МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Омский государственный технический университет»

ПРИМЕРЫ ВЫПОЛНЕНИЯ ЗАДАНИЙ К ПРАКТИЧЕСКИМ ЗАНЯТИЯМ ПО ДИСЦИПЛИНЕ «ИНФОРМАТИКА»

Учебное текстовое электронное издание
локального распространения

Омск
Издательство ОмГТУ
2015

Сведения об издании: 1, 2

Составитель *В. Ф. Нестерук*

Примеры выполнения заданий к практическим занятиям по дисциплине «Информатика» : метод. указания / Минобрнауки России, ОмГТУ ; [сост. В. Ф. Нестерук]. – Омск : Изд-во ОмГТУ, 2015.

Приводятся краткие теоретические сведения по терминологии алгебры логики и минимизации логических функций с помощью карт Карно. Рассматриваются примеры графического представления алгоритмов выполнения арифметических операций над двоичными кодами чисел и примеры машинной реализации этих операций.

Для студентов, обучающихся по направлению «Информатика и вычислительная техника».

Рекомендовано редакционно-издательским советом
Омского государственного технического университета

1 электронный оптический диск

Оригинал-макет издания выполнен в Microsoft Office Word 2007
с использованием возможностей Adobe Acrobat X.

Минимальные системные требования:

- процессор Intel Pentium 1,3 ГГц и выше;
- оперативная память 256 Мб;
- свободное место на жестком диске 260 Мб;
- операционная система Microsoft Windows XP/Vista/7;
- разрешение экрана 1024×576 и выше;
- акустическая система не требуется;
- дополнительные программные средства Adobe Acrobat Reader 5.0 и выше.

Редактор *Т. А. Москвитина*
Компьютерная верстка *Ю. П. Шелехиной*

Сводный темплан 2015 г.
Подписано к использованию 18.05.15.
Объем 612 Кб.

Издательство ОмГТУ.
644050, г. Омск, пр. Мира, 11; т. 23-02-12
Эл. почта: info@omgtu.ru

Общие сведения

В первой части методических указаний содержатся краткие теоретические сведения по терминологии алгебры логики и минимизации логических функций с помощью карт Карно. Рассматриваются соответствующие примеры. В приложении 1 приводится перечень заданий в расчёте на две студенческие группы по 30 человек. Даётся методика работы с компьютерными тестами для проверки результатов минимизации.

Во второй части методических указаний приводятся примеры графического представления алгоритмов выполнения арифметических операций над двоичными кодами чисел и машинной реализации этих операций. В приложении 2 даётся перечень соответствующих заданий на разработку алгоритмов выполнения арифметических операций по заданному методу над двоичными числами, представленными в одном из машинных кодов.

Терминология алгебры логики

Алгебра логики – это алгебра высказываний типа «истина» – «ложь». Допускается обозначать их символами 1 – 0, которые не рассматриваются как числа.

Логическая (булева) переменная – это **простое высказывание**, которое может принимать значения только из множества {0,1}.

Логические связки – объединяют простые высказывания в **логические функции**. Например, логическими связками могут быть символы: \/ (возможно +, «or», «или»); /\ (возможно *, «and», &, «и»).

Логическая функция (ЛФ) – функция f(x1, x2, ..., xn) называется логической (**переключательно**й или **булевой**) функцией **n** аргументов **xi** , если она и ее аргументы, могут принимать значения только из множества {0,1}.

Набор – совокупность значений аргументов логической функции. Любая ЛФ **n** аргументов может иметь **m =** 2^n наборов. Каждому набору значений аргументов приписывается **номер** (0,1,…, m–1), полученный как вес двоичного позиционного числа, совпадающего по кодировке с набором.

Количество различных ЛФ **n** аргументов конечно и равно **k** = 2^m. Каждой логической функции данного набора аргументов принято приписывать **номер (**0, 1,…, k–1**)**, полученный как вес двоичного позиционного числа, совпадающего по кодировке со значениями ЛФ на наборах от 0 до m–1.

x2	x1	f0	f1	f2	f3	f4	f5	f6	…	f11	f12	f13	f14	f15
0	0	0	1	0	1	0	1	0		1	0	1	0	1
0	1	0	0	1	1	0	0	1		1	0	0	1	1
1	0	0	0	0	0	10	1	1		0	1	1	1	1
1	1	0	0	0	0		0	0		1	1	1	1	1

Так, например, множество бинарных ЛФ f(x1, x2) включает в себя 16 функций.

За каждой бинарной ЛФ закреплено определённое название, например:

f14 – **дизъюнкция (**логическое **«или»,** логическое сложение, «**OR**»**);**

f8 – **конъюнкция (**логическое **«и»,** логическое умножение, «**AND**»);

f7 – **штрих Шеффера («и-не»);**

f1 – **стрелка Пирса** (**«или-не**»);

f6 – **исключающее ИЛИ** (**«сумма по mod2»**, «**XOR**»);

f5 – **инверсия x1** (**«не x1**», отрицание x1);

f3 – **инверсия x2** (**«не x2**» , отрицание x2).

Литерал – логическая переменная или ее инверсия (отрицание). Например, переменная A и ее инверсия !A – это одна переменная с разными значениями, но два литерала.

Таблица истинности – таблица, в которой всем возможным наборам аргументов соответствуют определённые значения ЛФ. Так, функция **XOR** имеет следующую таблицу истинности.

x2	x1	f6
0	0	0
0	1	1
1	0	1
1	1	0

Неполностью определенная ЛФ **n** переменных, это функция, определённая на числе наборов меньшем **m.**

Суперпозиция – подстановка в логическую функцию вместо ее аргументов других логических функций.

Функционально полная система логических функций – система, с помощью логических функций которой, применяя операции суперпозиции и подстановки, можно получить любую сколь угодно сложную логическую функцию.

Терм – группа логических переменных в прямой или инверсной форме (группа литералов), объединенных одним и тем же знаком логической связки: логического сложения ∨ (дизъюнкции) или же логического умножения ∧ (конъюнкции). В терме каждая переменная или ее отрицание встречается только один раз.

Ранг терма – количество переменных и их инверсий (количество литералов), входящих в данный терм. Терм, в который входят все переменные или их отрицания, имеет **максимальный ранг.**

Макстерм (дизъюнктивный терм) – терм, в котором литералы связаны знаком дизъюнкции.

Минтерм (конъюнктивный терм) – терм, в котором литералы связаны знаком конъюнкции.

Конституента единицы (К1) тождественна минтерму.

Конституента нуля (К0) тождественна макстерму.

Базис – функционально полная система элементарных функций. Пример классического базиса: {«И», «ИЛИ», «НЕ»}. Пример минимального базиса: {«штрих Шеффера»}.

Способы представления логических функций: табличный (рассмотрен выше), аналитический, числовой, геометрический (графический).

Нормальные формы аналитического представления ЛФ:

ДНФ – ЛФ = $\vee$ Fi, i$\leq$ m, где Fi – минтермы любого ранга;

КНФ – ЛФ = $\wedge$ Hi, i$\leq$ m, где Hi – макстермы любого ранга.

Совершенная (стандартная или каноническая) форма аналитического представления ЛФ:

СДНФ – содержит минтермы только максимального ранга;

СКНФ – содержит макстермы только максимального ранга.

Минимизация логической функции – получение из исходного аналитического представления ЛФ, используя операции алгебры логики, аналитическое представление этой ЛФ с меньшим числом литералов.

МДНФ — минимальная ДНФ, имеющая по сравнению с исходной ДНФ возможно меньшее количество минтермов с возможно меньшими рангами, ни один из которых исключить нельзя, или ДНФ, содержащая наименьшее количество литералов.

МКНФ — минимальная КНФ, имеющая по сравнению с исходной КНФ возможно меньшее количество макстермов с возможно меньшими рангами, ни один из которых исключить нельзя, или КНФ, содержащая наименьшее количество литералов.

Соседние термы – термы в ДНФ или КНФ, которые отличаются только одной переменной (в одном терме переменная без отрицания, а в другом – с отрицанием).

Числовой способ представления ЛФ. В случае СДНФ под знаком \/ перечисляются заключенные в скобки номера наборов, на которых функция равна единице. В случае СКНФ под знаком /\ перечисляются заключенные в скобки номера наборов, на которых функция равна нулю. Например, f5 = \/ (0, 2); f2 = /\ (0, 2, 3).

Геометрический (графический) способ представления ЛФ. Термы функции **n** переменных размещаются по вершинам n-мерного куба, дискретом каждой координаты которого является соответствующая логическая переменная.

Импликанта – некоторая логическая функция, входящая в данную ЛФ и обращаемая в ноль при наборе переменных, на котором ЛФ также равна нулю.

Импликанта дизъюнктивная – минтерм или группа минтермов ДНФ.

Импликанта конъюнктивная – любой макстерм, или группа макстермов исходной КНФ.

Простые или э**лементарные импликанты** – импликанта минтерма или макстерма, никакая собственная часть которого уже не является импликантой данной ЛФ.

Сокращенная форма аналитического представления ЛФ – дизъюнкция всех ее простых импликант.

Тупиковая форма аналитического представления ЛФ – дизъюнкция простых импликант, ни одну из которых исключить нельзя. Минимальные формы ЛФ выбирают из тупиковых форм с минимальным числом литералов.

Поглощение – A(A + B) = A; A + AB = A.

Склеивание – AB + !AB = B (A + !A) = B; (A + B)(A + !B) = A.

Неполное склеивание – xy + x!y = x + xy + x!y.

Формула развертывания – (x + y) = (x + y + z)(x + y + !z).

Теорема (законы) де Моргана – !(A + B) = !A*!B; !(A*B) = !A + !B.

Пример минимизации СДНФ ЛФ с помощью карт Карно

Карты Карно (модифицированные карты Вейча) представляют собой матрицу с количеством ячеек по числу возможных наборов переменных ЛФ, каждая ячейка которой соответствует определённому минтерму и помечается символом «истина» (например, 1 или любой символ) или «ложь» (например, 0 или любой символ, отличный от «истины») в соответствии со значениями данной ЛФ, причём минтермы в соседних ячейках являются соседними.

