А. Я. САВЕЛЬЕВ

д-р техн. наук, проф.

АРИФМЕТИЧЕСКИЕ И ЛОГИЧЕСКИЕ ОСНОВЫ ЦИФРОВЫХ АВТОМАТОВ

Допущено
Министерством высшего и среднего
специального образования СССР
в качестве учебника для студентов вузов,
обучающихся по специальности
«Электронные вычислительные машины»

МОСКВА
•ВЫСШАЯ ШКОЛА• 1980

ББК 32.97
С 12
УДК 681.3

Рецензенты: кафедра «Электронные вычислительные машины» Московского лесотехнического института и д-р техн. наук, проф. Г. Н. Соловьев (Московский инженерно-физический институт)

Савельев А. Я.

С12 Арифметические и логические основы цифровых автоматов: Учебник. — М.: Высш. школа, 1980. — 255 с., ил.

В пер.: 90 к.

В книге рассмотрены основные вопросы теории ЭВМ, широко используемые при проектировании ЭВМ: разработка машинных алгоритмов выполнения арифметических и логических операций, методы логического анализа и синтеза электронных схем ЭВМ; изложены требования к курсовой работе по данной дисциплине; дан пример типового задания на курсовую работу.

Теоретический материал иллюстрируется примерами.

Предназначается для студентов вузов, обучающихся по специальности «Электронные вычислительные машины». Может быть полезна специалистам, работающим в соответствующей области.

С $\frac{30502-333}{001(01)-80}$ 86—80 2405000000

БФ7
ББК 32.97

Посвящается 150-летию Московского высшего технического училища им. Н. Э. Баумана 1830—1980 гг.

ПРЕДИСЛОВИЕ

Настоящий учебник полностью соответствует программе курса «Арифметические и логические основы цифровых автоматов». В нем обобщены результаты долголетнего опыта чтения лекций автором в МВТУ им. Н. Э. Баумана, что позволило сконцентрировать внимание на наиболее существенных вопросах, необходимых для подготовки студентов к восприятию учебного материала, изучаемого в последующих курсах «Расчет и проектирование элементов ЭВМ», «Основы теории и проектирования ЦВМ» и др.

Содержание учебника делится на два раздела:

«Арифметика цифровых автоматов» (гл. 1—8);

«Логические основы цифровых автоматов» (гл. 9—11).

Материал обоих разделов связан в единое целое ссылками и общими примерами. Автор стремился изложить весь материал таким образом, чтобы показать неразрывную связь алгоритмов функционирования и схем, с помощью которых реализуются эти алгоритмы. Насколько это удалось, автор надеется судить по отзывам и замечаниям, которые можно направить по адресу: *Москва, К-51, Неглинная ул., д. 29/14, изд-во «Высшая школа»*, за что автор заранее благодарен.

Автор приносит свою искреннюю признательность коллективу кафедры «Электронные вычислительные машины» Московского лесотехнического института, а также д-ру техн. наук, проф. Г. Н. Соловьеву за все критические замечания, высказанные в процессе рецензирования рукописи. Автор также благодарен д-ру техн. наук, проф. В. Я. Петрову за ценные советы и любезно предоставленный материал, а также аспирантам и студентам, принимавшим непосредственное участие в отработке учебного материала для практических занятий.

Автор

Введение

По значимости для прогресса науки и техники появление ЭВМ можно поставить в один ряд с началом освоения космоса и практическим применением атомной энергии.

ЭВМ, появившись около 30 лет назад, открыли новую страницу в истории человеческих знаний и возможностей, высвободили тысячи вычислителей, в невиданных масштабах подняли производительность труда ученых, дали возможность изучать сложнейшие процессы. Сейчас нет ни одной отрасли народного хозяйства, где нельзя было бы применить ЭВМ; более того, целые разделы науки и техники не смогут существовать без них. Суммарная мощность ЭВМ определяет информационно-вычислительную мощь любой страны.

На XXV съезде КПСС отмечалось, что за истекшее десятилетие выпуск вычислительной техники увеличился в 4,3 раза, расширилось ее использование. Директивами съезда на десятую пятилетку предусмотрено увеличить выпуск средств вычислительной техники в 1,8 раза, развивать производство универсальных и управляющих вычислительных комплексов, периферийного оборудования, приборов, устройств регистрации и передачи информации для автоматизированных систем управления технологическими процессами и оптимального управления в отраслях народного хозяйства.

Появление ЭВМ было подготовлено историческим развитием средств вычислений. Древнейшим счетным инструментом, который дала сама природа человеку, была его собственная рука. От пальцевого счета берут свое начало пятиричная (одна рука) и десятичная (две руки) системы счисления. Издревне употреблялся еще один вид инструментального счета — деревянные палочки с зарубками (бирки) и веревки с узелками. Но с ростом и расширением торговли они не смогли удовлетворить потребности в средствах вычислений. И вскоре появился специальный счетный прибор, известный в древности под названием «абак», представляющий собой доску с вертикальными желобками, в которых передвигались камешки. Русский абак — счеты — появились на рубеже XVI—XVII вв. Главное их отличие — десятичный принцип счисления. Форма счетов, установленная более 250 лет назад, в настоящее время почти не изменилась. В XVII в. появились и первые логарифмические линейки.

С развитием общества росли потребности в различных вычислениях, которые становились более трудоемкими. Это явилось причиной появления механических счетных устройств. Первым среди них стало суммирующее устройство (1623) Паскаля. В нем часовой механизм был превращен в счетный и вместо стрелок двигался диск с нанесенными на нем числами. Впоследствии Б. Паскаль сделал несколько вариантов суммирующего устройства, но ни одно из них не получило практического применения: устройства были ненадежны в работе и без специальной подготовки пользоваться ими было невозможно. Тем не менее значение суммирующих устройств Паскаля в истории развития вычислительной техники огромно: они послужили переходным этапом от простых счетных приборов к машинам с механическими счетчиками, доказали возможность выполнения механическим устройством определенной части умственной деятельности человека. Создание этих устройств явилось мощным толчком к разработке новых счетных устройств. Работа над счетными устройствами велась в двух направлениях: создание чисто суммирующих устройств и устройств, выполняющих четыре действия арифметики (арифмометр).

В России первое суммирующее устройство было изобретено и изготовлено в 1770 г. Евной Якобсоном — часовым мастером и механиком в г. Несвиже. Суммирующие устройства, создаваемые в то время, не имели применения, они скорее выставлялись напоказ, чем использовались по назначению. Это объяснялось их ненадежностью, неудобством в эксплуатации, серьезными конструктивными недостатками из-за отсутствия необходимой материально-технической и технологической базы. Ввод чисел и выполнение операций в этих устройствах были медленными процессами.

Коренной перелом в создании счетных устройств произошел в середине XIX в., когда появилась необходимая технологическая база, обеспечивающая требуемую для них точность изготовления деталей счетных машин. Кроме того, общественно-экономическая обстановка (бурный рост промышленности, развитие банков и железных дорог) требовала создания надежных и быстродействующих счетных устройств. Для этого было необходимо в первую очередь изменить «медленную установку» чисел. Приближенно эту задачу решило изобретение клавишного ввода. Принципиально решена проблема была лишь с появлением радиоэлектроники. Тем не менее благодаря клавишному механическому вводу в середине 80-х годов XIX в. удалось организовать промышленный выпуск суммирующих устройств, получивших широкое распространение в первой половине нашего столетия. Начиная с 50-х годов в клавишных устройствах стали использовать электропривод, а затем и электронику.

Параллельно с развитием счетных суммирующих устройств создавались арифмометры. Первым в мире арифмометром стала «арифметическая машина» Лейбница, появившаяся в конце XVII в. Сначала Лейбниц пытался лишь улучшить машину Паскаля, но выяснилось, что для выполнения операций умножения и деления необходим совершенно новый принцип. Лейбниц блестяще разрешил эту задачу, предложив использовать цилиндр, на боковой поверхности которого, параллельно образующей, было расположено девять ступенек различной длины. Этот цилиндр впоследствии назвали ступенчатым валиком. Машина не получила широкого распространения, но основная идея Лейбница — идея ступенчатого валика — осталась действенной и плодотворной даже в XX в. На принципе ступенчатого валика был построен и арифмометр Томаса — первое в мире счетное устройство, изготовляемое промышленностью. Создавались арифмометры и другой конструкции. Основным их элементом было зубчатое колесо Однера с переменным числом зубьев.

В 1878 г. в России П. Л. Чебышевым был создан арифмометр, преимуществом которого являлось то, что перенос десятков из младшего разряда в старшие происходил постепенно в процессе накопления единиц и распространялся на последующие разряды. Идеи, заложенные в арифмометре Чебышева, лежат в основе многих видов современных вычислительных устройств.

Существенным недостатком суммирующих устройств и арифмометров считается невозможность значительного увеличения скорости вычислений. Производительность устройств определяется быстротой рук человека. Поэтому ввод информации и управление операциями необходимо передать в ведение машины. Впервые автоматизировал вычислительный процесс английский ученый Ч. Бэббедж, создав проект арифметической машины — прообраз современных компьютеров. Машина состояла из «склада» для хранения чисел (памяти), «мельницы» — для производства арифметических операций (арифметического устройства), устройства, управляющего в определенной последовательности операциями машины (устройство управления), устройства ввода и вывода данных. Для ввода данных предполагалось использовать перфорированную карту. Время на производство арифметических операций оценивалось Ч. Бэббеджем так: сложение и вычитание — 1 с; умножение и деление — 1 мин. Идеи Ч. Бэббеджа не были поддержаны современниками. К ним обратились только в 40-х годах этого столетия при создании автоматической универсальной вычислительной машины «Марк-2».

Первая действующая счетно-аналитическая машина была создана Г. Холлеритом для автоматизации длительной, однообразной и утомительной работы по обработке данных переписи населения в США в 1880 г. Как и в машине Бэббеджа, в качестве носителей информации использовались перфокарты, но все остальное оборудование: простой пробойник (перфоратор), сложный пробойник, сортировальная машина и табулятор — было оригинально.

Конец XIX и начало XX в. характеризуются бурным развитием электротехники, телефонии, радиотехники, а позднее электроники. Большой материал, накопленный в этой области, позволил создать вычислительную машину немеханического типа.

В 1947 г. была закончена работа над релейной вычислительной машиной «Марк-2», в которой впервые использовалась двоичная система счисления, а для запоминания чисел, выполнения арифметических операций и операций управления — электромеханические реле (13 000 шт.), обладающие двумя устойчивыми состояниями. В машине операции сложения и вычитания занимали примерно 0,125 с, умножения — 0,25 с.

Одной из удачных конструкций релейных вычислительных машин была машина РВМ-1, сконструированная и построенная под руководством советского инженера Н. И. Бессонова в 1956 г.

Главный недостаток релейных вычислительных машин — отсутствие хранимой в памяти программы (малая оперативная память), а также невысокая скорость работы и малая надежность.

В 1943 г. в Гарвардском университете под руководством американских ученых Д. Моучли и Д. Эккерта приступили к созданию электронной вычислительной машины (ЭВМ). К этому времени уже были известны и построены диоды (1904), триод (1905), триггер (1918). Машина создавалась по заказу артиллерийского управления и предназначалась для расчета баллистических таблиц. Завершенная в конце 1945 г. машина, получившая название ЭНИАК, имела громадные размеры: содержала 18 000 электронных ламп и 1500 реле, потребляла около 150 кВт электроэнергии — мощность, достаточная для работы небольшого завода. Использование электронных ламп позволило резко повысить скорость выполнения машинных операций: сложение — 0,0002 с, умножение — 0,0028 с. Управление счетом осуществлялось с помощью программ, набираемых вручную на многочисленных коммутационных досках и переключателях. Несоответствие между временем решения задачи и временем ее подготовки вручную было настолько большим, что выигрыш от скорости вычисления почти полностью покрывался проигрышем во времени на подготовительных операциях.

Создание вычислительной машины ЭНИАК положило начало бурному развитию ЭВМ первого поколения.

В СССР первая малая электронная счетная машина (МЭСМ) — прототип современных ЭВМ была создана в 1951 г. под руководством С. А. Лебедева. Для МЭСМ характерно наличие универсального арифметического устройства, выполнявшего 50 арифметических или логических операций в секунду. Связанное с универсальным арифметическим устройством оперативное запоминающее устройство в свою очередь могло быть соединено с долговременным запоминающим устройством, на котором осуществлялся ввод и хранение команд. В случае математической ошибки или переполнения разрядной сетки машина останавливалась. Ее потребляемая мощность составляла 25 кВт. По сравнению со специализированной машиной ЭНИАК (США, 1946) созданная С. А. Лебедевым машина имела принципиально новое решение.

МЭСМ была одной из первых в мире ЭВМ с параллельной обработкой кодов. Эта машина стала базовым прототипом для мирового цифрового математического машиностроения и обусловила переход к новому периоду развития искусства программирования.

Появление МЭСМ послужило мощным толчком для разработки широкого круга вопросов вычислительной математики: на машине было решено большое количество задач ядерной физики, осуществлен расчет линии электропередачи Куйбышев — Москва, решены задачи ракетной баллистики и др., решение которых вручную надолго задержало бы развитие некоторых важных направлений отечественной науки и техники. Разработка МЭСМ носила экспериментальный характер и явилась необходимым этапом создания первой быстродействующей электронной счетной машины.

В процессе создания МЭСМ разрабатывались, монтировались и опробовались быстродействующие устройства и узлы большой электронной счетной машины (БЭСМ), монтаж и отладка которой были завершены в 1953 г.

В течение нескольких последующих лет БЭСМ с быстродействием 8 тыс. операций/с, была самой быстродействующей машиной в Европе. На ней были решены многие задачи, считавшиеся ранее неразрешимыми из-за большого объема вычислений. Весьма примечательным было то, что ряд технических решений, воплощенных в БЭСМ, предвосхитил идеи ЭВМ второго поколения. Так, в состав машины входило специальное устройство контроля, а независимое подключение к памяти арифметического устройства и устройств ввода и вывода соответствовало структуре мультипрограммных машин. Особо важное значение имел схемный метод обращения к подпрограмме и возможность модификации команд с помощью систем местного и центрального управления командами, что открыло новые возможности в развитии искусства программирования.

Структура и основные схемы БЭСМ стали классическими; они были положены в основу быстродействующих машин БЭСМ-2, М-2 и др.

В 1953 г. под руководством Ю. А. Базилевского была создана цифровая вычислительная машина (ЦВМ) «Стрела», в 1954 г. под руководством Б. И. Рамеева — ЭВМ «Урал». Почти одновременно с этими машинами появились такие ЭВМ, как М-3, «Минск-1» и др., которые составили семейство отечественных ЭВМ первого поколения.

Характерными чертами ЭВМ первого поколения можно считать не только использование электронных ламп в основных и вспомогательных схемах, но и наличие параллельного арифметического устройства, разделение памяти машины на быстродействующую оперативную ограниченного объема (выполненную на электроннолучевой трубке или на ферритовых сердечниках) и медленную внешнюю большого объема (использовавшую накопители на магнитных барабанах и лентах), применение полупроводниковых диодов и магнитных сердечников в логических элементах машины, перфолент и перфокарт как носителей информации при вводе и выводе данных.

Среднее быстродействие ЭВМ первого поколения достигало десятка тысяч арифметических операций в секунду.

Поиск структур, обеспечивающих максимальную загрузку всех устройств за счет совмещения их работы во времени, привел к появлению ЭВМ второго поколения, в которых на смену ламповым схемам пришли транзисторные. Основу технической базы ЭВМ второго поколения составили полупроводниковые диоды и транзисторы.

ЭВМ второго поколения по сравнению с ЭВМ первого поколения отличались более высокой надежностью, меньшим потреблением энергии, более высоким быстродействием. Их быстродействие достигалось за счет повышения скорости переключения счетных и запоминающих элементов и изменений в структуре машины.

Наиболее мощной отечественной ЭВМ второго поколения является ЭВМ БЭСМ-6, созданная под руководством С. А. Лебедева. Трудно переоценить то значение и влияние на развитие вычислительной техники и других областей науки, которое оказало создание этой высокопроизводительной оригинальной по архитектуре и структуре отечественной вычислительной машины.

В нашей стране на основе БЭСМ-6 были созданы центры: коллективного пользования, управления, координационно-вычислительные и др. ЭВМ БЭСМ-6 до настоящего времени широко используется в системе проектирования для разработки математического обеспечения новых ЭВМ, моделирования сложных физических процессов и процессов управления.

Архитектуру и структуру семейства ЭВМ БЭСМ-1, БЭСМ-2, М-20, БЭСМ-3М, БЭСМ-4 характеризуют целостность концепций и изящные инженерные решения. Наиболее полно это проявилось в ЭВМ БЭСМ-6: несмотря на то что машина является сложной системой, механизмы функционирования ее устройств, их функциональные связи легко понимаются, четко интерпретируются, а следовательно, машина БЭСМ-6 эксплуатируется легко. Элементная база ЭВМ БЭСМ-6 совершенно новая. Все схемы машины записаны формулами булевой алгебры. Машина БЭСМ-6 ни по системе команд, ни по внутренней структурной организации не является копией какой-либо отечественной или зарубежной установки.

Для ЭВМ второго поколения характерен параллелизм в работе отдельных блоков, начиная от «перекрытия» времен выполнения отдельных команд и кончая параллельным выполнением двух команд или более из одной или из разных программ, что позволило достичь быстродействия до миллиона операций в секунду. Дальнейшее увеличение быстродействия ЭВМ тормозилось конструктивным выполнением электронных схем, собираемых из отдельных элементов — резисторов, конденсаторов, диодов, транзисторов.

Миниатюризация конструктивных элементов затрудняется необходимостью работы с каждым элементом в отдельности. Выходом из этих затруднений явилась интегральная технология. Малые интегральные схемы (МИС) стали базой машин третьего поколения, появившихся в 60-х годах.

В интегральных схемах роль электронных приборов и элементов выполняют небольшие группы молекул. Основой для таких схем служат полупроводниковые материалы, чаще всего кремний. Специально выращенные большие кристаллы кремния, имеющие очень высокую степень химической чистоты, разрезаются на отдельные пластины, на поверхности которых или внутри специальным способом формируются участки, обладающие свойствами конденсаторов, сопротивлений, диодов, транзисторов и т. д. Достаточно тончайшим металлическим выводом или просто «каналом связи» внутри кристалла соединить один его участок с другими, выполняющими ту или иную функцию, и интегральная схема готова. Одна интегральная схема заменяет большое число различных деталей и позволяет избавиться от многих недостатков полупроводниковых схем. Переход на интегральные схемы способствовал улучшению качества ЭВМ, уменьшению их габаритов и потребляемой ими энергии.

Интеграция различных элементов устранила многие причины, вызывающие возникновение неисправностей. Во-первых, у интегральных схем, состоящих из десятков элементов, небольшое количество вводов и выводов, а до перехода на интегральные схемы их было гораздо больше. Во-вторых, миниатюризация уменьшила нежелательные связи между элементами. Это положительно сказалось на увеличении быстродействия машины. Достоинств у интегральных схем немало. И все же такие характеристики ЭВМ, выполненных на интегральных схемах, как быстродействие и надежность, не являются главными, определяющими и основополагающими. Более существенное новшество — изменившиеся методы производства этих машин и организации их работы.

В чем же основное отличие современной ЭВМ третьего поколения от ее предшественников?

1. ЭВМ третьего поколения оперируют с произвольной буквенно-цифровой информацией. В них фактически соединились два направления предыдущих поколений машин: ЭВМ для делового, коммерческого применения с обработкой алфавитной информации и ЭВМ, предназначенные для научных учреждений и обработки цифровой информации.

2. Изменился порядок работы ЭВМ третьего поколения; эти машины построены по принципу независимой параллельной работы различных их устройств: процессоров, средств внешней памяти. Независимую работу устройств обеспечивают каналы, управляемые специальным устройством, куда поступает информация от потребителей ЭВМ. Это устройство и осуществляет первичную переработку информации, освобождая основное устройство от непроизводительной работы. Благодаря параллельной работе отдельных устройств ЭВМ может выполнять серию операций: переписывать информацию для очередной задачи с магнитной ленты или магнитного диска, выводить информацию для соответствующего устройства, вводить информации и т. д.

Типичными представителями ЭВМ третьего поколения являются машины единой системы (ЕС ЭВМ), представляющие собой семейство машин, предназначенных для решения научно-технических, экономических и управленческих задач, применения в различного рода АСУ и системах обработки данных. Они созданы совместными усилиями коллективов ученых, инженеров и рабочих Болгарии, Венгрии, ГДР, Польши, СССР и Чехословакии. Промышленный выпуск первых моделей ЕС ЭВМ был начат в 1972 г. Большое быстродействие (до 1,5 млн. операций) и широкие возможности являются базой для эффективного использования моделей ЕС ЭВМ.

В машинах третьего поколения в одной интегральной схеме совмещается несколько элементов. Это большое достижение миниатюризации, но еще не предел. ЭВМ четвертого поколения будут строиться на больших интегральных схемах (БИС). В одной такой схеме объемом всего лишь в доли кубического сантиметра уместится блок, занимавший в ЭВМ первого поколения целый шкаф. Ожидается повышение производительности ЭВМ. Если в ЭВМ третьего поколения быстродействие достигает 20—30 млн. операций, то машины четвертого поколения будут производить сотни миллионов операций в секунду. Соответственно возрастет и объем памяти. Наряду с усовершенствованием традиционных устройств памяти на магнитных дисках и лентах будет создана память без движущихся частей. Общий объем внешней памяти в крупнейших машинах четвертого поколения превысит 10^{14} символов, что эквивалентно библиотеке, состоящей из нескольких миллионов объемистых томов.

На основании вышеизложенного можно сделать вывод, что по мере развития ЭВМ стабильно проявляется тенденция к увеличению их быстродействия. Это объясняется рядом обстоятельств, главные из которых следующие. Так как основной узел ЭВМ — сумматор, производящий только операцию сложения, то решение любой задачи, с одной стороны, должно быть сведено к выполнению какого-то количества простых действий. Например, при умножении числа 241 на число 358 фактически необходимо число 241 сложить 16 раз. Аналогичным образом выполняются и другие арифметические операции. С другой стороны, предположим, что требуется решить некоторую задачу, объем вычислений в которой составляет 10^{13} операций сложения. Если вычислительная машина работает со скоростью 10^6 операций/с, то решение задачи займет 10^7 с, что примерно составит 3000 ч. Работать бесперебойно такой длительный срок ЭВМ не всегда могут. Поэтому естественно стремление пользователя решать даже самые сложные задачи за короткий промежуток времени, чтобы неисправности в работе ЭВМ не влияли на результат решения.

Элементной базой для ЭВМ пятого поколения, по-видимому, станет оптоэлектроника с использованием когерентного излучения. А поскольку скорость света значительно выше скорости электронов, то повысится как быстродействие машины, так и пропускная способность линий связи, по которым информация должна поступать в ЭВМ. Для решения этой задачи сделано уже немало. Созданы световоды с малыми потерями — на расстоянии в 1 км интенсивность света в них уменьшается всего в два раза.

Перспективным является параллельное преобразование информации, представляемой в виде голограмм и соответствующих вычислительных сред. Если соединить в одно целое быстродействующие запоминающие устройства и возможности голографии, то машины будущего смогут вместить в своей памяти и выдавать по первому же приказу все информационное богатство, накопления человечеством за многовековой путь развития.

Таким образом, поиск новых принципов построения ЭВМ, совершенствование уже известных алгоритмов выполнения арифметических и логических операций — главнейшие задачи для специалиста в области проектирования и создания ЭВМ. Все дальнейшее изложение данной книги посвящено изучению принципов построения алгоритмов функционирования машин и методов формального описания их работы.

Раздел 1

×	0	1
0	0	0
1	0	1

+	0	1
0	0	1
1	1	10

Арифметика цифровых автоматов

$$
\begin{array}{r}
101101 \\
+\ \ 0110 \\
\hline
110011
\end{array}
\qquad
\begin{array}{r}
\times\ 101101 \\
110 \\
\hline
+\ 1011010 \\
101101\ \\
\hline
10001110
\end{array}
$$

1 ГЛАВА

ОБЩИЕ СВЕДЕНИЯ О ВЫЧИСЛИТЕЛЬНЫХ МАШИНАХ

§ 1.1. Классификация

Вычислительная машина (ВМ) — *комплекс технических средств, объединенных общим управлением и предназначенных для вычислений и переработки информации по заданному алгоритму.*

Наличие общего управления — характерная черта ВМ (поэтому следует отличать ВМ от вычислительных (счетных) приборов и устройств, не имеющих общего управления).

При решении задач на ВМ, как правило, выполняется такая последовательность действий:

ввод информации или установка исходных данных;

переработка или преобразование введенной информации;

определение результатов и вывод переработанной информации.

Таким образом, информация определяет многие процессы, происходящие в ВМ.

Различают информацию непрерывную и дискретную.

Функция $f(t)$, изображенная на рис. 1.1 *а*, может быть представлена в непрерывном (рис. 1.1, *б*) и дискретном (рис. 1.1, *в*) видах. В непрерывном виде эта функция может принимать любые вещественные значения в данном диапазоне изменения аргумента t, т. е. множество значений непрерывной функции бесконечно. В дискретном виде функция $f(t)$ может принимать вещественные значения только при определенных значениях аргумента. Какой бы малый интервал дискретности (т. е. расстояние между соседними значениями аргумента) ни выбирался, множество значений дискретной функции для заданного диапазона изменений аргумента (если он не бесконечный) будет конечно или ограничено.

Вид перерабатываемой информации влияет на структуру ВМ, которые в зависимости от этого делят на два основных класса (рис. 1.2): аналоговые, цифровые.

Аналоговая вычислительная машина (АВМ) — *машина, оперирующая информацией, представленной в виде непрерывных изменений некоторых физических величин.*

В качестве физических переменных могут быть использованы сила тока в электрической цепи, изменение скорости или ускорения движения тела и т. п.

Многие явления в природе математически описываются одними и теми же уравнениями. Следовательно, появляется возможность с помощью одного физического процесса моделировать различные процессы, имеющие одно и то же математическое описание.

Цифровая вычислительная машина (ЦВМ) — *машина, оперирующая информацией, представленной в дискретном виде.*

В настоящее время в математике разработаны методы численного решения многих видов уравнений. Следовательно, появилась возможность решать различные уравнения и задачи с помощью набора простых арифметических и логических операций. Поэтому любая ЦВМ, как правило, является универсальным вычислительным средством. В противоположность этому АВМ по своему характеру предназначены для решения только узкого круга задач, т. е. АВМ специализированы для решения каких-то классов задач.

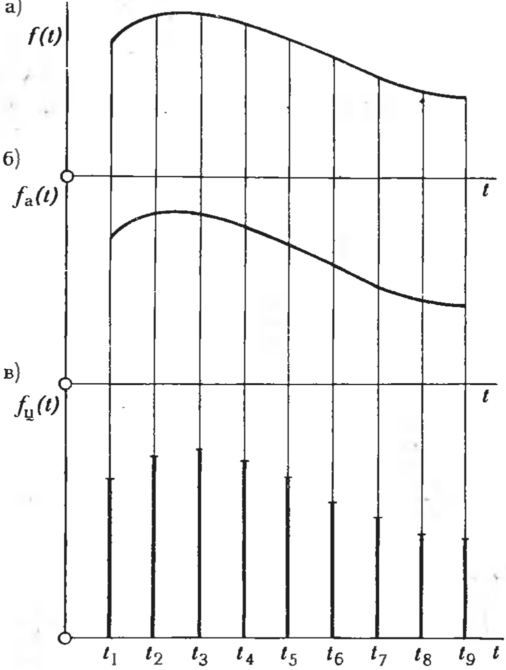


Рис. 1.1. Представление информации в непрерывном и дискретном видах

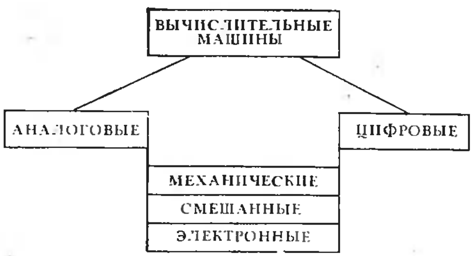


Рис. 1.2. Классификация вычислительных машин

В последние годы внимание разработчиков ВМ привлечено к созданию таких технических средств, в которых совмещены положительные качества ЦВМ (высокая точность и универсальность в отношении классов задач) и АВМ (оперативность ввода информации и быстродействие в выполнении операций). Такие машины получили название **комбинированных** или **гибридных ВМ.**

По принципу действия основных узлов АВМ и ЦВМ разделяют на механические, смешанные (гидромеханические, электромеханические, пневмомеханические и т. п.) и электронные.

Наибольшее распространение в настоящее время получили электронные вычислительные машины (ЭВМ), выполненные на основе применения новейших достижений электроники (полупроводники, интегральные схемы и т. д.). Основной задачей данной книги является изложение принципов построения алгоритмов функционирования этих машин.

§ 1.2. Электронные цифровые вычислительные машины

В зависимости от назначения (объема и характера решаемых задач), особенностей конструкции и структурного построения ЭВМ можно разделить на универсальные и специализированные (рис. 1.3).

Универсальная ЭВМ — *машина, обладающая широкими возможностями в отношении решения задач в различных отраслях науки, техники и народного хозяйства.*

Универсальные ЭВМ, как правило, характеризуются **быстродействием** — *количеством действий (вычислительных или логических операций), выполняемых машиной в единицу времени (секунду).*

По быстродействию универсальные ЭВМ делят на малые (до 5 тыс. операций/с), средние (до 200 тыс. операций/с), большие (свыше 200 тыс. операций/с).

В табл. 1.1 приведены характеристики некоторых наиболее распространенных типов машин единой системы, которые разработаны в социалистических странах.

Специализированные ЭВМ — *машины, предназначенные для решения какого-то определенного круга задач или одной задачи.*

Специализированные ЭВМ могут быть счетными, управляющими, информационными.

Счетные ЭВМ — *машины, решающие ограниченный круг задач или выполняющие только решение систем алгебраических уравнений.*

В отличие от универсальных машин в них реализован какой-то заранее заданный набор программ. Такие машины проще строить по принципу «жестких» программ. Специализированными счетными ЭВМ являются, например, машины «Проминь», «Наири» и т. п.

Управляющие ЭВМ — *машины, предназначенные для управления процессами или объектами в автоматическом режиме.*

Особенностью подобного типа машин является то, что любой процесс в реальном масштабе времени или объект характеризуется информацией непрерывного вида. Поэтому в состав управляющих ЭВМ должны входить преобразователи информации из непрерывного вида в дискретный (на входе ЭВМ) и из дискретного вида в не-

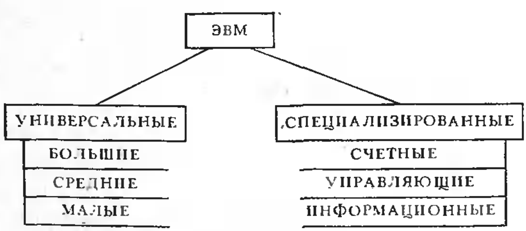


Рис. 1.3. Классификация ЭВМ

Таблица 1.1

Характеристики ЕС ЭВМ	Тип							
	ЕС-1010 (ВНР)	ЕС-1020 (СССР)	ЕС-1022 (СССР)	ЕС-1030 (СССР, ПНР)	ЕС-1035 (СССР)	ЕС-1040 (ГДР)	ЕС-1050 (СССР)	ЕС-1060 (СССР)
Среднее время выполнения операций, мкс:								
коротких операций	2,1—4,3	—	—	5—10	5,5	0,9—1,8	0,65	0,32
сложения — вычитания *	2,1—3,1	20—30	3,3—6,0	7—11	2—4	1,4—2,0	0,65	0,32
умножения — деления *	40	220—400	29—68	32—93	20—36	7—13,8	2—8	1,55—5,75
сложения — вычитания **	—	50—70	14—18	10—14	6—11	2,5—3,6	1,4	1,8
умножения — деления **	—	400—480	39—60	27—51	14—48	6,5—18,1	2—7,2	2,85—4,40
Средняя производительность, операций/с	$60\cdot10^3$	$15\cdot10^3$	$80\cdot10^3$	$70\cdot10^3$	До $200\cdot10^3$	$300\cdot10^3$	$500\cdot10^3$	$1,5\cdot10^6$
Емкость ОЗУ, кбайт	8—64	64—256	256—512	128—512	256—512	256—1024	256—1024	2048—8192
Количество селекторных каналов	1	2	2	3	4	6	6	6
Скорость передачи информации, кбайт, по каналам:								
мультиплексному	160	16	80	40	40	20—25	30—110	до 110
селекторному	240	300	700	800	740	1300	1300	1250

* Операции над числами, представленными в форме с фиксированной запятой.

