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

2

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: «Систе- мы 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** | © Савенко А. Г., Матвеев А. В., 2020  © УО «Белорусский государственный университет информатики  и радиоэлектроники», 2020 |

##### СОДЕРЖАНИЕ

|  |  |
| --- | --- |
| Введение ………………..…………………………………………………………… | 4 |
| 1BПрактическое занятие №1 Переход из одной системы счисления в другую …… | 5 |
| 3B1.1 Краткие теоретические сведения ……………………………………………… | 5 |
| 1.2 Методические указания и примеры решения задач ………………………….. | 6 |
| 1.3 Задачи для решения …………………………………………………………….. | 14 |
| 1.4 Контрольные вопросы ………………………………………………………….. | 16 |
| 8BПрактическое занятие №2 Двоичная арифметика с положительными числами... | 17 |
| 10B2.1 Краткие теоретические сведения ………………………………..…………….. | 17 |
| 2.2 Методические указания и примеры решения задач ………………………….. | 17 |
| 2.3 Задачи для решения …………………………………………………………….. | 25 |
| 2.4 Контрольные вопросы ………………………………………………………….. | 26 |
| Практическое занятие №3 Арифметика с алгебраическими числами ………....... | 27 |
| 3.1 Краткие теоретические сведения ………………………………..…………….. | 27 |
| 3.2 Методические указания и примеры решения задач ………………………….. | 29 |
| 3.3 Задачи для решения …………………………………………………………….. | 33 |
| 3.4 Контрольные вопросы ………………………………………………………….. | 34 |
| Практическое занятие №4 Арифметика с плавающей точкой ………………….. | 35 |
| 4.1 Краткие теоретические сведения ………………………………..…………….. | 35 |
| 4.2 Методические указания и примеры решения задач ………………………….. | 36 |
| 4.3 Задачи для решения …………………………………………………………….. | 42 |
| 4.4 Контрольные вопросы ………………………………………………………….. | 42 |
| Практическое занятие №5 Синтез логических схем …………………………….. | 43 |
| 5.1 Краткие теоретические сведения ………………………………..…………….. | 243 |
| 5.2 Методические указания и примеры решения задач ………………………….. | 46 |
| 5.3 Задачи для решения …………………………………………………………….. | 52 |
| 5.4 Контрольные вопросы ………………………………………………………….. | 53 |
| Практическое занятие №6 Минимизация логических выражений ……………… | 54 |
| 6.1 Краткие теоретические сведения ………………………………..…………….. | 54 |
| 6.2 Методические указания и примеры решения задач ………………………….. | 55 |
| 6.3 Задачи для решения …………………………………………………………….. | 59 |
| 6.4 Контрольные вопросы ………………………………………………………….. | 60 |
| Практическое занятие №7 Синтез цифровых автоматов ………..……………….. | 61 |
| 7.1 Краткие теоретические сведения ………………………………..…………….. | 61 |
| 7.2 Методические указания и примеры решения задач ………………………….. | 63 |
| 7.3 Задачи для решения …………………………………………………………….. | 68 |
| 7.4 Контрольные вопросы ………………………………………………………….. | 68 |
| Практическое занятие №8 Составление микропрограмм ……….……………….. | 69 |
| 8.1 Краткие теоретические сведения ………………………………..…………….. | 69 |
| 8.2 Методические указания и примеры решения задач ………………………….. | 70 |
| 8.3 Задачи для решения …………………………………………………………….. | 78 |
| 8.4 Контрольные вопросы ………………………………………………………….. | 80 |
| Литература …………………………………………………………….…………….. | 81 |

###### Введение

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

Целью практических занятий являются закрепление теоретического кур- са, приобретение навыков решения задач и активизация самостоятельной рабо- ты студентов.

Практические задания выполняются студентами на практических заняти- ях индивидуально. Каждое занятие рассчитано на два академических часа ауди- торных 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 си- стемы 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 сло- жения, вычитания, деления и умножения.

Третье практическое занятие содержит задачи арифметики с алгебраиче- скими числами при использовании операций сложения/вычитания в дополни- тельном, обратном и модифицированном кодах.

Четвертое практическое занятие содержит задачи арифметики с плаваю- щей 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 минималь- ного выражения с использованием минимизации с помощью карт Карно.

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

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

##### ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №1

**ПЕРЕХОД ИЗ ОДНОЙ СИСТЕМЫ СЧИСЛЕНИЯ В ДРУГУЮ**

Цель: приобрести практические навыки решения задач перехода из за- данной системы счисления в искомую.

###### Краткие теоретические сведения

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

Позиционная система счисления характеризуется тем, что «доля» некото- рой 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. Количе- ственная 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 ве- сов двух любых соседних разрядов имеют для такой системы одинаковое зна- чение. Это соотношение называется основанием системы счисления, которое в дальнейшем будем обозначать как *q*.

Общая запись числа в системе с равномерно распределенными весами имеет вид

|  |  |
| --- | --- |
| 𝑁𝑞 = 𝐴𝑛𝐴𝑛−1 … 𝐴1𝐴0. | (1.1) |

Значение такого числа определяется следующим образом: где *Ai* – цифра записи числа (0 ≤ *Ai* ≤ *q* – 1);

|  |  |
| --- | --- |
| 𝑁𝑞 = 𝐴𝑛 · 𝑞𝑛+ 𝐴𝑛−1 · 𝑞𝑛−1 + ⋯ + 𝐴1 · 𝑞1+ 𝐴0 · 𝑞0, | (1.2) |

*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, превыша- ющих значение девять, вводится дополнительное обозначение – латинские бук- вы, начиная с буквы *А* (для *q* = 16 это будут обозначения *А, В, C, D, E, F*).

Следовательно, запись одного и того же числа в различных сис- темах счисления будет тем длиннее, чем меньше основание системы счис- ления.

###### Методические указания и примеры решения задач

Существование различных систем счисления предполагает необходи- мость перевода записи числа из одной системы счисления в другую. Существу- ет 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 соотно- шения 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, ка- кая система счисления (исходная или искомая) является более привычной, име- ет две разновидности.

1. Если более привычной является искомая система счисления, то на осно- вании расширенной записи исходного числа подсчитываются значения ее от- дельных разрядов в новой системе счисления. Далее полученные значения сум- мируются.

*Пример*

Необходимо перевести из двоичной системы счисления в десятичную число 𝑁2 = 1100110.

По формуле (1.2) формируем расширенную запись числа:

𝑁2 = 1100110 = 1 · 26 + 1 · 25 + 0 · 24 + 0 · 23 + 1 · 22 + 1 · 21 + 0 · 20.

Далее рассчитываем вес двоичных разрядов в десятичной системе счис-

ления:

𝑁2 = 1100110 = 64 + 32 + 0 + 0 + 4 + 2 + 0 = 10210.

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 осно- вания счисления. Помимо этого, необходимо учитывать, что при преобразова- нии правильных дробей в общем случае результат получается неточный и перед началом преобразования необходимо подсчитать количество разрядов пред- ставления числа в новой системе счисления. Разрядность результата выбираем

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

При преобразовании правильных дробей сначала находим предваритель- ное значение представления заданного числа в новой системе счисления с ко- личеством разрядов на единицу большим, чем расчетная разрядность представ- ления числа в новой системе счисления. Дополнительный разряд в предвари- тельном результате преобразования используем для округления, позволяющего с рассчитанным числом разрядов найти окончательный результат.

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

*Пример*

Необходимо перевести правильную двоичную дробь 𝑁2 = 0.111 в пра- вильную десятичную.

Перед началом преобразования определяется, что разрядность записи за- данного 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:

𝑁2 = 0.111 = 1 · 2−1 + 1 · 2−2 + 1 · 2−3 = 0,5 + 0,25 + 0,125 = 0,875.

После округления получаем 𝑁2 = 0.111 = 0,910.

1. 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 мак- симальным возможным весом, но не превышающим значение преобразуемого числа. При этом, определив старший разряд с ненулевым значением, из исход- ного числа вычитается вес этого разряда, таким образом формируя остаток, ко- торый должен быть представлен еще не найденным младшим разрядом иско- мой 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 𝑁10 = 234.

Первый (старший) разряд с весом 27 = 128 будет иметь значение 1 в иско-

мой двоичной записи числа. С помощью остальных (младших) разрядов иско- мой записи числа необходимо представить значение 106, т. к. 106 – это остаток, полученный как разность между числами 234 и 128.

Второй разряд с весом 26 = 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, полу- ченный как разность между числами 106 и 64.

Третий разряд с весом 25 = 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 24 = 16 будет иметь в искомой двоичной записи числа значение 0, а остаток остается прежним – равным 10.

Пятый разряд с весом 23 = 8 будет иметь в искомой двоичной записи чис- ла значение 1, а остаток, определяемый как разность между числами 10 и 8, ра- вен 2.

Шестой разряд с весом 22 = 4 будет иметь в искомой двоичной записи числа значение 0, а остаток остается прежним – равным 2.

Седьмой разряд с весом 21 = 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 20 = 1 будет иметь в искомой двоичной записи числа значение 0, остаток равен 0.

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

𝑁10 = 234 = 111010102.

Аналогично данный способ применяется при переводе правильных дро-

бей, только с отрицательными степенями основания.

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

*Пример*

Необходимо перевести правильную десятичную дробь 𝑁10 = 0,7 в пра- вильную дробь в двоичной системе счисления.

Предварительный результат находим с точностью до пяти двоичных раз- рядов, 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 ис- комой записи числа необходимо представить значение 0,2 (0,2 – остаток, полу- ченный 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, получен- ный как разность чисел 0,20 и 0,13).

Четвертый разряд с весом 2–4 = 0,06 в искомой двоичной записи числа бу- дет 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 0,710 = 0.101102.

После округления пятого младшего разряда имеем 0,710 = 0.10112.

*Метод деления/умножения на новое основание*

Данный универсальный метод также предполагает использование расши- ренной записи числа (1.2) и имеет две разновидности: для целых и дробных чи- сел.

1. *Преобразование целых чисел*

Задачу представления числа *N*, заданного в системе *q*1, в системе счисле- ния с основанием *q*2 можно рассматривать как задачу поисков коэффициентов полинома, представляющего собой расширенную запись числа *N* в системе счисления *q*2:

|  |  |
| --- | --- |
| 𝑁𝑞1 = 𝐴0 + 𝐴1 · 𝑞1+ 𝐴2 · 𝑞2 + ⋯ + 𝐴𝑛−1 · 𝑞𝑛−1+ 𝐴𝑛 · 𝑞𝑛 = 𝑁𝑞2.  2 2 2 2 | (1.3) |

Преобразуем выражение (1.3), вынеся общий знаменатель за скобку (ри- сунок 1.1):

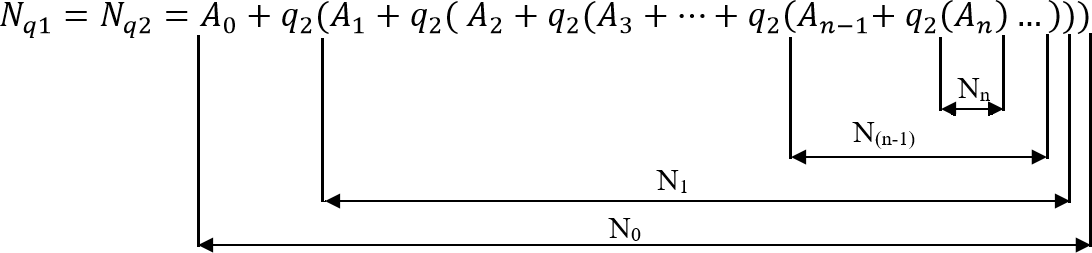
.

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 – *Nn*. Теперь, принимая во внимание выражение (1.3), можно утверждать, что при делении *Nq*1 на *q*2 будут получены целая часть частного (обозначим ее *int* (*Nq*1/*q*2)) и остаток (обозначим его

*rest* (*Nq*1/*q*2)), равный *A*0. Тогда остальные скобки будут иметь следующий вид:

𝑁1 : целая часть *int* (*N*1/*q*2), равная *N*2, и остаток *rest* (*N*1/*q*2), равный *A*1;

𝑞2

𝑁2 : целая часть *int* (*N*2/*q*2), равная *N*3, и остаток *rest* (*N*2/*q*2), равный *A*2;

𝑞2

…

𝑁(𝑛−2) : целая часть *int* (*N*(*n –* 2)/*q*2), равная *N*(*n* – 1), и остаток *rest* (*N*(*n –* 2)/*q*2),

𝑞2

равный *A*(*n –* 2);

𝑁(𝑛−1) : целая часть *int* (*N*(*n* – 1)/*q*2), равная *Nn* = *An*, и остаток *rest* (*N*(*n* – 1)/*q*2),

𝑞2

равный *A*(*n* – 1).

При этом *Nn* = *An* (*Nn* < *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 *Nq*1 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 фор- мирования 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 разря- дов используются значения остатков в порядке, обратном порядку их получе- ния.

*Пример*

Перевести десятичное число 𝑁10 = 432 в двоичную систему счисления методом деления на новое основание.

Согласно алгоритму сначала делим исходное число *N*10, а затем и получа- емые частные делим на значение нового основания *q* = 2 до получения частного со значением, меньшим чем 2:

432 : *int* (432/2) = 216 и *rest* (432/2) = 0;

2

216 : *int* (216/2) = 108 и *rest* (216/2) = 0;

2

108 : *int* (108/2) = 54 и *rest* (108/2) = 0;

2

54 : *int* (54/2) = 27 и *rest* (54/2) = 0;

2

27 : *int* (27/2) = 13 и *rest* (27/2) = 1;

2

13 : *int* (13/2) = 6 и *rest* (13/2) = 1;

2

6 : *int* (6/2) = 3 и *rest* (6/2) = 0;

2

3 : *int* (3/2) = 1 и *rest* (3/2) = 1.

2

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

Таким образом, 𝑁10 = 432 = 1101100002.