Пример разметки карты Карно для ЛФ от 4 переменных (n = 4) представлен на рис. 1. Ячейки карты помечены соответствующими символическими записями минтермов для данной разметки.

	!X4	X4		!X4	
!X1	**!X4*!X3*!X2*!X1**	**X4*!X3*!X2*!X1**	**X4*X3*!X2*!X1**	**!X4*X3*!X2*!X1**	!X2
	!X4*!X3*X2*!X1	**X4*!X3*X2*!X1**	**X4*X3*X2*!X1**	**!X4*X3*X2*!X1**	X2
X1	**!X4*!X3*X2*X1**	**X4*!X3*X2*X1**	**X4*X3*X2*X1**	**!X4*X3*X2*X1**	
	!X4*!X3*!X2*X1	**X4*!X3*!X2*X1**	**X4*X3*!X2*X1**	**!X4*X3*!X2*X1**	!X2
	!X3		X3		

Рис. 1

На рис. 2 в карте Карно размещена логическая функция, СДНФ которой имеет вид

f = !X4*!X3*!X2*!X1 + X4*!X3*X2*!X1 + X4*X3*X2*!X1 +
+ X4*!X3*X2*X1 + X4*X3*X2*X1 + !X4*X3*!X2*!X1 +
+ X4*!X3*!X2*X1 + !X4*X3*!X2*X1,

где символам !, *, + соответствуют логические операции инверсии, конъюнкции и дизъюнкции.

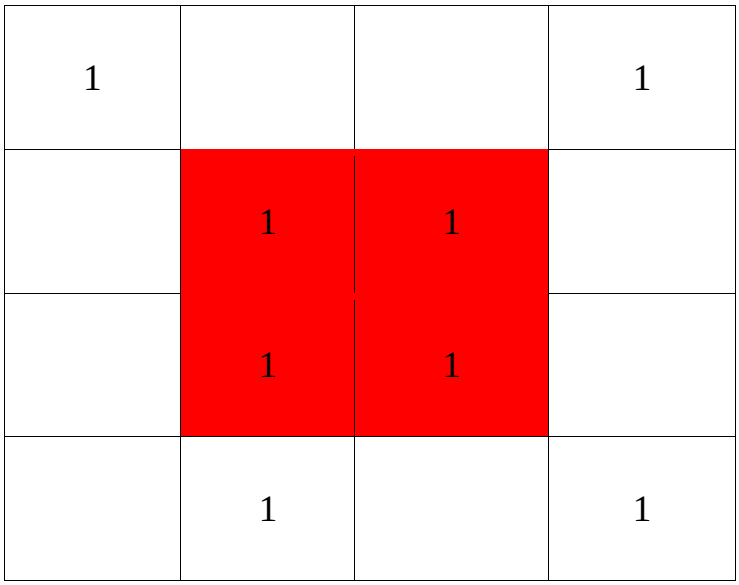

Рис. 2

Метод Карно предусматривает выполнение операций склеивания и покрытия. Склеиванию подлежат пары, четвёрки, восьмёрки и т. д. 2^n минтермов в соседних недиагональных ячейках. Матрица карты свёрнута в цилиндры по левой-правой и нижней-верхней границам. Например, крайние правые ячейки всех строк являются соседними для крайних левых ячеек этих строк, а нижние ячейки столбцов являются соседними для верхних ячеек этих столбцов.

Допускаются частичные покрытия одних склеек другими с целью охвата всех минтермов данной ДНФ, а также наличие минтермов, не вошедших ни в одну склейку.

В первую очередь выполняются склейки максимально возможного размера. На рис. 2 – это выделенная четвёрка минтермов в центре матрицы.

Далее выполняются склейки меньшего размера. На рис. 3–5 – это выделенные пары минтермов.

Склейка на рис. 5 частично покрывает склейку рис. 1. В результате выполненных склеек были охвачены все минтермы ЛФ. Каждая из склеек позволяет исключить из соответствующей группы минтермов те переменные, от которых эта склейка не зависит. Например, склейка на рис. 2 не зависит от переменных X1 и X3, что позволяет получить минимальную форму для данной склейки вида X4*X2, склейка на рис. 3 не зависит от X3, склейка на рис. 4 не зависит от X1, а склейка на рис. 5 не зависит от X2. Получаем соответствующие минимальные формы !X4*!X2*!X1, !X4*X3*!X2 и X4*!X3*X1.

Результирующая МДНФ ЛФ будет иметь вид

f = X4*X2 + !X4*!X2*!X1 + !X4*X3*!X2 + X4*!X3*X1.

1			1
	1	1	
	1	1	
	1		1

Рис. 3

1			1
	1	1	
	1	1	
	1		1

Рис. 4

В общем случае может быть несколько минимальных форм ЛФ, из которых выбирается та, которая в большей степени по тем либо иным критериям устраивает разработчика в плане дальнейшей аппаратной или программной реализации.

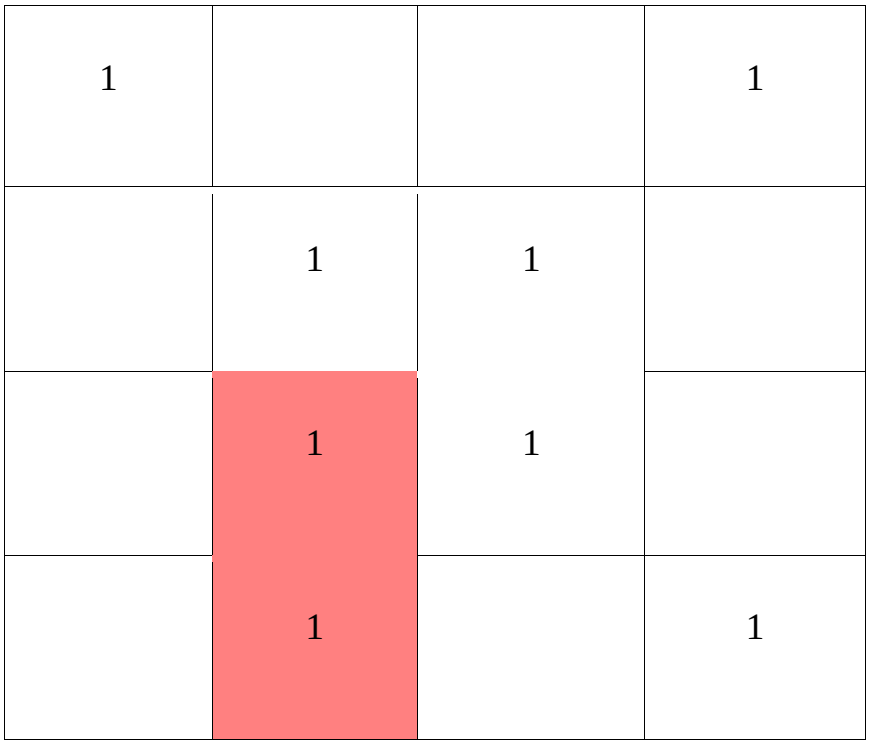

Рис. 5

Этапы выполнения задания по первой части практических занятий

Каждому студенту в качестве первой части задания выдаётся индивидуальная карта Карно (прил. 1). При выполнении задания необходимо:

– составить аналитическую форму СДНФ соответствующей ЛФ;

– выполнить минимизацию ЛФ по заданной карте Карно;

– составить аналитическую форму МДНФ ЛФ;

– выполнить проверку результата минимизации на компьютерном тесте;

– сдать отчёт преподавателю.

ТЕСТ НА ТЕМУ "МИНИМИЗАЦИЯ ЛОГИЧЕСКОЙ ФУНКЦИИ С ПОМОЩЬЮ КАРТ КАРНО "

Определите минимальную форму функции четырёх переменных

F= |X4*|X3*|X2*|X1+X4*|X3*|X2*|X1+X4*|X3*X2*|X1+X4*X3*X2*|X1+

+X4*X3*X2*X1+|X4*X3*X2*X1+|X4*X3*|X2*X1+|X4*|X3*|X2*X1

и укажите в каждом из прилагаемых ниже подпунктов количество входящих в минимальную форму функции минтермов соответствующего ранга

1. Укажите количество минтермов первого ранга:

отсутствуют
один
два
три
четыре
пять
шесть

2. Укажите количество минтермов второго ранга:

отсутствуют
один
два
три
четыре
пять
шесть

3. Укажите количество минтермов третьего ранга:

отсутствуют
один
два
три
четыре
пять
шесть

4. Укажите количество минтермов четвёртого ранга:

отсутствуют
один
два
три
четыре
пять
шесть

Показать результат

Очистить выбранные пункты

Работа с компьютерным тестом

После вызова .html- страницы соответствующего компьютерного теста открывается рабочее окно, в котором необходимо:

– проверить соответствие аналитической формы СДНФ;

– в каждом из подпунктов теста отметить строку, соответствующую числу минтермов данного ранга в полученной МДНФ;

– нажать на кнопку “Показать результат”.

В окне сообщений даётся оценка “удовлетворительно” в случае правильного ответа либо “неудовлетворительно” в противном случае (рис. 6).

Количество правильных ответов 0. Ваша оценка "НЕУДОВЛЕТВОРИТЕЛЬНО". Загляните в окно рядом с номером вопроса. Если Вы ответили правильно, то там (+), если Вы ошиблись, то там (-).

Рис. 6

При неудовлетворительном результате необходимо нажать на кнопку “Очистить выбранные пункты” и повторить тестирование после исправления ошибок.

Так как минимальных форм ЛФ может быть несколько и полученная студентом МДНФ может не совпадать с тестовой, следует обратиться к преподавателю и защитить правильность полученной МДНФ ЛФ.

