**УДК 681.3:007**

# МОДЕЛИРОВАНИЕ РАЗРЯДНОГО РАСПАРАЛЛЕЛИВАНИЯ СРАВНЕНИЙ СТРОКОВЫХ И ЧИСЛОВЫХ ЭЛЕМЕНТОВ

**Ромм Я.Е., Белоконова С.С.**

*Таганрогский институт имени А.П. Чехова (филиал) РГЭУ (РИНХ), Таганрог, Россия (347936, Ростовская область, г. Таганрог, ул. Инициативная, 48), e-mail: romm@list.ru*

**В статье представлен метод поразрядно-параллельного алгебраического сложения $n$-разрядных чисел, который не включает операций вычисления переноса, и способ поразрядно-параллельного сравнения чисел и элементов строкового типа на его основе. Отмечены особенности применения метода для выполнения операций сравнения с временной сложностью $O(1)$ независимо от числа символов сравниваемых строк, это выявляет потенциальную применимость метода для ускорения информационного поиска. Основной и конкретной целью излагаемой работы является программный эксперимент по моделированию поразрядно-параллельных сравнений элементов строкового типа для верификации разрабатываемого метода и алгоритмов на его основе. Приводится код программы на языке Delphi, посредством которой выполняется моделирование сравнений с разрядным распараллеливанием, описываются результаты эксперимента, подтверждающие достоверность метода и правильность работы предложенных алгоритмов.**

Ключевые слова: разрядное распараллеливание алгебраического сложения $n$-разрядных чисел, поразрядно-параллельное сравнение слов, моделирование сравнений с разрядным распараллеливанием.

# THE BIT MODELING PARALLELIZATION OF STRING AND NUMERIC COMPARISONS OF ELEMENTS

**Romm Y.E., Belokonova S.S.**

*Taganrog Institute of Chekhov A.P. (branch) RGEU (RINH), Taganrog, Russia, (347936, Rostovskaya obl., Taganrog, Iniciativnaya str., 48), e-mail: romm@list.ru*

**The paper presents a method of bit by bit-parallel algebraic addition-bit integers, which does not include the operations of calculating transfer, and bit by bit-parallel way to compare numbers and string elements based on it. The features of the method to perform the comparison with the time complexity $O(1)$ regardless of the number of characters strings being compared, it identifies the potential applicability of the method to speed information retrieval. The main purpose of the stated and the specific work program is an experiment on the modeling-wise manner parallel comparisons string elements for the verification of methods and algorithms based on it. The code of the program in the language of Delphi, which is performed through comparisons with simulation parallelization bit, describes the experimental results confirming the accuracy of the method and the correct operation of the proposed algorithms.**

Keywords: bit parallelization algebraic addition-bit numbers, bit by bit, parallel comparison of words, modeling comparisons with bit parallelization.

**Постановка вопроса.** Тема ускорения информационного поиска актуальна практически для всех существующих информационных систем [9, 10]. В качестве алгоритмической основы разработки быстродействующих систем поиска можно рассматривать разрядное распараллеливание операций сравнения слов и элементов строкового типа. С этой целью ниже излагается метод поразрядно-параллельного алгебраического сложения чисел, который не включает операций вычисления переноса, далее, излагается способ его переноса на случай поразрядно-параллельного сравнения чисел и данных строкового типа. Целью переноса является разработка основы для максимально параллельного выполнения базовых операций поиска, включая сравнение, сортировку, и

распространение метода одновременно на поиск чисел и строк. Конкретной задачей излагаемого сообщения является программный эксперимент по моделированию предложенного поразрядно-параллельного сравнения элементов строкового типа для верификации разрабатываемого метода и алгоритмов на его основе. В работе приводится программа, по которой выполняется моделирование, и описываются результаты эксперимента.