1. *Преобразование дробных чисел*

Задачу представления дробного числа *Mq*1 в новой системе счисления с основанием *q*2 можно рассматривать как поиск коэффициентов полинома, пред- ставляющего собой расширенную запись числа *M* в системе счисления *q*2:

|  |  |
| --- | --- |
| 𝑀 = 𝐵 · 𝑞−1+ 𝐵 · 𝑞−2 + ⋯ + 𝐵 · 𝑞−(𝑛−1)+ 𝐵 · 𝑞−𝑛 = 𝑀 .  𝑞1 1 2 2 2 𝑛−1 2 𝑛 2 𝑞2 | (1.4) |

Преобразуем выражение (1.4), вынеся общий знаменатель за скобку (ри- сунок 1.2):

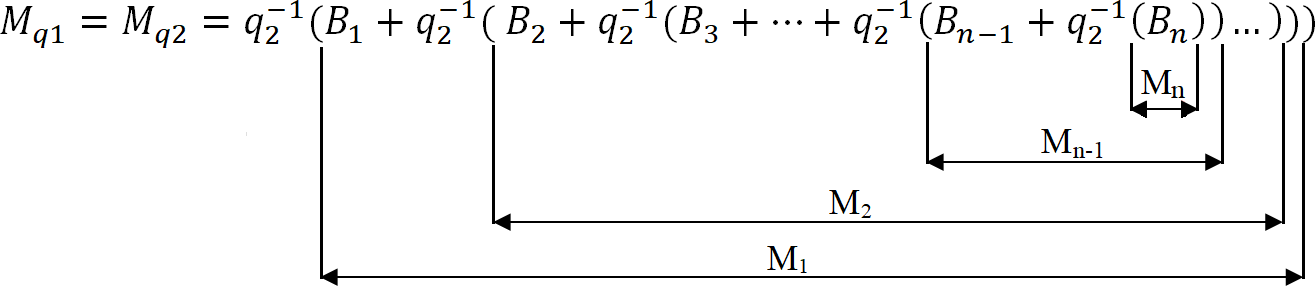
.

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 – *Mn* – 1 и выражение в *n*-й скобке – *Mn*.

Так как число *Mq*1 – правильная дробь, то при умножении *Mq*1 на основа- ние *q*2 будет получено произведение, в общем случае состоящее из целой части *int* (*Mq*1·*q*2) и дробной части *DF* (*Mq*1·*q*2). Тогда остальные скобки будут иметь следующий вид:

*M*1·*q*2 = (*int* (*M*1·*q*2) = *B*2) + (*DF* (*M*1·*q*2) = *M*2);

*M*2·*q*2 = (*int* (*M*2·*q*2) = *B*3) + (*DF* (*M*2·*q*2) = *M*3);

…

*Mn* – 1·*q*2 = (*int* (*Mn*-1·*q*2) = *Bn*) + (*DF* (*Mn* – 1·*q*2) = *Mn*); *Mn*·*q*2 = (*int* (*Mn*·*q*2) = *Bn*+1) + (*DF* (*Mn*·*q*2) = *Mn*+1).

Отсюда следует правило формирования коэффициентов полинома, кото-

рые и являются разрядами записи заданного числа *M* в системе счисления с ос- нованием *q*2:

* определяется количество разрядов *n* в записи числа *Mq*2 в новой системе счисления. Разрядность результата выбирается таким образом, чтобы ошибка представления результата была бы не более половины единицы младшего раз- ряда в исходной записи числа;
* исходное число *Mq*1 умножается на *q*2, при этом будет получено сме- шанное число;
* дробная часть полученного произведения снова умножается на *q*2 и т. д. Процесс умножения повторяется *n* раз. В качестве искомых разрядов новой за- писи числа используются результаты выполненных операций умножения сле- дующим 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 си- стеме 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 ме- тодом умножения на новое основание.

Сначала необходимо определить количество разрядов числа *M* в двоич- ной системе счисления. Так как исходная запись числа содержит один десятич- ный разряд, то запись данного числа в двоичной системе должна содержать че- тыре разряда. Учитывая округление, находим предварительный двоичный эк- вивалент с пятью разрядами.

Умножаем исходное число *M*10, а затем дробные части последовательно получаемых произведений на новое основание *q* = 2:

0,7·2 = 1,4 (*int* (0,7·2) = 1 и *DF* (0,7·2) = 0,4);

0,4·2 = 0,8 (*int* (0,4·2) = 0 и *DF* (0,4·2) = 0,8);

0,8·2 = 1,6 (*int* (0,8·2) = 1 и *DF* (0,8·2) = 0,6);

0,6·2 = 1,2 (*int* (0,6·2) = 1 и *DF* (0,6·2) = 0,2);

0,2·2 = 0,4 (*int* (0,2·2) = 0 и *DF* (0,2·2) = 0,4).

Таким образом, *M*10 = 0,7 = 0.101102.

После округления имеем *M*10 = 0,710 = 0.10112.

*Метод преобразования с использованием особого соотношения основа- ний заданной и искомой систем счисления*

Данный метод не является универсальным и применим тогда, когда ис- ходное *q*1 и новое *q*2 основания могут быть связаны через целую степень, т. е. когда выполняется одно из двух условий:

|  |  |
| --- | --- |
| 𝑞𝑚 = 𝑞1,  2 | (1.5) |

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

|  |  |
| --- | --- |
| 𝑞𝑚 = 𝑞2,  1 | (1.6) |

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

Если выполняется условие (1.5), тогда запись числа 𝑁𝑞1 = 𝑎𝑛𝑎𝑛−1 … 𝑎1𝑎0

в системе с новым основанием *q*2 определяется следующим образом:

- каждому разряду *аi* 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 объедине- ния всех полученных *m*-разрядных групп.

*Пример*

Перевести восьмеричное число *N*8 = 47601.62 в двоичную систему счис- ления.

Основания исходной и новой систем счисления можно выразить через це- лую степень следующим образом: 23 = 8.

Согласно алгоритму ставим в соответствие каждой цифре исходной запи- си восьмеричного числа трехразрядный двоичный эквивалент (триады) (табли- ца 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 полу- ченных триад двоичного представления восьмеричных цифр в единый двоич- ный эквивалент с учетом точки, отделяющей целую и дробную части:

47601.628 = 100111110000001.1100102.

Если выполняется условие (1.6), тогда запись числа 𝑁𝑞1 = 𝑎𝑛𝑎𝑛−1 … 𝑎1𝑎0

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 систе- мы 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 шестнадцатерич- ную систему счисления.

Основания исходной и новой систем счисления можно выразить через це- лую степень следующим образом: 24 = 16.

Разбиваем исходную запись числа на группы по четыре разряда вправо и

влево от точки, в крайних левой и правой группах недостающие разряды запол- няем нулями и каждой полученной группе из четырех разрядов ставим в соот- ветствие 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 недо- стающие разряды в крайней правой и крайней левой от точки тетрадах (группа из четырех разрядов), которые заполняются нулями.

Формируем окончательный результат посредством объединения полу- ченных цифр в единый шестнадцатеричный эквивалент с учетом точки, отде- ляющей целую и дробную части:

1101111100.11101002 = 37C.E816.

Следует также отметить, что применять метод преобразования с исполь- зованием 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, ко- торая может быть связана с заданной и искомой системами одновременно усло- виями (1.5) и (1.6), то можно сперва найти эквивалент заданного числа в про- межуточной 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 = 27514.738 в шестнадцатеричную си- стему счисления.

Согласно условиям (1.5) и (1.6) промежуточной системой счисления в данном случае может являться двоичная система счисления: 23 = 8 и 24 = 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 27514.738 = 010111101001100.1110112.

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 под- черкнутый 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, 27514.738 = 010111101001100.1110112 = 2F4C.EC16.

###### Задачи для решения

Задание 1

Сформировать расширенную запись двоичных чисел:

|  |  |  |
| --- | --- | --- |
| а) 10110102; | г) 1010101012; | ж) 11001102; |
| б) 1000110112; | д) 10001102; | з) 10001012; |
| в) 1111101002; | е) 11101112; | и) 101010112. |

Задание 2

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

а) 1100102; в) 1110112; д) 1110002;

б) 101011012; г) 100101112; е) 101110102.

Задание 3

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

а) 0.10102; в) 0.11112; д) 0.10012;

б) 0.0101012; г) 0.0010112; е) 0.0111002.

Задание 4

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

а) 13410; д) 19310; и) 17710;

б) 40610; е) 44410; к) 47910;

в) 0,8210; ж) 0,6810; л) 0,7210;

г) 0,4510; з) 0,3310; м) 0,2410.

Задание 5

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

|  |  |  |
| --- | --- | --- |
| а) *N*10 = 146; *N*2 = … ; | д) *N*10 = 125; *N*2 = … ; | и) *N*10 = 199; *N*2 = … ; |
| б) *N*10 = 315; *N*8 = … ; | е) *N*10 = 350; *N*8 = … ; | к) *N*10 = 399; *N*8 = … ; |
| в) *N*10 = 0,48; *N*2 = … ; | ж) *N*10 = 0,54; *N*2 = … ; | л) *N*10 = 0,69; *N*2 = … ; |
| г) *N*10 = 0,77; *N*8 = … ;  Задание 6 | з) *N*10 = 0,83; *N*8 = … ; | м) *N*10 = 0,89; *N*8 = … . |

Перевести заданные целые и дробные числа из двоичной системы счис- ления в восьмеричную, используя метод особого соотношения оснований си- стем счисления:

а) 11001102; д) 10101012; и) 11101002;

б) 1011000012; е) 1101011012; к) 10011110112;

в) 10110.10012; ж) 11000.00112; л) 10001.11112;

г) 1101.010112; з) 1110.011012; м) 10111.1110112.

Задание 7

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

|  |  |  |
| --- | --- | --- |
| а) 2AF47916; | г) A52C8316; | ж) 4AAC5F16; |
| б) 4C6B5.1E16; | д) 2DA04.4C16; | з) A62B1.E816; |
| в) 3AB7.C216; | е) 15DA.5C916; | и) 5E6B.C0A16. |

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) 534728; в) 325708; д) 167358;

б) 25601.4218; г) 63042.7418; е) 42065.1728.

Задание 9

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

а) 65836210 ; в) 35076110; д) 25710810;

б) 13902510; г) 27340110; е) 12345610.

Задание 10

Перевести заданные целые и дробные числа из заданной системы счисле- ния в искомую:

|  |  |  |
| --- | --- | --- |
| а) *N*7 = 146; *N*2 = … ; | д) *N*6 = 125; *N*2 = … ; | и) *N*9 = 78; *N*2 = … ; |
| б) *N*8 = 1542; *N*5 = … ; | е) *N*8 = 1242; *N*4 = … ; | к) *N*8 = 2715; *N*6 = … ; |
| в) *N*10 = 0,17; *N*2 = … ; | ж) *N*10 = 0,54; *N*2 = … ; | л) *N*10 = 0,75; *N*2 = … ; |
| г) *N*16 = 0.4A8; *N*10 = … ; | з) *N*16 = 0.B41; *N*10 = … ; | м) *N*16 = 0.1C8; *N*10 = … . |

###### Контрольные вопросы

1. Какие методы перевода из заданной системы счисления в искомую яв- ляются 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 преоб- разовании 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 соотноше- ния 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.

###### Краткие теоретические сведения

В настоящее время практически не осталось аналоговой техники. Все устройства – от бытовых приборов до космических аппаратов – являются циф- ровыми. Как известно, цифровые устройства, в частности компьютерная техни- ка, при обработке информации использует в основном двоичную систему счис- ления.

Именно поэтому в большей своей части компьютерная арифметика явля- ется двоичной арифметикой. Этому есть объяснение. Во-первых, алгоритмы арифметических операций двоичной арифметики (т. е. арифметики, использу- ющей двоичную позиционную систему) очень просты и являются в определен- ном 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 алгеб- ры логики.

В двоичной системе счисления арифметические операции (сложение, вы- читание, умножение и деление) выполняются по тем же правилам, что и в при- вычной нам десятичной системе счисления, т. к. они обе являются позицион- ными (наряду с восьмеричной, шестнадцатеричной и др.).

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

###### 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 ис- пользуются всего две цифры (нуль и единица), то нагляднее всего можно отоб- разить эти правила в виде таблицы, где будет всего четыре возможные комби- нации результата выполнения операций сложения, вычитания и умножения.

*Сложение положительных двоичных чисел*

В общем виде правила сложения двоичных положительных чисел выгля- дят следующим образом:

* два двоичных положительных числа складываются в столбик поразряд- но, 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, рав- ная нулю с переносом единицы в старший разряд;
* при переносе единицы из младшего в старший разряд сначала склады- ваются значения соответствующих разрядов двух двоичных чисел по вышеопи- санным правилам, затем к полученной поразрядной сумме прибавляется пере- несенная 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 101100111012 и 111010001102.

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 раз- рядах. Лишняя единица в одиннадцатом разряде переходит в двенадцатый. Та- ким образом, конечное значение (сумма) будет иметь уже не одиннадцать, а двенадцать разрядов.

В общем случае при формировании значения в текущем разряде резуль- тата 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 вы- глядят 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 раз- ность, равная единице;
    - при вычитании из единичного разряда единичного разряда получается разность, равная нулю (также нулевой разряд);
    - при вычитании из нулевого разряда единичного разряда получается раз- ность, 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 |
| Вычи- таемое | 0 | **0** | **1\*** | **1** |
| 1 | **1\*** | **0\*** | **0** |
| *Примечание* – \* – заем единицы из старшего разряда. | | | | |

*Пример*

Найти разность двух положительных двоичных чисел 11001012 и 1011102. Выполняем вычитание поразрядно в столбик, начиная с младшего разряда:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | • | • | • | • | • |  |  |
| – | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 |
|  | 0 | 1 | 1 | 0 | 1 | 1 | 1 |

При вычитании во втором разряде из нуля единицы происходит заем еди- ницы из старшего третьего разряда, а разность получается равная единице. Та- ким образом, после заема в третьем разряде для вычитания необходимо прове- сти 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 «нуле- вой» разряд, из которого уже происходил заем в младший разряд, а вычитае- мое – 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 седь- мого разряда, а разность получается равная единице. В седьмом разряде уменьшаемым и вычитаемым будут являться нулевые разряды, поэтому раз- ность будет равняться также нулю.

*Умножение двоичных положительных чисел*

В общем виде правила умножения двоичных положительных чисел вы- глядят следующим образом:

* два двоичных положительных числа умножаются в столбик поразрядно, при этом формирование произведения выполняется за счет сложения частич- ных 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 раз- ряд каждого частичного произведения при умножении с младшего или старше- го 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 |
| Мно- житель | 0 | **0** | **0** |
| 1 | **0** | **1** |

*Пример*

Найти произведение двух положительных двоичных чисел 10010112 и 1101012, 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 по- ложительных двоичных чисел. Начиная со второго частичного произведения происходит сдвиг следующего частичного произведения на один разряд влево. Количество разрядов частичных произведений соответствует количеству разря- дов множимого.