** Операции над числами, представленными в форме с плавающей запятой.

прерывный (на выходе машины). Примерами такого типа ЭВМ служат машины из агрегатной системы вычислительной техники (АСВТ) и системы малых ЭВМ (СМ ЭВМ), основные характеристики которых представлены в табл. 1.2. Поскольку управляющие ЭВМ работают в условиях реального масштаба времени, к ним предъявляют высокие требования по надежности.

Информационные ЭВМ — *машины, предназначенные для обработки больших массивов информации, когда сама обработка сводится к выполнению не сложных, но многочисленных действий.*

Например, поиск нужной информации в библиотеке состоит в переработке огромного количества литературы и нахождении только тех книг, которые отвечают на заданный вопрос. Для таких машин (или систем) характерно наличие больших накоплений информации с возможностью передачи ее из одного накопителя в другой, а также то, что количество операций по вводу и выводу информации существенно больше по сравнению с операциями преобразования информации.

§ 1.3. Структурная схема электронной цифровой вычислительной машины

Для автоматического решения задач ЭВМ должна включать в себя следующие устройства. Ввод данных в ЭВМ осуществляется через **устройство ввода информации,** основное назначение которого — преобразование входной информации, представленной в символах **входного алфавита,** в информацию, записанную символами **внутреннего алфавита.** Для подготовки исходных данных для решения задач, программ решения задачи и других вспомогательных данных служат **устройства подготовки данных** (УПД), основной функцией которых является преобразование всякой информации (числовой, текстовой, графической и пр.) в информацию, записанную в символах входного алфавита.

Входная информация — *информация любого вида (числовая, текстовая, графическая, электрические сигналы и т. п.), представленная в символах входного алфавита.*

Входная информация записывается в память ЭВМ.

Память ЭВМ — *функциональная часть ЭВМ, предназначенная для запоминания и (или) выдачи выходной информации, промежуточных и окончательных результатов, вспомогательной информации.*

В памяти машины находится также программа решения задач, через команды которой осуществляется управление работой всей машины. Вся информация в памяти ЭВМ представляется символами внутреннего алфавита, выбор которого влияет на многие характеристики ЭВМ.

Основными параметрами, характеризующими память, являются емкость и время обращения к памяти.

Емкость памяти — *количество слов информации, которое можно записать в памяти.*

Таблица 1.2

Характеристики ЭВМ	Тип						
	М-6000	М-7000	М-400	М-4030	СМ-1	СМ-2	СМ-4
Среднее время выполнения операции, мкс:							
сложения	5	2,5—35	4,3—7,2	6,25—13,5	2,5	2,2—18	1,2—22
умножения	43	11—35*	—	18,2	36,6	10—23*	11—35*
деления	57	18—32	—	—	—	17—40	13—52
Емкость основной памяти, кбайт	До 64	16—130	16—32	128—512	64	256	248
Система счисления	Двоичная	Двоичная	Двоичная	Двоичная, десятичная	Двоичная	Двоичная	Двоичная
Количество подключаемых периферийных устройств	До 8	До 48	4	128	55	56	До 96
Каналы:							
инкрементный	+	—	—	+	—	—	—
прямого доступа	+	—	—	+	—	—	—
Скорость, кбайт/с, передачи информации по каналам:							
инкрементному	250	—	—	50** 140***	—	—	—
прямого доступа	650	680	—	300	До 270 (кслов/с)	700 (кслов/с)	800 (кслов/с)

* Первое число соответствует операциям над числами, представленными в форме с фиксированной запятой, второе число — операциям над числами, представленными в форме с плавающей запятой,

** Мультиплексный канал,

*** Селекторный канал,

При этом **словом** является любая конечной длины упорядоченная последовательность символов алфавита.

Время обращения — *интервал времени между началом и окончанием ввода (вывода) информации в память (из памяти).*

Таким образом, время обращения характеризует затраты времени на поиск места и запись (чтение) слова в память.

Ячейка памяти — *часть памяти, содержащая слово.*

Следовательно, емкость памяти можно выразить количеством слов или количеством ячеек, а время обращения определяет время поиска заданной ячейки и время записи (выборки) слов в (из) этой ячейке. Длина ячейки памяти измеряется количеством **битов** (один бит равен одному двоичному разряду) или **байтов** (один байт равен восьми битам). Ячейка памяти может вмещать информацию разной длины или разного формата. Формат измеряется словом, двойным словом или полусловом в зависимости от принятого для данной ЭВМ способа представления информации.

Для построения запоминающих устройств в качестве физических элементов используют электронные схемы, ферритовые магнитные материалы, магнитные ленты и диски, барабаны с магнитным покрытием, оптические запоминающие элементы и т. д.

Арифметическо-логическое устройство (АЛУ) — *функциональная часть ЭВМ, выполняющая логические и арифметические действия, необходимые для переработки информации, хранящейся в памяти.*

АЛУ характеризуется:

временем выполнения элементарных операций;

средним быстродействием, т. е. количеством арифметических или логических действий (операций), выполняемых в единицу времени (секунду);

набором элементарных действий, которые оно выполняет;

видом алфавита или системы счисления, в которой производятся все действия (выбор системы счисления оказывает влияние на все технические характеристики АЛУ).

Быстродействие АЛУ современных ЭВМ лежит в довольно широких пределах — от сотен операций в секунду до десятков миллионов операций в секунду.

АЛУ может выполнять такие элементарные действия, как сравнение, сложение, вычитание, умножение, деление и целый набор логических операций.

Устройство управления (УУ) — *функциональная часть ЭВМ, предназначенная для автоматического управления ходом вычислительного процесса, обеспечивающая взаимодействие всех частей машины в соответствии с программой решения задачи.*

УУ обращается в память машины, выбирает очередную команду, расшифровывает ее и вырабатывает сигналы, указывающие другим устройствам, что им надлежит делать. Управление от программы решения задачи, которая хранится в памяти ЭВМ, обеспечивает полную автоматизацию процесса решения. Поэтому универсальные ЭВМ называют **программно-управляемыми автоматами.**

Человек может вмешаться в ход решения задачи через пульт оператора, соединенный с устройством управления.

Выводные устройства (ВУ) — *устройства, осуществляющие преобразование результатов решения задачи, представленных в символах внутреннего алфавита, в выходную информацию, представленную символами выходного алфавита. ВУ предназначены для выдачи информации из машины.*

В зависимости от вида выходной информации различают устройства выходные печатающие, графические, отображающие и т. д.

Универсальные ЭВМ, как правило, выдают информацию в виде, удобном для человека. Специализированные ЭВМ могут выдавать на выходе электрические управляющие сигналы или другую информацию, что определяется характером системы, в которой работает машина.

Рассмотренный состав структурной схемы ЭВМ можно назвать классическим. В современных вычислительных машинах комплекс устройств, охватывающий АЛУ, часть оперативной памяти и УУ, принято называть процессором.

Процессор — *самостоятельная функциональная часть ЭВМ, непосредственно осуществляющая процесс преобразования информации и управления им.*

Скорость работы внешних устройств ЭВМ, включающих устройства ввода и вывода информации, значительно меньше скорости работы процессора. Поэтому для более эффективного использования возможностей процессора к нему через входной и выходной каналы подключают несколько одновременно обслуживаемых внешних устройств. Такая схема организации работы ЭВМ характерна для машин третьего поколения (рис. 1.4).

Наличие входного и выходного каналов, а также средств и методов взаимодействия (интерфейс) ЭВМ с внешними устройствами позволяет существенно повысить скорость работы всего комплекса от ввода информации в машину до вывода ее. Фактически для осуществления подобного принципа работы необходимо иметь несколько ЭВМ, каждая из которых выполняет разные функции: управление работой всего комплекса устройств, выполнение арифметических и логических действий, ввод и вывод информации. Все это в конечном итоге свидетельствует о существенном усложнении структуры ЭВМ, и эта тенденция сохраняется для машин четвертого поколения, для которых уже в полной мере можно применять термин «вычислительные системы».

Понятие «вычислительные системы» пока не определено достаточно четко. Наиболее часто под **вычислительной системой (ВС)** *понимается взаимосвязанная совокупность средств вычислительной техники, включающая в себя не менее двух процессов или вычислительных машин, из которых роль основного процессора выполняет одна машина* (см. [23]). В таком понимании различают **разделимые и неразделимые вычислительные системы.** Как правило, разделимые вычислительные системы состоят из нескольких вычислительных машин, каждая из которых может работать самостоятель-

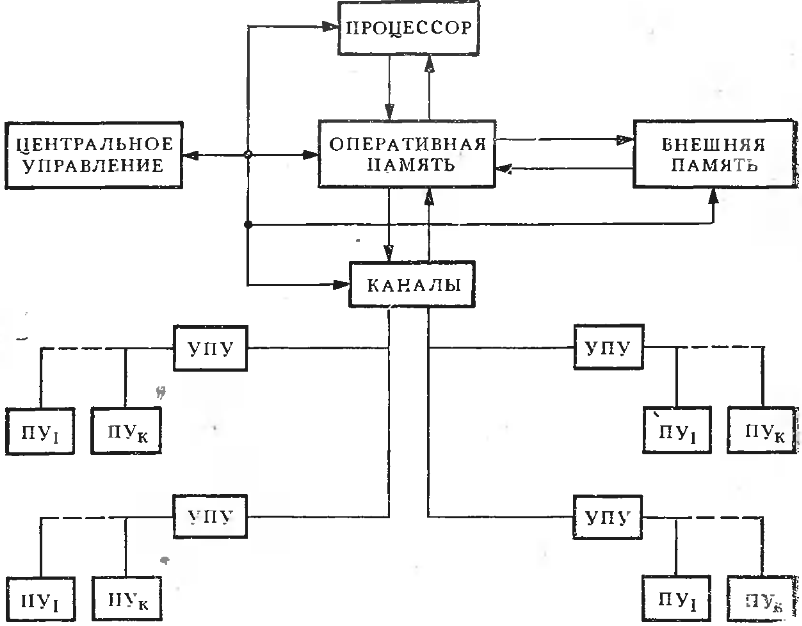


Рис. 1.4. Структурная схема ЭВМ третьего поколения

но. Неразделимые системы (многопроцессорные или мультипроцессорные) состоят из нескольких процессоров, которые могут выполнять свои функции только в составе системы.

На рис. 1.5 представлена структурная схема разделимой вычислительной системы, которую иногда называют также вычислительным комплексом.

До сих пор все выше изложенное касалось в основном технических средств ЭВМ или ВС, обеспечивающих функционирование машины. Чтобы эти технические средства выполняли предписанные функции, необходимы программные средства, обеспечивающие управление работой ЭВМ. В современных ЭВМ или ВС эту роль отводят операционным системам.

Операционные системы — *комплексы связанных между собой программ, выполняющих автоматический выбор и ввод необходимых для решения данной задачи программных средств и автоматическое управление ходом вычислительного процесса.*

Тенденция в развитии электронной вычислительной техники идет не только в направлении усложнения структур технических средств, но и в направлении аппаратной реализации многих функций, которые раньше реализовывались с помощью программных средств.

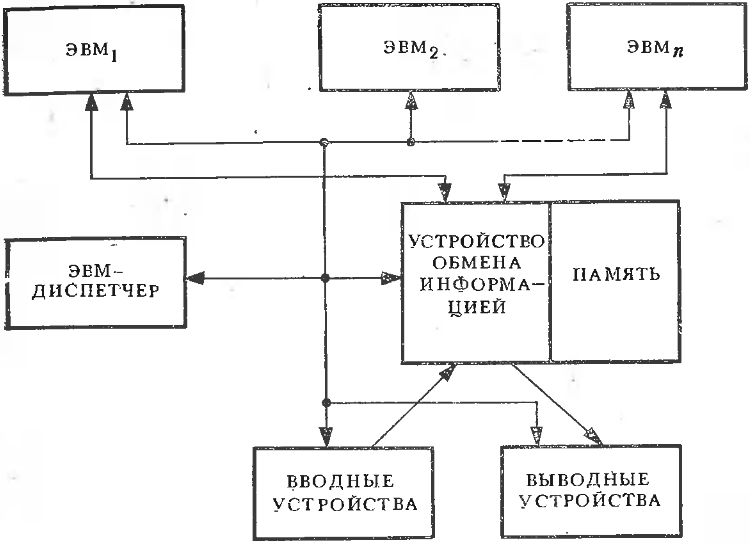


Рис. 1.5. Структурная схема вычислительной системы

§ 1.4. Общие сведения о цифровом автомате

Необходимость формального описания ЭВМ и отдельных ее частей возникает при проектировании и требует применения специального математического аппарата.

Представим себе один из автоматов, с помощью которых продают билеты на поезда для пригородного сообщения. В зависимости от набора монет автомат выдает билет до той или иной зоны. Есть автоматы, которые при этом могут выдать сдачу. Набор монет и нажатие кнопки номера зоны определяют вид выходного результата (билет до соответствующей зоны). Можно сказать, что от набора монет изменяется состояние автомата и появляется на выходе разная информация в виде билетов на поездку. Для пользователя подобный автомат представляется как «черный ящик». Представление о «черном ящике» может быть распространено и на ЭВМ.

Цифровой автомат — *устройство, характеризующееся набором внутренних состояний* $A=\{a_1(t), a_2(t), \ldots, a_n(t)\}$, *в которые оно попадает под воздействием команд программы решения задачи.*

Пусть имеется цифровой автомат с одним входом и одним выходом (рис. 1.6). Тогда математической моделью цифрового автомата является некоторый **абстрактный автомат,** заданный следующим образом: в начальный момент времени $t=t_0$ автомат находится в состоянии $a(t_0)=a_1$ и остается в нем до момента $t=t_1$, когда

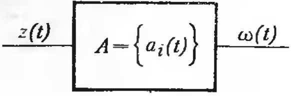


Рис. 1.6. Цифровой автомат

появляется входной сигнал $z(t_1)$. Под воздействием сигнала $z(t_1)$ автомат переходит из состояния a_1 в состояние $a(t_1) = a_2$. При этом возникает выходной сигнал $\omega(t_1)=\omega_1$, определяемый как функция $\omega(t_1)=\lambda[a(t_1), z(t_1)]$. Таким образом, можно принять, что при подаче произвольного сигнала $z(t)=z_f$ автомат переходит из состояния $a(t)$ в состояние $a(t+1)$, которое есть функция $a(t+1)=\delta[a(t), z(t)]$, и в результате вырабатывает выходной сигнал $\omega(t)$.

Выходные сигналы могут вырабатываться при каждом переходе автомата из состояния $a(t)$ в состояние $a(t+1)$ или только при определенных сочетаниях входного сигнала и состояний автомата.

Алгоритм преобразования (переработки) информации — *совокупность правил перехода автомата из одного состояния в другое в зависимости от входной информации и внутренних состояний автомата.*

Следовательно, дискретным входным и выходным сигналам, а также состояниям автомата можно присвоить какие-то условные обозначения или символы некоторого алфавита. При этом возникают понятия входного, внутреннего и выходного алфавитов. Например, если в качестве входного алфавита использовать десятичные цифры 0—9 и разделитель вида «,», то можно представить не только любые числа, целые или дробные, но и текстовую информацию, предварительно закодировав числами буквы русского и латинского алфавитов. Тогда десятичные числа или комбинации чисел являются словами входной информации. Если в качестве выходного алфавита использовать русские буквы, то выходными словами будут любые последовательности символов вида, например «море», «язык», «АААЕЕ» и т. д.

Таким образом, смысл использования понятия «абстрактный автомат» состоит в том, что оно реализует некоторое отображение множества слов входного алфавита Z в множество слов выходного алфавита W.

Понятие «состояние автомата» отражает необходимость описания систем, выходные сигналы которых зависят не только от входных сигналов в данный момент времени, но и от сигналов, поступивших на входы системы ранее. Состояние соответствует некоторой памяти о прошлом, позволяя таким образом устранить время как явную переменную и выразить выходные сигналы как функцию состояний и входных сигналов. Это означает, что работу абстрактного автомата следует рассматривать применительно к конкретным интервалам времени, так как каждому интервалу будет соответствовать свой выходной сигнал $\omega_i(t)$. При этом предполагают, что выходной сигнал на одном из выходов автомата может появиться после соответствующего этому же моменту времени входного сигнала $z_i(t)$ с одновременным переходом из состояния $a_i(t-1)$ в состояние $a_i(t)$. Для того чтобы определить, на каком из выходов должен появиться сигнал, необходимо задавать правила образования выходного сигнала.

Таким образом, абстрактный автомат можно описать с помощью следующих параметров:

a_1 — начальное состояние автомата;

$A=\{a_1, a_2, \ldots, a_m\}$ — множество внутренних состояний;

$Z=\{z_1, z_2, \ldots, z_f\}$ — множество входных сигналов;

$W=\{\omega_1, \omega_2, \ldots, \omega_k\}$ — множество выходных сигналов;

$\Delta=\{\delta_1, \delta_2, \ldots, \delta_n\}$ — совокупность правил перехода из одного состояния в другое;

$\lambda=\{\lambda_1, \lambda_2, \ldots, \lambda_l\}$ — совокупность функций выходов.

Вышеизложенные понятия относятся к абстрактной теории цифровых автоматов, в которой рассматривают автоматы, имеющие один вход и один выход. Следовательно, в этом случае из множества λ задаются правила образования выходного сигнала, т. е. функция λ, и правила перехода из одного внутреннего состояния в другое, т. е. функция δ. Такие ограничения объясняются невозможностью описаний работы абстрактного автомата в общем виде.

Применить приведенное выше формальное описание к ЭВМ можно только в самом общем виде, ограничивая круг рассматриваемых устройств устройствами, входящими в состав процессора. В самом деле, в зависимости от команд, подаваемых устройством управления, арифметическо-логическое устройство будет осуществлять соответствующие действия (изменение внутренних состояний) с выдачей необходимых результатов. Однако изменение внутренних состояний ЭВМ носит настолько сложный характер, что этот процесс невозможно отобразить в аналитическом виде. Поэтому понятие цифрового автомата целесообразно использовать применительно к алгоритмам, реализующим некоторую программу или последовательность операций. При этом каждая операция представляется как элементарное действие, осуществляемое в процессе переработки информации.

§ 1.5. Понятие об алгоритме

Алгоритм — *конечная совокупность точно сформулированных правил решения какой-то задачи.*

Примечание. Термин «алгоритм» своим происхождением обязан имени узбекского математика Аль-Хорезми, который еще в IX в. сформулировал правила выполнения четырех арифметических действий. Появившееся несколько позже слово «алгорифм» связано с Евклидом, древнегреческим математиком, сформулировавшим правила нахождения наибольшего общего делителя двух чисел. В современной математике употребляют термин «алгоритм».

По форме задания алгоритмы могут быть словесными и математическими.

Примером словесной формы алгоритма служит алгоритм Евклида для нахождения наибольшего общего делителя двух чисел a и b.

1. Обозревая два числа a и b, переходи к следующему пункту.

2. Сравни обозреваемые числа (a равно b, a меньше b, a больше b) и переходи к следующему пункту.

3. Если a и b равны, то прекрати вычисление: каждое из чисел дает искомый результат. Если числа не равны, то переходи к следующему пункту.

4. Если первое число меньше второго, то переставь их местами; переходи к следующему пункту.

5. Вычитай второе число из первого, обозревай два числа: вычитаемое и остаток; переходи к п. 2.

По указаниям этого алгоритма можно найти наибольший общий делитель для любой пары целых чисел. Следовательно, массовость — одна из главных черт словесного алгоритма, позволяющего применять его для решения широкого круга задач.

Примером алгебраической формы алгоритма может быть любая математическая формула для нахождения какой-то величины. Например, значение корней уравнения вида $ax^2+bx+c=0$ можно найти по формуле $x_{1,2}=[-b\pm\sqrt{b^2-4ac}]/2a$, которая представляет собой алгоритм нахождения этих корней. Однако, для того чтобы реализовать математическую форму алгоритма, требуется дать еще ряд словесных указаний.

Детерминированный алгоритм — *алгоритм, имеющий место при четкой и ясной системе правил и указаний и однозначных действиях.*

Случайный алгоритм — *алгоритм, предусматривающий возможность случайного выбора тех или иных правил.*

Алгоритм должен обеспечить получение результата через конечное число шагов для любой задачи определенного класса. В противном случае задача неразрешима.

Таким образом, алгоритм дает возможность ответить на вопрос «что делать?» в каждый момент времени. Однако создать алгоритм не всегда возможно, и это подтверждается, например, следующей алгоритмически неразрешимой ситуацией. В городе живет парикмахер, который бреет только тех, кто не бреется сам. Спрашивается, что делать парикмахеру с самим собой? Построить алгоритм решения подобной ситуации невозможно, так как не существует непротиворечивой последовательности правил, следуя которой можно было бы выйти из положения парикмахера.

Численный алгоритм — *алгоритм, соответствующий решению поставленной задачи с помощью арифметических действий (сложение, вычитание, умножение, деление).*

Логический алгоритм — *алгоритм, используемый в случае, если при решении задачи приходится применять некоторые логические действия.*

Таким образом, процесс решения задачи на ЭВМ прежде всего должен быть выражен каким-то алгоритмом. Разработка алгоритмов решения задач — цель работы программиста, а разработка алгоритмов функционирования цифрового автомата для решения поставленных задач — цель деятельности инженера-разработчика.

2 ГЛАВА

ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В ЦИФРОВОМ АВТОМАТЕ

§ 2.1. Системы счисления

Система счисления — *совокупность приемов и правил для записи чисел цифровыми знаками.*

Наиболее известна десятичная система счисления, в которой для записи чисел используются цифры 0, 1, . . ., 9.

Способов записи чисел цифровыми знаками существует бесчисленное множество. Любая предназначенная для практического применения система счисления должна обеспечивать:

возможность представления любого числа в рассматриваемом диапазоне величин;

единственность представления (каждой комбинации символов должна соответствовать одна и только одна величина);

простоту оперирования числами.

Все системы представления чисел делят на позиционные и непозиционные.

Самый простой способ записи чисел может быть представлен в виде выражения

$$A_{(D)}=D_1+D_2+\ldots+D_k=\sum_{i=1}^{i=k} D_i,$$

где $A_{(D)}$ — запись числа A в системе счисления D; D_i — символы системы, образующие базу $D=\{D_1, D_2, \ldots, D_k\}$.

По этому принципу построены непозиционные системы счисления.

Непозиционная система счисления — *система, для которой значение символа не зависит от его положения в числе.*

Принципы построения таких систем не сложны. Для их образования используют в основном операции сложения и вычитания. Например, система с одним символом-палочкой встречалась у многих народов. Для изображения какого-то числа в этой системе нужно записать количество палочек, равное данному числу. Эта система неэффективна, так как запись числа получается длинной. Другим примером непозиционной системы счисления является римская система, использующая набор следующих символов: I, V, X, L, C, D, M и т. д. В этой системе существует отклонение от правила независимости значения цифры от положения в числе. В числах LX и XL символ X принимает два различных значения: +10 — в первом случае и —10 — во втором случае.

В общем случае системы счисления можно построить по следующему принципу:

$$A_{(B)}=a_1B_1+a_2B_2+\ldots+a_nB_n, \tag{2.1}$$

где $A_{(B)}$ — запись числа в системе счисления с основанием B; a_i — цифры (символы) системы счисления с основанием B; B_i — базисы, или основания, системы.

Если в (2.1) положить, что $B_i=q^i$, то

$$B_i=qB_{i-1}. \tag{2.2}$$

Позиционные системы счисления — *системы, удовлетворяющие равенству (2.2).*

Естественная позиционная система счисления имеет место, если q — целое, положительное число.

В позиционной системе счисления значение цифры определяется ее положением в числе: один и тот же знак принимает различное значение. Например, в десятичном числе 222 первая цифра справа означает две единицы, соседняя с ней — два десятка, а левая — две сотни.

Любая позиционная система счисления характеризуется основанием.

Основание (базис) q естественной позиционной системы счисления — *количество знаков или символов, используемых для изображения числа в данной системе.*

Поэтому возможно бесчисленное множество позиционных систем, так как за основание можно принять любое число, образовав, таким образом, новую систему. Например, запись числа в шестнадцатиричной системе может производиться с помощью следующих знаков (цифр): 0, 1, ..., 9, α_1, α_2, ..., α_i (вместо α_i можно записать любые другие символы, например $\bar{0}$, $\bar{1}$, ..., $\bar{5}$).

Для позиционной системы счисления справедливо равенство

$$A_{(q)}=a_nq^n+a_{n-1}q^{n-1}+\ldots+a_1q^1+a_0q^0+a_{-1}q^{-1}+\ldots+a_{-m}q^{-m}, \tag{2.3}$$

или

$$A_{(q)}=\sum_{i=-m}^{i=n} a_iq^i,$$

где $A_{(q)}$ — произвольное число, записанное в системе счисления с основанием q; a_i — коэффициенты ряда (цифры системы счисления); n, m — количество целых и дробных разрядов.

На практике используют сокращенную запись чисел:

$$A_{(q)}=a_na_{n-1}\ldots a_1a_0a_{-1}\ldots a_{-m}.$$

В восьмеричной системе счисления числа изображают с помощью цифр 0, 1, ..., 7. Например, $124{,}537_{(8)}=1\cdot8^2+2\cdot8^1+4\cdot8^0+$ $+5\cdot8^{-1}+3\cdot8^{-2}+7\cdot8^{-3}$.

В двоичной системе счисления используют цифры 0, 1. Например, $1001{,}1101_{(2)} = 1\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 1\cdot 2^{-1} + 1\cdot 2^{-2} + 0\cdot 2^{-3} + 1\cdot 2^{-4}$.

Для записи чисел в троичной системе берут цифры 0, 1, 2. Например, $2122_{(3)} = 2\cdot 3^3 + 1\cdot 3^2 + 2\cdot 3^1 + 2\cdot 3^0$.

В табл. 2.1 приведены эквиваленты десятичных цифр в различных системах счисления.

Таблица 2.1

Десятичная цифра	Эквиваленты в других системах счисления				
	$q=2$	$q=3$	$q=5$	$q=8$	$q=16$
0	0000	000	00	00	0
1	0001	001	01	01	1
2	0010	002	02	02	2
3	0011	010	03	03	3
4	0100	011	04	04	4
5	0101	012	10	05	5
6	0110	020	11	06	6
7	0111	021	12	07	7
8	1000	022	13	10	8
9	1001	100	14	11	9
10	1010	101	20	12	$\bar{1}$

Для любой позиционной системы счисления справедливо, что основание изображается символом 10 в своей системе, т. е. любое число можно записать в виде

$$A_{(q)} = a_n 10^n + a_{n-1} 10^{n-1} + \ldots + a_1 10^1 + a_0 10^0 + a_1 10^{-1} + \ldots \ldots + a_{-m} 10^{-m}. \tag{2.4}$$

В ЭВМ используют в основном позиционные системы счисления. В дальнейшем для простоты изложения будем употреблять термин «система счисления», имея в виду позиционные системы.

Вес разряда P_i числа в позиционной системе счисления — *это отношение*

$$P_i = q^i / q^0 = q^i, \tag{2.5}$$

где i — номер разряда справа налево.

Если разряд имеет вес $P_i = 10^k$, то следующий старший разряд будет иметь вес $P_{i+1} = 10^{k+1}$, а соседний младший разряд — вес $P_{i-1} = 10^{k-1}$. Такая взаимосвязь разрядов приводит к необходимости передачи информации между ними.

Если в данном разряде накопилось значение единиц, равное или большее q, то должна происходить передача единицы в соседний старший разряд. При сложении такие передачи информации называют **переносами**, а при вычитании — **заемами.** Передача переносов или заемов происходит последовательно от разряда к разряду.

Длина числа (ДЧ) — *количество позиций (или разрядов) в записи числа.*

В техническом аспекте длина числа интерпретируется как **длина разрядной сетки** (ДРС).

Для разных систем счисления характерна разная длина разрядной сетки, необходимая для записи одного и того же числа. Например, $96 = 140_{(8)} = 10120_{(3)} = 1100000_{(2)}$. Здесь одно и то же число, записанное в разных базисах, имеет разную длину разрядной сетки. Чем меньше основание системы, тем больше длина числа.

Если длина разрядной сетки задана, то это ограничивает максимальное (или минимальное) по абсолютному значению число, которое может быть записано.

Пусть длина разрядной сетки равна любому положительному числу, например n. Тогда

$$A_{(q)\,\max} = q^n - 1; \quad A_{(q)\,\min} = -(q^n - 1).$$

Диапазон представления (ДП) **чисел в заданной системе счисления** — *интервал числовой оси, заключенный между максимальным и минимальным числами, представленными длиной разрядной сетки:*

$$A_{(q)\,\max} \geqslant \text{ДП} \geqslant A_{(q)\min}. \tag{2.6}$$

§ 2.2. Выбор системы счисления

Правильный выбор системы счисления — важный практический вопрос, решаемый при проектировании ЭВМ, поскольку от его решения зависят такие технические характеристики машины, как скорость вычислений, объем памяти, сложность алгоритмов выполнения арифметических операций.

При выборе системы счисления для ЭВМ необходимо учитывать следующее:

основание системы счисления определяет количество устойчивых состояний, которые должен иметь функциональный элемент, выбранный для изображения разрядов числа;

длина числа существенно зависит от основания системы счисления;

система счисления должна обеспечить простые алгоритмы выполнения арифметических и логических операций.

Десятичная система, столь привычная в повседневной жизни, не является наилучшей с точки зрения ее технической реализации в ЭВМ. Известные в настоящее время элементы, обладающие десятью устойчивыми состояниями (элементы на основе сегнетокерамики, декатроны и др.), имеют невысокую скорость переключения, а следовательно, не могут обеспечить соответствующее быстродействие машины.

Подавляющее большинство компонентов электронных схем, применяемых для построения вычислительных машин, — двухпозиционные. С этой точки зрения наиболее подходящей для ЭВМ является двоичная система счисления. Но рационально ли использование этой системы с точки зрения затрат оборудования? Для ответа на этот вопрос введем следующий показатель.

Показатель экономичности системы — *произведение основания системы на длину разрядной сетки, выбранную для записи чисел в этой системе:*

$$C = qN, \tag{2.7}$$

где q — основание системы счисления; N — количество разрядов.

Если принять, что каждый разряд числа представлен не одним элементом с q устойчивыми состояниями, а q элементами, каждый из которых имеет одно устойчивое состояние, то показатель экономичности укажет условное количество оборудования, которое необходимо затратить на представление чисел в этой системе.

Максимальное число, которое можно изобразить в системе с основанием q,

$$A_{(q)\max} = q^N - 1. \tag{2.8}$$

Из (2.8) можно найти требуемую длину разрядной сетки:

$$N = \log_q (A_{(q)\max} + 1). \tag{2.9}$$

Тогда для любой системы счисления

$$C = q \log_q (A_{(q)\max} + 1).$$

Представим, что величина q принимает любые значения (целочисленные и дробные), т. е. является непрерывной величиной. Это необходимо для того, чтобы рассматривать величину C как функцию от величины q. Данное допущение не является строгим, однако позволяет получить довольно интересный вывод: если за единицу измерения оборудования принят условный элемент с одним устойчивым состоянием, то для сравнения двух систем счисления можно ввести относительный показатель экономичности

$$F = \frac{q \log_q (A_{(q)\max} + 1)}{2 \log_2 (A_{(2)\max} + 1)}, \tag{2.10}$$

позволяющий сравнить любую систему счисления с двоичной системой счисления.

Рассматривая функцию F как непрерывную, нетрудно показать, что она имеет минимум и монотонно возрастает влево и вправо, проходя при этом следующие значения:

q	2	3	4	6	8	10
F	1,000	0,946	1,000	1,148	1,333	1,505

На рис. 2.1 представлена зависимость величины F от основания системы счисления q. Нижняя точка графика соответствует минимуму функции $F(q)$, определяемому из условия $dF/dq = 0$, что соот-

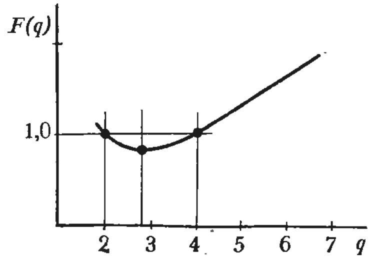


Рис. 2.1. Выбор основания системы счисления

ветствует значению $q=\mathrm{e}$. Следовательно, с точки зрения минимальных затрат условного оборудования наиболее экономичной является система счисления с основанием, равным $\mathrm{e}\approx 2{,}72$.

