Фалина Ирина Николаевна
Кандидат педагогических наук, доцент кафедры информатики СУНЦ МГУ им. М.В. Ломоносова.

Компьютерная фибоначчиева арифметика

В №6 за 2011 год была опубликована статья И.Н. Фалиной «Компьютер и фибоначчиева система счисления», в которой рассказывалось о фибоначчиевой системе счисления (ФСС): её алфавите и базисе, о правилах перевода целых чисел из фибоначчиевой системы счисления в десятичную и обратно, о таком свойстве этой системы, как избыточность. Рассматривалась избыточность естественных языков и необходимость использования избыточных кодов для помехоустойчивой передачи информации.

Статья вызвала большой отклик со стороны наших читателей. В этом номере журнала вниманию читателя предлагается продолжение статьи. Вы узнаете о специфических операциях, которые можно производить над числами, представленными в ФСС. О том, что эти операции составляют функционально полную систему, т. е. с их помощью можно выразить любую арифметическую или логическую операцию.

Операции свёртки и развёртки в ФСС

Фибоначчиева система счисления является позиционной с алфавитом, состоящим из двух цифр: 0 и 1, а её базисом является последовательность чисел Фибоначчи 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 ... . (F_0 = = 1 в базис не включается, $F_k = F_{k-1} +$ + F_{k-2}). В фибоначчиевой системе, как и во всех позиционных системах счисления, «вес» каждого разряда определяется базисом этой системы. Нумерацию разрядов числа в фибоначчиевой системе принято начинать с 1; такая нумерация удобна, поскольку вес k-го разряда равен k-ому числу Фибоначчи.

Фибоначчиева система счисления обладает удивительным свойством – свойством избыточности. Оказывается, что многие числа в ФСС могут выглядеть по-разному. Например, десятичное число 40 можно записать в ФСС четырьмя разными способами:

$40_{10} = 10001001_{fib} = 10000111_{fib} =$ $=1101001_{fib} = 1100111_{fib}$.

Различные фибоначчиевы представления одного и того же числа могут быть получены друг из друга с помощью специальных фибоначчиевых операций *свёртки* (011 ⇒ ⇒ 100) и *развёртки* (100 ⇒ 011). Эти операции выполняются над фибоначчиевой записью числа и не меняют значения числа. Покажем это.

Пусть в записи числа A первая слева комбинация «11» стоит на $(p+1)$-м и p-м местах (на $(p+2)$-м месте обязательно стоит 0). Тогда значение числа A равно:

$$A = B + F_{p+1} + F_p + C,$$

где B – сумма чисел Фибоначчи, соответствующих ненулевым разрядам в записи числа до комбинации «11», C – сумма чисел Фибоначчи, соответствующих ненулевым разрядам в записи числа после комбинации «11». В соответствии с тождеством

$$F_k = F_{k-1} + F_{k-2} \qquad (1)$$

мы можем написать $A = B + F_{p+2} + C$. Это значит, что комбинация «011» заменилась комбинацией «100», но значение числа A не изменилось.

Например, $A = 10011_{fib} = {}_{8\,5\,3\,2\,1}10011 = {}_{8\,5\,3\,2\,1}10100 = 11_{10}$.

Аналогичные рассуждения применимы и к операции развёртки.

Замечание 1. Операция свёртки *уменьшает* количество единиц в фибоначчиевой записи числа, но может увеличить на единицу количество цифр в записи, если исходная запись начиналась с двух единиц. Например, $\underline{11}0101_{fib} = \underline{1000}101_{fib}$.

Действительно, пусть в записи числа A было k цифр, из них две старшие равны 1. Тогда значение числа A равно:

$$A = F_k + F_{k-1} + B,$$

где B – сумма остальных чисел Фибоначчи, соответствующих ненулевым разрядам записи числа. В соответствии с тождеством (1) мы можем написать

$$A = F_k + F_{k-1} + B = F_{k+1} + B.$$

А это и означает, что в записи числа стало на одну цифру больше, при этом старшая $(k+1)$-я цифра равна 1, а k-я и $(k-1)$-я цифры равны 0.

Замечание 2. Операция развёртки *увеличивает* количество единиц в фибоначчиевой записи числа, но может уменьшить на единицу количество цифр в записи, если исходная запись начиналась с комбинации «100». Например, $\underline{100}0101_{fib} = \underline{11}0101_{fib}$.