*Пример*

Найти произведение двух положительных двоичных чисел 10010112 и 1101012, 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 множи- мого. Следует отметить, что при суммировании частичных произведений, сло- жение 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 полу- чается (2×*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 0.1102 и 0.0112, 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 положи- тельные 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 количе- ство разрядов частного, которые подлежат определению.

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

* упрощается процедура подбора очередной цифры т. к. в двоичной си- стеме очередной цифрой может быть одна из двух: либо нуль, либо единица;
* упрощается процедура умножения найденной цифры частного на дели-

тель.

В общем виде правила деления двоичных положительных чисел выглядят

следующим образом:

* два двоичных положительных числа делятся в столбик, как и десятич- ные. Сначала в делимом со старшего разряда берется *n* разрядов, где *n* – число разрядов в делителе.
* если число, образуемое этими *n* разрядами, больше делителя, то записы- вается единичный старший разряд частного, а от этих *n* старших разрядов де- лимого отнимается делитель, получается первая частичная разность;
* если число, образуемое этими *n* разрядами, меньше делителя, то записы- вается нулевой старший разряд частного, а от этих *n* старших разрядов делимо- го 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 ча- стичной разности получается число, большее делителя, то записывается еди- ничный старший разряд частного и вычитается очередная частичная разность;
* если после добавления последующего старшего разряда делимого к ча- стичной разности получается число, меньшее делителя, то добавляется следу- ющий старший разряд делимого, а в частное записывается нулевой разряд и вычитается очередная частичная разность;
* вычитание частичной разности выполняется по правилам вычитания по- ложительных 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 ненуле- вой – в противном случае).

*Пример*

Найти частное положительных двоичных чисел 100011112 и 11012.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 проис- ходит заем единицы по рассмотренным выше правилам вычитания положи- тельных двоичных чисел. Полужирным шрифтом отмечены очередные добав- ляемые к частичной разности разряды делимого.

*Сложение и вычитание положительных двоично-десятичных чисел*

Двоично-десятичная система счисления представляет собой синтез дво- ичной и десятичной систем счисления. Каждый разряд десятичного числа (каж- дая 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 чи- сел в общем виде выглядят следующим образом:

* необходимо добавить двоичную шестерку (0110) в те тетрады, из кото- рых был перенос при их переполнении при двоичном сложении. Необходи- мость 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, сформирован- ный 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, сформи- рованный по правилам двоичного вычитания, приносит в тетраду значение, рав- ное шестнадцать, а для десятичного вычитания заем должен был принести в тет- раду значение, равное десяти. Заем, сформированный по правилам двоичного вычитания, принес значение, на шестерку больше необходимого;
* необходимо добавить двоичную шестерку (0110) в те тетрады, в кото- рых получено значение, большее девяти. Такая коррекция обуславливается тем, что по правилам десятичной арифметики в таких тетрадах должен быть выра- ботан перенос, и, чтобы его выработать, по правилам двоичной арифметики в тетраду нужно добавить значение, равное шести.

*Пример*

Найти сумму десятичных чисел *А* = 392710 и *B* = 485610, используя двоич- но-десятичную систему счисления.

В соответствие каждому разряду десятичных чисел *A* и *B* находим двоич- ные 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 арифме- тики 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 переполне- ние и перенос в старшую тетраду, а звездочкой – тетрада, в которой в результа- те двоичного сложения получилось значение, большее девяти (тетрада превы-

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

*Пример*

Найти разность десятичных чисел *B* = 485610 и *А* = 392710, используя дво- ично-десятичную систему счисления.

В соответствие каждому разряду десятичных чисел *A* и *B* находим двоич- ные 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).

###### 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; д) *A* = 11100101, *B* = 1010101; б) *A* = 11101011, *B* = 11010101; е) *A* = 10010111, *B* = 10111001; в) *A* = 0.0101, *B* = 0.1001; ж) *A* = 0.0111, *B* = 0.1011;

г) *A* = 0.1011, *B* = 0.1111; з) *A* = 0.1111, *B* = 0.1101.

Задание 5

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

а) *A* = 101010000, *B* = 11100; в) *A* = 100111000, *B* = 11000;

б) *A* = 100001100010, *B* = 100101; 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; в) *A* = 1073, *B* = 2541;

б) *A* = 3744, *B* = 2916; 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; в) *A* = 2821, *B* = 1343;

б) *A* = 3285, *B* = 2563; г) *A* = 3615, *B* = 2674.

###### Контрольные вопросы

1. Чему будет равна сумма двух единичных разрядов при сложении поло- жительных двоичных чисел?
2. Как происходит формирование разности нулевого и единичного разря- дов положительных двоичных чисел?
3. В чем заключается особенность формирования разности двух положи- тельных двоичных чисел, если из разряда уменьшаемого происходил заем еди- ницы в младший разряд?
4. В какую сторону происходит сдвиг частичных произведений при умно- жении двух положительных двоичных чисел, начиная с младшего разряда? Начиная со старшего разряда?
5. Какое количество разрядов делимого используется на первом этапе де- ления 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 двоич- ной 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 модифици- рованных кодах.

###### 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 разря- дом, в котором устанавливается нуль, если число положительное, и единица, если число отрицательное. Поле модуля отражает количественную оценку чис- ла и для каждого кода формируется по-разному. Количество разрядов поля мо- дуля 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) раз-

ряд знакового поля.

При кодировании числа в прямой код (ПК) запись целого числа *А* форми- руется по следующему правилу:

|  |  |
| --- | --- |
| [𝐴] = 0. 𝐴, если 𝐴 ≥ 0,  ПК {1. |𝐴|, если 𝐴 < 0. | (3.1) |

При кодировании числа в дополнительный код (ДК) запись целого числа

*А* формируется по следующему правилу:

|  |  |
| --- | --- |
| [𝐴] = {0. 𝐴, если 𝐴 ≥ 0,  ДК 1. (𝑞𝑛 + 𝐴), если 𝐴 < 0, | (3.2) |

где *q* – основание системы счисления;

*n* – разрядность модульного поля;

*qn* – максимальная невключенная граница диапазона изменения представляе- мых чисел.

При кодировании числа в дополнительный код запись дробного числа *B*

формируется по следующему правилу:

|  |  |
| --- | --- |
| 𝐵 = 0. 𝐵, если 𝐵 ≥ 0,  ДК {1. (1 + 𝐵), если 𝐵 < 0, | (3.3) |

где 1 – максимальная невключенная граница диапазона изменения представля- емых чисел.

При кодировании числа в обратный код (ОК) запись целого числа *А* фор- мируется по следующему правилу:

|  |  |
| --- | --- |
| [𝐴] = 0. 𝐴, если 𝐴 ≥ 0,  ОК {1. ((𝑞𝑛 − 1) + 𝐴), если 𝐴 < 0, | (3.4) |

где *q* – основание системы счисления;

*n* – разрядность модульного поля;

*qn* – 1 – максимальная включенная граница диапазона изменения представ- ляемых чисел.

При кодировании числа в обратный код запись дробного числа *B* форми- руется по следующему правилу:

|  |  |
| --- | --- |
| 𝐵 = 0. 𝐵, если 𝐵 ≥ 0,  ОК {1. ((1 − 𝑞𝑛) + 𝐵), если 𝐵 < 0, | (3.5) |

где 1 – *qn* – максимальная включенная граница диапазона изменения представ- ляемых чисел.

Таким образом, из вышеописанного следует:

* положительное алгебраическое число в прямом, обратном и дополни- тельном 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 об-

ратном 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 вычи- тания заменяется на операцию сложения с изменением знака второго операнда (изменением значения разряда знакового поля на противоположное) и сложение осуществляется по правилам двоичной арифметики.

###### 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 про- инвертировать 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. проинвер- тировать модуль прямого кода) и к полученному коду прибавить единицу в младший разряд;
    - для перевода из дополнительного кода назад в прямой также необходи- мо 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 сложе- ния в обратном и дополнительном кодах выглядят следующим образом:

* + - при сложении чисел, представленных в дополнительном коде, выполня- ется сложение разрядов, представляющих запись операндов, по правилам дво- ичной арифметики по всей длине записи чисел, не обращая внимание на грани- цу, разделяющую знаковое и модульное поля. Переполнение разрядности зна- кового 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, раз- деляющую 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 резуль- тате такого сложения будет получен обратный код суммы заданных операндов.

*Пример*

Сформировать запись десятичных чисел *A* = 18210 и *B* = –10710 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* = 18210 = +101101102, *B* = –10710 = –11010112.

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

*A* = 18210 = +0101101102, *B* = –10710 = –0011010112.

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 пред- ставлением в прямом коде:

[*A*]ПК = [*A*]ОК = [*A*]ДК = 0.010110110.

Число *B* – отрицательное и будет иметь следующее представление в пря- мом коде:

[*B*]ПК = 1.001101011.

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

111111111 + (–001101011) = 110010100.

Таким образом, [*B*]ОК = 1.110010100.

Для определения модульной части числа *B* в дополнительном коде необ- ходимо прибавить к невключенной границе диапазона (2*n* = 1000000000) число *В* (по формуле (3.2)):

1000000000 + (–001101011) = 110010101.

Таким образом, [*B*]ДК = 1.110010101.

Поскольку в обратном и дополнительном кодах операция вычитания за- меняется на операцию сложения с заменой знака вычитаемого на противопо- ложный, приведем необходимые операции в соответствующий вид:

*С*1 = *А* + *В*, *С*2 = *А* + (–*В*), *С*3 = *В* + (–*А*), *С*4 = (–*А*) + (–*В*).

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 *С*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 |
| корректировка единицей в младший раз- ряд при переполнении знакового поля – | + | ~~1~~ | 0. | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
|  |  |  |  |  |  |  |  |  |  | 1 |
| [*C*1]ок = [*C*1]пк = |  |  | 0. | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1. |

В рассмотренном примере нахождения значения числа *С*1 возникает пе- реполнение разрядности знакового поля, которое учитывается как плюс едини- ца в младший разряд согласно правилам сложения в обратном коде.

Находим значение *С*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 *С*2 вычитание за- меняется на сложение и второе слагаемое инвертируется и берется с противо- положным знаком (–*B*).

Находим значение *С*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 *С*3 вычитание за- меняется на сложение, и второе слагаемое инвертируется и берется с противо- положным знаком (–*A*). Переполнение разрядности знакового поля по правилам сложения в дополнительном коде игнорируется. В результате получается отри- цательная сумма в дополнительном коде, которую необходимо проинвертиро- вать и добавить единицу в младший разряд для перевода в прямой код.

Находим значение *С*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 *С*4 вычитание за- меняется на сложение, и оба слагаемых инвертируются и берутся с противопо- ложным знаком (–*A* и –*B*). В результате получается отрицательная сумма в до- полнительном коде, которую необходимо проинвертировать и добавить едини- цу 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 мо- дульного поля. При этом, если в поле знака имеет место значение 01, то результат положительный, а если значение 10, то полученный результат отрицательный (ос- новным носителем знака числа является левый разряд знакового поля).

*Пример*

Найти значения чисел *C*1 =*A* + *B*, *C*2 = *A* – *B*, используя модифицирован- ный обратный код (МОК), и значения чисел *C*3 = *B* – *A*, *C*4 = –*A* – *B*, используя модифицированный дополнительный код (МДК), если *A* = 101002 и *B* = –011102. Результат представить в модифицированном прямом коде (МПК).

Поскольку в обратном и дополнительном кодах операция вычитания за- меняется на операцию сложения с заменой знака вычитаемого на противопо- ложный, приведем необходимые операции в соответствующий вид:

*С*1 = *А* + *В*, *С*2 = *А* + (–*В*), *С*3 = *В* + (–*А*), *С*4 = (–*А*) + (–*В*).

Находим необходимые значения чисел в модифицированных кодах (мо- дифицированный 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 *С*1 = *A* + *B*, используя модифицированный обратный код:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [*A*]мок = |  | + | 0 | 0. | 1 | 0 | 1 | 0 | 0 |
| [*B*]мок = | 1 | 1. | 1 | 0 | 0 | 0 | 1 |
| корректировка единицей в младший разряд при переполнении разрядности знакового поля – | + | ~~1~~ | 0 | 0. | 0 | 0 | 1 | 0 | 1 |
|  |  |  |  |  |  |  | 1 |
| [*C*1]мок = [*C*1]мпк = |  |  | 0 | 0. | 0 | 0 | 1 | 1 | 0. |

В рассмотренном примере нахождения значения числа *С*1 возникает пе- реполнение разрядности знакового поля, которое учитывается как плюс едини- ца в младший разряд согласно правилам сложения в обратном коде. Перепол- нения модульного поля не произошло, поскольку разряды знакового поля оди- наковы.

Находим значение *С*2 = *A* + (–*B*), используя модифицированный обратный

код:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [*A*]мок = | + | 0 | 0. | 1 | 0 | 1 | 0 | 0 |  |
| [–*B*]мок = | 0 | 0. | 0 | 1 | 1 | 1 | 0 |  |
| переполнение модульного поля, т. к. зна- ковые разряды имеют разное значение – |  | 0 | 1. | 0 | 0 | 0 | 1 | 0 | – сдвинуть точку влево |
| [*C*2]мок = [*C*2]мпк = | 0 | 0. | 1 | 0 | 0 | 0 | 1 | 0. |  |

В рассмотренном примере нахождения значения числа *С*2 возникает пе- реполнение модульного поля, поскольку разряды знакового поля различны. Младший разряд знакового поля необходимо перенести в модульное поле. Зна- чение суммы положительно, поскольку носитель знака – старший разряд знако- вого поля – равен нулю.

Находим значение *С*3 = *B* + (–*A*), используя модифицированный дополни- тельный код:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [*B*]мдк = | + | 1 | 1. | 1 | 0 | 0 | 1 | 0 |  |
| [–*A*]мдк = | 1 | 1. | 0 | 1 | 1 | 0 | 0 |  |
| переполнение модульного поля, т. к. зна- ковые 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 *С*3 возникает пе- реполнение модульного поля, поскольку разряды знакового поля различны. Младший разряд знакового поля необходимо перенести в модульное поле. Зна- чение суммы отрицательно, поскольку носитель знака – старший разряд знако- вого поля – равен единице, и для получения записи числа в прямом коде необ- ходимо проинвертировать модульное поле и прибавить единицу в младший разряд модульного поля.

Находим значение *С*4 = (–*A*) + (–*B*), используя модифицированный допол- нительный 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 *С*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 проинвер- тировать модульное поле и прибавить единицу в младший разряд.