Используя (2.10), можно доказать, что троичная система счисления экономичнее двоичной. Троичную систему счисления применяют только в ЭВМ «Сетунь». В подавляющем большинстве ЭВМ используют двоичную систему счисления, что объясняется простотой реализации арифметических действий.

Правила двоичной арифметики весьма просты:

сложение

$$0+0=0$$
$$0+1=1$$
$$1+0=0$$
$$1+1=\boxed{1}\,0$$

перенос единицы в старший разряд;

вычитание

$$0-0=0$$
$$1-0=1$$
$$1-1=0$$
$$0-1=\boxed{1}\,1$$

заем единицы в старшем разряде;

умножение

$$0\times 0=0$$
$$0\times 1=0$$
$$1\times 0=0$$
$$1\times 1=1.$$

Однако использование двоичной системы счисления для ЭВМ связано с преодолением дополнительных трудностей, вызванных необходимостью перевода входной информации в двоичную систему счисления и двоичной информации в выходную информацию.

§ 2.3. Методы перевода чисел из одной позиционной системы счисления в другую

В соответствии с (2.3) числа в разных системах счисления можно представить следующим образом:

$$A_{(q_1)}=a_nq_1^n+a_{n-1}q_1^{n-1}+\ldots+a_1q_1^1+a_0q_1^0+a_{-1}q_1^{-1}+\ldots$$
$$\ldots+a_{-m}q_1^{-m}=b_kq_2^k+b_{k-1}q_2^{k-1}+\ldots+b_1q_2^1+b_0q_2^0+b_{-1}q_2^{-1}+\ldots$$
$$\ldots+b_{-s}q_2^{-s}=A_{(q_2)}. \qquad (2.11)$$

Значит, в общем виде задачу перевода числа из системы счисления с основанием q_1 в систему счисления с основанием q_2 можно представить как задачу определения коэффициентов b_j нового ряда, изображающего число в системе с основанием q_2. В такой постановке задачу перевода можно решить подбором коэффициентов b_j. Основная трудность при этом заключается в выборе максимальной степени $q_2{}^k$, которая еще содержится в числе $A_{(q_1)}$. Все действия должны выполняться по правилам q_1-арифметики, т. е. по правилам исходной системы счисления.

После нахождения максимальной степени основания проверяют «вхождение» в заданное число всех степеней нового основания, меньших максимального. Следует иметь в виду, что каждая из отмеченных степеней может «входить» в ряд не более $q_2—1$ раз, так как для любого коэффициента ряда накладывается ограничение:

$$\left.\begin{aligned}0\leqslant a_i\leqslant q_1-1;\\ 0\leqslant b_j\leqslant q_2-1.\end{aligned}\right\} \qquad (2.12)$$

Пример 2.1. Перевести десятичное число $A=96$ в троичную систему счисления ($q_2=3$).
Решение. $96=0\cdot3^5+1\cdot3^4+0\cdot3^3+1\cdot3^2+2\cdot3^1+0\cdot3^0=10120_{(3)}$.
Ответ: $A_{(3)}=10120$.

Примечание. Здесь и в дальнейшем при записи десятичных чисел индекс опускается.

Рассмотренный в примере 2.1 прием может быть использован только при ручном переводе. Для реализации машинных алгоритмов перевода применяют следующие методы.

Перевод целых чисел делением на основание q_2 новой системы счисления. Целое число $A_{(q_1)}$ в системе с основанием q_2 будет записано в виде

$$A_{(q_2)}=b_kq_2^k+b_{k-1}q_2^{k-1}+\ldots+b_1q_2^1+b_0q_2^0.$$

Переписав это выражение по схеме Горнера, получим

$$A_{(q_2)}=(\ldots((b_kq_2+b_{k-1})q_2+b_{k-2})q_2+\ldots+b_1)q_2+b_0. \qquad (2.13)$$

Правую часть выражения (2.13) разделим на величину основания q_2. В результате определим первый остаток b_0 и целую часть $(\ldots((b_kq_2+b_{k-1})q_2+\ldots+b_1)$. Разделив целую часть на q_2, найдем второй остаток b_1. Повторяя процесс деления $k+1$ раз, получим последнее целое частное b_k, которое, по условию, меньше основания системы q_2 и является старшей цифрой числа, представленного в системе с основанием q_2.

Пример 2.2. Перевести десятичное число $A=98$ в двоичную систему счисления ($q_2=2$).

Решение:

$$
\begin{array}{rrrrrrr}
-\,98 & \underline{|\,2} & & & & & \\
\underline{98} & -\,49 & \underline{|\,2} & & & & \\
b_0=0 & \underline{48} & -\,24 & \underline{|\,2} & & & \\
 & b_1=1 & \underline{24} & -\,12 & \underline{|\,2} & & \\
 & & b_2=0 & \underline{12} & -\,6 & \underline{|\,2} & \\
 & & & b_3=0 & \underline{6} & -\,3 & \underline{|\,2} \\
 & & & & b_4=0 & \underline{2} & 1=b_6 \\
 & & & & & b_5=1 &
\end{array}
$$

$\longleftarrow$

Ответ: $A_{(2)}=1100010_{(2)}$.

Пример 2.3. Перевести двоичное число $A_{(2)}=1101001_{(2)}$ в десятичную систему счисления ($q_2=10$). Основание q_2 изображается в двоичной системе эквивалентом $q_2=1010_{(2)}$.

Решение:

$$
\begin{array}{rrr}
-\,1101001 & \underline{|\,1010} & \\
\underline{1010} & -\,1010 & \underline{|\,1010} \\
-\,001100 & \underline{1010} & b_2=1 \\
\underline{1010} & b_1=0000 & \\
b_0=0101 & &
\end{array}
$$

$\longleftarrow$

Ответ: $A=105$. На основании табл. 2.1 можно записать $b_0=0101_{(2)}=5$; $b_1=0000_{(2)}=0$; $b_2=0001_{(2)}=1$.

Этот способ применяют только для перевода целых чисел.

Перевод правильных дробей умножением на основание q_2 новой системы счисления. Пусть исходное число, записанное в системе счисления с основанием q_1, имеет вид

$$A_{(q_1)}=a_{-1}q_1^{-1}+a_{-2}q_1^{-2}+\ldots+a_{-m}q_1^{-m}.$$

Тогда в новой системе с основанием q_2 это число будет изображено как $0, b_{-1}\, b_{-2}\ldots b_{-k}$, или

$$A_{(q_2)}=b_{-1}q_2^{-1}+b_{-2}q_2^{-2}+\ldots+b_{-k}q_2^{-k}+\ldots$$

Переписав это выражение по схеме Горнера, получим

$$A_{(q_2)}=q_2^{-1}(b_{-1}+q_2^{-1}(b_{-2}+\ldots+q_2^{-1}(b_{-(k-1)}+q_2^{-1}b_{-k}))\ldots) \tag{2.14}$$

Если правую часть выражения (2.14) умножить на q_2, то найдем новую неправильную дробь, в целой части которой будет число b_{-1}. Умножив затем оставшуюся дробную часть на величину основания q_2, получим дробь, в целой части которой будет b_{-2}.

Повторяя процесс умножения k раз, найдем все k цифр числа в новой системе счисления. При этом все действия должны выполняться по правилам q_1-арифметики и, следовательно, в целой части получающихся дробей будут проявляться эквиваленты цифр новой системы счисления, записанные в исходной системе счисления.

Пример 2.4. Перевести десятичную дробь $A=0{,}625$ в двоичную систему счисления $(q_2=2)$.

Решение:

$$\begin{array}{r r|r}
 & 0, & 625 \\
 & \times & 2 \\ \cline{2-3}
b_{-1} & 1, & 250 \\
 & \times & 2 \\ \cline{2-3}
b_{-2} & 0, & 500 \\
 & \times & 2 \\ \cline{2-3}
b_{-3} & 1, & 000 \\
 & \times & 2 \\ \cline{2-3}
b_{-4} & 0, & 000
\end{array}$$

Ответ: $A_{(2)}=0{,}1010_{(2)}$.

Пример 2.5. Перевести двоичную дробь $A_{(2)}=0{,}1101$ в десятичную систему счисления $(q_2=1010_{(2)})$.

Решение.

$$\begin{array}{r r|l}
 & 0, & \times\begin{array}{r}1101\\1010\end{array} \\ \cline{2-3}
b_{-1}=8 & 1000, & \times\begin{array}{r}0010\\1010\end{array} \\ \cline{2-3}
b_{-2}=1 & 0001, & \times\begin{array}{r}0100\\1010\end{array} \\ \cline{2-3}
b_{-3}=2 & 0010, & \times\begin{array}{r}1000\\1010\end{array} \\ \cline{2-3}
b_{-4}=5 & 0101, & \quad 0000
\end{array}$$

Ответ: $A=0{,}8125$.

При переводе правильных дробей из одной системы счисления в другую можно получить дробь в виде бесконечного или расходящегося ряда. Процесс перевода можно закончить, если появится дробная часть, имеющая во всех разрядах нули, или будет достигнута заданная точность перевода (получено требуемое количество разрядов результата). Последнее означает, что при переводе дроби необходимо указать количество разрядов числа в новой системе счисления. Естественно, что при этом возникает погрешность перевода чисел, которую надо оценивать (см. § 2.7).

Для перевода неправильных дробей из одной системы счисления в другую необходим раздельный перевод целой и дробной частей по правилам, описанным выше. Полученные результаты записывают в виде новой дроби в системе с основанием q_2.

Пример 2.6. Перевести десятичную дробь $A=98{,}625$ в двоичную систему счисления ($q_2=2$).

Решение. Результаты перевода соответственно целой и дробной частей возьмем из примеров 2.2 и 2.4.

Ответ: $A_{(2)}=1100010,\ 1010_{(2)}$.

Табличный метод перевода. В простейшем виде табличный метод заключается в следующем: имеется таблица всех чисел одной системы с соответствующими эквивалентами из другой системы; задача перевода сводится к нахождению соответствующей строки таблицы и выбору из нее эквивалента. Такая таблица очень громоздка и требует большой емкости памяти для хранения.

Другой вид табличного метода заключается в том, что имеются таблицы эквивалентов в каждой системе только для цифр этих систем и степеней основания (положительных и отрицательных); задача перевода сводится к тому, что в выражение ряда (2.3) для исходной системы счисления надо подставить эквиваленты из новой системы для всех цифр и степеней основания и произвести соответствующие действия (умножения и сложения) по правилам q_2-арифметики. Полученный результат этих действий будет изображать число в новой системе счисления.

Пример 2.7. Перевести десятичное число $A=113$ в двоичную систему счисления, используя таблицу эквивалентов цифр и степеней основания ($q_2=2$).

Таблица эквивалентов:

Десятичное число	Двоичный эквивалент
10^0	0 001
10^1	1 010
10^2	1 100 100

Решение. Подставив значения двоичных эквивалентов десятичных цифр и степеней основания в (2.11), получим

$$A=113=1\cdot10^2+1\cdot10^1+3\cdot10^0=001\cdot1100100+0001\cdot1010+$$
$$+0011\cdot0001=1110001_{(2)}.$$

Ответ: $1110001_{(2)}$.

Пример 2.8. Перевести двоичное число $A_{(2)}=11001{,}1_{(2)}$ в десятичную систему счисления ($q_2=10$).

Таблица эквивалентов:

Двоичное число	Десятичный эквивалент	Двоичное число	Десятичный эквивалент
0,1	$2^{-1}=0{,}5$	00100	$2^2=4$
00001	$2^0=1$	01000	$2^3=8$
00010	$2^1=2$	10000	$2^4=16$

Решение.

$$A=1\cdot16+1\cdot8+0\cdot4+0\cdot2+1\cdot1+1\cdot0{,}5=25{,}5.$$

Ответ: $A=25{,}5$.

Использование промежуточной системы счисления. Этот метод применяют при переводе из десятичной системы в двоичную и наоборот. В качестве промежуточной системы счисления можно использовать, например, восьмеричную систему.

Рассмотрим примеры, в которых перевод одного и того же числа в разные системы счисления осуществляется методом деления на основание новой системы. Запись будем вести в столбик, где справа от вертикальной черты записываются остатки деления на каждом шаге, а слева — целая часть частного.

Пример 2.9. Перевести десятичное число $A=121$ в двоичную систему счисления, используя в качестве промежуточной восьмеричную систему счисления.

Решение.

$$\begin{array}{r|l} \multicolumn{2}{c}{q_2=8} \\ \hline 121 & 1 \\ 15 & 7 \\ 1 & 1 \\ \hline \multicolumn{2}{c}{\text{3 шага}} \end{array} \qquad \begin{array}{r|l} \multicolumn{2}{c}{q_2=2} \\ \hline 121 & 1 \\ 60 & 0 \\ 30 & 0 \\ 15 & 1 \\ 7 & 1 \\ 3 & 1 \\ 1 & 1 \\ \hline \multicolumn{2}{c}{\text{7 шагов}} \end{array}$$

Ответ: $A=121=171_{(8)}=1111001_{(2)}$.

Сравнивая эти примеры, видим, что при переводе числа из десятичной системы в восьмеричную требуется в два с лишним раза меньше шагов, чем при переводе в двоичную систему. Если при этом учесть, что восьмеричная система связана с двоичной соотношением $8^k=(2^3)^k$, то перевод из восьмеричной системы в двоичную и наоборот можно осуществить простой заменой восьмеричных цифр их двоичными эквивалентами в соответствии с табл. 2.2.

Триада — *двоичный эквивалент восьмеричных цифр.*

Таблица 2.2

Восьмеричное число	Двоичный эквивалент	Восьмеричное число	Двоичный эквивалент
0	000	4	100
1	001	5	101
2	010	6	110
3	011	7	111

Пример 2.10. Перевести двоичное число $A_{(2)}=1011{,}0111_{(2)}$ в восьмеричную систему счисления.

Решение. Исходное число условно разбиваем на триады справа налево для целых чисел и слева направо для правильной дроби. Затем заменяем каждую триаду в соответствии с табл. 2.2.

$$A_{(2)} = 001 \quad 011, \quad 011 \quad 100$$

$$A_{(8)} = 1 \qquad 3, \qquad 3 \quad 4$$

Ответ: $A_{(8)}=13{,}34$.

Обобщая этот пример, можно сделать следующее заключение: в качестве промежуточных систем счисления целесообразно использовать системы с основанием $q=2^k$. При этом существенно упрощается преобразование информации из системы счисления с основанием $q=2^k$ в двоичную систему и наоборот. Преобразование фактически сводится к тому, что символы первоначальной информации, заданной в системе с основанием $q=2^k$, заменяются соответствующими двоичными эквивалентами (табл. 2.3). Обратное преобразование из двоичной системы в систему с основанием $q=2^k$ сводится к тому, что двоичный код разбивается на группы по k двоичных разрядов в каждой (начиная от младших разрядов для целых чисел или с первого разряда после запятой для правильных дробей); эти группы заменяются (диады, триады, пентады и т. д.) соответствующими символами исходной системы счисления.

Таблица 2.3

Десятичное число	Двоичный эквивалент для $q=2^4$	Десятичное число	Двоичный эквивалент для $q=2^4$
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

Системы счисления с основанием $q=2^k$ широко используют для записи программ решения задач, а также в машинах ЕС ЭВМ для ускорения выполнения арифметических операций.

§ 2.4. Разновидности двоичных систем счисления

Двоичная система счисления — *система счисления, в которой для изображения чисел используются два символа и веса разрядов в которой меняются по закону* $2^{\pm k}$ *(где* k *— произвольное целое число).*

Из определения следует, что для изображения чисел могут быть использованы не только символы 0, 1, но и символы 1, —1 или 0, —1.

Для удобства в дальнейшем символ —1 будем изображать как $\bar{1}$, а двоичную систему, в которой используются эти символы, — называть системой $(1, \bar{1})$.

Рассмотрим данную систему. Если для ряда (2.3) положить, что a_i принимает значения 1 или $\bar{1}$, то в случае $q=2$ число 99 запишется в виде $A=99=111\bar{1}\bar{1}\bar{1}1_{(2)}$.

Система $(1,\bar{1})$ отличается от естественной двоичной системы тем, что среди используемых символов отсутствует нуль. Это обстоятельство делает невозможным представление в системе $(1,\bar{1})$ некоторых чисел в виде конечного множества. В то же время существуют числа, которые не имеют единственного изображения, например число 1 может быть представлено в виде

$$1=1\underbrace{\bar{1}\,\bar{1}\ldots\bar{1}}_{k}=\underbrace{1\,0\,0\,0\ldots0}_{k}-\underbrace{1\,1\ldots1}_{k}=2^k-(2^k-2^0), \qquad (2.15)$$

где $k=1, 2, \ldots, n$.

Соотношение (2.15) выражает связь между естественной двоичной системой и системой $(1, \bar{1})$. Невозможность представить конечным образом некоторые целые и дробные числа, например $20=$ $=11\bar{1}\bar{1}1$, $111\ldots_{(2)}$, приводит к использованию бесконечных дробей, что обусловливает представление этих чисел в системе $(1, \bar{1})$ с определенной погрешностью.

Для получения конечного представления как четных, так и нечетных чисел в системе $(1, \bar{1})$ польским ученым И. Баньковским была предложена следующая запись чисел:

$$A_{(1,\bar{1})}=\sum_{i=1}^{n} a_i 2^i-2^{-1}, \qquad (2.16)$$

где $a_i=\{1, \bar{1}\}$.

Пример 2.11. Перевести число $A_{(2)}=100101$ в систему $(1, \bar{1})$.

Решение. Используя соотношение (2.15), перевод сводится к замене комбинаций 001 и 01 комбинациями соответственно $\bar{1}\bar{1}1$ и $1\bar{1}$.

Ответ: $A_{(2)}=11\bar{1}\bar{1}11\bar{1}$.

Для перевода чисел в систему $(1, \bar{1})$ по методу Баньковского необходимо учитывать следующее: в случае нечетного числа перевод осуществляют по правилу (2.15), а затем в разряд $i=-1$ записывают единицу; при переводе четного числа его сначала превращают в нечетное добавлением единицы в младший разряд и только после этого переводят в систему $(1, \bar{1})$ по правилу (2.15) как нечетное число. Затем к полученному результату в разряд $i=-1$ записывают $\bar{1}$.

Пример 2.12. Перевести в систему $(1, \bar{1})$ двоичное число $A_{(2)}=11000_{(2)}$.

Решение. В соответствии с правилом Баньковского это число превращаем в нечетное число 11001. После этого заменяем в изображении числа комбинацию 001 на комбинацию $1\bar{1}\bar{1}$ и приписываем в разряд после запятой цифру $\bar{1}$.

Ответ: $A_{(1,\bar{1})}=111\bar{1}\bar{1},\bar{1}$.

Избыточная система счисления с основанием q — *система, где для записи чисел используется количество символов большее, чем* q, например избыточная двоичная система с символами 0, 1, $\bar{1}$ или избыточная троичная система с символами $\bar{2}$, $\bar{1}$, 0, 1, 2.

Избыточная двоичная система связана с обычной двоичной системой соотношением

$$\underbrace{1\,1\,1\ldots1}_{k}=\sum_{i=1}^{k}2^i=2^{k+1}-2^1=\underbrace{1\,0\,0\ldots0\,\bar{1}}_{k}. \qquad (2.17)$$

Формула (2.17) позволяет осуществлять переход от одной системы к другой, например $A=11110001=1000\bar{1}0001$.

В избыточной двоичной системе одни и те же числа можно представить несколькими способами, т. е. $A=0{,}01110011=0{,}100\bar{1}0011=$ $=0{,}100\bar{1}010\bar{1}$. Этот пример показывает, что при переходе к избыточной системе можно уменьшить количество единиц в изображении числа.

Избыточность системы счисления, характеризуемая симметричностью символов, дает возможность в ряде случаев упростить выполнение арифметических действий. Например, избыточную двоичную систему счисления используют в некоторых алгоритмах ускорения операции умножения.

§ 2.5. Системы счисления с отрицательным основанием

Еще в работах К. Шеннона было показано, что в системе счисления с основанием $q<-1$ и символами $0, -1, \ldots, -(q-1)$ можно представить любое действительное число, для чего справедливо выражение (2.3). При этом если некоторое число представляется в системе с целочисленным отрицательным основанием, то такое представление будет единственным для всех чисел, кроме чисел, равных X:

$$X=(\pm 1)\frac{q^k}{-q+1}+rq^{k+1},$$

где q — основание системы $(q<0)$; r и k — любые целые числа.

В самом деле, для системы счисления с основанием $q=-2$ десятичное число $X={}^1/_3$ может быть представлено бесконечными дробями в виде

$$X_{(-2)}=\begin{cases} 0,\ 010101\ldots; \\ 1,\ 101010. \end{cases}$$

Двоичную систему счисления с основанием $q=-2$, в которой используются символы 0, 1, назовем минус-двоичной системой счисления. Эта система дает возможность представлять как положительные, так и отрицательные числа (табл. 2.4).

Таблица 2.4

Десятичное число	Двоичный эквивалент для $q=-2$	Десятичное число	Двоичный эквивалент для $q=-2$
0	000000	+10	001010
1	000001	−10	011110
−1	000011	+15	010011
+5	000101	−15	110001
−5	001111	+21	010101
		−21	111111

Из таблицы видно, что методы перевода десятичных чисел в систему счисления с основанием $q=-2$ аналогичны методам перевода, рассмотренным ранее, но при переводе десятичных чисел необходимо учитывать следующее: при использовании метода последовательного деления на основание новой системы все остатки от деления на каждом шаге должны быть положительными числами, которые не превышают абсолютного значения нового основания q. Это правило распространяется и для случая перевода правильных дробей методом последовательного умножения на основание q, где появляющиеся целые части дробей также должны быть положительными числами, значение которых меньше величины q.

Пример 2.13. Перевести десятичное число $A=21$ в минус-двоичную систему счисления методом деления на основание системы ($q_2=-2$).

Решение.

$$\begin{array}{r|l} 21 & 1 \\ -10 & 0 \\ 5 & 1 \\ -2 & 0 \\ \hline 1 & 1 \end{array}$$

Ответ: $A=21=10101_{(-2)}$.

В случае перевода правильной дроби (или дробной части смешанной дроби) необходимо, чтобы дробь на каждом шаге удовлетворяла требованию

$$\frac{|q|}{|q|+1} \leqslant A \leqslant \frac{1}{|q|+1} \quad \text{или} -\frac{2}{3} \leqslant A \leqslant \frac{1}{3}, \qquad (2.18)$$

где A — дробь, переводимая в систему счисления с основанием $q=-2$.

Если ограничение (2.18) не выполняется, тогда дробь A представляют в виде $A=1-\Delta$, где Δ должно лежать в указанных пределах.

Пример 2.14. Перевести десятичную дробь $A=0{,}625$ в минус-двоичную систему счисления с точностью до трех знаков после запятой.

Решение. Исходная дробь не удовлетворяет неравенству (2.18). Преобразуем ее к виду:

$$0{,}625=\boxed{1}-0{,}375$$

$$\begin{array}{r}
-0{,}375\\
\times\quad -2\\
\hline
0{,}750
\end{array}\quad \text{неравенство (2.18) не удовлетворяется;}$$

$$\begin{array}{r}
\boxed{1}-0{,}250\\
\times\quad -2\\
\hline
0{,}500
\end{array}\quad \text{неравенство (2.18) не удовлетворяется;}$$

$$\begin{array}{r}
\boxed{1}-0{,}500\\
\times\quad -2\\
\hline
1{,}000
\end{array}\quad \text{неравенство (2.18) удовлетворяется.}$$

Ответ: $0{,}625=1{,}111_{(-2)}$.

§ 2.6. Формы представления чисел

Число 0,028 можно записать так: $28\cdot10^{-3}$, или 0,03 (с округлением), или $2{,}8\cdot10^{-2}$ и т. д. Разнообразие форм в записи одного числа может послужить причиной затруднений для работы цифрового автомата. Во избежание этого нужно либо создать специальные алгоритмы распознавания числа, либо указывать каждый раз форму его записи. Второй путь проще.

Существует две формы записи чисел: естественная и нормальная.

При естественной форме число записывается в естественном натуральном виде, например 12560 — целое число, 0,003572 — правильная дробь, 4,89760 — неправильная дробь.

При нормальной форме запись одного числа может принимать разный вид в зависимости от ограничений, накладываемых на ее форму. Например, число 12560 может быть записано так: $12560=$ $=1{,}256\cdot10^4=0{,}1256\cdot10^5=125600\cdot10^{-1}$ и т. д.

Машинное (автоматное) изображение числа — *представление числа A в разрядной сетке цифрового автомата.*

Условно обозначим машинное изображение числа символом $[A]$. Тогда справедливо следующее соотношение:

$$A=[A]\,K_A,$$

где K_A — коэффициент, величина которого зависит от формы представления числа в автомате.

Представление чисел с фиксированной запятой (точкой). Естественная форма представления числа в цифровом автомате характеризуется тем, что положение его разрядов в машинном изображении остается всегда постоянным независимо от величины самого числа. Существует также другое название этой формы записи чисел — **представление чисел с фиксированной запятой (точкой).**

Чтобы упростить функционирование цифрового автомата, необходимо ограничить входную информацию какой-то одной областью чисел (на вход автомата желательно подавать либо целые числа, либо правильные дроби, либо любые числа), что позволит определить значения масштабного коэффициента K_A. Например, если на вход цифрового автомата поступают только правильные дроби, то

$$-1 < [A]_ф < 1, \tag{2.19}$$

где $[A]_ф$ — машинное изображение числа для формы представления с фиксированной запятой.

Тогда число A будет представлено в виде

$$A = [A]_ф K_A.$$

Выбор масштабного коэффициента K_A, удовлетворяющего условию (2.19), означает, что в машинном изображении запятая всегда стоит после целой части дроби, т. е. перед ее старшим разрядом. Следовательно, можно хранить только дробную часть числа (цифровую часть), а в разряде целой части писать дополнительную информацию.

Так как числа бывают положительные и отрицательные, то формат (разрядная сетка) машинного изображения разбивается на **знаковую часть** и **поле числа** (рис. 2.2, *а*). В знаковую часть записывается информация о знаке. Примем, что знак положительного числа «+» изображается символом 0, а знак отрицательного числа «—» изображается символом 1.

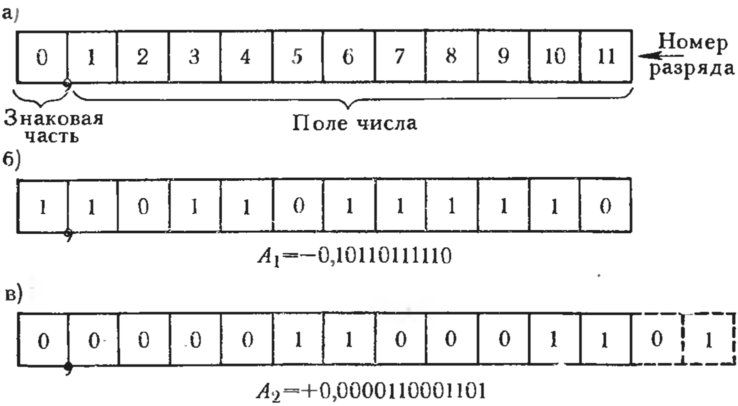


Рис. 2.2. Представление чисел в форме с фиксированной запятой

Если на вход цифрового автомата поступают целые числа, например, как в ЕС ЭВМ, то в разрядной сетке (в формате машинного изображения) один разряд отводится под знак числа, а последующие разряды образуют поле числа. Диапазон представимых чисел в этом случае от $-(2^n-1)$ до $+(2^n-1)$, где n — количество разрядов без знаковой части.

Задачу выбора масштабного коэффициента K_A усложняет стремление сохранить соответствие разрядов всех чисел, которыми оперирует цифровой автомат. Пусть имеется цифровой автомат с разрядной сеткой длиной 12 двоичных разрядов (рис. 2.2, *а*). Надо определить масштабный коэффициент для чисел $A_1 = -1011{,}0111110$ и $A_2 = 0{,}110001101$.

Для того чтобы выполнить условие (2.19), необходимо число A_1, большее по абсолютному значению, записать в виде $A_1 = -0{,}10110111110 \cdot 2^4$. Отсюда $[A_1]_{ф} = 1{,}10110111110$, что соответствует величине масштабного коэффициента $K_{A_1} = 2^4$. Число A_2 должно войти в разрядную сетку автомата с сохранением соответствия разрядов, т. е. $K_{A_1} = K_{A_2}$. Следовательно, $A_2 = +0{,}0000110001101 \cdot 2^4$, или $[A_2]_{ф} = 0{,}00001100011$. На рис. 2.2, *б*, *в* показаны изображения этих чисел в разрядной сетке автомата.

Из примера видно, что представление чисел в форме с фиксированной запятой может привести к погрешности представления. Так, для числа A_2 абсолютная погрешность представления оценивается величиной части числа, не уместившейся в разрядную сетку, т. е. величиной $0{,}0000000000001 \cdot 2^4$. В некоторых случаях очень малые числа представляются в машине изображением, называемым **машинным нулем**. Следовательно, ошибка представления зависит от правильности выбора масштабных коэффициентов. Вычисление последних должно производиться таким образом, чтобы исключить возможность появления в процессе функционирования автомата чисел, машинные изображения которых не удовлетворяют условию (2.19). Если в результате операции появляется число, по абсолютному значению большее единицы, то возникает **переполнение разрядной сетки** автомата, что нарушает нормальное функционирование цифрового автомата.

П р е д с т а в л е н и е ч и с е л в ф о р м е с п л а в а ю щ е й з а п я т о й. В нормальной форме число записывается следующим образом:

$$A_н = m_A p_A, \tag{2.20}$$

где m_A — мантисса числа A; p_A — порядок числа A (характеристика числа).

Как видно из ранее изложенного, такое представление чисел не однозначно. Чтобы избежать этого. обычно вводят некоторые ограничения. Наиболее распространенным и удобным для представления в ЭВМ ограничением является ограничение вида

$$q^{-1} \leqslant |m_A| < 1, \tag{2.21}$$

где q — основание системы счисления.

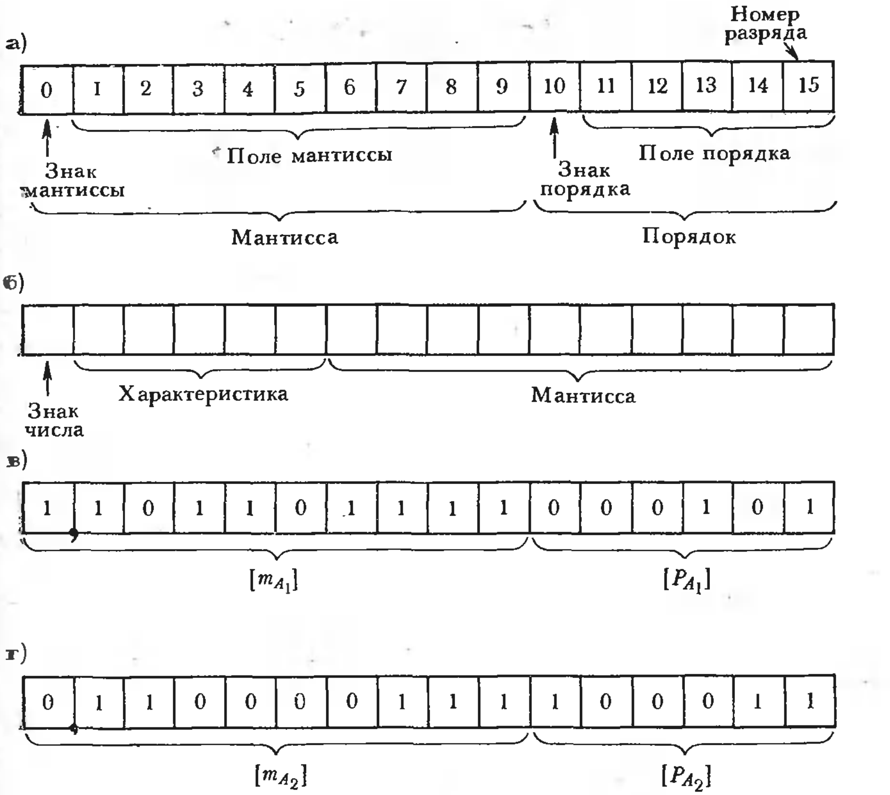


Рис. 2.3. Представление чисел в форме с плавающей запятой

Нормализованная форма представления чисел — *форма представления чисел, для которой справедливо условие* (2.21).

Поскольку в этом случае абсолютное значение мантиссы лежит в пределах от q^{-1} до $1-q^{-n}$, где n — количество разрядов для изображения мантиссы без знака, положение разрядов числа в его автоматном изображении не постоянно. Поэтому такую форму представления чисел называют также **формой представления с плавающей запятой.**

Формат машинного изображения числа с плавающей запятой должен содержать знаковые части и поля для мантиссы и порядка (рис. 2.3, *а*). Выделяются специальные разряды для изображения знака числа (мантиссы) и знака порядка или характеристики (рис. 2.3, *а*, *б*). Кодирование знаков остается таким же, как было указано ранее для чисел с фиксированной запятой.

