Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Институт информационных технологий

Кафедра информационных систем и технологий

А. Г. Савенко, А. В. Матвеев

ОСНОВЫ КОМПЬЮТЕРНОЙ ТЕХНИКИ. ПРАКТИЧЕСКИЕ ЗАНЯТИЯ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия для специальности 1-40 01 01 «Программное обеспечение информационных технологий»

Минск БГУИР 2020

УДК 004.38(076)
ББК 32.973я73
С12

Р е ц е н з е н т ы:

кафедра экологических информационных систем учреждения образования «Международный государственный экологический институт имени А. Д. Сахарова» Белорусского государственного университета (протокол №8 от 19.03.2019 г.);

доцент кафедры программного обеспечения информационных систем и технологий Белорусского национального технического университета кандидат технических наук, доцент Н. Н. Гурский

Савенко, А. Г.

С12 Основы компьютерной техники. Практические занятия : учеб.-метод. пособие / А. Г. Савенко, А. В. Матвеев. – Минск : БГУИР, 2020. – 82 с. : ил.

ISBN 978-985-543-516-8.

Учебно-методическое пособие состоит из восьми практических занятий. Приводятся краткие теоретические сведения по теме занятия, методические указания и порядок решения задач, задачи для решения, а также контрольные вопросы и литература.

Практические занятия отражают материал по следующим темам: «Системы счисления и переход из одной системы счисления в другую», «Двоичная арифметика с положительными числами», «Арифметика с алгебраическими числами», «Арифметика с плавающей точкой», «Синтез логических схем», «Минимизация логических выражений», «Синтез цифровых автоматов», «Написание микропрограмм».

УДК 004.38(076)
ББК 32.973я73

ISBN 978-985-543-516-8

СОДЕРЖАНИЕ

Введение

В настоящее время во всем мире компьютерная техника внедряется во все сферы человеческой деятельности, и знания в этой области являются необходимыми для специалистов, разрабатывающих программное обеспечение, в том числе и программное обеспечения встроенных систем.

Целью практических занятий являются закрепление теоретического курса, приобретение навыков решения задач и активизация самостоятельной работы студентов.

Практические задания выполняются студентами на практических занятиях индивидуально. Каждое занятие рассчитано на два академических часа аудиторных занятий и два часа самостоятельной подготовки.

Учебно-методическое пособие включает восемь практических занятий по основным темам изучаемой дисциплины.

Первое практическое занятие содержит задачи по переходу из одной системы счисления в другую, используя преобразования с использованием весов разрядов методом деления/умножения на основание новой системы счисления, а также методом особого соотношения оснований систем счисления.

Второе практическое занятие содержит задачи двоичной арифметики с положительными двоичными и двоично-десятичными числами: операции сложения, вычитания, деления и умножения.

Третье практическое занятие содержит задачи арифметики с алгебраическими числами при использовании операций сложения/вычитания в дополнительном, обратном и модифицированном кодах.

Четвертое практическое занятие содержит задачи арифметики с плавающей точкой: операции сложения/вычитания, умножения/деления над числами, представленными в форме с плавающей точкой.

Пятое практическое занятие содержит задачи синтеза логических схем в заданном логическом базисе.

Шестое практическое занятие содержит задачи формирования минимального выражения с использованием минимизации с помощью карт Карно.

Седьмое практическое занятие содержит задачи синтеза логической схемы цифрового автомата, заданного таблицей переходов и таблицей выходов.

Восьмое практическое занятие содержит задачи построения микропрограммы для заданной граф-схемы алгоритма.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №1
ПЕРЕХОД ИЗ ОДНОЙ СИСТЕМЫ СЧИСЛЕНИЯ В ДРУГУЮ

Цель: приобрести практические навыки решения задач перехода из заданной системы счисления в искомую.

1.1 Краткие теоретические сведения

Арифметическая обработка чисел во многом определяется системами счисления, представляющими собой совокупность используемых цифр, и набором правил, позволяющих однозначно представлять числовую информацию. Десятичная система счисления, привычная людям, не является единственной. Электронная вычислительная техника использует также двоичную, восьмеричную и шестнадцатеричную системы счисления. Все эти системы являются позиционными.

Позиционная система счисления характеризуется тем, что «доля» некоторой цифры в количественной оценке записанного числа определяется не только видом цифры, но и местоположением (позицией) данной цифры в записи числа, т. е. каждая позиция (разряд) в записи числа имеет определенный вес. Количественная оценка записанного числа в такой системе счисления определяется как сумма произведений значения цифр, составляющих запись числа, умноженных на вес позиции, в которой располагается цифра.

Десятичная система счисления является также системой с равномерно распределенными весами, которые характеризуются тем, что соотношение весов двух любых соседних разрядов имеют для такой системы одинаковое значение. Это соотношение называется основанием системы счисления, которое в дальнейшем будем обозначать как q.

Общая запись числа в системе с равномерно распределенными весами имеет вид

$$N_q = A_n A_{n-1} \dots A_1 A_0. \quad (1.1)$$

Значение такого числа определяется следующим образом:

$$N_q = A_n \cdot q^n + A_{n-1} \cdot q^{n-1} + \cdots + A_1 \cdot q^1 + A_0 \cdot q^0, \quad (1.2)$$

где A_i – цифра записи числа ($0 \le A_i \le q - 1$);

q – основание системы счисления.

Запись числа N в виде (1.1) называется кодированной записью числа, а запись в форме (1.2) называется расширенной записью.

Для обозначения цифр в различных системах счисления используется обозначение соответствующих цифр десятичной системы счисления (от 0 до 9), а в системах счисления с основанием q, большим десяти, для цифр, превышающих значение девять, вводится дополнительное обозначение – латинские буквы, начиная с буквы *A* (для q = 16 это будут обозначения *A, B, C, D, E, F*).

Следовательно, запись одного и того же числа в различных системах счисления будет тем длиннее, чем меньше основание системы счисления.

1.2 Методические указания и примеры решения задач

Существование различных систем счисления предполагает необходимость перевода записи числа из одной системы счисления в другую. Существует ряд методов преобразований чисел в различных системах счисления. Среди них можно выделить два универсальных и один особый.

К универсальным методам относятся:

- метод преобразования с использованием весов разрядов в исходной и искомой записи числа;
- метод деления/умножения на новое основание.

Особым методом, применяемым при переводе из определенных систем счисления в определенную, является метод с использованием особого соотношения заданной и искомой систем счисления.

Метод преобразования с использованием весов разрядов

Данный универсальный метод предполагает использование расширенной записи числа (1.2) в некоторой системе счисления и, в зависимости от того, какая система счисления (исходная или искомая) является более привычной, имеет две разновидности.

1 Если более привычной является искомая система счисления, то на основании расширенной записи исходного числа подсчитываются значения ее отдельных разрядов в новой системе счисления. Далее полученные значения суммируются.

Пример

Необходимо перевести из двоичной системы счисления в десятичную число $N_2 = 1100110$.

По формуле (1.2) формируем расширенную запись числа:

$$N_2 = 1100110 = 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0.$$

Далее рассчитываем вес двоичных разрядов в десятичной системе счисления:

$$N_2 = 1100110 = 64 + 32 + 0 + 0 + 4 + 2 + 0 = 102_{10}.$$

Искомое число в десятичной системе счисления $N_{10} = 102$.

При преобразовании правильных дробей используется тот же алгоритм, но при расчете весов отдельных разрядов берем отрицательные степени основания счисления. Помимо этого, необходимо учитывать, что при преобразовании правильных дробей в общем случае результат получается неточный и перед началом преобразования необходимо подсчитать количество разрядов представления числа в новой системе счисления. Разрядность результата выбираем

таким образом, чтобы ошибка представления результата была бы не более половины единицы младшего разряда в исходной записи числа.

При преобразовании правильных дробей сначала находим предварительное значение представления заданного числа в новой системе счисления с количеством разрядов на единицу большим, чем расчетная разрядность представления числа в новой системе счисления. Дополнительный разряд в предварительном результате преобразования используем для округления, позволяющего с рассчитанным числом разрядов найти окончательный результат.

При переводе из двоичной в десятичную систему счисления берем соотношение, согласно которому один десятичный разряд соответствует точности представления четырехразрядным двоичным числом.

Пример

Необходимо перевести правильную двоичную дробь $N_2 = 0.111$ в правильную десятичную.

Перед началом преобразования определяется, что разрядность записи заданного числа в новой системе счисления должна быть равна единице, поэтому сначала находится предварительная запись заданного числа в новой системе счисления с двумя или более двоичными разрядами:

$$N_2 = 0.111 = 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 0{,}5 + 0{,}25 + 0{,}125 = 0{,}875.$$

После округления получаем $N_2 = 0.111 = 0{,}9_{10}$.

2 Если более привычной является исходная система счисления, то запись заданного числа в новой системе счисления определяется разряд за разрядом, начиная со старшего. Первым значащим разрядом будет являться разряд с максимальным возможным весом, но не превышающим значение преобразуемого числа. При этом, определив старший разряд с ненулевым значением, из исходного числа вычитается вес этого разряда, таким образом формируя остаток, который должен быть представлен еще не найденным младшим разрядом искомой записи числа в новой системе счисления. Далее, используя полученный остаток, аналогичным способом находится второй старший разряд записи числа в новой системе счисления, определяется новый остаток и т. д.

Пример

Необходимо перевести из десятичной системы счисления в двоичную число $N_{10} = 234$.

Первый (старший) разряд с весом $2^7 = 128$ будет иметь значение 1 в искомой двоичной записи числа. С помощью остальных (младших) разрядов искомой записи числа необходимо представить значение 106, т. к. 106 – это остаток, полученный как разность между числами 234 и 128.

Второй разряд с весом $2^6 = 64$ будет иметь в искомой двоичной записи числа значение 1. С помощью остальных (более младших) разрядов искомой записи числа необходимо представить значение 42, т. к. 42 – это остаток, полученный как разность между числами 106 и 64.

Третий разряд с весом $2^5 = 32$ будет иметь в искомой двоичной записи числа значение 1, а остаток, определяемый как разность между числами 42 и 32, будет равен 10.

Четвертый разряд с весом $2^4 = 16$ будет иметь в искомой двоичной записи числа значение 0, а остаток остается прежним – равным 10.

Пятый разряд с весом $2^3 = 8$ будет иметь в искомой двоичной записи числа значение 1, а остаток, определяемый как разность между числами 10 и 8, равен 2.

Шестой разряд с весом $2^2 = 4$ будет иметь в искомой двоичной записи числа значение 0, а остаток остается прежним – равным 2.

Седьмой разряд с весом $2^1 = 2$ будет иметь в искомой двоичной записи числа значение 1, а остаток, определяемый как разность между числами 2 и 2, равен 0.

Восьмой разряд с весом $2^0 = 1$ будет иметь в искомой двоичной записи числа значение 0, остаток равен 0.

Таким образом, записывая полученные значения последовательно со старшего разряда, получаем

$$N_{10} = 234 = 11101010_2.$$

Аналогично данный способ применяется при переводе правильных дробей, только с отрицательными степенями основания.

При переводе из десятичной в двоичную систему счисления берем соотношение, согласно которому четыре двоичных разряда соответствуют точности представления одного десятичного разряда.

Пример

Необходимо перевести правильную десятичную дробь $N_{10} = 0{,}7$ в правильную дробь в двоичной системе счисления.

Предварительный результат находим с точностью до пяти двоичных разрядов, причем пятый разряд используем только для округления при переходе к четырехразрядному окончательному результату.

Первый (старший) разряд с весом $2^{-1} = 0{,}5$ искомой двоичной записи числа будет иметь значение 1. С помощью остальных (младших) разрядов искомой записи числа необходимо представить значение 0,2 (0,2 – остаток, полученный как разность чисел 0,7 и 0,5).

Второй разряд с весом $2^{-2} = 0{,}25$ в искомой двоичной записи числа будет иметь значение 0. Остаток остается прежним.

Третий разряд с весом $2^{-3} = 0{,}13$ в искомой двоичной записи числа будет иметь значение 1. С помощью остальных (более младших) разрядов искомой записи числа необходимо представить значение 0,07 (0,07 – остаток, полученный как разность чисел 0,20 и 0,13).

Четвертый разряд с весом $2^{-4} = 0{,}06$ в искомой двоичной записи числа будет иметь значение 1, а остаток – 0,01 (0,01 – остаток, полученный как разность чисел 0,07 и 0,06).

Пятый разряд с весом 2^{-5} = 0,03 искомой двоичной записи числа будет иметь значение 0.

Таким образом, записывая полученные значения последовательно со старшего разряда, получаем $0{,}7_{10} = 0.10110_2$.

После округления пятого младшего разряда имеем $0{,}7_{10} = 0.1011_2$.

Метод деления/умножения на новое основание

Данный универсальный метод также предполагает использование расширенной записи числа (1.2) и имеет две разновидности: для целых и дробных чисел.

1 Преобразование целых чисел

Задачу представления числа N, заданного в системе q_1, в системе счисления с основанием q_2 можно рассматривать как задачу поисков коэффициентов полинома, представляющего собой расширенную запись числа N в системе счисления q_2:

$$N_{q1} = A_0 + A_1 \cdot q_2^1 + A_2 \cdot q_2^2 + \cdots + A_{n-1} \cdot q_2^{n-1} + A_n \cdot q_2^n = N_{q2}. \qquad (1.3)$$

Преобразуем выражение (1.3), вынеся общий знаменатель за скобку (рисунок 1.1):

$$N_{q1} = N_{q2} = A_0 + q_2(A_1 + q_2(\, A_2 + q_2(A_3 + \cdots + q_2(A_{n-1} + q_2(A_n) \ldots))).$$

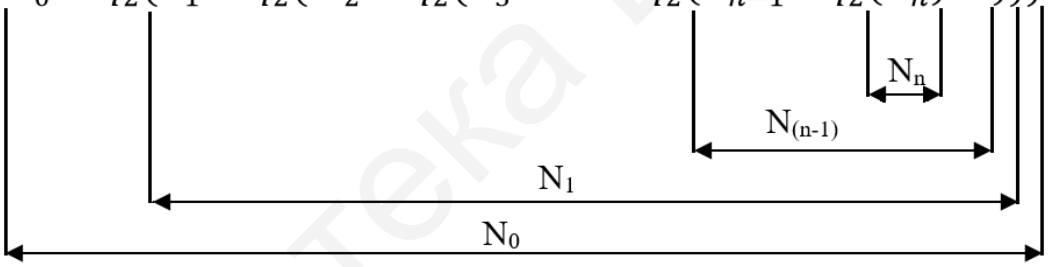


Рисунок 1.1 – Преобразованное выражение

Обозначим все преобразованное выражение как N_0, выражение в первой скобке – N_1, выражение во второй скобке – N_2 и т. д., выражение в $(n-1)$-й скобке – $N_{(n-1)}$, выражение в n-й скобке – N_n. Теперь, принимая во внимание выражение (1.3), можно утверждать, что при делении N_{q1} на q_2 будут получены целая часть частного (обозначим ее *int* (N_{q1}/q_2)) и остаток (обозначим его *rest* (N_{q1}/q_2)), равный A_0. Тогда остальные скобки будут иметь следующий вид:

$\frac{N_1}{q_2}$: целая часть *int* (N_1/q_2), равная N_2, и остаток *rest* (N_1/q_2), равный A_1;

$\frac{N_2}{q_2}$: целая часть *int* (N_2/q_2), равная N_3, и остаток *rest* (N_2/q_2), равный A_2;

…

$\frac{N_{(n-2)}}{q_2}$: целая часть *int* $(N_{(n-2)}/q_2)$, равная $N_{(n-1)}$, и остаток *rest* $(N_{(n-2)}/q_2)$, равный $A_{(n-2)}$;

$\frac{N_{(n-1)}}{q_2}$: целая часть *int* ($N_{(n-1)}/q_2$), равная $N_n = A_n$, и остаток *rest* ($N_{(n-1)}/q_2$), равный $A_{(n-1)}$.

При этом $N_n = A_n$ ($N_n < q_2$).

Отсюда следует правило формирования коэффициентов полинома (1.3), которые и являются разрядами записи заданного числа N в системе счисления с основанием q_2:

- необходимо разделить исходное число N_{q1} на новое основание q_2, при этом получив целое частное и остаток;

- полученный остаток снова необходимо разделить на q_2; процесс деления продолжается до тех пор, пока частное будет не меньше нового основания q_2. Если очередное сформированное частное будет меньше чем q_2, то процесс формирования записи заданного числа в новой системе с основанием q_2 считается законченным, а в качестве искомых разрядов новой записи числа используются результаты выполненных операций деления следующим образом: в качестве старшего разряда берется значение последнего частного, для остальных разрядов используются значения остатков в порядке, обратном порядку их получения.

Пример

Перевести десятичное число $N_{10} = 432$ в двоичную систему счисления методом деления на новое основание.

Согласно алгоритму сначала делим исходное число N_{10}, а затем и получаемые частные делим на значение нового основания $q = 2$ до получения частного со значением, меньшим чем 2:

$\frac{432}{2}$: *int* (432/2) = 216 и *rest* (432/2) = 0;

$\frac{216}{2}$: *int* (216/2) = 108 и *rest* (216/2) = 0;

$\frac{108}{2}$: *int* (108/2) = 54 и *rest* (108/2) = 0;

$\frac{54}{2}$: *int* (54/2) = 27 и *rest* (54/2) = 0;

$\frac{27}{2}$: *int* (27/2) = 13 и *rest* (27/2) = 1;

$\frac{13}{2}$: *int* (13/2) = 6 и *rest* (13/2) = 1;

$\frac{6}{2}$: *int* (6/2) = 3 и *rest* (6/2) = 0;

$\frac{3}{2}$: *int* (3/2) = 1 и *rest* (3/2) = 1.

Старшим разрядом записи числа N в двоичной системе счисления будет значение последнего частного, а далее записываем остатки в обратном порядке.

Таким образом, $N_{10} = 432 = 110110000_2$.

2 Преобразование дробных чисел

Задачу представления дробного числа M_{q1} в новой системе счисления с основанием q_2 можно рассматривать как поиск коэффициентов полинома, представляющего собой расширенную запись числа M в системе счисления q_2:

$$M_{q1} = B_1 \cdot q_2^{-1} + B_2 \cdot q_2^{-2} + \cdots + B_{n-1} \cdot q_2^{-(n-1)} + B_n \cdot q_2^{-n} = M_{q2}. \quad (1.4)$$

Преобразуем выражение (1.4), вынеся общий знаменатель за скобку (рисунок 1.2):

$$M_{q1} = M_{q2} = q_2^{-1}(B_1 + q_2^{-1}(B_2 + q_2^{-1}(B_3 + \cdots + q_2^{-1}(B_{n-1} + q_2^{-1}(B_n))\ldots))).$$

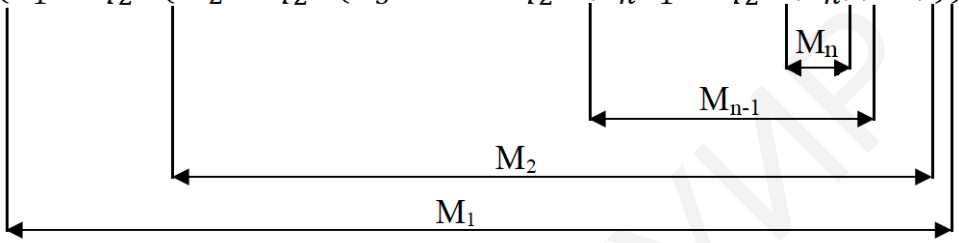


Рисунок 1.2 – Преобразованное выражение

Обозначим выражение в первой скобке как M_1, выражение во второй скобке – M_2 и т. д., выражение в $(n-1)$-й скобке – M_{n-1} и выражение в n-й скобке – M_n.

Так как число M_{q1} – правильная дробь, то при умножении M_{q1} на основание q_2 будет получено произведение, в общем случае состоящее из целой части int $(M_{q1} \cdot q_2)$ и дробной части DF $(M_{q1} \cdot q_2)$. Тогда остальные скобки будут иметь следующий вид:

$$M_1 \cdot q_2 = (int\ (M_1 \cdot q_2) = B_2) + (DF\ (M_1 \cdot q_2) = M_2);$$
$$M_2 \cdot q_2 = (int\ (M_2 \cdot q_2) = B_3) + (DF\ (M_2 \cdot q_2) = M_3);$$
$$\ldots$$
$$M_{n-1} \cdot q_2 = (int\ (M_{n-1} \cdot q_2) = B_n) + (DF\ (M_{n-1} \cdot q_2) = M_n);$$
$$M_n \cdot q_2 = (int\ (M_n \cdot q_2) = B_{n+1}) + (DF\ (M_n \cdot q_2) = M_{n+1}).$$

Отсюда следует правило формирования коэффициентов полинома, которые и являются разрядами записи заданного числа M в системе счисления с основанием q_2:

- определяется количество разрядов n в записи числа M_{q2} в новой системе счисления. Разрядность результата выбирается таким образом, чтобы ошибка представления результата была бы не более половины единицы младшего разряда в исходной записи числа;

- исходное число M_{q1} умножается на q_2, при этом будет получено смешанное число;

- дробная часть полученного произведения снова умножается на q_2 и т. д. Процесс умножения повторяется n раз. В качестве искомых разрядов новой записи числа используются результаты выполненных операций умножения следующим образом: в качестве первого старшего разряда искомой записи числа в системе счисления с новым основанием берется значение целой части первого

произведения, в качестве второго старшего разряда искомой записи числа в системе счисления с новым основанием берется значение целой части второго произведения и т. д.

Пример

Перевести десятичное число $M_{10} = 0{,}7$ в двоичную систему счисления методом умножения на новое основание.

Сначала необходимо определить количество разрядов числа M в двоичной системе счисления. Так как исходная запись числа содержит один десятичный разряд, то запись данного числа в двоичной системе должна содержать четыре разряда. Учитывая округление, находим предварительный двоичный эквивалент с пятью разрядами.

Умножаем исходное число M_{10}, а затем дробные части последовательно получаемых произведений на новое основание $q = 2$:

$0{,}7 \cdot 2 = 1{,}4$ (*int* $(0{,}7 \cdot 2) = 1$ и *DF* $(0{,}7 \cdot 2) = 0{,}4$);
$0{,}4 \cdot 2 = 0{,}8$ (*int* $(0{,}4 \cdot 2) = 0$ и *DF* $(0{,}4 \cdot 2) = 0{,}8$);
$0{,}8 \cdot 2 = 1{,}6$ (*int* $(0{,}8 \cdot 2) = 1$ и *DF* $(0{,}8 \cdot 2) = 0{,}6$);
$0{,}6 \cdot 2 = 1{,}2$ (*int* $(0{,}6 \cdot 2) = 1$ и *DF* $(0{,}6 \cdot 2) = 0{,}2$);
$0{,}2 \cdot 2 = 0{,}4$ (*int* $(0{,}2 \cdot 2) = 0$ и *DF* $(0{,}2 \cdot 2) = 0{,}4$).

Таким образом, $M_{10} = 0{,}7 = 0.10110_2$.

После округления имеем $M_{10} = 0{,}7_{10} = 0.1011_2$.

Метод преобразования с использованием особого соотношения оснований заданной и искомой систем счисления

Данный метод не является универсальным и применим тогда, когда исходное q_1 и новое q_2 основания могут быть связаны через целую степень, т. е. когда выполняется одно из двух условий:

$$q_2^m = q_1, \tag{1.5}$$

применимое для перехода из системы с большим основанием в систему с меньшим основанием;

$$q_1^m = q_2, \tag{1.6}$$

применимое для перехода из системы с меньшим основанием в систему с большим основанием.

Если выполняется условие (1.5), тогда запись числа $N_{q1} = a_n a_{n-1} \dots a_1 a_0$ в системе с новым основанием q_2 определяется следующим образом:

- каждому разряду a_i исходной записи числа ставится в соответствие его m-разрядный эквивалент в системе счисления с основанием q_2;

- конечная запись всего заданного числа формируется за счет объединения всех полученных m-разрядных групп.

Пример

Перевести восьмеричное число $N_8 = 47601.62$ в двоичную систему счисления.

Основания исходной и новой систем счисления можно выразить через целую степень следующим образом: $2^3 = 8$.

Согласно алгоритму ставим в соответствие каждой цифре исходной записи восьмеричного числа трехразрядный двоичный эквивалент (триады) (таблица 1.1).

Таблица 1.1 – Соответствие двоичных триад цифрам восьмеричного числа

Представление цифр восьмеричного числа	4	7	6	0	1	.	6	2
Соответствие триад в двоичной системе счисления	100	111	110	000	001	.	110	010

Формируем окончательный результат посредством объединения полученных триад двоичного представления восьмеричных цифр в единый двоичный эквивалент с учетом точки, отделяющей целую и дробную части:

$$47601.62_8 = 100111110000001.110010_2.$$

Если выполняется условие (1.6), тогда запись числа $N_{q1} = a_n a_{n-1} \dots a_1 a_0$ в системе с новым основанием q_2 определяется следующим образом:

- исходная запись числа разбивается на группы по m разрядов, двигаясь от точки вправо и влево (недостающие разряды в крайних группах (слева и справа) дополняются нулями);

- каждой полученной группе ставится в соответствие цифра новой системы счисления;

- искомая запись заданного числа в новой системе счисления образуется из цифр, соответствующих группам, на которые была разбита исходная запись.

Пример

Перевести двоичное число N_2 = 1101111100.1110100 в шестнадцатеричную систему счисления.

Основания исходной и новой систем счисления можно выразить через целую степень следующим образом: $2^4 = 16$.

Разбиваем исходную запись числа на группы по четыре разряда вправо и влево от точки, в крайних левой и правой группах недостающие разряды заполняем нулями и каждой полученной группе из четырех разрядов ставим в соответствие цифру шестнадцатеричной системы счисления (таблица 1.2).

Таблица 1.2 – Соответствие двоичных тетрад цифрам шестнадцатеричного числа

Представление тетрад в двоичной системе счисления	**00**11	0111	1100	.	1110	100**0**
Соответствие цифр шестнадцатеричного числа	3	7	C	.	E	8

В таблице 1.2 полужирным подчеркнутым шрифтом обозначены недостающие разряды в крайней правой и крайней левой от точки тетрадах (группа из четырех разрядов), которые заполняются нулями.