###### Задачи для решения

Задание 1

Сформировать запись десятичных чисел *A*10 и *B*10 в прямом, обратном и дополнительном двоичных кодах:

а) *A* = 20710, *B* = –15510; в) *A* = 28610, *B* = –17710;

б) *A* = –32810, *B* = 24610; г) *A* = –36710, *B* = 29410.

Задание 2

Найти значения чисел *C*1 = *A* + *B*, *C*2 = *A – B*, *C*3 = *B* – *A*, *C*4 = –*A* – *B*, ис- пользуя обратный двоичный код. Результаты представить в прямом коде:

а) *A* = 12810, *B* = 11210; г) *A* = 14110, *B* = 10810;

б) *A* = 24710, *B* = –17910; д) *A* = 26910, *B* = –20510;

в) *A* = 21210, *B* = –30310; е) *A* = 25010, *B* = –32110.

Задание 3

Найти значения чисел *C*1 = *A* + *B*, *C*2 = *A – B*, *C*3 = *B* – *A*, *C*4 = –*A* – *B*, ис- пользуя дополнительный двоичный код. Результаты представить в прямом дво- ичном коде:

а) *A* = 14710, *B* = 13210; в) *A* = 26510, *B* = –33310; д) *A* = 29910, *B* = –28110 б) *A* = 28810, *B* = –22810; г) *A* = 16210, *B* = 15110; е) *A* = 27110, *B* = –35910.

Задание 4

Найти значения чисел *C*1 = *A* + *B*, *C*2 = *A – B*, *C*3 = *B* – *A*, *C*4 = –*A* – *B*, ис- пользуя модифицированный обратный двоичный код. Результаты представить в модифицированном прямом двоичном коде:

а) *A* = 11610, *B* = 10110; в) *A* = 21710, *B* = –31010; д) *A* = 24410, *B* = –21310; б) *A* = 22210, *B* = –16610; г) *A* = 15910, *B* = 11110; е) *A* = 26010, *B* = –30910.

Задание 5

Найти значения чисел *C*1 = *A* + *B*, *C*2 = *A – B*, *C*3 = *B* – *A*, *C*4 = –*A* – *B*, ис- пользуя модифицированный дополнительный двоичный код. Результаты пред- ставить в модифицированном прямом двоичном коде:

а) *A* = 13510, *B* = 10310; г) *A* = 16110, *B* = 12910;

б) *A* = 29910, *B* = –26710; д) *A* = 25610, *B* = –23210;

в) *A* = 24010, *B* = –30110; е) *A* = 26810, *B* = –32910.

###### Контрольные вопросы

1. Как выполняется операция вычитания в дополнительном и обратном кодах?
2. Как формируется запись положительного алгебраического числа в об- ратном и дополнительном кодах?
3. Как формируется запись отрицательного алгебраического числа в об- ратном и дополнительном кодах?
4. Как учитывается переполнение знакового поля при сложении в обрат- ном коде?
5. Как учитывается переполнение знакового поля при сложении в допол- нительном коде?
6. Чем отличаются модифицированные коды от базовых и в чем их пре- имущество?
7. Как определяется переполнение модульного поля в модифицированных кодах?
8. Какой разряд знакового поля является знакоопределяющим в модифи- цированных кодах?

##### ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №4 АРИФМЕТИКА С ПЛАВАЮЩЕЙ ТОЧКОЙ

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

###### Краткие теоретические сведения

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

При представлении числа в форме с плавающей точкой это число в об- щем случае представляет собой смешанную дробь и имеет формат, представ- ленный на рисунке 4.1.

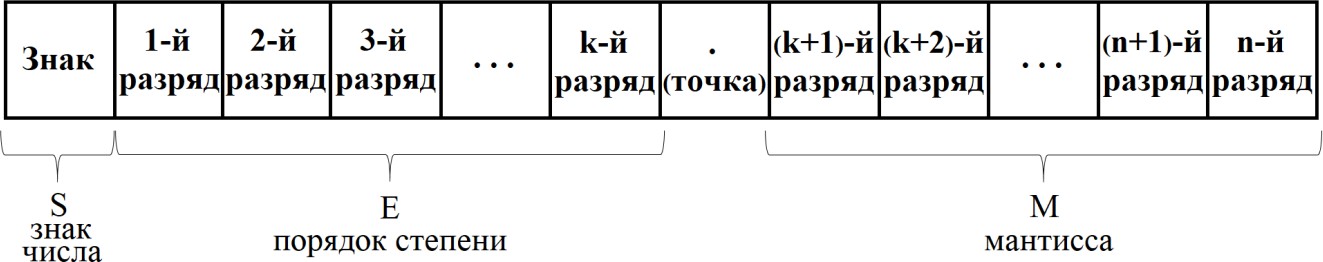


Рисунок 4.1 – Формат представления числа в форме с плавающей точкой

Местоположение точки в записи числа может быть различным, а т. к. са- ма 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 обра-

зом:

|  |  |
| --- | --- |
| 𝑄 = 𝑞𝐸 · 𝑀, | (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 дво- ичной системы счисления мантисса должна удовлетворять неравенству

|  |  |
| --- | --- |
| 1 > |𝑀| ≥ 0,5. | (4.2) |

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

###### Методические указания и примеры решения задач

Арифметические операции для чисел, представленных в форме с плава- ющей точкой, в общем выполняются по правилам арифметики алгебраических чисел, но с определенными дополнительными операциями, связанными с фор- мой представления чисел.

*Сложение/вычитание чисел с плавающей точкой*

Операция сложения/вычитания чисел с плавающей точкой предполагает наличие одинаковых порядков у операндов, подлежащих суммированию (вычи- танию). 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 фиксиро- ванной точкой, арифметические операции выполняются в обратном, дополни- тельном или их модифицированных кодах. Операция вычитания также заменя- ется на операцию сложения с изменением знака вычитаемого на противопо- ложный.

*Пример*

Найти разность *С* чисел *А* и *В*, представленных с плавающей точкой, если эти числа представлены в виде порядков [*AE*]ПК = 1.001 и [*BE*]ПК = 0.001 соответ- ственно и мантисс [*AM*]ПК = 1.11001 и [*BM*]ПК = 0.11100 соответственно. При вы- читании использовать модифицированный дополнительный код.

Сначала необходимо выровнять порядки. Для этого из порядка числа *A* вычитается порядок числа *B*. Вычитание также заменяется сложением, при этом *BE* заменяется на –*BE*. Находим [*AE*]ДК и [–*BE*]ДК:

[*AE*]ДК = 1.111 и [–*BE*]ДК = 1.111.

Находим разность порядков:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| + | 1. | 1 | 1 | 1 | *–* [*AE*]дк; |
| 1. | 1 | 1 | 1 | *–* [*–BE*]дк; |
|  | 1. | 1 | 1 | 0 | – разность порядков в дополнительном коде; |
|  | 1. | 0 | 1 | 0 | – разность порядков в прямом коде. |

Так как знак разности порядков отрицательный, то в качестве общего по- рядка, а следовательно, и предварительного значения порядка искомого резуль- тата *СE*\*, берется порядок второго числа (*BE*). 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 (*AM*) умножить на 2–2, т. е. выполнить арифметический сдвиг на два разряда вправо:

𝐴∗ = 𝐴𝑀 ⋅ 2−2 = 1.11001 ⋅ 2−2 = 1.00110.

𝑀

Таким образом, после выравнивания порядков операндов будем иметь

следующую форму представления операндов:

[*AM*

\*]ПК

= 1.00110, [*BM*

\*]ПК

= 1.11100.

Предварительное значение мантиссы определяется как *CM*\* = *AM*\* – *BM*\*. Определять предварительное значение мантиссы будем в модифициро-

ванном дополнительном коде, как указано в условии. Для этого находим значе- ния [*A* \*] и [–*B* \*] (второй операнд берется с отрицательным знаком, по- скольку операция вычитания в модифицированном дополнительном коде заме- няется на операцию сложения):

*M* МДК *M* МДК

[*AM*

\*]МДК

= 11.11010, [–*BM*

\*]МДК

= 11.00100.

Находим предварительное значение *CM*\*:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| + | 1 | 1. | 1 | 1 | 0 | 1 | 0 | – [*AM*]мдк; |
| 1 | 1. | 0 | 0 | 1 | 0 | 0 | – [–*BM*]мдк; |
| 1 | 1 | 0. | 1 | 1 | 1 | 1 | 0 | – [*C \**]мдк; *M* |
|  | 1 | 0. | 0 | 0 | 0 | 1 | 0 | – [*C \**]мпк.  *M* |

Из записи [*С* \*] , полученной после вычитания мантисс операндов с выравненными порядками, видно, что нормализация представления результата нарушена. Поэтому для данного примера необходимо выполнить этап устране- ния нарушения нормализации. В данном случае нарушение нормализации слева от точки, т. к. получено [*С* \*] с ненулевой целой частью (неодинаковые раз- ряды в знаковом поле использованного модифицированного дополнительного кода).

*M* МПК

*M* МПК

Поэтому необходимо выполнить этап устранения нарушения нормализа- ции. Для того чтобы привести полученную предварительную мантиссу к нор- мализованной форме, достаточно ее умножить на 2–1, т. е. выполнить ее ариф- метический сдвиг вправо. В результате будем иметь конечное значение мантиссы:

С𝑀 = С∗ ⋅ 2−1 = 1.00010 ⋅ 2−1 = 1.10001.

𝑀

Арифметический сдвиг предварительного значения мантиссы *СM*\* сопро-

вождается изменением ранее найденного предварительного значения порядка результата *СE*\*на плюс единицу. Находим окончательное значение порядка *СE*:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| + | 0 | 0. | 0 | 0 | 1 | – [*C \**]мдк; *E* |
| 0 | 0. | 0 | 0 | 1 | *–* +1; |
|  | 0 | 0. | 0 | 1 | 0 | – [*CE*]мдк = [*CE*]мпк. |

После устранения нарушения нормализации окончательный результат будет иметь вид *С*: {[*CE*]ПК = 0.010, [*CM*]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 ос- новании порядка и мантиссы множимого и множителя. Для двоичных чисел это имеет вид

|  |  |
| --- | --- |
| 𝐶 = 𝐴 ⋅ 𝐵 = 2𝐴𝐸 ⋅ 𝐴𝑀 ⋅ 2𝐵𝐸 ⋅ 𝐵𝑀 = 2𝐴𝐸+𝐵𝐸 ⋅ 𝐴𝑀 ⋅ 𝐵𝑀 = 2𝐶𝐸 ⋅ 𝐶𝑀. | (4.3) |

Из (4.3) следует, что порядок произведения определяется как сумма по- рядков сомножителей, а мантисса произведения – как произведение мантисс сомножителей. Однако, учитывая то, что при умножении мантисс может про- изойти нарушение нормализации, в результате указанных действий будет найдено предварительное значение порядка и мантиссы искомого произведения и окончательное значение произведения будет найдено только после устране- ния 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 посред- ством 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 [*AE*]ПК = 1.010 и [*BE*]ПК = 0.001 соответственно и мантисс [*AM*]ПК = 1.1010 и [*BM*]ПК = 0.1001 соот- ветственно. При выполнении операций использовать обратный код. При умно- жении 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 *CE*\* определяется как сумма порядков сомножителей *CE*\* = *AE* + *BE*:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| + | 1. | 1 | 0 | 1 | – [*AE*]ок; |
| 0. | 0 | 0 | 1 | – [*BE*]ок; |
|  | 1. | 1 | 1 | 0 | – [*C \**]ок; *E* |
|  | 1. | 0 | 0 | 1 | – [*C \**]пк.  *E* |

Абсолютное значение предварительного значения мантиссы произведе- ния *CM*\* определяется как произведение модулей мантисс сомножителей:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | × | 0. | 1 | 0 | 1 | 0 | *– |*[*AM*]*|*ок; |
| 0. | 1 | 0 | 0 | 1 | *– |*[*BM*]*|*ок; |
|  |  |  | + |  | 0 | 1 | 0 | 1 | 0 | – 1-е частичное произведение; |
| 0 | 0 | 0 | 0 | 0 |  | – 2-е частичное произведение; |
|  |  | + |  |
| 0 | 0 | 0 | 0 | 0 |  |  | – 3-е частичное произведение; |
|  | + |  |
| 0 | 1 | 0 | 1 | 0 |  |  |  | – 4-е частичное произведение; |
| + |  |
| 0 | 0 | 0 | 0 | 0 |  |  |  |  | – 5-е частичное произведение; |
|  | 0. | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | *\* \**  *– |*[*CM* ]*|*ок = *|*[*CM* ]*|*пк. |

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

Таким образом, с учетом определенного знака, нормализации и округле- ния до четырех разрядов конечное значение мантиссы [*CM*]ПК = 1.1011.

С учетом нормализации конечное значение порядка [*CE*]ПК = 1.010. Конечное значение произведения *С*: {[*CE*]ПК = 1.010, [*CM*]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 = (2𝐴𝐸 ⋅ 𝐴𝑀) ∶ (2𝐵𝐸 ⋅ 𝐵𝑀) = 2𝐴𝐸−𝐵𝐸 ∶ (𝐴𝑀 ⋅ 𝐵𝑀) = 2𝐶𝐸 ⋅ 𝐶𝑀. | (4.4) |

Из (4.4) следует, что порядок частного определяется как разность поряд- ков делимого и делителя, а мантисса – как частное от деления мантиссы дели- мого на мантиссу делителя. Однако, учитывая то, что при делении мантисс мо- жет произойти нарушение нормализации, в результате указанных действий бу- дет 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, пред- ставленных 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 наруше- ние имеет место) соответствующей корректировкой предварительного значения порядка и мантиссы искомого частного;
* при формировании мантиссы частного нормализованных чисел с плава- ющей точкой возможен только один вид нарушения нормализации – нарушение нормализации слева от точки.

Деление мантисс выполняется по правилам деления чисел с фиксирован- ной точкой с восстановлением остатка или без восстановления остатка.

*Деление с восстановлением остатка* выполняется потактно за (*n* + 2) так- та (*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 дели- мого), вычитается делитель (выполняется пробное вычитание) – тем самым формируется новый остаток;
  + анализируется знак нового остатка. Если знак отрицательный, то осу- ществляется восстановление остатка, т. е. к полученному новому остатку при- бавляется 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 вы- рабатывается специальный сигнал о том, что искомое частное не является пра- вильной 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 дели- мого). Если знак положительный, то из остатка вычитается делитель, в против- ном случае делитель прибавляется. Таким образом формируется новый остаток;
  + если новый остаток положительный, то в очередном разряде формируе- мого 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 нару- шении 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 [*AE*]ПК = 1.010 и [*BE*]ПК = 0.001 со- ответственно и мантисс [*AM*]ПК = 1.1010 и [*BM*]ПК = 0.1001 соответственно. При выполнении операций использовать модифицированный обратный код. При де- лении мантисс использовать метод деления без восстановления остатка.