Рассмотрим пример записи чисел в форме с плавающей запятой. Пусть в разрядную сетку цифрового автомата (рис. 2.3) необходимо записать двоичные числа $A_1 = -10110,1111$ и $A_2 =$

$=+0{,}000110010111$. Прежде всего эти числа необходимо записать в нормальной форме (рис. 2.3, *в*, *г*). Порядок чисел выбирают таким образом: чтобы для них выполнялось условие (2.21), т. е. $A_1=-0{,}101101111\cdot 2^{+5}$ и $A_2=+0{,}110010111\cdot 2^{-3}$, он должен быть записан в двоичной системе счисления. Так как система счисления для заданного автомата остается постоянной, то нет необходимости указывать ее основание, достаточно лишь представить показатель порядка (характеристику) числа.

Поскольку для изображения порядка выделено пять цифровых разрядов и один разряд для знака, их машинные изображения будут

$$[p_{A_1}]=0\ 001\ 01;$$

$$[p_{A_2}]=1\ 000\ 11.$$

Машинные изображения мантисс соответственно

$$[m_{A_1}]=1{,}101101111;$$

$$[m_{A_2}]=0{,}110010111.$$

Изображения чисел A_1 и A_2 в форме с плавающей запятой показаны на рис. 2.3, *в*, *г*.

§ 2.7. Погрешности представления чисел

Представление числовой информации в цифровом автомате, как правило, влечет за собой появление погрешностей (ошибок), величина которых зависит как от формы представления чисел, так и от длины разрядной сетки автомата.

Абсолютная погрешность представления — *разность между истинным значением входной величины A и ее значением, полученным из машинного изображения $A_{\text{м}}$*, т. е.

$$\Delta[A]=A-A_{\text{м}}.$$

Относительная погрешность представления — *величина*

$$\delta[A]=\Delta[A]/A_{\text{м}}. \tag{2.22}$$

Входные величины независимо от количества значащих цифр могут содержать грубые ошибки, возникающие из-за опечаток, ошибочных отсчетов показаний каких-либо приборов, некорректной постановки задачи или отсутствия более полной и точной информации. Например, часто принимают $\pi=3{,}14$. Однако эта величина может быть получена с более высокой точностью (800 знаков и более). Если принять, что точное значение $\pi=3{,}14159265$, то абсолютная погрешность равна $\Delta[A]=0{,}00159265$.

Часто некоторая величина в одной системе счисления имеет конечное значение, а в другой системе счисления становится бесконечной величиной, например, дробь $^1/_{10}$ имеет конечное десятичное представление, но, будучи переведена в двоичную систему счисления, становится бесконечной дробью $0{,}0001100110011\ldots$

Следовательно, при переводе чисел из одной системы счисления в другую неизбежно возникают погрешности, оценить которые нетрудно, если известны истинные значения входных чисел.

В соответствии с (2.19) числа изображаются в машине в виде

$$A_{q_1}=[A]K_A,$$

где величину масштабного коэффициента K_A выбирают так, что абсолютное значение машинного изображения числа A в системе счисления с основанием $q_1=2$ всегда меньше 1:

$$A_{q_1}=K_A[a_{-1}q_1^{-1}+a_{-2}q_1^{-2}+\ldots+a_{-n}q_1^{-n}+a_{-(n+1)}q^{-(n+1)}+\ldots].$$

Так как длина разрядной сетки автомата равна n двоичных разрядов после запятой, то абсолютная погрешность перевода десятичной информации в систему с основанием q_1 будет

$$\Delta[A]=a_{-(n+1)}q_1^{-(n+1)}+a_{-(n+2)}q_1^{-(n+2)}+\ldots=\sum_{i=-(n+1)}^{i=-\infty}a_iq_1^i. \quad (2.23)$$

Если $q_1=2$, то максимальное значение этой погрешности при $a_i=1$

$$\Delta[A]_{\max}=\sum_{i=-(n+1)}^{i=-\infty}1\cdot 2^i=2^{-n}\sum_{i=-1}^{i=-\infty}2^i=2^{-n}. \quad (2.24)$$

Из (2.24) следует, что максимальная погрешность перевода десятичной информации в двоичную не будет превышать единицы младшего разряда разрядной сетки автомата. Минимальная погрешность перевода равна нулю.

Таким образом, **усредненная абсолютная погрешность** перевода чисел в двоичную систему счисления

$$\Delta[A]=(0+2^{-n})/2=0{,}5\cdot 2^{-n}.$$

Для представления чисел в форме с фиксированной запятой абсолютное значение машинного изображения числа лежит в пределах

$$2^{-n}\leqslant|[A]_{\text{ф}}|\leqslant 1-2^{-n}. \quad (2.25)$$

Следовательно, относительные погрешности представления для минимального значения числа равны:

$$\delta[A]_{\text{ф}\,\min}=\frac{\Delta[A]}{[A]_{\text{ф}\,\max}}=\frac{0{,}5\cdot 2^{-n}}{1-2^{-n}}.$$

Для ЭВМ, как правило, $n=20\div 64$; поэтому $1\gg 2^{-n}$, откуда

$$\delta[A]_{\text{ф}\,\min}\approx 0{,}5\cdot 2^{-n}.$$

Аналогично для максимального значения:

$$\delta[A]_{\text{ф}\,\max}=\frac{\Delta[A]}{[A]_{\text{ф}\,\min}}=\frac{0{,}5\cdot 2^{-n}}{2^{-n}}=0{,}5=2^{-1}. \quad (2.26)$$

Из (2.26) следует, что погрешности представления малых чисел в форме с фиксированной запятой могут быть очень значительными.

Для представления чисел в форме с плавающей запятой абсолютное значение мантиссы лежит в пределах

$$2^{-1} \leqslant |[mA]_n| \leqslant 1-2^{-n}. \qquad (2.27)$$

Так как погрешность (2.24) относится только к величине мантиссы, то для нахождения погрешности представления числа в форме с плавающей запятой величину этой погрешности надо умножить на величину порядка числа p_A:

$$\delta [A]_{n\,\max} = \frac{0{,}5 \cdot 2^{-n} p_A}{2^{-1} p_A} = 2^{-n}$$

и

$$\delta [A]_{n\,\min} = \frac{0{,}5 \cdot 2^{-n}\, p_A}{(1-2^{-n})\, p_A} = 0{,}5 \cdot 2^{-n}, \qquad (2.28)$$

где n — количество разрядов для представления мантиссы числа.

Из (2.28) следует, что относительная точность представления чисел в форме с плавающей запятой почти не зависит от величины числа.

Задание для самоконтроля

1. Перевести десятичное число $A=121$ в двоичную систему счисления.

2. Перевести двоичное число $A=10001010111{,}01$ в десятичную систему счисления.

3. Перевести десятичное число $A=135{,}656$ в двоичную систему счисления с точностью до пяти знаков после запятой.

4. Сколько потребуется двоичных разрядов для изображения десятичного числа $A=10^{18}$?

5. Перевести двоичное число $A_{(2)}=10111011$ в десятичную систему счисления методом деления на основание.

6. Перевести восьмеричное число $A_{(8)}=345{,}766$ в двоичную систему счисления.

7. Записать десятичное число $A=79{,}346$ в двоично-десятичной форме.

8. Перевести десятичную дробь $A=63^9/64$ в двоичную систему счисления.

9. Перевести восьмеричную дробь $A_{(8)}=63^5/40$ в двоичную систему счисления.

10. Перевести восьмеричное число $A_{(8)}=326$ в троичную систему счисления.

11. Перевести восьмеричное число $A_{(8)}=15{,}647$ в двоичную систему счисления.

12. Перевести троичное число $A_{(3)}=1211$ в пятиричную систему счисления.

13. Для какой системы счисления с основанием $q_2=x$ справедливо равенство $121=441_{(x)}$?

14. Записать машинное изображение в форме с плавающей запятой для десятичного числа $A=-3{,}375$, если для мантиссы имеется шесть двоичных разрядов со знаком и для порядков — три двоичных разряда (со знаком).

15. Определить масштабные коэффициенты для чисел $A_{(2)}=-10110{,}111010001$ и $B_{(2)}=0{,}00111000110001$ при условии, что машинное изображение числа содержит десять двоичных разрядов со знаком.

16. Перевести двоичное число $A_{(2)}=0{,}011000100$ в систему $(1, \bar{1})$.

17. Перевести число $A_{(1,\bar{1})}=\bar{1}11\bar{1}\bar{1},\bar{1}$ из системы $(1, \bar{1})$ в двоичную систему счисления.

3 ГЛАВА

СЛОЖЕНИЕ ЧИСЕЛ НА ДВОИЧНЫХ СУММАТОРАХ

§ 3.1. Формальные правила двоичной арифметики

В выполнении арифметических действий всегда участвуют два числа или более. В результате арифметической операции появляется новое число

$$C = A \bigtriangledown B, \tag{3.1}$$

где ∇ — знак арифметического действия (сложение, вычитание, умножение, деление).

Операнд — *число, участвующее в арифметической операции, выполняемой цифровым автоматом.*

Так как цифровой автомат оперирует только машинными изображениями чисел, то последние выступают в качестве операндов. Следовательно, для машинных операций более правильно выражение (3.1) написать в виде

$$[C] = [A] \bigtriangledown [B], \tag{3.2}$$

где [ ] — обозначение машинных изображений операндов.

Рассмотрим формальные правила выполнения арифметических операций сложения и вычитания на уровне разрядов операндов.

На основе правил двоичной арифметики можно записать правила сложения двоичных цифр так, как показано в табл. 3.1, где a_i, b_i — разряды операндов A и B соответственно; c_i — результат сложения (сумма); Π_i — перенос из данного разряда в соседний старший.

Двоичный полусумматор — *устройство, выполняющее арифметические действия по правилам, указанным в табл.* 3.1.

Таблица 3.1

a_i	b_i	c_i	Π_i
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Появление единицы переноса при сложении двух разрядов несколько изменяет правила сложения двоичных цифр (табл. 3.2).

Таблица 3.2

a_i	b_i	Π_{i-1}	c_i	Π_i
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Обобщая вышеизложенное, можно сформулировать правила поразрядных действий при сложении операндов A и B:

$$a_i + b_i + \Pi_{i-1} = c_i + \Pi_i, \tag{3.3}$$

где a_i, b_i — i-й разряд 1-го и 2-го операндов соответственно; c_i — i-й разряд суммы; Π_{i-1} — перенос из $(i-1)$-го разряда; Π_i — перенос в $(i+1)$-й разряд (переносы принимают значения 0 или 1).

Двоичный сумматор — *устройство, выполняющее арифметические действия по правилам, указанным в табл.* 3.2.

Условные обозначения двоичных полусумматоров и сумматоров показаны на рис. 3.1.

На основе правил двоичной арифметики можно записать правила вычитания двоичных цифр так, как показано в табл. 3.3, где a_i — i-й разряд уменьшаемого; b_i — i-й разряд вычитаемого; c_i — i-й разряд разности; z_{i+1} — заем в старшем разряде.

Таблица 3.3

a_i	b_i	c_i	z_{i+1}	a_i	b_i	c_i	z_{i+1}
0	0	0	0	1	1	0	0
1	0	1	0	0	1	1	−1

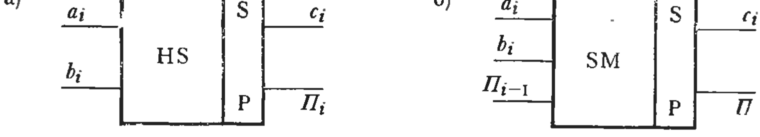


Рис. 3.1. Условное обозначение полусумматора (*а*) и двоичного сумматора (*б*)

Заем равносилен вычитанию единицы из старшего разряда. С учетом единицы займа из старшего соседнего разряда правила вычитания двоичных цифр можно записать так, как показано в табл. 3.4 (чтобы отличить заем от переноса, перед единицей поставлен знак минус).

Таблица 3.4

a_i	b_i	z_i	c_i	z_{i+1}
0	0	0	0	0
1	0	0	1	0
1	1	0	0	0
0	1	0	1	—1
0	0	—1	1	—1
1	0	—1	0	0
1	1	—1	1	—1
0	1	—1	0	—1

Если A — уменьшаемое (1-й операнд), B — вычитаемое (2-й операнд), то для поразрядных действий

$$a_i - b_i + z_i = c_i + z_{i+1}, \tag{3.4}$$

где a_i, b_i, c_i — соответственно i-е разряды уменьшаемого, вычитаемого и разности; z_i — заем из младшего i-го разряда; z_{i+1} — заем в старшем, $(i+1)$-м разряде.

Двоичный вычитатель — *устройство, выполняющее арифметическое действие по правилам, указанным в табл.* 3.4.

С точки зрения технической реализации всегда проще сложить два электрических сигнала, чем вычесть их друг из друга. Поэтому двоичные сумматоры являются основным устройством любой ЭВМ. Для того чтобы с их помощью выполнять такие арифметические действия, как вычитание, умножение, деление и др., необходимы соответствующие алгоритмы.

§ 3.2. Представление отрицательных чисел

Одним из способов выполнения операции вычитания с помощью двоичного сумматора является замена знака вычитаемого на противоположный и прибавление его к уменьшаемому:

$$A - B = A + (-B). \tag{3.5}$$

Этим операцию арифметического вычитания заменяют операцией алгебраического сложения. Последняя и становится основной операцией двоичного сумматора. Возникает вопрос: как представлять отрицательные числа в цифровом автомате?

Для машинного представления отрицательных чисел используют коды прямой, дополнительный, обратный.

Рассмотрим применение этих кодов для чисел, представленных в форме с фиксированной запятой.

Примечание. Для простоты изложения в дальнейшем будут рассматриваться числа по модулю, меньшему 1. Это существенно упрощает вычисление масштабных коэффициентов.

Прямой код числа $A=-0, a_1 a_2 ... a_n$ — *машинное изображение этого числа в виде* $[A]_{\text{пр}}=1, a_1 a_2 ... a_n$.

Из определения следует, что в прямом коде все цифровые разряды отрицательного числа остаются неизменными, а в знаковой части записывается единица. Например, если $A=-0{,}101110$, то $[A]_{\text{пр}}=1{,}101110$.

Положительное число в прямом коде не меняет своего изображения. Например, если $A=0{,}110101$, то $[A]_{\text{пр}}=0{,}110101$.

В прямом коде в разрядную сетку цифрового автомата можно записать следующее максимальное по абсолютному значению число:

$$[A]_{\text{пр max}}=0,\ 1\ldots 1=1-2^{-n},$$

где n — количество разрядов разрядной сетки цифрового автомата.

Очевидно, что диапазон изменения машинных изображений для прямого кода лежит в пределах

$$-(1-2)^{-n}\leqslant [A]_{\text{пр}}\leqslant (1-2^{-n}).$$

Ранее (см. гл. 2) прямой код был использован для записи чисел в разрядной сетке цифрового автомата.

Правила преобразования чисел в прямой код можно сформулировать так:

$$[A]_{\text{пр}}=\begin{cases} A, & \text{если} \quad A\geqslant 0; \\ 1+|A|, & \text{если} \quad A<0. \end{cases} \tag{3.6}$$

Дополнительный код числа $A=-0, a_1 ... a_n$ — *такое машинное изображение этого числа* $[A]_{\text{д}}=1, \bar{a}_1 \bar{a}_2 ... \bar{a}_n$, *для которого* $\bar{a}_i=0$, *если* $a_i=1$, и $\bar{a}_i=1$, *если* $a_i=0$, *за исключением последнего значащего разряда, для которого* $\bar{a}_k=1$ *при* $a_k=1$.

Например, число $A=-0{,}101110$ запишется в дополнительном коде так: $[A]_{\text{д}}=1{,}010010$.

Дополнительный код является математическим дополнением до основания системы счисления:

$$|A|+[A]_{\text{д}}=q, \tag{3.7}$$

где $|A|$ — абсолютное значение числа A.

Так как положительные числа не меняют своего изображения в дополнительном коде, то правила преобразования в дополнительный код можно записать следующим образом:

$$[A]_{\text{д}}=\begin{cases} A, & \text{если} \quad A\geqslant 0; \\ q+A, & \text{если} \quad A<0. \end{cases} \tag{3.8}$$

Максимальное дополнительное число, представляемое при этом, равно $1-2^{-n}$.

Наибольшее отрицательное число, которое можно записать в дополнительном коде, определим следующим образом. Предположим, что наибольшее отрицательное число $A'=-0{,}11...11$. Тогда изображение этого числа в дополнительном коде $[A']_д=1{,}00...01$. Если к числу A' добавить единицу в самый младший разряд, то в результате получим число $-1{,}00...0$. Преобразовав это число по формальным правилам, получим $[A]_{д\,min}=1{,}00...0$.

Следовательно, диапазон изменения машинных изображений чисел для формы представления с запятой, фиксированной перед старшим разрядом, в дополнительном коде будет $-1\leqslant[A]_д\leqslant(1-2^{-n})$.

Как отмечалось ранее, для ЕС ЭВМ машинные изображения чисел — всегда целые числа. При этом наибольшее положительное число состоит из целой части, все разряды которой равны единице, и знакового разряда, равного нулю (например, в случае 16 двоичных разрядов (два байта) максимальное положительное число имеет вид 0111111111111111, т. е. равно $2^{15}-1$); наибольшее отрицательное число состоит из целой части, все разряды которой равны нулю, и знакового разряда, равного единице, т. е. имеет вид 1000000000000000. В этом случае говорят о форме представления чисел с фиксированной точкой. Таким образом, соотношение (3.7) для представления целых чисел в дополнительном коде принимает следующий вид:

$$|A|+[A]_д=q^{k+1}, \tag{3.9}$$

где k — количество разрядов в целой части машинного изображения числа $(k=\overline{0,k})$.

Следовательно, формула (3.7) — частный случай формулы (3.9) при $k=0$.

Обратный код числа $A=-0,\ a_1a_2...a_n$ — *такое машинное изображение этого числа* $[A]_{об}=1,\ \overline{\overline{a}}_1\overline{\overline{a}}_2...\overline{\overline{a}}_n$, *для которого* $\overline{\overline{a}}_i=0$, *если* $a_i=1$, *и* $\overline{\overline{a}}_i=1$, *если* $a_i=0$.

Из определения следует, что обратный код двоичного числа является инверсным изображением самого числа, в котором все разряды исходного числа принимают инверсное (обратное) значение, т. е. все нули заменяются на единицы, а все единицы — на нули, например если $A=-0{,}101110$, то $[A]_{об}=1{,}010001$.

Следовательно, для обратного кода чисел, представленных в форме с запятой, фиксированной перед старшим разрядом, справедливо соотношение

$$|A|+[A]_{об}=q-q^{-n}, \tag{3.10}$$

где $|A|$ — абсолютная величина числа A; n — количество разрядов после запятой в изображении числа.

Правила преобразования чисел в обратный код можно сформулировать следующим образом:

$$[A]_{об}=\begin{cases} A, & \text{если } A\geqslant 0; \\ q-q^{-n}+A, & \text{если } A<0. \end{cases} \tag{3.11}$$

Сравнив (3.7) и (3.10), видим, что

$$[A]_д=[A]_{об}+q^{-n}. \tag{3.12}$$

Соотношение (3.12) используют для получения дополнительного кода отрицательных чисел следующим образом: сначала инвертируется цифровая часть исходного числа, в результате получается его обратный код; затем добавляется единица в младший разряд цифровой части числа и тем самым получается дополнительный код этого изображения.

Пример 3.1. Найти обратный и дополнительный коды числа $A=-0{,}111000$.

Решение. Используя определение обратного кода, получим $[A]_{об}=1{,}000111$.

Для нахождения дополнительного кода числа добавим единицу в младший разряд его изображения:

$$\begin{array}{r} +\begin{array}{r} 1{,}000111 \\ 1 \end{array} \\ \hline [A]_д=1{,}001000 \end{array}$$

Ответ: $[A]_{об}=1{,}000111$; $[A]_д=1{,}001000$.

В обратном коде можно изображать максимальное положительное число $[A]_{об\,max}=0{,}111\ldots1=1-2^{-n}$ и наибольшее отрицательное число $[A]_{об\,min}=-0{,}111\ldots1=-(1-2^{-n})$, записываемое в виде $1{,}000\ldots0$.

При проектировании цифровых автоматов необходимо учитывать неоднозначное изображение нуля в обратном коде:

$$\begin{cases} +0-0{,}00\ldots\,0, \\ -0-1{,}11\ldots\,1. \end{cases}$$

Использование различных способов изображения отрицательных чисел в цифровом автомате обусловливает целый ряд особенностей выполнения операции алгебраического сложения двоичных чисел.

§ 3.3. Сложение чисел, представленных в форме с фиксированной запятой, на двоичном сумматоре прямого кода

Двоичный сумматор прямого кода (ДСПК) — *сумматор, в котором отсутствует цепь поразрядного переноса между старшим цифровым и знаковым разрядами* (рис. 3.2, *а*).

В соответствии с определением на ДСПК можно складывать только числа, имеющие одинаковые знаки, т. е. такой сумматор не

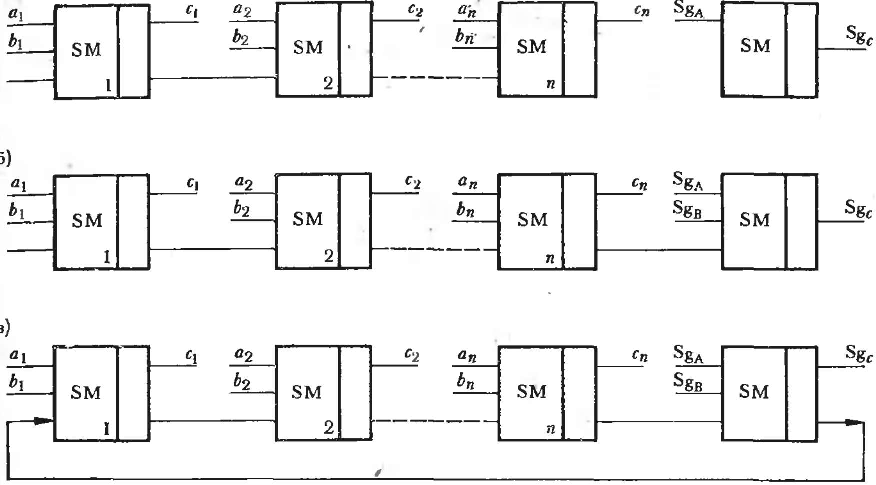


Рис. 3.2. Структурные схемы сумматоров:
а — прямого кода; *б* — дополнительного кода; *в* — обратного кода

может выполнять операцию алгебраического сложения. В самом деле, пусть заданы операнды

$$[A]_{\text{пр}} = Sg_A, a_1 a_2 \ldots a_n;$$
$$[B]_{\text{пр}} = Sg_B, b_1 b_2 \ldots b_n,$$

где Sg_A, Sg_B — соответственно содержимое знаковых разрядов изображений для A и B (символ происходит от английского слова sign — знак); a_i, b_i — цифровые разряды изображений.

Если $Sg_A = Sg_B$, то сумма чисел будет иметь знак любого из слагаемых, а цифровая часть результата получится после сложения цифровых частей операндов.

Пример 3.2. Сложить числа $A=0{,}1011$, $B=0{,}0100$ на сумматоре прямого кода.

Решение.

$$\begin{array}{ll} [A]_{\text{пр}} = 0{,}1011 & Sg_A = 0, \\ + & \\ [B]_{\text{пр}} = 0{,}0100 & Sg_B = 0, \\ \hline [C]_{\text{пр}} = 0{,}1111 & Sg_C = 0. \end{array}$$

Ответ: $C=0{,}1111$.

Пример 3.3. Сложить числа $A=-0{,}0101$, $B=-0{,}1001$ на сумматоре прямого кода.

Решение.

$$\begin{array}{lll} [A]_{\text{пр}} = 1{,}0101; & Sg_A = 1 & ,0101 \\ & & + \\ [B]_{\text{пр}} = 1{,}1001, & Sg_B = 1 & ,1001 \\ \cline{3-3} & Sg_C = 1 & ,1110 \end{array}$$

Ответ: $[C]_{\text{пр}}=1{,}1110$.

При сложении чисел на ДСПК возможен случай, когда абсолютное значение суммы операндов превышает единицу. Тогда имеет место переполнение разрядной сетки автомата. Признаком переполнения будет наличие единицы переноса из старшего разряда цифровой части сумматора. В этом случае должен вырабатываться сигнал переполнения $\varphi=1$, по которому происходит автоматический останов машины и корректировка масштабных коэффициентов с таким расчетом, чтобы избежать появления переполнения.

§ 3.4. Сложение чисел, представленных в форме с фиксированной запятой, на двоичном сумматоре дополнительного кода

Двоичный сумматор дополнительного кода (ДСДК) — *сумматор, оперирующий изображениями чисел в дополнительном коде.*

Характерной особенностью ДСДК является наличие цепи поразрядного переноса из старшего разряда цифровой части в знаковый разряд (рис. 3.2, *б*).

Для того чтобы определить правила сложения чисел на ДСДК, необходимо доказать следующую теорему.

Теорема: *сумма дополнительных кодов чисел есть дополнительный код результата.*

При доказательстве этой теоремы предполагаем, что числа представлены в форме с фиксированной запятой, стоящей перед старшим разрядом. Рассмотрим следующие случаи:

1) $A>0$, $B>0$, $A+B<1$.

Так как $[A]_д=A$, $[B]_д=B$, то $[A]_д+[B]_д=A+B=[A+B]_д$ — результат положительный.

2) $A<0$, $B>0$, $|A|>B$.

Здесь $[A]_д=q+A$, $[B]_д=B$. Тогда $[A]_д+[B]_д=q+A+B=[A+B]_д$ — результат отрицательный.

3) $A<0$, $B>0$, $|A|<B$.

Здесь $[A]_д=q+A$; $[B]_д=B$. Тогда $[A]_д+[B]_д=A+q+B$. Так как значение этой суммы больше q, то появляется единица переноса из знакового разряда, что равносильно изъятию из суммы q единиц, т. е. результат равен $[A]_д+[B]_д=A+B$.

4) $A<0$, $B<0$, $|A+B|<1$.

Здесь $[A]_д=q+A$; $[B]_д=q+B$. Тогда $[A]_д+[B]_д=q+A+q+B=$ $=[A+B]_д$ — результат отрицательный (здесь появляется единица переноса из знакового разряда).

Таким образом, теорема справедлива для всех случаев, в которых не возникает переполнение разрядной сетки, что позволяет складывать машинные изображения чисел по правилам двоичной арифметики (см. табл. 3.2).

Пример 3.4. Найти сумму чисел $A=0{,}1010$, $B=0{,}0100$, используя сумматор дополнительного кода.

Решение. Складываются машинные изображения этих чисел:

$$\begin{array}{r} [A]_д = 0,1010 \\ + \\ [B]_д = 0,0100 \\ \hline [C]_д = 0,1110. \end{array}$$

Ответ: $C=0{,}1110$.

Пример 3.5. Найти сумму чисел $A=-0{,}1011$, $B=0{,}0100$ на сумматоре дополнительного кода.

Решение. Складываются машинные изображения этих чисел:

$$\begin{array}{r} [A]_д = 1,0101 \\ + \\ [B]_д = 0,0100 \\ \hline [C]_д = 1,1001. \end{array}$$

Ответ: $C=-0{,}0111$.

Пример 3.6. Найти сумму чисел $A=0{,}1011$, $B=-0{,}0100$ на сумматоре дополнительного кода.

Решение. Складываются машинные изображения этих чисел:

$$\begin{array}{r} [A]_д = 0,1011 \\ + \\ [B]_д = 1,1100 \\ \hline [C]_д = 0,0111. \end{array}$$

Ответ: $C=0{,}0111$.

§ 3.5. Сложение чисел, представленных в форме с фиксированной запятой, на двоичном сумматоре обратного кода

Двоичный сумматор обратного кода (ДСОК) — *сумматор, оперирующий изображениями чисел в обратном коде.*

Характерной особенностью ДСОК является наличие цепи кругового или циклического переноса из знакового разряда в младший разряд цифровой части (рис. 3.2, *в*).

Для вывода правил сложения чисел на ДСОК необходимо доказать следующую теорему.

Теорема: *сумма обратных кодов чисел есть обратный код результата.*

Доказательство этой теоремы приведем с помощью приема, который был использован в § 3.4 при тех же ограничениях.

Рассмотрим следующие основные случаи:

1) $A>0$, $B>0$, $A+B<1$.

Тогда $[A]_{об}+[B]_{об}=[A+B]_{об}=A+B$.

2) $A<0$, $B>0$, $|A|>B$.

Здесь $[A]_{об}=q-q^{-n}+A$, $[B]_{об}=B$. Тогда $[A]_{об}+[B]_{об}=q-q^{-n}+A+B=[A+B]_{об}$, так как результат отрицательный.

3) $A<0$, $B>0$, $A<B$.

Здесь $[A]_{об}=q-q^{-n}+A$. Тогда $[A]_{об}+[B]_{об}=q-q^{-n}+A+B$. Так как сумма $(A+B)$ положительная, то правая часть этого выражения становится больше q, что вызывает появление единицы переноса из знакового разряда. Поскольку в ДСОК существует цепь переноса из знакового разряда в младший разряд (величина переноса из знакового разряда равна $q-q^{-n}$), то $[A]_{об}+[B]_{об}=A+B$; результат положительный.

4) $A<0$, $B<0$, $|A+B|<1$.

Здесь $[A]_{об}=q-q^{-n}+A$; $[B]_{об}=q-q^{-n}+B$. Следовательно, $[A]_{об}+[B]_{об}=q-q^{-n}+A+q-q^{-n}+B$. Здесь появляется единица переноса, что равносильно изъятию из суммы величины $q-q^{-n}$, т. е. $[A]_{об}+[B]_{об}=q-q^{-n}+A+B=[A+B]_{об}$.

5) $|A|=|B|$, $A>0$, $B<0$.

Тогда $[B]_{об}=q-q^{-n}+B$. Следовательно, $[A]_{об}+[B]_{об}=A+q-q^{-n}+B=q-q^{-n}$ — результат указывает на то, что сумма равна нулю (получили одно из изображений нуля в обратном коде).

Таким образом показано, что на ДСОК машинные изображения чисел складываются также по правилам, приведенным в табл. 3.2.

Пример 3.7. Найти сумму чисел $A=0{,}0101$ и $B=0{,}0111$, используя сумматор обратного кода.

Решение. Складываются машинные изображения этих чисел:

$$\begin{array}{r} [A]_{об}=0{,}0101 \\ + \\ [B]_{об}=0{,}0111 \\ \hline C=0{,}1100. \end{array}$$

Ответ: $C=0{,}1100$.

Пример 3.8. Найти сумму чисел $A=-0{,}0101$ и $B=0{,}0111$, используя ДСОК.

Решение. Складываются машинные изображения этих чисел:

$$\begin{array}{r} [A]_{об}=1{,}1010 \\ + \\ [B]_{об}=0{,}0111 \\ \hline 0{,}0001 \\ + \longrightarrow 1 \\ \hline C=0{,}0010. \end{array}$$

Ответ: $C=0{,}0010$.

Пример 3.9. Найти сумму чисел $A=0{,}0101$ и $B=-0{,}0111$, используя ДСОК.

Решение. Складываются машинные изображения этих чисел:

$$\begin{array}{r} [A]_{об}=0{,}0101 \\ + \\ [B]_{об}=1{,}1000 \\ \hline [C]_{об}=1{,}1101. \end{array}$$

Ответ: $C=-0{,}0010$.

Пример 3.10. Найти сумму чисел $A=-0{,}0101$ и $B=-0{,}1000$, используя ДСОК.

Решение. Складываются машинные изображения этих чисел:

$$\begin{array}{r} [A]_{об}=1{,}1010 \\ + \\ [B]_{об}=1{,}0111 \\ \hline 1{,}0001 \\ + \quad 1 \\ \hline [C]_{об}=1{,}0010. \end{array}$$

Ответ: $C=-0{,}1101$.

Примечание. В дальнейшем для упрощения записи передача циклического переноса будет осуществляться сразу при получении результата и отдельно фиксироваться не будет.

§ 3.6. Переполнение разрядной сетки

Как уже указывалось ранее, при сложении чисел одинакового знака, представленных в форме с фиксированной запятой, может возникнуть переполнение разрядной сетки.