Основы машинной арифметики

Арифметические операции в ЦВМ выполняются преимущественно над позиционными двоичными кодами чисел. Любое целое число может быть представлено в виде ряда

$$A = \sum_{i} a_i * p^i \text{ , где: } i = 0, \ldots,(n-1);\ a_i = 0,1;\ p = 2 \text{ для } A < 2^n$$

Из каждого коэффициента ряда образуется цифра i-й позиции n-разрядного кода двоичного кода числа А. Разряд младшей позиции кода размещается справа (i = 0), разряд старшей позиции – слева (i = n–1). Напри-

мер, число четырнадцать разлагается по степеням числа 2 в следующий ряд:

$$A = 1*8 + 1*4 + 1*2 + 0*1.$$

Соответствующий позиционный двоичный код числа А имеет вид 1110. Допускается расширение кода нулями слева для получения требуемой разрядности. Например, этот же код в восьмиразрядной сетке имеет вид 00001110.

Полученный код является беззнаковым. Для кодирования чисел со знаками перед левой границей кода вводятся один (простой код) или два (модифицированный код) знаковых разряда. Знаки положительных чисел кодируются соответственно 0, а знаки отрицательных чисел кодируются 1.

Различают прямые, обратные и дополнительные коды. В прямых кодах арифметические операции выполняются отдельно над модулями чисел и отдельно над знаковыми битами. В обратных и дополнительных кодах арифметические операции выполняются над полноразрядными кодами, включая знаковые биты.

Прямой код числа получается из беззнакового путём добавления знакового бита (битов) слева от старшего бита. Так, для беззнакового кода числа 1110 прямой код положительного числа имеет вид 0.1110 (простой код) или 00.1110 (модифицированный код), а прямой код отрицательного числа – соответственно 1.1110 или 11.1110. Точка (или запятая) отделяет знак от модуля числа как для чисел, меньших единицы, так и для чисел, больших единицы. В первом случае для простых кодов позиции битов отсчитываются справа от точки, начиная с –1 в сторону убывания: –1, –2, …, –(n–1), а во втором случае – с (n –2),…,1,0 для n-разрядного кода (включая знаковый бит).

Для n-разрядных модифицированных кодов позиции битов будут соответственно в диапазонах –1, –2,…, –(n–2) и (n–3),…,1,0.

Обратные коды чисел получают из прямых кодов путём инверсии битов модуля отрицательных чисел. Для рассмотренных выше примеров коды положительных чисел остаются без изменения, а коды отрицательных чисел будут иметь вид 1.0001 или 11.0001.

Дополнительные коды чисел получают из обратных кодов отрицательных чисел путём прибавления единицы младшего разряда к младшему разряду кода. Коды положительных чисел остаются без изменения. Для ранее рассмотренных примеров дополнительные коды будут иметь вид 1.1111 и 11.1111.

Системы представления двоичных чисел

Основными системами представления являются система с фиксированной точкой (или запятой) и система с плавающей точкой (или запятой) – полулогарифмическая система.

Система с фиксированной точкой была рассмотрена в примерах, иллюстрирующих коды двоичных чисел. Ноль в любом коде должен быть представлен в виде положительного нуля.

Система с плавающей точкой предполагает определять вес числа по формуле

$$A = Am * 10^{p},$$

где Am – мантисса числа А;

10 – основание системы счисления (в данном случае двойка);

p – порядок числа А.

Мантисса всегда является числом, меньшим единицы по модулю, а порядок – целым числом.

Так, для ранее рассмотренных примеров беззнакового целого числа 1110 возможны следующие варианты представления в прямых кодах:

– положительные числа Am = 0.1110 (или 00.1110), p = 0.0100 (или 00.0100), т. е. вес порядка 4;

Am = 0.01110, p = 0.0101, т. е. вес порядка 5.

В первом случае для прямых кодов число считается нормализованным (в старшем разряде модуля мантиссы находится 1), во втором случае число денормализовано (в старшем разряде модуля мантиссы находится 0);

– отрицательные числа Am = 1.1110 (или 11.1110), p = 0.0100 (или 00.0100), т. е. вес порядка 4;

Am = 1.01110, p = 0.0101, т. е. вес порядка 5.

Если беззнаковое число 1110 рассматривается как меньшее единицы по модулю, соответствующие примеры примут вид:

– положительные числа Am = 0.1110 (или 00.1110), p = 0.0000 (или 00.0000), т. е. вес порядка 0;

Am = 0.01110, p = 0.0001, т. е. вес порядка 1;

– отрицательные числа Am = 1.1110 (или 11.1110), p = 0.0000 (или 00.0000), т. е. вес порядка 0;

Am = 1.01110, p = 0.0001, т. е. вес порядка.

Порядок числа может быть и отрицательным, например, когда в результате выполнения арифметической операции была получена денормализованная мантисса и при её нормализации порядок перешёл в отрицательную область. Пусть результат операции содержит Am = 0.000101101 и порядок p = 0.0001. После нормализации мантисса примет вид Am = 0.101101000, а порядок станет равным p = 1.0010, т. е. минус 2.

Отличие в представлении обратных и дополнительных кодов по сравнению с прямым кодом состоит в том, что для отрицательных мантисс признаком нормализации будет несовпадение знаковых битов и бита справа от точки (т. е. старшего значащего бита). Нулевая мантисса в любом коде должна быть представлена в виде положительного нуля.

Примеры выполнения машинных операций

Элементарные арифметические и логические преобразования в ЦВМ в основном выполняются в комбинационном арифметико-логическом устройстве (АЛУ). Для хранения кодов двоичных чисел используются регистры (Рг) или ячейки памяти (ЯП) запоминающих устройств (ЗУ). В каждом регистре и в каждой ячейке памяти возможно хранение одного числа. В Рг и ЯП возможны операции записи (Зп), чтения (Чт) и хранения. Основными арифметическими машинными операциями являются алгебраические суммирование, вычитание (короткие операции), умножение и деление (длинные операции).

Выполнение коротких машинных операций в системе с фиксированной точкой

На рис. 6 представлен алгоритм суммирования чисел в прямых кодах.

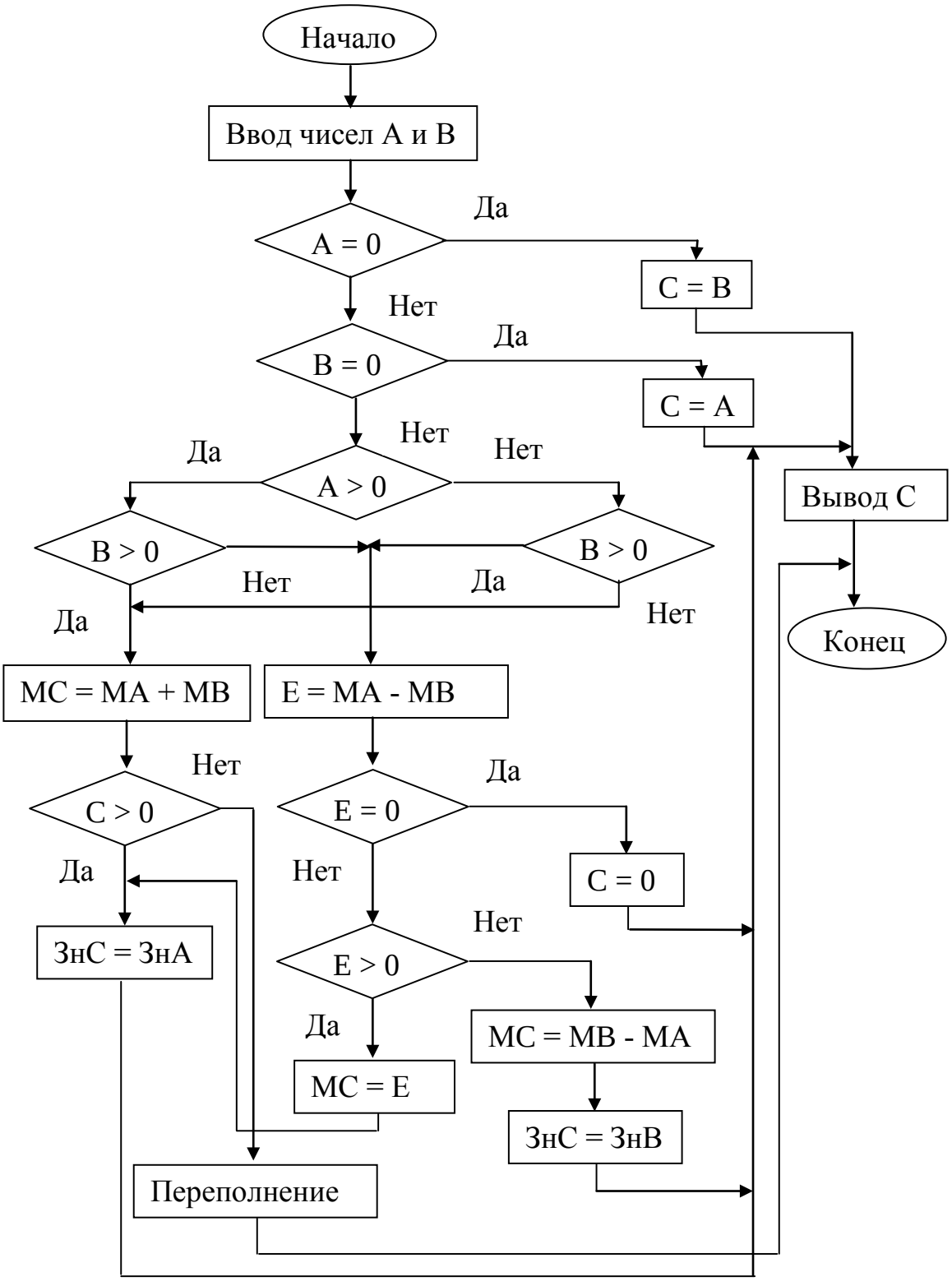

Рис. 6

Перед выполнением машинной операции требуется закрепить регистры АЛУ и ячейки ЗУ за операндами, промежуточными и конечными результатами. В современных процессорах число регистров колеблется от 8–10 до нескольких сотен. Поэтому в последующих примерах ЯП будут использоваться лишь для хранения больших массивов данных (например, при реализации табличных преобразований).

Для реализации алгоритма в ЦВМ потребуются два регистра РгА и РгВ для хранения операндов А и В, регистр РгК константы 0111…1 для выделения модуля, регистры модулей операндов РгМА и РгМВ, регистр результата РгС, регистр РгН (накапливающий регистр АЛУ), регистр РгО ошибки, в котором формируется сообщение о переполнении (отсутствие ошибки кодируем нулём).