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

Предварительное значение порядка частного *CE*\* определяется как раз- ность порядков делимого и делителя *CE*\* = *AE* – *BE*. Операция вычитания заме- няется на операцию сложения с изменением знака вычитаемого на противопо- ложный. Таким образом:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | + | 1 | 1. | 1 | 0 | 1 | – [*AE*]мок; |
| 1 | 1. | 1 | 1 | 0 | – [–*BE*]мок; |
| + | ~~1~~ | 1 | 1. | 1 | 1 | 0 | – корректировка единицей в младший разряд при переполнении разрядности знакового поля; |
|  |  |  |  |  | 1 |
|  |  | 1 | 1. | 1 | 0 | 0 | – [*С \**]мок; *E* |
|  |  | 1 | 1. | 0 | 1 | 1 | – [*С \**]мпк.  *E* |

Абсолютное значение предварительного значения мантиссы частного *CM*\*

определяется за шесть тактов деления:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | + | 0 | 0. | 1 | 0 | 1 | 0 | – [|*AM|*]мок; |
| 1 | 1. | 0 | 1 | 1 | 0 | – [–*|BM|*]мок; |
| + | ~~1~~ | 0 | 0. | 0 | 0 | 0 | 0 | – корректировка единицей в младший разряд при переполнении разрядности знакового поля; |
|  |  |  |  |  |  | 1 |
|  |  | 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 | – [–*|BM|*]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 | – [*|BM|*]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 | – [*|BM|*]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 | – [*|BM|*]мок; |
| + | ~~1~~ | 0 | 0. | 0 | 1 | 1 | 0 | – корректировка единицей в младший разряд при переполнении разрядности знакового поля; |
|  |  |  |  |  |  | 1 |
|  |  | 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 | – [–*|BM|*]мок; |
| + | ~~1~~ | 0 | 0 | 0 | 1 | 0 | 0 | *–* корректировка единицей в младший разряд при переполнении разрядности знакового поля; |
|  |  |  |  |  |  | 1 |
|  |  | 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

|*CM*\*| = 1,00011, а с учетом округления – |*CM*\*| = 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*: {[*CE*]ПК = 1.010, [*CM*]ПК = 1.1001}.

###### Задачи для решения

Задание 1

Найти сумму *С* чисел *А* и *В*, представленных с плавающей точкой, если эти числа представлены в виде порядков *AE* и *BE* и мантисс *AM* и *BM* соответ- ственно. При сложении использовать модифицированный дополнительный код:

а) [*AE*]ПК = 1.010, [*BE*]ПК = 0.001, [*AM*]ПК = 1.1001, [*BM*]ПК = 0.0111;

б) [*AE*]ПК = 0.001, [*BE*]ПК = 1.010, [*AM*]ПК = 0.1101, [*BM*]ПК = 1.1010.

Задание 2

Найти произведение *С* чисел *А* и *В*, представленных с плавающей точкой, если эти числа представлены в виде порядков *AE* и *BE* и мантисс *AM* и *BM* соот- ветственно. При умножении использовать модифицированный обратный код:

а) [*AE*]ПК = 1.001, [*BE*]ПК = 0.010, [*AM*]ПК = 1.0101, [*BM*]ПК = 0.1010;

б) [*AE*]ПК = 0.010, [*BE*]ПК = 1.001, [*AM*]ПК = 0.1001, [*BM*]ПК = 1.0111.

Задание 3

Найти частное *С* чисел *А* и *В*, представленных с плавающей точкой, если эти числа представлены в виде порядков *AE* и *BE* и мантисс *AM* и *BM* соответ- ственно. При делении использовать модифицированный обратный код:

а) [*AE*]ПК = 1.010, [*BE*]ПК = 0.001, [*AM*]ПК = 1.1100, [*BM*]ПК = 0.0010;

б) [*AE*]ПК = 0.100, [*BE*]ПК = 1.010, [*AM*]ПК = 0.0101, [*BM*]ПК = 1.0011.

###### 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 логиче- ских схем в заданном булевом базисе.

###### 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 высо- кому сигналу на выходе компоненты.

Основные понятия алгебры логики:

* + - *логическая переменная* – это такая переменная, которая может прини- мать одно из двух значений: истинно или ложно (да или нет, единица или нуль);
    - *логическая константа* – это такая постоянная величина, значением ко- торой 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 зависи- мости 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 логи- ческого выражения.

*Таблица истинности* является универсальным средством задания логиче- ской функции. Она включает все наборы для заданного количества перемен- ных, определяющих значение логической функции, с указанием значений, ко- торые 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 не- сложной логической функции.

*Логическим выражением* называется комбинация логических переменных и констант, связанных элементарными базовыми логическими функциями (или логическими операциями), которые могут разделяться скобками. Например, ло- гическую функцию *у*1 можно представить в виде логического выражения

|  |  |
| --- | --- |
| 𝑦1 = (̅̅𝑥̅̅̅∙̅̅𝑥̅̅̅̅+̅̅̅𝑥̅̅̅∙̅̅𝑥̅̅̅+̅̅̅𝑥̅̅̅̅∙̅̅𝑥̅̅̅) ∙ (𝑥 + 𝑥 + 𝑥 ) + 𝑥 ∙ 𝑥 ∙ 𝑥 ,  1 2 1 3 2 3 1 2 3 1 2 3 | (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 назы- вают *базисом*. В качестве элементарных логических функций функционально полных систем логических функций используются функции одной или двух ло- гических 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

|  |  |
| --- | --- |
| 𝑦0 = 0, | (5.2) |

где значение *y*0 является константой; где *y*1 равняется значению переменной;

|  |  |
| --- | --- |
| 𝑦1 = 𝑥, | (5.3) |

|  |  |
| --- | --- |
| 𝑦2 = 𝑥̅, | (5.4) |

где *y*2 – равняется инверсному значению переменной;

|  |  |
| --- | --- |
| 𝑦3 = 1, | (5.5) |

где *y*3 – константа.

Все возможные варианты функций двух переменных приведены в табли- це 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 *систе- мой 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

|  |  |  |  |
| --- | --- | --- | --- |
| *yi* | 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 | 𝑥1 ∙ 𝑥2 (𝑥1𝑥2, 𝑥1 ∨ 𝑥2) |
| *y*2 | Запрет по *x*2 | Неверно, что если *x*1, то *x*2 | 𝑥1𝑥̅̅2̅ |
| *y*3 | *f*(*x*1) | Функция одной переменной | *x*1 |
| *y*4 | Запрет по *x*1 | Неверно, что если *x*2, то *x*1 | ̅𝑥̅1̅𝑥2 |
| *y*5 | *f*(*x*2) | Функция одной переменной | *x*2 |
| *y*6 | Функция  неравнозначности | *x*1 не равно *x*2 | ̅𝑥̅1̅𝑥2 + 𝑥1̅𝑥̅2̅ |
| *y*7 | Дизъюнкция | Или *x*1, или *x*2 | 𝑥1 + 𝑥2 |
| *y*8 | Функция Пирса | Ни *x*1, ни *x*2 | ̅𝑥̅1̅̅+̅̅̅̅𝑥̅2̅ |
| *y*9 | Функция  равнозначности | *x*1 равно *x*2 | ̅𝑥̅1̅𝑥̅̅2̅ + 𝑥1𝑥2 |
| *y*10 | *f*(*x*2) | Функция одной переменной | ̅𝑥̅2̅ |
| *y*11 | Импликация | Если *x*2, то *x*1 | ̅𝑥̅1̅̅𝑥̅2̅ + 𝑥1 |
| *y*12 | *f*(*x*1) | Функция одной переменной | ̅𝑥̅1̅ |
| *y*13 | Импликация | Если *x*1, то *x*2 | ̅𝑥̅1̅̅𝑥̅2̅ + 𝑥2 |
| *y*14 | Функция Шеффера | Неверно, что и *x*1, и *x*2 | ̅𝑥̅1̅̅𝑥̅2̅ |
| *y*15 | Константа 1 | Тождественная единица | 1 |

При работе с булевыми логическими выражениями используются следу- ющие законы и правила:

* + - переместительный (коммутативный) закон:

𝑥1𝑥2 = 𝑥2𝑥1;

𝑥1 + 𝑥2 = 𝑥2 + 𝑥1;

* + - сочетательный (ассоциативный) закон:

(𝑥1𝑥2)𝑥3 = 𝑥1(𝑥2𝑥3);

𝑥1 + (𝑥2 + 𝑥3) = (𝑥1 + 𝑥2) + 𝑥3;

* + - распределительный (дистрибутивный) закон:

𝑥1(𝑥2 + 𝑥3) = 𝑥1𝑥2 + 𝑥1𝑥3;

* + - правило де Моргана:
    - операция склеивания:

𝑥̅̅1̅̅𝑥̅2̅ = ̅𝑥̅1̅ + 𝑥̅̅2̅;

̅𝑥̅1̅̅+̅̅̅̅𝑥̅2̅ = 𝑥̅1𝑥̅2;

𝑥𝑖𝐴 + 𝑥̅𝜄𝐴 = 𝐴,

(𝑥𝑖 + 𝐵)(𝑥̅𝜄 + 𝐵) = 𝐵,

где *A* – логическое произведение переменных и их отрицаний;

*B* – логическая сумма переменных и их отрицаний;

* + - операции с отрицаниями:

𝑥̅̅ = 𝑥;

𝑥̅ ∙ 𝑥 = 0;

𝑥̅ + 𝑥 = 1;

- операции с константами:

𝑥 + 1 = 1;

𝑥 ∙ 1 = 𝑥;

𝑥 + 0 = 𝑥;

𝑥 ⋅ 0 = 0;

- операции с одинаковыми операндами:

𝑥 + 𝑥 + ⋯ + 𝑥 = 𝑥;

𝑥 ∙ 𝑥 ∙ … ∙ 𝑥 = 𝑥.

###### Методические указания и примеры решения задач

Логические схемы строятся на основе логических элементов, набор кото- рых определяется заданным логическим базисом.

Для базиса Буля в качестве логических элементов используются элемен- ты, 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 соответ- ствующих графических обозначений логических элементов, а связи между ни- ми определяются последовательностью выполнения логических операций в за- данном выражении.

## И ИЛИ

*x*1

&

*x*2

*y = x*1*x*2  *x*1

*x*2

1

*y = x*1*+x*2

## НЕ НЕ



*y*

&

*x*1



*y*

1

*= x*1

*x*1

*= x*1

Рисунок 5.1 – Графические обозначения логических элементов

*Пример*

Синтезировать логическую схему в базисе И, ИЛИ, НЕ, реализующую ло- гическое выражение

|  |  |
| --- | --- |
| 𝑦1 = (̅̅𝑥̅̅̅𝑥̅̅̅+̅̅̅𝑥̅̅̅̅) ∙ (𝑥 + 𝑥 𝑥 ) + 𝑥 ̅𝑥̅̅.  1 2 3 1 2 3 1 3 | (5.6) |

Входными сигналами синтезируемой схемы являются *x*1, *x*2, *x*3, а выход- ным – *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

𝑦1 = 𝑡1,1 + 𝑡1,2,

|  |  |  |
| --- | --- | --- |
| где | 𝑡1,1 = (̅̅𝑥̅̅̅𝑥̅̅̅+̅̅̅𝑥̅̅̅̅) ∙ (𝑥 + 𝑥 𝑥 );  1 2 3 1 2 3 | (5.7) |
|  | 𝑡1,2 = 𝑥1̅𝑥̅3̅. | (5.8) |

Для реализация этой операции потребуется элемент ИЛИ с двумя входа- ми, 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).

*t*1,1 *t*1,2

1

*y1 = t*1,1*+t*1,2

Рисунок 5.2 – Результат синтеза логического выражения на первом шаге

Шаг 2. На данном этапе рассмотрим операцию для формирования сигна- лов *t*1,1 и *t*1,2. Выражение (5.7) можно переписать следующим образом:

|  |  |  |
| --- | --- | --- |
|  | 𝑡1 = 𝑡2,1𝑡2,2, | (5.9) |
| где | 𝑡2,1 = 𝑥̅̅1̅̅𝑥̅2̅̅̅+̅̅̅𝑥̅3̅; | (5.10) |
|  | 𝑡2,2 = 𝑥1 + 𝑥2𝑥3. | (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).

*t*2,1 *t*2,2

&

*t*1,1 *= t*2,1*t*2,2

Рисунок 5.3 – Результат синтеза логического выражения для сигнала t1,1

Выражение (5.8) можно переписать следующим образом:

𝑡1,2 = 𝑡2,3𝑡2,4,

где 𝑡2,3 = 𝑥1;

𝑡2,4 = 𝑥̅̅3̅.

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).

*t*2,3 = *x*1

&

*t*2,4

*t*1,2 *= t*2,3*t*2,4

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.

*t*2,1 *t*2,2

1

&

*t*1,2

&

*x*1 *t*2,4

*t*1,1

*y*1

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 элемен-

та НЕ и записать в виде выражения

|  |  |  |
| --- | --- | --- |
|  | 𝑡2,1 = 𝑡̅̅3̅,1̅, | (5.12) |
| где | 𝑡3,1 = ̅𝑥̅1̅̅𝑥̅2̅ + 𝑥3. | (5.13) |

Сигнал *t*2,2 можно записать как

|  |  |  |
| --- | --- | --- |
|  | 𝑡2,2 = 𝑡3,2 + 𝑡3,3, | (5.14) |
| где | 𝑡3,2 = 𝑥1; | (5.15) |
|  | 𝑡3,3 = 𝑥2𝑥3. | (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

|  |  |  |
| --- | --- | --- |
|  | 𝑡3,1 = 𝑡4,1 + 𝑡4,2, | (5.17) |
| где | 𝑡4,1 = 𝑥̅̅1̅̅𝑥̅2̅; | (5.18) |
|  | 𝑡4,2 = 𝑥3. | (5.19) |

В результате синтезирования сигнал *t*3,1 будет получен на выходе элемен- та 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:

|  |  |
| --- | --- |
| 𝑡4,1 = 𝑡̅5. | (5.20) |

Сигнал *t*5 формируется на выходе элемента И, на входы которого подают- ся сигналы *x*1 и *x*2. В конечном результате функция *y*1 примет вид, представлен- ный на рисунке 5.6.