Формируем окончательный результат посредством объединения полученных цифр в единый шестнадцатеричный эквивалент с учетом точки, отделяющей целую и дробную части:

$$1101111100.1110100_2 = 37C.E8_{16}.$$

Следует также отметить, что применять метод преобразования с использованием особого соотношения оснований иногда бывает целесообразно, даже

если основания исходной q_1 и новой q_2 систем счисления не могут быть связаны через целую степень напрямую. Если существует промежуточная система, которая может быть связана с заданной и искомой системами одновременно условиями (1.5) и (1.6), то можно сперва найти эквивалент заданного числа в промежуточной системе счисления, а затем из промежуточной системы перевести его в искомую. В ряде случаев такие преобразования выполняются быстрее, чем при использовании рассмотренных выше универсальных методов.

Пример

Перевести восьмеричное число $N_8 = 27514.73_8$ в шестнадцатеричную систему счисления.

Согласно условиям (1.5) и (1.6) промежуточной системой счисления в данном случае может являться двоичная система счисления: $2^3 = 8$ и $2^4 = 16$.

Сначала необходимо перевести число из заданной восьмеричной системы счисления в промежуточную двоичную (таблица 1.3).

Таблица 1.3 – Соответствие двоичных триад цифрам восьмеричного числа

Представление цифр восьмеричного числа	2	7	5	1	4	.	7	3
Соответствие триад в двоичной системе счисления	010	111	101	001	100	.	111	011

Таким образом, имеем $27514.73_8 = 010111101001100.111011_2$.

Далее переводим число из промежуточной двоичной системы счисления в искомую шестнадцатеричную, разбив в обе стороны от точки запись двоичного числа на тетрады и дополнив крайнюю левую и крайнюю правую тетрады недостающими нулевыми разрядами при необходимости (полужирный подчеркнутый шрифт) (таблица 1.4).

Таблица 1.4 – Соответствие двоичных тетрад цифрам шестнадцатеричного числа

Представление тетрад в двоичной системе	**0**010	1111	0100	1100	.	1110	11**00**
Соответствие цифр шестнадцатеричного числа	2	F	4	C	.	E	C

В итоге, $27514.73_8 = 010111101001100.111011_2 = 2F4C.EC_{16}$.

1.3 Задачи для решения

Задание 1

Сформировать расширенную запись двоичных чисел:

а) 1011010_2; г) 101010101_2; ж) 1100110_2;

б) 100011011_2; д) 1000110_2; з) 1000101_2;

в) 111110100_2; е) 1110111_2; и) 10101011_2.

Задание 2

Перевести заданные целые числа из двоичной системы счисления в десятичную, применяя метод преобразования с использованием весов разрядов:

а) 110010_2; в) 111011_2; д) 111000_2;
б) 10101101_2; г) 10010111_2; е) 10111010_2.

Задание 3

Перевести заданные дробные числа из двоичной системы счисления в десятичную, применяя метод преобразования с использованием весов разрядов:

а) 0.1010_2; в) 0.1111_2; д) 0.1001_2;
б) 0.010101_2; г) 0.001011_2; е) 0.011100_2.

Задание 4

Перевести заданные целые и дробные числа из десятичной системы счисления в двоичную, применяя метод преобразования с использованием весов разрядов:

а) 134_{10}; д) 193_{10}; и) 177_{10};
б) 406_{10}; е) 444_{10}; к) 479_{10};
в) $0{,}82_{10}$; ж) $0{,}68_{10}$; л) $0{,}72_{10}$;
г) $0{,}45_{10}$; з) $0{,}33_{10}$; м) $0{,}24_{10}$.

Задание 5

Перевести заданные целые и дробные числа из десятичной системы счисления в искомую, используя метод деления/умножения на новое основание:

а) $N_{10} = 146$; $N_2 = \ldots$; д) $N_{10} = 125$; $N_2 = \ldots$; и) $N_{10} = 199$; $N_2 = \ldots$;
б) $N_{10} = 315$; $N_8 = \ldots$; е) $N_{10} = 350$; $N_8 = \ldots$; к) $N_{10} = 399$; $N_8 = \ldots$;
в) $N_{10} = 0{,}48$; $N_2 = \ldots$; ж) $N_{10} = 0{,}54$; $N_2 = \ldots$; л) $N_{10} = 0{,}69$; $N_2 = \ldots$;
г) $N_{10} = 0{,}77$; $N_8 = \ldots$; з) $N_{10} = 0{,}83$; $N_8 = \ldots$; м) $N_{10} = 0{,}89$; $N_8 = \ldots$.

Задание 6

Перевести заданные целые и дробные числа из двоичной системы счисления в восьмеричную, используя метод особого соотношения оснований систем счисления:

а) 1100110_2; д) 1010101_2; и) 1110100_2;
б) 101100001_2; е) 110101101_2; к) 1001111011_2;
в) 10110.1001_2; ж) 11000.0011_2; л) 10001.1111_2;
г) 1101.01011_2; з) 1110.01101_2; м) 10111.111011_2.

Задание 7

Перевести заданные целые и дробные числа из шестнадцатеричной системы счисления в двоичную, используя метод особого соотношения оснований систем счисления:

а) $2AF479_{16}$; г) $A52C83_{16}$; ж) $4AAC5F_{16}$;
б) $4C6B5.1E_{16}$; д) $2DA04.4C_{16}$; з) $A62B1.E8_{16}$;
в) $3AB7.C2_{16}$; е) $15DA.5C9_{16}$; и) $5E6B.C0A_{16}$.

Задание 8

Перевести заданные целые и дробные числа из восьмеричной системы счисления в двоичную, а затем из двоичной в шестнадцатеричную, используя метод особого соотношения оснований систем счисления:

а) 53472_8; в) 32570_8; д) 16735_8;
б) 25601.421_8; г) 63042.741_8; е) 42065.172_8.

Задание 9

Перевести заданные целые числа из десятичной системы счисления в восьмеричную методом деления на новое основание, а затем в двоичную методом особого соотношения оснований систем счисления:

а) 658362_{10}; в) 350761_{10}; д) 257108_{10};
б) 139025_{10}; г) 273401_{10}; е) 123456_{10}.

Задание 10

Перевести заданные целые и дробные числа из заданной системы счисления в искомую:

а) $N_7 = 146$; $N_2 = \ldots$; д) $N_6 = 125$; $N_2 = \ldots$; и) $N_9 = 78$; $N_2 = \ldots$;
б) $N_8 = 1542$; $N_5 = \ldots$; е) $N_8 = 1242$; $N_4 = \ldots$; к) $N_8 = 2715$; $N_6 = \ldots$;
в) $N_{10} = 0{,}17$; $N_2 = \ldots$; ж) $N_{10} = 0{,}54$; $N_2 = \ldots$; л) $N_{10} = 0{,}75$; $N_2 = \ldots$;
г) $N_{16} = 0.4A8$; $N_{10} = \ldots$; з) $N_{16} = 0.B41$; $N_{10} = \ldots$; м) $N_{16} = 0.1C8$; $N_{10} = \ldots$.

1.4 Контрольные вопросы

1 Какие методы перевода из заданной системы счисления в искомую являются универсальными и почему?

2 Чем отличается перевод целых и перевод дробных чисел из заданной системы счисления в искомую методом с использованием весов разрядов?

3 Как формируется запись числа в искомой системе счисления при преобразовании методом деления/умножения на новое основание?

4 Какие условия должны выполняться для использования метода особого соотношения оснований систем счисления?

5 В каком случае целесообразно использовать метод особого соотношения оснований систем счисления, если не выполняются необходимые для этого условия?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №2
ДВОИЧНАЯ АРИФМЕТИКА С ПОЛОЖИТЕЛЬНЫМИ ЧИСЛАМИ

Цель: приобрести практические навыки решения арифметических задач с двоичными положительными числами.

2.1 Краткие теоретические сведения

В настоящее время практически не осталось аналоговой техники. Все устройства – от бытовых приборов до космических аппаратов – являются цифровыми. Как известно, цифровые устройства, в частности компьютерная техника, при обработке информации использует в основном двоичную систему счисления.

Именно поэтому в большей своей части компьютерная арифметика является двоичной арифметикой. Этому есть объяснение. Во-первых, алгоритмы арифметических операций двоичной арифметики (т. е. арифметики, использующей двоичную позиционную систему) очень просты и являются в определенном смысле простейшими среди подобных алгоритмов для всех позиционных числовых систем. Во-вторых, дискретные (не аналоговые) электронные схемы, как самые современные, так и использовавшиеся много лет назад, имеют в определенном смысле двоичную природу и легко описываются на языке алгебры логики.

В двоичной системе счисления арифметические операции (сложение, вычитание, умножение и деление) выполняются по тем же правилам, что и в привычной нам десятичной системе счисления, т. к. они обе являются позиционными (наряду с восьмеричной, шестнадцатеричной и др.).

Кроме того, помимо двоичной системы счисления со своими правилами двоичной арифметики также существует двоично-десятичная система счисления. В отличие от остальных арифметик позиционных систем счисления двоично-десятичная арифметика имеет свои особенности, выраженные в необходимости корректировки результата арифметических операций путем добавления в тетраду двоичной «шестерки» (в двоичном коде – 0110) при переносе в старший разряд или при заеме из старшего разряда, а также при превышении в тетраде значения, большего девяти.

2.2 Методические указания и примеры решения задач

Как было отмечено ранее, арифметические операции в двоичной системе счисления с положительными числами выполняются по тем же правилам, что и в десятичной системе счисления. Поскольку в двоичной системе счисления используются всего две цифры (нуль и единица), то нагляднее всего можно отобразить эти правила в виде таблицы, где будет всего четыре возможные комбинации результата выполнения операций сложения, вычитания и умножения.

Сложение положительных двоичных чисел

В общем виде правила сложения двоичных положительных чисел выглядят следующим образом:

- два двоичных положительных числа складываются в столбик поразрядно, начиная с младшего разряда;
- при сложении нулевого разряда с нулевым получается сумма, равная нулю (также нулевой разряд);
- при сложении нулевого разряда с единичным получается сумма, равная единице (единичный разряд);
- при сложении единичного разряда с единичным получается сумма, равная нулю с переносом единицы в старший разряд;
- при переносе единицы из младшего в старший разряд сначала складываются значения соответствующих разрядов двух двоичных чисел по вышеописанным правилам, затем к полученной поразрядной сумме прибавляется перенесенная единица, также по вышеописанным правилам.

Наглядно данные правила изображены в таблице 2.1.

Таблица 2.1 – Правила сложения двоичных положительных чисел

		Первое слагаемое	
		0	1
Второе слагаемое	0	**0**	**1**
	1	**1**	**0***

Примечание – * – перенос единицы в старший разряд.

Пример

Необходимо выполнить сложение двух положительных двоичных чисел 10110011101_2 и 11101000110_2.

Выполняем сложение поразрядно в столбик, начиная с младшего разряда:

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

При сложении в третьем разряде происходит переполнение, и единица переходит в четвертый разряд, где также вызывает переполнение. Аналогичная ситуация происходит в пятом, шестом, девятом, десятом и одиннадцатом разрядах. Лишняя единица в одиннадцатом разряде переходит в двенадцатый. Таким образом, конечное значение (сумма) будет иметь уже не одиннадцать, а двенадцать разрядов.

В общем случае при формировании значения в текущем разряде результата приходится дважды применять приведенную таблицу 2.1: первый раз – при сложении соответствующих разрядов операндов, формируя так называемую поразрядную сумму, и второй – для сложения разряда сформированной пораз-

рядной суммы и переноса, пришедшего из ближайшего младшего разряда, если таковое произошло.

Вычитание положительных двоичных чисел

В общем виде правила вычитания двоичных положительных чисел выглядят следующим образом:

- два двоичных положительных числа вычитаются в столбик поразрядно, начиная с младшего разряда;
- при вычитании из нулевого разряда нулевого же разряда получается разность, равная нулю (также нулевой разряд);
- при вычитании из единичного разряда нулевого разряда получается разность, равная единице;
- при вычитании из единичного разряда единичного разряда получается разность, равная нулю (также нулевой разряд);
- при вычитании из нулевого разряда единичного разряда получается разность, равная единице, при этом происходит заем единицы из старшего разряда;
- при вычитании из нулевого разряда, из которого уже происходил заем единицы, единичного разряда получается разность, равная нулю, и при этом также происходит заем единицы из старшего разряда;
- при вычитании из нулевого разряда, из которого уже происходил заем единицы, нулевого разряда получается разность, равная единице, и при этом также происходит заем единицы из старшего разряда.

Наглядно данные правила изображены в таблице 2.2.

Таблица 2.2 – Правила вычитания двоичных положительных чисел

		Уменьшаемое		
		0	0, из которого взята единица в младший разряд	1
Вычитаемое	0	**0**	**1***	**1**
	1	**1***	**0***	**0**

Примечание – * – заем единицы из старшего разряда.

Пример

Найти разность двух положительных двоичных чисел 1100101_2 и 101110_2.

Выполняем вычитание поразрядно в столбик, начиная с младшего разряда:

$$\begin{array}{r} \bullet\ \bullet\ \bullet\ \bullet\ \bullet \\ 1\ 1\ 0\ 0\ 1\ 0\ 1 \\ -\ 0\ 1\ 0\ 1\ 1\ 1\ 0 \\ \hline 0\ 1\ 1\ 0\ 1\ 1\ 1 \end{array}$$

При вычитании во втором разряде из нуля единицы происходит заем единицы из старшего третьего разряда, а разность получается равная единице. Таким образом, после заема в третьем разряде для вычитания необходимо провести заем единицы из старшего четвертого разряда, а разность получается равная

единице. При вычитании в четвертом разряде происходит заем единицы из старшего пятого разряда и, т. к. уменьшаемое в четвертом разряде – это «нулевой» разряд, из которого уже происходил заем в младший разряд, а вычитаемое – единица, то разность будет равна нулю. При вычитании в пятом разряде происходит заем единицы из старшего шестого разряда и, т. к. уменьшаемое в пятом разряде – это «нулевой» разряд из которого уже происходил заем в младший разряд, а вычитаемое тоже нуль, то разность будет равна единице. При вычитании в шестом разряде происходит заем единицы из старшего седьмого разряда, а разность получается равная единице. В седьмом разряде уменьшаемым и вычитаемым будут являться нулевые разряды, поэтому разность будет равняться также нулю.

Умножение двоичных положительных чисел

В общем виде правила умножения двоичных положительных чисел выглядят следующим образом:

- два двоичных положительных числа умножаются в столбик поразрядно, при этом формирование произведения выполняется за счет сложения частичных произведений, которые формируются посредством умножения множимого на отдельные разряды множителя с учетом веса соответствующего разряда множителя;
- умножение может выполняться как начиная с младшего разряда, так и начиная со старшего (в отличие от десятичной арифметики);
- частичное произведение для разряда множителя равняется нулю, если этот разряд равен нулю;
- частичное произведение для разряда множителя равняется множимому, взятому с соответствующим весом, если разряд множителя равен единице;
- суммирование частичных произведений производится также в столбик с учетом веса соответствующего разряда (и сдвига влево или вправо на один разряд каждого частичного произведения при умножении с младшего или старшего разряда соответственно) по вышеописанным правилам сложения.

Наглядно данные правила изображены в таблице 2.3.

Таблица 2.3 – Правила умножения двоичных положительных чисел

		Множимое	
		0	1
Множитель	0	**0**	**0**
	1	**0**	**1**

Пример

Найти произведение двух положительных двоичных чисел 1001011_2 и 110101_2, начиная формирование частичных произведений с младшего разряда.

Выполняем умножение поразрядно в столбик, начиная с младшего разряда:

				×	1	0	0	1	0	1	1	– множимое;
					1	1	0	1	0	1		– множитель;
					1	0	0	1	0	1	1	– 1-е частичное произведение;
			+	0	0	0	0	0	0	0		– 2-е частичное произведение;
				0	1	0	0̇	1	0	1	1	– сумма 1-го и 2-го произведений;
		+	1	0	0	1	0	1	1			– 3-е частичное произведение;
			1	0	1	1	1	0	1	1	1	– сумма 1, 2 и 3-го произведений;
	+	0	0	0	0	0	0	0				– 4-е частичное произведение;
		0̇	1̇	0̇	1̇	1̇	1	0	1	1	1	– сумма 1, 2, 3 и 4-го произведений;
+	1	0	0	1	0	1	1					– 5-е частичное произведение;
	1	1	0	0̇	0̇	1	0	0	1	1	1	– сумма 1, 2, 3, 4 и 5-го произведений;
+	1	0	0	1	0	1	1					– 6-е частичное произведение;
	1	1	1	1	1	0	0	0	0	1	1	1 – конечное произведение (сумма всех частичных произведений).

Таким образом, количество частичных произведений соответствует числу разрядов множителя. В примере точками сверху отмечены разряды, в которые происходит перенос единицы по рассмотренным выше правилам сложения положительных двоичных чисел. Начиная со второго частичного произведения происходит сдвиг следующего частичного произведения на один разряд влево. Количество разрядов частичных произведений соответствует количеству разрядов множимого.

Пример

Найти произведение двух положительных двоичных чисел 1001011_2 и 110101_2, начиная формирование частичных произведений со старшего разряда.

Выполняем умножение поразрядно в столбик, начиная со старшего разряда:

	×	1	0	0	1	0	1	1				– множимое;
		1	1	0	1	0	1					– множитель;
	1	0	0̇	1̇	0̇	1̇	1					– 1-е частичное произведение;
+		1	0	0	1	0	1	1				– 2-е частичное произведение;
	1	1	1	0	0	0	0	1				– сумма 1-го и 2-го произведений;
+			0	0	0	0	0	0	0			– 3-е частичное произведение;
	1	1	1	0	0	0	0	1	0			– сумма 1, 2 и 3-го произведений;
+				1	0	0	1	0	1	1		– 4-е частичное произведение;
	1	1	1	1	0	0	1	1	1	1		– сумма 1, 2, 3 и 4-го произведений;
+					0	0	0	0	0	0	0	– 5-е частичное произведение;
	1	1	1	1	0̇	0̇	1̇	1̇	1	1	0	– сумма 1, 2, 3, 4 и 5-го произведений;
+						1	0	0	1	0	1	1 – 6-е частичное произведение;
	1	1	1	1	1	0	0	0	0	1	1	1 – конечное произведение (сумма всех частичных произведений).

Аналогично количество частичных произведений соответствует числу разрядов множителя. Начиная со второго частичного произведения происходит сдвиг следующего частичного произведения на один разряд вправо. Количество

разрядов частичных произведений соответствует количеству разрядов множимого. Следует отметить, что при суммировании частичных произведений, сложение производится по вышеописанным правилам, т. е. с младшего разряда, и при переполнении единица переносится в старший разряд. В примере точками сверху отмечены разряды, в которые происходит перенос единицы.

Как видно, вне зависимости от того, с младшего или со старшего разряда производится умножение, результат получается одинаковый.

При умножении правильных двоичных дробей действуют те же правила. При умножении одного n-разрядного числа на другое n-разрядное число получается $(2\times n)$-разрядное произведение, в котором младшие n-разряды можно округлять. Точка, отделяющая целую и дробные части числа, игнорируется при формировании частичных произведений и ставится только в самом конце, когда получено конечное произведение.

Пример

Найти произведение двух положительных правильных двоичных дробей 0.110_2 и 0.011_2, начиная формирование частичных произведений с младшего разряда.

Выполняем умножение поразрядно в столбик, начиная с младшего разряда:

			×	0.	1	1	0	– множимое;
				0.	0	1	1	– множитель;
				0	1	1	0	–1-е частичное произведение;
		+	0̇	1̇	1	0		–2-е частичное произведение;
			1	0	0	1	0	– сумма 1-го и 2-го произведений;
	+	0	0	0	0			– 3-е частичное произведение;
		0	1	0	0	1	0	– сумма 1, 2 и 3-го произведений;
+	0	0	0	0				– 4-е частичное произведение;
0.	0	1	0	0	1	0		– конечное произведение (сумма всех частичных произведений).

Аналогичным образом можно умножать правильные двоичные положительные дроби, начиная со старшего разряда.

Деление двоичных положительных чисел

Деление в любой системе счисления в общем случае является неточной операцией, поэтому при ее выполнении прежде всего устанавливается количество разрядов частного, которые подлежат определению.

Деление в двоичной системе счисления может выполняться точно так же, как и в десятичной, однако формирование частного двоичных операндов реализуется гораздо проще, т. к. выполняются следующие условия:

- упрощается процедура подбора очередной цифры т. к. в двоичной системе очередной цифрой может быть одна из двух: либо нуль, либо единица;

- упрощается процедура умножения найденной цифры частного на делитель.

В общем виде правила деления двоичных положительных чисел выглядят следующим образом:

- два двоичных положительных числа делятся в столбик, как и десятичные. Сначала в делимом со старшего разряда берется *n* разрядов, где *n* – число разрядов в делителе.

- если число, образуемое этими *n* разрядами, больше делителя, то записывается единичный старший разряд частного, а от этих *n* старших разрядов делимого отнимается делитель, получается первая частичная разность;

- если число, образуемое этими *n* разрядами, меньше делителя, то записывается нулевой старший разряд частного, а от этих *n* старших разрядов делимого отнимается *n* нулевых разрядов – получается первая частичная разность;

- далее к полученным частичным разностям добавляется следующий старший разряд делимого;

- если после добавления последующего старшего разряда делимого к частичной разности получается число, большее делителя, то записывается единичный старший разряд частного и вычитается очередная частичная разность;

- если после добавления последующего старшего разряда делимого к частичной разности получается число, меньшее делителя, то добавляется следующий старший разряд делимого, а в частное записывается нулевой разряд и вычитается очередная частичная разность;

- вычитание частичной разности выполняется по правилам вычитания положительных двоичных чисел, описанным выше;

- процесс деления продолжается, пока к очередной частичной разности не будет добавлен самый младший разряд делимого и не будет получен остаток от деления (нулевой, если делимое делится на делитель без остатка, или ненулевой – в противном случае).

Пример

Найти частное положительных двоичных чисел 10001111_2 и 1101_2.

Делимое –		1	0	0	0	1	1	1	1	1	1	0	1	– делитель;
	–	0	0	0	0					0	1	0	1	1 – частное;
1-я разность –		1̇	0̇	0	0	**1**								
	–		1	1	0	1								
2-я разность –		0	1̇	0̇	0	**1**	**1**							
	–			1	1	0	1							
3-я разность –				0	1	1	0	**1**						
	–				1	1	0	1						
					0	0	0	0						– остаток.

В примере точками сверху отмечены старшие разряды, в которых происходит заем единицы по рассмотренным выше правилам вычитания положительных двоичных чисел. Полужирным шрифтом отмечены очередные добавляемые к частичной разности разряды делимого.

Сложение и вычитание положительных двоично-десятичных чисел

Двоично-десятичная система счисления представляет собой синтез двоичной и десятичной систем счисления. Каждый разряд десятичного числа (каждая цифра) представляется в виде двоичной тетрады (четырех разрядов). Ариф-

метические операции сложения и вычитания в двоично-десятичной системе счисления выполняются по правилам двоичной арифметики, рассмотренным выше, с последующими определенными корректировками.

Правила коррекции при сложении и вычитании двоично-десятичных чисел в общем виде выглядят следующим образом:

- необходимо добавить двоичную шестерку (0110) в те тетрады, из которых был перенос при их переполнении при двоичном сложении. Необходимость такой коррекции обуславливается тем, что перенос в старшую тетраду, сформированный по правилам двоичного суммирования, перенес из младшей тетрады значение, равное шестнадцати, а для десятичного сложения перенос должен был перенести значение, равное десяти. То есть перенос, сформированный по правилам двоичной арифметики, убрал из младшей тетрады значение, на шестерку большее, чем необходимо;

- необходимо вычесть двоичную шестерку (0110) из тех тетрад, из которых произошел заем в младшие тетрады. Это обуславливается тем, что заем, сформированный по правилам двоичного вычитания, приносит в тетраду значение, равное шестнадцать, а для десятичного вычитания заем должен был принести в тетраду значение, равное десяти. Заем, сформированный по правилам двоичного вычитания, принес значение, на шестерку больше необходимого;

- необходимо добавить двоичную шестерку (0110) в те тетрады, в которых получено значение, большее девяти. Такая коррекция обуславливается тем, что по правилам десятичной арифметики в таких тетрадах должен быть выработан перенос, и, чтобы его выработать, по правилам двоичной арифметики в тетраду нужно добавить значение, равное шести.

Пример

Найти сумму десятичных чисел $A = 3927_{10}$ и $B = 4856_{10}$, используя двоично-десятичную систему счисления.

В соответствие каждому разряду десятичных чисел A и B находим двоичные тетрады:

$A_{10} = 3\ \ 9\ \ 2\ \ 7$, $B_{10} = 4\ \ 8\ \ 5\ \ 6$,

$A_{2-10} = 0011\ 1001\ 0010\ 0111$; $B_{2-10} = 0100\ \ 1000\ \ 0101\ \ 0110$.

Складываем двоично-десятичные числа по правилам двоичной арифметики и в необходимых тетрадах выполняем корректировку:

		•		*	
	0011	1001	0010	0111	– число A_{2-10};
+	0100	1000	0101	0110	– число B_{2-10};
	1000	0001	0111	1101	– двоичная сумма;
+		0110		0110	– коррекция;
	1000	0111	1000	0011	– двоично-десятичная сумма.

В примере точкой обозначена тетрада, в которой произошли переполнение и перенос в старшую тетраду, а звездочкой – тетрада, в которой в результате двоичного сложения получилось значение, большее девяти (тетрада превы-

сила значение 1001). По правилам в этих тетрадах необходимо провести коррекцию (прибавить к ним двоичную шестерку).

Пример

Найти разность десятичных чисел $B = 4856_{10}$ и $A = 3927_{10}$, используя двоично-десятичную систему счисления.

В соответствие каждому разряду десятичных чисел A и B находим двоичные тетрады:

$A_{10} =$ 3 9 2 7, $B_{10} =$ 4 8 5 6,

$A_{2-10} =$ 0011 1001 0010 0111 ; $B_{2-10} =$ 0100 1000 0101 0110.