Считаем, что операнды и константа загружены в соответствующие регистры, АЛУ может выполнять над двоичными кодами элементарные операции вида: сложение, левый логический сдвиг, обнуление, инверсия, конъюнкция, дизъюнкция. По результату операций АЛУ могут формироваться логические условия: выход левого сдвига ВСЛ, нулевой результат Z (Z = 1, если результат нулевой). На входы АЛУ может подключаться содержимое любого регистра или пары регистров. Результат любой элементарной операции загружается в РН с выхода АЛУ. Возможны пересылки между любой парой регистров, которые осуществляются через АЛУ.

Процесс суммирования будет состоять из следующих тактов.

1. Выделяем модуль А РгН : = (РгА) & (РгК)
2. Сохраняем модуль А РгМА : = (РгН)
3. Выделяем модуль В РгН : = (РгВ) & (РгК)
4. Сохраняем модуль В РгМВ : = (РгН)
5. Формируем инверсию константы (для выделения знака) РгН : = !(РгК)
6. Сохраняем инверсию константы РгК : = (РгН)
7. Проверка А на 0 РгН := (РгА), если Z = 1, идти к 25, иначе к 8
8. Проверка В на 0 РгН := (РгВ), если Z = 1, идти к 26, иначе к 9
9. Проверка знака А РгН := (РгА) & (РгК), если Z = 1, идти к 27, иначе к 10
10. Проверка знака В РгН := (РгВ) & (РгК), если Z = 1, идти к 15, иначе к 11
11. Выполняем суммирование модулей РгН := (РгМА) + (РгМВ)

12. Сохраняем модуль суммы РгС := (РгН)

13. Проверяем на переполнение РгН := <- (РгН), если ВСЛ = 1, идти к 28, иначе к 14

14. Выделяем знак А РгН := (РгА) & (РгК), идти к 21.

15. Выполняем прямое вычитание модулей РгН := (РгМА) – (РгМВ), если Z = 1, идти к 29, иначе к 16

16. Сохраняем модуль результата РгС := (РгН)

17. Проверяем знак разности, выполняя левый сдвиг РгН := <- (РгН), если ВСЛ = 1, идти к 18, иначе к 14

18. Выполняем обратное вычитание модулей РгН := (РгМВ) – (РгМА)

19. Сохраняем модуль результата РгС := (РгН)

20. Выделяем знак В РгН := (РгВ) & (РгК)

21. Объединяем модуль результата со знаком РгН := (РгС) \/ (РгН)

22. Сохраняем результат РгС := (РгН)

23. Формируем нулевое сообщение (пп. 23, 24) РгН := 0

24. РгО := (РгН). Конец.

25. Формируем результат С = В РгН := (РгВ), идти к 22

26. Формируем результат С = А РгН := (РгА), идти к 22

27. Проверка знака В РгН := (РгВ) & (РгК), если Z = 1, идти к 11, иначе к 15

28. Формируем ненулевое сообщение об ошибке РгН := (РгК), идти к 24

29. Формируем нулевой результат РгН := 0, идти к 22.

Рассмотрим числовые примеры, иллюстрирующие работу приведённой выше процедуры.

Пусть А = 1.0010110, В = 0.0110111

1. РгН := 1. 0010110 & 0.1111111 = 0.0010110
2. РгМА := 0.0010110
3. РгН := 0.0110111 & 0.1111111 = 0.0110111
4. РгМВ := 0.0110111
5. РгН := 1.0000000
6. РгК := 1.0000000
7. РгН := 1.0010110, Z = 0, идти к 8
8. РгН := 0.0110111, Z = 0, идти к 9
9. РгН := 1.0010110 & 1.0000000 = 1.0000000, Z = 0, идти к 10
10. РгН := 0.0110111 & 1.0000000 = 0.0000000, Z = 1, идти к 15

15. РгН := 0.0010110 – 0.0110111 = 1.1011111, Z = 0, идти к 16

16. РгС := 1.1011111

17. РгН := 10111110, ВСЛ = 1, идти к 18

18. РгН := 0.0110111 – 0.0010110 = 0.0100001

19. РгС := 0.0100001

20. РгН := 0.0110111 & 1.0000000 = 0.0000000

21. РгН := 0.0100001 ∨ 0.0000000 = 0.0100001

22. РгС := 0.0100001

23. РгН := 0.0000000

24. РгО := 0.0000000. Конец.

Пусть А = 0.1011110, В = 0.0110010

1. РгН := 0.1011110 & 0.1111111 = 0.1011110

2. РгМА := 0.1011110

3. РгН := 0.0110010 & 0.1111111 = 0.0110010

4. РгМВ := 0.0110010

5. РгН := 1.0000000

6. РгК := 1.0000000

7. РгН := 0.1011110, Z = 0, идти к 8

8. РгН := 0.0110010, Z = 0, идти к 9

9. РгН := 0.1011110 & 1.0000000 = 0.0000000, Z = 1, идти к 27

27. РгН := 0.0110010 & 1.0000000= 0.0000000, Z = 1, идти к 11

11. РгН := 0.1011110 + 0.0110010 = 1.0010000

12. РгС := 1.0010000

13. РгН := 0.0100000, ВСЛ = 1, идти к 28

28. РгН := 1.0000000, идти к 24

24. РгО := 1.0000000. Переполнение. Конец.

В первом примере суммировались разнознаковые числа, переполнение отсутствует, сформировано сообщение об отсутствии ошибки. Во втором примере суммировались равнознаковые числа, произошло переполнение, сформировано сообщение об ошибке.

На рис. 7 представлен алгоритм суммирования чисел в дополнительных кодах. Коды чисел суммируются целиком без отделения знаков, пе-

ренос из старшего разряда теряется. Признаком переполнения служит несоответствие знаков результата и одного из равнознаковых операндов. В систему операций АЛУ дополнительно введена операция суммирования по модулю два (m2).

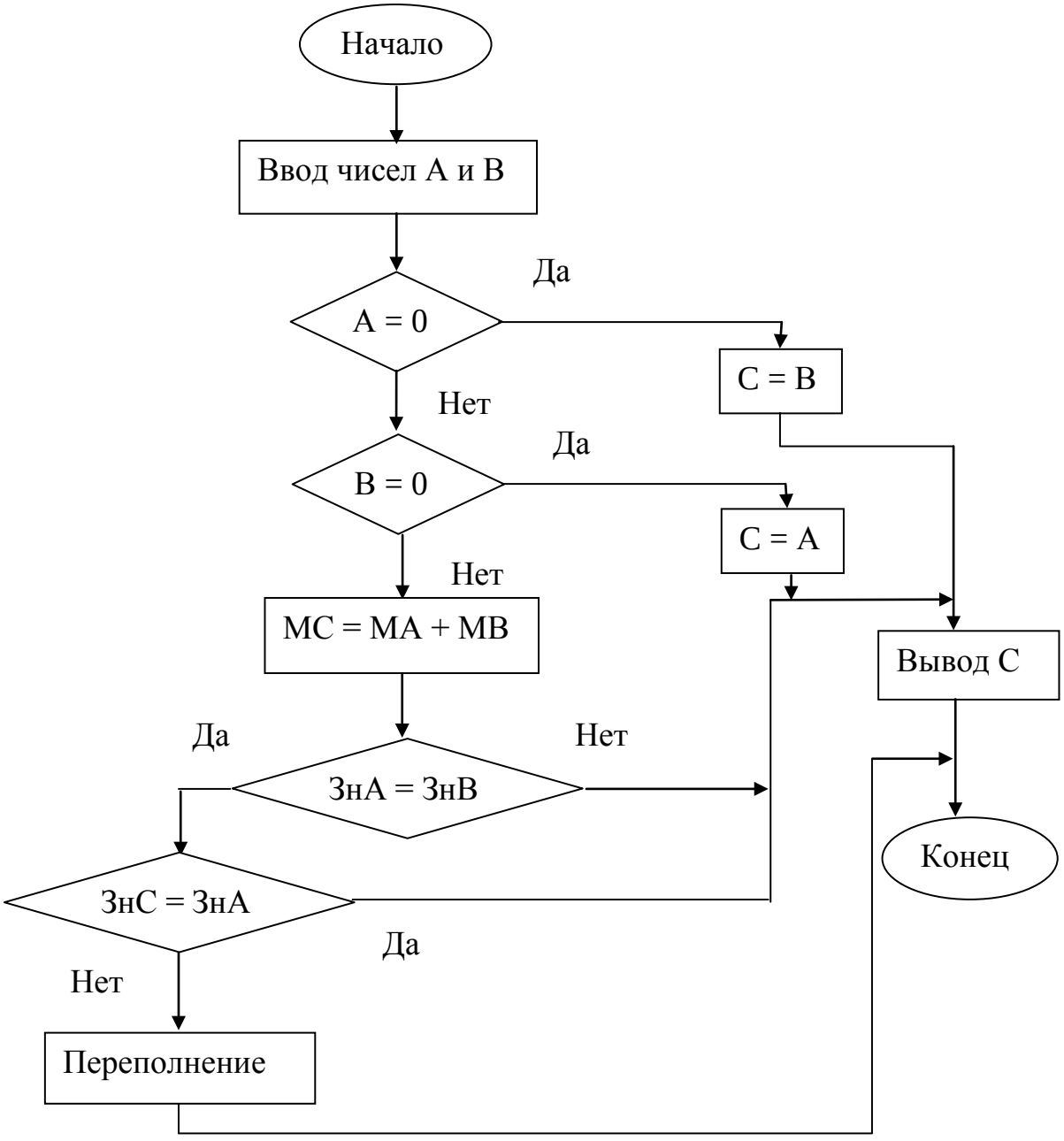

Рис. 7

Процесс суммирования будет состоять из следующих тактов.