*x*1 *x*2



*t*5

*t*4,1

*t*4,2*=x*3

1

1

&

*x*2 *x*3



&

1

1

&

*t*1,2

&

*t*2,4

1

*t*3,1

*t*3,2=*x*1 *t*3,3

1

*x*3

*t*2,1

*t*2,2

*x*1

*t*1,1

*y*1

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.

*x*1 *x*2 *x*3

*t*1,1



*t*2,2

*t*2,4

*t*5

1

1

1

&

1

*t*3,2

*t*3,3

*t*1,2

&

1

*t*4,1 *t*4,2

*t*3,1

&

1

*t*2,1

&

*y*1

Рисунок 5.7 – Результат синтеза логического выражения

Для решения задачи синтеза в базисе И-НЕ необходимо использовать ба- зовый 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:

|  |  |
| --- | --- |
| 𝑥1 + 𝑥2 = 𝑥̅̅̅̅+̅̅̅̅𝑥̅̅ = 𝑥̅̅̅1̅̅𝑥̅̅2̅;  1 2 | (5.21) |
| 𝑥1 ∙ 𝑥2 = 𝑥̅̅1̅̅𝑥̅2̅. | (5.22) |

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

### И

*x*1 *y = x*1*x*2



&

&

*x*2

### НЕ



*x*1

&

*x*1 &

*x*2



&

**ИЛИ**

*y = x*1*+x*2

&

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

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

**ИЛИ-НЕ**



1

Рисунок 5.10 – Схемная реализация базового элемента ИЛИ-НЕ

Логическая операция И в базисе ИЛИ-НЕ примет вид

𝑥1 ∙ 𝑥2 = 𝑥̅̅1̅̅∙̅̅𝑥̅2̅ = 𝑥̅̅̅1̅̅+̅̅̅̅𝑥̅̅2̅.

Логическая операция ИЛИ реализуется следующим образом:

𝑥1 + 𝑥2 = 𝑥̅̅̅̅+̅̅̅̅𝑥̅̅.

1

2

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

# ИЛИ

*x* 1



1

*x*2

# НЕ

*y = x*1*+x*2 **И**

*x*1



1

1

*y = x*1*+x*2

1

*= x*1 *x*2



*x*1

*y*

1

1

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 = (̅̅𝑥̅̅̅𝑥̅̅̅+̅̅̅𝑥̅̅̅̅) ∙ (𝑥 + 𝑥 𝑥 ) + 𝑥 ̅𝑥̅̅ = ̅̅𝑥̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅  1 1 2 3 1 2 3 1 3 (̅̅1̅̅𝑥̅2̅ ∙ 𝑥̅3̅) ∙ (̅̅𝑥̅̅̅̅∙̅𝑥̅̅̅̅̅̅𝑥̅̅̅̅̅̅) ∙ 𝑥̅1̅̅𝑥̅3̅.  1 2 3 | (5.23) |

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

*x*1 *x*2 *x*3



&

&

&

&

&

&

&

&

&

*y*1

&

&

&

*Пример*

Рисунок 5.12 – Результат синтеза выражения в базисе И-НЕ

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

|  |  |
| --- | --- |
| ̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅̅  𝑦1 = (̅̅̅𝑥̅̅̅𝑥̅̅̅̅+̅̅̅𝑥̅̅̅) ∙ (𝑥 + 𝑥 𝑥 ) + 𝑥 ̅𝑥̅̅ = (̅̅𝑥̅̅̅̅̅̅̅+̅̅̅̅𝑥̅̅̅+̅̅̅𝑥̅̅̅)̅ + (̅̅𝑥̅̅̅̅+̅̅̅𝑥̅̅̅̅+̅̅̅̅𝑥̅̅̅̅̅̅) + 𝑥̅̅̅̅+̅̅̅̅𝑥̅̅.  1 2 3 1 2 3 1 3 1 2 3 1 2 3 1 3 | (5.24) |

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

*x*1 *x*2 *x*3



1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

*y*1

Рисунок 5.13 – Результат синтеза выражения в базисе ИЛИ-НЕ

###### Задачи для решения

Задание 1

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

Варианты задач:

1) 𝑦1 = 𝑥1 + 𝑥2 + ̅𝑥̅̅1̅̅𝑥̅3̅̅̅+̅̅̅𝑥̅2̅;

2) 𝑦2 = ̅𝑥̅1̅̅𝑥̅̅̅2̅̅̅+̅̅̅𝑥̅1̅̅𝑥̅3̅ + 𝑥3;

3) 𝑦3 = ̅𝑥̅1̅̅𝑥̅̅2̅𝑥3 + ̅𝑥̅1̅̅+̅̅̅̅𝑥̅3̅;

1. 𝑦4 = 𝑥1̅𝑥̅2̅+(𝑥4 + 𝑥1)̅𝑥̅3̅;
2. 𝑦5 = ̅𝑥̅̅̅̅(̅̅𝑥̅̅̅+̅̅̅𝑥̅̅̅̅)(̅𝑥̅̅ + 𝑥 );

3

1

2

2

3

1. 𝑦6 = (̅̅𝑥̅̅̅̅+̅̅̅𝑥̅̅̅) + ̅𝑥̅̅̅𝑥̅̅̅𝑥 ;

3

1

1

2

3

1. 𝑦7 = 𝑥2𝑥̅3 + (𝑥1 + 𝑥2)𝑥4;
2. 𝑦8 = 𝑥4 + 𝑥3 + 𝑥̅̅̅̅(̅𝑥̅̅̅̅+̅̅̅𝑥̅̅̅);

2

4

1

1. 𝑦9 = ̅𝑥̅2̅̅+̅̅̅̅𝑥̅̅4̅̅𝑥̅̅1̅ + ̅𝑥̅3̅̅𝑥̅2̅;
2. 𝑦10 = (𝑥2̅𝑥̅3̅̅𝑥̅4̅ + 𝑥̅1) ∙ 𝑥̅4;
3. 𝑦11 = 𝑥̅̅1̅̅+̅̅̅̅𝑥̅̅2̅ ∙ (̅̅𝑥̅̅̅+̅̅̅𝑥̅̅̅ ̅)̅̅∙̅̅𝑥̅̅̅;

3

4

1

1. 𝑦12 = (̅̅𝑥̅̅̅+̅̅̅̅𝑥̅̅̅) + 𝑥 ̅(̅𝑥̅̅̅̅̅+̅̅̅𝑥̅̅̅̅);

4

1

1

2

3

1. 𝑦13 = 𝑥̅̅̅̅𝑥̅̅ ̅(̅̅𝑥̅̅̅̅+̅̅̅𝑥̅̅̅) + 𝑥 ;

4

1

2

4

3

1. 𝑦14 = 𝑥̅3 + (̅̅𝑥̅̅̅̅𝑥̅̅̅̅+̅̅̅𝑥̅̅̅̅)̅𝑥̅̅;

1

2

3

4

1. 𝑦15 = (̅̅𝑥̅̅̅+̅̅̅𝑥̅̅̅̅𝑥̅̅̅̅)𝑥

3

4

1

2

4

+ 𝑥 .

###### Контрольные вопросы

53

1. Назовите особенности синтеза логических схем по логическим выраже- ниям в базисе И-НЕ.
2. Назовите особенности синтеза логических схем по логическим элемен- там 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 Мор-

гана.

##### ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №6 МИНИМИЗАЦИЯ ЛОГИЧЕСКИХ ВЫРАЖЕНИЙ

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

###### Краткие теоретические сведения

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

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

*n* = 1

*n* = 2

*n* = 3

|  |  |  |
| --- | --- | --- |
|  | *x*1 | *x*1 |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  | *x*1 | x1 |
| *x*2 |  |  |
| *x*2 |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| *x*1*x*2 | *x*1*x*2 | *x*1*x*2 | *x*1*x*2 |
| *x*3 |  |  |  |  |
| *x*3 |  |  |  |  |

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 со- вершенной 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

|  |  |
| --- | --- |
| 𝑦1 = 𝑥̅1𝑥̅2𝑥̅3 + 𝑥̅1𝑥̅2𝑥3 + 𝑥̅1𝑥2𝑥3 + 𝑥1𝑥̅2𝑥̅3; | (6.1) |
| 𝑦2 = 𝑥̅1𝑥̅2𝑥̅3 + 𝑥̅1𝑥2𝑥̅3 + 𝑥̅1𝑥2𝑥3 + 𝑥1𝑥2𝑥̅3 + 𝑥1𝑥̅2𝑥̅3. | (6.2) |

СКНФ примет следующий вид:

|  |  |
| --- | --- |
| 𝑦1 = (𝑥1 + 𝑥̅2 + 𝑥3) ∙ (𝑥̅1 + 𝑥̅2 + 𝑥3) ∙ (𝑥̅1 + 𝑥̅2 + 𝑥̅3) ∙ (𝑥̅1 + 𝑥2 + 𝑥̅3); | (6.3) |
| 𝑦2 = (𝑥1 + 𝑥2 + 𝑥̅3) ∙ (𝑥̅1 + 𝑥̅2 + 𝑥̅3) ∙ (𝑥̅1 + 𝑥2 + 𝑥̅3). | (6.4) |

В процессе минимизации может возникнуть задача перехода между ос- новными 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.

###### 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.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *x*1*x*2 | *x*1*x*2 |
| *x*1*x*2 | | *x*1*x*2 | |
| *x*3 |  | |  | |  |  |
| *x*3 |  | |  | |  |  |

Рисунок 6.2 – Карта Карно для функции трех переменных

Рассмотрим функцию по конституентам. Первая конституента представ- ляет собой набор 𝑥̅1𝑥̅2𝑥̅3, следовательно, в ячейку соответствующую данному набору необходимо установить единицу (рисунок 6.3).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *x*1*x*2 | *x*1*x*2 |
| *x*1*x*2 | | *x*1*x*2 | |
| *x*3 | 1 | |  | |  |  |
| *x*3 |  | |  | |  |  |

Рисунок 6.3 – Запись конституенты 𝑥̅1𝑥̅2𝑥̅3 в соответствующий набор

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *x*1*x*2 | *x*1*x*2 |
| *x*1*x*2 | | *x*1*x*2 | |
| *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 выра- жение 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 уче- том вышеприведенных требований;
* контурами необходимо охватить все ячейки с единичными значениями;
* контуров должно быть минимальное количество.

*Логическими соседями* являются такие две ячейки, наборы которых отли- чаются только одной переменной: в одной эта переменная должна иметь пря- мое, в другой – обратное значение. Следует отметить, что логическими соседя- ми также являются ячейки, располагающиеся в соответствующих строках край- него левого столбца и крайнего правого столбца, а также в соответствующих столбцах крайней верхней и крайней нижней строках.

Запись минимального выражения для заданной функции формируется та- ким образом, чтобы конъюнкция, соответствующая контуру, включала только те переменные, которые имеют постоянное значение во всех клетках, охвачен- ных рассматриваемым контуром. Контуры для функции *y*1 имеют вид, пред- ставленный на рисунке 6.5.

II



*x*1*x*2 *x*1*x*2 *x*1*x*2 *x*1*x*2

*x*3

*x*3

1

1

1

1

I

III

Рисунок 6.5 – Контуры для функции *y*1, представленной в виде карт Карно

Контур I представляет собой две ячейки, соответствующие наборам 𝑥̅1𝑥̅2𝑥̅3 и 𝑥̅1𝑥̅2𝑥3. Постоянными переменными для ячеек в данном контуре являются 𝑥̅1𝑥̅2. Тогда минимальное логическое выражение для контура I примет вид

𝑥̅1𝑥̅2𝑥̅3 + 𝑥̅1𝑥̅2𝑥3 = 𝑥̅1𝑥̅2.

Минимальное логическое выражение для функции *y*1 примет вид

|  |  |
| --- | --- |
| 𝑦1 = 𝑥̅1𝑥̅2 + 𝑥̅1𝑥3 + 𝑥̅2𝑥̅3, | (6.5) |

где 𝑥̅1𝑥̅2 – минимальное логическое выражение для контура I;

𝑥̅1𝑥3 – минимальное логическое выражение для контура II;

𝑥̅2𝑥̅3 – 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 каче- стве наборов, соответствующим ячейкам карт Карно, используются простые дизъюнкции, а в соответствующие ячейки устанавливаются значения нуль:

|  |  |
| --- | --- |
| 𝑦2 = (𝑥1 + 𝑥2 + 𝑥̅3) ∙ (𝑥̅1 + 𝑥̅2 + 𝑥̅3) ∙ (𝑥̅1 + 𝑥2 + 𝑥̅3). | (6.6) |

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

#### I

*x*1*x*2 *x*1*x*2

*x*3

0

0

0

*x*3

*x*1*x*2

*x*1*x*2

II

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:

|  |  |
| --- | --- |
| 𝑦2 = (𝑥̅1 + 𝑥̅3) ∙ (𝑥2 + 𝑥̅3). | (6.7) |

*Пример*

Выполнить минимизацию функции *f*, заданной в числовой фор- ме 𝑓(𝑥1𝑥2𝑥3𝑥4) = (0,2,5,7,8,10,13,15), на единичных наборах. Записать мини- мальную 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(𝑥1𝑥2𝑥3𝑥4), заданная в виде таблицы истинности

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер набора | 𝑥1 | 𝑥2 | 𝑥3 | 𝑥4 | 𝑓(𝑥1𝑥2𝑥3𝑥4) |
| 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(𝑥1𝑥2𝑥3𝑥4) = 𝑥̅1𝑥̅2𝑥̅3𝑥̅4 + 𝑥̅1𝑥̅2𝑥3𝑥̅4 + 𝑥̅1𝑥2𝑥̅3𝑥4 +  +𝑥̅1𝑥2𝑥3𝑥4 + 𝑥1𝑥̅2𝑥̅3𝑥̅4 + 𝑥1𝑥̅2𝑥3𝑥̅4 + 𝑥1𝑥2𝑥̅3𝑥4 + 𝑥1𝑥2𝑥3𝑥4. | (6.8) |

Разместим простые конъюнкции на карте Карно (рисунок 6.7).