Вычитаем двоично-десятичные числа по правилам двоичной арифметики и в необходимых тетрадах выполняем корректировку:

		•		•	
	0100	1000	0101	0110	– число B_{2-10};
–	0011	1001	0010	0111	– число A_{2-10};
	0000	1111	0010	1111	– двоичная разность;
–		0110		0110	– коррекция;
	0000	1001	0010	1001	– двоично-десятичная разность.

В примере точками обозначены тетрады, в которые взят заем из старших тетрад. По правилам в этих тетрадах необходимо провести коррекцию (вычесть из них двоичную шестерку).

2.3 Задачи для решения

Задание 1

Найти сумму двух положительных двоичных чисел A и B:

а) $A = 110101$, $B = 100010$;
в) $A = 101001$, $B = 111010$;
б) $A = 101011010$, $B = 11000101$;
г) $A = 110010101$, $B = 11011101$.

Задание 2

Найти разность двух положительных двоичных чисел A и B:

а) $A = 10011100$, $B = 111001$;
в) $A = 10010101$, $B = 100101$;
б) $A = 10101100$, $B = 1011111$;
г) $A = 11001001$, $B = 1011011$.

Задание 3

Найти произведение двух положительных двоичных чисел A и B, начиная формирование частичных произведений с младшего разряда:

а) $A = 10110110$, $B = 1001101$;
д) $A = 11001101$, $B = 1001110$;
б) $A = 11001011$, $B = 10110101$;
е) $A = 11101011$, $B = 10001101$;
в) $A = 0.1011$, $B = 0.1101$;
ж) $A = 0.1111$, $B = 0.1101$;
г) $A = 0.0111$, $B = 0.1001$;
з) $A = 0.0101$, $B = 0.1011$.

Задание 4

Найти произведение двух положительных двоичных чисел *A* и *B*, начиная формирование частичных произведений со старшего разряда:

а) *A* = 10010010, *B* = 1011111;
б) *A* = 11101011, *B* = 11010101;
в) *A* = 0.0101, *B* = 0.1001;
г) *A* = 0.1011, *B* = 0.1111;
д) *A* = 11100101, *B* = 1010101;
е) *A* = 10010111, *B* = 10111001;
ж) *A* = 0.0111, *B* = 0.1011;
з) *A* = 0.1111, *B* = 0.1101.

Задание 5

Найти частное двух положительных двоичных чисел *A* и *B*:

а) *A* = 101010000, *B* = 11100;
б) *A* = 100001100010, *B* = 100101;
в) *A* = 100111000, *B* = 11000;
г) *A* = 100000111011, *B* = 101011.

Задание 6

Найти сумму двух положительных десятичных чисел *A* и *B*, используя двоично-десятичную систему счисления:

а) *A* = 1245, *B* = 2836;
б) *A* = 3744, *B* = 2916;
в) *A* = 1073, *B* = 2541;
г) *A* = 3807, *B* = 2764.

Задание 7

Найти разность двух положительных десятичных чисел *A* и *B*, используя двоично-десятичную систему счисления:

а) *A* = 2946, *B* = 1567;
б) *A* = 3285, *B* = 2563;
в) *A* = 2821, *B* = 1343;
г) *A* = 3615, *B* = 2674.

2.4 Контрольные вопросы

1 Чему будет равна сумма двух единичных разрядов при сложении положительных двоичных чисел?

2 Как происходит формирование разности нулевого и единичного разрядов положительных двоичных чисел?

3 В чем заключается особенность формирования разности двух положительных двоичных чисел, если из разряда уменьшаемого происходил заем единицы в младший разряд?

4 В какую сторону происходит сдвиг частичных произведений при умножении двух положительных двоичных чисел, начиная с младшего разряда? Начиная со старшего разряда?

5 Какое количество разрядов делимого используется на первом этапе деления двух положительных двоичных чисел?

6 При превышении какого значения тетрады необходима корректировка двоичной суммы при сложении в двоично-десятичной системе счисления?

7 На какое значение и почему происходит корректировка тетрад двоичной разности при вычитании в двоично-десятичной системе счисления?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №3
АРИФМЕТИКА С АЛГЕБРАИЧЕСКИМИ ЧИСЛАМИ

Цель: приобрести практические навыки решения арифметических задач с алгебраическими числами в прямом, дополнительном, обратном и модифицированных кодах.

3.1 Краткие теоретические сведения

В компьютерной технике для представления и обработки алгебраических чисел используются следующие базовые коды:

- прямой (ПК);
- дополнительный (ДК);
- обратный (ОК).

Все три разновидности кодирования чисел имеют формат представления, содержащий два поля: поле знака и поле модуля (рисунок 3.1).

Рисунок 3.1 – Формат представление алгебраических двоичных чисел

Во всех разновидностях кодировки поле знака представлено одним разрядом, в котором устанавливается нуль, если число положительное, и единица, если число отрицательное. Поле модуля отражает количественную оценку числа и для каждого кода формируется по-разному. Количество разрядов поля модуля определятся диапазоном изменения отображаемых чисел или точностью их представления.

Отдельно можно выделить модифицированные базовые коды. Поле знака в таких кодах представлено двумя разрядами по следующему правилу:

1) 00 – для положительных чисел;
2) 11 – для отрицательных чисел;
3) 01 – при переполнении поля модуля. Число считается положительным;
4) 10 – при переполнении поля модуля. Число является отрицательным.

То есть основным носителем знака числа является старший (левый) разряд знакового поля.

При кодировании числа в прямой код (ПК) запись целого числа A формируется по следующему правилу:

$$[A]_{\text{ПК}} = \begin{cases} 0.A, \text{если } A \geq 0, \\ 1.|A|, \text{если } A < 0. \end{cases} \quad (3.1)$$

При кодировании числа в дополнительный код (ДК) запись целого числа A формируется по следующему правилу:

$$[A]_{\text{ДК}} = \begin{cases} 0.A, \text{если } A \geq 0, \\ 1.(q^n + A), \text{если } A < 0, \end{cases} \quad (3.2)$$

где q – основание системы счисления;

n – разрядность модульного поля;

q^n – максимальная невключенная граница диапазона изменения представляемых чисел.

При кодировании числа в дополнительный код запись дробного числа B формируется по следующему правилу:

$$B_{\text{дк}} = \begin{cases} 0.B, \text{если } B \geq 0, \\ 1.(1+B), \text{если } B < 0, \end{cases} \quad (3.3)$$

где 1 – максимальная невключенная граница диапазона изменения представляемых чисел.

При кодировании числа в обратный код (ОК) запись целого числа A формируется по следующему правилу:

$$[A]_{\text{ок}} = \begin{cases} 0.A, \text{если } A \geq 0, \\ 1.((q^n - 1) + A), \text{если } A < 0, \end{cases} \quad (3.4)$$

где q – основание системы счисления;

n – разрядность модульного поля;

$q^n - 1$ – максимальная включенная граница диапазона изменения представляемых чисел.

При кодировании числа в обратный код запись дробного числа B формируется по следующему правилу:

$$B_{\text{ок}} = \begin{cases} 0.B, \text{если } B \geq 0, \\ 1.((1 - q^n) + B), \text{если } B < 0, \end{cases} \quad (3.5)$$

где $1 - q^n$ – максимальная включенная граница диапазона изменения представляемых чисел.

Таким образом, из вышеописанного следует:

- положительное алгебраическое число в прямом, обратном и дополнительном кодах имеет одинаковое представление;

- для того чтобы перевести отрицательное число из обратного в прямой код (и наоборот), необходимо дополнить его модуль до включенной границы;

- для того чтобы перевести отрицательное число из дополнительного в прямой код (и наоборот), необходимо дополнить его модуль до невключенной границы.

Для двоичных чисел включенной границей является значение, равное 2^n, а невключенной границей является значение, равное $2^n - 1$.

Отсюда следует, что запись модульной части отрицательного числа в обратном коде представляет собой инверсию модульной части записи этого числа в прямом коде, т. е. нули заменяются единицами, а единицы – нулями. В свою очередь, запись модульной части отрицательного числа в дополнительном коде отличается от записи в обратном коде на значение, соответствующее единице младшего разряда.

Обратный и дополнительный коды позволяют упростить арифметические операции над алгебраическими числами. Поэтому, как правило, значения чисел хранятся в компьютерной технике в прямом коде, а при выполнении операций над этими числами применяется обратный или дополнительный код.

При использовании дополнительного или обратного кода операция вычитания заменяется на операцию сложения с изменением знака второго операнда (изменением значения разряда знакового поля на противоположное) и сложение осуществляется по правилам двоичной арифметики.

3.2 Методические указания и примеры решения задач

Исходя из (3.1), (3.2) и (3.4) в общем виде правила формирования модуля отрицательного целого двоичного числа заключаются в следующем:

- чтобы сформировать модульную часть записи отрицательного числа в обратном коде, необходимо в модульной части записи этого числа в прямом коде взять обратные значения всех двоичных разрядов (т. е. проинвертировать модуль прямого кода);
- для перевода из обратного кода назад в прямой также необходимо проинвертировать запись модульной части числа в обратном коде;
- чтобы сформировать модульную часть записи отрицательного числа в дополнительном коде, необходимо в модульной части записи этого числа в прямом коде взять обратные значения всех двоичных разрядов (т. е. проинвертировать модуль прямого кода) и к полученному коду прибавить единицу в младший разряд;
- для перевода из дополнительного кода назад в прямой также необходимо проинвертировать запись модульной части числа в дополнительном коде и к полученному коду прибавить единицу в младший разряд.

Сложение алгебраических чисел в обратном и дополнительном кодах

Поскольку вычитание в обратном и дополнительном кодах заменяется на операцию сложения с изменением знака вычитаемого числа, то правила сложения в обратном и дополнительном кодах выглядят следующим образом:

- при сложении чисел, представленных в дополнительном коде, выполняется сложение разрядов, представляющих запись операндов, по правилам двоичной арифметики по всей длине записи чисел, не обращая внимание на границу, разделяющую знаковое и модульное поля. Переполнение разрядности знакового поля, т. е. перенос, возникший из крайнего разряда модульного поля, игнорируется. В результате такого сложения будет получен дополнительный код суммы заданных операндов;
- при сложении чисел, представленных в обратном коде, выполняется сложение разрядов, представляющих запись операндов, по правилам двоичной арифметики по всей длине записи чисел, не обращая внимания на границу, разделяющую знаковое и модульное поля. Переполнение разрядности знакового поля, т. е. перенос, возникший из крайнего разряда модульного поля, должен быть учтен как плюс единица в младший разряд полученной суммы. В результате такого сложения будет получен обратный код суммы заданных операндов.

Пример

Сформировать запись десятичных чисел $A = 182_{10}$ и $B = -107_{10}$ в прямом, обратном и дополнительном двоичных кодах и найти значения чисел $C_1 = A + B$, $C_2 = A - B$, используя обратный код, и значения чисел $C_3 = B - A$, $C_4 = -A - B$, используя дополнительный код. Все результаты арифметических операций представить в прямом коде.

Сначала переводим десятичные числа A и B в двоичные:

$$A = 182_{10} = +10110110_2,\ B = -107_{10} = -1101011_2.$$

Число A имеет восемь разрядов, а число B – семь разрядов. Поскольку над этими числами будут выполняться арифметические операции, необходимо привести их к одинаковому количеству разрядов модульной части, учитывая также ожидаемый результат выполнения заданных арифметических операций. Таким образом, количество разрядов модульной части представления чисел в различных кодах должно быть равным девяти. То есть число A дополняется одним старшим нулевым разрядом, а число B дополняется двумя старшими нулевыми разрядами:

$$A = 182_{10} = +010110110_2,\ B = -107_{10} = -001101011_2.$$

Далее формируем запись чисел в прямом, обратном и дополнительном кодах согласно (3.1), (3.2) и (3.4). Поскольку число A – положительное, его представление в обратном и дополнительном кодах будет совпадать с представлением в прямом коде:

$$[A]_{ПК} = [A]_{ОК} = [A]_{ДК} = 0.010110110.$$

Число B – отрицательное и будет иметь следующее представление в прямом коде:

$$[B]_{ПК} = 1.001101011.$$

Для определения модульной части числа B в обратном коде прибавим к включенной границе диапазона ($2^n - 1 = 111111111$) число B (по формуле (3.4)):

$$111111111 + (-001101011) = 110010100.$$

Таким образом, $[B]_{ОК} = 1.110010100$.

Для определения модульной части числа B в дополнительном коде необходимо прибавить к невключенной границе диапазона ($2^n = 1000000000$) число B (по формуле (3.2)):

$$1000000000 + (-001101011) = 110010101.$$

Таким образом, $[B]_{ДК} = 1.110010101$.

Поскольку в обратном и дополнительном кодах операция вычитания заменяется на операцию сложения с заменой знака вычитаемого на противоположный, приведем необходимые операции в соответствующий вид:

$$C_1 = A + B,\ C_2 = A + (-B),\ C_3 = B + (-A),\ C_4 = (-A) + (-B).$$

Находим необходимые значения чисел:

$[A]_{ПК} = 0.010110110$;	$[A]_{ОК} = 0.010110110$;	$[A]_{ДК} = 0.010110110$;
$[-A]_{ПК} = 1.010110110$;	$[-A]_{ОК} = 1.101001001$;	$[-A]_{ДК} = 1.101001010$;
$[B]_{ПК} = 1.001101011$;	$[B]_{ОК} = 1.110010100$;	$[B]_{ДК} = 1.110010101$;
$[-B]_{ПК} = 0.001101011$;	$[-B]_{ОК} = 0.001101011$;	$[-B]_{ДК} = 0.001101011$.

Находим значение $C_1 = A + B$, используя обратный код:

$[A]$ок =		0. 0 1 0 1 1 0 1 1 0
$[B]$ок =	+	1. 1 1 0 0 1 0 1 0 0
		1 0. 0 0 1 0 0 1 0 1 0
корректировка единицей в младший разряд при переполнении знакового поля –	+	1
$[C_1]$ок = $[C_1]$пк =		0. 0 0 1 0 0 1 0 1 1.

В рассмотренном примере нахождения значения числа C_1 возникает переполнение разрядности знакового поля, которое учитывается как плюс единица в младший разряд согласно правилам сложения в обратном коде.

Находим значение $C_2 = A + (–B)$, используя обратный код:

$[A]$ок =		0. 0 1 0 1 1 0 1 1 0
$[–B]$ок =	+	0. 0 0 1 1 0 1 0 1 1
$[C_2]$ок = $[C_2]$пк =		0. 1 0 0 1 0 0 0 0 1.

В рассмотренном примере нахождения значения числа C_2 вычитание заменяется на сложение и второе слагаемое инвертируется и берется с противоположным знаком $(–B)$.

Находим значение $C_3 = B + (–A)$, используя дополнительный код:

$[B]$дк =		1. 1 1 0 0 1 0 1 0 1
$[–A]$дк =	+	1. 1 0 1 0 0 1 0 1 0
$[C_3]$дк =		1 1. 0 1 1 0 1 1 1 1 1
$[C_3]$пк =		1. 1 0 0 1 0 0 0 0 1.

В рассмотренном примере нахождения значения числа C_3 вычитание заменяется на сложение, и второе слагаемое инвертируется и берется с противоположным знаком $(–A)$. Переполнение разрядности знакового поля по правилам сложения в дополнительном коде игнорируется. В результате получается отрицательная сумма в дополнительном коде, которую необходимо проинвертировать и добавить единицу в младший разряд для перевода в прямой код.

Находим значение $C_4 = (–A) + (–B)$, используя дополнительный код:

$[–A]$дк =		1. 1 0 1 0 0 1 0 1 0
$[–B]$дк =	+	0. 0 0 1 1 0 1 0 1 1
$[C_4]$дк =		1. 1 1 0 1 1 0 1 0 1
$[C_4]$пк =		1. 0 0 1 0 0 1 0 1 1.

В рассмотренном примере нахождения значения числа C_4 вычитание заменяется на сложение, и оба слагаемых инвертируются и берутся с противоположным знаком $(–A$ и $–B)$. В результате получается отрицательная сумма в дополнительном коде, которую необходимо проинвертировать и добавить единицу в младший разряд для перевода в прямой код.

Модифицированные коды

Иногда бывает трудно заранее определить разрядность модульной части алгебраических двоичных чисел, особенно при выполнении последовательных арифметических операций. Может возникнуть ситуация, когда при сложении двух чисел с одинаковым знаком в результате переполнения модульного поля

получается сумма с противоположным слагаемым знаком. Выполнение таких операций в модифицированных кодах решает данную проблему.

Если в результате сложения чисел в модифицированном коде полученный результат имеет в знаковом поле одинаковые значения в обоих разрядах (00 или 11), то переполнения модульного поля нет. Если же разряды знакового поля имеют неодинаковые значения (10 или 01), то имеет место переполнение модульного поля. При этом, если в поле знака имеет место значение 01, то результат положительный, а если значение 10, то полученный результат отрицательный (основным носителем знака числа является левый разряд знакового поля).

Пример

Найти значения чисел $C_1 = A + B$, $C_2 = A - B$, используя модифицированный обратный код (МОК), и значения чисел $C_3 = B - A$, $C_4 = -A - B$, используя модифицированный дополнительный код (МДК), если $A = 10100_2$ и $B = -01110_2$. Результат представить в модифицированном прямом коде (МПК).

Поскольку в обратном и дополнительном кодах операция вычитания заменяется на операцию сложения с заменой знака вычитаемого на противоположный, приведем необходимые операции в соответствующий вид:

$$C_1 = A + B,\ C_2 = A + (-B),\ C_3 = B + (-A),\ C_4 = (-A) + (-B).$$

Находим необходимые значения чисел в модифицированных кодах (модифицированный прямой код – МПК, модифицированный обратный код – МОК, модифицированный дополнительный код – МДК):

$[A]_{МПК} = 00.10100;$	$[A]_{МОК} = 00.10100;$	$[A]_{МДК} = 00.10100;$
$[-A]_{МПК} = 11.10100;$	$[-A]_{МОК} = 11.01011;$	$[-A]_{МДК} = 11.01100;$
$[B]_{МПК} = 11.01110;$	$[B]_{МОК} = 11.10001;$	$[B]_{МДК} = 11.10010;$
$[-B]_{МПК} = 00.01110;$	$[-B]_{МОК} = 00.01110;$	$[-B]_{МДК} = 00.01110.$

Находим значение $C_1 = A + B$, используя модифицированный обратный код:

$$\begin{array}{rr} [A]\text{мок} = & 0\,0.\,1\,0\,1\,0\,0 \\ [B]\text{мок} = & +\ 1\,1.\,1\,0\,0\,0\,1 \\ \hline \text{корректировка единицей в младший разряд при переполнении разрядности знакового поля} - & 1\ 0\,0.\,0\,0\,1\,0\,1 \\ & +\ 1 \\ \hline [C_1]\text{мок} = [C_1]\text{мпк} = & 0\,0.\,0\,0\,1\,1\,0. \end{array}$$

В рассмотренном примере нахождения значения числа C_1 возникает переполнение разрядности знакового поля, которое учитывается как плюс единица в младший разряд согласно правилам сложения в обратном коде. Переполнения модульного поля не произошло, поскольку разряды знакового поля одинаковы.

Находим значение $C_2 = A + (-B)$, используя модифицированный обратный код:

$$\begin{array}{rr} [A]\text{мок} = & 0\,0.\,1\,0\,1\,0\,0 \\ [-B]\text{мок} = & +\ 0\,0.\,0\,1\,1\,1\,0 \\ \hline \text{переполнение модульного поля, т. к. знаковые разряды имеют разное значение} - & 0\,1.\,0\,0\,0\,1\,0 \quad - \text{сдвинуть точку влево} \\ [C_2]\text{мок} = [C_2]\text{мпк} = & 0\,0.\,1\,0\,0\,0\,1\,0. \end{array}$$

В рассмотренном примере нахождения значения числа C_2 возникает переполнение модульного поля, поскольку разряды знакового поля различны. Младший разряд знакового поля необходимо перенести в модульное поле. Значение суммы положительно, поскольку носитель знака – старший разряд знакового поля – равен нулю.

Находим значение $C_3 = B + (-A)$, используя модифицированный дополнительный код:

$[B]$мдк = 1 1. 1 0 0 1 0

$+$ $[-A]$мдк = 1 1. 0 1 1 0 0

переполнение модульного поля, т. к. знаковые разряды имеют разное значение – 1 1 0. 1 1 1 1 0 – сдвинуть точку влево

$[C_3]$мдк = 1 1. 0 1 1 1 1 0 – проинвертировать и добавить единицу в младший разряд

$[C_3]$мпк = 1 1. 1 0 0 0 1 0.

В рассмотренном примере нахождения значения числа C_3 возникает переполнение модульного поля, поскольку разряды знакового поля различны. Младший разряд знакового поля необходимо перенести в модульное поле. Значение суммы отрицательно, поскольку носитель знака – старший разряд знакового поля – равен единице, и для получения записи числа в прямом коде необходимо проинвертировать модульное поле и прибавить единицу в младший разряд модульного поля.

Находим значение $C_4 = (-A) + (-B)$, используя модифицированный дополнительный код:

$[-A]$мдк = 1 1. 0 1 1 0 0

$+$ $[-B]$мдк = 0 0. 0 1 1 1 0

$[C_4]$мдк = 1 1. 1 1 0 1 0 – проинвертировать и добавить единицу в младший разряд

$[C_4]$мпк = 1 1. 0 0 1 1 0.

В рассмотренном примере нахождения значения числа C_4 переполнение модульного поля не возникает, поскольку разряды знакового поля одинаковы. Значение суммы отрицательно, поскольку оба разряда знакового поля равны единице, и для получения записи числа в прямом коде необходимо проинвертировать модульное поле и прибавить единицу в младший разряд.

3.3 Задачи для решения

Задание 1

Сформировать запись десятичных чисел A_{10} и B_{10} в прямом, обратном и дополнительном двоичных кодах:

а) $A = 207_{10}$, $B = -155_{10}$;

б) $A = -328_{10}$, $B = 246_{10}$;

в) $A = 286_{10}$, $B = -177_{10}$;

г) $A = -367_{10}$, $B = 294_{10}$.

Задание 2

Найти значения чисел $C_1 = A + B$, $C_2 = A - B$, $C_3 = B - A$, $C_4 = -A - B$, используя обратный двоичный код. Результаты представить в прямом коде:

а) $A = 128_{10}$, $B = 112_{10}$;
б) $A = 247_{10}$, $B = -179_{10}$;
в) $A = 212_{10}$, $B = -303_{10}$;
г) $A = 141_{10}$, $B = 108_{10}$;
д) $A = 269_{10}$, $B = -205_{10}$;
е) $A = 250_{10}$, $B = -321_{10}$.

Задание 3

Найти значения чисел $C_1 = A + B$, $C_2 = A - B$, $C_3 = B - A$, $C_4 = -A - B$, используя дополнительный двоичный код. Результаты представить в прямом двоичном коде:

а) $A = 147_{10}$, $B = 132_{10}$; в) $A = 265_{10}$, $B = -333_{10}$; д) $A = 299_{10}$, $B = -281_{10}$
б) $A = 288_{10}$, $B = -228_{10}$; г) $A = 162_{10}$, $B = 151_{10}$; е) $A = 271_{10}$, $B = -359_{10}$.

Задание 4

Найти значения чисел $C_1 = A + B$, $C_2 = A - B$, $C_3 = B - A$, $C_4 = -A - B$, используя модифицированный обратный двоичный код. Результаты представить в модифицированном прямом двоичном коде:

а) $A = 116_{10}$, $B = 101_{10}$; в) $A = 217_{10}$, $B = -310_{10}$; д) $A = 244_{10}$, $B = -213_{10}$;
б) $A = 222_{10}$, $B = -166_{10}$; г) $A = 159_{10}$, $B = 111_{10}$; е) $A = 260_{10}$, $B = -309_{10}$.

Задание 5

Найти значения чисел $C_1 = A + B$, $C_2 = A - B$, $C_3 = B - A$, $C_4 = -A - B$, используя модифицированный дополнительный двоичный код. Результаты представить в модифицированном прямом двоичном коде:

а) $A = 135_{10}$, $B = 103_{10}$;
б) $A = 299_{10}$, $B = -267_{10}$;
в) $A = 240_{10}$, $B = -301_{10}$;
г) $A = 161_{10}$, $B = 129_{10}$;
д) $A = 256_{10}$, $B = -232_{10}$;
е) $A = 268_{10}$, $B = -329_{10}$.

3.4 Контрольные вопросы

1 Как выполняется операция вычитания в дополнительном и обратном кодах?

2 Как формируется запись положительного алгебраического числа в обратном и дополнительном кодах?

3 Как формируется запись отрицательного алгебраического числа в обратном и дополнительном кодах?

4 Как учитывается переполнение знакового поля при сложении в обратном коде?

5 Как учитывается переполнение знакового поля при сложении в дополнительном коде?

6 Чем отличаются модифицированные коды от базовых и в чем их преимущество?

7 Как определяется переполнение модульного поля в модифицированных кодах?

8 Какой разряд знакового поля является знакоопределяющим в модифицированных кодах?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №4
АРИФМЕТИКА С ПЛАВАЮЩЕЙ ТОЧКОЙ

Цель: приобрести практические навыки решения арифметических задач над числами, представленными в форме с плавающей точкой.

4.1 Краткие теоретические сведения

Числовая информация представляется в компьютерной технике в форме с фиксированной или плавающей точкой. При представлении в форме с фиксированной точкой положение точки, отделяющей целую и дробную части, в записи числа фиксировано.

При представлении числа в форме с плавающей точкой это число в общем случае представляет собой смешанную дробь и имеет формат, представленный на рисунке 4.1.