**Метод поразрядно-параллельного алгебраического сложения** заимствуется из [1, 2]. Пусть два двоичных полинома

$$P_1 = \sum_{i=0}^{n} \beta_i 2^i\ ,\ \beta_i = \begin{cases} 0 \\ 1 \end{cases},\quad P_2 = \sum_{i=0}^{n} \gamma_i 2^i\ ,\ \gamma_i = \begin{cases} 0 \\ 1 \end{cases}, \tag{1}$$

условно расположены в двух входных разрядных сетках (РС), $\text{РС}_{\text{вх}}^{(0)}$ и $\text{РС}_{\text{вх}}^{(1)}$, номера клеток которых совпадают с индексами коэффициентов из (1).

| | | | | | |
|---|---|---|---|---|---|
| $\beta_n$ | $\beta_{n-1}$ | ... | $\beta_1$ | $\beta_0$ | $\text{РС}_{\text{вх}}^{(0)}$ |
| $\gamma_n$ | $\gamma_{n-1}$ | ... | $\gamma_1$ | $\gamma_0$ | $\text{РС}_{\text{вх}}^{(1)}$ . |

(2)

Синхронно и взаимно независимо по всем номерам разрядов $j$ из (2) выполняется операция $\beta_j + \gamma_j$ суммирования двоичных коэффициентов равного веса по вертикали ($\text{СВ}_j$). Результат $\text{СВ}_j$ представляется в двоичном коде

$$\beta_j + \gamma_j = \sum_{\ell=0}^{1} \Delta_{\ell j} 2^{\ell}\ ,\ \Delta_{\ell j} = \begin{cases} 0 \\ 1 \end{cases},\ j=0,1,...,n\,. \tag{3}$$

Коэффициенты из (3) располагаются по диагонали двух выходных РС, – $\text{РС}_{\text{вых}}^{(0)}$ и $\text{РС}_{\text{вых}}^{(1)}$, образуя диагональную от $j$-го разряда запись ($\text{Д}_j$-запись) вида

| | $j+1$ | $j$ | $j-1$ | |
|---|---|---|---|---|
| $\text{СВ}_j$: + | | $\beta_j$ | | $\text{РС}_{\text{вх}}^{(0)}$ |
| | | $\gamma_j$ | | $\text{РС}_{\text{вх}}^{(1)}$ |
| $\text{Д}_j$ - запись: | | $\Delta_{0j}$ | | $\text{РС}_{\text{вых}}^{(0)}$ |
| | $\Delta_{1j}$ | | | $\text{РС}_{\text{вых}}^{(1)}$ . |

(4)

Предварительный результат сложения:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $\Delta_{n+1}$ | $\Delta_n$ | $\Delta_{n-1}$ | ... | $\Delta_j$ | ... | $\Delta_1$ | $\Delta_0$ | $\text{РС}_{\text{вых}}^{(0)}$ |
| $\nabla_{n+1}$ | $\nabla_n$ | $\nabla_{n-1}$ | ... | $\nabla_j$ | ... | $\nabla_1$ | $\nabla_0$ | $\text{РС}_{\text{вых}}^{(1)}$ , |

(5)

где $\Delta_j = \Delta_{0j}$, $\nabla_j = \Delta_{1(j-1)}$ из (4), $j = 0, 1, ..., n, n+1$.

Все переносы в (5) оказываются взаимно отделенными вертикальными парами нулей:

$$\begin{pmatrix} \Delta_k \\ \nabla_k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},\ k = \text{const}\,,$$

где $\Delta_k$, $\nabla_k$ из (5), $0 \le k \le n+1$, поскольку [2] при любом $j$ из (4) складываются не более двух единиц одинакового веса. Аналогично, в (4) по диагонали $\text{Д}_j$ – записи от 1 в $\text{РС}^{(0)}_{\text{вых}}$ не может располагаться 1 в $\text{РС}^{(1)}_{\text{вых}}$. На этой основе корректным является следующее преобразование промежуточного набора (5). Все разряды, имеющие значение $\Delta_j \ne 0$, из $\text{РС}^{(0)}_{\text{вых}}$ (и только из $\text{РС}^{(0)}_{\text{вых}}$) в (5) подвергаются тождественному преобразованию:

$$\Delta_j \equiv 2\Delta_j - \Delta_j, \quad j = 0, 1, ..., n. \tag{6}$$

При этом правая часть (6) размещается в $j$-й клетке $\text{РС}^{(0)}_{\text{вых}}$ и в $(j+1)$-й клетке $\text{РС}^{(1)}_{\text{вых}}$ в порядке $\text{Д}_j$ – записи для всех номеров $j$ преобразуемых разрядов:

$$\Delta_j \longrightarrow \begin{array}{|c|c|} \hline & j \\ & \Delta_j \\ \hline 0 & \\ \hline \multicolumn{2}{c}{j+1} \end{array} \longrightarrow 2\Delta_j - \Delta_j \longrightarrow \begin{array}{|c|c|} \hline & j \\ & -\Delta_j \\ \hline \Delta_j & \\ \hline \multicolumn{2}{c}{j+1} \end{array} \begin{array}{l} \text{РС}^{(0)}_{\text{вых}} \\ \text{РС}^{(1)}_{\text{вых}} \end{array} \tag{7}$$