#### I



*x*1*x*2 *x*1*x*2 *x*1*x*2 *x*1*x*2

*x*3*x*4 *x*3*x*4 *x*3*x*4

*x*3*x*4

1

1

1

1

1

1

1

1

II

Рисунок 6.7 – Карта Карно для функции 𝑓(𝑥1𝑥2𝑥3𝑥4)

В итоге минимальная функция примет вид

|  |  |
| --- | --- |
| 𝑓(𝑥1𝑥2𝑥3𝑥4) = 𝑥̅2𝑥̅4 + 𝑥2𝑥4. | (6.9) |

В случае, когда результат необходимо записать в конъюнктивной нор- мальной форме (КНФ), необходимо либо изначально по таблице истинности записать СКНФ исходной функции, либо прибегнуть к решению задачи пере- хода 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 𝑦3 = 𝑥̅1𝑥̅2𝑥̅3 + 𝑥̅1𝑥2𝑥̅3 + 𝑥̅1𝑥2𝑥3 + 𝑥1𝑥2𝑥̅3 +

+ 𝑥1𝑥̅2𝑥̅3 найти запись этой функции в СКНФ.

Сначала необходимо записать отрицание данной функции, т. е. записать

дизъюнкцию простых конъюнкций, где простые конъюнкции представляют со- бой конституенты нуля. В свою очередь конституенты нуля – это те наборы, которые не являются наборами конституент единиц:

|  |  |
| --- | --- |
| ̅𝑦̅3̅ = 𝑥̅1𝑥̅2𝑥3 + 𝑥1𝑥2𝑥3 + 𝑥1𝑥̅2𝑥3. | (6.10) |

Здесь представлена дизъюнкция тех наборов, которые не использовались в функции *y*2. Запишем отрицание правой и левой частей выражения (6.10):

̅𝑦̅3̅ = ̅𝑥̅̅1̅𝑥̅̅̅2̅𝑥̅̅3̅̅+̅̅̅𝑥̅̅1̅𝑥̅̅2̅𝑥̅̅3̅̅+̅̅̅𝑥̅1̅̅𝑥̅̅2̅̅𝑥̅3̅.

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

правило де Моргана:

𝑦̅̅3̅ = 𝑦3.

𝑦3 = 𝑥̅̅̅1̅̅𝑥̅̅2̅̅𝑥̅3̅ ∙ 𝑥̅̅1̅̅𝑥̅2̅̅𝑥̅3̅ ∙ 𝑥̅̅1̅̅𝑥̅̅2̅̅𝑥̅3̅.

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

𝑦3 = (𝑥̅̅1 + 𝑥̅̅2 + 𝑥̅3) ∙ (𝑥̅1 + 𝑥̅2 + 𝑥̅3) ∙ (𝑥̅1 + 𝑥̅̅2 + 𝑥̅3).

Применив правило двойного отрицания, получим

|  |  |
| --- | --- |
| 𝑦3 = (𝑥1 + 𝑥2 + 𝑥̅3) ∙ (𝑥̅1 + 𝑥̅2 + 𝑥̅3) ∙ (𝑥̅1 + 𝑥2 + 𝑥̅3). | (6.11) |

Чтобы убедиться в правильности приведения формы записи функции *y*3, необходимо обратить внимание на функцию *y*2, для которой ранее были запи- саны 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.

###### Задачи для решения

Задание 1

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

а) 𝑓(𝑥1𝑥2𝑥3𝑥4) = (0,1,4,5,7,9,13,15); г) 𝑓(𝑥1𝑥2𝑥3𝑥4) = (0,1,2,4,8,9,10,14);

б) 𝑓(𝑥1𝑥2𝑥3𝑥4) = (0,1,4,5,6,7,13,15); д) 𝑓(𝑥1𝑥2𝑥3𝑥4) = (0,2,5,7,8,10,13,15);

в) 𝑓(𝑥1𝑥2𝑥3𝑥4) = (0,2,3,4,6,8,10,11); е) 𝑓(𝑥1𝑥2𝑥3𝑥4) = (0,1,8,9,10,11,14,15).

Задание 2

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

а) 𝑓(𝑥1𝑥2𝑥3𝑥4) = (2,3,6,7,9,11,13,15); г) 𝑓(𝑥1𝑥2𝑥3𝑥4) = (2,3,9,10,11,13,14,15);

б) 𝑓(𝑥1𝑥2𝑥3𝑥4) = (0,3,5,6,9,10,12,15); д) 𝑓(𝑥1𝑥2𝑥3𝑥4) = (2,3,5,7,10,11,13,15);

в) 𝑓(𝑥1𝑥2𝑥3𝑥4) = (1,3,5,7,9,11,13,15); е) 𝑓(𝑥1𝑥2𝑥3𝑥4) = (1,4,3,10,11,12,14,15).

###### 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, представлен- ных в СДНФ, с использованием карт Карно.
4. Назовите алгоритм минимизации логических выражений, представлен- ных в СКНФ, с использованием карт Карно.

##### ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №7 СИНТЕЗ ЦИФРОВЫХ АВТОМАТОВ

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

###### Краткие теоретические сведения

Цифровые автоматы определяются следующими характеристиками:

{𝐴, 𝑊, 𝑍, 𝛿, 𝜆, 𝑎𝑛},

где 𝐴 = {𝑎1, 𝑎2, … , 𝑎𝑛} – множество состояний цифрового автомата;

𝑊 = {𝑤1,𝑤2, … , 𝑤𝑛} – множество выходных сигналов цифрового автомата;

𝑍 = {𝑧1,𝑧2, … , 𝑧𝑛} – 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) = 𝜆(𝑎(𝑡), 𝑧(𝑡));

𝛿 – функция перехода цифрового автомата в новое состояние 𝑎(𝑡 + 1) в за- висимости от его текущего состояния 𝑎(𝑡) и действующего на его входе сигна- ла 𝑧(𝑡):

𝑎(𝑡 + 1) = 𝛿(𝑎(𝑡), 𝑧(𝑡));

𝑎𝑛 ∈ 𝐴 – начальное состояние цифрового автомата.

Два автомата называются *эквивалентными*, если при одинаковом множе-

стве входных и выходных сигналов и при любом начальном состоянии для лю- бого входного набора сигналов они формируют одинаковые выходные наборы сигналов. Цифровой автомат называется *конечным*, если используемые для его задания множества конечны. Цифровой автомат является *полностью опреде- ленным*, 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 со- стояние являются функцией текущего состояния и действующего на входе сиг- нала;
* автомат Мили – его выходной сигнал прямо не зависит от входного и определяется состоянием автомата. Его работа задается в виде следующих уравнений:

𝑤(𝑡 + 1) = 𝜆(𝑎(𝑡));

𝑎(𝑡 + 1) = 𝛿(𝑎(𝑡), 𝑧(𝑡)).

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 Таблица выходов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *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 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *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 𝑎𝑗 и получа- ет на вход сигнал 𝑧𝑖, то он переходит в новое состояние 𝑎(𝑡 + 1), которое запи- сано в клетке таблицы переходов, расположенной на пересечении *j*-й колонки и

1. й строки, после чего вырабатывается выходной сигнал 𝑤(𝑡 + 1), который определяется значением в клетке, расположенной в таблице выходов на пересе- чении колонки *j* и строки *i*.

Автомат Мура задается с использованием одной таблицы, которая опре- деляет 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 нахо- дится в состоянии 𝑏𝑗 и получает на вход сигнал 𝑧𝑖, то он переходит в новое со- стояние, которое записано в клетке таблицы переходов, расположенной на пе- ресечении *j*-й колонки и *i*-й строки, и вырабатывает выходной сигнал, который определяется новым состоянием цифрового автомата.

###### 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 со- стояние.

Входной сигнал (Z)

Цифровой автомат

Сигнал о состоянии (A)

Управление памятью

Выходной сигнал (W)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | |  |
| Логика цифрового автомата | |  | Память цифрового автомата | |
|  |  | | |  |

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, использо-

ванных 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 логиче- ских переменных 𝑤. Множество выходных сигналов включает три элемента, поэтому для кодирования каждой из них достаточно использовать комбинации из двух переменных.

Таблица переходов Таблица выходов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | a1 | a2 | a3 | a4 |
| z1 | a2 | a4 | a2 | a2 |
| z2 | a4 | a4 | a2 | a2 |
| z3 | a2 | a1 | a2 | a3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | a1 | a2 | a3 | a4 |
| z1 | w3 | w1 | w1 | w3 |
| z2 | w2 | w1 | w2 | w3 |
| z3 | w2 | w1 | w2 | w1 |

Рисунок 7.4 – Автомат Мили

Шаг 3. Кодирование состояний выполним через набор логических пере- менных 𝑄. Множество состояний включает четыре элемента, поэтому для ко- дирования каждого из них достаточно использовать комбинации двух перемен- ных. Полученные результаты представлены на рисунке 7.5.

Кодирование

выходных сигналов

Кодирование

входных сигналов

Кодирование

состояний

|  |  |  |
| --- | --- | --- |
|  | *y*1 | *y*2 |
| *w*1 | 0 | 1 |
| *w*2 | 1 | 0 |
| *w*3 | 1 | 1 |

|  |  |  |
| --- | --- | --- |
|  | *x*1 | *x*2 |
| *z*1 | 0 | 1 |
| *z*2 | 1 | 0 |
| *z*3 | 1 | 1 |

|  |  |  |
| --- | --- | --- |
|  | *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, пред- ставленный на рисунке 7.6.

#### Таблица переходов Таблица выходов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *Q*1*Q*2 | *Q*1*Q*2 | *Q*1*Q*2 | *Q*1*Q*2 |
| *x*1*x*2 | 10 | 00 | 10 | 10 |
| *x*1*x*2 | 00 | 00 | 10 | 10 |
| *x*1*x*2 | 10 | 01 | 10 | 11 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *Q*1*Q*2 | *Q*1*Q*2 | *Q*1*Q*2 | *Q*1*Q*2 |
| *x*1*x*2 | 11 | 01 | 01 | 11 |
| *x*1*x*2 | 10 | 01 | 10 | 11 |
| *x*1*x*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 табли- це заданы переходы и в качестве переменных выступают 𝑄1 и 𝑄2, во второй таблице в качестве переменных используются 𝑦1 и 𝑦2, которые кодируют вы- ходные сигналы.

Шаг 5. Реализацию памяти выполним на двух *Т*-триггерах, формирующих парафазные выходные сигналы 𝑄1, 𝑄̅1, используемые для кодировки состояний. Шаг 6. Составим логические выражения для логических функций, ис-

пользованных для кодировки выходных сигналов:

|  |  |
| --- | --- |
| 𝑦1 = 𝑄̅1𝑄2𝑥̅1𝑥2 + 𝑄̅1𝑄2𝑥1𝑥̅2 + 𝑄̅1𝑄2𝑥1𝑥2 +  +𝑄1𝑄2𝑥1𝑥̅2 + 𝑄1𝑄2𝑥1𝑥2 + 𝑄̅1𝑄̅2𝑥̅1𝑥2 + 𝑄̅1𝑄̅2𝑥1𝑥2; | (7.1) |
| 𝑦2 = 𝑄̅1𝑄̅2𝑥̅1𝑥2 + 𝑄̅1𝑄̅2𝑥1𝑥̅2 + 𝑄̅1𝑄̅2𝑥1𝑥2 +  +𝑄̅1𝑄2𝑥̅1𝑥2 + 𝑄1𝑄̅2𝑥̅1𝑥2 + 𝑄1𝑄̅2𝑥1𝑥̅2 +  +𝑄1𝑄̅2𝑥1𝑥2 + 𝑄1𝑄2𝑥̅1𝑥2. | (7.2) |

Приведенные выражения составляются следующим образом. При форми- ровании сигнала 𝑦1в кодированной таблице выходных сигналов (рисунок 7.6) выбираются все случаи, когда 𝑦1 имеет единичное значение, и для каждого из них формируется конъюнкция, отражающая начальное состояние и входные сигналы. Например, первая конъюнкция 𝑄̅1𝑄2𝑥̅1𝑥2 соответствует случаю, когда

текущее состояние автомата имеет значение 𝑄̅1𝑄2 и на вход поступает сигнал

𝑥̅1𝑥2. В клетке, соответствующей данным значениям, записан код 11, первый разряд данного кода соответствует переменной 𝑦1 (рисунок 7.7).

Таблица выходов

Разряд соответствующий переменной y1

Код состояния соответствующий переменной y1 = 1

Код входного сигнала соответствующий переменной y1 = 1



*x*1*x*2

11

10

01

10

*x*1*x*2

10

01

10

11

01

01

11

*x*1*x*2

*Q*1*Q*2

*Q*1*Q*2

*Q*1*Q*2

*Q*1*Q*2

01

Рисунок 7.7 – Формирование логических выражений для выходных сигналов на примере первой конъюнкции сигнала 𝑦1

Выражение для 𝑦2 формируется аналогично, но рассматриваются значе- ния второго разряда двухразрядного кода, соответствующего выходному сигна- лу цифрового автомата.

Шаг 7. Составим логические выражения для логических функций, ис- пользованных для кодировки выходных сигналов:

|  |  |
| --- | --- |
| 𝑞𝑇1 = 𝑄̅1𝑄2𝑥̅1𝑥2 + 𝑄̅1𝑄2𝑥1𝑥2 + 𝑄1𝑄̅2𝑥̅1𝑥2 + 𝑄1𝑄̅2𝑥1𝑥̅2 +  +𝑄1𝑄̅2𝑥1𝑥2 + 𝑄̅1𝑄̅2𝑥̅1𝑥2 + 𝑄̅1𝑄̅2𝑥1𝑥̅2 + 𝑄̅1𝑄̅2𝑥1𝑥2; | (7.3) |
| 𝑞𝑇2 = 𝑄̅1𝑄̅2𝑥1𝑥2 + 𝑄̅1𝑄2𝑥̅1𝑥2 + 𝑄̅1𝑄2𝑥1𝑥̅2 + 𝑄̅1𝑄2𝑥1𝑥2 +  +𝑄1𝑄̅2𝑥1𝑥2 + 𝑄1𝑄2𝑥1𝑥̅2 + 𝑄1𝑄2𝑥1𝑥2. | (7.4) |

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

по каждому входному сигналу меняет свое состояние на противоположное, то сигнал на вход *Т*-триггера надо подавать только в тех случаях, когда его состо- яние отличается от текущего. Таким образом, в выражении (7.3) первого разря- да сигнала *Т*-входа используются конъюнкции, соответствующие случаям, ко- гда значение первого разряда кода нового состояния противоположно его начальному состоянию. Например, первая конъюнкция в выражении для 𝑞𝑇1

имеет вид 𝑄̅1𝑄2𝑥̅1𝑥2 и соответствует случаю, когда начальное состояние имеет

значение 𝑄̅1𝑄2, или 01, на вход поступает сигнал 𝑥̅1𝑥2, а код нового состояния

𝑄1𝑄̅2, 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 *qT*1 *Q*1(t+1) = 1, значение изменилось на

противоположное

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *Q*1*Q*2 | *Q*1*Q*2 | *Q*1*Q*2 | *Q*1*Q*2 |
| *x*1*x*2 | 10 | 00 | 10 | 10 |
| *x*1*x*2 | 00 | 00 | 10 | 10 |
| *x*1*x*2 | 10 | 01 | 10 | 11 |

Разряд начального состояния автомата

cоответствующего коду *qT*1 *Q*1(*t*)=*Q*1 (0)





Код входного сигнала соответствующий переменной qT1

Разряд нового состояния, соответствующего коду 𝑞𝑇1

для входного сигнала 𝑥1𝑥̅2 𝑄1(𝑡 + 1) = 0 , значение не изменилось, в выражение **НЕ** записываем

Рисунок 7.8 – Формирование логических выражений для сигналов перехода на примере первой конъюнкции сигнала 𝑞𝑇1

Выражение для 𝑞𝑇2 формируется аналогично, но рассматриваются изме- нения значения второго разряда двухразрядного кода состояния после учета воздействия входного сигнала.