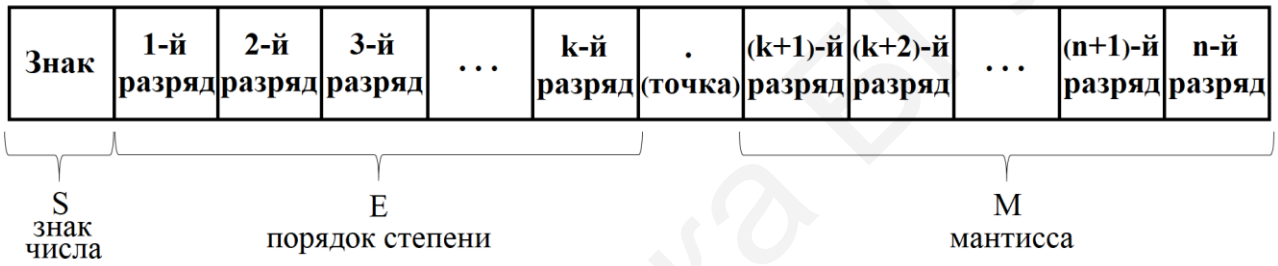


Рисунок 4.1 – Формат представления числа в форме с плавающей точкой

Местоположение точки в записи числа может быть различным, а т. к. сама точка в записи числа не присутствует, то, чтобы задать число однозначно, необходимы не только его запись, но и информация о том, где в записи числа располагается точка, отделяющая целую и дробную части.

При представлении в форме с плавающей точкой число Q представляется в виде трех частей:

- знака числа (S);
- мантиссы (M), отображающей запись числа, которая представляется в виде правильной дроби с форматом фиксированной точки;
- порядок степени (E), отображающий местоположение точки в записи числа, который представляется в виде целого числа с форматом фиксированной точки.

При этом количественная оценка числа Q определяется следующим образом:

$$Q = q^E \cdot M, \tag{4.1}$$

где q – основание системы счисления.

В компьютерной технике числа с плавающей точкой представляются в так называемой нормализованной форме, при которой в прямом коде мантисса нормализованного числа в старшем разряде модуля имеет ненулевое значение.

Для двоичной системы счисления нормализованная мантисса должна иметь в старшем разряде модуля прямого кода значение, равное единице, т. е. для двоичной системы счисления мантисса должна удовлетворять неравенству

$$1 > |M| \geq 0{,}5. \qquad (4.2)$$

Относительная ошибка при представлении чисел в форме с плавающей точкой существенно меньше, чем в случае с фиксированной точкой. Она, а также больший диапазон изменения представляемых чисел, являются основным преимуществом представления чисел с плавающей точкой.

4.2 Методические указания и примеры решения задач

Арифметические операции для чисел, представленных в форме с плавающей точкой, в общем выполняются по правилам арифметики алгебраических чисел, но с определенными дополнительными операциями, связанными с формой представления чисел.

Сложение/вычитание чисел с плавающей точкой

Операция сложения/вычитания чисел с плавающей точкой предполагает наличие одинаковых порядков у операндов, подлежащих суммированию (вычитанию). Именно поэтому сложение/вычитание таких чисел предполагает три этапа реализации операции:

- выравнивание порядков;
- сложение мантисс операндов, имеющих одинаковые порядки;
- определение нарушения нормализации и ее устранение.

Как и в случае с целыми числами, представленными в форме с фиксированной точкой, арифметические операции выполняются в обратном, дополнительном или их модифицированных кодах. Операция вычитания также заменяется на операцию сложения с изменением знака вычитаемого на противоположный.

Пример

Найти разность C чисел A и B, представленных с плавающей точкой, если эти числа представлены в виде порядков $[A_E]_{ПК} = 1.001$ и $[B_E]_{ПК} = 0.001$ соответственно и мантисс $[A_M]_{ПК} = 1.11001$ и $[B_M]_{ПК} = 0.11100$ соответственно. При вычитании использовать модифицированный дополнительный код.

Сначала необходимо выровнять порядки. Для этого из порядка числа A вычитается порядок числа B. Вычитание также заменяется сложением, при этом B_E заменяется на $–B_E$. Находим $[A_E]_{ДК}$ и $[–B_E]_{ДК}$:

$$[A_E]_{ДК} = 1.111 \text{ и } [–B_E]_{ДК} = 1.111.$$

Находим разность порядков:

$$\begin{array}{rl} & 1.\ 1\ 1\ 1 \quad – [A_E]_{ДК};\\ + & 1.\ 1\ 1\ 1 \quad – [–B_E]_{ДК};\\ \hline & 1.\ 1\ 1\ 0 \quad – \text{разность порядков в дополнительном коде;}\\ & 1.\ 0\ 1\ 0 \quad – \text{разность порядков в прямом коде.} \end{array}$$

Так как знак разности порядков отрицательный, то в качестве общего порядка, а следовательно, и предварительного значения порядка искомого результата C_E^*, берется порядок второго числа (B_E). Для того чтобы взять в качестве порядка первого числа порядок второго числа, т. е. в нашем случае увеличив его порядок на два, необходимо мантиссу этого меньшего числа (A_M) умножить на 2^{-2}, т. е. выполнить арифметический сдвиг на два разряда вправо:

$$A_M^* = A_M \cdot 2^{-2} = 1.11001 \cdot 2^{-2} = 1.00110.$$

Таким образом, после выравнивания порядков операндов будем иметь следующую форму представления операндов:

$$[A_M^*]_{\text{ПК}} = 1.00110,\ [B_M^*]_{\text{ПК}} = 1.11100.$$

Предварительное значение мантиссы определяется как $C_M^* = A_M^* - B_M^*$.

Определять предварительное значение мантиссы будем в модифицированном дополнительном коде, как указано в условии. Для этого находим значения $[A_M^*]_{\text{МДК}}$ и $[-B_M^*]_{\text{МДК}}$ (второй операнд берется с отрицательным знаком, поскольку операция вычитания в модифицированном дополнительном коде заменяется на операцию сложения):

$$[A_M^*]_{\text{МДК}} = 11.11010,\ [-B_M^*]_{\text{МДК}} = 11.00100.$$

Находим предварительное значение C_M^*:

$$\begin{array}{rl} & 1\ 1.\ 1\ 1\ 0\ 1\ 0 \quad -[A_M]\text{мдк;} \\ + & 1\ 1.\ 0\ 0\ 1\ 0\ 0 \quad -[-B_M]\text{мдк;} \\ \hline \cancel{1} & 1\ 0.\ 1\ 1\ 1\ 1\ 0 \quad -[C_M^*]\text{мдк;} \\ & 1\ 0.\ 0\ 0\ 0\ 1\ 0 \quad -[C_M^*]\text{мпк.} \end{array}$$

Из записи $[C_M^*]_{\text{МПК}}$, полученной после вычитания мантисс операндов с выравненными порядками, видно, что нормализация представления результата нарушена. Поэтому для данного примера необходимо выполнить этап устранения нарушения нормализации. В данном случае нарушение нормализации слева от точки, т. к. получено $[C_M^*]_{\text{МПК}}$ с ненулевой целой частью (неодинаковые разряды в знаковом поле использованного модифицированного дополнительного кода).

Поэтому необходимо выполнить этап устранения нарушения нормализации. Для того чтобы привести полученную предварительную мантиссу к нормализованной форме, достаточно ее умножить на 2^{-1}, т. е. выполнить ее арифметический сдвиг вправо. В результате будем иметь конечное значение мантиссы:

$$C_M = C_M^* \cdot 2^{-1} = 1.00010 \cdot 2^{-1} = 1.10001.$$

Арифметический сдвиг предварительного значения мантиссы C_M^* сопровождается изменением ранее найденного предварительного значения порядка результата C_E^* на плюс единицу. Находим окончательное значение порядка C_E:

$$\begin{array}{rl} & 0\ 0.\ 0\ 0\ 1 \quad -[C_E^*]\text{мдк;} \\ + & 0\ 0.\ 0\ 0\ 1 \quad -+1; \\ \hline & 0\ 0.\ 0\ 1\ 0 \quad -[C_E]\text{мдк} = [C_E]\text{мпк.} \end{array}$$

После устранения нарушения нормализации окончательный результат будет иметь вид C: $\{[C_E]_{\text{ПК}} = 0.010,\ [C_M]_{\text{ПК}} = 1.10001\}$.

Умножение чисел с плавающей точкой

С точки зрения представления чисел в форме с плавающей точкой поиск произведения C двух таких чисел A и B сводится к поиску произведения на основании порядка и мантиссы множимого и множителя. Для двоичных чисел это имеет вид

$$C = A \cdot B = 2^{A_E} \cdot A_M \cdot 2^{B_E} \cdot B_M = 2^{A_E + B_E} \cdot A_M \cdot B_M = 2^{C_E} \cdot C_M. \qquad (4.3)$$

Из (4.3) следует, что порядок произведения определяется как сумма порядков сомножителей, а мантисса произведения – как произведение мантисс сомножителей. Однако, учитывая то, что при умножении мантисс может произойти нарушение нормализации, в результате указанных действий будет найдено предварительное значение порядка и мантиссы искомого произведения и окончательное значение произведения будет найдено только после устранения нарушения нормализации.

Последовательность действий нахождения произведения двух чисел, представленных в форме с плавающей точкой, имеет следующий общий вид:

- определяется знак произведения как сумма по модулю двух знаковых разрядов мантисс сомножителей;
- определяется предварительное значение порядка произведения посредством суммирования порядков сомножителей;
- определяется предварительное значение мантиссы произведения как произведение мантисс сомножителей;
- устраняется нарушение нормализации мантиссы произведения (если нарушение имеет место) соответствующей корректировкой предварительного значения порядка и мантиссы искомого произведения;
- при формировании мантиссы произведения нормализованных чисел с плавающей точкой возможен только один вид нарушения нормализации – нарушение нормализации справа от точки с появлением нуля только в старшем разряде мантиссы.

Пример

Найти произведение C двух чисел A и B, представленных с плавающей точкой, если эти числа представлены в виде порядков $[A_E]_{ПК}$ = 1.010 и $[B_E]_{ПК}$ = 0.001 соответственно и мантисс $[A_M]_{ПК}$ = 1.1010 и $[B_M]_{ПК}$ = 0.1001 соответственно. При выполнении операций использовать обратный код. При умножении мантисс использовать метод умножения, начиная с младшего разряда множителя.

Сначала определяем знак искомого произведения. Так как знаки мантисс сомножителей неодинаковые, знак произведения будет отрицательным.

Предварительное значение порядка произведения C_E^* определяется как сумма порядков сомножителей $C_E^* = A_E + B_E$:

$$\begin{array}{rl} & 1.\ 1\ 0\ 1 \quad -[A_E]\text{ок}; \\ + & 0.\ 0\ 0\ 1 \quad -[B_E]\text{ок}; \\ \hline & 1.\ 1\ 1\ 0 \quad -[C_E^*]\text{ок}; \\ & 1.\ 0\ 0\ 1 \quad -[C_E^*]\text{пк}. \end{array}$$

Абсолютное значение предварительного значения мантиссы произведения C_M^* определяется как произведение модулей мантисс сомножителей:

$$
\begin{array}{rl}
 & 0.\ 1\ 0\ 1\ 0 \quad -|[A_M]|\text{ок;}\\
\times & 0.\ 1\ 0\ 0\ 1 \quad -|[B_M]|\text{ок;}\\
\hline
 & 0\ 1\ 0\ 1\ 0 \quad -\text{1-е частичное произведение;}\\
+ & 0\ 0\ 0\ 0\ 0 \quad -\text{2-е частичное произведение;}\\
+ & 0\ 0\ 0\ 0\ 0 \quad -\text{3-е частичное произведение;}\\
+ & 0\ 1\ 0\ 1\ 0 \quad -\text{4-е частичное произведение;}\\
+ & 0\ 0\ 0\ 0\ 0 \quad -\text{5-е частичное произведение;}\\
\hline
 & 0.\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0 \quad -|[C_M^*]|\text{ок} = |[C_M^*]|\text{пк.}
\end{array}
$$

Мантисса предварительного произведения ненормализованная, поэтому необходимо сдвинуть мантиссу влево на один разряд, а предварительное значение порядка произведения уменьшить на единицу.

Таким образом, с учетом определенного знака, нормализации и округления до четырех разрядов конечное значение мантиссы $[C_M]_{\text{ПК}} = 1.1011$.

С учетом нормализации конечное значение порядка $[C_E]_{\text{ПК}} = 1.010$.

Конечное значение произведения C: $\{[C_E]_{\text{ПК}} = 1.010, [C_M]_{\text{ПК}} = 1.1011\}$.

Деление чисел с плавающей точкой

С точки зрения представления чисел в форме с плавающей точкой поиск частного C двух таких чисел A и B сводится к поиску частного на основании порядка и мантиссы делимого и делителя. Для двоичных чисел это имеет вид

$$C = A : B = (2^{A_E} \cdot A_M) : (2^{B_E} \cdot B_M) = 2^{A_E - B_E} : (A_M \cdot B_M) = 2^{C_E} \cdot C_M. \quad (4.4)$$

Из (4.4) следует, что порядок частного определяется как разность порядков делимого и делителя, а мантисса – как частное от деления мантиссы делимого на мантиссу делителя. Однако, учитывая то, что при делении мантисс может произойти нарушение нормализации, в результате указанных действий будет найдено предварительное значения порядка и мантиссы искомого частного. Окончательные значения порядка и мантиссы частного будут определены после устранения нарушения нормализации в предварительном результате.

Последовательность действий нахождения частного двух чисел, представленных в форме с плавающей точкой, имеет следующий общий вид:

- определяется знак частного;
- определяется предварительное значение порядка частного как разность порядков делимого и делителя;
- определяется предварительное значение мантиссы частного как частное мантисс делимого и делителя;
- устраняется нарушение нормализации мантиссы частного (если нарушение имеет место) соответствующей корректировкой предварительного значения порядка и мантиссы искомого частного;
- при формировании мантиссы частного нормализованных чисел с плавающей точкой возможен только один вид нарушения нормализации – нарушение нормализации слева от точки.

Деление мантисс выполняется по правилам деления чисел с фиксированной точкой с восстановлением остатка или без восстановления остатка.

Деление с восстановлением остатка выполняется потактно за (n + 2) такта (n – разрядность модульного поля представления чисел). На каждом такте определяется один разряд частного.

На каждом такте выполняются следующие действия:

- из остатка, полученного на предыдущем такте (на первом такте из делимого), вычитается делитель (выполняется пробное вычитание) – тем самым формируется новый остаток;

- анализируется знак нового остатка. Если знак отрицательный, то осуществляется восстановление остатка, т. е. к полученному новому остатку прибавляется делитель. Если знак положительный, то восстановление остатка не происходит;

- если после пробного вычитания был получен положительный результат, то в очередном разряде формируемого частного устанавливается единица;

- выполняется умножение на два (арифметический сдвиг влево) нового или восстановленного остатка.

На первом такте определяется разряд целой части искомого частного. Для правильной дроби этот разряд должен иметь нулевое значение. Если на первом такте будет получен единичный разряд целой части искомого частного, то вырабатывается специальный сигнал о том, что искомое частное не является правильной дробью. После выполнения последнего (n + 2)-го такта анализируется последний найденный разряд частного, и если он равен единице, то в n-й разряд частного прибавляется единица.

Деление без восстановления остатка также выполняется за (n + 2) такта.

На каждом такте выполняются следующие действия:

- анализируется знак остатка (на первом такте анализируется знак делимого). Если знак положительный, то из остатка вычитается делитель, в противном случае делитель прибавляется. Таким образом формируется новый остаток;

- если новый остаток положительный, то в очередном разряде формируемого частного устанавливается единица, в противном случае – нуль;

- выполняется умножение на два (арифметический сдвиг влево) нового остатка.

Разряд частного, определенный на первом такте, также является разрядом целой части частного, и если он ненулевой, то вырабатывается сигнал о нарушении формы представления чисел в виде правильной дроби.

После выполнения последнего (n + 2)-го такта выполняется округление.

Пример

Найти частное C двух чисел A и B, представленных с плавающей точкой, если эти числа представлены в виде порядков $[A_E]_{ПК} = 1.010$ и $[B_E]_{ПК} = 0.001$ соответственно и мантисс $[A_M]_{ПК} = 1.1010$ и $[B_M]_{ПК} = 0.1001$ соответственно. При выполнении операций использовать модифицированный обратный код. При делении мантисс использовать метод деления без восстановления остатка.

Сначала определяем знак искомого частного. Так как знаки мантисс делимого и делителя неодинаковые, то знак частного будет отрицательным.

Предварительное значение порядка частного C_E^* определяется как разность порядков делимого и делителя $C_E^* = A_E - B_E$. Операция вычитания заменяется на операцию сложения с изменением знака вычитаемого на противоположный. Таким образом:

```
    1 1. 1 0 1  – [A_E]мок;
+   1 1. 1 1 0  – [–B_E]мок;
-----------------
  1 1 1. 1 1 0  – корректировка единицей в младший разряд
+             1   при переполнении разрядности знакового поля;
-----------------
    1 1. 1 0 0  – [C_E*]мок;
    1 1. 0 1 1  – [C_E*]мпк.
```

Абсолютное значение предварительного значения мантиссы частного C_M^* определяется за шесть тактов деления:

```
    0 0. 1 0 1 0  – [|A_M|]мок;
+   1 1. 0 1 1 0  – [–|B_M|]мок;
-----------------
  1 0 0. 0 0 0 0  – корректировка единицей в младший разряд при переполнении
+             1     разрядности знакового поля;
-----------------
    0 0. 0 0 0 1  – положительный остаток 1-го такта: 1-й разряд частного = 1;
    0 0. 0 0 1 0  – остаток после арифметического сдвига влево;
+   1 1. 0 1 1 0  – [–|B_M|]мок;
-----------------
    1 1. 1 0 0 0  – отрицательный остаток 2-го такта: 2-й разряд частного = 0;
    1 1. 0 0 0 1  – остаток после арифметического сдвига влево;
+   0 0. 1 0 0 1  – [|B_M|]мок;
-----------------
    1 1. 1 0 1 0  – отрицательный остаток 3-го такта: 3-й разряд частного = 0;
    1 1. 0 1 0 1  – остаток после арифметического сдвига влево;
+   0 0. 1 0 0 1  – [|B_M|]мок;
-----------------
    1 1. 1 1 1 0  – отрицательный остаток 4-го такта: 4-й разряд частного = 0;
    1 1. 1 1 0 1  – остаток после арифметического сдвига влево;
+   0 0. 1 0 0 1  – [|B_M|]мок;
-----------------
  1 0 0. 0 1 1 0  – корректировка единицей в младший разряд при переполнении
+             1     разрядности знакового поля;
-----------------
    0 0. 0 1 1 1  – положительный остаток 5-го такта: 5-й разряд частного = 1;
    0 0. 1 1 1 0  – остаток после арифметического сдвига влево;
+   1 1. 0 1 1 0  – [–|B_M|]мок;
-----------------
  1 0 0 0 1 0 0   – корректировка единицей в младший разряд при переполнении
+             1     разрядности знакового поля;
-----------------
    0 0. 0 1 0 1  – положительный остаток 6-го такта: 6-й разряд частного = 1;
    0 0. 1 0 1 0  – остаток после арифметического сдвига влево.
```

Таким образом, учитывая знаки остатков, полученных на шести тактах, абсолютное предварительное значение мантиссы искомого частного равно

$|C_M^*| = 1{,}00011$, а с учетом округления – $|C_M^*| = 1{,}0010$.

Мантисса частного ненормализованная (нарушение нормализации слева от точки, т. к. получена ненулевая целая часть), поэтому необходимо сдвинуть мантиссу вправо на один разряд, а предварительное значение порядка частного увеличить на единицу. После нормализации окончательное значение мантиссы и порядка частного с учетом определенных ранее знаков равны

$$C: \{[C_E]_{ПК} = 1.010, [C_M]_{ПК} = 1.1001\}.$$

4.3 Задачи для решения

Задание 1

Найти сумму C чисел A и B, представленных с плавающей точкой, если эти числа представлены в виде порядков A_E и B_E и мантисс A_M и B_M соответственно. При сложении использовать модифицированный дополнительный код:

а) $[A_E]_{ПК} = 1.010$, $[B_E]_{ПК} = 0.001$, $[A_M]_{ПК} = 1.1001$, $[B_M]_{ПК} = 0.0111$;

б) $[A_E]_{ПК} = 0.001$, $[B_E]_{ПК} = 1.010$, $[A_M]_{ПК} = 0.1101$, $[B_M]_{ПК} = 1.1010$.

Задание 2

Найти произведение C чисел A и B, представленных с плавающей точкой, если эти числа представлены в виде порядков A_E и B_E и мантисс A_M и B_M соответственно. При умножении использовать модифицированный обратный код:

а) $[A_E]_{ПК} = 1.001$, $[B_E]_{ПК} = 0.010$, $[A_M]_{ПК} = 1.0101$, $[B_M]_{ПК} = 0.1010$;

б) $[A_E]_{ПК} = 0.010$, $[B_E]_{ПК} = 1.001$, $[A_M]_{ПК} = 0.1001$, $[B_M]_{ПК} = 1.0111$.

Задание 3

Найти частное C чисел A и B, представленных с плавающей точкой, если эти числа представлены в виде порядков A_E и B_E и мантисс A_M и B_M соответственно. При делении использовать модифицированный обратный код:

а) $[A_E]_{ПК} = 1.010$, $[B_E]_{ПК} = 0.001$, $[A_M]_{ПК} = 1.1100$, $[B_M]_{ПК} = 0.0010$;

б) $[A_E]_{ПК} = 0.100$, $[B_E]_{ПК} = 1.010$, $[A_M]_{ПК} = 0.0101$, $[B_M]_{ПК} = 1.0011$.

4.4 Контрольные вопросы

1 Каким образом осуществляется выравнивание порядков при сложении чисел, представленных в форме с плавающей точкой?

2 Каким образом осуществляется нормализация при умножении чисел, представленных в форме с плавающей точкой?

3 Какой вид нарушения нормализации имеет место быть при умножении и какой – при делении чисел, представленных в форме с плавающей точкой?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №5
СИНТЕЗ ЛОГИЧЕСКИХ СХЕМ

Цель: приобрести практические навыки решения задач синтеза логических схем в заданном булевом базисе.

5.1 Краткие теоретические сведения

При синтезе и анализе схем ЭВМ широкое применение находит алгебра логики. Двоичное представление информации соответствует характеру работы отдельных компонент вычислительной техники, например, низкому или высокому сигналу на выходе компоненты.

Основные понятия алгебры логики:

- *логическая переменная* – это такая переменная, которая может принимать одно из двух значений: истинно или ложно (да или нет, единица или нуль);

- *логическая константа* – это такая постоянная величина, значением которой может быть одно из двух значений: истинно или ложно (да или нет, единица или нуль);

- *логическая функция* – это такая функция, которая может принимать одно из двух значений: истинно или ложно (да или нет, единица или нуль) в зависимости от текущего значения ее аргументов, в качестве которых используются логические переменные.

Количество аргументов (n) может быть разным, например: при $n = 1$ – функция одного аргумента, при $n > 1$ – функция нескольких аргументов.

Зависимость логической функции от переменных (аргументов) может быть задана в виде таблицы истинности, словесного описания или в виде логического выражения.

Таблица истинности является универсальным средством задания логической функции. Она включает все наборы для заданного количества переменных, определяющих значение логической функции, с указанием значений, которые принимает функция для каждого набора. В одной таблице истинности может задаваться несколько логических функций, зависящих от одних и тех же переменных.

Словесное описание может использоваться в случае сравнительно несложной логической функции.

Логическим выражением называется комбинация логических переменных и констант, связанных элементарными базовыми логическими функциями (или логическими операциями), которые могут разделяться скобками. Например, логическую функцию y_1 можно представить в виде логического выражения

$$y_1 = \overline{(x_1 \cdot x_2 + x_1 \cdot x_3 + x_2 \cdot x_3)} \cdot (x_1 + x_2 + x_3) + x_1 \cdot x_2 \cdot x_3, \qquad (5.1)$$

где «+», «·», а также верхняя черта – знаки базовых логических функций.

Набор элементарных логических операций, с помощью которых можно задать любую, сколь угодно сложную логическую функцию, называется *функ-*

ционально полной системой логических функций. Иногда такую систему называют *базисом*. В качестве элементарных логических функций функционально полных систем логических функций используются функции одной или двух логических переменных.

Функции одной переменной приведены в таблице 5.1.

Таблица 5.1 – Функции одной переменной

x	y_0	y_1	y_2	y_3
0	0	0	1	1
1	0	1	0	1

Если записать функции одной переменной в виде логических выражений, то получим

$$y_0 = 0, \tag{5.2}$$

где значение y_0 является константой;

$$y_1 = x, \tag{5.3}$$

где y_1 равняется значению переменной;

$$y_2 = \bar{x}, \tag{5.4}$$

где y_2 – равняется инверсному значению переменной;

$$y_3 = 1, \tag{5.5}$$

где y_3 – константа.

Все возможные варианты функций двух переменных приведены в таблице 5.2. Информация по функциям двух переменных приведена в таблице 5.3.

Наиболее распространенной в алгебре логики является функционально полная система логических функций, которая в качестве базовых логических функций использует функцию одной переменной НЕ (функция отрицания) и две функции двух переменных: И (конъюнкция или логическое умножение) и ИЛИ (дизъюнкция или логическое сложение). Эта система называется *системой булевых функций*, или *булев базис*. В алгебре логики имеется целый раздел «Алгебра Буля», посвященный этому базису.