Признаком переполнения разрядной сетки сумматора прямого кода является появление единицы переноса из старшего разряда цифровой части числа.

Примеры.

1. $A=0{,}1010$ и $B=0{,}1101$.

$$\begin{array}{r} [A]_{\text{пр}} = 0{,}1010 \\ + \\ [B]_{\text{пр}} = 0{,}1101 \\ \hline [C]_{\text{пр}} \neq 0{,}0111 \end{array}$$

1 — единица переноса

2. $A=-0{,}1100$ и $B=-0{,}1010$.

$$\begin{array}{r} [A]_{\text{пр}} = 1{,}1100 \\ + \\ [B]_{\text{пр}} = 1{,}1010 \\ \hline [C]_{\text{пр}} \neq 1{,}0110 \end{array}$$

1 — единица переноса.

Признаком переполнения разрядной сетки сумматора дополнительного кода при сложении положительных чисел является отрицательный знак результата, а при сложении отрицательных чисел — положительный знак результата.

Примеры.

3. $A=0{,}1011$ и $B=0{,}1010$.

$$\begin{array}{r} [A]_{\text{д}} = 0{,}1011 \\ + \\ [B]_{\text{д}} = 0{,}1010 \\ \hline [C]_{\text{д}} \neq 1{,}0101. \end{array}$$

4. $A=-0{,}1011$ и $B=-0{,}1001$.

$$\begin{array}{r} [A]_{\text{д}} = 1{,}0101 \\ + \\ [B]_{\text{д}} = 1{,}0111 \\ \hline [C]_{\text{д}} \neq 0{,}1100. \end{array}$$

Признаком переполнения разрядной сетки сумматора обратного кода является знак результата, противоположный знакам операндов.

Примеры.

5. $A=0{,}0111$ и $B=0{,}1101$.

$$\begin{array}{r} [A]_{об}=0{,}0111 \\ + \\ \underline{[B]_{об}=0{,}1101} \\ [C]_{об}\neq 1{,}0100. \end{array}$$

6. $A=-0{,}0110$ и $B=-0{,}1101$.

$$\begin{array}{r} [A]_{об}=1{,}1001 \\ + \\ \underline{[B]_{об}=1{,}0010} \\ [C]_{об}\neq 0{,}1100. \end{array}$$

Для обнаружения переполнения разрядной сетки в составе цифрового автомата должны быть предусмотрены аппаратные средства, автоматически вырабатывающие признак переполнения — сигнал φ.

Чтобы обнаружить переполнение разрядной сетки ДСОК и ДСДК, вводится вспомогательный разряд в знаковую часть изображения числа (рис. 3.3), который называют **разрядом переполнения**. Такое представление числа называется **модифицированным**.

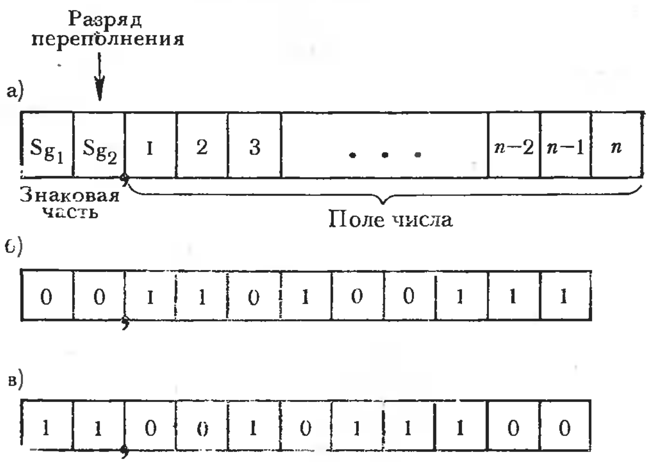


Рис. 3.3. Представление чисел в модифицированном коде:
а — разрядная сетка; *б* — положительное число; *в* — отрицательное число

Тогда в случае появления переполнения сигнал φ вырабатывается при условии

$$\varphi=1, \quad \text{если} \quad \begin{cases} Sg_1 \wedge \overline{Sg}_2 = 1; \\ \overline{Sg}_1 \wedge Sg_2 = 1, \end{cases} \tag{3.13}$$

в остальных случаях $\varphi=0$. Это подтверждается следующими примерами (для иллюстрации взяты числа из примеров § 3.6).

Примечание. Здесь и в дальнейшем тексте символ & означает логическую функцию И, а черта над символом $\overline{S}_g$ — функцию НЕ (см. § 9.1).

Примеры.

3а. $[A]^{м}_{д} = 00{,}1011$ — модифицированное изображение операнда A;
$+$
$[B]^{м}_{д} = 00{,}1010$ — модифицированное изображение операнда B;

$[C]^{м}_{д} = 01{,}0101$ — 01 — признак переполнения в знаковых разрядах, $\varphi=1$.

4а. $[A]^{м}_{д} = 11{,}0101$ — модифицированное изображение операнда A;
$+$
$[B]^{м}_{д} = 11{,}0111$ — модифицированное изображение операнда B;

$[C]^{м}_{д} = 10{,}1100$ — 10 — признак переполнения в знаковых разрядах, $\varphi=1$.

5а. $[A]^{м}_{об} = 00{,}0111$ — модифицированное изображение операнда A;
$+$
$[B]^{м}_{об} = 00{,}1101$ — модифицированное изображение операнда B;

$[C]^{м}_{об} = 01{,}0100$ — 01, — признак переполнения в знаковых разрядах, $\varphi=1$.

6а. $[A]^{м}_{об} = 11{,}1001$ — модифицированное изображение операнда A;
$+$
$[B]^{м}_{об} = 11{,}0010$ — модифицированное изображение операнда B;

$[C]^{м}_{об} = 10{,}1100$ — 10 — признак переполнения в знаковых разрядах, $\varphi=1$.

§ 3.7. Особенности сложения чисел, представленных в форме с плавающей запятой

Как было указано ранее, числа, представленные в форме с плавающей запятой, изображаются двумя частями — мантиссой и порядком.

При операции алгебраического сложения действия, выполняемые над мантиссами и порядками, различны. Следовательно, в цифровом автомате должны быть два раздельных устройства для обработки мантисс и для обработки порядков.

Поскольку для чисел с плавающей запятой справедливо условие (2.21), это означает, что всякий результат, не удовлетворяющий этому условию, должен быть приведен в соответствие с формулой (2.21). Такую операцию называют **нормализацией числа.**

Операция нормализации числа состоит из проверки выполнимости условия (2.21) и сдвига изображения мантиссы в ту или иную сторону. Сдвиги могут осуществляться на один разряд и более в левую или правую сторону в пределах разрядной сетки машины.

Простой сдвиг — *операция, выполняемая по следующим правилам:*

Исходная комбинация	Сдвинутая влево на один разряд	Сдвинутая вправо на один разряд
$0,\ a_1a_2\ldots a_n$	$a_1,\ a_2a_3\ldots a_n0$	$0,\ 0a_1\ldots a_{n-1}$
$1,\ a_1a_2\ldots a_n$	$a_1,\ a_2a_3\ldots a_n\alpha$	$0,\ 1a_1\ldots a_{n-1}$

Модифицированный сдвиг — *операция над модифицированными изображениями, выполняемая следующим образом:*

Исходная комбинация	Сдвинутая влево на один разряд	Сдвинутая вправо на один разряд
$00,\ a_1a_2\ldots a_n$	$0a_1,\ a_2a_3\ldots a_n0$	$00,\ 0a_1a_2\ldots a_{n-1}$
$01,\ a_1a_2\ldots a_n$	$1a_1,\ a_2a_3\ldots a_n0$	$00,\ 1a_1a_2\ldots a_{n-1}$
$10,\ a_1a_2\ldots a_n$	$0a_1,\ a_2a_3\ldots a_n\alpha$	$11,\ 0a_1a_2\ldots a_{n-1}$
$11,\ a_1a_2\ldots a_n$	$1a_1,\ a_2a_3\ldots a_n\alpha$	$11,\ 1a_1a_2\ldots a_{n-1}$

Примечание. Величина α зависит от кода: для дополнительного кода $\alpha=0$, для обратного кода $\alpha=1$.

Нарушение нормализации числа — *невыполнение условия (2.21).*

Так как условие (2.21) содержит два неравенства, то может быть нарушение справа и слева.

Признаком нарушения нормализации числа справа γ *(когда величина результата равна или превышает единицу) является наличие разноименных комбинаций в знаковых разрядах сумматора, т. е.*

$$\gamma=1, \quad \text{если} \begin{cases} Sg_1\wedge\overline{S}g_2=1; \\ \overline{S}g_1\wedge Sg_2=1 \end{cases} \tag{3.14}$$

(в остальных случаях $\gamma=0$),
где γ — признак нарушения нормализации числа справа, указывающий на необходимость сдвига числа вправо на один разряд.

Признаком нарушения нормализации числа слева δ *(когда результат по абсолютной величине оказывается меньше* $1/q$*) является наличие одинаковых комбинаций в разряде переполнения и старшем разряде цифровой части сумматора* (P_1):

$$\delta=1, \quad \text{если} \begin{cases} Sg_2\wedge P_1=1; \\ \overline{S}g_2\wedge\overline{P_1}=1 \end{cases} \tag{3.15}$$

(в остальных случаях $\delta=0$),
где δ — признак нарушения нормализации, указывающий на необходимость сдвига числа влево на один разряд.

Таким образом, операция нормализации числа будет состоять из совокупности сдвигов и проверки наличия признаков нарушения γ и δ.

Рассмотрим сложение чисел $A=m_Ap_A$ и $B=m_Bp_B$, имеющих одинаковый порядок $p_A=p_B$. Обе мантиссы удовлетворяют условию нормализации.

Сложение мантисс осуществляется на соответствующем сумматоре по правилам, изложенным ранее для чисел, представленных в форме с фиксированной запятой. Если после сложения мантисса результата удовлетворяет условию нормализации (т. е. $\delta=0$, $\gamma=0$), то к этому результату приписывается порядок любого из операндов. В противном случае происходит нормализация числа.

Пример 3.11. Найти сумму чисел $A=0{,}1000\cdot 2^{-3}$ и $B=-0{,}1011\cdot 2^{-3}$.

Мантиссы и порядок обрабатываются на сумматорах дополнительного кода (шесть разрядов для мантиссы и четыре разряда для порядка).

Решение. Сначала записываются машинные изображения операндов:

$$[m_A]_д = 00{,}1000; \qquad [p_A]_д = 1{,}101;$$

$$[m_B]_д = 11{,}0101; \qquad [p_B]_д = 1{,}101,$$

а затем — мантиссы:

$$\begin{array}{r} 00{,}1000 \\ +\ \ 11{,}0101 \\ \hline [m_C]_д = 11{,}1101. \end{array}$$

Здесь $Sg_2 \& P_1=1$, т. е. $\delta=1$, $\gamma=0$, значит, необходим сдвиг мантиссы влево на разряд:

$$[m'_C]_д = 11{,}1010, \ (\delta = 1, \ \gamma = 0).$$

Одновременно со сдвигом влево нужна коррекция порядка, т. е. уменьшение его величины на единицу (что равносильно прибавлению кода 1,111):

$$\begin{array}{r} [p_C]_д = 1{,}101 \\ +\,1{,}111 \\ \hline [p'_C]_д = 1{,}100. \end{array}$$

Так как после сдвига снова $\delta=1$, то осуществляется еще раз сдвиг и коррекция порядка:

$$[m''_C]_д = 11{,}0100, \ (\delta = 0, \ \gamma = 0) \qquad \begin{array}{r} [p'_C]_д = 1{,}100 \\ +\,1{,}111 \\ \hline [p''_C]_д = 1{,}011. \end{array}$$

Так будет продолжаться до тех пор, пока величина δ не станет равной нулю. Следовательно, $[m''_C]_д$ удовлетворяет условию нормализации и результат равен

$$[m''_C]_д = 11{,}0100;$$

$$[p''_C]_д = 1{,}011.$$

Ответ: $C=-0{,}1100\cdot 2^{-5}$.

Пример 3.12. Найти сумму чисел $A=-0{,}1100\cdot 2^{4}$ и $B=-0{,}1000\cdot 2^{4}$.

Рассмотрим этот пример для сумматоров обратного кода (шесть разрядов для мантиссы и четыре разряда для порядка).

Решение. Машинные изображения операндов записываются в следующем виде:

$$[m_A]_{об} = 11{,}0011; \quad [p_A] = [p_B]_{об} = 0{,}100.$$

$$[m_B]_{об} = 11{,}0111.$$

Затем складываются мантиссы:

$$\begin{array}{r} 11{,}0011 \\ +\ \ 11{,}0111 \\ \hline [m_C]_{об} = 10{,}1011 \end{array} \ (\delta = 0, \ \gamma = 1).$$

Здесь произошло нарушение нормализации справа и требуется модифицированный сдвиг мантиссы результата вправо на один разряд:

$$[m'_C]_{об} = 11,0101 \ (\delta = 0, \ \gamma = 0).$$

Одновременно со сдвигом производится коррекция порядка результата на величину $+0,001$, или $[p'_C]_{об} = 0,100 + 0,001 = 0,101$, в результате получается окончательный результат.

Ответ: $C = -0,1010 \cdot 2^{+5}$.

Рассмотрим наиболее общий случай сложения чисел, представленных в форме с плавающей запятой, когда их порядки не равны друг другу, т. е. $p_A \neq p_B$. Для операции сложения чисел необходимым условием является соответствие разрядов операндов друг другу. Значит, прежде всего нужно уравнять порядки, что, естественно, повлечет за собой временное нарушение нормализации одного из слагаемых. Выравнивание порядков означает, что порядок меньшего числа надо увеличить на величину $\Delta P = |p_A - p_B|$, что означает сдвиг мантиссы меньшего числа вправо на количество разрядов, равное ΔP.

Следовательно, цифровой автомат должен самостоятельно определять, какой из двух операндов меньший. На это укажет знак разности $p_A - p_B$: положительный знак будет при $p_A \geqslant p_B$, а отрицательный — при $p_A < p_B$.

Операции сложения и вычитания чисел в форме с плавающей запятой осуществляются во всех ныне действующих машинах по изложенным выше правилам.

Пример 3.13. Сложить числа $A = 0,1011 \cdot 2^{-2}$ и $B = -0,1001 \cdot 2^{-3}$.

Мантисса и порядок обрабатываются на сумматорах обратного кода (шесть двоичных разрядов для мантиссы и четыре двоичных разряда для порядка).

Решение. Прежде всего записываются машинные изображения чисел и определяется, какой из двух порядков больше:

$$[m_A]_{об} = 00,1011; \qquad [p_A]_{об} = 1,101;$$

$$[m_B]_{об} = 11,0110; \qquad [p_B]_{об} = 1,100,$$

$$[\Delta p]_{об} = [p_A]_{об} - [p_B]_{об}.$$

Величину $(-[p_B]_{об})$ обозначим $[\overline{p}_B]_{об}$, что означает изменение знака числа p_B на обратный, т. е. $[\overline{p}_B]_{об} = 0,011$. Тогда

$$[\Delta p]_{об} = [p_A]_{об} + [\overline{p}_B]_{об} = 0,001.$$

Так как величина ΔP положительная, то $p_A > p_B$. Следовательно, надо сдвинуть мантиссу числа B вправо на количество разрядов, равное ΔP, т. е. на один разряд: $[\overrightarrow{m}_B]_{об} = 11,1011$ (сдвиг модифицированный, стрелка над символом m_B показывает сдвиг в соответствующую сторону). Теперь порядки операндов равны и дальнейшие действия производятся в последовательности, аналогичной последовательности, рассмотренной в примере 3.12.

Складываются изображения мантисс:

$$\begin{array}{r} [m_A]_{об} = 00,1011 \\ + \\ [\overrightarrow{m}_B]_{об} = 11,1011 \\ \hline [m_C]_{об} = 00,0111 \ (\delta = 1, \ \gamma = 0). \end{array}$$

Осуществляется нормализация мантиссы ($\delta=1$) и соответствующая коррекция порядка:

$$[m'_C]_{об} = 00,1110 \ (\delta = 0, \ \gamma = 0),$$

$$[p'_C]_{об} = 1,101 + 1,110 = 1,100.$$

Так как нарушений нормализации нет, то получен окончательный результат. *Ответ:* $C = 0,1110 \cdot 2^{-3}$.

Пример 3.14. Сложить числа, заданные в форме с фиксированной точкой.

$$m_A = 100110; \quad X_A = 101;$$

$$m_B = -111001; \quad X_B = 011.$$

Для выполнения операции сложения используется сумматор дополнительного кода, имеющий семь битов для мантиссы со знаком, четыре бита для характеристики со знаком.

Решение. Сначала записываются машинные изображения мантисс:

$$[m_A]^{м}_{д} = 00.100110;$$

$$[m_B]^{м}_{д} = 11.000111.$$

Примечание. Исходные числа в памяти машины можно хранить либо в прямом, либо в обратном (дополнительном) кодах. Если числа хранятся в памяти машины в прямом коде, то при выполнении операции сложения (вычитания) на сумматорах обратного (дополнительного) кода необходимо произвести преобразование из прямого кода в обратный (дополнительный) код. По окончании операции должно производиться преобразование результата из обратного (дополнительного) кода в прямой.

При выполнении данного примера предполагается, что числа в памяти машины хранятся в дополнительном коде.

Прежде всего необходимо сравнить характеристики:

$$\Delta X = [X_A]_{д} - [X_B]_{д} = 0.101 + 1.101 = 0.010.$$

Разность характеристик — положительная: второй порядок меньше первого на 2. Следовательно, мантисса второго числа сдвигается на два разряда (сдвиг модифицированный) и после этого мантиссы складываются:

$$\begin{array}{l} [\overrightarrow{m}_B]_{д} = 11.110001 \\ \qquad\qquad + \\ \underline{[m_A]_{д} = 00.100110} \\ \mathbf{[m_C]_{д} = 00.010111 \ \ (\delta = 1, \ \gamma = 0).} \end{array}$$

Так как $\delta = 1$, то производится сдвиг влево на один разряд с коррекцией характеристики:

$$[m'_C] = 00101110 \ (\delta = 0, \ \gamma = 0),$$

$$[X'_C] = 0.101 + 1.111 = 0.100.$$

Таким образом, окончательный результат получен в нормализованном виде. *Ответ:* $C = +101110$, $X_C = 100$.

Пример показан для случая, когда мантисса является целым числом и представляется в форме с фиксированной точкой перед старшим разрядом. Как видно, сформулированные выше правила выполнения алгоритма алгебраического сложения действуют в данном случае без существенных изменений.

При реализации операций сложения (вычитания) чисел, представленных в форме с плавающей запятой, может возникнуть переполнение разрядной сетки сумматора порядков (характеристик):

мантисса получается нормализованной и правильной, а порядок (характеристика) не соответствует. Следовательно, необходимо вырабатывать сигнал переполнения сумматора порядков.

Нормализация результата операции сложения (вычитания) приводит и к исчезновению порядка (т. е. характеристика становится отрицательной), несмотря на то что мантисса отлична от нуля. В ЕС ЭВМ вводится специальный разряд, в котором записывается нуль или единица: при нулевом значении этого разряда в результате операции записывается истинный нуль, т. е. число с нулевой мантиссой, положительным знаком и нулевой характеристикой; при единичном значении этого разряда к характеристике прибавляется $+X_{\max}$.

§ 3.8. Оценка точности выполнения арифметических операций

Выбор системы счисления и длины разрядной сетки машины, а также формы представления числа в машине тесно связаны с обеспечением заданной точности вычислений. Важное значение имеет также оценка точности арифметических вычислений при использовании в машинах чисел, представленных в форме с фиксированной и плавающей запятой. При операциях сложения и вычитания (при условии отсутствия переполнения в естественной форме) можно считать, что они выполняются точно.

Для чисел, представленных в форме плавающей запятой, при операциях сложения и вычитания необходимо выравнивать порядки, что ведет к потере некоторых разрядов мантиссы при сдвиге. Поэтому при нормализованной форме представления чисел сама операция алгебраического сложения также является источником погрешностей.

Таким образом, причинами погрешностей вычислений на ЭВМ могут быть:

1) неточное задание исходных данных, участвующих в выполняемой операции (либо из-за ограниченности разрядной сетки машины, либо из-за погрешностей перевода информации из одной системы счисления в другую);

2) использование приближенных методов вычислений, что само по себе дает методическую погрешность (например, использование рядов Ньютона и Тейлора при интегрировании);

3) округление результатов элементарных операций, что в свою очередь может привести к появлению накопленных погрешностей;

4) сбои в работе ЭВМ (эта причина может быть устранена введением системы контроля выполнения любых операций).

П о г р е ш н о с т и   в ы п о л н е н и я   а р и ф м е т и ч е с к и х   о п е р а ц и й. Погрешности выполнения арифметических действий в цифровых автоматах могут быть оценены, если рассматривать их как элементарную операцию над операндами.

Рассмотрим числа $A=[A]+\Delta A$ и $B=[B]+\Delta B$, заданные с абсолютными погрешностями.

Результаты операций сложения и вычитания этих чисел будут иметь вид

$$A+B=[A]+[B]+\Delta A+\Delta B;$$

$$A-B=[A]-[B]+(\Delta A-\Delta B).$$

При выполнении операций умножения получаем

$$AB=[A][B]+[A]\Delta B+[B]\Delta A+\Delta A\Delta B.$$

Так как произведение $\Delta A\Delta B$ — величина второго порядка малости, то ею можно пренебречь. Следовательно,

$$AB\approx[A][B]+[A]\Delta B+[B]\Delta A,$$

т. е. абсолютная погрешность произведения

$$\boxed{\Delta AB=[A]\Delta B+[B]\Delta A.}$$

При выполнении операции деления получаем

$$\frac{A}{B}=\frac{[A]+\Delta A}{[B]+\Delta B}=\frac{[A]+\Delta A}{[B]}\left(\frac{1}{1+\Delta B/[B]}\right).$$

Второй сомножитель в правой части уравнения разложим в ряд. После преобразований получим

$$\frac{A}{B}=\frac{[A]}{[B]}-\frac{[A]\Delta B}{([B])^2}+\frac{[A](\Delta B)^2}{([B])^3}+\frac{\Delta A}{[B]}-\frac{\Delta A\Delta B}{([B])^2}-\ldots \quad (3.16)$$

Пренебрегая членами второго порядка малости, (3.16) можно упростить:

$$\frac{A}{B}\approx\frac{[A]}{[B]}+\frac{\Delta A}{[B]}-\frac{[A]\Delta B}{([B])^2}.$$

Отсюда абсолютная погрешность частного

$$\Delta\frac{A}{B}=\frac{\Delta A}{[B]}-\frac{[A]\Delta B}{([B])^2}.$$

Аналогичным образом можно вывести выражения для относительных погрешностей:

при сложении — вычитании

$$\delta_{A\pm B}=\frac{[A]}{[A]+[B]}\cdot\frac{\Delta A}{[A]}\pm\frac{[B]}{[A]+[B]}\cdot\frac{\Delta B}{[B]}; \quad (3.17)$$

при умножении

$$\boxed{\delta_{AB}=\frac{\Delta A}{[A]}+\frac{\Delta B}{[B]};} \quad (3.18)$$

при делении

$$\delta_{A/B}=\frac{\Delta A}{[A]}-\frac{\Delta B}{[B]}. \quad (3.19)$$

Погрешности округления. Если предположить, что исходная информация не содержит никаких ошибок и все вычислительные процессы конечны и не приводят к ошибкам, то в этом случае все равно присутствует третий тип ошибок — ошибки округления. Предположим, что вычисления производят на некоторой гипотетической машине, в которой каждое число представляется пятью значащими цифрами, и что необходимо сложить числа 9,2654 и 7,1625, причем эти числа точные. Сумма чисел равна 16,4279, она содержит шесть значащих цифр и не помещается в разрядной сетке машины. Поэтому шестизначный результат будет округлен до значения 16,428. В результате возникает погрешность округления.

Так как вычислительные машины всегда работают с конечным количеством значащих цифр, то потребность в округлении возникает довольно часто. Вопросы округления относятся только к действительным числам; это объясняется тем, что ЭВМ автоматически выравнивает порядки действительных чисел при сложении и вычитании.

В самом деле, для чисел, представленных в форме с плавающей запятой, справедливо, что

$$A_{(q)} = m_A q^k,$$

где $1/q \leqslant |m_A| < 1$.

Если для представления мантиссы используется только n разрядов, то изображение числа разбивается на две части:

$$A_{(q)} = [m_A]\, q^n + \underbrace{[A_0]\, q^{k-n}}_{A_0},$$

где A_0 — «хвост» числа, не попавший в разрядную сетку.

В зависимости от того, как учитывается величина A_0 в машинном изображении, существует несколько способов округления:

1) отбрасывание A_0. При этом возникает относительная погрешность, равная

$$\delta_{\text{окр}} = \frac{|A_0|\, q^{k-n}}{|m_A|\, q^k}.$$

Так как $q^{-1} \leqslant |m_A| < 1$; $0 \leqslant |A_0| < 1$, то

$$\delta_{\text{окр}} \leqslant \frac{1 q^{k-n}}{q^{-1} q^k} = q^{-(n-1)}, \tag{3.20}$$

т. е. *математическое ожидание погрешности округления не зависит от величины самого числа, а зависит только от количества разрядов в машине для любой системы счисления.*

Дисперсия этой величины примерно равна $\frac{1}{12}\, q^{-2n}$;

2) симметричное округление. При этом производится анализ величины A_0. Тогда принимают, что

$$[A]=\begin{cases} [m_A]\,q^n, & \text{если } |A_0|<q^{-1}; \\ [m_A]\,q^n+q^{k-n}, & \text{если } |A_0|\geqslant q^{-1}, \end{cases} \tag{3.21}$$

что соответствует прибавлению единицы к младшему разряду мантиссы. Абсолютная погрешность округления при этом

$$\Delta_{\text{окр}}=\begin{cases} |A_0|\,q^{k-n}; \\ |1-A_0|\,q^{k-n}. \end{cases} \tag{3.22}$$

Максимально возможное значение модуля абсолютной погрешности равно 0,5 q^{k-n}.

Математическое ожидание относительной погрешности округления

$$\delta_{\text{окр}}\leqslant\frac{0{,}5q^{k-n}}{m_A q^k}=0{,}5q^{-(n-1)}, \tag{3.23}$$

т. е. ошибка не превышает половины единицы младшего разряда.

Способ симметричного округления наиболее часто используют на практике;

3) округление по дополнению. В этом случае для округления берется информация, содержащаяся в $(n+1)$-м разряде.

При $q=2$, если в $(n+1)$-м разряде содержится единица, в n-й разряд добавляется единица; если в $(n+1)$-м разряде находится нуль, содержимое разрядов правее n-го отбрасывается;

4) случайное округление. Для такого округления необходимо иметь датчик случайных величин (1 или 0), который выдает единицу в самый младший разряд машинного изображения числа. Погрешность округления является случайной величиной с нулевым математическим ожиданием.

Оценка накопленной погрешности при вычислениях на машине особенно затруднительна при использовании чисел в форме с плавающей запятой. При таком представлении возможно перемещение ошибки из младших разрядов мантиссы в старшие разряды. Это происходит, например, при вычитании друг из друга близких по значению мантисс. В результирующей мантиссе первые ненулевые разряды оказываются сдвинутыми в правую часть разрядной сетки машины. При нормализации они перемещаются в левую часть разрядной сетки, давая бóльшую погрешность результата.

Для автоматической оценки накопленной ошибки при вычислении чисел в форме с плавающей запятой в разрядной сетке машины кроме числовой информации записываются также информации об ошибке, содержащейся в числовой информации. При этом предполагается, что ошибки всех чисел — независимые величины и их распределение подчинено нормальному закону. Эти допущения весьма существенны, так как на практике ошибки при вычислениях, конечно, являются зависимыми величинами и их распределение может быть далеким от нормального. Кроме того, принимается, что все числа, записанные в разрядной сетке машины, имеют погрешность $\pm 0{,}5$ последней значащей цифры. Значение этой вероятностной ошибки записывается в исходных данных в разрядах, находящихся правее самого младшего разряда мантиссы. После арифметических операций нормализация осуществляется не всегда, а лишь в случаях, когда срабатывает критерий сдвига, оценивающий величину погрешности, вносимой в число в процессе нормализации.

Следует отметить также, что оценка точности вычислений на машинах зависит не только от состава выполняемых операций, но и от их следования друг за другом.

Задание для самоконтроля

1. Написать изображения чисел $A = -0{,}101010$ и $B = 0{,}100010$ в прямом, обратном и дополнительном кодах.

2. Возможно ли переполнение разрядной сетки, если числа с плавающей запятой складываются, умножаются, делятся?

3. Сложить на сумматоре прямого кода числа $A = -0{,}11101$ и $B = 0{,}10100$.

4. Сложить на сумматоре обратного кода числа $A = 0{,}10110$, $B = -0{,}10110$.

5. Сложить на сумматоре дополнительного кода числа $A = 0{,}11001$ и $B = 0{,}10111$.

6. Указать признак переполнения разрядной сетки на сумматоре обратного кода при сложении отрицательных чисел и положительных чисел.

7. Применимы ли понятия обратного, дополнительного и прямого кодов для представления чисел в минус-двоичной системе счисления?

4 ГЛАВА

УМНОЖЕНИЕ ДВОИЧНЫХ ЧИСЕЛ

§ 4.1. Основные методы выполнения операции умножения в двоичной системе счисления

Применительно к двоичной системе счисления наиболее известны следующие основные способы выполнения операции умножения:

1) умножение начиная с младших разрядов множителя:

```
      1101 — множимое,
    × 1101 — множитель,
      ----
      1101
     0000
  + 1101    — частные произведения,
   1101
  --------
  10101001 — произведение;
```

2) умножение начиная со старших разрядов множителя:

```
   1101        — множимое,
 × 1101        — множитель,
   ----
   1101
 +  1101       — частные произведения,
     0000
      1101
   --------
   10101001    — произведение.
```

В обоих случаях операция умножения состоит из ряда последовательных операций сложения частных произведений. Операциями сложения управляют разряды множителя: если в каком-то разряде множителя находится единица, то к сумме частных произведений добавляется множимое с соответствующим сдвигом; если в разряде множителя — нуль, то множимое не прибавляется.

Таким образом, кроме операции сложения чисел для получения произведения необходима операция сдвига чисел. При этом появляется возможность сдвигать множимое или сумму частных произведений, что дает основание для разных методов реализации операции умножения.

Метод 1. Пусть A — множимое ($A>0$), B — множитель ($B>0$), C — произведение. Тогда в случае представления чисел в

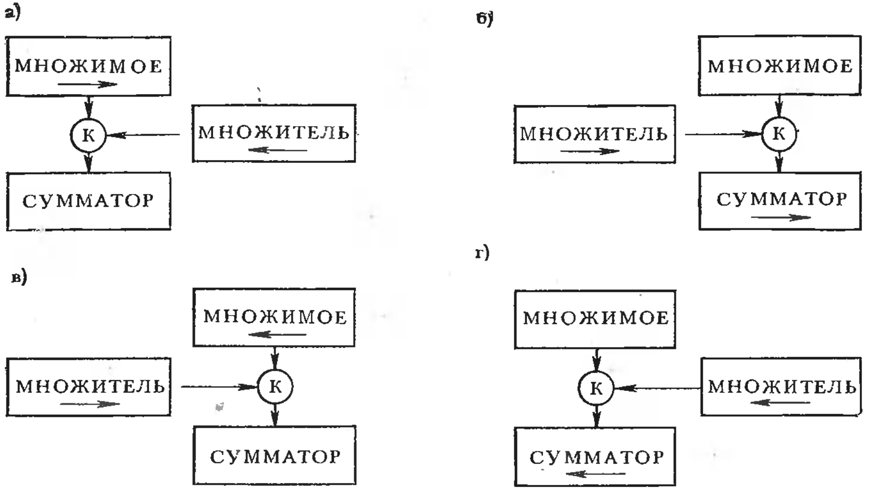


Рис. 4.1. Структурные схемы множительного устройства

форме с фиксированной запятой получаем:

$$A=0,\ a_1a_2\ldots a_n;$$

$$B=0,\ b_1b_2\ldots b_n=b_1 2^{-1}+b_2 2^{-2}+\ldots+b_n 2^{-n}.$$

Отсюда

$$C=A\cdot B=0,\ a_1a_2\ldots a_n(b_1 2^{-1}+\ldots+b_n 2^{-n})=$$
$$=(2^{-1}\cdot 0,\ a_1a_2\ldots a_n)\,b_1+(2^{-2}\cdot 0,\ a_1a_2\ldots a_n)\,b_2+\ldots$$
$$\ldots+(2^{-n}\cdot 0,\ a_1a_2\ldots a_n)\,b_n. \qquad (4.1)$$

Множитель 2^{-n} означает сдвиг на n разрядов вправо числа, которое заключено в скобки, т. е. в данном случае сдвигается вправо множимое и умножение начинается со старших разрядов.