Шаг 8. Прежде чем перейти к задаче синтеза полученных логических вы- ражений, 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:

|  |  |
| --- | --- |
| 𝑦1 = 𝑄̅1𝑥2 + 𝑄2𝑥1; | (7.5) |
| 𝑦2 = 𝑥̅1𝑥2 + 𝑄̅2𝑥1; | (7.6) |
| 𝑞𝑇1 = 𝑄̅1𝑥2 + 𝑄̅2𝑥1 + 𝑄̅2𝑥2; | (7.7) |
| 𝑞𝑇2 = 𝑄2𝑥2 + 𝑄2𝑥1 + 𝑥1𝑥2. | (7.8) |

***y*1 *y*2**

*Q*1*Q*2 *Q*1*Q*2 *Q*1*Q*2 *Q*1*Q*2

*x*1*x*2

*x*1*x*2 1 1

*x*1*x*2 1 1 1

*x*1*x*2

1 1

*Q*1*Q*2 *Q*1*Q*2 *Q*1*Q*2 *Q*1*Q*2

*x*1*x*2

*x*1*x*2 1 1 1 1

*x*1*x*2 1

1

*x*1*x*2 1

1

***qT*1 *qT*2**

*Q*1*Q*2 *Q*1*Q*2 *Q*1*Q*2 *Q*1*Q*2

*x*1*x*2

*x*1*x*2 1 1

1

*x*1*x*2 1 1

1

*x*1*x*2 1

1

*Q*1*Q*2 *Q*1*Q*2 *Q*1*Q*2 *Q*1*Q*2

*x*1*x*2

*x*1*x*2

1 1

*x*1*x*2 1 1 1 1

*x*1*x*2

1 1

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.

*Q*1 *Q*2 *x*1 *x*2 *Q*1 *Q*2 *x*1 *x*2

1

*z*1

*z*2

*z*3

СИ

*x*1

1

*x*

2

1

1

*y*1

*y*2

1

DC 1 *w*

2

*w*

2

C

3 *w*

4 *w*

&

1

T TT1

C

1

&

1

*q*T1

&

1

1 *q*T2

&

T TT2

C

&

C

1

2

C

1

2

3

4

&

&

2

3

4

Рисунок 7.10 – Результат синтеза цифрового автомата минимизированных выражений

Шаг 9. Выходные сигналы формируются на основе кодированных с ис- пользованием декодера *DC*.

###### Задачи для решения

Задание 1

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *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 |

*а 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 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *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 |

*б 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 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *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 |

*в e*

*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

###### 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 мик- ропрограммы для заданной граф-схемы алгоритма.

###### Краткие теоретические сведения

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

Исходные данные

…...



Блок регистров

УС

УС

БУ

Операционный блок

Схемы контроля

УС

Результат

Ошибки

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 ре- зультат.

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

###### Методические указания и примеры решения задач

Блок управления АЛУ может строиться на принципе микропрограммиро- вания (программируемая логика) или на принципе с жесткой логикой (аппарат- ный принцип). В любом случае удобной формой задания поведения управляе- мого объекта является кодированная ГСА. При использовании микропрограм- много принципа выработка необходимой последовательности сигналов управ- ления 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, по- строенный в соответствии с заданной ГСА, вырабатывает последовательность выходных сигналов, используемых для управления объектом.

При микропрограммном принципе построения блока управления алго- ритм 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 мик- рокоманда должна обеспечить выработку сигналов соответствующих микро- операций 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, определяю- щую следующую микрокоманду при возможном ветвлении (по крайней мере, по двум направлениям) для продолжения вычислительного процесса.

Задание всей перечисленной информации в едином формате микроко- манды 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 выпол-

нению микрооперацию. Если допустимая длина микрокоманды достаточно ве- лика, 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 по- сле адреса расположения текущей выполняемой микрокоманды.

Допустимая длина микрокоманды

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |

*Т Y Y*

Рисунок 8.2 – Формат операционной микрокоманды

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

* типа микрокоманды (*Т*);
* условия (*Х*);
* адреса (*А*);
* модификатора дисциплины перехода (*М*).

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 об- щим количеством условий.

Поле адреса используется, чтобы задать местоположение в памяти адре- сов первых микрокоманд двух возможных ветвей продолжения процесса. В ка- честве начальной микрокоманды одной ветви продолжения используется мик- рокоманда, расположенная по адресу, следующему в памяти (*АC*) за адресом те- кущей выполняемой микрокомандой (*АT*), а адрес начальной микрокоманды другой ветви задается в самой микрокоманде (*А*).

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 ... *ym*}, признаки перехода (условия) – *X* = {*x*1, *x*2 … *xn*};
* регистр адреса для задания адреса микрокоманды. Начальный адрес (*АH*) формируется посредствам ФНА (формирователя начального адреса);
* дешифраторы кода микрооперации и кода условия;
* синхроимпульсы (СИ), по которым выполняются условия.

Приведенная схема устройства микропрограммного управления соответ- ствует 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 исполь- зуется поле *М* (поле модификатора дисциплины перехода).



Управляемый объект

*y*1 *y*2 *y*3 *... ym*

*x*1 *x*2 *x*3

*...*

*xn*

1

2

3

.

.

.

*k*

*C*

DC

1

2

3

.

.

.

.

*m*

1

2

3

.

.

.

*p*

*C*

DC

1

2

3

.

.

.

.

*n*

& 1 &

...

&

&

&

&

1

СИ

T Y X

A

&

0 1

*p p*+1 *k*-1 *k*

T=0 X

A

&

*k*-1 *k*

T=1

Y

Регистр микрокоманды

&

РПАМК, если *xi*=0, то разрешить передачу адреса

Регистр адреса

A, если *xi*=0

Ан ФНА

Ат+1, если *xi*=1, либо тип команды T=1

Запоминающее устройство

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 2 | 3 |

*Пример*

Рисунок 8.3 – Структурная схема устройства управления

Составить микропрограмму для реализации ГСА (рисунок 8.4). Управля- емый объект характеризуется следующими параметрами:

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

1

*y*1

0

2

*x*1

1

0

3

*x*4

8

*y*21 *y*15 *y*23

1

9

*x*7

1

4

*y*48 *y*5 *y*66

0

10

*y*17

5

*x*15

1

0 11

0

6

*x*15

1

*y*5 *y*3

12

*y*10

7

*y*20, *yk*

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 … *y*120}, значит, для кода микрооперации потребуется ис- пользование 7 бит (27 > 120).
* оставшиеся 8 из 16 бит используются для поля кода микрооперации *Y*2 (7 бит) и поля *yk* (1 бит, *yk* = 1, если микрокоманда является последней для дан- ной микропрограммы, в противном случае *yk* = 0).

Формат микрокоманды переходов примет следующий вид:

* нулевой бит необходим , чтобы задать тип микрокоманды (*T* = 0);
* *X* = {*x*1, *x*2 … *x*15}, тогда для кодирования условий должно быть выделено 4 бита (24 > 15);
* 10 бит необходимы, чтобы задать код адреса (*А*), т. к. длина одной ячей- ки памяти *L* = 16 бит = 2 байта и таких ячеек можно задать 2 Кбайт/2 байта, равные 1024 шт. (210 = 1024);
* последний разряд необходим, чтобы задать модификатор (*М*) перехода. Если *M* = 0, то для *xi* = 0 адрес следующей выполняемой команды (*АC*) равен *АT* + 1, а для *xi* = 1 равен *А* (т. е. адресу, указанному в поле адреса). Для значения *M* = 1 условия обратные: для *xi* = 1 адрес следующей выполняемой команды (*АC*) равен *АT* + 1, а для *xi* = 0 равен *А*.

Форматы микропрограммы представлены на рисунке 8.5.

МКП

0 1 2 3 4 5

...

14 15

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 |  |  |  |  |  |  |  | 1 |

T *X* А М

МКО

0 1 ... 7 8

...

14 15

T *Y1 Y2 yk*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  |  |  |  |  |

Рисунок 8.5 – Форматы микропрограмм для рассматриваемой задачи

Формирование микропрограммы следует начинать с первой вершины ГСА, соответственно код микрооперации в ЗУ соответствует начальному адре- су *АH*, 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 эквива- лент индекса реализуемой микрооперации (*y*1 соответствует 0000001). Поле второй микрооперации остается незаданным (заполняется нулями). Данная микрокоманда не является последней в микропрограмме, значит, *yk* = 0. Вторая микрокоманда реализует вторую вершину ГСА и выполняется вслед за первой микрокомандой. Поэтому адресу второй микрокоманды соответствует увели- ченный на единицу адрес первой микрокоманды (53110 = 10000100112). Полу- ченные результаты по первой микрокоманде представлены в таблице 8.1.

Таблица 8.1 – Микропрограмма заданной ГСА на первом этапе

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер микро-  команды | Номер верши- ны 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 *Т* = 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 (*АT* = 1000010011, *АC* = 1000010100), в против- ном случае переходим по адресу данной микрокоманды, записанному в поле *А*. Описанное поведение перехода соответствует модификатору *M* = 1. Следую- щей рассматриваемой вершиной будет вершина 8. Следует отметить, что за- полнение поля *А* на данном этапе является невозможным, пока не будет рассчи- тан адрес последней команды в данной ветви. Это поле остается незаполнен- ным. Таким образом, результаты на данном этапе примут вид, представленный в таблице 8.2.

Таблица 8.2 – Микропрограмма заданной ГСА на втором этапе

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер  микро- команды | Номер вер- шины 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 яв- ляется то, что в вершине 8 записаны три операционные команды. Так как код микрооперационной команды содержит лишь два поля для операций, то необ- ходимо представить вершины 8.1 и 8.2 (рисунок 8.6).

8

8.1

*y*21 *y*15

8.2

*y*23

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 микро-  команды | Номер вер- шины 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 *М* в МКП присутствует.

Демонстрация первого случая соответствует поведению микропрограммы в заданном примере при *М* = 1. Тогда адресом выполняемой микрокоманды 5 должен быть адрес микрокоманды, реализующий вершину 8, но такая микро- команда 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 следующе- му адресу за адресом микрокоманды 5 (рисунок 8.7). В поле *А* микрокоманды, реализующей вершину 9.1, необходимо записать адрес микрокоманды, реали- зующий вершину 8 (8.1). При таком условии микропрограмма, выполняя мик- рокоманду 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 (*АC* = *A*9.1 + 1).



8

*y*21 *y*15 *y*23

9

*x*7

1

0

9.1

*x* (0)

0

0

*y*17

10

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

Реализация представленных вершин с помощью микрокода представлена в таблице 8.4.

Таблица 8.4 – Микропрограмма заданной ГСА на четвертом этапе

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер микро-  команды | Номер вер-  шины 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 микроко- манды, реализующей вершину 9.1.

Для второго способа такой проблемы нет, однако требуется наличие до- полнительного поля модификатора *М*. В данном случае разрядность микроко- манды позволяет использовать такое поле. Модификатор *М* определяет, для ка- кой ветки адрес следующей команды задается, а для какой используется следу- ющий после текущего. Задав в поле модификатора значение нуль (*М* = 0), мик- ропрограмма будет использовать увеличенный на единицу адрес для ветви при условии *xi* = 0. Тогда полученная микропрограмма примет вид, представленный в таблице 8.5.

Таблица 8.5 – Микропрограмма заданной ГСА на пятом этапе

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер микро-  команды | Номер вер-  шины 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 *М*, сле- дует продолжать заполнение таблицы 8.5.

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

Таблица 8.6 – Микропрограмма заданной ГСА на шестом этапе

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер микро-  команды | Номер вер-  шины 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.

6

*y*5 *y*3

12

*y*10

6.1

*x*0(0)

0

7

*y*20, *yk*

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 микро-  команды | Номер вер-  шины 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.

###### 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 *yk*, 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 *yk*, 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 *yk*, 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 *yk*, 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 *yk*, 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 *yk*, чтобы задать последнюю микрокоманду микропрограммы);
* емкость памяти для записи программы – 2 Кбайт;
* длина ячейки памяти – 16 бит;
* начальный адрес размещения микропрограммы равен нулю.

0

*x*7

1

*x*5

1

0

0

*x*9

1

*y*17

*y*27*y*17

*y*1

*y*15*y*89

*y*2*y*16*yk*

*x*19

0

1

*y*32

*y*17

*y*15*y*45

0

*x*1

0

*x*5

1

1

*y*7*y*12*yk*

*y*4*y*11*y*22

0

*x*13

1

*x*1

1

0

*x*9

0

1

*x*7

1

0

*x*10

1

0

*y*51*y*43

*y*71*y*21*yk*

*y*16

*y*1*y*5

*y*99*y*55

*y*90*y*48

*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

###### Контрольные вопросы

1. Что такое АЛУ? Назовите основные блоки АЛУ.
2. Изобразите структурную схему микропрограммного устройства управ- ления.
3. Назовите типы микрокоманд блока управления, поля микрокоманд.
4. В чем заключаются особенности составления микропрограмм с исполь- зованием модификатора дисциплины переходов и без него?

###### Литература

* 1. Гашков, С. Б. Системы счисления и их применение / C. 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. Саве- льев. – 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 мик- росхем / 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 60х84 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