Таблица 5.2 – Функции двух переменных

№ п/п	x_1	x_2	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
2	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
3	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Таблица 5.3 – Описание функций двух переменных

y_i	Название функции	Чтение функции	Запись в виде булева выражения
y_0	Константа 0	Тождественный нуль	0
y_1	Конъюнкция	И x_1, и x_2	$x_1 \cdot x_2$ (x_1x_2, $x_1 \vee x_2$)
y_2	Запрет по x_2	Неверно, что если x_1, то x_2	$x_1\overline{x_2}$
y_3	$f(x_1)$	Функция одной переменной	x_1
y_4	Запрет по x_1	Неверно, что если x_2, то x_1	$\overline{x_1}x_2$
y_5	$f(x_2)$	Функция одной переменной	x_2
y_6	Функция неравнозначности	x_1 не равно x_2	$\overline{x_1}x_2 + x_1\overline{x_2}$
y_7	Дизъюнкция	Или x_1, или x_2	$x_1 + x_2$
y_8	Функция Пирса	Ни x_1, ни x_2	$\overline{x_1 + x_2}$
y_9	Функция равнозначности	x_1 равно x_2	$\overline{x_1x_2} + x_1x_2$
y_{10}	$f(x_2)$	Функция одной переменной	$\overline{x_2}$
y_{11}	Импликация	Если x_2, то x_1	$\overline{x_1x_2} + x_1$
y_{12}	$f(x_1)$	Функция одной переменной	$\overline{x_1}$
y_{13}	Импликация	Если x_1, то x_2	$\overline{x_1x_2} + x_2$
y_{14}	Функция Шеффера	Неверно, что и x_1, и x_2	$\overline{x_1x_2}$
y_{15}	Константа 1	Тождественная единица	1

При работе с булевыми логическими выражениями используются следующие законы и правила:

- переместительный (коммутативный) закон:

$$x_1x_2 = x_2x_1;$$
$$x_1 + x_2 = x_2 + x_1;$$

- сочетательный (ассоциативный) закон:

$$(x_1x_2)x_3 = x_1(x_2x_3);$$
$$x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3;$$

- распределительный (дистрибутивный) закон:

$$x_1(x_2 + x_3) = x_1x_2 + x_1x_3;$$

- правило де Моргана:

$$\overline{x_1x_2} = \overline{x_1} + \overline{x_2};$$
$$\overline{x_1 + x_2} = \bar{x}_1\bar{x}_2;$$

- операция склеивания:

$$x_iA + \overline{x_i}A = A,$$
$$(x_i + B)(\overline{x_i} + B) = B,$$

где A – логическое произведение переменных и их отрицаний;

B – логическая сумма переменных и их отрицаний;

- операции с отрицаниями:

$$\bar{\bar{x}} = x;$$
$$\bar{x} \cdot x = 0;$$
$$\bar{x} + x = 1;$$

- операции с константами:

$$x + 1 = 1;$$
$$x \cdot 1 = x;$$
$$x + 0 = x;$$
$$x \cdot 0 = 0;$$

- операции с одинаковыми операндами:

$$x + x + \cdots + x = x;$$
$$x \cdot x \cdot \ldots \cdot x = x.$$

5.2 Методические указания и примеры решения задач

Логические схемы строятся на основе логических элементов, набор которых определяется заданным логическим базисом.

Для базиса Буля в качестве логических элементов используются элементы, реализующие базовые логические функции И, ИЛИ, НЕ, которые имеют приведенные на рисунке 5.1 графические обозначения. При синтезе схемы по логическому выражению логические операции представляются в виде соответствующих графических обозначений логических элементов, а связи между ними определяются последовательностью выполнения логических операций в заданном выражении.

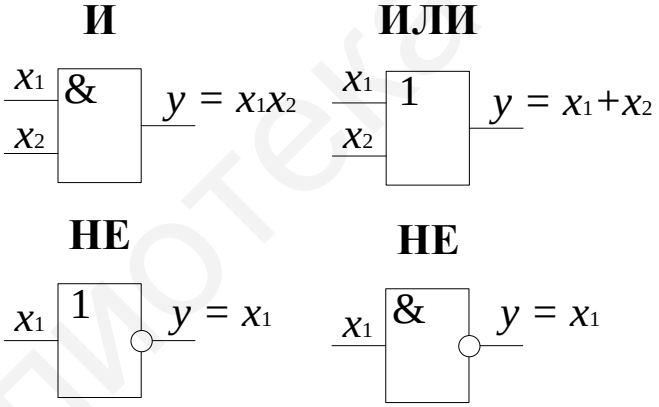


Рисунок 5.1 – Графические обозначения логических элементов

Пример

Синтезировать логическую схему в базисе И, ИЛИ, НЕ, реализующую логическое выражение

$$y_1 = \overline{(\overline{x_1 x_2} + x_3)} \cdot (x_1 + x_2 x_3) + x_1 \overline{x_3}. \quad (5.6)$$

Входными сигналами синтезируемой схемы являются x_1, x_2, x_3, а выходным – y_1. Начинать операцию представления данного выражения в виде схемы можно либо с последней операции, либо с первой. Рассмотрим первый вариант.

Шаг 1. Логическое выражение (5.6) можно представить в виде

$$y_1 = t_{1,1} + t_{1,2},$$

где

$$t_{1,1} = \overline{(\overline{x_1 x_2} + x_3)} \cdot (x_1 + x_2 x_3); \quad (5.7)$$

$$t_{1,2} = x_1 \overline{x_3}. \quad (5.8)$$

Для реализация этой операции потребуется элемент ИЛИ с двумя входами, на которые будут поданы сигналы $t_{1,1}$ и $t_{1,2}$, а на выходе данного элемента будет сформирован сигнал y_1 (рисунок 5.2).

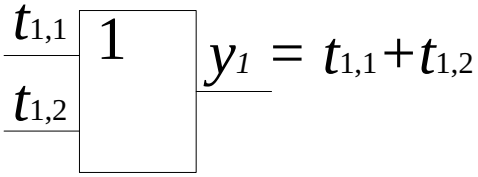


Рисунок 5.2 – Результат синтеза логического выражения на первом шаге

Шаг 2. На данном этапе рассмотрим операцию для формирования сигналов $t_{1,1}$ и $t_{1,2}$. Выражение (5.7) можно переписать следующим образом:

$$t_1 = t_{2,1}t_{2,2}, \tag{5.9}$$

где
$$t_{2,1} = \overline{\overline{x_1 x_2} + x_3}; \tag{5.10}$$

$$t_{2,2} = x_1 + x_2 x_3. \tag{5.11}$$

Для реализации такой операции потребуется элемент И с двумя входами, на которые будут поданы сигналы $t_{2,1}$ и $t_{2,2}$, а на выходе сформирован сигнал $t_{1,1}$ (рисунок 5.3).

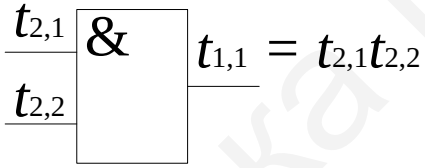


Рисунок 5.3 – Результат синтеза логического выражения для сигнала $t_{1,1}$

Выражение (5.8) можно переписать следующим образом:

$$t_{1,2} = t_{2,3}t_{2,4},$$

где $t_{2,3} = x_1$;

$t_{2,4} = \overline{x_3}$.

Для реализации такой операции потребуется элемент И с двумя входами, на которые будут поданы сигналы $t_{2,3}$ $t_{2,4}$, а на выходе сформирован сигнал $t_{1,2}$ (рисунок 5.4).

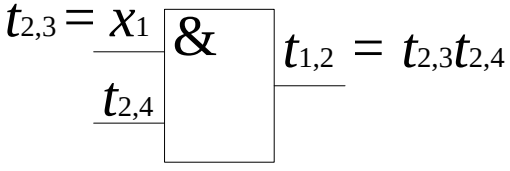


Рисунок 5.4 – Результат синтеза логического выражения для сигнала $t_{1,2}$

Таким образом, синтезируемая логическая схема после выполнения двух шагов примет вид, изображенный на рисунке 5.5.

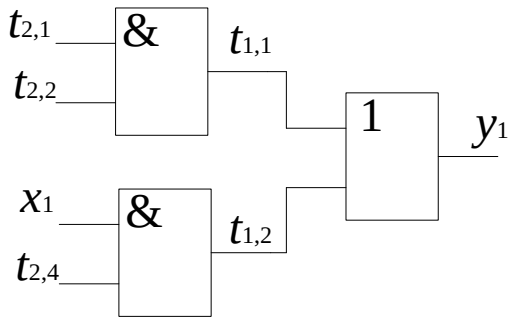


Рисунок 5.5 – Промежуточный результат синтеза логического выражения

Шаг 3 и последующие выполняются по аналогии с предыдущими шагами.

В результате сигнал $t_{2.1}$ можно получить с помощью логического элемента НЕ и записать в виде выражения

$$t_{2,1} = \overline{t_{3,1}}, \tag{5.12}$$

где

$$t_{3,1} = \overline{x_1 x_2} + x_3. \tag{5.13}$$

Сигнал $t_{2,2}$ можно записать как

$$t_{2,2} = t_{3,2} + t_{3,3}, \tag{5.14}$$

где

$$t_{3,2} = x_1; \tag{5.15}$$

$$t_{3,3} = x_2 x_3. \tag{5.16}$$

Сигнал $t_{2,4}$ получается на выходе элемента НЕ, на вход которого подается сигнал x_3.

В свою очередь выражение (5.13) можно представить как

$$t_{3,1} = t_{4,1} + t_{4,2}, \tag{5.17}$$

где

$$t_{4,1} = \overline{x_1 x_2}; \tag{5.18}$$

$$t_{4,2} = x_3. \tag{5.19}$$

В результате синтезирования сигнал $t_{3,1}$ будет получен на выходе элемента ИЛИ. Сигнал $t_{3,3}$ (см. выражение (5.16)) будет получен на выходе элемента И, на входы которого подаются сигналы x_2 и x_3.

Для формирования сигнала $t_{4,1}$ необходимо применить элемент НЕ, на вход которого подается сигнал t_5:

$$t_{4,1} = \overline{t_5}. \tag{5.20}$$

Сигнал t_5 формируется на выходе элемента И, на входы которого подаются сигналы x_1 и x_2. В конечном результате функция y_1 примет вид, представленный на рисунке 5.6.

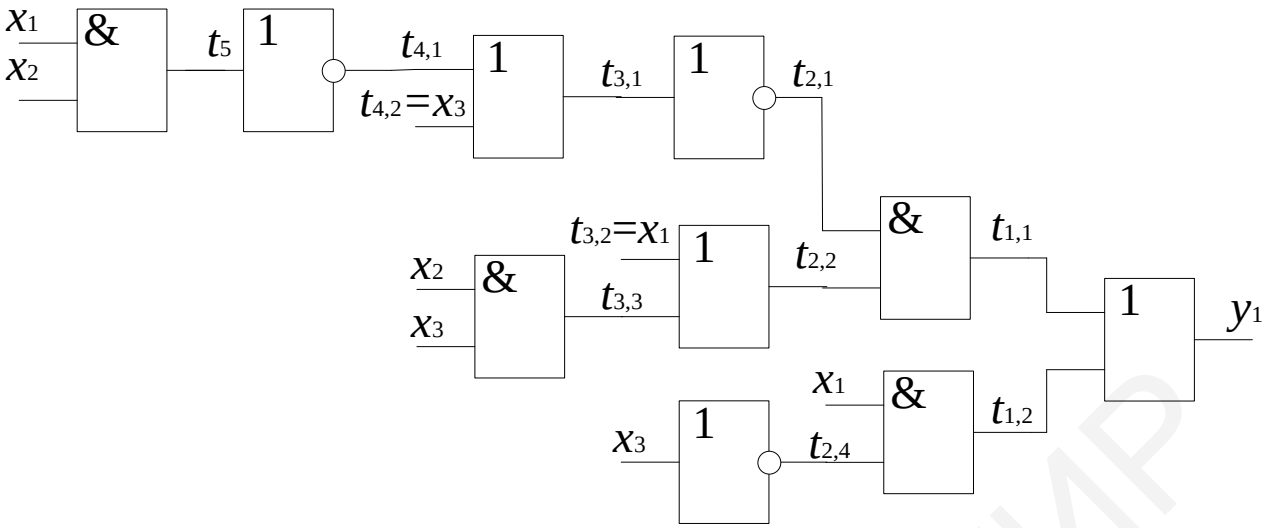


Рисунок 5.6 – Результат синтеза логического выражения

Чтобы привести схему к более понятному виду, входные сигналы можно задать из одной точки. Тогда схема примет вид, представленный на рисунке 5.7.

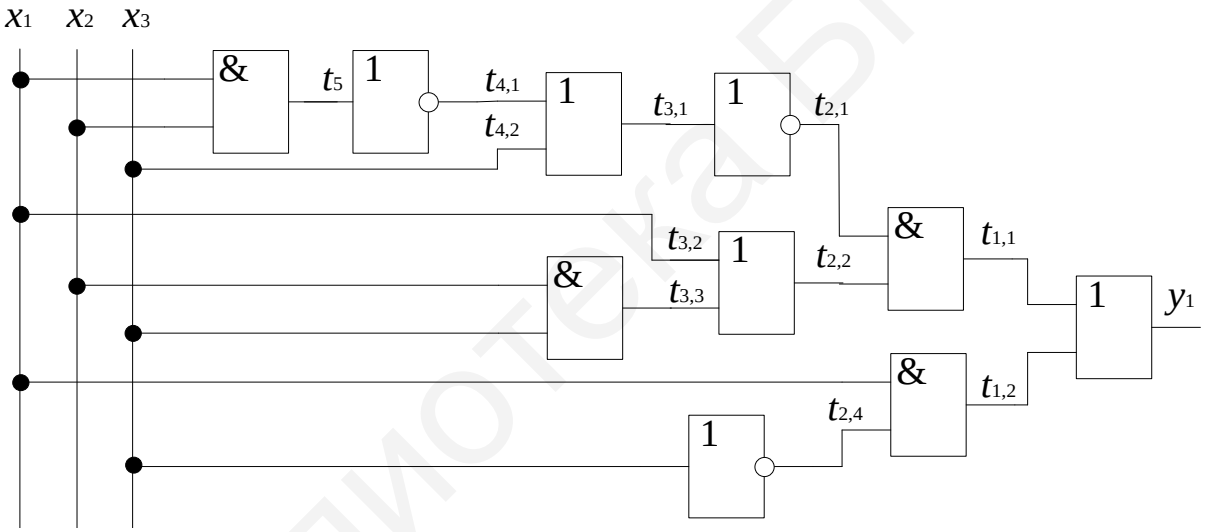


Рисунок 5.7 – Результат синтеза логического выражения

Для решения задачи синтеза в базисе И-НЕ необходимо использовать базовый элемент, приведенный на рисунке 5.8.

И-НЕ

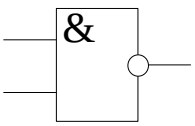


Рисунок 5.8 – Базовый элемент для синтеза функции в базисе И-НЕ

Таким образом, логические операции ИЛИ, И в базисе И-НЕ исходя из правила де Моргана реализуются следующим образом:

$$x_1 + x_2 = \overline{\overline{x_1 + x_2}} = \overline{\bar{x}_1 \bar{x}_2}; \quad (5.21)$$

$$x_1 \cdot x_2 = \overline{\overline{x_1 x_2}}. \quad (5.22)$$

Графическое представление базовых логический операций в базисе И-НЕ имеет вид, представленный на рисунке 5.9.

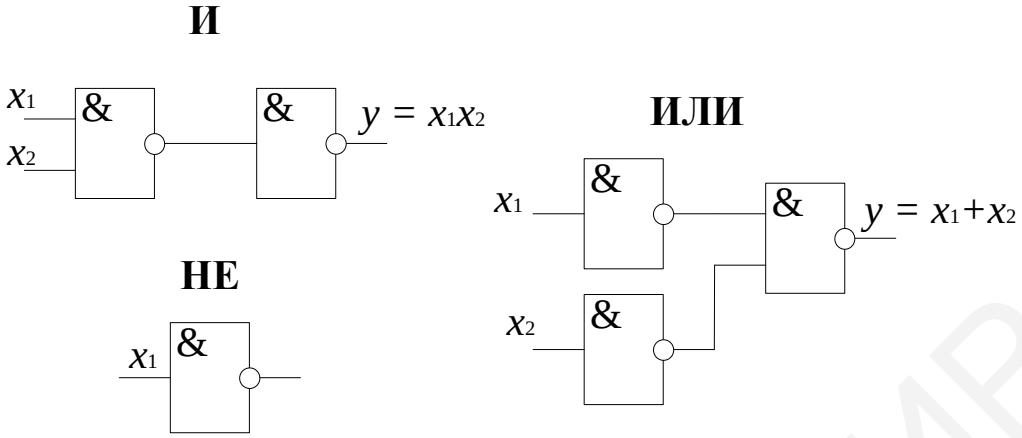


Рисунок 5.9 – Графическое представление логических операций И, ИЛИ, НЕ в базисе И-НЕ

Схемная реализация базового элемента для решения задачи синтеза в базисе ИЛИ-НЕ представлена на рисунке 5.10.

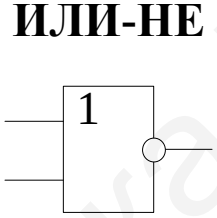


Рисунок 5.10 – Схемная реализация базового элемента ИЛИ-НЕ

Логическая операция И в базисе ИЛИ-НЕ примет вид

$$x_1 \cdot x_2 = \overline{\overline{x_1 \cdot x_2}} = \overline{\bar{x}_1 + \bar{x}_2}.$$

Логическая операция ИЛИ реализуется следующим образом:

$$x_1 + x_2 = \overline{\overline{x_1 + x_2}}.$$

На рисунке 5.11 приведены графические изображения логических элементов в базисе ИЛИ-НЕ.

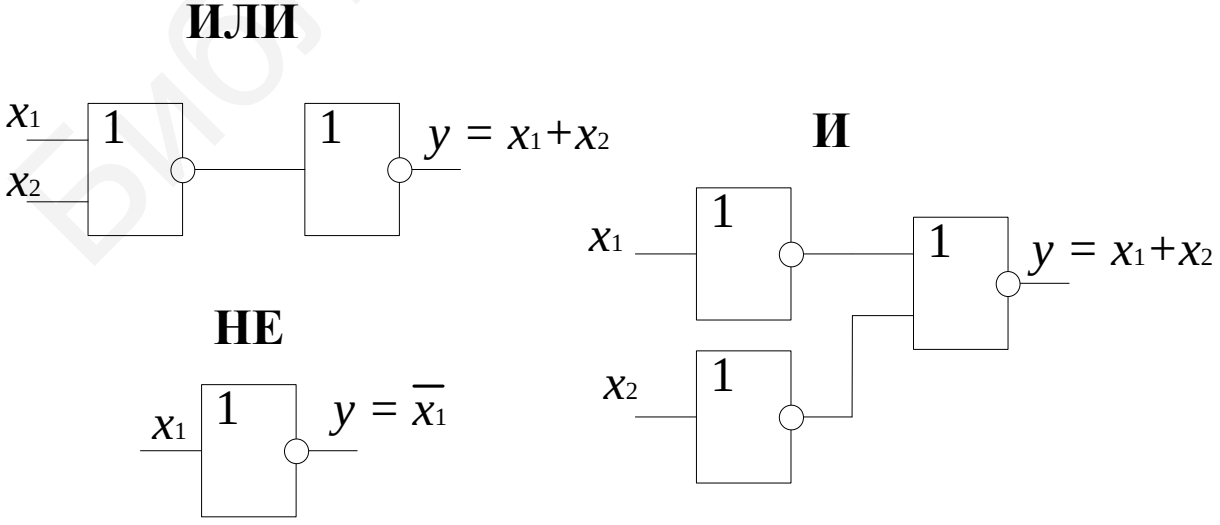


Рисунок 5.11 – Схемная реализация логических операций И, ИЛИ, НЕ в базисе ИЛИ-НЕ

Пример

Синтезировать логическое выражение (5.6) в базисе И-НЕ.

Используя правило де Моргана, преобразуем исходное выражение таким образом, чтобы остались только логические операции И, НЕ и последней была операция отрицания

$$y_1 = \overline{(\overline{x_1x_2} + x_3)} \cdot (x_1 + x_2x_3) + x_1\overline{x_3} = \overline{\overline{(\overline{x_1x_2} \cdot \overline{x_3})} \cdot \overline{(\overline{x_1} \cdot \overline{x_2x_3})} \cdot \overline{x_1\overline{x_3}}}. \quad (5.23)$$

Полученное выражение позволяет синтезировать соответствующую схему в базисе И-НЕ (рисунок 5.12).

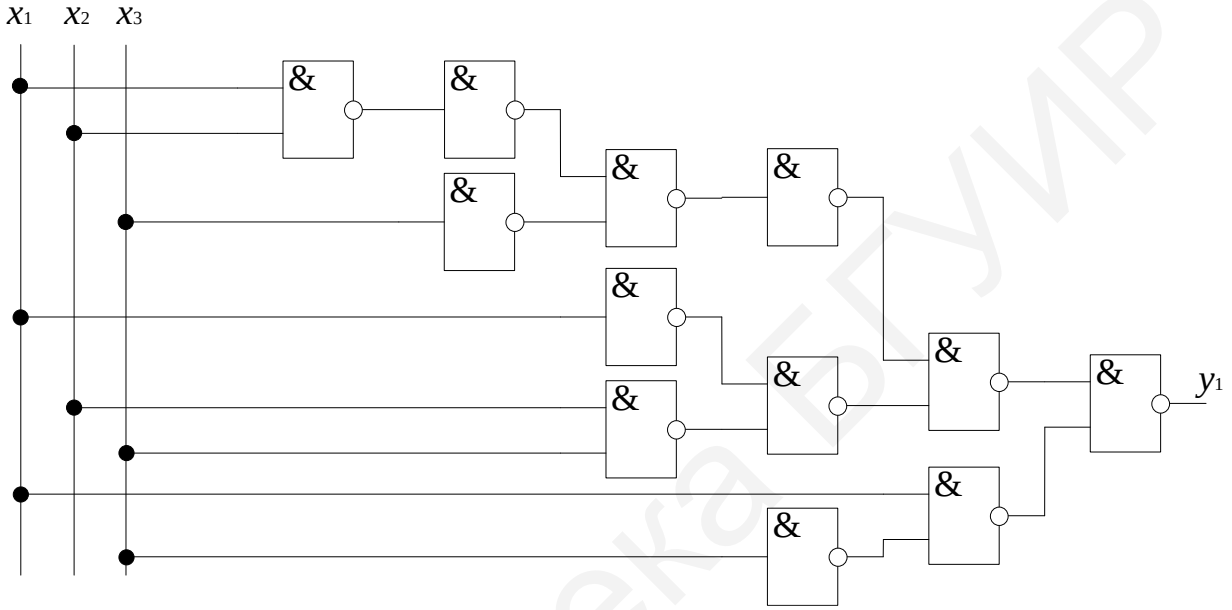


Рисунок 5.12 – Результат синтеза выражения в базисе И-НЕ

Пример

Синтезировать логическое выражение (5.6) в базисе ИЛИ-НЕ.

Используя правило де Моргана, преобразуем исходное выражение таким образом, чтобы остались только логические операции ИЛИ, НЕ, и последней была операция отрицания

$$y_1 = \overline{(\overline{x_1x_2} + x_3)} \cdot (x_1 + x_2x_3) + x_1\overline{x_3} = \overline{\left(\overline{\overline{x_1} + \overline{x_2}} + x_3\right) + (x_1 + \overline{\overline{x_2} + \overline{x_3}}) + \overline{\overline{x_1} + x_3}}. \quad (5.24)$$

Полученное выражение позволяет синтезировать соответствующую схему в базисе ИЛИ-НЕ (рисунок 5.13).

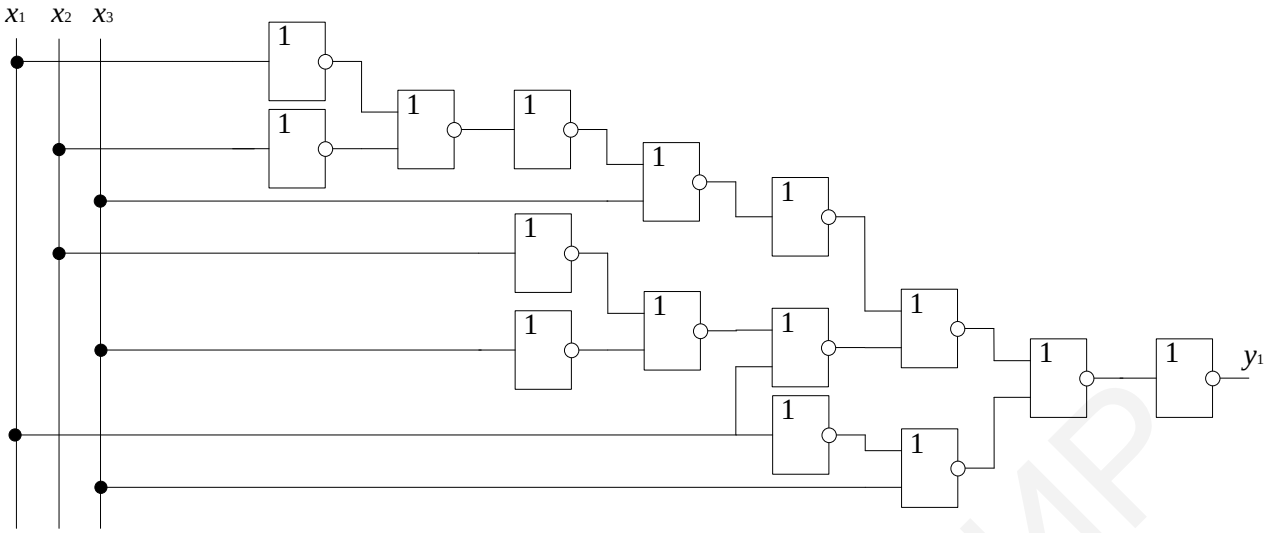


Рисунок 5.13 – Результат синтеза выражения в базисе ИЛИ-НЕ

5.3 Задачи для решения

Задание 1

Реализовать заданные выражения в виде логических схем, состоящих из функционально полного набора логических элементов И, ИЛИ, НЕ (см. варианты задач, приведенных ниже).

Задание 2

Синтезировать логические схемы в базисе И-НЕ, реализующие заданные логические выражения (см. варианты задач, приведенных ниже).

Задание 3

Синтезировать логические схемы в базисе ИЛИ-НЕ, реализующие заданные логические выражения (см. варианты задач, приведенных ниже).