1. Проверка A на 0 РгН := (РгA), если Z = 1, идти к 12, иначе к 2
2. Проверка B на 0 РгН := (РгB), если Z = 1, идти к 13, иначе к 3

3. Суммирование кодов чисел РгН := (РгА) + (РгВ)

4. Сохраняем результат РгС := (РгН)

5. Проверка равенства знаков операндов (пп. 5, 6) РгН := (РгА) m2 (РгВ)

6. РгН := <- (РгН), если ВСЛ = 1, идти к 9, иначе к 7

7. Проверка равенства знаков А и С (пп. 7, 8) РгН := (РгА) m2 (РгС)

8. РгН := <- (РгН), если ВСЛ = 1, идти к 11, иначе к 9

9. Формируем нулевое сообщение (пп. 9, 10) РгН := 0

10. РгО := (РгН). Конец

11. Формируем ненулевое сообщение об ошибке РгН := (РгК), идти к 10

12. Формируем результат С = В РгН := (РгВ), идти к 9

13. Формируем результат С = А РгН := (РгА), идти к 9.

Рассмотрим числовые примеры, иллюстрирующие работу процедуры суммирования дополнительных кодов чисел.

Пусть А = 1.1101010, В = 0.0110111

1. РгН := 1.1101010, Z = 0, идти к 2

2. РгН := 0.0110111, Z = 0, идти к 3

3. РгН := 1.1101010 + 0.0110111 = 0.0100001

4. РгС := 0.0100001

5. РгН := 1.1101010 m2 0.0110111 = 1.1011101

6. РгН := 1.0111010, ВЛС = 1, идти к 9

9. РгН := 0.0000000

10. РгО := 0.0000000. Конец.

Пусть А = 0.1011110, В = 0.0110010

1. РгН := 0.1011110, Z = 0, идти к 2

2. РгН := 0.0110010, Z = 0, идти к 3

3. РгН := 0.1011110 + 0.0110010 = 1.0010000

4. РгС := 1.0010000

5. РгН := 0.1011110 m2 0.0110010 = 0.1101100

6. РгН := 1.1011000, ВЛС = 0, идти к 7

7. РгН := 0.1011110 m2 1.0010000 = 1.1001110

8. РгН := 1.0011100, ВЛС = 1, идти к 11

11. РгН := 0.1111111, идти к 10

10. РгО := 0.1111111. Переполнение. Конец.

Отличие суммирования обратных кодов чисел от суммирования дополнительных кодов чисел заключается в учёте циклического переноса из старшего разряда АЛУ в младший разряд. Циклический перенос реализуется аппаратно. Коды чисел также суммируются целиком без отделения знаков, алгоритм соответствует рис. 8. Внешне числовые примеры не отличаются от двух предыдущих за исключением возможного появления отрицательного нуля в результате операции. Поэтому конечный результат всегда необходимо проверять на подобный исход и в случае появления отрицательного нуля необходимо преобразовать его в положительный код нуля.

В обратном коде отрицательный ноль имеет представление 1.111…1. Для его выявления можно выполнить инкремент на +1 или инверсию с дальнейшей проверкой на положительный код нуля. Ниже приводится соответствующий фрагмент операции.

…

n. Проверка на отрицательный ноль через инверсию

РгН := !(РгС), если Z = 1, идти к n+1, иначе к m

n+1. Инверсия результата

РгС := (РгН), идти к m

…

m. Конец.

Соответствующий фрагмент числового примера при (РгС) = 1.111…1 имеет вид

…

n. РгН := 0.000…0, Z = 1, идти к n+1

n+1. РгС := 0.000…0, идти к m

…

m. Конец.

Выполнение машинных операций в системе с плавающей точкой

При реализации операций в системе с плавающей точкой необходимо выполнение следующих условий:

– операнды поступают в нормализованном виде (т. е. модули мантисс M должны находиться в диапазоне $1 > M \geq 0{,}5$);

– результаты операции также должны быть нормализованы.

Мантиссы и порядки обрабатываются раздельно по правилам операций над числами с фиксированной точкой, но в отличие от мантисс порядки являются целыми числами. При отрицательном переполнении порядка результат может быть приравнен к нулю, а при положительном переполнении порядка фиксируется выход за диапазон представления.

Короткие операции (суммирование и вычитание) выполняются над числами с равными порядками. При выравнивании порядков один из операндов подвергается денормализации вправо (мантисса сдвигается вправо, а порядок инкрементируется). Если число сдвигов превышает разрядность мантиссы, операнд приравнивается к машинному нулю.

При выполнении длинных операций (умножение и деление) порядки складываются или вычитаются.

В процессе нормализации результата операции также выполняются преобразования порядка, что может также привести к его переполнению.

На рис. 8 приведён алгоритм процедуры выравнивания порядков pA и pB. Предполагается, что мантиссы Am и Bm нормализованы. Можно увеличивать меньший порядок, денормализуя вправо соответствующую мантиссу. Ниже при рассмотрении примера машинного выравнивания порядков в ранее сформированную систему операций АЛУ добавлены операции вычитания кодов чисел, правого арифметического сдвига, инкремента на +1 и декремента на –1. Числа представлены в дополнительных кодах. Введены следующие сокращения: РгПА – регистр порядка А; РгПВ – регистр порядка В; РгПС – регистр порядка С; РгМА – регистр мантиссы А; РгМВ – регистр мантиссы В; РгМС – регистр мантиссы С; РгБ – буферный регистр.

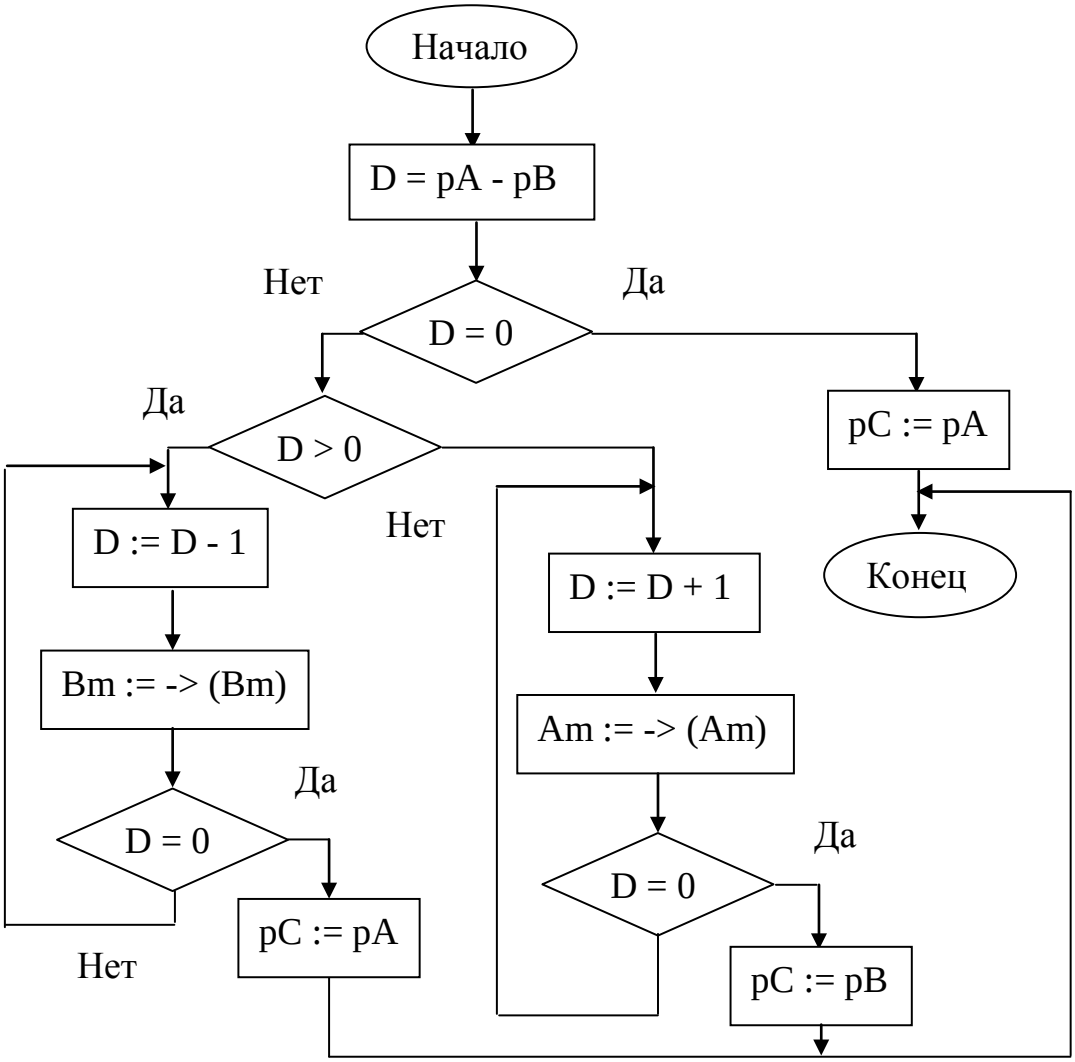

Рис. 8

1. Сравнение порядков
   РгН := (РгПА) – (РгПВ), если Z = 1, идти к 9, иначе к 2
2. Сохранение разности порядков
   РгБ := (РгН)
3. Проверка знака разности
   РгН := <- (РгБ), если ВСЛ = 1, идти к 11, иначе к 4
4. Денормализация Bm (пп. 4, 5, 6, 7, 8)
   РгН := (РгБ) – 1
5. РгБ := (РгН)
6. РгН := ->(РгМВ)
7. РгМВ := (РгН)
8. РгН := (РгБ), если Z = 1, идти к 9, иначе к 4

9. Формирование порядка С

РгН := (РгПА)

10. РгПС := (РгН). Конец

11. Денормализация Am (пп. 11, 12, 13, 14, 15)

РгН := (РгБ) + 1

12. РгБ := (РгН)

13. РгН := ->(РгМА)

14. РгМА := (РгН)

15. РгН := (РгБ), если Z = 1, идти к 16, иначе к 11

16. РгН := (РгПВ), идти к 10; Формирование порядка С.

Пусть имеется два нормализованных числа с различными положительными порядками: Am = 0.1011101 pA = 0.0000111 и Bm = 0.1101100 pB = 0.0000100. Рассмотрим числовой пример по выравниванию порядков согласно приведённой выше процедуры.