Преобразование (7) корректно, поскольку в $(j+1)$-й клетке $\text{РС}^{(1)}_{\text{вых}}$ всегда находится ноль – до применения (6) – при условии, что в $\text{РС}^{(0)}_{\text{вых}}$ $\Delta_j \ne 0$, иначе $\text{Д}_j$ – запись означала бы результат сложения трех единиц веса $j$-го разряда, тогда как их число не больше двух [1, 2].

Вслед за тем параллельно по всем $j = 0, 1, ..., n, n+1$ выполняется шаг вертикального сложения, результатом которого в каждом разряде всегда будет либо 0, либо $+1$, либо $-1$ [1, 2]. При этом значение знакоразрядного кода расположится в $\text{РС}^{(0)}_{\text{вых}}$, где, таким образом, сформируется окончательное значение вычисляемой суммы $P_1$ и $P_2$ из (1).

Временная сложность $T$ параллельного сложения в знакоразрядном коде составит $\approx 2\tau$ при количестве элементов $S\ell$ сумматора пропорциональном числу разрядов слагаемых:

$$T \approx 2\tau = O(1), \quad S\ell \approx 4(n+1) = O(n).$$

**Сравнение двоичных чисел.** Предполагается, что сравниваемые числа имеют формат (1) с нумерацией разрядов справа налево. Пусть число $A$ принимается за уменьшаемое, $B$ – за вычитаемое, тогда $B$ записывается в обратном коде (единицы заменяются нулями, нули – единицами), затем используется тождественное преобразование [6]:

$$A - B = A + \left(\left(2^{n+1} - 1\right) - B\right) - \left(2^{n+1} - 1\right), \tag{8}$$

с учетом

$$2^{n+1} - 1 = 2^n + 2^{n-1} + 2^{n-2} + \ldots + 2^1 + 2^0, \tag{9}$$

или, в позиционной системе,

$$2^{n+1} - 1 = \underbrace{1\ 1\ 1\ \ldots\ 1\ 1}_{n+1}. \tag{10}$$

Над числом $A$ и числом $B$, взятым в обратном коде, параллельно по всем номерам разрядов выполняется операция $\mathrm{СВ}_j$, затем $A$ и $B$ (в обратном коде) суммируются по схеме (2) – (7). Для восстановления правильного результата вычитания учитывается (8) – (10): от однорядного знакоразрядного двоичного кода полученного результата следует вычесть $2^{n+1}-1$. Это равносильно тому, что с полученным результатом складывается $-2^{n+1}+1$ – к младшему разряду добавляется $+1$, а к разряду веса $2^{n+1}$ добавляется $-1$.

Знак сравнения $A$ и $B$ определяется знаком старшего ненулевого разряда однорядного знакоразрядного двоичного числа, полученного в скорректированном окончательном результате. Если знак этого разряда отрицательный, то $A < B$, если положительный, то $A > B$, если все разряды нулевые, то $A = B$.

**Разрядное распараллеливание сравнения строк.** Метод переносится [4, 5] на сравнение элементов строкового типа, для краткости, – строк. Все символы представляются в ASCII-коде в двоичной форме. Набор двоичных кодов всех символов строки в порядке расположения интерпретируется как единое числовое значение. Сравнение полученных двоичных кодов выполняется путем алгебраического сложения, изложенным поразрядно-параллельным способом, однако при этом, в соответствии с лексикографическим упорядочением слов, выравнивание весов разрядов выполняется по старшим разрядам, недостающие младшие разряды дополняются нулями. Например, сравниваются слова 'poisk' и 'primer'. Их двоичный код с выравниваем старших разрядов соответственно примет вид:

```
0 1 1 1 0 0 0 0 0 1 1 0 1 1 1 1 0 1 1 0 1 0 0 1 0 1 1 1 0 0 1 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 1 1 1 0 0 1 0 0 1 1 0 1 0 0 1 0 1 1 0 1 1 0 1 0 1 1 0 0 1 0 1 0 1 1 1 0 0 1 0
```