Варианты задач:

1) $y_1 = x_1 + x_2 + \overline{\overline{x_1 x_3} + x_2}$;
2) $y_2 = \overline{\overline{x_1 x_2} + x_1 x_3} + x_3$;
3) $y_3 = \overline{x_1 \overline{x_2}} x_3 + \overline{x_1 + x_3}$;
4) $y_4 = x_1 \overline{x_2} + (x_4 + x_1)\overline{x_3}$;
5) $y_5 = \overline{\overline{x_3}(\overline{x_1} + x_2)}(\overline{x_2} + x_3)$;
6) $y_6 = \overline{(x_3 + x_1)} + \overline{x_1 \bar{x}_2} x_3$;
7) $y_7 = x_2 \bar{x}_3 + (x_1 + x_2) x_4$;
8) $y_8 = \overline{x_4 + x_3 + \overline{x_2 (x_4 + x_1)}}$;
9) $y_9 = \overline{x_2 + \bar{x}_4 \bar{x}_1} + \overline{x_3 x_2}$;
10) $y_{10} = (x_2 \overline{x_3 x_4} + \bar{x}_1) \cdot \bar{x}_4$;
11) $y_{11} = \overline{x_1 + \bar{x}_2} \cdot \overline{(x_3 + \bar{x}_4) \cdot \bar{x}_1}$;
12) $y_{12} = \overline{(x_4 + x_1)} + x_1 (\bar{x}_2 + \bar{x}_3)$;

13) $y_{13} = \overline{x_4\bar{x}_1(x_2 + x_4)} + x_3$;
14) $y_{14} = \bar{x}_3 + \overline{(\bar{x}_1\bar{x}_2 + x_3)}x_4$;
15) $y_{15} = \overline{(x_3 + \overline{x_4x_1})}x_2 + x_4$.

5.4 Контрольные вопросы

1 Назовите особенности синтеза логических схем по логическим выражениям в базисе И-НЕ.

2 Назовите особенности синтеза логических схем по логическим элементам в базисе ИЛИ-НЕ.

3 Приведите алгоритм синтеза логических схем по логическим элементам в заданном булевом базисе.

4 Назовите и поясните законы и правила алгебры Буля, правило де Моргана.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №6
МИНИМИЗАЦИЯ ЛОГИЧЕСКИХ ВЫРАЖЕНИЙ

Цель: получить практические навыки в решении задач минимизации логических функций методом с использованием карт Карно.

6.1 Краткие теоретические сведения

Существует множество различных вариантов выражений, соответствующих одной логической функции, поэтому важным этапом перед реализацией функции в виде логической схемы является выбор оптимального выражения. Решить проблему такого выбора позволяет процедура минимизации логического выражения. Одним из самых часто используемых методов является минимизация методом с использованием карт Карно (или диаграмм Вейча).

Карта Карно для *n* логических переменных представляет собой прямоугольную таблицу, приближенную к форме квадрата. Каждая ячейка таблицы соответствует одному набору логических переменных. Две соседние ячейки должны соответствовать наборам, различающимся значениями одной переменной.

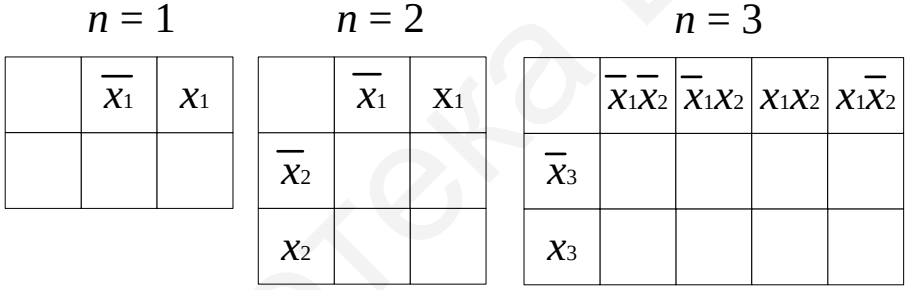


Рисунок 6.1 – Примеры карт Карно при *n* = 1, 2, 3 логических элемента

Исходная функция должна быть представлена в одной из канонических форм: либо совершенной дизъюнктивной нормальной форме (СДНФ), либо совершенной конъюнктивной нормальной форме (СКНФ).

Совершенная дизъюнктивная нормальная форма – это форма, которая представляет собой дизъюнкцию простых конъюнкций. В СДНФ простые конъюнкции содержат все переменные в своей прямой или инверсной форме и отражают собой наборы, на которых функция принимает единичные значения. Такие конъюнкция называются конституентами единицы.

Совершенная конъюнктивная нормальная форма – это форма, которая представляет собой конъюнкцию простых дизъюнкций. В СКНФ простые дизъюнкции содержат все переменные в своей прямой или инверсной форме и отражают собой инверсию конституент нуля. Конституента нуля – набор (конъюнкция переменных в прямой или инверсной форме), на котором функция принимает нулевые значения.

СДНФ и СКНФ легко сформировать на основе таблицы истинности. Например, функции заданы на основе таблицы 6.1.

Таблица 6.1 – Табличное задание функций y_1, y_2

№	x_1	x_2	x_3	y_1	y_2
0	0	0	0	1	1
1	0	0	1	1	0
2	0	1	0	0	1
3	0	1	1	1	1
4	1	0	0	1	1
5	1	0	1	0	0
6	1	1	0	0	1
7	1	1	1	0	0

СДНФ в таком случае примет вид

$$y_1 = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2x_3 + x_1\bar{x}_2\bar{x}_3; \quad (6.1)$$

$$y_2 = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 + \bar{x}_1x_2x_3 + x_1x_2\bar{x}_3 + x_1\bar{x}_2\bar{x}_3. \quad (6.2)$$

СКНФ примет следующий вид:

$$y_1 = (x_1 + \bar{x}_2 + x_3) \cdot (\bar{x}_1 + \bar{x}_2 + x_3) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_3) \cdot (\bar{x}_1 + x_2 + \bar{x}_3); \quad (6.3)$$

$$y_2 = (x_1 + x_2 + \bar{x}_3) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_3) \cdot (\bar{x}_1 + x_2 + \bar{x}_3). \quad (6.4)$$

В процессе минимизации может возникнуть задача перехода между основными каноническими формами записи функции. Например, если функция задана в СДНФ, требуется найти ее СКНФ. Такой переход можно выполнить, составив по заданной СДНФ таблицу истинности для необходимой функции, а на основе полученной таблицы составить СКНФ. Данную задачу можно решить также другим способом – при использовании правила де Моргана.

6.2 Методические указания и примеры решения задач

Для различных форм записи СДНФ и СКНФ карты Карно заполняются схожим образом.

Сначала рассмотрим пример с СДНФ.

Например, минимизировать функцию y_1 (выражение (6.1)) с помощью карт Карно. Заданная функция является функцией трех переменных, поэтому карта Карно примет вид, представленный на рисунке 6.2.

	$\bar{x}_1\bar{x}_2$	$\bar{x}_1x_2$	x_1x_2	$x_1\bar{x}_2$
$\bar{x}_3$				
x_3				

Рисунок 6.2 – Карта Карно для функции трех переменных

Рассмотрим функцию по конституентам. Первая конституента представляет собой набор $\bar{x}_1\bar{x}_2\bar{x}_3$, следовательно, в ячейку соответствующую данному набору необходимо установить единицу (рисунок 6.3).

	$\bar{x}_1\bar{x}_2$	$\bar{x}_1x_2$	x_1x_2	$x_1\bar{x}_2$
$\bar{x}_3$	1			
x_3				

Рисунок 6.3 – Запись конституенты $\bar{x}_1\bar{x}_2\bar{x}_3$ в соответствующий набор

Те же действия проведем с остальными конституентами. Итоговая запись функции в карту Карно будет иметь вид, приведенный на рисунке 6.4.

	$\bar{x}_1\bar{x}_2$	$\bar{x}_1x_2$	x_1x_2	$x_1\bar{x}_2$
$\bar{x}_3$	1			1
x_3	1	1		

Рисунок 6.4 – Запись функции в карту Карно

Для минимизации функции, представленной в карте Карно, необходимо охватить множество ячеек карты Карно, а затем записать минимальное выражение для заданной функции.

Охват ячеек карты контурами выполняется с соблюдением правил:

- контур должен иметь прямоугольную форму;
- в контур может входить такое количество ячеек, которое равно целой степени числа два;
- в контур могут входить ячейки, являющиеся логическими соседями;
- в контур необходимо включить максимальное количество ячеек с учетом вышеприведенных требований;
- контурами необходимо охватить все ячейки с единичными значениями;
- контуров должно быть минимальное количество.

Логическими соседями являются такие две ячейки, наборы которых отличаются только одной переменной: в одной эта переменная должна иметь прямое, в другой – обратное значение. Следует отметить, что логическими соседями также являются ячейки, располагающиеся в соответствующих строках крайнего левого столбца и крайнего правого столбца, а также в соответствующих столбцах крайней верхней и крайней нижней строках.

Запись минимального выражения для заданной функции формируется таким образом, чтобы конъюнкция, соответствующая контуру, включала только те переменные, которые имеют постоянное значение во всех клетках, охваченных рассматриваемым контуром. Контуры для функции y_1 имеют вид, представленный на рисунке 6.5.

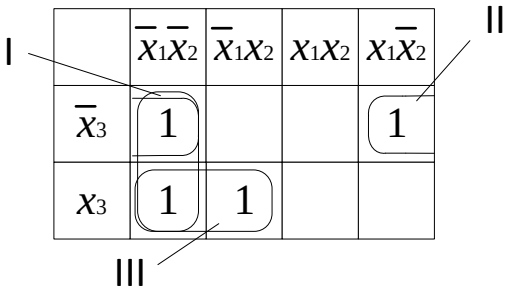


Рисунок 6.5 – Контуры для функции y_1, представленной в виде карт Карно

Контур I представляет собой две ячейки, соответствующие наборам $\bar{x}_1\bar{x}_2\bar{x}_3$ и $\bar{x}_1\bar{x}_2x_3$. Постоянными переменными для ячеек в данном контуре являются $\bar{x}_1\bar{x}_2$. Тогда минимальное логическое выражение для контура I примет вид

$$\bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 = \bar{x}_1\bar{x}_2.$$

Минимальное логическое выражение для функции y_1 примет вид

$$y_1 = \bar{x}_1\bar{x}_2 + \bar{x}_1x_3 + \bar{x}_2\bar{x}_3, \qquad (6.5)$$

где $\bar{x}_1\bar{x}_2$ – минимальное логическое выражение для контура I;

$\bar{x}_1x_3$ – минимальное логическое выражение для контура II;

$\bar{x}_2\bar{x}_3$ – минимальное логическое выражение для контура III.

Полученное выражение (6.5) является результатом минимизации функции y_1 – методом с использованием карт Карно.

Рассмотрим пример, когда исходная функция задана в форме СКНФ.

Необходимо выполнить минимизацию функции y_2 с использованием карт Карно. Алгоритм минимизации в этом случае похож на предыдущий. В качестве наборов, соответствующим ячейкам карт Карно, используются простые дизъюнкции, а в соответствующие ячейки устанавливаются значения нуль:

$$y_2 = (x_1 + x_2 + \bar{x}_3) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_3) \cdot (\bar{x}_1 + x_2 + \bar{x}_3). \qquad (6.6)$$

Карта Карно для данной функции примет вид, изображенный на рисунке 6.6.

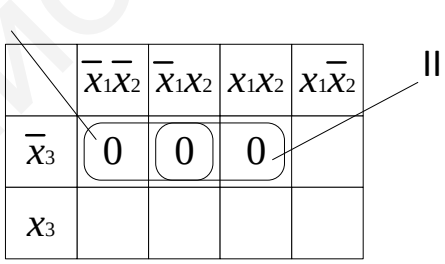


Рисунок 6.6 – Карта Карно для функции y_2, представленной в форме СКНФ

На полученной карте Карно можно выделить два контура. Таким образом, в результате минимизации получаем следующую функцию:

$$y_2 = (\bar{x}_1 + \bar{x}_3) \cdot (x_2 + \bar{x}_3). \qquad (6.7)$$

Пример

Выполнить минимизацию функции *f*, заданной в числовой форме $f(x_1x_2x_3x_4) = (0,2,5,7,8,10,13,15)$, на единичных наборах. Записать минимальную функцию в дизъюнктивной нормальной форме (ДНФ).

Если представить исходную функцию в виде таблицы истинности, она примет вид, представленный в таблице 6.2.

Таблица 6.2 – Функция $f(x_1x_2x_3x_4)$, заданная в виде таблицы истинности

Номер набора	x_1	x_2	x_3	x_4	$f(x_1x_2x_3x_4)$
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

По условию задачи необходимо записать минимальную функцию в форме ДНФ, поэтому представим исходную функцию в форме СДНФ:

$$f(x_1x_2x_3x_4) = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 + \bar{x}_1\bar{x}_2x_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3x_4 + \\ +\bar{x}_1x_2x_3x_4 + x_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3\bar{x}_4 + x_1x_2\bar{x}_3x_4 + x_1x_2x_3x_4. \quad (6.8)$$

Разместим простые конъюнкции на карте Карно (рисунок 6.7).

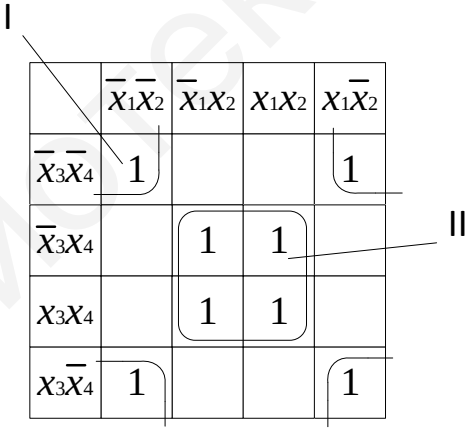


Рисунок 6.7 – Карта Карно для функции $f(x_1x_2x_3x_4)$

В итоге минимальная функция примет вид

$$f(x_1x_2x_3x_4) = \bar{x}_2\bar{x}_4 + x_2x_4. \quad (6.9)$$

В случае, когда результат необходимо записать в конъюнктивной нормальной форме (КНФ), необходимо либо изначально по таблице истинности записать СКНФ исходной функции, либо прибегнуть к решению задачи перехода из одной канонической формы к другой.

Переход от формы записи выражения в СДНФ к форме записи в СКНФ, кроме способа с использования таблицы истинности, может быть реализован при помощи правила де Моргана.

Пример

По заданной СДНФ функции $y_3 = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 + \bar{x}_1x_2x_3 + x_1x_2\bar{x}_3 + + x_1\bar{x}_2\bar{x}_3$ найти запись этой функции в СКНФ.

Сначала необходимо записать отрицание данной функции, т. е. записать дизъюнкцию простых конъюнкций, где простые конъюнкции представляют собой конституенты нуля. В свою очередь конституенты нуля – это те наборы, которые не являются наборами конституент единиц:

$$\overline{y_3} = \bar{x}_1\bar{x}_2x_3 + x_1x_2x_3 + x_1\bar{x}_2x_3. \quad (6.10)$$

Здесь представлена дизъюнкция тех наборов, которые не использовались в функции y_2. Запишем отрицание правой и левой частей выражения (6.10):

$$\overline{\overline{y_3}} = \overline{\bar{x}_1\bar{x}_2x_3 + x_1x_2x_3 + x_1\bar{x}_2x_3}.$$

Для левой части применим правило двойного отрицания, а для правой – правило де Моргана:

$$\overline{\overline{y_3}} = y_3.$$

$$y_3 = \overline{\bar{x}_1\bar{x}_2x_3} \cdot \overline{x_1x_2x_3} \cdot \overline{x_1\bar{x}_2x_3}.$$

Еще раз применим правило де Моргана к конъюнкциям в правой части:

$$y_3 = (\bar{\bar{x}}_1 + \bar{\bar{x}}_2 + \bar{x}_3) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_3) \cdot (\bar{x}_1 + \bar{\bar{x}}_2 + \bar{x}_3).$$

Применив правило двойного отрицания, получим

$$y_3 = (x_1 + x_2 + \bar{x}_3) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_3) \cdot (\bar{x}_1 + x_2 + \bar{x}_3). \quad (6.11)$$

Чтобы убедиться в правильности приведения формы записи функции y_3, необходимо обратить внимание на функцию y_2, для которой ранее были записаны как СДНФ (выражение (6.2)), так и СКНФ (выражение (6.4)). Запись функции y_2 соответствует записи функции y_3 (выражение (6.11)).

Приведение формы записи функции, представленной в СКНФ к форме в СДНФ, производится аналогичным образом.

6.3 Задачи для решения

Задание 1

Выполнить минимизацию функции *f*, заданной в числовой форме, на единичных наборах. Записать минимальную функцию в форме ДНФ:

а) $f(x_1x_2x_3x_4) = (0,1,4,5,7,9,13,15)$;
б) $f(x_1x_2x_3x_4) = (0,1,4,5,6,7,13,15)$;
в) $f(x_1x_2x_3x_4) = (0,2,3,4,6,8,10,11)$;
г) $f(x_1x_2x_3x_4) = (0,1,2,4,8,9,10,14)$;
д) $f(x_1x_2x_3x_4) = (0,2,5,7,8,10,13,15)$;
е) $f(x_1x_2x_3x_4) = (0,1,8,9,10,11,14,15)$.

Задание 2

Выполнить минимизацию функции *f*, заданной в числовой форме, на нулевых наборах. Записать минимальную функцию в форме КНФ:

а) $f(x_1x_2x_3x_4) = (2,3,6,7,9,11,13,15)$;
б) $f(x_1x_2x_3x_4) = (0,3,5,6,9,10,12,15)$;
в) $f(x_1x_2x_3x_4) = (1,3,5,7,9,11,13,15)$;
г) $f(x_1x_2x_3x_4) = (2,3,9,10,11,13,14,15)$;
д) $f(x_1x_2x_3x_4) = (2,3,5,7,10,11,13,15)$;
е) $f(x_1x_2x_3x_4) = (1,4,3,10,11,12,14,15)$.

6.4 Контрольные вопросы

1 Поясните понятие совершенной дизъюнктивной нормальной формы представления логических функций.

2 Поясните понятие совершенной конъюнктивной нормальной формы представления логических функций.

3 Назовите алгоритм минимизации логических выражений, представленных в СДНФ, с использованием карт Карно.

4 Назовите алгоритм минимизации логических выражений, представленных в СКНФ, с использованием карт Карно.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №7
СИНТЕЗ ЦИФРОВЫХ АВТОМАТОВ

Цель занятия: получить практические навыки в решении задач синтеза логической схемы цифрового автомата, заданного таблицей переходов и таблицей выходов.

7.1 Краткие теоретические сведения

Цифровые автоматы определяются следующими характеристиками:

$$\{A, W, Z, \delta, \lambda, a_n\},$$

где $A = \{a_1, a_2, \dots, a_n\}$ – множество состояний цифрового автомата;

$W = \{w_1, w_2, \dots, w_n\}$ – множество выходных сигналов цифрового автомата;

$Z = \{z_1, z_2, \dots, z_n\}$ – множество входных сигналов цифрового автомата;

λ – функция выработки выходного сигнала $w(t+1)$ в зависимости от текущего состояния $a(t)$ цифрового автомата и действующего на его входе сигнала $z(t)$:

$$w(t+1) = \lambda(a(t), z(t));$$

δ – функция перехода цифрового автомата в новое состояние $a(t+1)$ в зависимости от его текущего состояния $a(t)$ и действующего на его входе сигнала $z(t)$:

$$a(t+1) = \delta(a(t), z(t));$$

$a_n \in A$ – начальное состояние цифрового автомата.

Два автомата называются *эквивалентными*, если при одинаковом множестве входных и выходных сигналов и при любом начальном состоянии для любого входного набора сигналов они формируют одинаковые выходные наборы сигналов. Цифровой автомат называется *конечным*, если используемые для его задания множества конечны. Цифровой автомат является *полностью определенным*, если каждой паре $a(t), z(t)$ ставится в соответствие пара $a(t+1), w(t+1)$. В противном случае цифровой автомат называется *частично определенным*, или просто частичным.

Выделяют два основных вида цифровых автоматов:

- автомат Мура – автомат общего типа – его входной сигнал и новое состояние являются функцией текущего состояния и действующего на входе сигнала;

- автомат Мили – его выходной сигнал прямо не зависит от входного и определяется состоянием автомата. Его работа задается в виде следующих уравнений:

$$w(t+1) = \lambda(a(t));$$
$$a(t+1) = \delta\big(a(t), z(t)\big).$$

Цифровые автоматы задаются с помощью таблиц или графов.

Автомат Мили задается с помощью таблиц переходов или выходов, либо с помощью одной объединенной таблицы (рисунок 7.1).

Таблица переходов

	a_1	a_2	a_3	a_4
z_1	a_2	a_4	a_2	a_2
z_2	a_4	a_4	a_2	a_2
z_3	a_2	a_1	a_2	a_3

Таблица выходов

	a_1	a_2	a_3	a_4
z_1	w_3	w_1	w_1	w_3
z_2	w_2	w_1	w_2	w_3
z_3	w_2	w_1	w_2	w_1

Объединенная таблица

	a_1	a_2	a_3	a_4
z_1	a_2/w_3	a_4/w_1	a_2/w_1	a_2/w_3
z_2	a_4/w_2	a_4/w_1	a_2/w_2	a_2/w_3
z_3	a_2/w_2	a_1/w_1	a_2/w_2	a_3/w_1

Рисунок 7.1 – Примеры автомата Мили

Описать формирование выходного сигнала можно следующим образом: когда автомат в текущий момент времени t находится в состоянии a_j и получает на вход сигнал z_i, то он переходит в новое состояние $a(t+1)$, которое записано в клетке таблицы переходов, расположенной на пересечении j-й колонки и i-й строки, после чего вырабатывается выходной сигнал $w(t+1)$, который определяется значением в клетке, расположенной в таблице выходов на пересечении колонки j и строки i.

Автомат Мура задается с использованием одной таблицы, которая определяет правило формирования нового состояния и выходной сигнал (см. рисунок 7.2).

	w_2	w_4	w_1	w_3
	b_1	b_2	b_3	b_4
z_1	b_2	b_4	b_2	b_2
z_2	b_4	b_4	b_2	b_2
z_3	b_2	b_1	b_2	b_3

Рисунок 7.2 – Пример автомата Мура

Выходной сигнал формируется следующим образом: когда автомат находится в состоянии b_j и получает на вход сигнал z_i, то он переходит в новое состояние, которое записано в клетке таблицы переходов, расположенной на пересечении j-й колонки и i-й строки, и вырабатывает выходной сигнал, который определяется новым состоянием цифрового автомата.

7.2 Методические указания и примеры решения задач

Основными составляющими цифрового автомата с точки зрения синтеза являются память и логическая часть (рисунок 7.3). Память хранит информацию о предыстории цифрового автомата. Логическая часть на основании входного сигнала и поступающего из памяти сигнала вырабатывает выходной сигнал. Используя входной сигнал и сигнал состояния, логическая часть вырабатывает сигнал управления памятью, обеспечивающий переход из текущего в новое состояние.

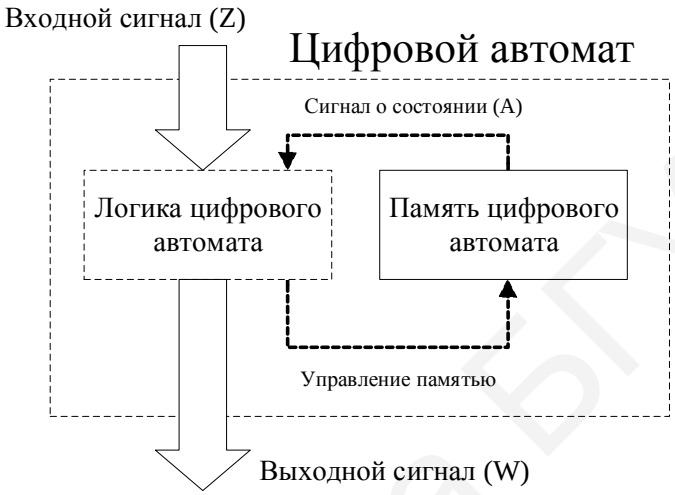


Рисунок 7.3 – Структурная схема автомата

Память реализуется на триггерах, а сигналы управления памятью должны соответствовать заданным переходам и типу используемых триггеров.

Алгоритм выполнения синтеза цифрового автомата примет следующий вид:

- кодирование входных сигналов в виде набора логических переменных;
- кодирование выходных сигналов в виде набора логических функций;
- кодирование состояний цифрового автомата;
- формирование кодированной таблицы переходов и выходов;
- выбор типа запоминающего элемента;
- составление логических выражений для логических функций, использованных для кодировки выходных сигналов;
- составление логических выражений для сигналов управления памятью;
- синтез логических схем для сформированных логических выражений;
- формирование выходных сигналов цифрового автомата на основании кодирующих их функций.

Пример

Синтезировать цифровой автомат, заданный в виде таблиц, приведенных на рисунке 7.4.

Шаг 1. Кодирование входных сигналов выполним через набор логических переменных x. Множество входных сигналов включает три элемента, поэтому для их кодирования достаточно использовать комбинации двух переменных.

Шаг 2. Кодирование выходных сигналов выполним через набор логических переменных w. Множество выходных сигналов включает три элемента, поэтому для кодирования каждой из них достаточно использовать комбинации из двух переменных.

Таблица переходов

	a_1	a_2	a_3	a_4
z_1	a_2	a_4	a_2	a_2
z_2	a_4	a_4	a_2	a_2
z_3	a_2	a_1	a_2	a_3

Таблица выходов

	a_1	a_2	a_3	a_4
z_1	w_3	w_1	w_1	w_3
z_2	w_2	w_1	w_2	w_3
z_3	w_2	w_1	w_2	w_1

Рисунок 7.4 – Автомат Мили

Шаг 3. Кодирование состояний выполним через набор логических переменных Q. Множество состояний включает четыре элемента, поэтому для кодирования каждого из них достаточно использовать комбинации двух переменных. Полученные результаты представлены на рисунке 7.5.

Кодирование выходных сигналов

	y_1	y_2
w_1	0	1
w_2	1	0
w_3	1	1

Кодирование входных сигналов

	x_1	x_2
z_1	0	1
z_2	1	0
z_3	1	1

Кодирование состояний

	Q_1	Q_2
a_1	0	1
a_2	1	0
a_3	1	1
a_4	1	1

Рисунок 7.5 – Результаты кодирования сигналов в процессе синтеза автомата

Шаг 4. Таблица кодированных переходов и выходов примет вид, представленный на рисунке 7.6.