1. РгН := 0.0000111 – 0.0000100 = 0.0000011, Z = 0, идти к 2

2. РгБ := 0.0000011

3. РгН := 0.0000110, ВСЛ = 0, идти к 4

4. РгН := 0.0000010

5. РгБ := 0.0000010

6. РгН := 0.0110110

7. РгМВ := 0.0110110

8. РгН := 0.0000010, Z = 0, идти к 4

4. РгН := 0.0000001

5. РгБ := 0.0000001

6. РгН := 0.0011011

7. РгМВ := 0.0011011

8. РгН := 0.0000001, Z = 0, идти к 4

4. РгН := 0.0000000

5. РгБ := 0.0000000

6. РгН := 0.0001101

7. РгМВ := 0.0001101

8. РгН := 0.0000000, Z = 1, идти к 9

9. РгН := 0.0000111

10. РгПС := 0.0000111. Конец.

Таким образом, модуль pA был на 3 больше модуля pB, что привело к денормализации вправо Bm. Результирующий порядок pC приравнен к pA и равен 0.0000111.

Рассмотрим данный пример при наличии разных отрицательных порядков: pA = 1.1111001 и pB = 1.1111011 (коды дополнительные).

1. РгН := 1.1111001 – 1.1111011 = 1.1111110, Z = 0, идти к 2
2. РгБ := 1.1111110
3. РгН := 1.1111100, ВСЛ = 1, идти к 11

11. РгН := 1.1111111
12. РгБ := 1.1111111
13. РгН := 0.0101110
14. РгМА := 0.0101110
15. РгН := 1.1111111, Z = 0, идти к 11

11. РгН := 0.0000000
12. РгБ := 0.0000000
13. РгН := 0.0010111
14. РгМА := 0.0010111
15. РгН := 0.0000000, Z = 1, идти к 16
16. РгН := 1.1111011, идти к 10

10. РгПС := 1.1111011. Конец.

В рассмотренном примере pA = –7 был меньше pB = –5. В результате была денормализована мантисса Am на два разряда вправо. Результирующий порядок pC = – 5 (т. е. большему порядку).

На рис. 9 представлен алгоритм процедуры нормализации денормализованного вправо результата для отрицательных чисел в обратном и дополнительном кодах.

При нормализации денормализованных вправо положительных чисел выполняются левый сдвиг мантиссы и инкремент порядка до появления единицы в старшем разряде модуля мантиссы независимо от кодов этих чисел. Алгоритм подобен рассмотренному выше.

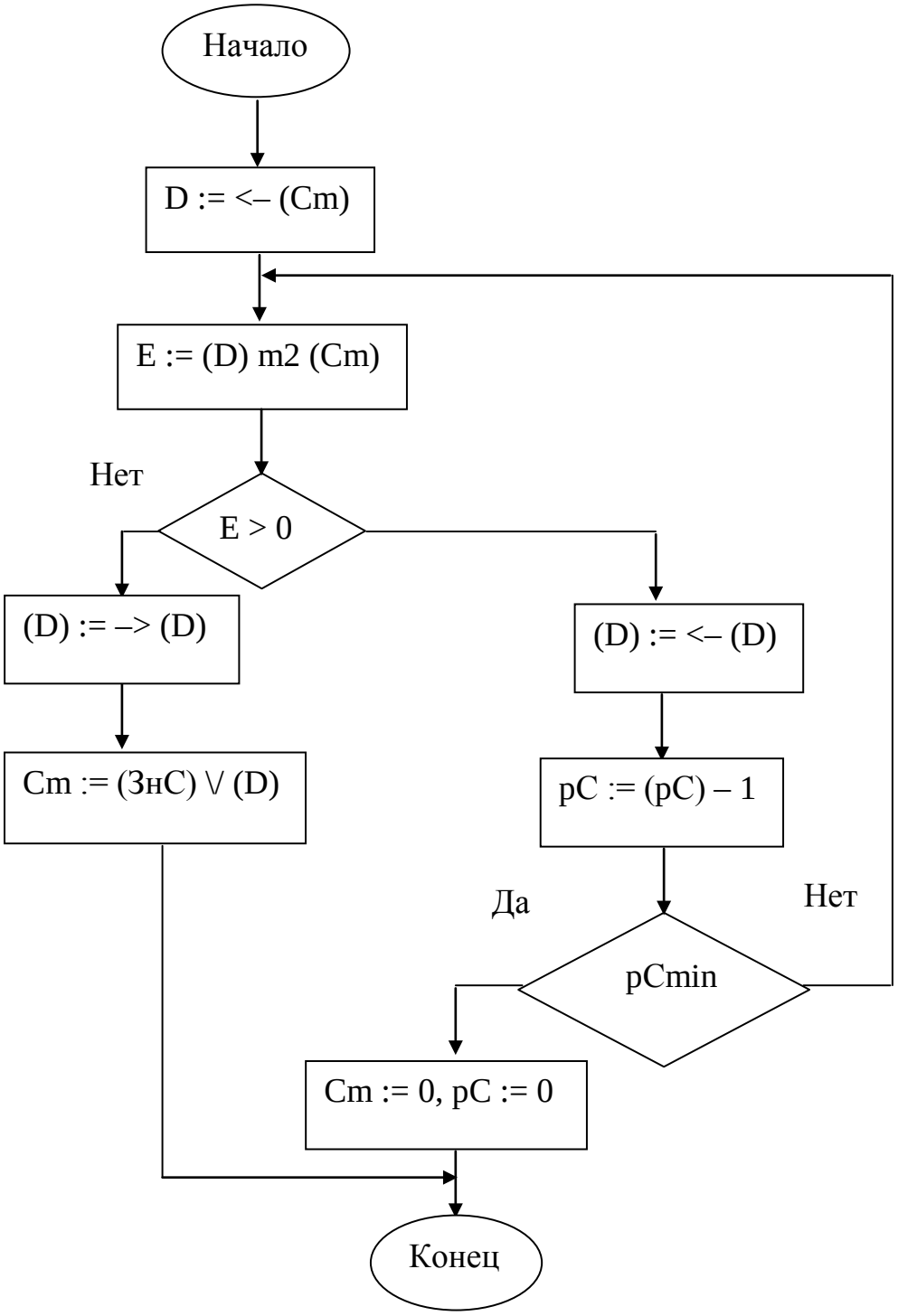

Рис. 9

Также может наблюдаться денормализация результата влево при переполнении результата суммирования равнознаковых мантисс. В этом случае нормализация осуществляется правым сдвигом мантиссы с восстановлением знака и инкрементом порядка результата. Если при этом порядок переполняется, то фиксируется переполнение результата и нормализация невозможна.

Выполнение длинной машинной операции в системе с фиксированной точкой

На рис. 10 приведён алгоритм выполнения операции умножения чисел, меньших единицы, в прямых кодах с анализом младших разрядов множителя и сдвигом суммы частичных произведений вправо с округлением. Округление позволяет оставаться результату в пределах разрядной сетки процессора и обеспечить точность в половину веса младшего разряда.

Выполнение операции умножения разворачивается во времени в соответствии со счётчиком циклов СЦ. Количество циклов на единицу меньше разрядности процессора при одном знаковом разряде (n–1 на рис. 10) либо на два при двух знаковых разрядах. В каждом цикле анализируется значение младшего бита МВ и осуществляется сдвиг МВ и МС вправо.

Введём в систему операций АЛУ операцию правого сдвига, формирующую выходной сигнал ВСП по состоянию младшего разряда. Под СЦ выделим регистр РгСЦ, а для накопления модуля произведения – регистр РгМС. Машинная реализация алгоритма умножения будет иметь следующее описание (считаем, что А, В, СЦ и константа 1.000…0 загружены в соответствующие регисты).

1. Проверка на 0 операнда А
   РгН := (РгА), если Z = 1, идти к 22, иначе к 2
2. Проверка на 0 операнда В
   РгН := (РгВ), если Z = 1, идти к 22, иначе к 3
3. Формирование знака С (пп. 3, 4, 5)
   РгН := (РгН) m2 (РгА)
4. РгН := (РгН) & (РгК)
5. РгС := (РгН)
6. Обнуление РгМС (пп. 6. 7)
   РгН := (РгМС) m2 (РгМС)
7. РгМС := (РгН)
8. Получение МА (пп. 8, 9, 10)
   РгН := !(РгК)
9. РгН := (РгА) & (РгН)
10. РгМА := (РгН)