Структурная схема рассмотренного множительного устройства представлена на рис. 4.1, *а*.

М е т о д 2. Пусть $A=0,\ a_1a_2\ldots a_n$ — множимое и $B=0,\ b_1b_2\ldots b_n$ — множитель.

Множитель можно легко преобразовать, используя метод Горнера:

$$B=(\ldots((b_n\cdot 2^{-1}+b_{n-1})\cdot 2^{-1}+b_{n-2})\cdot 2^{-1}+\ldots+b_2)\cdot 2^{-1}+b_1)\cdot 2^{-1}.$$

Тогда

$$C=AB=(\ldots((b_n\cdot 0,\ a_1a_2\ldots a_n)\,2^{-1}+b_{n-1}0,\ a_1a_2\ldots a_n)\,2^{-1}+\ldots$$
$$\ldots+b_1\cdot 0,\ a_1a_2\ldots a_n)\,2^{-1}. \qquad (4.2)$$

Здесь умножение начинается с младших разрядов и сдвигается вправо сумма частных произведений. Структурная схема множительного устройства, реализующего этот метод, представлена на рис. 4.1, *б*.

Метод 3. Пусть $A=0,\ a_1a_2...a_n$ — множимое и $B=0,\ b_1b_2...b_n$ — множитель.

Множитель, используя метод Горнера, можно записать так:

$$B=2^{-n}(b_1\cdot 2^{n-1}+b_2\cdot 2^{n-2}+\ldots+b_{n-1}\cdot 2^1+b_n\cdot 2^0)=$$
$$=2^{-n}(\ldots((b_1\cdot 2^1+b_2)\,2^1+\ldots+b_{n-1})\,2^1+b_n).$$

В этом случае

$$C=AB=2^{-n}(b_n\cdot 0,\ a_1a_2\ldots a_n+(2^1\cdot 0,\ a_1a_2\ldots a_n)\,b_{n-1}+\ldots$$
$$\ldots+(2^{n-1}\cdot 0,\ a_1a_2\ldots a_n)\,b_1), \qquad (4.3)$$

что означает: умножение начинается с младших разрядов, и множимое сдвигается влево на один разряд в каждом такте. Схема множительного устройства представлена на рис. 4.1, *в*.

Метод 4. Пусть $A=0,\ a_1a_2...a_n$ — множимое и $B=0,\ b_1b_2...b_n$ — множитель.

Если множитель B записать по методу Горнера:

$$C=AB=2^{-n}(\ldots(2^1(b_1\cdot 0,\ a_1a_2\ldots a_n)+b_2\cdot 0,\ a_1a_2\ldots a_n)\,2^1+\ldots$$
$$\ldots+b_{n-1}\cdot 0,\ a_1a_2\ldots a_n)\,2^1+b_n\cdot 0,\ a_1a_2\ldots a_n), \qquad (4.4)$$

то умножение начинается со старшего разряда и в каждом такте сдвигается влево сумма частных произведений. Схема множительного устройства представлена на рис. 4.1, *г*.

Таким образом, для реализации операции умножения необходимо иметь сумматор, регистры для хранения множимого и множителя и схему анализа разрядов множителя. Сумматор и регистры должны иметь цепи сдвига содержимого в ту или иную сторону в соответствии с принятым методом умножения.

Анализ формул (4.1)—(4.4) показывает, что с формальной точки зрения процесс умножения двух чисел может быть представлен:

при последовательном выполнении — в виде многократно повторяющегося по количеству разрядов цикла

$$S_i=S_{i-1}+A\cdot b_i, \qquad (4.5)$$

где S_{i-1}, S_i — суммы частных произведений на $(i—1)$-м и i-м шагах соответственно;

при параллельном выполнении — суммой членов диагональной матрицы, для которой заданы по строкам $A\cdot 2^{-i}$, а по столбцам — b_i, где i — текущий номер разряда.

Примечание. В дальнейшем основное внимание будет уделено последовательному принципу выполнения операции умножения.

При точном умножении двух чисел количество цифр в произведении превышает количество цифр сомножителей в пределе в два раза. При умножении нескольких чисел количество цифр произведения может оказаться еще больше. Конечное число разрядов в устройствах цифрового автомата вынуждает ограничиваться максимально удвоенным количеством разрядов сумматоров.

При ограничении количества разрядов сумматора в произведение вносится погрешность. В случае большого объема вычислений погрешности одного знака накладываются друг на друга, в результате чего общая погрешность сильно возрастает. Поэтому существенное значение имеет округление результатов умножения, что дает возможность сделать погрешность произведения знакопеременной, а математическое ожидание погрешности (при условии, что отброшенные младшие разряды могут с одинаковой вероятностью иметь любое из возможных значений) — равным нулю. При этом предельное по абсолютной величине значение погрешности будет наименьшим из возможных при заданном количестве значащих цифр, т. е. равным половине младшего разряда.

При выполнении операции умножения чисел возможен выход за пределы разрядной сетки только со стороны младших разрядов в силу ограничения, которое было положено на числа, представленные в форме с фиксированной запятой. Точное произведение получается во всех четырех методах умножения, однако при этом требуется разное количество оборудования.

§ 4.2. Умножение чисел, представленных в форме с фиксированной запятой, на двоичном сумматоре прямого кода

Пусть заданы машинные изображения двух чисел:

$$[A]_{\text{пр}} = \mathrm{Sg}_A,\ a_1 a_2 \ldots a_n;$$

$$[B]_{\text{пр}} = \mathrm{Sg}_B,\ b_1 b_2 \ldots b_n.$$

Тогда их произведение

$$[C]_{\text{пр}} = \mathrm{Sg}_C,\ c_1 c_2 \ldots c_n,$$

где $\mathrm{Sg}_C = \mathrm{Sg}_A \oplus \mathrm{Sg}_B$, $\oplus$ — знак сложения по модулю 2 (см. § 9.2).

При выполнении этой операции должны быть заданы структурная схема устройства, на котором производится операция, и метод умножения.

Пример 4.1. Умножить числа

$$[A]_{\text{пр}} = 1,11010;$$

$$[B]_{\text{пр}} = 0,11001.$$

При умножении будут использованы метод 2 и устройство, показанное на рис. 4.2.

Запись всех действий, выполняемых устройством, осуществляется с помощью условных обозначений, т. е. $:=$ — оператор присваивания (означает, что блоку, который указан слева от оператора, присваивается значение, указанное справа от оператора; $[\overrightarrow{\text{Pr}\,A}]$ — сдвиг содержимого регистра вправо на один разряд; [СМ] — содержимое сумматора; И. П. — исходное положение.

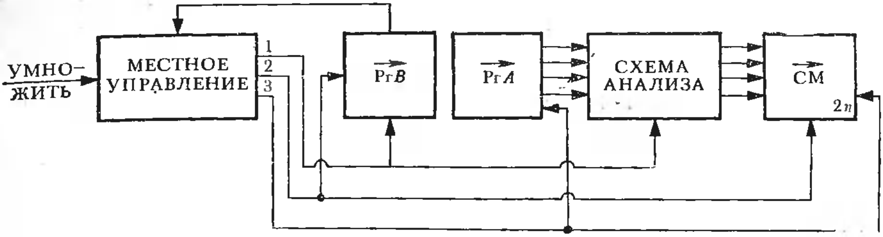


Рис. 4.2. *Множительное устройство*

Решение. Знак произведения определяем отдельно от цифровой части в соответствии с уравнением

$$\mathrm{Sg}_C = \mathrm{Sg}_A \oplus \mathrm{Sg}_B = 1 \oplus 0 = 1.$$

Получение цифровой части можно показать в виде следующей записи. Пусть сумматор имеет 10 разрядов без учета знака, а регистры — 5 разрядов без знака. Введем обозначения $[A']$, $[B']$ — соответственно изображения цифровой части множимого и цифровой части множителя.

Последовательность действий в процессе выполнения операции умножения представлена в виде табл. 4.1.

Ответ: $[C]_{\text{пр}} = 1{,}1010001010$.

Таблица 4.1

Сумматор (СМ)	Регистр (Рг B)	Примечание
0000000000	11001	И. п. [СМ] : =0; [Рг A] : =$[A']$; [Рг B] : =$[B']$;
+ 11010		$b_5=1$; [СМ]:=[СМ]+[Рг A];
1101000000		$[\overrightarrow{\text{Рг}}\,B]$; $[\overrightarrow{\text{СМ}}]$;
0110100000	-1100	$b_4=0$; $[\overrightarrow{\text{Рг}}\,B]$; $[\overrightarrow{\text{СМ}}]$;
0011010000	--110	$b_3=0$; $[\overrightarrow{\text{Рг}}\,B]$; $[\overrightarrow{\text{СМ}}]$;
0001101000	---11	$b_2=1$; [СМ]:=[СМ]+[Рг A];
+ 11010		$[\overrightarrow{\text{СМ}}]$; $[\overrightarrow{\text{Рг}}\,B]$;
1110101000		
0111010100	----1	$b_1=1$; [СМ]:=[СМ]+[Рг A];
+ 11010		[Рг B]; $[\overrightarrow{\text{СМ}}]$
10100010100*		
1010001010		Конец

* Если в процессе выполнения умножения возникает единица переноса из старшего разряда, то ее надо сохранять.

Чтобы процесс умножения происходил правильно, необходимо предусмотреть блокировку выработки сигнала переполнения φ, так как возможно временное переполнение на каком-то шаге умножения (см. пример 4.1). Пример показывает, что в данном случае

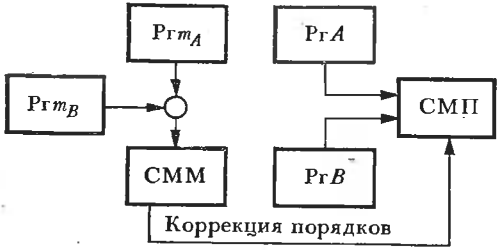


Рис. 4.3. Устройство для умножения чисел с плавающей запятой.

не обязательно иметь сумматор длиной $2n$ разрядов. Хранение «хвостов» произведения можно осуществлять в освобождающихся разрядах регистра множителя. Для этого достаточно обеспечить цепь передачи информации из младшего разряда сумматора в старший разряд регистра множителя.

Примечание. Во всех приведенных ниже примерах будет применяться рассмотренный способ.

§ 4.3. Особенности умножения чисел, представленных в форме с плавающей запятой

Для чисел, представленных в форме с плавающей запятой, обязательным является представление в виде мантиссы и порядка (характеристики). При операции умножения действия, выполняемые над мантиссами и порядками, различны: мантиссы перемножаются, порядки складываются. Очевидно, что результат умножения может получиться ненормализованным, тогда потребуется нормализация с соответствующей коррекцией порядка результата. Следовательно, структурная схема множительного устройства должна измениться (см. рис. 4.3).

Рассмотрим пример выполнения операций умножения чисел, заданных в прямом коде.

Пример 4.2. Умножить числа $A = -0{,}11001 \cdot 2^{-3}$ и $B = 0{,}10011 \cdot 2^{+1}$.

В качестве множительного устройства используется схема, показанная на рис. 4.3, где PгA и PгB — соответственно регистры для порядков p_A и p_B.

Решение. Мантиссы перемножаются по правилам, рассмотренным для чисел, представленных в форме с фиксированной запятой. Для перемножения мантисс используется сумматор прямого кода, а для сложения порядков — сумматор обратного кода.

Сначала записываются машинные изображения чисел:

$$[m_A]_{пр} = 1{,}11001; \qquad [p_A]_{об} = 1{,}100;$$

$$[m_B]_{пр} = 0{,}10011; \qquad [p_B]_{об} = 0{,}001.$$

Последовательность действий в процессе выполнения операции умножения мантисс представим в табл. 4.2.

После выполнения указанных действий находится мантисса произведения

$$[m_C]_{пр} = 1{,}0111011011.$$

Одновременно с этим над порядками производится операция сложения:

$$[p_C]_{об} = [p_A]_{об} + [p_B]_{об} = 1{,}100 + 0{,}001 = 1{,}101.$$

Таблица 4.2

Знак результата	Сумматор (СМ)	Регистр (Рг)	Примечание
$Sg_C = Sg_A \oplus$ $\oplus Sg_B = 1 \oplus 0 = 1$	00000	10011	И. П. Рг $m_B := [m_B]$; Рг $m_A := [m_A]$; Рг $P_A := [P_A]$; Рг $P_B := [P_B]$; См $m := 0$.
	+ 11001		$b_5 = 1$; См $m := [\text{См } m] + [\text{Рг } m_A]$;
	11001		
	011001	01001	$\overrightarrow{\text{См } m}$; $\overrightarrow{\text{Рг } m_B}$;
	+ 11001		$b_4 = 1$; См $m := [\text{См } m] + [\text{Рг } m_A]$
	1001011		
	1001011	00100	$\overrightarrow{\text{См } m}$; $\overrightarrow{\text{Рг } m_B}$;
	01001011	00010	$b_3 = 0$; $\overrightarrow{\text{См } m}$; $\overrightarrow{\text{Рг } m_B}$;
	001001011	00001	$b_2 = 0$; $\overrightarrow{\text{См } m}$; $\overrightarrow{\text{Рг } m_B}$;
	+ 11001		$b_1 = 1$; См $m := [\text{См } m] + [\text{Рг } m_A]$;
	111011011		
	0111011011	00000	$\overrightarrow{\text{См } m}$; $\overrightarrow{\text{Рг } m_B}$
			Конец

Так как мантисса результата не удовлетворяет условию нормализации (нарушена левая граница: $\delta = 1$, $\gamma = 0$), то производится сдвиг мантиссы влево на один разряд:

$$[m_C'] = 1{,}1110110110,$$

и коррекция порядка

$$[p_C]_{об} = [p_C]_{об} + 1{,}110 = 1{,}101 + 1{,}110 = 1{,}100.$$

Если сумматор мантисс содержит только n разрядов, то после округления получается исходный результат.

Ответ: $C = -0{,}11110 \cdot 2^{-3}$.

При выполнении операции умножения может иметь место ряд особых случаев. Например:

если один из сомножителей равен нулю, то произведение также равно нулю. Следовательно, необходимо предусмотреть блокировку выполнения алгоритма умножения и формировать результат, равный нулю;

если порядок результата равен наибольшей отрицательной величине, то необходимо формировать истинный нуль;

если множимое является наибольшим отрицательным числом, то множитель следует увеличить на 2^{n-1}, т. е. сдвинуть влево.

Эти особые случаи можно предусмотреть в алгоритме операции введением анализатора сомножителей на 0 в начале операции (первый случай) или коррекцией произведения на основании признаков результата (второй и третий случаи).

§ 4.4. Умножение чисел, представленных в форме с фиксированной запятой, на двоичном сумматоре дополнительного кода

В случаях, когда числа в машине хранятся в дополнительных кодах, целесообразно все операции над числами производить на сумматоре дополнительного кода. Однако при этом возникает ряд особенностей, которые необходимо учитывать.

Произведение дополнительных кодов сомножителей равно дополнительному коду результата только в случае положительного множителя.

Пусть множимое A — любое число, т. е. $A=[A]_д$, а множитель $B>0$. Тогда

$$AB=[A]_д\cdot 0,\ b_1b_2\ldots b_n=[A]_д\, b_1\cdot 2^{-1}+[A]_д\, b_2\cdot 2^{-2}+\ldots$$
$$\ldots+[A]_д\, b_n\cdot 2^{-n}. \tag{4.6}$$

На основании теоремы о сложении дополнительных кодов можно утверждать, что в правой части уравнения (4.6) стоит дополнительный код результата.

Таким образом, умножение на сумматоре дополнительного кода заключается в анализе разрядов множителя и при $b_i=1$ в прибавлении дополнительного кода множимого к содержимому сумматора. При этом должны осуществляться модифицированные сдвиги.

Пример 4.3. Умножить на сумматоре дополнительного кода (используется метод 2) числа $A=-0{,}10101$ и $B=0{,}10011$.

Решение. Сначала записываются машинные изображения чисел:

$$[A]_д^м=11{,}01011;$$

$$[B]_д^м=00{,}10011.$$

Последовательность действий, производимых над числами, представлена в табл. 4.3.

Ответ: $C=AB=-0{,}0110001111$.

Теперь рассмотрим случай, когда множимое A — любое число, а множитель $B<0$. Тогда

$$[B]_д=1,\ \overline{b_1}\overline{b_2}\ldots\overline{b_n}.$$

На основании (3.7) можно записать, что $B=[B]_д-2$, или $B=0,\overline{b}_1\overline{b}_2\ldots\overline{b}_n-1$. Следовательно, произведение чисел

$$AB=A\,(0,\ \overline{b_1}\overline{b_2}\ldots\overline{b_n}-1)=A\cdot 0,\ \overline{b_1}\overline{b_2}\ldots\overline{b_n}-A. \tag{4.7}$$

Формула (4.7) показывает, что при отрицательном множителе произведение дополнительных кодов операндов не равно дополнительному коду результата. Если ввести замену $-A=\overline{A}$, то можно вывести следующее правило.

Если множитель отрицательный, то произведение чисел на сумматоре дополнительного кода получается прибавлением поправки $[\overline{A}]$ *к произведению дополнительных кодов сомножителей.*

Таблица 4.3

Сумматор (СМ)	Регистр (Рг B)	Примечание
00,00000	10011	И. П. СМ:=0; Рг A:=$[A]_д^м$; Рг B:=$[B']$;
+		$b_5=1$; [СМ]:=[СМ]+[Рг A];
11,01011		
11,01011		$[\overrightarrow{\text{Рг}\,B}]$; $[\overrightarrow{\text{СМ}}]$;
11,10101 →	11001	$b_4=1$; [СМ]:=[СМ]+[Рг A];
+		
11,01011		
11,00000		$[\overrightarrow{\text{Рг}\,B}]$; $[\overrightarrow{\text{СМ}}]$;
11,10000 →	01100	$b_3=0$; $[\overrightarrow{\text{Рг}\,B}]$; $[\overrightarrow{\text{СМ}}]$;
11,11000 →	00110	$b_2=0$; $[\overrightarrow{\text{Рг}\,B}]$; $[\overrightarrow{\text{СМ}}]$;
11,11100 →	00011	$b_1=1$; [СМ]:=[СМ]+[Рг A];
+		
11,01011		
11,00111	0001	
11,10011 →	10001	$[\overrightarrow{\text{Рг}\,B}]$; $[\overrightarrow{\text{СМ}}]$
		Конец

Пример 4.4. Умножить на сумматоре дополнительного кода (по методу 2) с использованием структурной схемы примера 4.1 числа $A=-0{,}10111$ и $B=-0{,}11001$.

Решение. Сначала записываются машинные изображения чисел:

$$[A]_д^м = 11{,}01001;$$

$$[B]_д^м = 11{,}00111;$$

$$[\overline{A}] = 00{,}10111.$$

Последовательность действий, производимых над числами, показана в табл. 4.4.

Ответ: $C=AB=0{,}1000111111$.

Таким образом, на сумматоре дополнительного кода в процессе перемножения машинных изображений операндов получаем одновременно знаковую и цифровую части произведения.

§ 4.5. Умножение чисел на двоичном сумматоре обратного кода

По аналогии с предыдущим случаем рассмотрим правила умножения операндов, заданных в обратном коде.

Произведение обратных кодов сомножителей равно обратному коду результата только в случае положительного множителя.

Пусть множимое $A=[A]_{об}$, а множитель $B>0$. Тогда $AB=[A]_{об}\,0,\,b_1b_2...b_n=[A]_{об}b_1\cdot 2^{-1}+[A]_{об}b_2\cdot 2^{-2}+\ ...\ +[A]_{об}b_n\cdot 2^{-n}$.

По теореме о сложении обратных кодов, в правой части данного уравнения получается обратный код результата.

Таблица 4.4

Сумматор (СМ)	Регистр (Рг B)	Примечание
00,00000	00111	И. П. СМ:=0; РгB:=$[B']_{д}$; РгA:=$[A]_{д}^{м}$;
+		$\bar{b}_5=1$; [СМ]:=[СМ]+$[A]_{д}^{м}$;
11,01001		
11,01001		
11,10100	→ 10011	$[\overrightarrow{СМ}]$; $[\overrightarrow{Рг\, B}]$;
+		$\bar{b}_4=1$; [СМ]:=[СМ]+$[A]_{д}^{м}$;
11,01001		
10,11101		
11,01110	→ 11001	$[\overrightarrow{СМ}]$; $[\overrightarrow{Рг\, B}]$;
+		
11,01001		$\bar{b}_3=1$; [СМ]:=[СМ]+$[A]_{д}^{м}$;
10,10111		$[\overrightarrow{СМ}]$; $[\overrightarrow{Рг\, B}]$;
11,01011	→ 11100	$\bar{b}_2=0$; $[\overrightarrow{СМ}]$; $[\overrightarrow{Рг\, B}]$;
11,10101	→ 11110	$\bar{b}_1=0$; $[\overrightarrow{СМ}]$; $[\overrightarrow{Рг\, B}]$;
11,11010	→ 11111	
+		[СМ]:=[СМ]+$[\bar{A}]$
00,10111		
00,10001	11111	Конец

Следовательно, умножение на сумматоре обратного кода также заключается в анализе разрядов множителя, и если оказывается, что очередной разряд множителя равен единице, то к содержимому сумматора добавляется обратный код множимого.

Пример 4.5. Умножить на сумматоре обратного кода (структурная схема взята из примера 4.1) числа $A=-0{,}10011$ и $B=0{,}11001$.

Решение. Сначала записываются машинные изображения чисел:

$$[A]_{об}^{м}=11{,}01100;$$

$$[B]_{об}^{м}=00{,}11001.$$

Последовательность действий, производимых над числами, представлена в табл. 4.5.

Ответ: $[C]_{об}^{м}=11{,}1000100100$; $C=AB=-0{,}0111011011$.

Пусть $A=[A]_{об}$ и $B<0$. Тогда

$$[B]_{об}=1,\ \bar{\bar{b}}_1\bar{\bar{b}}_2\ldots\bar{\bar{b}}_n.$$

В соответствии с (3.10)

$$[B]_{об}=2+B-2^{-n}.$$

Следовательно,

$$B=0,\ \bar{\bar{b}}_1\bar{\bar{b}}_2\ldots\bar{\bar{b}}_n+2^{-n}-1.$$

В результате произведение будет равно

$$AB=[A]_{об}\cdot 0,\ \bar{\bar{b}}_1\bar{\bar{b}}_2\ldots\bar{\bar{b}}_n+[A]_{об}\cdot 2^{-n}+\bar{A}. \qquad (4.8)$$

Таблица 4.5

Сумматор (СМ)	Регистр (Рг B)	Примечание
11,11111	11001	И. П. См:=0; Рг A:=$[A]_{об}$; Рг B:=$[B']$;
+ 11,01100		$b_5=1$; $[СМ]:=[СМ]+[A]^{м}_{об}$;
11,01100		$[\overrightarrow{СМ}]$; $[\overrightarrow{Рг}\, B]$;
11,10110	→ 01100	$b_4=0$; $[\overrightarrow{СМ}]$; $[\overrightarrow{Рг}\, B]$;
11,11011	→ 00110	$b_3=0$; $[\overrightarrow{СМ}]$; $[\overrightarrow{Рг}\, B]$;
11,11101	→ 10011	$b_2=1$; $[СМ]:=[СМ]+[A]^{м}_{об}$;
+ 11,01100		
11,01010	10011	
11,10101	→ 01001	$[\overrightarrow{СМ}]$; $[\overrightarrow{Рг}\, B]$;
+ 11,01100		$b_1=1$; $[СМ]:=[СМ]+[A]^{м}_{об}$;
11,00010	01001	$[\overrightarrow{СМ}]$; $[\overrightarrow{Рг}\, B]$;
11,10001	→ 00100	Конец

На основании формулы (4.8) можно сформулировать правило: *если множитель отрицательный, то произведение чисел на сумматоре обратного кода получается прибавлением поправок* $[\bar{A}]$ *и* $[A]_{об}\cdot 2^{-n}$ *к произведению обратных кодов сомножителей.*

Пример 4.6. Умножить на сумматоре обратного кода (используется метод 2 и структурная схема из примера 4.1) числа $A=-0{,}110101$ и $B=-0{,}101000$.

Решение. Сначала записываются машинные изображения чисел:

$$[A]^{м}_{об}=11{,}001010;$$

$$[B]^{м}_{об}=11{,}010111;$$

$$[\bar{A}]=00{,}110101.$$

Последовательность действий, производимых над числами, показана в **табл. 4.6.**

Ответ: $AB=00{,}100010$.

Таким образом, в общем случае на сумматоре обратного кода произведение получается сразу со знаком и длиной в n разрядов, так как на последнем шаге умножения прибавляются числа разных знаков, из-за чего нельзя к результату приписать так называемый «хвост», хранящийся в регистре множителя.

§ 4.6. Метод сокращенного умножения

В специализированных машинах иногда используют метод сокращенного умножения начиная со старших разрядов. Особенность этого метода состоит в том, что в произведении получается только n старших разрядов. Существуют разные пути выполнения сокращенного умножения. Рассмотрим наиболее распространенные.

Таблица 4.6

Сумматор (СМ)	Регистр (Рг B)	Примечание
11,111111	010111	И. П. [СМ]:=0 Рг A:=$[A]^{\text{м}}_{\text{об}}$; Рг B:=$[B]^{\text{м}}_{\text{об}}$;
+ 11,001010		Добавление $[A]^{\text{м}}_{\text{об}}$ в СМ, т. е. [СМ]:=[СМ]+$[A]^{\text{м}}_{\text{об}}$;
11,001010		
+ 11,001010		$b_6=1$; [СМ]:=[СМ]+$[A]^{\text{м}}_{\text{сб}}$;
10,010101		
11,001010 →	101011	$[\overrightarrow{\text{СМ}}]$; $[\overrightarrow{\text{Рг}\,B}]$;
+ 11,001010		$b_5=1$; [СМ]:=[СМ]+$[A]^{\text{м}}_{\text{об}}$;
10,010101		
11,001010 →	110101	$[\overrightarrow{\text{СМ}}]$; $[\overrightarrow{\text{Рг}\,B}]$;
+ 11,001010		$b_4=1$; [СМ]:=[СМ]+$[A]^{\text{м}}_{\text{сб}}$;
10,010101		
11,001010 →	111010	$[\overrightarrow{\text{СМ}}]$; $[\overrightarrow{\text{Рг}\,B}]$;
11,100101 →	011101	$b_3=0$; $[\overrightarrow{\text{СМ}}]$; $[\overrightarrow{\text{Рг}\,B}]$;
+ 11,001010		$b_2=1$; [СМ]:=[СМ]+$[A]^{\text{м}}_{\text{об}}$;
10,110000		
11,011000 →	001110	$[\overrightarrow{\text{СМ}}]$; $[\overrightarrow{\text{Рг}\,B}]$;
11,101100	000111	$b_1=0$; $[\overrightarrow{\text{СМ}}]$; $[\overrightarrow{\text{Рг}\,B}]$;
+ 00,110101		[СМ]:=[СМ]+$[\bar{A}]$
00,100010		Конец

В некоторых машинах реализован метод, который можно объяснить с помощью следующей схемы.

Полная схема

$$\begin{array}{lr}
\times & A=00,10011 \\
 & B=00,11001 \\
\hline
+ & |\overline{00}|10011 \\
 & \overline{0}\ 010011 \\
\hline
+ & |\overline{01}|\ 11001 \\
 & 0000000 \\
\hline
+ & |\overline{01}|\ 10010 \\
 & 0000000 \\
\hline
+ & |\overline{01}|00100 \\
 & 0010011 \\
\hline
 & |\ \overline{00}|11011
\end{array}$$

Сокращенная схема

$$\begin{array}{ll}
 & 00 \\
 & \ \ 01 \\
+ & \ \ \ \ 01 \\
 & \ \ \ \ \ \ 01 \\
 & \ \ \ \ \ \ \ \ 00 \\
\hline
 & 0,011100
\end{array}$$

Таким образом, $AB=0{,}01110\cdot 2^5$.

Из этой схемы видно, что произведение получается, если складывать только старшие два разряда от сумм, полученных на каждом шаге умножения (показано справа). В этом случае достаточно иметь только n-разрядный сумматор. Однако следует учитывать, что для получения n точных знаков в произведении необходимо ввести дополнительные разряды.

В самом деле, пусть $A=0,\ a_1a_2...a_n$ и $B=0,\ b_1b_2...b_n$. Тогда, если все $b_i=1$, то

$$\begin{array}{l|l|l}
b_1=1 & a_1a_2\ldots a_{n-1}\ a_n & \\
b_2=1 & \quad a_1\ldots a_{n-2}\ a_{n-1} & a_n \\
b_3=1 & +a_1\ldots\ \ldots\ a_{n-2} & a_{n-1}\ a_n \\
b_n=1 & \qquad\qquad\quad a_1 & a_2\ldots a_{n-1}\,a_n \\
 & \underbrace{\qquad\qquad\qquad\qquad}_{n\ \text{разрядов}} & \underbrace{\qquad\qquad\quad}_{n-1\ \text{разрядов}}
\end{array}$$

Предположим, что все разряды, стоящие справа от черты, отбрасываются. Если просуммировать только n разрядов, то вносится погрешность, так как не будут учтены переносы из отброшенных разрядов в разряды слева от вертикальной черты. Эти переносы могут распространиться на k разрядов слева от черты. Предполагается, что каждый разряд дает перенос, равный единице в каждом такте суммирования. Пусть все $a_i=1$. Тогда количество единиц переноса из отброшенной части будет равно

$$\Delta_{\max}=\frac{n-1}{2}+\frac{n-2}{2^2}+\frac{n-3}{2^3}+\ldots+\frac{n-(n-1)}{2^{n-1}}. \qquad (4.9)$$

При достаточно больших значениях n (4.9) может быть записана так:

$$\Delta_{\max}\approx n-2. \qquad (4.10)$$

Очевидно, что единицы переноса распространятся на x разрядов сумматора. Тогда произведение будет содержать только $n-x$ точных разрядов.

Следовательно, чтобы получить результат с точностью до n разрядов, необходимо выделить $n+x$ разрядов в сумматоре. Количество дополнительных разрядов при этом можно приближенно оценить величиной

$$x=\lg(n-2)/\lg 2. \qquad (4.11)$$

Ниже приведены результаты расчета по (4.11):

n	24	32	40	48	64	72
x	4,5	4,9	5,3	5,5	6,0	6,2

§ 4.7. Ускорение операции умножения

В программах решения различных задач операции умножения встречаются достаточно часто. Поэтому методам выполнения умножения, его ускорения и рациональному построению множительных устройств всегда уделяется большое внимание.

По времени выполнения умножение относят к длинным операциям. Так, затраты времени на умножение двух чисел в прямом коде можно оценить следующей формулой (для случая последовательного анализа разрядов множителя):

$$t_{\text{умн}}=\sum_{i=1}^{i=n}(t_{\text{сдв}}+P_i t_{\text{сл}}), \tag{4.12}$$

где $t_{\text{сдв}}$ — время выполнения сдвига числа на один разряд; $t_{\text{сл}}$ — время суммирования на сумматоре; P_i — вероятность появления единицы в разрядах множителя; n — количество разрядов множителя.

Аналогичные формулы можно написать и для других методов умножения.

Анализируя (4.12), можно наметить следующие пути сокращения времени умножения: 1) уменьшение затрат времени на сдвиг и суммирование операндов; 2) уменьшение количества слагаемых в формуле, т. е. уменьшение разрядов множителя n. Сказанное можно реализовать логическими или аппаратными (схемными) средствами. В дальнейшем тексте будет обращено внимание на логические средства.

Рассмотрим возможности изменения величин P_i и $t_{\text{сл}}$.

Наиболее простым способом изменения величин P_i и $t_{\text{сл}}$ является пропуск тактов суммирования в случаях, когда очередная цифра множителя равна нулю. Этот способ может быть применен для систем счисления, содержащих нуль как одну из цифр (например, для систем (0, 1), (0, $\overline{1}$, 1) и т. п.). Исключением в этом отношении являются симметричные системы (1,$\overline{1}$) и им подобные. Однако в системе (1, $\overline{1}$) можно осуществить переход к системе (0, 1) или (0, $\overline{1}$, 1), используя соотношения

$$1\,\overline{1}\,\overline{1}\,\overline{1}\ldots\overline{1}=0\,0\,0\ldots1. \tag{4.13}$$

Такая возможность практически имеется всегда независимо от длины последовательностей 0 или 1. Для ускорения операции умножения можно использовать также наличие последовательностей 0 или 1 (аналогично, $\overline{1}$). Например, последовательность вида... $0\underbrace{100\ldots0}_{k}$ дает возможность сразу осуществить сдвиг на k разрядов, не производя операции, а последовательность вида $\ldots\underbrace{011\ldots1}_{k}$ переходом к избыточной системе (0, $\overline{1}$, 1) может быть заменена на последовательность вида $\ldots1\underbrace{00\ldots\overline{1}}_{k}$, что также приводит к уменьшению количества операций сложений.