Таблица переходов

	$\overline{Q}_1Q_2$	$Q_1\overline{Q}_2$	Q_1Q_2	$\overline{Q}_1\overline{Q}_2$
$\overline{x}_1x_2$	10	00	10	10
$x_1\overline{x}_2$	00	00	10	10
x_1x_2	10	01	10	11

Таблица выходов

	$\overline{Q}_1Q_2$	$Q_1\overline{Q}_2$	Q_1Q_2	$\overline{Q}_1\overline{Q}_2$
$\overline{x}_1x_2$	11	01	01	11
$x_1\overline{x}_2$	10	01	10	11
x_1x_2	10	01	10	01

Рисунок 7.6 – Кодированные характеристики автомата Мили

Двухразрядный код в клетках таблицы перехода формируется следующим образом: первый разряд отображает значение одной переменной, а второй – значение второй переменной. При этом если значение разряда равно единице, то переменная имеет прямое значение, если нулю, то обратное. В первой таблице заданы переходы и в качестве переменных выступают Q_1 и Q_2, во второй таблице в качестве переменных используются y_1 и y_2, которые кодируют выходные сигналы.

Шаг 5. Реализацию памяти выполним на двух *T*-триггерах, формирующих парафазные выходные сигналы $Q_1, \bar{Q}_1$, используемые для кодировки состояний.

Шаг 6. Составим логические выражения для логических функций, использованных для кодировки выходных сигналов:

$$y_1 = \bar{Q}_1 Q_2 \bar{x}_1 x_2 + \bar{Q}_1 Q_2 x_1 \bar{x}_2 + \bar{Q}_1 Q_2 x_1 x_2 + \\ + Q_1 Q_2 x_1 \bar{x}_2 + Q_1 Q_2 x_1 x_2 + \bar{Q}_1 \bar{Q}_2 \bar{x}_1 x_2 + \bar{Q}_1 \bar{Q}_2 x_1 x_2; \quad (7.1)$$

$$y_2 = \bar{Q}_1 \bar{Q}_2 \bar{x}_1 x_2 + \bar{Q}_1 \bar{Q}_2 x_1 \bar{x}_2 + \bar{Q}_1 \bar{Q}_2 x_1 x_2 + \\ + \bar{Q}_1 Q_2 \bar{x}_1 x_2 + Q_1 \bar{Q}_2 \bar{x}_1 x_2 + Q_1 \bar{Q}_2 x_1 \bar{x}_2 + \\ + Q_1 \bar{Q}_2 x_1 x_2 + Q_1 Q_2 \bar{x}_1 x_2. \quad (7.2)$$

Приведенные выражения составляются следующим образом. При формировании сигнала y_1в кодированной таблице выходных сигналов (рисунок 7.6) выбираются все случаи, когда y_1 имеет единичное значение, и для каждого из них формируется конъюнкция, отражающая начальное состояние и входные сигналы. Например, первая конъюнкция $\bar{Q}_1 Q_2 \bar{x}_1 x_2$ соответствует случаю, когда текущее состояние автомата имеет значение $\bar{Q}_1 Q_2$ и на вход поступает сигнал $\bar{x}_1 x_2$. В клетке, соответствующей данным значениям, записан код 11, первый разряд данного кода соответствует переменной y_1 (рисунок 7.7).

Таблица выходов

Разряд соответствующий переменной y_1

Код состояния соответствующий переменной $y_1 = 1$

Код входного сигнала соответствующий переменной $y_1 = 1$

	$\overline{Q_1}Q_2$	$Q_1\overline{Q_2}$	Q_1Q_2	$\overline{Q_1}\overline{Q_2}$
$\overline{x_1}x_2$	11	01	01	11
$x_1\overline{x_2}$	10	01	10	11
x_1x_2	10	01	10	01

Рисунок 7.7 – Формирование логических выражений для выходных сигналов на примере первой конъюнкции сигнала y_1

Выражение для y_2 формируется аналогично, но рассматриваются значения второго разряда двухразрядного кода, соответствующего выходному сигналу цифрового автомата.

Шаг 7. Составим логические выражения для логических функций, использованных для кодировки выходных сигналов:

$$q_{T1} = \bar{Q}_1 Q_2 \bar{x}_1 x_2 + \bar{Q}_1 Q_2 x_1 x_2 + Q_1 \bar{Q}_2 \bar{x}_1 x_2 + Q_1 \bar{Q}_2 x_1 \bar{x}_2 + \\ + Q_1 \bar{Q}_2 x_1 x_2 + \bar{Q}_1 \bar{Q}_2 \bar{x}_1 x_2 + \bar{Q}_1 \bar{Q}_2 x_1 \bar{x}_2 + \bar{Q}_1 \bar{Q}_2 x_1 x_2; \quad (7.3)$$

$$q_{T2} = \bar{Q}_1 \bar{Q}_2 x_1 x_2 + \bar{Q}_1 Q_2 \bar{x}_1 x_2 + \bar{Q}_1 Q_2 x_1 \bar{x}_2 + \bar{Q}_1 Q_2 x_1 x_2 + \\ + Q_1 \bar{Q}_2 x_1 x_2 + Q_1 Q_2 x_1 \bar{x}_2 + Q_1 Q_2 x_1 x_2. \quad (7.4)$$

Приведенные выражения формируются следующим образом. Так как в рассматриваемом случае элементом памяти выбран *T*-триггер (шаг 5), который

по каждому входному сигналу меняет свое состояние на противоположное, то сигнал на вход *T*-триггера надо подавать только в тех случаях, когда его состояние отличается от текущего. Таким образом, в выражении (7.3) первого разряда сигнала *T*-входа используются конъюнкции, соответствующие случаям, когда значение первого разряда кода нового состояния противоположно его начальному состоянию. Например, первая конъюнкция в выражении для q_{T1} имеет вид $\bar{Q}_1 Q_2 \bar{x}_1 x_2$ и соответствует случаю, когда начальное состояние имеет значение $\bar{Q}_1 Q_2$, или 01, на вход поступает сигнал $\bar{x}_1 x_2$, а код нового состояния $Q_1 \bar{Q}_2$, или 10, имеет в первом разряде значение, отличное от значения первого разряда кода начального состояния (рисунок 7.8).

Таблица переходов

Разряд нового состояния соответствующего коду q_{T1} $Q_1(t+1) = 1$, значение изменилось на противоположное

Разряд начального состояния автомата соответствующего коду q_{T1} $Q_1(t)=\bar{Q}_1$ (0)

	$\bar{Q}_1 Q_2$	$Q_1 \bar{Q}_2$	$Q_1 Q_2$	$\bar{Q}_1 \bar{Q}_2$
$\bar{x}_1 x_2$	10	00	10	10
$x_1 \bar{x}_2$	00	00	10	10
$x_1 x_2$	10	01	10	11

Код входного сигнала соответствующий переменной q_{T1}

Разряд нового состояния, соответствующего коду q_{T1} для входного сигнала $x_1 \bar{x}_2$ $Q_1(t+1) = 0$, значение не изменилось, в выражение **НЕ** записываем

Рисунок 7.8 – Формирование логических выражений для сигналов перехода на примере первой конъюнкции сигнала q_{T1}

Выражение для q_{T2} формируется аналогично, но рассматриваются изменения значения второго разряда двухразрядного кода состояния после учета воздействия входного сигнала.

Шаг 8. Прежде чем перейти к задаче синтеза полученных логических выражений, необходимо минимизировать записи этих выражений.

Выполним минимизацию функций с помощью метода с использованием карт Карно (рисунок 7.9).

Таким образом, выражения (7.1)–(7.4) примут следующий вид:

$$y_1 = \bar{Q}_1 x_2 + Q_2 x_1; \quad (7.5)$$

$$y_2 = \bar{x}_1 x_2 + \bar{Q}_2 x_1; \quad (7.6)$$

$$q_{T1} = \bar{Q}_1 x_2 + \bar{Q}_2 x_1 + \bar{Q}_2 x_2; \quad (7.7)$$

$$q_{T2} = Q_2 x_2 + Q_2 x_1 + x_1 x_2. \quad (7.8)$$

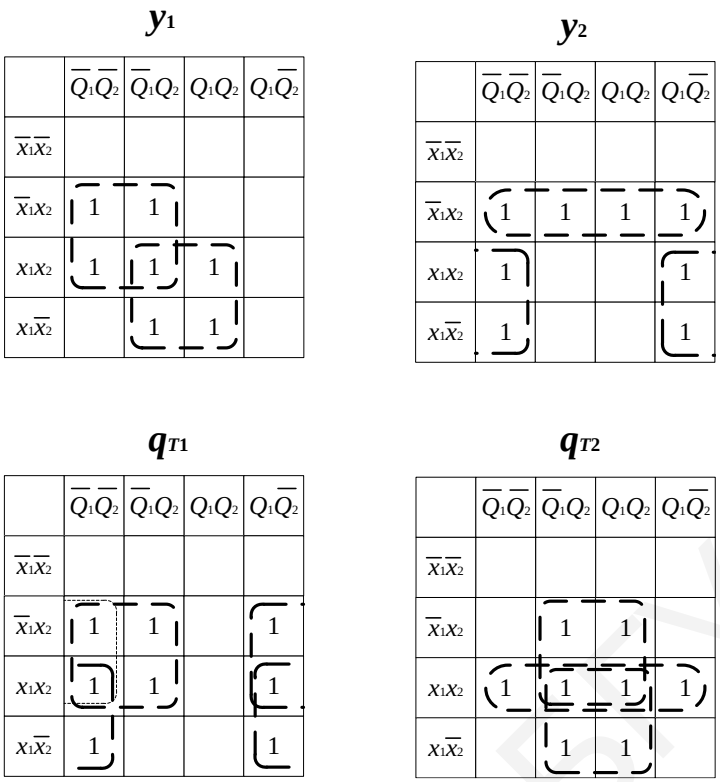


Рисунок 7.9 – Минимизация выходных сигналов и сигналов перехода с использованием карт Карно

Логическая схема, реализующая сформированные логические выражения для выходных сигналов и сигналов управления памятью цифрового автомата, имеет вид, представленный на рисунке 7.10.

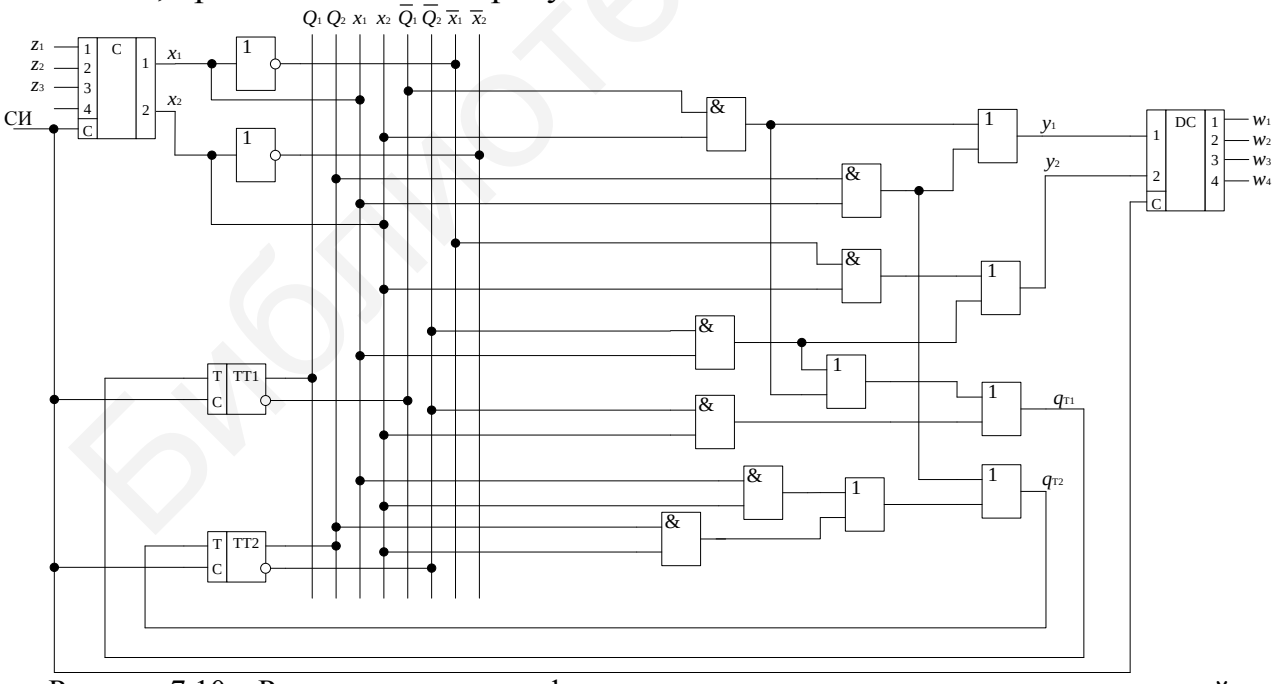


Рисунок 7.10 – Результат синтеза цифрового автомата минимизированных выражений

Шаг 9. Выходные сигналы формируются на основе кодированных с использованием декодера *DC*.

7.3 Задачи для решения

Задание 1

Синтезировать цифровой автомат, заданный в виде таблиц, приведенных на рисунке 7.11. Память реализовать на *T*-триггерах.

	a_1	a_2	a_3	a_4
z_1	a_3/w_1	a_1/w_4	a_2/w_1	a_3/w_4
z_2	a_4/w_1	a_4/w_3	a_1/w_3	a_2/w_4
z_3	a_3/w_2	a_2/w_3	a_4/w_4	a_3/w_4

а

	a_1	a_2	a_3	a_4
z_1	a_2/w_3	-	a_1/w_3	a_2/w_4
z_2	a_3/w_2	a_1/w_1	a_3/w_4	a_2/w_2
z_3	-	a_1/w_4	a_3/w_2	a_1/w_2

г

	a_1	a_2	a_3	a_4
z_1	a_1/w_2	a_3/w_1	a_2/w_2	a_4/w_3
z_2	a_3/w_3	a_4/w_2	a_1/w_4	a_2/w_1
z_3	a_3/w_2	a_4/w_1	a_2/w_2	a_3/w_1

б

	a_1	a_2	a_3	a_4
z_1	a_3/w_2	a_4/w_1	-	a_1/w_3
z_2	a_4/w_3	a_1/w_2	a_2/w_3	a_1/w_1
z_3	-	a_3/w_1	a_2/w_2	a_4/w_3

д

	a_1	a_2	a_3	a_4
z_1	a_3/w_4	a_1/w_3	a_2/w_4	a_3/w_3
z_2	a_1/w_2	a_4/w_3	a_3/w_2	a_2/w_3
z_3	a_1/w_4	a_1/w_1	a_2/w_1	a_4/w_1

в

	a_1	a_2	a_3	a_4
z_1	a_1/w_2	a_4/w_1	a_4/w_4	a_2/w_3
z_2	a_4/w_3	a_1/w_2	a_2/w_3	a_4/w_1
z_3	a_1/w_4	a_1/w_2	a_2/w_2	a_4/w_3

е

а – вариант 1; *б* – вариант 2; *в* – вариант 3; *г* – вариант 4; *д* – вариант 5; *е* – вариант 6

Рисунок 7.11 – Шесть вариантов заданий для синтеза автомата

7.4 Контрольные вопросы

1 Что такое цифровой автомат?
2 Какие характеристики определяют цифровой автомат?
3 Какие виды цифровых автоматов бывают?
4 Назовите основные узлы цифрового автомата с точки зрения синтеза.
5 Опишите алгоритм синтеза цифрового автомата.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №8
СОСТАВЛЕНИЕ МИКРОПРОГРАММ

Цель: приобрести практические навыки решения задач построения микропрограммы для заданной граф-схемы алгоритма.

8.1 Краткие теоретические сведения

Для выполнения логических и арифметических операций в компьютерной технике служит арифметико-логическое устройство (АЛУ). Обобщенная структурная схема АЛУ представлена на рисунке 8.1.

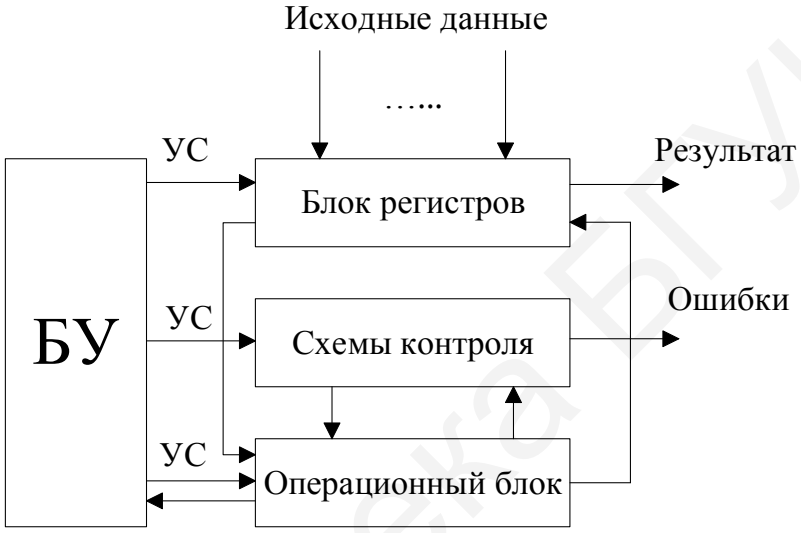


Рисунок 8.1 – Обобщенная структурная схема АЛУ

АЛУ можно разделить на два основных блока:
- блок управления (БУ);
- операционный блок (ОБ).

В свою очередь ОБ состоит из следующих типовых узлов:
- сумматор;
- операционные узлы, служащие для выполнения логических операций;
- мультиплексор;
- счетчик для подсчета тактов выполнения длинных операций;
- регистр флагов для фиксации особой информации, характеризующей результат.

Каждый из перечисленных типовых узлов может выполнять различные микрооперации, инициируемые по управляющим сигналам (УС). Выполнение любой арифметической операции представляет собой выполнение определенной последовательности микроопераций в узлах операционной части. Такие последовательности образуют алгоритм выполнения операций на уровне микроопераций, представлять который удобно, используя граф-схему алгоритма (ГСА).

8.2 Методические указания и примеры решения задач

Блок управления АЛУ может строиться на принципе микропрограммирования (программируемая логика) или на принципе с жесткой логикой (аппаратный принцип). В любом случае удобной формой задания поведения управляемого объекта является кодированная ГСА. При использовании микропрограммного принципа выработка необходимой последовательности сигналов управления имеющимся объектом выполняется за счет реализации микропрограммы, разработанной в соответствии с заданной ГСА. При использовании аппаратного принципа блок управления строится в виде цифрового автомата, который, построенный в соответствии с заданной ГСА, вырабатывает последовательность выходных сигналов, используемых для управления объектом.

При микропрограммном принципе построения блока управления алгоритм выполнения операции реализуется за счет выполнения особой программы, состоящей из отдельных команд, реализующих требуемую последовательность выполнения микроопераций. Такие команды называются микрокомандами, а совокупность микрокоманд – микропрограммой. Микропрограмма хранится в памяти. При представлении алгоритма операции в виде ГСА выполняемая микрокоманда должна обеспечить выработку сигналов соответствующих микроопераций и обеспечить переход к следующей микрокоманде, в том числе и при ветвлении вычислительного процесса в зависимости от проверяемых условий, характеризующих состояние управляемого объекта. Таким образом, в формате микрокоманды, в принципе, необходимо иметь несколько полей:

- микроопераций (*У*), используемое для задания одной или нескольких микроопераций;
- условий (*Х*), в котором задаются проверяемые условия, влияющие на ветвление вычислительного процесса;
- адреса (*А*), в котором необходимо задавать информацию, определяющую следующую микрокоманду при возможном ветвлении (по крайней мере, по двум направлениям) для продолжения вычислительного процесса.

Задание всей перечисленной информации в едином формате микрокоманды затруднительно. Как правило, используют два вида, а следовательно, и два формата микрокоманд:

- операционные;
- перехода.

Операционная микрокоманда задает выполняемую микрооперацию и включает два поля:

- типа микрокоманды (*Т*);
- микрооперации (*У*).

Так как типов два, то поле типа микрокоманды имеет размерность 1 бит.

Поле микрооперации задает в кодированной форме подлежащую выполнению микрооперацию. Если допустимая длина микрокоманды достаточно велика, то в одной операционной микрокоманде может задаваться более одной микрооперации (рисунок 8.2). В качестве следующей микрокоманды в этом

случае выбирается микрокоманда, расположенная в следующем адресе ЗУ после адреса расположения текущей выполняемой микрокоманды.

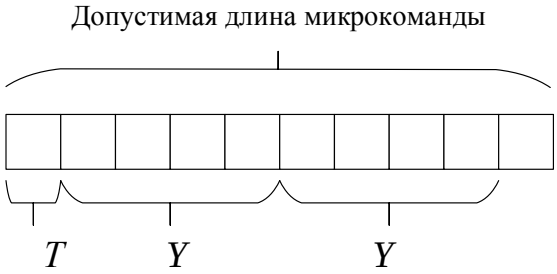


Рисунок 8.2 – Формат операционной микрокоманды

Микрокоманда перехода используется для организации ветвления на основании результата проверки некоторого условия (признака). Поэтому в ней необходимо задавать код проверяемого условия и информацию об адресах двух возможных ветвей продолжения процесса выполнения алгоритма. Приведенный формат включает следующие поля:

- типа микрокоманды (*T*);
- условия (*X*);
- адреса (*A*);
- модификатора дисциплины перехода (*M*).

По аналогии с форматом операционной микрокоманды поле типа имеет разрядность 1 бит.

Поле условия используется, чтобы задать условие, которое необходимо проверить при реализации данной микрокоманды. Его длина определяется общим количеством условий.

Поле адреса используется, чтобы задать местоположение в памяти адресов первых микрокоманд двух возможных ветвей продолжения процесса. В качестве начальной микрокоманды одной ветви продолжения используется микрокоманда, расположенная по адресу, следующему в памяти (A_C) за адресом текущей выполняемой микрокомандой (A_T), а адрес начальной микрокоманды другой ветви задается в самой микрокоманде (*A*).

На рисунке 8.3 приведена схема устройства управления, построенного на микропрограммном принципе, которое включает в себя:

- запоминающее устройство для хранения всех микропрограмм для управления объектом. Сигналы микроопераций представлены как $Y = \{y_1, y_2 \dots y_m\}$, признаки перехода (условия) – $X = \{x_1, x_2 \dots x_n\}$;
- регистр адреса для задания адреса микрокоманды. Начальный адрес (A_H) формируется посредствам ФНА (формирователя начального адреса);
- дешифраторы кода микрооперации и кода условия;
- синхроимпульсы (СИ), по которым выполняются условия.

Приведенная схема устройства микропрограммного управления соответствует случаю, когда операционная микрокоманда и команда перехода имеют одинаковую длину, в каждом адресе запоминающего устройства полностью помещается одна микрокоманда, в операционной микрокоманде задается для

выполнения только одна микрооперация. В микрокоманде перехода не используется поле *M* (поле модификатора дисциплины перехода).

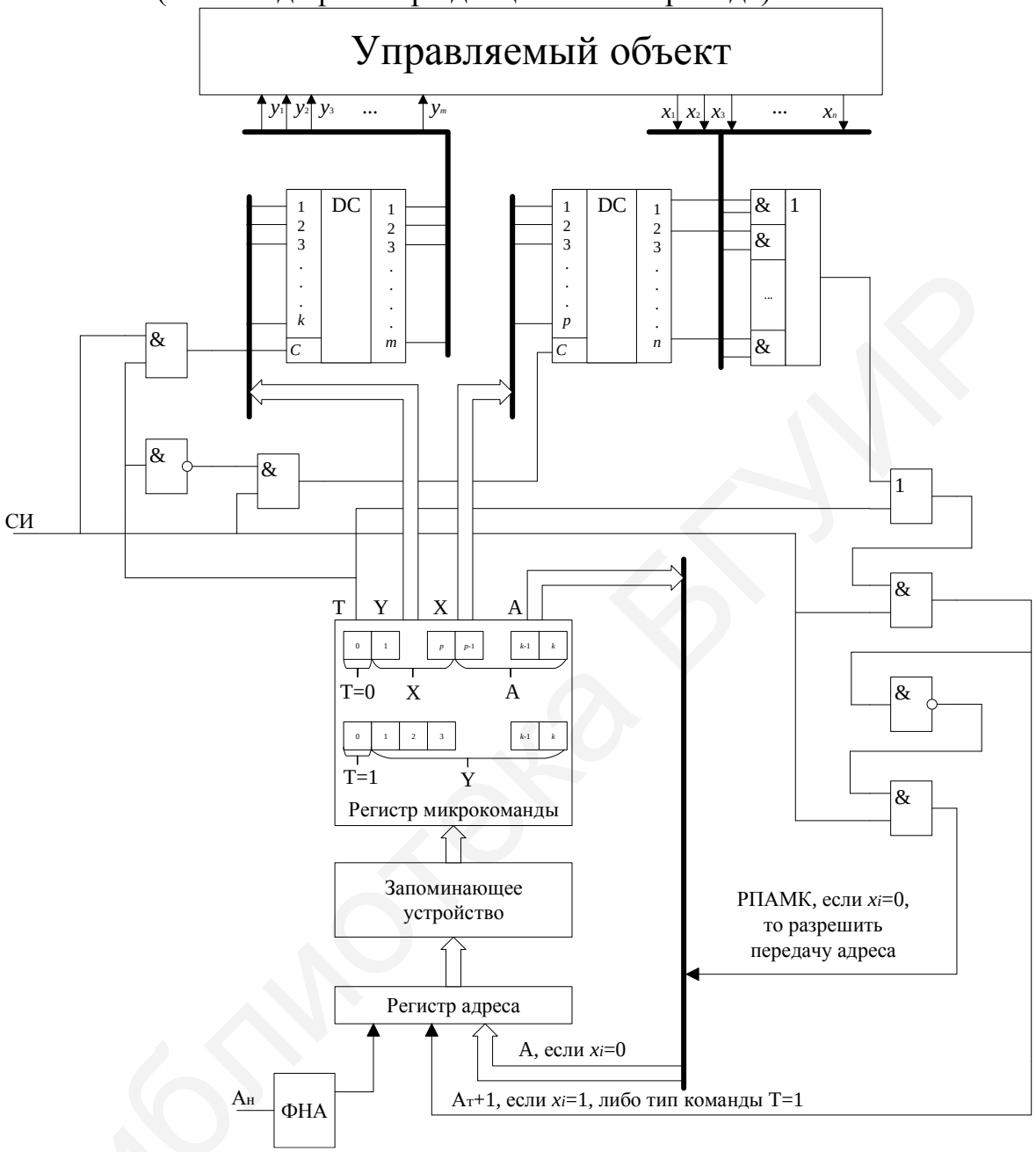


Рисунок 8.3 – Структурная схема устройства управления

Пример

Составить микропрограмму для реализации ГСА (рисунок 8.4). Управляемый объект характеризуется следующими параметрами:

- множество проверяемых условий – 15;
- множество выполняемых операций – 120 (предусмотреть поле y_k, чтобы задать последнюю микрокоманду микропрограммы);
- емкость памяти для записи программы – 2 Кбайт;
- длина ячейки памяти – 16 бит;
- начальный адрес размещения микропрограммы равен 530.