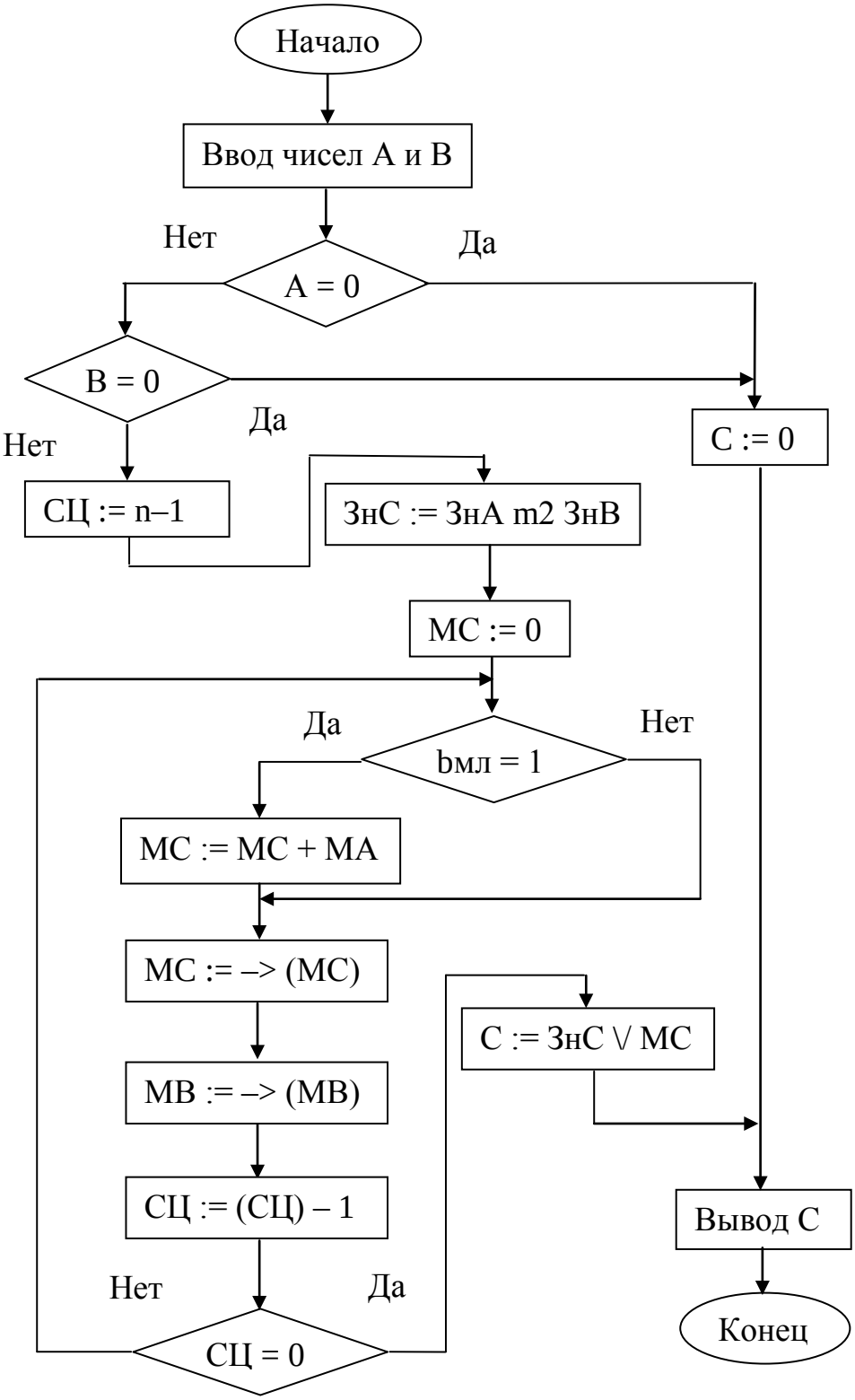

Рис. 10

11. Анализ младшего разряда В

    РгН := –>(РгВ), если ВСП = 1, идти к 19, иначе к 12

12. Сдвиг МС

    РгН := –>(РгМС)

13. Сохранение МС

    РгМС := (РгН)

14. Декремент СЦ

    РгН := (РгСЦ) – 1, если Z = 1, идти к 15, иначе к 18

15. Формирование результата (пп. 15, 16, 17)

    РгН := (РгС)

16. РгН := (РгМС) \/ (РгН)

17. РгС := (РгН). Конец.

18. Сохранение СЦ

    РгСЦ := (РгН), идти к 11

19. Накопление суммы частичных произведений (пп. 19, 20)

    РгН := (РгМС)

20. РгН := (РгН) + (РгМА)

21. РгМС := (РгН), идти к 12

22. Обнуление РгС (пп. 22, 23)

    РгН := (РгС) m2 (РгС), идти к 17.

Выполнение длинной машинной операции таблично-алгоритмическим методом

Таблично-алгоритмические вычисления предполагают использование таблиц косвенных функций, обращение к которым позволяет получить частичные результаты, из которых алгоритмическим путём формируют конечный результат.

Рассмотрим в качестве примера получение произведения двух чисел с фиксированной точкой с использованием таблицы квадратов чисел. Алгоритм вычисления произведения представлен на рис. 11. Считаем, что обращение к таблице осуществляется через регистр адреса таблицы РгАТ, а частичный результат с выхода таблицы, имеющий удвоенный формат, загружается в выходной регистр таблицы РгТ и в регистр-расширитель РгТР. Для работы с удвоенным форматом необходим и регистр-расширитель РгСР.

В качестве базы рассматриваемого метода используется формула

$$(A + B)^2 = A^2 + 2AB + B^2,$$

из которой следует

$$AB = ((A + B)^2 - A^2 - B^2)/2 .$$

Машинная операция по данному алгоритму выполняется следующим образом.

1. Проверка на 0 операнда А
   РгН := (РгА), если Z = 1, идти к 55, иначе к 2
2. Проверка на 0 операнда В
   РгН := (РгВ), если Z = 1, идти к 55, иначе к 3
3. Выделение модуля А (пп. 3, 4, 5)
   РгН := !(РгК)
4. РгН := (РгА) & (РгН)
5. РгМА := (РгН)
6. Выделение модуля В (пп. 6, 7, 8)
   РгН := !(РгК)
7. РгН := (РгВ) & (РгН)
8. РгМВ := (РгН)
9. Суммирование модулей
   РгН := (РгМА) + (РгН)
10. Обращение к таблице (пп. 10, 11)
    РгАТ := (РгН)
11. РгТР, РгТ := [(РгАТ)]
12. Сохранение квадрата суммы модулей (пп. 12, 13, 14, 15)
    РгН := (РгТ)
13. РгС : = (РгН)
14. РгН := (РгТР)
15. РгСР := (РгН)
16. Получение квадрата МА (пп. 16, 17, 18)
    РгН := (РгМА)
17. РгАТ := (РгН)
18. РгТР, РгТ := [(РгАТ)]

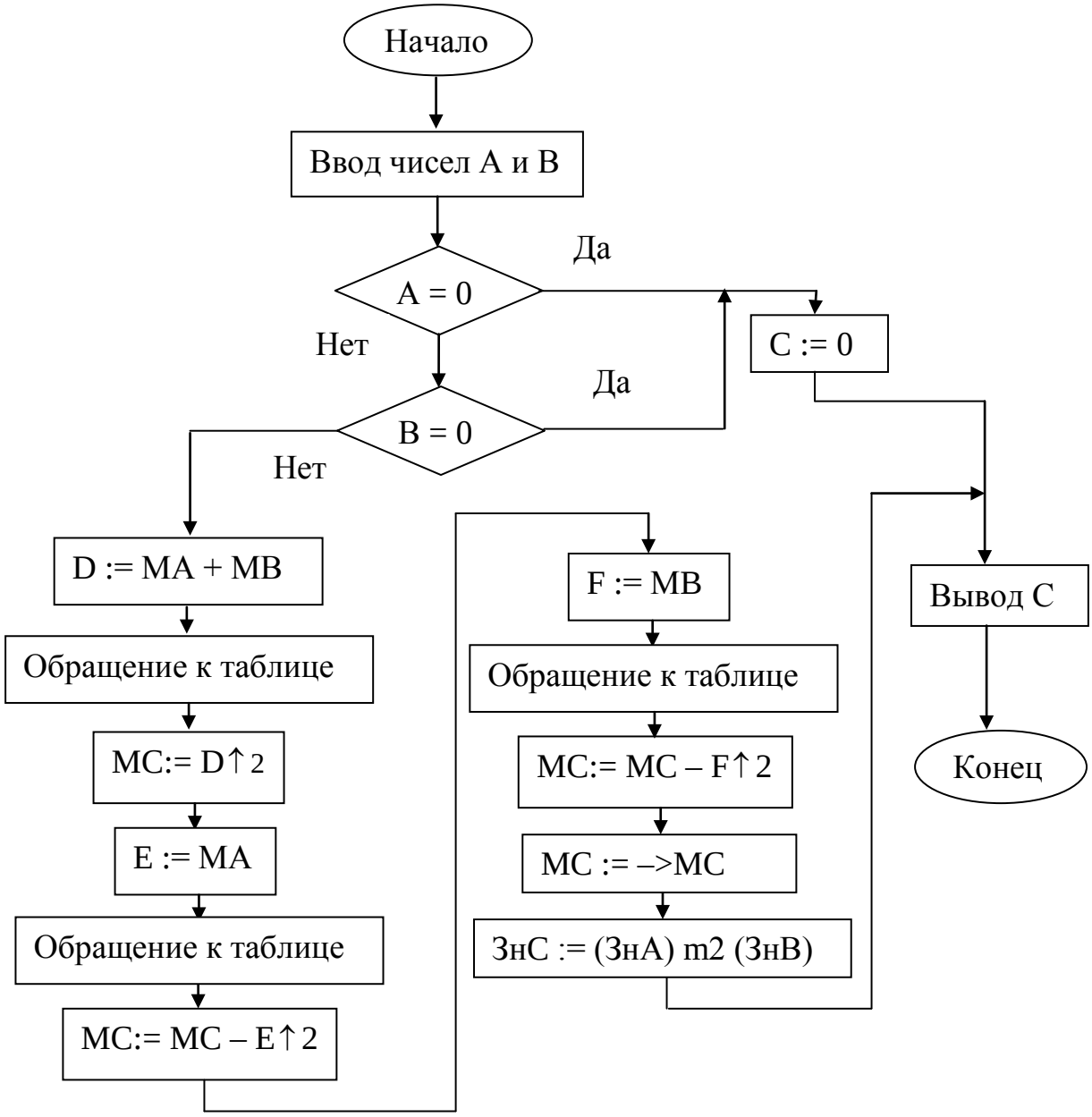

Рис. 11

19. Вычитание квадрата MA (пп. 19–29)
    РгН := (РгТ)
20. РгН := (РгС) – (РгН), если заём = 1, идти к 25, иначе к 21
21. РгС := (РгН)
22. РгН := (РгТР)
23. РгН := (РгСР) – (РгН)
24. РгСР := (РгН), идти к 30
25. РгС := (РгН)
26. РгН := (РгТР)
27. РгН := (РгСР) – (РгН)

28. РгН := (РгН) – 1
29. РгСР := (РгН)
30. Получение квадрата МВ (пп. 30, 31, 32)
    РгН := (РгМВ)
31. РгАТ := (РгН)
32. РгТР, РгТ := [(РгАТ)]
33. Вычитание квадрата МА (пп. 33 – 43)
    РгН := (РгТ)
34. РгН := (РгС) – (РгН), если заём = 1, идти к 25, иначе к 21
35. РгС := (РгН)
36. РгН := (РгТР)
37. РгН := (РгСР) – (РгН)
38. РгСР := (РгН), идти к 30
39. РгС := (РгН)
40. РгН := (РгТР)
41. РгН := (РгСР) – (РгН)
42. РгН := (РгН) – 1
43. РгСР := (РгН)
44. Сдвиг вправо модуля результата (пп. 44 - 50)
    РгН := –> (РгС)
45. РгС := (РгН)
46. РгН := –> (РгСР), если ВСП = 1, идти к 48, иначе к 47
47. РгСР := (РгН), идти к 50
48. РгСР := (РгН)
49. РгН := (РгС) \/ (РгК)
50. РгС := (РгН)
51. Формирование знака С (пп. 51, 52)
    РгН := (РгD) m2 (РгА)
52. РгН := (РгН) & (РгК)
53. РгН := (РгСР) \/ (РгН) ; Формирование результата
54. РгСР := (РгН). Конец.
55. РгН := (РгК) m2 (РгК); Обнуление результата
56. РгС := (РгН), идти к 54.

