo 11 for the decimal system. "The remainder from dividing any decimal number by 11 is found by breaking this number into pairs of digits, calculating the remainders from dividing the resulting two-digit numbers by 11, and then summing these remainders modulo 11." Note that it remains unclear how to obtain the residue from a two-digit number.

The rule for obtaining the residue modulo 11, based on the algorithm for division by $p+1$, given that in this case we are not interested in the quotient of the division, can be formulated as follows: $c_i = \{a_i - c_{i+1}\}_{11}$ $(i = n-1...0)$, where $c_n = a_n$ and c_0 is the desired residue.

For comparison, here is an example of calculating the residue modulo 11 for the number 7,839,756,840 using both rules. $\{7\ 839\ 756\ 840\}_{11} = \{\{78\}_{11} + \{39\}_{11} + \{75\}_{11} + \{68\}_{11} + +\{40\}_{11}\}_{11} = \{1+6+9+2+7\}_{11} = \{25\}_{11} = 3$,

(Using the new algorithm steps) 7 8 3 9 7 5 6 8 4 0 c_i: 7 1 2 7 0 5 1 7 8 3 i.e., $\{7\ 839\ 756\ 840\}_{11} = c_0 = 3$.

---

2. Algorithms for multiplication by $p-1$ and $p+1$

Let's consider the rules for obtaining the product of an integer A, having the representation $(a_n...a_0)_p$, by $p-1$ and $p+1$.

2.1. Algorithm for multiplication by $p-1$.

Using induction on n, it can be shown that $(p-1)(a_n...a_0)_p = (b_{n+1}...b_0)_p$ where $b_i (i = 0...n+1)$ are determined by the following recurrence formulas: $b_i = \{c_i - a_i\}_p$ $c_{i+1} = a_i + [c_i - a_i]_p$, $c_0 = a_{n+1} = 0$ (4)

When using this algorithm for converting integers $p \to p+1$, c_0 is set to the next p-ary digit of the number at

Using induction on n, it can be shown that $(p-1)(a_n...a_0)_p = (b_{n+1}...b_0)_p$ where $b_i(i=0...n+1)$ are determined by the following recurrence formulas: $b_i = \{c_i - a_i\}_p$ $c_{i+1} = a_i + [c_i - a_i]_p$, $c_0 = a_{n+1} = 0$ (4)

When using this algorithm for converting integers $p \to p+1$, c_0 is set to the next p-ary digit of the number at each step. Let's show the application of the algorithm when converting the number 3276 from the octal system to the nonary system (this example was considered earlier in section 1.2).

(Page 6)

3276 *(Table showing the calculation for 3276)* ...

Thus, $(3276)_8 = (2327)_9$. Applying the algorithm for multiplying fractional numbers by $p-1$, i.e., when converting $p \to p-1$, we have $(p-1)(\cdot a_1...a_n) = (c_1)_{p-1} + (\cdot b_1...b_n)_p$ where $b_i(i=n...1)$ and c_1 are determined by the recurrence formulas $b_i = \{c_{i+1} - a_i\}_p$ $c_i = a_i + [c_{i+1} - a_i]_p$, $c_{n+1} = 0$.

Here is an example of using this algorithm to convert the number .87514 from the nonary system to the octal system. *(Table showing the calculation for .87514)* ...

Thus, $(\cdot 87514)_9 = (\cdot 7667...)_8$.

2.2. Algorithm for multiplication by $p+1$.

Using induction on n, it can be shown that $(p+1)(a_n...a_0)_p = (b_{n+2}...b_0)_p$, where $b_i(i=0...n+2)$ are determined by the recurrence formulas $b_i = \{c_i + a_i\}_p$ $c_{i+1} = a_i + [c_i + a_i]_p$, $c_0 = a_{n+1} = a_{n+2} = 0$. (5)

When using the algorithm for converting integers $p \to p-1$, c_0 at each step is set to the next p-ary digit of the number. Let's show the application of this algorithm for converting the number 3973 from the decimal system to the nonary system.

(Page 7)

03973 *(Table showing the calculation for 3973)* ...

Thus, $(3973)_{10} = (5404)_9$.

Applying this multiplication algorithm to calculate the product of a fractional number by $p+1$, i.e., when converting $p \to p+1$, we have $(p+1)(\cdot a_1...a_n)_p = (c_1)_{p+1} + (\cdot b_1...b_n)p$, where $b_i(i=n...1)$ and c_1 are determined by the recurrence formulas $b_i = \{c_{i+1} + a_i\}_p$, $c_i = a_i + [c_{i+1} + a_i]_p$, $c_{n+1} = 0$.

Here is an example of using this algorithm to convert the number .1101011 from the binary system to the ternary system.

(Table showing the calculation for .1101011) ...

Thus, $(.1101011)_2 = (\cdot 2111...)_3$.

2.3. General form of algorithms for multiplication by $p-1$ and $p+1$.

The algorithms for multiplying integers by $p-1$ and $p+1$ can be written in general form as follows: $(p \pm 1)(a_n...a_o)_p = (b_{n+2}...b_o)p$, where $b_i(i=0...n+2)$ are determined by the recurrence formulas $b_i = \{c_i \pm a_i\}_p$, $c_{i+1} = a_i + [c_i \pm a_i]p$, $c_0 = a_{n+1} = a_{n+2} = 0$.

(Page 8)

3. Conclusion

The considered algorithms for division and multiplication of integers by $p-1$ and $p+1$ in the p-ary number system reduce to digit-wise operations (addition, subtraction, and residue calculation). The use of these algorithms allows for the formulation of simple rules for conversions $p \to p-1$ and $p \to p+1$. The simplicity of these rules suggests that their hardware implementation will not be associated with the cost of significant amounts of equipment or time.

system reduce to digit-wise operations (addition, subtraction, and residue calculation). The use of these algorithms allows for the formulation of simple rules for conversions $p \to p - 1$ and $p \to p + 1$. The simplicity of these rules suggests that their hardware implementation will not be associated with the cost of significant amounts of equipment or time.

Obviously, these algorithms do not solve the general problem of conversion from one number system to another. However, their area of applicability includes the following pairs of numbers: (9, 10), (2, 3), (3, 4), (8, 9), i.e., with the help of these algorithms, the issues of conversion from the ternary number system to the decimal system and to the binary system (most common in computing today), as well as the reverse conversions (10-3 and 2-3), can be resolved.

REFERENCES

1. Brusentsov N.P. On the use of ternary code and three-valued logic in digital machines. In coll.: "Computer technology and questions of cybernetics", issue 7. Moscow State University Publishing House, 1970.
2. Brins T. Data transmission is faster with ternary coding. "Electronics", 47, No. ll, 119-120, 1974.
3. Kartsev M.A. Arithmetic of digital machines. M, "Nauka", 1969.