Слово «poisk» интерпретируется как уменьшаемое, «primer» – как вычитаемое. После перевода вычитаемого в обратный код алгебраическому сложению подлежат:

```
0 1 1 1 0 0 0 0 0 1 1 0 1 1 1 1 0 1 1 0 1 0 0 1 0 1 1 1 0 0 1 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 0
1 0 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 1 1 0 1 0 0 1 0 0 1 0 1 0 0 1 1 0 1 0 1 0 0 0 1 1 0 1
```

Параллельно по всем номерам разрядов выполняется операция $\mathrm{СВ}_j$:

```
  0 1 1 1 0 0 0 0 0 1 1 0 1 1 1 1 0 1 1 0 1 0 0 1 0 1 1 1 0 0 1 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 0
+ 1 0 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 1 1 0 1 0 0 1 0 0 1 0 1 0 0 1 1 0 1 0 1 0 0 0 1 1 0 1
-------------------------------------------------------------------------------------------------
  0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 0 1
  0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0
```

Согласно (5) – (7) выполняется преобразование:

```
 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0 0 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0 0 -1 -1 -1 -1 -1 0 0 -1 -1 0 0 0 -1 -1 0 -1
+1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 0 0 0 1 1 0 1 0
```

Параллельное по всем разрядам суммирование по вертикали влечет:

```
1 0 0 0 0 0 0 0 0 0 0 -1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 -1 0 0 1 0 -1 1 -1
```

Результат корректируется в соответствии (8) – (11), что влечет окончательное значение алгебраической суммы

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

Старший ненулевой разряд отрицателен, следовательно, первое слово лексикографически меньше второго: 'poisk' < 'primer'.

**Замечание 1.** Поразрядно-параллельное нахождение старшего ненулевого разряда в знакоразрядном коде – самостоятельная схемотехническая задача. Такой разряд требуется выделить программно или аппаратно, что возможно, например, при помощи схемы, представленной в [7]. Можно, с другой стороны, выделить старший ненулевой разряд аналогично нормализации мантиссы числа с плавающей точкой с помощью параллельного сдвигателя, описанного в [8]. Ниже решение этой задачи не обсуждается.

С учетом данного замечания изложенное бинарное сравнение независимо от числа разрядов занимает время одной бинарной операции над двоичными коэффициентами одного веса –

$$T_{\text{сравнения строк}(n)} = \tau_{\text{бинарное}} = O(1),$$

независимо от $n$, где $n$ – число разрядов двоичного представления строки.