Библиографический список

1. Гуменюк, А. С. Информатика / А. С. Гуменюк, И. В. Потапов. – Омск : Изд-во ОмГМА, 2012.–175 с.

2. Потапов, В. И. Основы компьютерной арифметики и логики : учеб. пособие для вузов по направлению 230100 «Информатика и вычислительная техника» / В. И. Потапов, О. П. Шафеева, И. В. Червенчук; ОмГТУ. – Омск : Изд-во ОмГТУ, 2004. –172 с.

3. Потапов, В. И. Компьютерная арифметика и алгоритмическое моделирование арифметических операций : учеб. пособие / В. И. Потапов, О. П. Шафеева. – Омск : Изд-во ОмГТУ, 2005. – 96 с.

Перечень заданий по минимизации ЛФ с помощью карт Карно

1

	1		
	1		
1			1
1			1

2

	1	1	
	1	1	
1			1

3

1			1
1			1
1	1		1
1			1

4

1			1
1	1		1
1			1
1			1

5

1			1
	1	1	
	1	1	
			1

6

1			
	1	1	1
	1	1	
1			1

7

1			1
	1	1	1
	1	1	1
1			

8

1			1
	1	1	1
	1	1	1
1			1

9

1			1
1	1	1	1
1	1	1	1
1			1

10

1			
	1	1	1
	1	1	1
			1

11

1			1
1			1
	1		
	1		

12

1			1
	1	1	
	1	1	

13

	1	1	
	1	1	
		1	
	1	1	

14

	1	1	
		1	
	1	1	
	1	1	

15

	1	1	
1			1
1			1
1	1	1	

16

	1	1	1
1			
1			1
	1	1	

17

	1	1	
1			
1			
	1	1	1

18

	1	1	
1			
1			
	1	1	

19

		1	
		1	
1			1
1			1

20

1			1
1			1
1		1	1
1			1

21

1	1		
	1	1	
		1	1
1			1

22

1	1		
	1	1	
		1	1
			1

23

1			
	1	1	
		1	1
			1

24

1			1
	1	1	
		1	1
1			

25

1			1
	1	1	
		1	1
1			1

26

1			1
	1	1	
	1	1	1
1			1

27

1			1
	1	1	
1	1	1	1
1			1

28

1			1
	1	1	
1	1	1	
1			1

29

1			1
	1	1	1
1	1	1	
1			1

30

1			1
	1	1	1
	1	1	
1			1

31

1			1
	1	1	1
	1	1	
1			

32

1			1
	1	1	1
	1	1	
			1

33

1			1
	1	1	1
1	1	1	
			1

34

1			1
	1	1	1
1	1		
			1

35

1			1
	1	1	1
1			
			1

36

1			1
	1	1	1
1		1	
			1

37

1		1	1
	1	1	1
1		1	
			1

38

1		1	1
	1	1	1
1	1		
			1

39

1	1	1	1
	1	1	1
1	1		

40

1	1	1	1
	1	1	1
	1	1	

41

1	1	1	1
	1	1	1
		1	1

42

1	1	1	1
	1	1	1
		1	1
			1

43

1	1	1	
	1	1	1
		1	1
1			1

44

1	1		
	1	1	1
		1	1
1	1		1

45

1	1		
	1	1	
1		1	1
1	1		1

46

1	1		
	1		
1	1	1	1
1	1		1

47

1	1		
1	1		
1	1	1	
1	1		1

48

1			
1	1		
1	1	1	
1	1	1	1

49

		1	1
		1	1
		1	

50

		1	1
		1	1
		1	1

51

		1	1
		1	1
		1	
		1	

52

		1	1
		1	1
		1	1
		1	

53

		1	1
		1	1
		1	1
	1	1	

54

	1	1	
		1	1
		1	1
	1	1	

55

		1	1
	1	1	
		1	1
	1	1	

56

		1	1
		1	1
	1	1	
	1	1	

57

		1	1
		1	1
	1	1	
	1		1

58

		1	1
		1	1
	1		1
	1	1	

59

		1	1
		1	1
	1	1	
		1	1

60

	1	1	1
		1	1
	1	1	
		1	

Приложение 2

Перечень заданий к разделу «Моделирование арифметических операций»

Таблица

№ задания	Операция	Способ вычисления	Коды двоичных чисел	Представление чисел
1	2	3	4	5
1	Суммирование	ПР	ПК	ФТ (ПТ)
2	Суммирование	ПР	МПК	ФТ (ПТ)
3	Суммирование	ПР	ОК	ФТ (ПТ)
4	Суммирование	ПР	МОК	ФТ (ПТ)
5	Суммирование	ПР	ДК	ФТ (ПТ)
6	Суммирование	ПР	МДК	ФТ (ПТ)
7	Вычитание	ПР	ПК	ФТ (ПТ)
8	Вычитание	ПР	МПК	ФТ (ПТ)
9	Вычитание	ПР	ОК	ФТ (ПТ)
10	Вычитание	ПР	МОК	ФТ (ПТ)
11	Вычитание	ПР	ДК	ФТ (ПТ)
12	Вычитание	ПР	МДК	ФТ (ПТ)
13	Умножение младшими разрядами вперёд	ПР	ПК	ФТ (ПТ)
14	Умножение младшими разрядами вперёд	ПР	МПК	ФТ (ПТ)
15	Умножение младшими разрядами вперёд	ПР	ОК	ФТ (ПТ)
16	Умножение младшими разрядами вперёд	ПР	МОК	ФТ (ПТ)
17	Умножение младшими разрядами вперёд	ПР	ДК	ФТ (ПТ)
18	Умножение младшими разрядами вперёд	ПР	МДК	ФТ (ПТ)
19	Умножение старшими разрядами вперёд	ПР	ПК	ФТ (ПТ)
20	Умножение старшими разрядами вперёд	ПР	МПК	ФТ (ПТ)
21	Умножение старшими разрядами вперёд	ПР	ОК	ФТ (ПТ)
22	Умножение старшими разрядами вперёд	ПР	МОК	ФТ (ПТ)
23	Умножение старшими разрядами вперёд	ПР	ДК	ФТ (ПТ)
24	Умножение старшими разрядами вперёд	ПР	МДК	ФТ (ПТ)

Окончание табл.

25	Деление без восстановления остатка	ПР	ПК	ФТ (ПТ)
26	Деление без восстановления остатка	ПР	МПК	ФТ (ПТ)
27	Деление без восстановления остатка	ПР	ОК	ФТ (ПТ)
28	Деление без восстановления остатка	ПР	МОК	ФТ (ПТ)
29	Деление без восстановления остатка	ПР	ДК	ФТ (ПТ)
30	Деление без восстановления остатка	ПР	МДК	ФТ (ПТ)
31	Деление с восстановлением остатка	ПР	ПК	ФТ (ПТ)
32	Деление с восстановлением остатка	ПР	МПК	ФТ (ПТ)
33	Деление с восстановлением остатка	ПР	ОК	ФТ (ПТ)
34	Деление с восстановлением остатка	ПР	ОМК	ФТ (ПТ)
35	Деление с восстановлением остатка	ПР	ДК	ФТ (ПТ)
36	Деление с восстановлением остатка	ПР	МДК	ФТ (ПТ)
37	Умножение с анализом двух разрядов	ПРУ	ПК	ФТ (ПТ)
38	Умножение с анализом двух разрядов	ПРУ	МПК	ФТ (ПТ)
39	Умножение с анализом двух разрядов	ПРУ	ОК	ФТ (ПТ)
40	Умножение с анализом двух разрядов	ПРУ	ОМК	ФТ (ПТ)
41	Умножение с анализом двух разрядов	ПРУ	ДК	ФТ (ПТ)
42	Умножение с анализом двух разрядов	ПРУ	МДК	ФТ (ПТ)
43	Умножение с анализом трёх разрядов	ПРУ	ПК	ФТ (ПТ)
44	Умножение с анализом трёх разрядов	ПРУ	МПК	ФТ (ПТ)
45	Умножение с анализом трёх разрядов	ПРУ	ОК	ФТ (ПТ)
46	Умножение с анализом трёх разрядов	ПРУ	МОК	ФТ (ПТ)
47	Умножение с анализом трёх разрядов	ПРУ	ДК	ФТ (ПТ)
48	Умножение с анализом трёх разрядов	ПРУ	МДК	ФТ (ПТ)
49	Умножение аддитивным методом	ТА	ПК	ФТ (ПТ)
50	Умножение через таблицу квадратов	ТА	ПК	ФТ (ПТ)
51	Умножение через таблицу логарифмов	ТА	ПК	ФТ (ПТ)
52	Деление (умножение на обратное число)	ТА	ПК	ФТ (ПТ)
53	Деление через таблицу логарифмов	ТА	ПК	ФТ (ПТ)

Примечание: ПК – прямые коды; ОК – обратные коды; ДК – дополнительные коды; МПК – модифицированные прямые коды; МОК – модифицированные обратные коды; МДК – модифицированные дополнительные коды; ПР – программный метод; ПРУ – программный ускоренный метод; ТА – таблично-алгоритмический метод; ФТ – фиксированная точка; ПТ – плавающая точка.