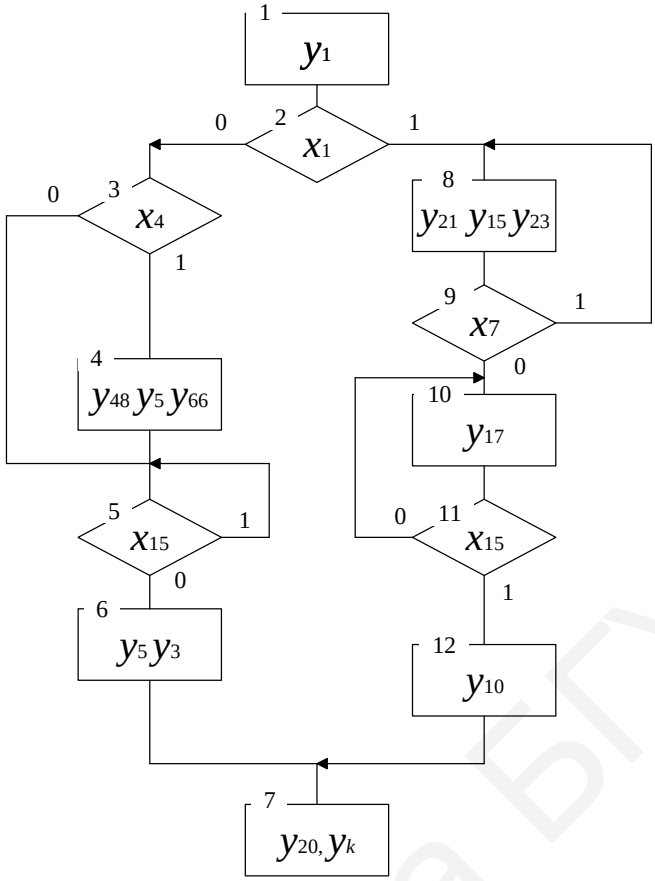


Рисунок 8.4 – ГСА для составления микропрограммы

Исходя из заданных параметров, длина ячейки L = 16 бит, тогда формат операционной микрокоманды (МКО) примет следующий формат:

- нулевой бит необходим, чтобы задать тип микрокоманды (T = 1);
- $Y_1 = \{y_1, y_2 \dots y_{120}\}$, значит, для кода микрооперации потребуется использование 7 бит ($2^7 > 120$).
- оставшиеся 8 из 16 бит используются для поля кода микрооперации Y_2 (7 бит) и поля y_k (1 бит, y_k = 1, если микрокоманда является последней для данной микропрограммы, в противном случае y_k = 0).

Формат микрокоманды переходов примет следующий вид:

- нулевой бит необходим , чтобы задать тип микрокоманды (T = 0);
- $X = \{x_1, x_2 \dots x_{15}\}$, тогда для кодирования условий должно быть выделено 4 бита ($2^4 > 15$);
- 10 бит необходимы, чтобы задать код адреса (A), т. к. длина одной ячейки памяти L = 16 бит = 2 байта и таких ячеек можно задать 2 Кбайт/2 байта, равные 1024 шт. (2^{10} = 1024);
- последний разряд необходим, чтобы задать модификатор (M) перехода. Если M = 0, то для x_i = 0 адрес следующей выполняемой команды (A_C) равен A_T + 1, а для x_i = 1 равен A (т. е. адресу, указанному в поле адреса). Для значения M = 1 условия обратные: для x_i = 1 адрес следующей выполняемой команды (A_C) равен A_T + 1, а для x_i = 0 равен A.

Форматы микропрограммы представлены на рисунке 8.5.

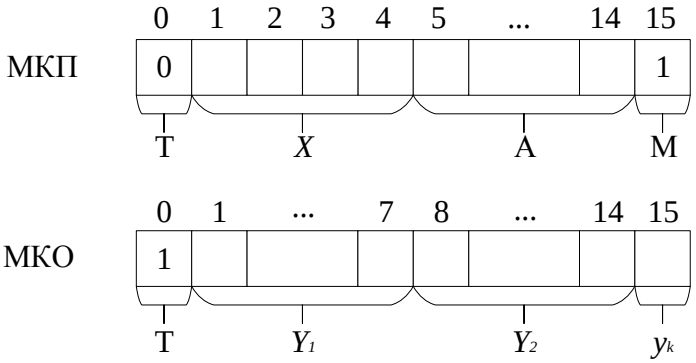


Рисунок 8.5 – Форматы микропрограмм для рассматриваемой задачи

Формирование микропрограммы следует начинать с первой вершины ГСА, соответственно код микрооперации в ЗУ соответствует начальному адресу A_H, заданному в двоичном коде. Данная микрокоманда будет иметь формат, соответствующий микрооперационной команде ($T = 1$), т. к. в вершине задан y_1. В поле первой микрооперации записывается семибитовый двоичный эквивалент индекса реализуемой микрооперации (y_1 соответствует 0000001). Поле второй микрооперации остается незаданным (заполняется нулями). Данная микрокоманда не является последней в микропрограмме, значит, $y_k = 0$. Вторая микрокоманда реализует вторую вершину ГСА и выполняется вслед за первой микрокомандой. Поэтому адресу второй микрокоманды соответствует увеличенный на единицу адрес первой микрокоманды ($531_{10} = 1000010011_2$). Полученные результаты по первой микрокоманде представлены в таблице 8.1.

Таблица 8.1 – Микропрограмма заданной ГСА на первом этапе

Номер микрокоманды	Номер вершины ГСА	Адрес микрокоманды	Код микрокоманды	Примечание
1	1	1000010010	1.0000001.0000000.0	–
2	2	1000010011	–	–

Вторая микрокоманда реализует вершину с условием, тогда поле T имеет значение $T = 0$ (микрокоманда перехода). В поле X записывается двоичный код выполняемого условия (x_1 соответствует 0001). В случае выполнения данного условия переходим к микрокоманде, расположенной по адресу, следующему за адресом данной микрокоманды (A_T = 1000010011, A_C = 1000010100), в противном случае переходим по адресу данной микрокоманды, записанному в поле A. Описанное поведение перехода соответствует модификатору $M = 1$. Следующей рассматриваемой вершиной будет вершина 8. Следует отметить, что заполнение поля A на данном этапе является невозможным, пока не будет рассчитан адрес последней команды в данной ветви. Это поле остается незаполненным. Таким образом, результаты на данном этапе примут вид, представленный в таблице 8.2.

Таблица 8.2 – Микропрограмма заданной ГСА на втором этапе

Номер микрокоманды	Номер вершины ГСА	Адрес микрокоманды	Код микрокоманды	Примечание
1	1	1000010010	1.0000001.0000000.0	–
2	2	1000010011	0.0001._____.1	Переход к вершине 3
3	8	1000010100	–	–

Отличительной особенностью вершины 8 по отношению к вершине 1 является то, что в вершине 8 записаны три операционные команды. Так как код микрооперационной команды содержит лишь два поля для операций, то необходимо представить вершины 8.1 и 8.2 (рисунок 8.6).

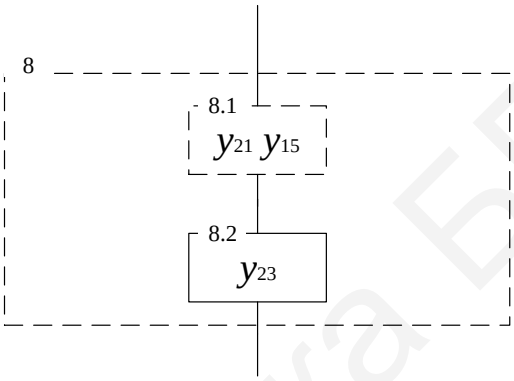


Рисунок 8.6 – Представление вершины 8

Далее по аналогии поступим с вершиной 1. Следующей выполняемой вершиной станет вершина 9. Микропрограмма на данном этапе имеет вид, представленный в таблице 8.3.

Таблица 8.3 – Микропрограмма заданной ГСА на третьем этапе

Номер микрокоманды	Номер вершины ГСА	Адрес микрокоманды	Код микрокоманды	Примечание
1	1	1000010010	1.0000001.0000000.0	–
2	2	1000010011	0.0001._____.1	Переход к вершине 3
3	8.1	1000010100	1.0010101.0001111.0	–
4	8.2	1000010101	1.0010111.0000000.0	–
5	9	1000010110	–	–

Реализацию вершины 9 рассмотрим двумя способами:

- когда поле модификатора *M* в МКП отсутствует;
- поле модификатора *M* в МКП присутствует.

Демонстрация первого случая соответствует поведению микропрограммы в заданном примере при $M = 1$. Тогда адресом выполняемой микрокоманды 5 должен быть адрес микрокоманды, реализующий вершину 8, но такая микрокоманда уже была и располагается по предыдущему адресу. Таким образом,

необходимо добавить виртуальную вершину 9.1 с нулевым условием (поле $X = 0$, такое условие никогда не выполнится), располагающуюся по следующему адресу за адресом микрокоманды 5 (рисунок 8.7). В поле A микрокоманды, реализующей вершину 9.1, необходимо записать адрес микрокоманды, реализующий вершину 8 (8.1). При таком условии микропрограмма, выполняя микрокоманду реализации вершины 9.1, всегда будет проходить по ложной ветви и следующей выполнять микрокоманду, реализующую вершину 8 (безусловная адресация). Адресацию микрокоманд можно продолжить со следующего адреса после адреса микрокоманды, реализующей вершину 9.1 ($A_C = A_{9.1} + 1$).

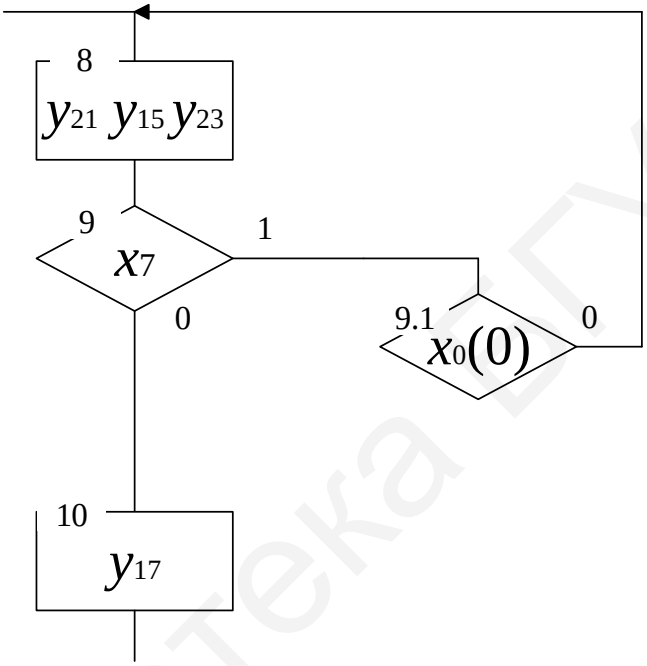


Рисунок 8.7 – Представление вершины 9 в случае отсутствия поля модификатора M (или $M = 1$)

Реализация представленных вершин с помощью микрокода представлена в таблице 8.4.

Таблица 8.4 – Микропрограмма заданной ГСА на четвертом этапе

Номер микрокоманды	Номер вершины ГСА	Адрес микрокоманды	Код микрокоманды	Примечание
1	1	1000010010	1.0000001.0000000.0	–
2	2	1000010011	0.0001._____.1	К вершине 3
3	8.1	1000010100	1.0010101.0001111.0	–
4	8.2	1000010101	1.0010111.0000000.0	–
5	9	1000010110	0.0111. 1000011000.1	К вершине 10
6	9.1	1000010111	0.0000. 1000010100.1	К вершине 8
7	10	1000011000	–	–

При такой реализации расходуется дополнительный адрес для микрокоманды, реализующей вершину 9.1.

Для второго способа такой проблемы нет, однако требуется наличие дополнительного поля модификатора *M*. В данном случае разрядность микрокоманды позволяет использовать такое поле. Модификатор *M* определяет, для какой ветки адрес следующей команды задается, а для какой используется следующий после текущего. Задав в поле модификатора значение нуль (*M* = 0), микропрограмма будет использовать увеличенный на единицу адрес для ветви при условии $x_i = 0$. Тогда полученная микропрограмма примет вид, представленный в таблице 8.5.

Таблица 8.5 – Микропрограмма заданной ГСА на пятом этапе

Номер микрокоманды	Номер вершины ГСА	Адрес микрокоманды	Код микрокоманды	Примечание
1	1	1000010010	1.0000001.0000000.0	–
2	2	1000010011	0.0001._____.1	К вершине 3
3	8.1	1000010100	1.0010101.0001111.0	–
4	8.2	1000010101	1.0010111.0000000.0	–
5	9	1000010110	0.0111. 1000010100.0	К вершине 8
6	10	1000010111	–	–

Так как в ходе решения определено использование модификатора *M*, следует продолжать заполнение таблицы 8.5.

Реализация вершин 10, 11, 12, 7 происходит аналогично предыдущим микрокомандам. Стоит отметить, что для микрокоманды, реализующей вершину 11, есть возможность сразу указать поле адреса (*A*), которое соответствует уже описанной микрокоманде. Микрокоманда, реализующая вершину 7, является последней в данной микропрограмме, поэтому в поле y_k данной микрокоманды следует указать единичное значение (таблица 8.6). После выполнения последней микрокоманды все последующие адреса запоминающего устройства (ЗУ) остаются незадействованными, поэтому в этой области можно расположить микрокоманды второй ветви, исходящей из вершины 2.

Таблица 8.6 – Микропрограмма заданной ГСА на шестом этапе

Номер микрокоманды	Номер вершины ГСА	Адрес микрокоманды	Код микрокоманды	Примечание
1	1	1000010010	1.0000001.0000000.0	–
2	2	1000010011	0.0001._____.1	К вершине 3
3	8.1	1000010100	1.0010101.0001111.0	–
4	8.2	1000010101	1.0010111.0000000.0	–
5	9	1000010110	0.0111. 1000010100.0	К вершине 8
6	10	1000010111	1.0010001.0000000.0	–
7	11	1000011000	0.1111. 1000010111.1	К вершине 10
8	12	1000011001	1.0001010.0000000.0	–
9	7	1000011010	1.0010111.0000000.1	–
10	3	1000011011	–	–

На рисунке 8.8 представлена виртуальная вершина безусловного перехода в заданной ГСА.

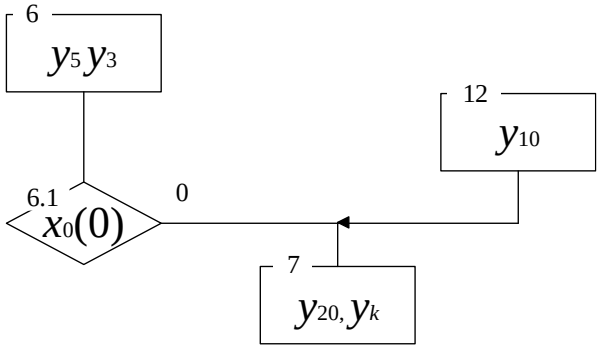


Рисунок 8.8 – Виртуальная вершина безусловного перехода в заданной ГСА

Конечная микропрограмма, реализующая заданную ГСА, представлена в таблице 8.7.

Таблица 8.7 – Микропрограмма, реализующая заданную ГСА

Номер микрокоманды	Номер вершины ГСА	Адрес микрокоманды	Код микрокоманды	Примечание
1	1	1000010010	1.0000001.0000000.0	–
2	2	1000010011	0.0001.1000011011.1	К вершине 3
3	8.1	1000010100	1.0010101.0001111.0	–
4	8.2	1000010101	1.0010111.0000000.0	–
5	9	1000010110	0.0111.1000010100.0	К вершине 8
6	10	1000010111	1.0010001.0000000.0	–
7	11	1000011000	0.1111.1000010111.1	К вершине 10
8	12	1000011001	1.0001010.0000000.0	–
9	7	1000011010	1.0010111.0000000.1	–
10	3	1000011011	0.0100.1000011110.1	К вершине 5
11	4.1	1000011100	1.0110000.0000101.0	–
12	4.2	1000011101	1.1000010.0000000.0	–
13	5	1000011110	0.1111.1000011110.0	К вершине 5
14	6	1000011111	1.0000101.0000011.0	–
15	6.1	1000100000	0.0000.1000011010.1	К вершине 7

На данном примере рассмотрен алгоритм составления микропрограммы по заданной ГСА. Разобраны основные моменты при организации условных и безусловных переходов, а также способы с использованием поля модификатора и без него.

8.3 Задачи для решения

Вариант 1

Составить микропрограмму для реализации ГСА (рисунок 8.9, *а*).

Управляемый объект характеризуется следующими параметрами:

- множество проверяемых условий – 10;

- множество выполняемых операций – 90 (предусмотреть поле y_k, чтобы задать последнюю микрокоманду микропрограммы);
- емкость памяти для записи программы 1Кбайт;
- длина ячейки памяти – 16 бит;
- начальный адрес размещения микропрограммы равен 110.

Вариант 2

Составить микропрограмму для реализации ГСА (рисунок 8.9, *б*).

Управляемый объект характеризуется следующими параметрами:
- множество проверяемых условий – 20;
- множество выполняемых операций – 90 (предусмотреть поле y_k, чтобы задать последнюю микрокоманду микропрограммы);
- емкость памяти для записи программы – 512 адресов;
- длина ячейки памяти – 16 бит;
- начальный адрес размещения микропрограммы равен 520.

Вариант 3

Составить микропрограмму для реализации ГСА (рисунок 8.9, *а*).

Управляемый объект характеризуется следующими параметрами:
- множество проверяемых условий – 12;
- множество выполняемых операций – 72 (предусмотреть поле y_k, чтобы задать последнюю микрокоманду микропрограммы);
- емкость памяти для записи программы – 768 адресов;
- длина ячейки памяти – 16 бит;
- начальный адрес размещения микропрограммы равен 256.

Вариант 4

Составить микропрограмму для реализации ГСА (рисунок 8.9, *б*).

Управляемый объект характеризуется следующими параметрами:
- множество проверяемых условий – 30;
- множество выполняемых операций – 48 (предусмотреть поле y_k, чтобы задать последнюю микрокоманду микропрограммы);
- емкость памяти для записи программы – 400 адресов;
- длина ячейки памяти – 16 бит;
- начальный адрес размещения микропрограммы равен 128.

Вариант 5

Составить микропрограмму для реализации ГСА (рисунок 8.9, *в*).

Управляемый объект характеризуется следующими параметрами:
- множество проверяемых условий – 14;
- множество выполняемых операций – 100 (предусмотреть поле y_k, чтобы задать последнюю микрокоманду микропрограммы);
- емкость памяти для записи программы – 1000 адресов;

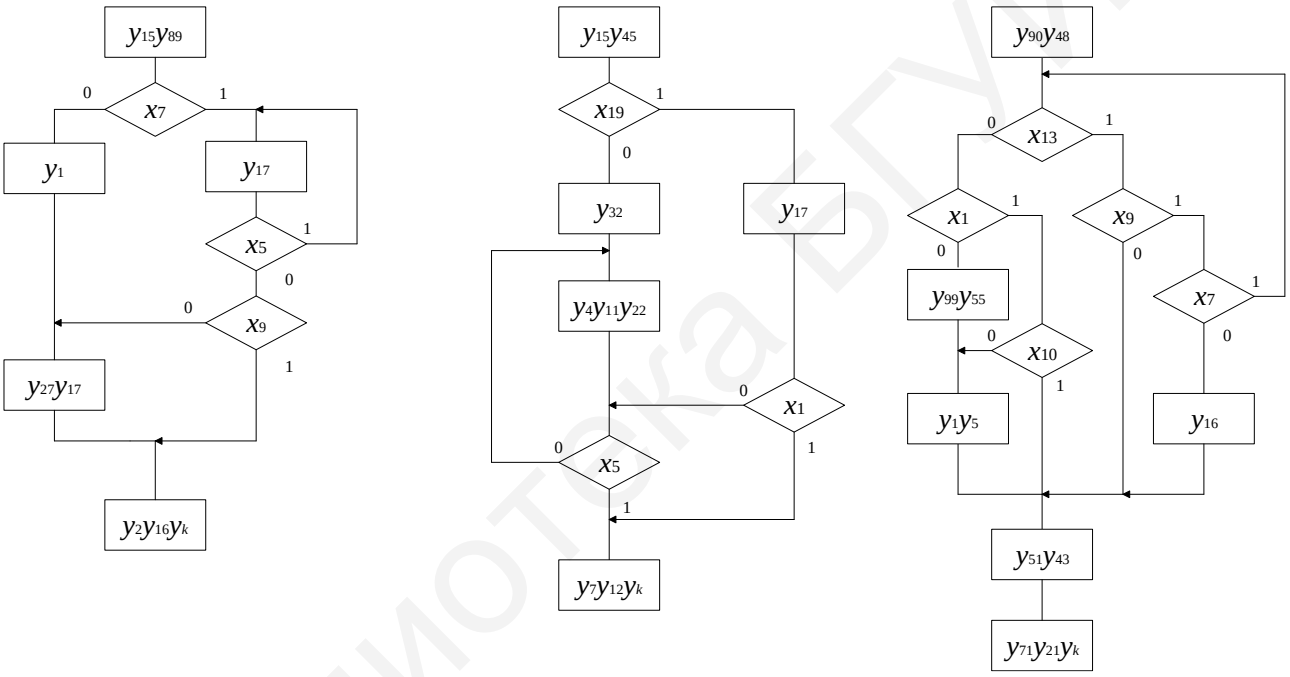
- длина ячейки памяти – 16 бит;
- начальный адрес размещения микропрограммы равен 400.

Вариант 6

Составить микропрограмму для реализации ГСА (рисунок 8.9, *в*).

Управляемый объект характеризуется следующими параметрами:

- множество проверяемых условий – 14;
- множество выполняемых операций – 100 (предусмотреть поле y_k, чтобы задать последнюю микрокоманду микропрограммы);
- емкость памяти для записи программы – 2 Кбайт;
- длина ячейки памяти – 16 бит;
- начальный адрес размещения микропрограммы равен нулю.



а *б* *в*

а – ГСА для задания 1, 3; *б* – ГСА для заданий 2, 4; *в* – ГСА для заданий 5, 6

Рисунок 8.9 – ГСА

8.4 Контрольные вопросы

1 Что такое АЛУ? Назовите основные блоки АЛУ.

2 Изобразите структурную схему микропрограммного устройства управления.

3 Назовите типы микрокоманд блока управления, поля микрокоманд.

4 В чем заключаются особенности составления микропрограмм с использованием модификатора дисциплины переходов и без него?

Литература

1. Гашков, С. Б. Системы счисления и их применение / С. Б. Гашков. – М. : МЦНМО, 2004. – 52 с.

2. Андреева, Е. Н. Системы счисления и компьютерная арифметика / Е. Н. Андреева, И. Н. Фалина. – 2-е изд. – М. : Лаборатория базовых знаний, 2000. – 248 с.

3. Савельев, А. Я. Основы информатики : учебник для вузов / А. Я. Савельев. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2001. – 328 с.

4. Луцик, Ю. А. Арифметические и логические основы вычислительной техники : учеб. пособие / Ю. А. Луцик, И. В. Лукьянова. – Минск : БГУИР, 2014. – 174 с.

5. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум. – 5-е изд. – СПб. : Питер, 2007. – 848 с.

6. Потехин, В. А. Схемотехника цифровых устройств / В. А. Потехин. – Томск : В-Спектр, 2012. – 250 с.

7. Мержи, И. Теория и практика применения цифровых логических микросхем / И. Мержи. – М. : НТ-Пресс, 2007. – 256 с.

8. Схемотехника электронных систем. Цифровые устройства / В. И. Бойко [и др.]. – СПб. : БХВ-Петербург, 2004. – 512 с.

9. Бибило, П. Н. Синтез логических схем с использованием языка VHDL / П. Н. Бибило. – М. : Солон-Пресс, 2009. – 385 с.

10. Савельев, А. Я. Прикладная теория цифровых автоматов : учебник для вузов по специальности ЭВМ / А. Я. Савельев. – М. : Высш. шк., 1987. – 272 с.

11. Карпов, Ю. Г. Теория автоматов / Ю. Г. Карпов. – СПб. : Питер, 2003. – 208 с.

Св. план 2019, поз. 68

Учебное издание

Савенко Андрей Геннадьевич
Матвеев Андрей Владимирович

ОСНОВЫ КОМПЬЮТЕРНОЙ ТЕХНИКИ. ПРАКТИЧЕСКИЕ ЗАНЯТИЯ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редактор *Е. В. Иванюшина*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *В. М. Задоля*

Подписано в печать 06.12.2019. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 5,0. Уч.-изд. л. 4,8. Тираж 50 экз. Заказ 132.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
Ул. П. Бровки, 6, 220013, г. Минск