Если над записью числа в ФСС выполнить все возможные *свёртки*, то мы придём к специальному фибоначчиевому представлению числа, называемому *минимальной формой*, в которой нет двух рядом стоящих единиц. Если же в фибоначчиевой записи выполнить все возможные операции развёртки, то придём к специальному фибоначчиевому изображению, называемо-

му *максимальной*, или *развёрнутой формой*, в которой рядом не встречается двух нулей. Любое число A представляется в минимальной или максимальной форме единственным способом.

Правило приведения фибоначчиевой записи числа к минимальной форме

1. Просматривая запись числа слева направо, найдём первую комбинацию «11».

2. Выполним над ней операцию свёртки:

2.1. если комбинация «11» стоит в начале записи числа, то количество цифр в записи числа увеличится на единицу. К записи числа слева припишется 1, а комбинация «11» заменится на «00»;

2.2. если самая левая комбинация «11» находится не в начале записи, это означает, что перед ней обязательно стоит 0. Тогда комбинация «011» заменяется на «100».

3. Последовательно выполним операции свёртки для всех комбинаций «11», двигаясь по записи числа слева направо.

4. Если в получившейся записи есть две рядом стоящие единицы, то переход на п. 1, иначе получившаяся запись будет искомой минимальной формой.

Пример 2. Даны два числа $A = 1111111_{fib}$ и $B = 1011101_{fib}$. Требуется привести их к минимальной форме.

Для приведения числа к минимальной форме выполним над ним все возможные операции свёртки, двигаясь по записи числа слева направо:

$A = \underline{11}111111_{fib} \Rightarrow 100\underline{11}1111_{fib} \Rightarrow 10100\underline{11}1_{fib} \Rightarrow 10101001_{fib}$.

Единицы, над которыми выполняется операция свёртки, подчёркнуты.

$B = 10\underline{11}101_{fib} \Rightarrow 1100101_{fib}$.

После первого прохода по числу и выполнения операции свёртки оказалось, что вновь полученная запись опять содержит комбинацию «11». Следовательно, для полученной записи надо снова выполнить все возможные свёртки:

$B = \underline{11}00101_{fib} \Rightarrow 10000101_{fib}$.

То есть, $B = 10\underline{11}101_{fib} \Rightarrow \Rightarrow \underline{1}100101_{fib} \Rightarrow 10000101_{fib}$.

Вопрос читателю. Какое самое большое число в ФСС, представленное в минимальной форме, можно записать в k разрядах?

Очевидно, что самая старшая цифра должна быть 1. Далее для построения максимального числа цифры 0 и 1 должны чередоваться. Тогда максимальное число, которое можно записать в k разрядах, равно 10101...10 при чётном k и 10101...01 при нечётном k. Сравним для двоичной системы и ФСС максимальные числа, которые можно записать в 8- и 16-тиразрядной ячейках.

	Двоичная система счисления	Фибоначчиева система счисления
8-миразрядная ячейка	$2^8 - 1 = 255$	$10101010_{fib} = 54$
16-тиразрядная ячейка	$2^{16} - 1 = 65\,535$	$1010101010101010_{fib} = 2583$

Выполнение операции сложения в ФСС

Выполнение операции сложения в позиционных системах счисления происходит по таблицам сложения. Для десятичной системы вы знаете эту таблицу наизусть. Для двоичной системы таблица сложения довольно проста.

Таблица 1

0 + 0 =	0
0 + 1 =	1
1 + 0 =	1
1 + 1 =	10

Последняя строка в таблице 1 задаёт правило формирования переноса при двоичном сложении единиц (1 + 1 = 10).

Построим таблицу сложения для ФСС. Если оба слагаемых имеют 1 в разряде с номером $k \geq 3$, то при сложении единица в k-ом разряде сохраняется и возникает перенос двух единиц в соседние младшие разряды: в $(k-1)$-й и $(k-2)$-й. Это следует из тождества:

$$F_k + F_k = F_k + F_{k-1} + F_{k-2}.$$

Для $k = 1$ результат сложения:

1 + 1 = **10** (выделен 1-й разряд).

Для $k = 2$ результат сложения:

1 + 1 = **101** (выделен 2-й разряд).