Таким образом, переход от одной разновидности двоичной системы счисления к другой при преобразовании множителя позволяет получить выигрыш во времени выполнения операции в целом.

При этом возникают определенной длины последовательности 0 или 1, что в конечном счете приводит к необходимости одновременного анализа нескольких разрядов множителя и сдвигу на произвольное число разрядов.

Рассмотрим простую иллюстрацию этого положения. Пусть множитель $B=0{,}0111100011101111000$ необходимо преобразовать таким образом, чтобы получить меньшее количество единиц в его изображении. Если представить этот множитель в системе счисления $(0,\ 1,\ \bar{1})$, то получим наименьшее число единиц в его изображении: $B=0{,}1000\bar{1}00100\bar{1}100\bar{1}000$ (содержит только шесть единиц взамен десяти).

Чтобы оценить относительную эффективность того или иного логического метода ускорения умножения, можно воспользоваться формулой

$$\sigma=\frac{\sum\limits_{j=1}^{j=m}(t_{\text{сдв}}+P_j t_{\text{сл}})}{\sum\limits_{i=1}^{i=n}(t_{\text{сдв}}+P_i t_{\text{сл}})}, \tag{4.14}$$

где m — число шагов при выполнении операций умножения ускоренным методом (количество групп для анализа); P_i, P_j — вероятности появления единиц в анализируемой i-й или j-й группе разрядов; σ — коэффициент эффективности.

Если $\sigma<1$, то применяемый метод дает эффект; если же $\sigma\geqslant 1$ — эффекта не будет.

Группировка разрядов множителя и анализ этих групп позволяют предложить несколько методов ускорения операции умножения.

1. Анализ двух разрядов множителя одновременно. На основании вышеизложенного можно составить следующие правила преобразования разрядов множителя $B=0,\ b_1 b_2 \ldots b_{n-1} b_n$.

Разбиение множителя на группы длиной k разрядов означает переход к новой системе счисления с основанием 2^k. Если при этом удается сократить количество элементарных действий, выполняемых при операции умножения (сложения и сдвиги), то желаемый эффект имеет место. Рассмотрим сказанное на примере одновременного анализа двух разрядов множителя начиная с младших разрядов.

Основой для правил преобразования разрядов множителя служат следующие соображения.

Комбинации вида 00 и 01 не преобразуются. Комбинация вида 10 означает, что необходим предварительно сдвиг множимого влево, сложение и затем сдвиг содержимого сумматора вправо на два разряда. Комбинация вида 11 заменяется на комбинацию вида $10\bar{1}$, что означает запоминание единицы для следующей анализируемой пары цифр множимого и вычитание на данном шаге с последующим сдвигом на два разряда.

Таблица 4.7

Анализируемая пара разрядов $b_{j+1} b_j$	Перенос из предыдущей пары разрядов	Преобразованная пара разрядов $b'_{j+1} b'_j$	Примечание
00	0	00	
01	0	01	
10	0	10	Предварительный сдвиг множимого
11	0	$0\bar{1}$	Запоминается единица для следующей пары разрядов
00	1	01	
01	1	10	Предварительный сдвиг множимого
10	1	$0\bar{1}$	Запоминается единица для следующей пары разрядов
11	1	00	Запоминается единица для следующей пары разрядов

В табл. 4.7 представлены правила преобразования множителя для системы $(0\ 1, \bar{1})$.

В результате преобразования множителя меняется характер действий, которые должны выполняться при операции умножения, т. е. имеют место операции сложения и вычитания. Следовательно, такой способ умножения может быть реализован только на сумматорах обратного или дополнительного кодов. Основной выигрыш во времени получается за счет того, что на каждом шаге умножения производится либо сложение, либо вычитание множимого и сдвиг на два разряда одновременно. Особенностью является дополнительный сдвиг множимого в противоположную сторону в двух случаях из восьми и запоминание единицы для передачи в следующую пару разрядов множителя. Естественно, что это потребует усложнить устройство местного управления умножением.

При одновременном анализе двух разрядов начиная со старших правила преобразования существенно изменяются. Прежде всего на характер действий влияет значение соседнего справа разряда по отношению к анализируемой паре разрядов. Если его содержимое равно нулю, комбинации вида 00 и 01 выполняются, как в предыдущем случае. Комбинация вида 10 заменяется равнозначной ей комбинацией вида $1\bar{1}0$, что означает предварительный сдвиг множимого и вычитание его. Комбинация вида 11 заменяется комбинацией вида $10\bar{1}$, что означает вычитание множимого на данном шаге. В случае, если значение соседнего справа разряда равно единице, происходит изменение преобразуемых состояний и выполнение действий в соответствии с табл. 4.8.

Таблица 4.8

Анализируемая пара разрядов $b_j b_{j+1}$	Соседний справа разряд b_{j+2}	Преобразованная пара разрядов $b'_j b'_{j+1}$	Примечание
00	0	00	
01	0	01	
10	0	$\bar{1}0$	Предварительный сдвиг множимого
11	0	$0\bar{1}$	
00	1	01	
01	1	$\bar{1}0$	Предварительный сдвиг множимого
10	1	$0\bar{1}$	
11	1	00	

При этом анализ надо начинать с пустой пары.

Пример 4.7. Преобразовать множитель $B=0,\ 10\,11\,00\,11\,10\,01\,10$.

Решение. Следуя правилам табл. 4.8, преобразованный множитель будет иметь вид

$$B_{\text{преобр}} = 01{,}010101001010.$$

Ответ: $B=01{,}0\bar{1}010100\bar{1}0\bar{1}0$.

2. Анализ произвольного количества разрядов множителя. Идея метода состоит в том, что выявляются последовательности нулей или единиц и затем производится групповая обработка разрядов множителя. В случае, если встречается группа вида $\ldots\underbrace{0\,100\ldots0}_{k}$, производится сразу сдвиг на k разрядов и прибавление множимого в сумматор. Если анализируется группа $\ldots 11\underbrace{00\ldots0}_{k}$, то производится сдвиг на k разрядов и вычитается множимое из содержимого сумматора. При анализе группы разрядов вида $\ldots 0\underbrace{011\ldots1}_{k}$ производится замена ее на новую группу вида $\ldots\underbrace{100\ldots\bar{1}}_{k}$, что означает вычитание множимого на первом шаге, сдвиг на k разрядов и анализ группы следующих разрядов, образовавшейся после преобразования. Такой метод ускорения операции умножения требует создания сдвигающего устройства, обладающего возможностью одновременного сдвига на произвольное количество разрядов.

Конечно, если бы числа состояли из достаточно длинного ряда последовательностей 0 или 1, то такой метод дал бы высокий эффект в смысле сокращения времени на обработку. Однако, как правило, в разрядах чаще чередуются 0 или 1. Это значит, что в подобных случаях рассмотренный выше метод не дает никакого эффекта. Поэтому он может быть применен только в комбинации с обычными методами последовательного умножения.

3. Умножение в системе счисления с основанием $q=2^k$. В некоторых машинах ЕС ЭВМ используется переход на новое основание системы счисления вида $q=2^k$, за счет чего удается существенно сократить время выполнения операции умножения.

Пусть $k=4$; это означает, что числа представляются в шестнадцатиричной системе счисления. При этом с помощью приема, аналогичного приему, использованному в случае одновременного анализа двух разрядов множителя, появляется возможность анализировать сразу тетраду (четыре двоичных разряда), рассматриваются текущая цифра (тетрада) множителя и его предыдущая цифра (тетрада). В зависимости от значений цифры множителя в предыдущем разряде (равна или больше восьми, т. е. старший разряд тетрады равен или нет единице) производятся разные действия (табл. 4.9). Для реализации этого приема требуется также предварительно готовить множимое A, увеличивая его в 1, 2, 3 и 6 раз.

Таблица 4.9

Анализируемая цифра	Анализируемая тетрада	Действия на сумматоре при значении предыдущей цифры	
		$\geqslant 8$	<8
0	0000	$+1A$	0
1	0001	$+2A$	$+1A$
2	0010	$+3A$	$+2A$
3	0011	$+(2A+2A)$	$+3A$
4	0100	$+(3A+2A)$	$+(2A+2A)$
5	0101	$+6A$	$+(2A+3A)$
6	0110	$+(6A+1A)$	$+6A$
7	0111	$+(6A+2A)$	$+(6A+A)$
8	1000	$-(1A+6A)$	$+(6A+2A)$
9	1001	$-6A$	$-(6A+A)$
(α_1) 10	1010	$-(3A+2A)$	$-6A$
(α_2) 11	1011	$-(2A+2A)$	$-(3A+2A)$
(α_3) 12	1100	$-3A$	$-(2A+2A)$
(α_4) 13	1101	$-2A$	$-3A$
(α_5) 14	1110	$-A$	$-2A$
(α_6) 15	1111	0	$-A$

Анализ четырех двоичных разрядов одновременно дает возможность сразу осуществлять сдвиг на четыре двоичных разряда.

Так как при операции умножения необходимо складывать и вычитать коды, то потребуется сумматор дополнительного или обратного кода.

Пример 4.8. Умножить два числа $A=0{,}00001101$ и $B=-0{,}00001110$ с одновременным анализом четырех разрядов множителя. Используется сумматор дополнительного кода.

Решение. Для умножения потребуются числа $A=0{,}0000\,1101$, $2A=0{,}0001\,1010$, $3A=0{,}0010\,0111$, $6A=0{,}0100\,1110$.

Множитель представляется в дополнительном коде $[B]_д=1{,}\underbrace{1111}_{b_1}\underbrace{0010}_{b_2}$.

Шаг 1. Анализ первой цифры множителя для $q=2^4$ (предыдущая цифра равна 0): $b_2=0010$ — на основании табл. 4.9 надо вызвать множимое $+2A=0{,}00011010$ и направить его в сумматор. Содержимое сумматора при этом станет $0{,}00011010$.

Шаг 2. Анализ второй цифры множителя (предыдущая цифра меньше восьми): $b_1=1111$ — на основании табл. 4.9 надо вызвать множимое $-1A=1{,}1111\,0011$, сдвинуть его на четыре разряда влево и направить в сумматор, где производится следующее действие:

$$
\begin{array}{r}
+\begin{array}{r} 0{,}00011010 \\ 1{,}00110000 \end{array} \\
\hline
1{,}01001010
\end{array}
$$

Ответ: $AB=-0{,}1011\ 0110$.

§ 4.8. Матричные методы умножения

Существует ряд методов умножения, основанных на суммировании групп частных произведений с последующим объединением сумм вместе с переносами для получения произведения. Например, частные произведения группируются по три и подаются на входы цепочки сумматоров. Выходы цепочки сумматоров подключаются к регистрам, запоминающим отдельно получившуюся сумму и переносы в другие группы. Выходы этих регистров объединяются в группы по три и подключаются уже к другим сумматорам. В конце цепочки складывается только сумма и переносы (для слагаемых). Такая раздельная обработка промежуточных сумм и переносов требует так называемого «дерева сумматоров». Подобный метод умножения использован в вычислительных машинах IBM-360.

Существуют также ускоренные методы умножения, основанные на использовании матриц промежуточных результатов. Рассмотрим схему умножения на примере двух пятиразрядных чисел:

$$
\begin{array}{rccccccccc}
\times & & & & A=a_5 & a_4 & a_3 & a_2 & a_1 \\
 & & & & B=b_5 & b_4 & b_3 & b_2 & b_1 \\
\hline
 & & & & a_5b_1 & a_4b_1 & a_3b_1 & a_2b_1 & a_1b_1 \\
+ & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
 & a_5b_5 & a_4b_5 & a_3b_5 & a_2b_5 & a_1b_5 & & & \\
\hline
 & C_{10} & C_9 & \ldots & \ldots & \ldots & \ldots & C_3\ C_2 & C_1
\end{array}
$$

Эту схему умножения можно представить также в виде матрицы (табл. 4.10).

Таблица 4.10

b_i	a_i				
	a_5	a_4	a_3	a_2	a_1
b_1	a_5b_1	a_4b_1	a_3b_1	a_2b_1	a_1b_1
b_2	a_5b_2	a_4b_2	a_3b_2	a_2b_2	a_1b_2
b_3	a_5b_3	a_4b_3	a_3b_3	a_2b_3	a_1b_3
b_4	a_5b_4	a_4b_4	a_3b_4	a_2b_4	a_1b_4
b_5	a_5b_5	a_4b_5	a_3b_5	a_2b_5	a_1b_5

Каждый элемент этой матрицы равен 0 или 1. Произведение двух чисел можно получить, если суммировать элементы матрицы в следующем порядке:

$$\begin{array}{c|r|r|l}
\ldots & a_3\ b_1 & a_2\ b_1 & a_1b_1 \\
\ldots & +a_2\ b_2 & +a_1\ b_2 & \\
\ldots & +a_1\ b_3 & &
\end{array}$$

Так как при суммировании по столбцам складываются только 0 или 1, то операцию сложения можно выполнить с помощью счетчиков. Однако при значительном числе разрядов сомножителей потребуются счетчики с большим количеством входов и выходов, что существенно увеличивает время суммирования. Но этот принцип умножения можно реализовать таким образом, что количество входов счетчиков на каждом этапе не будет больше трех. Значит, для этих целей можно использовать одноразрядные двоичные полусумматоры и сумматоры.

Наиболее известными среди матричных алгоритмов умножения являются алгоритмы Дадда, Уоллеса (матричный) и алгоритм с сохранением переносов. На рис. 4.4 представлена структурная схема множительного устройства для реализации матричного алгоритма, а на рис. 4.5 — схема, с помощью которой может быть реализован алгоритм Дадда. Как видно из рисунков, алгоритмы отличаются друг от друга не только группировкой частных произведений, но и количеством ступеней преобразования: для матричного алгоритма количество ступеней преобразования равно четырем, т. е. на единицу меньше числа разрядов сомножителей, а для алгоритма Дадда — равно трем. Но с другой стороны, группировка разрядов по методу Дадда требует более сложного устройства управления операцией умножения.

Матричные методы выполнения операции умножения требуют большего количества оборудования по сравнению с методами по-

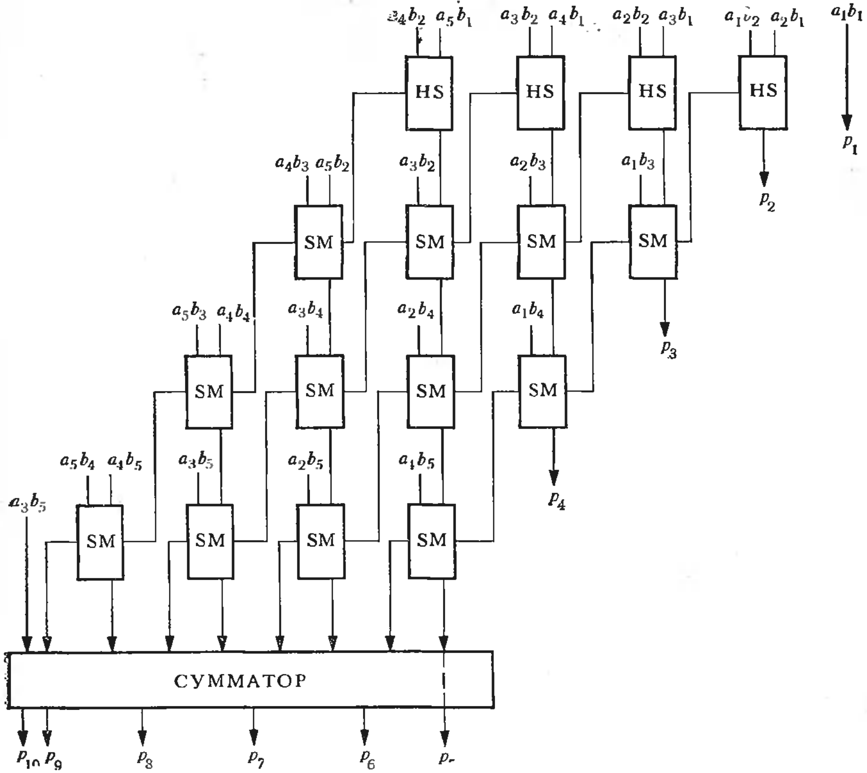


Рис. 4.4. Схема множительного устройства с запоминанием переносов

следовательного анализа разрядов или групп разрядов множителя и дают больший выигрыш во времени. Однако в связи с широким развитием микроэлектронных и особенно больших интегральных схем (БИС) ограничения по количеству оборудования становятся все менее строгими, поэтому рассмотренные выше методы применяют на практике.

Задание для самоконтроля

1. Умножить на сумматоре прямого кода числа $A = -0{,}1100011$ и $B = -0{,}1011101$.

2. Умножить в обратном и дополнительном кодах числа $A = -0{,}11$, $B = -0{,}11$.

3. Преобразовать множитель $B = 0{,}110011101011$ в избыточную двоичную систему $(0, 1, \overline{1})$.

4. Умножить по способу одновременного анализа двух разрядов множителя числа $A = 0{,}1100011101$ и $B = -0{,}1100100011$:

а) начиная с младших разрядов;

б) начиная со старших разрядов.

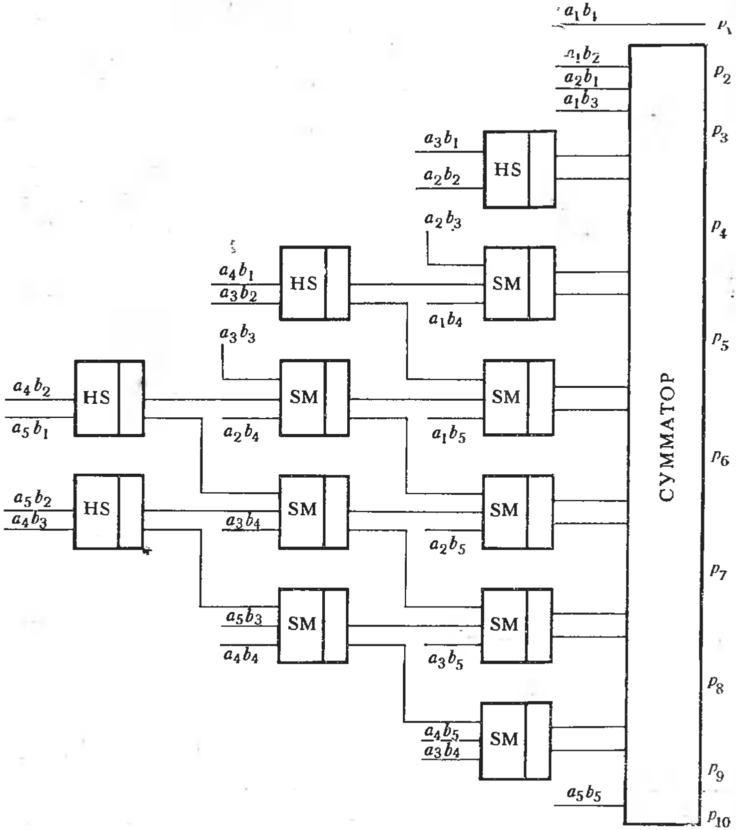


Рис. 4.5. Структурная схема множительного устройства

5. Умножить комбинированным методом с одновременным анализом произвольного количества разрядов множителя числа $A = -0{,}100000001$ и $B = 0{,}011001110$.

6. Умножить, использовав промежуточную систему счисления с $q=2^4$ с одновременным анализом четырех разрядов множителя, числа $A=0{,}11001101$ и $B=0{,}10001100$.

7. Умножить числа $A=0{,}1101101$ и $B=0{,}1000111$, используя метод $AB=2A\dfrac{B}{2}$, если B — четное; $AB = 2A\dfrac{B-1}{2} + A$, если B — нечетное.

8. Умножить числа $A=0{,}100101 \cdot 2^{+6}$ и $B=0{,}110001 \cdot 2^{+7}$, если задана разрядность для мантиссы — семь двоичных разрядов (со знаком) и четыре двоичных разряда (со знаком) для порядка.

5 ГЛАВА

ДЕЛЕНИЕ ДВОИЧНЫХ ЧИСЕЛ

§ 5.1. Методы выполнения операции деления

Из существующего множества разных методов выполнения операции деления рассмотрим наиболее распространенные.

Прежде всего это так называемый «школьный» алгоритм деления, заключающийся в том, что делитель на каждом шаге вычитается столько раз из делимого (начиная со старших разрядов), сколько это возможно для получения наименьшего положительного остатка. Тогда в очередной разряд частного записывается цифра, равная числу делителей, содержащихся в делимом на данном шаге. Таким образом, весь процесс деления сводится к операциям вычитания и сдвига.

Другой метод выполнения операции деления заключается в умножении делимого на обратную величину делителя:

$$C=\frac{A}{B}=A\frac{1}{B}. \tag{5.1}$$

Здесь возникает новая операция — нахождение обратной величины,— осуществляемая по известным приближенным формулам (например, разложением в биномиальный ряд Ньютона и т. п.). В этом случае в состав команд машины должна входить специальная операция нахождения обратной величины.

К наиболее распространенным методам выполнения операции деления относится также метод, заключающийся в использовании приближенной формулы для нахождения частного от деления двух величин. От метода умножения делимого на обратную величину он отличается только тем, что частное определяется по некоторой формуле, которая сводится к выполнению операций сложения, вычитания и умножения.

Фактически два последних метода пригодны для создания специализированных машин, в которых операция деления встречается не часто. В этом случае ее целесообразно реализовать программным путем. Поэтому в универсальных вычислительных машинах, как правило, реализуется разновидность «школьного» алгоритма деления.

В общем случае «школьный» алгоритм деления на примере двоичных чисел выглядит следующим образом:

$$
\begin{array}{rr|l}
\text{делимое} - & -\;1100100 & 10\,10 - \text{делитель} \\
\text{делитель} - & 1010 & 10\,10 - \text{частное} \\
\text{остаток} - & -\;00101 & \\
 & 1010 & \\
 & -\;1011 & \\
 & + & \\
\text{восстановление остатка} & 1010 & \\
 & -\;01010 & \\
 & 1010 & \\
 & 0000 &
\end{array}
$$

Здесь цифры частного получаются последовательно начиная со старшего разряда путем вычитания делителя из делимого на первом шаге, а затем делителя из полученного остатка. Если получен положительный остаток, то цифра частного равна единице, если остаток отрицательный, то цифра частного равна нулю, при этом восстанавливается предыдущий положительный остаток.

В случае положительного остатка для получения следующей цифры частного последний остаток сдвигается влево на один разряд (либо делитель вправо на один разряд) и из него вычитается делитель и т. д.

В случае отрицательного остатка восстанавливается предыдущий положительный остаток прибавлением к отрицательному остатку делителя и восстановленный остаток сдвигается на один разряд влево (либо сдвигается делитель вправо на один разряд) и из него вычитается делитель. Такой алгоритм деления получил название **алгоритма деления с восстановлением остатка.** Формально все действия алгоритма можно описать следующим образом.

Пусть A — делимое, B — делитель и C — частное, при этом $A=0,\, a_1 a_2 \dots a_n$; $B=0,\, b_1 b_2 \dots b_n$; $C=0,\, c_1 c_2 \dots c_n$.

Тогда на первом шаге определяется остаток:

$$A_1 = A - B2^{-1},$$

если $A_1 \geqslant 0$, то $C_1 = 1$;
если же $A_1 < 0$, то $C_1 = 0$ и восстанавливается предыдущее число.
Пусть $A_1 > 0$. Тогда процесс продолжается:

$$A_2 = A_1 - B2^{-2}.$$

Пусть $A_2 < 0$, $C_2 = 0$. Тогда производится восстановление остатка:

$$A_1 = A_2 + B2^{-2}.$$

Этот остаток принимается за A_2' и процесс продолжается:

$$A_3 = A_1 - B2^{-3}$$

и т. д.

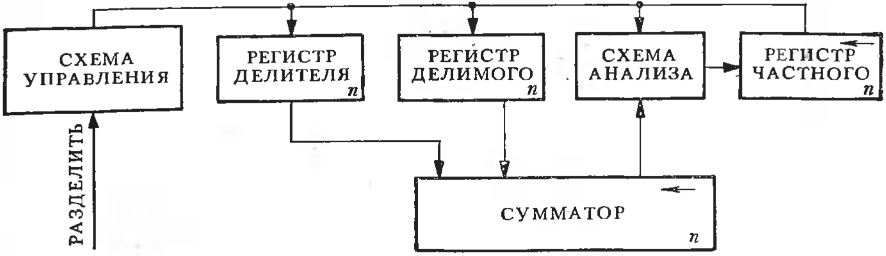


Рис. 5.1. Структурная схема реализации операции деления

В общем виде действия на любом i-м шаге можно описать следующим образом:

$$A_i = A_{i-1} - B2^{-i}, \tag{5.2}$$

если $A_i \geqslant 0$, то $C_i = 1$, производятся соответствующие сдвиги и снова возвращение к (5.2);

если $A_i < 0$, то $C_i = 0$ и восстанавливается остаток

$$A_{i-1} = A_i + B2^{-i}; \tag{5.3}$$

на следующем шаге процесс продолжается по (5.2) при условии, что предыдущий положительный остаток получен по (5.3).

Алгоритм операции деления, основанный на применении сформулированных выше правил, выраженных (5.2) и (5.3), может быть реализован на сумматорах обратного или дополнительного кода.

Таким образом, в процессе операции деления должны участвовать абсолютные величины делимого и делителя.

Состав блоков, необходимый для выполнения операции деления, определяется следующим образом. Так как частное получается в результате последовательного выполнения вычитаний, то необходим сумматор для алгебраического сложения. Как известно, для этих целей применимы сумматоры либо обратного, либо дополнительного кода. Необходимы также регистры для хранения делителя и частного. Делимое можно сохранять либо в специальном регистре, либо в сумматоре. Для реализации последовательной схемы получения цифр частного можно предложить следующую структурную схему, показанную на рис. 5.1. Всеми действиями такой схемы управляет местный блок управления, функционирование которого осуществляется по описанным ниже алгоритмам.

§ 5.2. Деление чисел, представленных в форме с фиксированной запятой, с восстановлением остатков

При выполнении деления с восстановлением остатков прежде всего необходимо определить правила образования знака результата (частного от деления).

Если обозначить знаки чисел A и B, Sg_A и Sg_B, то знак частного Sg_C получится по формуле

$$\mathrm{Sg}_C = \mathrm{Sg}_A \oplus \mathrm{Sg}_B.$$

Цифровые части делимого и делителя должны быть представлены в дополнительном или обратном коде, и над ними выполняются следующие действия:

Знак делимого A	+	+	—	—
Знак делителя B	+	—	+	—
Что делать	$A+\overline{B}$	$A+B$	$A+B$	$A+\overline{B}$

Здесь символ $\overline{B}$ указывает на изменение знака операнда на противоположный.

Таким образом, если знаки делимого и делителя разные, то производится сложение их машинных изображений; если знаки одинаковые — вычитание их машинных изображений.

При делении чисел, представленных в форме с фиксированной запятой, возможно переполнение разрядной сетки. Переполнение возникает тогда, когда $|A| \geqslant |B|$. Поэтому операция деления чисел, представленных в форме с фиксированной запятой, выполняется только тогда, когда $|A| < |B|$, т. е. первым действием в начале процесса всегда будет сдвиг сумматора и регистра делителя.

Рассмотрим реализацию алгоритма деления двух чисел с восстановлением остатков.

Пример 5.1. Разделить числа $A = -0{,}100111$ и $B = -0{,}110001$. Используется структурная схема, показанная на рис. 5.1.

Р е ш е н и е. Операция производится на сумматоре обратного кода. Прежде всего записываем машинные изображения чисел:

$$[A]^{\text{м}}_{\text{об}} = 11{,}011000;$$

$$[B]^{\text{м}}_{\text{об}} = 11{,}001110.$$

Последовательность выполнения действий над числами показана в табл. 5.1.
Ответ: $C = 0{,}110010$.

Операция деления относится к разряду неточных операций, поскольку результат, как правило, получается с некоторой погрешностью. Поэтому признаком окончания операции деления может быть либо достижение заданной точности (количества разрядов в частном), либо получение очередного остатка, равного нулю. Как правило, формальным признаком конца операции деления принимается количество сдвигов: при достижении числа сдвигов, равного количеству разрядов в частном, вырабатывается сигнал окончания операции деления.

Требуется несколько интерпретировать правила определения цифры частного. В примере 5.1 производится сравнение знаков делимого и остатка на каждом шаге: при совпадении знаков в частном записывается единица, при несовпадении — нуль. Можно сравнивать знаки делителя и остатка, тогда единица в очередной раз-

Таблица 5.1

Сумматор (СМ)	Регистр (Рг C)	Примечание
11011000	000000	И. П. СМ$:=[A]^{\text{м}}_{\text{об}}$; Рг $B:=[B]^{\text{м}}_{\text{об}}$; Рг $C:=0$;
10110001	00000—	сдвиг сумматора и регистров
+00110001		СМ$:=[\text{СМ}]+[\overline{B}]^{\text{м}}_{\text{об}}$;
11100010	000001	$c_1=1$;
11000101	00001—	сдвиг сумматора и регистров
+00110001		СМ$:=[\text{СМ}]+[\overline{B}]^{\text{м}}_{\text{об}}$;
11110110	000011	$c_2=1$;
11101101	00011—	сдвиг сумматора и регистров
+00110001		СМ$:=[\text{СМ}]+[\overline{B}]^{\text{м}}_{\text{об}}$;
00011111	000110	$c_3=0$;
+11001110		восстановление остатка
11101101		СМ$:=[\text{СМ}]+[B]^{\text{м}}_{\text{об}}$;
11011011	00110—	сдвиг сумматора регистров
+00110001		СМ$:=[\text{СМ}]+[\overline{B}]^{\text{м}}_{\text{об}}$;
00001101	001100	$c_4=0$;
+11001110		восстановление остатка СМ$:=[\text{СМ}]+[B]^{\text{м}}_{\text{об}}$;
11011011		
10110111	01100—	сдвиг сумматора и регистров
+00110001		СМ$:=[\text{СМ}]+[\overline{B}]^{\text{м}}_{\text{об}}$;
11101000	011001	$c_5=1$;
11010001	11001—	сдвиг сумматора и регистров
+00110001		СМ$:=[\text{СМ}]+[\overline{B}]^{\text{м}}_{\text{об}}$;
00000011	110010	$c_6=0$;
+11001110		восстановление остатка СМ$:=[\text{СМ}]+[B]^{\text{м}}_{\text{об}}$
11010001	110010	Конец

ряд частного записывается при несовпадении знаков, а нуль — при совпадении.

Затраты времени на выполнение операции деления можно определить по формуле

$$t_{\text{дел}}=n\,[t_{\text{сдв}}+(1+P)\,t_{\text{сл}}], \tag{5.4}$$

где n — длина разрядной сетки; $t_{\text{сдв}}$ — время выполнения сдвига на один разряд; $t_{\text{сл}}$ — время выполнения операции сложения; P — вероятность появления нулей в разрядах частного.

Формула (5.4) справедлива для сумматоров обратного и дополнительного кодов для алгоритма деления с восстановлением остатков.

§ 5.3. Деление чисел, представленных в форме с фиксированной запятой, без восстановления остатков

В рассмотренном ранее алгоритме в случае, когда очередная цифра частного C_i была равна нулю, производилась операция восстановления предыдущего положительного остатка по (5.3). Восстановленный остаток принимался за A_i' и процесс деления продолжался, т. е.

$$A_{i+1}=A_i'-B2^{-(i+1)}=A_i+B2^{-i}-B2^{-(i+1)}.$$

Произведя несложные преобразования в этой формуле, получим

$$A_{i+1}=A_i+B2^{-(i+1)}. \qquad (5.5)$$

Из (5.5) следует, что восстанавливать остаток не надо. Таким образом, появилась возможность реализовать алгоритм операции деления иначе, т. е. на первом шаге производится вычитание делителя из остатка по (5.2): если $A_i \geqslant 0$, то $C_i=1$ и на следующем шаге снова используется (5.2); если $A_i<0$, то $C_i=0$ и на следующем шаге после сдвигов используется (5.5). Этот алгоритм получил название **алгоритма деления без восстановления остатков.** Рассмотрим его работу на примере 5.2.