**Программный эксперимент.** Целью работы является численное моделирование изложенного метода и программный эксперимент для проверки его универсальности и достоверности. Ниже представлена консольная программа Delphi, затем описаны результаты эксперимента.

```
program Proverka;
{$APPTYPE CONSOLE}
uses
SysUtils;
  CONST nn=100000;
S1='poisk'; S2='primer'  ;
type mass=array[0..nn{*8}] of Integer;
var
a,b,a1,b1,a4,b4, a2,b2,a3,b3:mass;
s3:string;
cod,ii,y,n,m,i,k:integer;
function IntToBin(x:LongInt):string;
varres:string;
begin
RES:='';
while x<>0 doBegin
if (x mod 2=0) then res:='0'+res else res:='1'+res;
x:=x div 2;  end;
while (Length(Res)<>8) do res:='0'+res;
IntToBin:=Res;
end;
procedure Assc(c:string;nm:Integer; var x:mass);
beginfor i:=1 to nm do x[i]:=Ord(c[i]);end;

procedure Assctobin(x: mass;nm:Integer; var x1:mass);
beginy:=1;
for i:=1 to n do
```

```
begin
   s3:=inttobin(x[i]);
for ii:=1 to 8 dobegin
val(s3[ii],x1[y],cod);y:=y+1;end;
end;
end;
procedure srav(varx,y:mass;nm:Integer; varxr,yr:mass);
var i:integer;
begin
for i:=1 to nm*8 do
if y[i]=1 then y[i]:=0 else y[i]:=1; writeln; writeln;
for i:=1 to nm*8 do
if (x[i]=1) and (y[i]=1) then
begin a2[i]:=0;b2[i-1]:=1; end
else a2[i]:=a1[i]+b1[i];
Writeln ('massiv a2 i b2');
for i:=1 to n*8 do write(a2[i]:5);writeln;writeln;
for i:=1 to n*8 do write(b2[i]:5);writeln; writeln;
for i:=0 to nm*8 do
if (a2[i]=1) then a3[i]:=-1 else a3[i]:=a3[i];
for i:=0 to nm*8 dobegin b3[i]:=b2[i];
if (a3[i]=-1) then b3[i-1]:=1; end;
Writeln ('massiv a3 i b3');
for i:=1 to n*8 do write(a3[i]:5); writeln; writeln;
for i:=1 to n*8 do write(b3[i]:5);writeln; writeln;
for i:=0 to n*8 do
if (a3[i]=1) and (b3[i]=1) then
beginxr[i]:=0;yr[i-1]:=1;end
elsexr[i]:=a3[i]+b3[i];
xr[0]:= xr[0]-1;xr[nm*8]:=xr[nm*8]+1;end;
begin
n:=length(s1); m:=length(s2);
Assc(s1,n,a); Assc(s2,m,b);
writeln ('assci-kodeslova 1');
for i:=1 to n do  write(a[i]:5);writeln;writeln;
writeln ('massiv b ');
for i:=1 to m do  write(b[i]:5); Writeln; Writeln;
Assctobin(a,n,a1);Assctobin(b,m,b1);
if m>n then n:=m;
Writeln ('massiv a1 i b1');
for i:=1 to n*8 do write(a1[i]:5);writeln;writeln;
for i:=1 to n*8 do write(b1[i]:5); writeln; writeln;
srav(a1,b1,n,a4,b4);
Writeln ('massiv a4 i b4');
for i:=0 to n*8 dowrite(a4[i]:5); writeln; writeln;
for i:=0 to n*8 do write(b4[i]:5); writeln; writeln;
i:=0;
while (a4[i]=0)and (i<>n*8) do i:=i+1;
if a4[i]<0 then writeln ('1<2') else
if a4[i]>0 then writeln ('1>2')else writeln ('1=2');
Readln;
Readln;
{ TODO -oUser -cConsole Main : Insert code here }
end.
```

1. На вход программы были поданы 2 предложения S1 и S2.

*S1='Уже было начало июня, когда князь Андрей, возвращаясь домой, въехал опять в ту березовую рощу, в которой этот старый, корявый дуб так странно и памятно поразил его.'*

*S2='Уже было начало июня, когда, возвращаясь домой, мы въехали в березовую рощу.'*

В результате работы программы был получен верный результат сравнения: *S1< S2*.

2. Если в этих же исследуемых строках удалить все знаки препинания, то получится другой, но также верный результат: *S1> S2*.

3. В следующем примере в сравнении рассматривается только фрагменты строк, сформированные следующим образом:

```
t:=pos('мой',s1);
s11:=copy(s1,t,6);
writeln(s11);
t:=pos('мой',s2);
s22:=copy(s2,t,6);
```

*S1='Уже было начало июня, когда князь Андрей, возвращаясь до***мой, в***ъехал опять в ту березовую рощу, в которой этот старый, корявый дуб так странно и памятно поразил его.';*

*S2='Уже было начало июня, когда возвращаясь до***мой, в***ъехали в березовую рощу.';*

Таким образом, на вход программы для сравнения были поданы следующие фрагменты строк:

*S11 ='мой, в'*

*S22='мой, в'*

ASCII-код *символов S11:* 236 238 233 44 32 226

ASCII-код *символов S22:*236 238 233 44 32 226

Соответственные двоичные формы ASCII-кодов:

1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0 0 1 0
1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0 0 1 0

С представлением вычитаемого в обратном коде получится:

1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0 0 1 0
0 0 0 1 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 0 1 1 1 1 1 0 0 0 1 1 1 0 1

Представление промежуточной суммы:

0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Преобразование (6), (7) влечет:

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

Результат поразрядно-параллельного выполнения $\text{СВ}_j$:

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

Коррекция суммы в соответствии с формулами (8), (9):

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Таким образом, сравниваемые элементы строк равны между собой: *S11=S22*.

4. Метод непосредственно применим для сравнения целых чисел с тем изменением, что сравниваемые числа выравниваются по младшему разряду.