Таблицу сложения в ФСС можно записать в виде:

Таблица 2

	1–ый разряд	2–ой разряд	k–ый разряд
0 + 0 =	0	0	0
0 + 1 =	1	1	1
1 + 0 =	1	1	1
1 + 1 =	10	101	111

Возникающий при выполнении сложения перенос будем накапливать в отдельной ячейке R, которую назовём *многоразрядным переносом*. Для первого и второго разрядов многоразрядный перенос не формируется, переносимые единицы сразу записываются в формируемую сумму. Таким образом, не возникнет ситуация, когда в одном из разрядов перенос состоит более чем из одной единицы.

Алгоритм сложения целых чисел в ФСС. Перед сложением суммируемые фибоначчиевые числа приводятся к минимальной форме. Поразрядное сложение выполняется справа налево.

Будем использовать следующие обозначения:

S – промежуточная сумма;

R – многоразрядный перенос.

1. Выполним поразрядное сложение чисел в соответствии с таблицей 2.

2. Промежуточная сумма S приводится к минимальной форме.

3. Если полученный многоразрядный перенос R равен 0, то переход на п. 6.

4. Выполняется сложение $S + R$, в результате которого формируется новое значение S и R.

5. Переход на п. 2.

6. Промежуточная сумма S, приведённая к минимальной форме, и есть результат сложения. Конец алгоритма.

Пример 3. Требуется сложить два числа $a = 10101_{fib}$ и $b = 1001_{fib}$ в ФСС.

В нашем примере числа уже записаны в минимальной форме, две единицы стоят в первом разряде, поэтому многоразрядный перенос равен 0. В соответствии с таблицей 2 получим:

		1	0	1	0	1
	+		1	0	0	1
S	=	1	1	1	1	0

Результат приведём к минимальной форме, для этого дважды выполним операцию свёртки:

$$\underline{11}110 \Rightarrow 100\underline{11}0 \Rightarrow 101000.$$

Проверим себя, переведём исходные слагаемые и полученный результат в десятичную систему счисления: $a = {}^{8\,5\,3\,2\,1}10101 = 12$, $b = {}^{5\,3\,2\,1}1001 = 6$, $\mathbf{S} = {}^{13\,8\,5\,3\,2\,1}1\,01000 = 18$.

Пример 4. Сложим десятичные числа $A = 20_{10}$ и $B = 20_{10}$ в фибоначчиевой системе счисления. Сначала запишем числа в ФСС в минимальной форме: $A = B = 20_{10} = 101010_{fib}$.

Операция сложения будет выполняться в несколько шагов. В этом примере две единицы стоят во втором, четвёртом и шестом разрядах, но при сложении на первом шаге перенос бу-

дет возникать только для четвёртого и шестого разрядов (см. таблицу 2).

1)
$$\begin{array}{r} 101010 \\ +\,101010 \\ \hline S = 101101 \\ R = 011110 \end{array}$$

Получили промежуточную сумму S = = 101101 и многоразрядный перенос R = 11110.

2) Приведём S к минимальной форме: 101101 ⇒ 110001 ⇒ 1000001.

3) Вычислим $S + R$:

$$\begin{array}{r} 1000001 \\ +\;\;11110 \\ \hline S{=}1011111 \end{array}$$

Получили промежуточную сумму 1011111 и нулевой многоразрядный перенос.

4) Запишем S в минимальной форме:

$10\underline{11}111 \Rightarrow 1100\underline{111} \Rightarrow \underline{1}101001 \Rightarrow$ ⇒10001001.

Проверим себя:

$$\begin{array}{c} {\scriptstyle 34\,21\,13\,8\,5\,3\,2\,1} \\ 1\;\;0\;\;0\;0\,1\,0\,0\,1 = 34+5+1 = 40\,. \end{array}$$

Алгоритм фибоначчиевого сложения кажется сложным и утомительным, но после приобретения некоторого опыта эта процедура оказывается даже занятной.

Базовые фибоначчиевые операции

Любое число в ФСС в минимальной форме записывается однозначным образом. Так как цифрами ФСС являются 0 и 1, то если при передаче фибоначчиева представления числа в минимальной форме появились две подряд идущие единицы, можно однозначно сказать, что информация пришла с искажением (потерей).

«Естественная» избыточность кодов Фибоначчи, которая проявляется в «множественности» представлений одного и того же числа, может быть использована для целей контроля числовых преобразований в цифровых устройствах.

В 80-х годах XX столетия в Таганрогском радиотехническом институте группа учёных под руководством профессора А.П. Стахова пришла к идее создания процессора, позволяющего обнаруживать сбои триггеров, возникающие в момент их переключения [4]. Такой процессор они называли *помехоустойчивым процессором*. Создание помехоустойчивого процессора базировалось на следующих принципах:

- вся обрабатываемая информация должна быть представлена в фибоначчиевой системе счисления;
- необходимо выбрать некоторый набор *базовых операций*, на основе которых может быть реализован любой алгоритм обработки информации;
- ввести эффективную систему схемного контроля базовых операций.

В качестве базовых операций были выбраны:

1) свёртка S;
2) развёртка R;
3) перемещение M;
4) поглощение P.

Первые две операции были рассмотрены выше.

Микрооперация *перемещения* является поразрядной двуместной операцией $M(A, B)$. Если регистр A имеет двоичную цифру 1 в k-м разряде, а регистр B в этом разряде имеет цифру 0, то в результате операции перемещения цифра 1 из регистра A перемещается в регистр B, а цифра 0 из регистра B перемещается в регистр A.

$$\begin{array}{|ccc|} \hline A & 1 & 0 \\ \mathbf{M} & & = \\ B & 0 & 1 \\ \hline \end{array}$$

Микрооперация *поглощения* $P(A, B)$ также является поразрядной двуместной операцией и состоит в том, что две двоичные едини-

цы одного и того же разряда регистров A и B взаимно уничтожаются, то есть заменяются нулями.

A	1	0
P		=
B	1	0

Функциональная полнота базовых фибоначчиевых операций

Покажем, что любую логическую операцию можно выполнить через базовые фибоначчиевы операции [5]. Для этого достаточно показать, что с помощью этих операций можно реализовать конъюнкцию, дизъюнкцию и отрицание. А так как система операций $\{\wedge, \vee, \neg\}$ является полной [3], то в итоге мы сможем выразить через базовые операции любые другие.

Вопрос читателю. Что нам даёт знание того, что некоторая система операций над двоичными последовательностями является полной?

Так как работа всего компьютера построена на логических преобразованиях двоичных последовательностей, то для проектирования компьютера достаточно уметь строить ограниченный набор логических элементов, каждый из которых реализует определённую операцию или функцию, входящую в некоторую полную систему. Известно, что современная цифровая техника строится на основе трёх логических элементов – конъюнктора, дизъюнктора и инвертора. Но оказывается, что базовые операции $\{M, P\}$ вместе с константой 1 также образуют полную систему.

1) Покажем, что через базовые операции $\{M, P\}$ с использованием константы 1 можно выразить любую логическую операцию булевой логики. Напомним таблицы истинности для основных логических операций булевой логики – конъюнкции, дизъюнкции и отрицания.

Таблица 3

x	y	x ∧ y	x ∨ y	¬x
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Операции перемещения $M(A, B) = (A', B')$ и поглощения $P(A, B) = (A'', B'')$ можно рассматривать как логические операции с двумя аргументами и двумя значениями. Составим таблицы истинности для этих операций.

Таблица 4

		$(A', B') = M(A, B)$		$(A'', B'') = P(A, B)$	
A	B	A'	B'	A''	B''
0	0	0	0	0	0
0	1	0	1	0	1
1	0	0	1	1	0
1	1	1	1	0	0

Сравнивая таблицы 3 и 4, мы видим, что A' содержит результат поразрядной конъюнкции, т. е. $A' = A \wedge B$, а B' – результат поразрядной дизъюнкции, т. е. $B' = A \vee B$.

Логическая операция *отрицание* сводится к выполнению операции *поглощения* над исходной кодовой комбинацией A и константой 1. В результате операции поглощения $(A'', B'') = P(A, B)$, где $B \equiv 1$, мы получаем $B'' = \overline{A}$ и $A'' \equiv 0$.

2) Покажем, что через базовые операции $\{S, R, M, P\}$ можно реали-

зовать фибоначчиево сложение регистров A и B. В основе этой реализации лежит идея перебрасывания единиц из k-го разряда регистра A в k-й разряд регистра B, если там был 0. При этом на каждом шаге сумма модифицированных регистров остаётся прежней. Алгоритм заканчивается, когда регистр A становится равным 0, в этом случае искомая сумма накоплена в регистре B.

В качестве примера просуммируем два числа: $A_0 = 10100100 = 50_{10}$ и $B_0 = 01010100 = 32_{10}$.