Например, представленная программа выдаст сравнение строковых констант `'16'` и `'24'`: *S1*<*S2*.

5. При сравнении целых чисел разной разрядности следует предварительно дополнить меньшее число предшествующими незначимыми нулями. Так, для сравнения чисел 160 и 24 они предварительно должны быть преобразованы в строки вида `'160'` и `'024'`.

6. С выравниванием порядков эта схема переносится на сравнение вещественных чисел с плавающей точкой. В этом случае представленная программа выдаёт: `'16.05'>'16.03'`, `'16.005'<'17.003'`, `'016.050'<'170.003'`.

Во всех вариантах эксперимента данная программная модель давала правильные результаты поразрядно-параллельных сравнений чисел и слов.

В [3] приводится близкий к изложенному способ, при этом он непосредственно объединяет поиск чисел и слов. В файле при помощи сортировки ищутся абсолютные величины разностей ASCII-кодов строк или их элементов, `abs (ord()-ord())`, или непосредственно разности чисел в формате представления языка программирования, `abs (x-y)`. Нулевое значение разности, в последнем случае с точностью до приближения, означает искомое значение. С выравниванием порядков эти операции осуществимы поразрядно-параллельным способом, отсюда на изложенной основе возможно объединение операций поиска чисел с плавающей точкой и элементов строкового типа.

**Заключение.** В статье представлен метод разрядного распараллеливания операции сравнения для информационного поиска. Метод основан на бинарном алгебраическом сложении полноразрядных чисел без вычислений переноса. Указаны особенности применения метода к поразрядно-параллельному сравнению чисел и строк. Основное содержание статьи составляет программная модель выполнения операций сравнения на основе данного метода и описание результатов программного эксперимента. Эксперимент подтверждает достоверность метода и правильность работы предложенных алгоритмов.

## Список литературы

1. Ромм Я.Е. Метод вертикальной обработки потока целочисленных групповых данных. I. Приложение к бинарным арифметическим операциям // Кибернетика и системный анализ. – 1998. – № 3. – С. 123-151.

2. Ромм Я.Е. Метод вертикальной обработки потока целочисленных групповых данных. II. Приложение к бинарным арифметическим операциям // Кибернетика и системный анализ. – 1998. – № 6. – С. 146-162.

3. Ромм Я.Е., Белоконова С.С. Детерминированный информационный поиск на основе сортировки с распараллеливанием базовых операций. – М.: Научный мир, 2014. – 198 с.

4. Ромм Я.Е., Иванова А.С. Вертикальное групповое алгебраическое суммирование применительно к сортировке со слиянием и параллельному поиску / ТГПИ. – Таганрог, 2012. – 44 с. Деп. в ВИНИТИ 03.09.2012, № 362 – В 2012.

5. Ромм Я.Е., Иванова А.С. Вертикальные групповые арифметические операции над целочисленными данными без вычисления переноса. // «Фундаментальные исследования». №11 (4). – 2012. – С. 960 – 964

6. Ромм Я.Е., Чабанюк Д.А. Применение метода вертикальной обработки информации к операциям сравнения и поиска / ТГПИ – Таганрог, 2013. – 32 с. Деп. в ВИНИТИ 20.05.13, № 141 – В 2013.

7. Ромм Я.Е., Чабанюк Д.А. Сравнение слов с единичной временной сложностью // Известия ЮФУ. Технические науки. – № 7(156). – 2014. – С. 230 – 238.

8. Ромм Я.Е. Бесконфликтные и устойчивые методы детерминированной параллельной обработки / Диссертация на соискание ученой степени доктора технических наук. – Таганрог: ТРТУ. – 1998. – 546 с.; ВНТИ Центр. – № 05.990.001006.

9. Царев Р.Ю. Структуры и алгоритмы обработки данных. – Красноярск: Сиб.федер.ун-т, 2013. – 160с.

10. Чернов А.Ф. Ускорение поиска в индексах на основе R-деревьев // Программные продукты и системы, 2012. №2 (98), С. 46 - 50.

**Рецензенты:**

Боженюк А.В., д.т.н., профессор кафедры информационно аналитических систем безопасности Инженерно-технологическая академия Южного федерального университета, г. Таганрог;

Карелин В.П., д.т.н., профессор, заведующий кафедрой прикладной математики и информационных технологий Таганрогского института управления и экономики, г. Таганрог.