1 шаг	Выполним операцию перемещения: $(A_1, B_1) = M(A_0, B_0)$.	$A_0 = 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0$ $B_0 = 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0$ $A_1 = 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0$ $B_1 = 0\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 0$
2 шаг	Выполним все возможные операции развёртки над A_1: $A_2 = R(A_1)$.	$A_2 = 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1$
3 шаг	Выполним все возможные свёртки над B_1: $B_2 = S(B_1)$.	$B_2 = 1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0$
4 шаг	Выполним операцию перемещения: $(A_3, B_3) = M(A_2, B_2)$.	$A_2 = 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1$ $B_2 = 1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0$ $A_3 = 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0$ $B_3 = 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1$
5 шаг	Приведём B_3 к минимальной форме, выполнив операцию свёртки: $B_4 = S(B_3)$.	$B_4 = 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1$

Проверим себя: $B_4 = 101001001_{fib} = 55 + 21 + 5 + 1 = 82_{10}$.

Можно показать, что операции вычитания, умножения и деления также реализуются через базовые операции. Таким образом, через базовые операции $\{S, R, M, P\}$ можно выразить любую логическую и любую арифметическую операцию. И, следовательно, мы можем проектировать Фибоначчи–процессор на основе базовых микроопераций.

Вопрос читателю. Какое преимущество будет иметь такой Фибоначчи–процессор в сравнении с классическими компьютерами, основанными на классической двоичной системе счисления?

Ответ на этот вопрос состоит в том, что такой процессор обладает возможностью контроля ошибок, которые могут возникнуть в процессоре под влиянием различных внешних и внутренних факторов [6].

К сожалению, создать Фибоначчи-компьютер по разным причинам пока так и не удалось: группа А.П. Стахова спроектировала только экспериментальный Ф-процессор, который действительно сам определял, произошёл ли сбой при работе процессора. При разработке элементной базы Ф-процессора основным операционным элементом стало устройство приведения кода Фибоначчи к минимальной форме. Это устройство реализовывалось через RS-триггеры и логические элементы И и ИЛИ.

Теоретические основы данного направления представляют несомненный интерес и могут стать источником новых идей не только в компьютерной области. Современ-

ные математики проявляют большой интерес к «фибоначчиевому» направлению. В 1963 г. группа американских математиков, возглавляемая Вернером Хоггаттом, организовала математическую Фибоначчи-Ассоциацию, которая выпускает журнал «The Fibonacci Quarterly» и ежегодно с 1984 г. проводит международную конференцию «Fibonacci Numbers and Their Applications».

Литература

1. *Фалина И.Н.* Компьютер и фибоначчиева система счисления // Потенциал. – 2011. – №6.
2. *Андреева Е., Фалина И.* Системы счисления и компьютерная арифметика. – М.: БИНОМ. Лаборатория знаний, 2004.
3. *Андреева* Е., *Босова Л., Фалина И.* Математические основы информатики. – М.: БИНОМ. Лаборатория знаний, 2007.
4. Помехоустойчивые коды: Фибоначчи – компьютер. – Сб. статей. Серия «Радиоэлектроника и связь». – М.: Знание, 1989.
5. *Стахов А.П.* Введение в алгоритмическую теорию измерений. – М.: Сов. Радио, 1977.
6. *Стахов А.П.* Коды золотой пропорции. – М., 1984.

Юмор Юмор Юмор Юмор Юмор Юмор

Программист сидит за компьютером и сосредоточенно отлаживает программу. К нему подходит сынишка и спрашивает:

– Папа, а почему солнышко каждый день встает на востоке, а садится на западе?

– Ты это проверял?

– Проверял.

– Хорошо проверял?

– Хорошо.

– Работает?

– Работает.

– Каждый день работает?

– Да, каждый день.

– Тогда ради бога, сынок, ничего не трогай, ничего не меняй!

* * *

Программист пришел к своему другу-музыканту посмотреть на новое пианино. Долго ходил вокруг, наконец произнес:

– Клавиатура неудобная – всего 88 клавиш, из них половина – функциональные, причем ни одна не подписана... Хотя... нажимать Shift ногой – весьма оригинально!

* * *

Программист стоит возле окна и попеременно то открывает, то закрывает его. Друг спрашивает:

– Что ты делаешь?

– Нет, ну ты представляешь: открыть могу, закрыть могу, а свернуть не могу!