Пример 5.2. Разделить числа $A=0{,}10011$ и $B=-0{,}11001$, используя структурную схему устройства деления, представленную на рис. 5.1.

Решение. Для выполнения операции применен сумматор дополнительного кода. Прежде всего записываются машинные изображения чисел:

$$[A]_{\text{д}}^{\text{м}}=00{,}10011;$$

$$[B]_{\text{д}}^{\text{м}}=11{,}00111.$$

Последовательность выполнения действий при операции деления показаны в табл. 5.2.

Ответ: $C=-0{,}11000$.

Здесь очередная цифра частного определялась сравнением знаков делимого и остатка. Однако часто это бывает не просто сделать: для этого надо сохранять знак делимого. Нахождение цифры частного можно упростить, если воспользоваться правилом:

Знак делимого A	+	+	—	—
Знак делителя B	+	—	+	—
Что делать	$\overline{A}+B$	$\overline{A}+\overline{B}$	$A+B$	$A+\overline{E}$

Следовательно, использованная здесь операция перемены знака позволяет определять цифру частного так: если остаток имеет

Таблица 5.2

Сумматор (СМ)	Регистр (Рг С)	Примечание
0010011	00000	И. П. СМ:$=[A]^{м}_{д}$; Рг $B:=[B]^{м}_{д}$; Рг $C:=0$;
0100110	0000—	сдвиг сумматора и регистров
+ 1100111		СМ:$=$[СМ]$+[B]^{м}_{д}$;
0001101	00001	$c_1=1$;
0011010	0001—	сдвиг сумматора и регистров
+ 1100111		СМ:$=$[СМ]$+[B]^{м}_{д}$;
0000001	00011	$c_2=1$;
0000010	0011—	сдвиг сумматора и регистров
+ 1100111		СМ:$=$[СМ]$+[B]^{м}_{д}$;
1101001	00110	$c_3=0$;
1010010	0110—	сдвиг сумматора и регистров
+ 0011001		СМ:$=$[СМ]$+[\overline{B}]^{м}_{д}$;
1101011	01100	$c_4=0$;
1010110	1100—	сдвиг сумматора и регистров
+ 0011001		СМ:$=$[СМ]$+[\overline{B}]^{м}_{д}$;
1101111	11000	$c_5=0$
		Конец

отрицательный знак, значит очередная цифра частного равна единице, если знак остатка положительный — значит $C_i=0$ (т. е. переписывается содержимое знакового разряда сумматора на каждом шаге в очередной разряд регистра частного). Алгоритм должен работать так, что если делитель равен 0 или делимое равно максимальному отрицательному числу, то вырабатывается сигнал переполнения. Алгоритм деления без восстановления остатков используется в некоторых моделях машин Единой системы.

Рассмотрим пример, где цифры частного образуются с использованием рассмотренного выше правила.

Пример 5.3. Разделить на сумматоре обратного кода числа $A=0{,}10011$ и $B=$ $=-0{,}11001$.

Решение. В соответствии с правилами (см. с. 98) проводится перемена знаков:

$$[\overline{A}]^{м}_{об}=11{,}01100;$$

$$[\overline{B}]^{м}_{об}=00{,}11001.$$

Операция деления осуществляется в последовательности, указанной в табл. 5.3.

Ответ: $C=-0{,}11000$.

Сумматор (СМ)	Регистр (Рг С)	Примечание
1101100	00000	И. П.: СМ:$=[\bar{A}]^{м}_{об}$; Рг B:$=[\bar{B}]^{м}_{об}$; Рг:=0;
1011001	0000—	сдвиг сумматора и регистров
+ 0011001		СМ:=[СМ]+$[\bar{B}]^{м}_{об}$;
1110010	00001	$c_1=1$;
1100101	0001—	сдвиг сумматора и регистров
+ 0011001		СМ:=[СМ]+$[\bar{B}]^{м}_{об}$;
1111110	00011	$c_2=1$;
1111101	0011—	сдвиг сумматора и регистров
+ 0011001		СМ:=[СМ]+$[B]^{м}_{об}$;
0010111	00110	$c_3=0$;
0101110	0110—	сдвиг сумматора и регистров
+ 1100110		СМ:=[СМ]+$[B]^{м}_{об}$;
0010101	01100	$c_4=0$;
0101010	1100—	сдвиг сумматора и регистров
+ 1100110		СМ:=[СМ]+$[B]^{м}_{об}$;
0010001	11000	$c_5=0$; Конец

§ 5.4. Особенности деления чисел, представленных в форме с плавающей запятой

Для получения частного от деления двух чисел, представленных в форме с плавающей запятой, необходимо выполнить следующие действия:

$$m_C=m_A/m_B \quad \text{и} \quad p_C=p_A-p_B.$$

Так как мантиссы делимого и делителя являются нормализованными числами, то при делении возможны случаи, когда: 1) $|m_A|\geqslant|m_B|$; 2) $|m_A|<|m_B|$.

Если мантисса делимого больше или равна мантиссе делителя, то в конце операции деления потребуется нормализация частного (нарушение правой границы). Таким образом, алгоритм деления начинается с операции вычитания делителя из делимого и записи единицы в целую часть частного. Все остальные действия над мантиссами аналогичны действиям над числами, представленными в форме с фиксированной запятой.

Если мантисса делимого меньше мантиссы делителя, то после операции вычитания на первом шаге получится отрицательный остаток, что означает нуль в целой части мантиссы частного и продолжение алгоритма деления по рассмотренным выше правилам для чисел, представленных в форме с фиксированной запятой. Таким образом, частное всегда получается в прямом коде, а действия над мантиссами осуществляются на ДСОК или ДСДК.

Так как при операции деления порядки чисел складываются, то возможно переполнение разрядной сетки в сумматоре порядков. При переполнении в сторону отрицательных величин мантисса результата превращается в машинный нуль, а порядку присваивается наибольшее отрицательное значение. Если делитель равен нулю, необходимы выработка сигнала $\varphi=1$ и останов машины. Эти частные случаи предусмотрены при реализации алгоритмов в ЕС ЭВМ, где используется алгоритм деления без восстановления остатков. При этом мантисса делимого имеет длину в два раза больше, чем делитель, что иллюстрируется примером 5.4.

Пример 5.4. Разделить числа $A=0{,}10001111\cdot 2^3$ и $B=0{,}1111\cdot 2^2$.

Решение. Рассматривается случай, когда $|m_A|<|m_B|$.

Прежде всего записываются машинные изображения мантиссы делимого $[m_A]^{\text{м}}_{\text{д}}=00{,}10001111$; мантиссы делителя $[m_B]^{\text{м}}_{\text{д}}=00{,}1111$ и $[\overline{m}_B]^{\text{м}}_{\text{д}}=11{,}0001$.

Все действия выполняются на сумматоре дополнительного кода в последовательности, указанной в табл. 5.4.

Таблица **5.4**

Сумматор (СМ)	Регистр (Рг C)	Примечание
00,10001111	0000	И. П. СМ:=$[m_A]$; Рг C:=0; РгB:=$[m_B]$
01,00011110	000—	сдвиг сумматоров и регистров
+ 11,0001		СМ:=[СМ]+$[\overline{m}_B]$;
00,00101110	0001	$c_1=1$
00,01011100	001—	сдвиг сумматоров и регистров
+ 11,0001		СМ:=[СМ]+$[\overline{m}_B]$;
11,01101100	0010	$c_2=0$
10,11011000		сдвиг сумматоров и регистров
+ 00,1111		СМ:=[СМ]+$[m_B]$;
11,11001000	0100	$c_3=0$
11,10010000		сдвиг сумматоров и регистров
+ 00,1111		СМ:=[СМ]+$[m_B]$;
00,10000000		$c_4=1$ Конец

Одновременно вычисляется порядок частного следующим образом:

$$p_C=p_A-p_B=0{,}011-0{,}010=0{,}001$$

и определяется знак частного $Sg_C=Sg_A+Sg_B=0+0=0$.

Ответ: $C=0{,}1001\cdot 2^1$.

§ 5.5. Ускорение операции деления

Идея метода ускорения операции деления заключается в том, что в случае образования достаточно малого или достаточно большого по абсолютной величине остатка очередные цифры частного будут группой одинаковых цифр — либо нулей, либо единиц, поэтому продолжение процесса деления обычным способом излишне, так как эту группу цифр можно записать в частное сразу.

Сказанное подтверждается примером деления десятичных и двоичных чисел:

$$
1)\quad \begin{array}{r|l}
2281825 & 2275 \\
\underline{-\,2275} & \overline{1\underbrace{00}_{k-1}3} \\
\underbrace{000}_{k}6825 & \\
\underline{-\,6825} & \\
0000 &
\end{array}
\qquad\qquad
2)\quad \begin{array}{r|l}
1010001 & 1001 \\
\underline{-\,1001} & \overline{1\underbrace{00}_{k-1}1} \\
\underbrace{000}_{k}1001 & \\
\underline{-\,1001} & \\
0000 &
\end{array}
$$

Здесь на первом шаге получился малый по величине положительный остаток, который содержит нули в трех старших разрядах. В частном после первой цифры записаны нули в два следующих разряда. Можно утверждать, что при малом положительном остатке в частное можно сразу записать k—1 нулей в соответствующие разряды, остаток сдвинуть на k разрядов влево, вычесть из него делитель и далее продолжить операцию деления.

Если в результате вычитания на каком-то шаге получается большой положительный остаток, в котором k старших разрядов содержат единицы (для двоичных чисел), то по меньшей мере в (k—1)-е разряды частного можно записать единицу:

$$
\begin{array}{l|l}
-\;1000000000\ldots & 10001 \\
\underline{10001} & \overline{1\underbrace{1110}_{k-1}} \\
0\underbrace{111}_{k}10 & \\
-\underline{10001} & \\
011010 & \\
-\underline{10001} & \\
010010 & \\
-\underline{10001} & \\
000001\ldots &
\end{array}
$$

Примечание. Эта идея была предложеиа группой советских иииженеров (Э. И. Клямко и др.).

Таким образом, в данном случае процесс деления должен осуществляться комбинированным способом, с анализом последовательности нулей (если остаток положительный) или последовательности единиц (если остаток отрицательный) в старших разрядах остатка. При обнаружении такой последовательности длиной k сразу можно записать в $(k-1)$-е разряды частного соответственно нули или единицы. После этого произвести сдвиг на $(k-1)$-й разряд и продолжить операцию деления.

В других случаях операция выполняется по обычному алгоритму.

Естественно, что рассмотренный метод ускорения операции деления может дать эффект только тогда, когда встречаются последовательности нулей или единиц в остатках.

Для ускорения операции деления можно воспользоваться также методом одновременного определения двух цифр частного. Этот метод реализован в некоторых ЭВМ третьего поколения (например, «Иллиак-III») и является развитием изложенной выше идеи.

Пусть A — делимое; B — делитель; A_{i-1} — остаток на $(i-1)$-м шаге деления.

Для остатка справедливо

$$0 \leqslant |A_{i-1}| < B.$$

Если теперь остаток сдвинуть на два разряда, то $2^2 A_{i-1}$. Тогда следующую пару цифр частного можно найти из условий, представленных в табл. 5.5.

Таблица 5.5

Условие для анализа	Пара цифр частного	A_i
$2^2 A_{i-1} < B$	00	$2^2 A_{i-1}$
$B \leqslant 2^2 A_{i-1} < 2B$	01	$2^2 A_{i-1} - B$
$2B \leqslant 2^2 A_{i-1} < 3B$	10	$2^2 A_{i-1} - 2B$
$3B \leqslant 2^2 A_{i-1}$	11	$2^2 A_{i-1} - 3B$

Изложенный алгоритм деления может быть упрощен, если каждую пару цифр частного определять исходя из анализа нескольких старших разрядов делителя B и сдвинутого остатка $2^2 A_{i-1}$. При этом если старшие разряды остатка $2^2 A_{i-1}$ совпадают со старшими разрядами групп, предполагается, что $2^2 A_{i-1}$ принадлежит к области с наибольшим значением пары цифр частного и исходя из этого находится остаток A_i. Если A_i — отрицательный, это значит, что получена величина остатка, уменьшенная на B:

$$-B \leqslant A'_i = (A_i - B) < 0. \tag{5.6}$$

Тогда коррекция остатка осуществляется следующим образом: если раньше проверялось условие $jB \leqslant 2^2 A_{i-1} \leqslant (j+1)B$, где $j = 0, 1, 2, 3$, то теперь проверяется эквивалентное ему условие

$$(j-4)B \leqslant 2^2(A_{i-1} - B) < (j-3)B. \tag{5.7}$$

Окончательно можно сформулировать правила, представленные в табл. 5.6.

Таблица 5.6

$A_{i-1} \geqslant 0$		$A'_{i-1} = A_{i-1} - B < 0$		Пара цифр частного	
условие для анализа	A_i	условие для анализа	A_i	$A_{i-1} \geqslant 0$	$A'_{i-1} < 0$
$2^2 A_{i-1} < B$	$2^2 A_{i-1}$	$2^2 A'_{i-1} < -3A$	$2^2 A_{i-1} + 4B$	00	—
$B \leqslant 2^2 A_{i-1} < 2B$	$2^2 A_{i-1} - B$	$-3A \leqslant 2^2 A'_{i-1} < -2A$	$2^2 A_{i-1} + 3B$	01	00
$2B \leqslant 2^2 A_{i-1} < 3B$	$2^2 A_{i-1} - 2B$	$-2A \leqslant 2^2 A_{i-1} < -A$	$2^2 A_{i-1} + 2B$	10	01
$3B \leqslant 2^2 A_{i-1}$	$2^2 A_{i-1} - 3B$	$-A \leqslant 2^2 A_{i-1} < 0$	$2^2 A_{i-1} + B$	11	10

§ 5.6. Операция извлечения квадратного корня

Операция извлечения квадратного корня как самостоятельная операция в систему команд ЭВМ включается в случае, когда приходится относительно часто прибегать к ее выполнению (не менее 2% от общего числа операций). Такие ситуации достаточно часто могут встречаться при создании специализированных ЭВМ.

Основным приемом для приближенного вычисления квадратного корня в универсальных ЭВМ является метод итераций, основанный на формуле Ньютона.

Пусть $y = \sqrt{x}$. Тогда $F(x, y) = y^2 - x = 0$, $F'(x, y) = 2y$. Считая, что $y_n \approx y$ есть приближенное значение искомой функции, по теореме Лагранжа можно определить

$$F(x, y_n) = F(x, y_n) - F(x, y) = (y_n - y) F'_y(x, \overline{y}_n),$$

где $\overline{y}_n$ — некоторое промежуточное значение между y_n и y. Тогда

$$y_{n+1} = y_n - (y_n^2 - x)/2y_n,$$

или

$$y_{n+1} = 0.5\,(y_n + x/y_n), \tag{5.8}$$

где $n = 0, 1, 2, \ldots$

Формула (5.8) может быть положена в основу алгоритма вычисления квадратного корня.

На практике используют и несколько иной метод. Пусть задано подкоренное число $A = 0, a_1 a_2 \ldots a_n$. Предположим, что удалось найти $(k-1)$-ю цифру значения корня, равную $0, b_1 b_2 \ldots b_{k-1}$. По условию, очередной остаток $A_{k-1} = A - (0, b_1 b_2 \ldots b_{k-1})^2 > 0$. Очередная цифра b_k может быть нуль или единица. Очевидно, что если $A_k > 0$, то $b_k = 1$. При этом

$$A_k = A - (0,\ b_1 b_2 \ldots b_{k-1} 1)^2 = A - (0,\ b_1 b_2 \ldots b_{k-1})^2 - \\ - 2 \cdot 2^{-k} \cdot 0,\ b_1 b_2 \ldots b_{k-1} - 2^{-2k},$$

или

$$A_k = A_{k-1} - 2^{-(k-1)} \cdot 0,\ b_1 b_2 \ldots b_{k-1} 01. \tag{5.9}$$

Таблица 5.7

Сумматор (СМ)	Регистр (Рг B)	Примечание
00,100101	000000	И. П. СМ:$=[A]^{\text{м}}_{\text{доп}}$; Рг B:$=0$;
+ 11,110000		СМ:$=$[СМ]$+[-0{,}01]_{\text{д}}$;
00,010101	000001	$b_1=1$;
00,101010	00001—	сдвиг
+ 11,011000		СМ:$=$[СМ]$+[-0{,}101]_{\text{д}}$;
00,000010	000011	$b_2=1$;
00,000100	00011—	сдвиг
+ 11,001100		СМ:$=$[СМ]$+[-0{,}1101]_{\text{д}}$;
11,010000	000110	$b_3=0$;
+ 00,110100		восстановление остатка
00,000100		
00,001000	00110—	сдвиг
+ 11,001110		СМ:$=$[СМ]$+[-0{,}11001]_{\text{д}}$;
11,010110	001100	$b_4=0$;
+ 00,110010		восстановленне остатка
00,001000		
00,010000	01100—	сдвиг
+ 11,001111		СМ:$=$[СМ]$+[-0{,}110001]_{\text{д}}$;
11,011111	011000	$b_5=0$;
+ 00,110001		восстановление остатка
00,010000		
00,100000	11000—	сдвиг
+ 11,001111		СМ:$=$[СМ]$+[-0{,}1100001]_{\text{д}}$;
11,101111	110000	$b_6=0$
		Конец

Следовательно, для получения очередного остатка надо к уже найденному числу $0, b_1b_2 \ldots b_{k-1}$ приписать пару цифр 01 и вычесть полученное число, предварительно сдвинутое на $(k-1)$-й разряд, из предыдущего остатка. При этом если $A_k \geqslant 0$, то $b_k=1$; если $A_k<0$, то $b_k=0$.

Таким образом, (5.9) показывает, что операция извлечения квадратного корня напоминает операцию деления на переменный делитель, равный $0, b_1b_2 \ldots b_{k-1}01$. Первое значение переменного делителя равно 0,01.

Пример 5.5. Извлечь квадратиый корень из числа $A=0,100101$ на сумматоре дополнительного кода.

Решение. См. табл. 5.7.

Ответ: $B=\sqrt{A}=0,110000$.

При обращении с отрицательными числами следует вырабатывать сигнал переполнения, который покажет нарушение правильного прохождения алгоритма.

Задание для самоконтроля

1. Разделить заданные в прямом коде числа:
а) $[A]_{пр}=1,100011$ и $[B]_{пр}=1,110011$; б) $[A]_{пр}=0,100111$ и $[B]_{пр}=1,100011$.

2. Разделить заданные в обратном коде методом с восстановлением остатков числа $[A]_{об}=1,011001$ и $[B]_{об}=0,11001$.

3. Разделить заданные в дополнительном коде методом без восстановления остатков числа $[A]_{доп}=0,110000$ и $[B]_{доп}=1,000111$.

4. Найти частное от деления чисел $[A]_{об}=0,1010001111$ и $[B]_{об}=1,1010110010$ чисел с использованием ускоренного метода.

5. Можно ли применить правила двоичного деления к делению чисел, представленных в минус-двоичной системе?

6. Произвести на сумматоре обратного кода деление чисел $A=-0,10011\cdot 2^{-6}$ и $B=0,100101\cdot 2^{-4}$, представленных в форме с плавающей запятой.

7. Произвести деление на сумматоре дополнительного кода чисел $A=0,101101\cdot 2^{5}$ и $B=-0,111001\cdot 2^{3}$, представленных в форме с плавающей запятой.

8. Возможно ли переполнение разрядной сетки при делении чисел, представленных в форме с плавающей запятой?

9. Извлечь квадратный корень из числа $A=0,111111$ на сумматоре обратного кода.

6 ГЛАВА

ИНФОРМАЦИОННЫЕ ОСНОВЫ КОНТРОЛЯ РАБОТЫ ЦИФРОВОГО АВТОМАТА

§ 6.1. Общие положения

Рассмотренные ранее алгоритмы выполнения арифметических операций обеспечат правильный результат только в случае, если машина работает без нарушений. При возникновении какого-либо нарушения нормального функционирования результат будет неверным, однако пользователь об этом не узнает, если не будут предусмотрены меры, сигнализирующие о появлении ошибки. Следовательно, с одной стороны, разработчиками машины должны быть предусмотрены меры для создания системы обнаружения возможной ошибки, а с другой стороны, должны быть проработаны меры, позволяющие исправить ошибки. Эти функции следует возложить на систему контроля работы цифрового автомата.

Система контроля — *совокупность методов и средств, обеспечивающих определение правильности работы автомата в целом или его отдельных узлов, а также автоматическое исправление ошибки.*

Ошибки в работе цифрового автомата могут быть вызваны либо выходом из строя какой-то детали, либо отклонением от нормы параметров (например, изменение напряжения питания) или воздействием внешних помех. Вызванные этими нарушениями ошибки могут принять постоянный или случайный характер. Постоянные ошибки легче обнаружить и выявить. Случайные ошибки, обусловленные кратковременными изменениями параметров, наиболее опасны и их труднее обнаружить.

Поэтому система контроля должна строиться с таким расчетом, чтобы она позволяла обнаружить и по возможности исправить любые нарушения. При этом надо различать следующие виды ошибок результата:

1) возникающие из-за погрешностей в исходных данных;
2) обусловленные методическими погрешностями;
3) появляющиеся из-за возникновения неисправностей в работе машины.

Первые два вида ошибок не являются объектом для работы системы контроля. Конечно, погрешности перевода или представления числовой информации в разрядной сетке автомата приведут к возникновению погрешности в результате решения задачи. Эту погрешность можно заранее рассчитать и, зная ее максимальную величину, правильно выбрать длину разрядной сетки машины. Методические погрешности также учитываются предварительно.

Проверка правильности функционирования отдельных устройств машины и выявление неисправностей может осуществляться по двум направлениям:

профилактический контроль, задача которого — предупреждение появления возможных ошибок в работе;

оперативный контроль, задача которого — проверка правильности выполнения машиной всех операций.

Решение всех задач контроля становится возможным только при наличии определенной **избыточности.** Избыточность может быть создана либо аппаратными (схемными) средствами, либо логическими или информационными средствами.

Примечание. Схемная избыточность будет рассматриваться в дисциплинах, в которых изучаются устройства ЭВМ. В данной главе описаны методы логического контроля, использующие информационную избыточность.

К методам логического контроля, например, можно отнести следующие приемы. В ЭВМ первого и второго поколений отсутствие системы оперативного контроля приводило к необходимости осуществления «двойного счета», когда каждая задача решалась дважды и в случае совпадения ответов принималось решение о правильности функционирования ЭВМ.

Если в процессе решения какой-то задачи вычисляются тригонометрические функции, то для контроля можно использовать известные соотношения между этими функциями, например $\sin^2\alpha + \cos^2\alpha = 1$. Если это соотношение выполняется с заданной точностью на каждом шаге вычислений, то можно с уверенностью считать, что ЭВМ работает правильно.

Вычисление определенного интеграла с заданным шагом интегрирования можно контролировать сравнением полученных при этом результатов с теми результатами, которые соответствуют более крупному шагу. Такой «сокращенный» алгоритм даст, видимо, более грубые оценки и, по существу, требует дополнительных затрат машинного времени.

Все рассмотренные примеры свидетельствуют о том, что такие методы контроля позволяют лишь зафиксировать факт появления ошибки, но не определяют место, где произошла эта ошибка. Для оперативного контроля работы ЭВМ определение места, где произошла ошибка, т. е. решение задачи поиска неисправности, является весьма существенным вопросом.

§ 6.2. Систематические коды

Как уже указывалось, функции контроля можно осуществлять при информационной избыточности. Такая возможность появляется при использовании специальных методов кодирования информации. В самом деле, некоторые методы кодирования информации допускают наличие разрешенных и запрещенных комбинаций. В качестве примера можно привести двоично-десятичные системы представления числовой информации (Д-коды). Появление запрещенных комбинаций для подобного представления свидетельствует об ошибке в результатах решения задачи. Такой метод можно использовать для контроля десятичных операций. Однако он является частным примером и не решает общей задачи.

Задача кодирования информации представляется как некоторое преобразование числовых данных в заданной системе счисления. В частном случае эта операция может быть сведена к группированию символов (представление в виде триад или тетрад) или представлению в виде символов позиционной системы счисления. Так как любая позиционная система не несет в себе избыточности информации и все кодовые комбинации являются разрешенными, то использовать такие системы для контроля не представляется возможным.

Систематический код — *код, содержащий в себе кроме информационных контрольные разряды.*

В контрольные разряды записывается некоторая информация об исходном числе. Поэтому можно говорить, что систематический код обладает избыточностью. При этом абсолютная избыточность будет выражаться количеством контрольных разрядов k, а относительная избыточность — отношением k/n, где $n=m+k$ — общее количество разрядов в кодовом слове (m — количество информационных разрядов).

Понятие корректирующей способности кода обычно связывают с возможностью обнаружения и исправления ошибки. Количественно корректирующая способность кода определяется вероятностью обнаружения или исправления ошибки. Если имеем n-разрядный код и вероятность искажения одного символа будет P, то вероятность того, что искажены k символов, а остальные $n-k$ символов не искажены, по теореме умножения вероятностей будет

$$W=P^k(1-P)^{n-k}.$$

Число кодовых комбинаций, каждая из которых содержит k искаженных элементов, равна числу сочетаний из n по k:

$$C_n^k=\frac{n!}{k!\,(n-k)!}.$$

Тогда вероятность искажения

$$P_\Sigma=\sum_{i=1}^{k}\frac{n!}{i!\,(n-i)!}P^i(1-P)^{n-i}.$$

Так как на практике $P=10^{-3}\div10^{-4}$, наибольший вес в сумме вероятностей имеет вероятность искажения одного символа. Следовательно, основное внимание нужно обратить на обнаружение и исправление одиночной ошибки.

Корректирующая способность кода связана также с понятием кодового расстояния.

Кодовое расстояние $d(A, B)$ для кодовых комбинаций A и B *определяется как вес такой третьей кодовой комбинации, которая получается сложением исходных комбинаций по модулю 2.*

Примечание. Это определение совпадает с понятием кодового расстояния по Хэмингу. Поэтому в теории кодирования оно называется **хэминговым расстоянием.**

Вес кодовой комбинации $V(A)$ — *количество единиц, содержащихся в кодовой комбинации.*

Пример 6.1. Найти веса и кодовое расстояние для комбинаций $A=100111001$ и $B=011011100$.

Решение. Веса для кодовых комбинаций

$$V(A)=\sum_{i=1}^{i=9} a_i=5 \quad \text{и} \quad V[B]=\sum_{i=1}^{i=9} b_i=5.$$

Находим кодовую комбинацию $C=A\oplus B=111100101$, для которой определяется вес, равный кодовому расстоянию для A и B:

$$V(C)=d(A,\ B)=\sum_{i=1}^{i=9} c_i=6.$$

Ответ: $d(A, B)=6$.

Коды можно рассматривать и как некоторые геометрические (пространственные) фигуры. Например, триаду можно представить в виде единичного куба, имеющего координаты вершин, которые отвечают двоичным символам (рис. 6.1). В этом случае кодовое расстояние воспринимается как сумма длин ребер между соответствующими вершинами куба (принято, что длина одного ребра равна 1). Оказывается, что любая позиционная система отличается тем свойством, что минимальное кодовое расстояние равно 1.

В теории кодирования показано, что систематический код обладает способностью обнаружить ошибки только тогда, когда минимальное кодовое расстояние для него больше или равно $2t$, т. е.

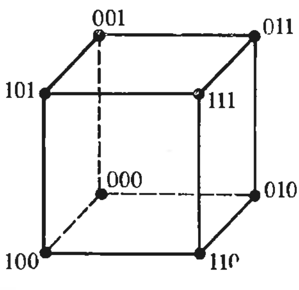


Рис. 6.1. Геометрическое представление кодов

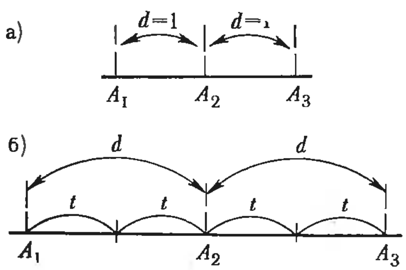


Рис. 6.2. Кодовые расстояния:
а — для позиционной системы; *б* — при избыточном кодировании

$d_{\min} \geqslant 2t$, где t — кратность обнаруживаемых ошибок (в случае обнаружения одиночных ошибок $t=1$). Это означает, что между соседними кодовыми комбинациями должна существовать по крайней мере одна кодовая комбинация (рис. 6.2).

§ 6.3. Кодирование по методу четности—нечетности

Если в математическом коде выделен один контрольный разряд ($k=1$), то к каждому двоичному числу добавляется один избыточный разряд и в него записывается 1 или 0 с таким условием, чтобы сумма цифр в каждом числе была по модулю 2 равна 0 для случая четности или 1 для случая нечетности. Появление ошибки в кодировании обнаружится по нарушению четности (нечетности). При таком кодировании допускается, что может возникнуть только одна ошибка. В самом деле, для случая четности правильным будет только половина возможных комбинаций. Чтобы одна допустимая комбинация превратилась в другую, должно возникнуть по крайней мере два нарушения или четное число нарушений. Пример реализации метода четности представлен в табл. 6.1.

Таблица 6.1

Число	Контрольный разряд	Проверка
10101011	1	0
11001010	0	0
10010001	1	0
11001011	0	1—нарушение

Такое кодирование имеет минимальное кодовое расстояние, равное 2.

Можно представить и несколько видоизмененный способ контроля по методу четности — нечетности. Длинное число разбивается на группы, каждая из которых содержит l разрядов. Контрольные разряды выделяются всем группам по строкам и по столбцам согласно следующей схеме:

$$\begin{array}{ccccc|c}
a_1 & a_2 & a_3 & a_4 & a_5 & k_1 \\
a_6 & a_7 & a_8 & a_9 & a_{10} & k_2 \\
a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & k_3 \\
a_{16} & a_{17} & a_{18} & a_{19} & a_{20} & k_4 \\
a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & k_5 \\
\hline
k_6 & k_7 & k_8 & k_9 & k_{10} &
\end{array}$$

Увеличение избыточности информации приводит к тому, что появляется возможность не только обнаружить ошибку, но и исправить ее. В самом деле, пусть произошла неисправность в каком-то из разрядов этого числа (представим, что разряд a_{18} изменил состояние, т. е. $a_{18}=1$). Это приведет к тому, что при проверке на четность сумма $\sum_i a_i+k_i$ по соответствующим строкам и столбцам изменится для значений, которые содержат элемент a_{18}, т. е. это будет четвертая сверху строка и третий слева столбец. Следовательно, нарушение четности по этой строке и столбцу можно зафиксировать, что в конечном счете означает обнаружение не только самой ошибки, но и места, где возникла ошибка. Изменив содержимое отмеченного разряда (в данном случае a_{18}) на противоположное, можно исправить ошибку.

Пример 6.2. Определить и исправить ошибку в передаваемой информации вида

$$\begin{array}{c|c}
1001110 & 0 \\
1110101 & 0 \\
0101101 & 0 \\
1010110 & 0 \\
1101011 & 1 \\
\hline
0001011 &
\end{array}$$

Для контроля использовать метод четности по строкам и столбцам (контрольный столбец 8, контрольная строка 6).

Решение. Прежде всего осуществим проверку на четность по каждой строке:

$$k_1=0;\quad k_2=1;\quad k_3=0;\quad k_4=0;\quad k_5=0.$$

Затем проверим на четность информацию по столбцам:

$$k_6=0;\quad k_7=1;\quad k_8=0;\quad k_9=0;\quad k_{10}=0;\quad k_{11}=0;\quad k_{12}=0.$$

Проверка показывает, что ошибка произошла в информации второй строки и второго слева столбца. Следовательно, разряд, содержащий ошибочную информацию, находится на пересечении второй строки и второго столбца.

Ответ:

$$\begin{array}{c|c}
1001110 & 0 \\
1010101 & 0 \\
0101101 & 0 \\
1010110 & 0 \\
1101011 & 1 \\
\hline
0001011 &
\end{array}$$

Контроль по методу четности — нечетности широко используют в ЭВМ для контроля записи, считывания информации в запоминающих устройствах на магнитных носителях.

§ 6.4. Коды Хэминга

Коды, предложенные американским ученым Р. Хэмингом, обладают способностью не только обнаружить, но и исправить одиночные ошибки. Эти коды — систематические.

Предположим, что имеется код, содержащий m информационных разрядов и k контрольных разрядов. Запись на k позиций определяется при проверке на четность каждой из проверяемых k групп информационных символов. Пусть было проведено k проверок. Если результат проверки свидетельствует об отсутствии ошибки, запишем 0, если есть ошибка — запишем 1. Запись полученной последовательности символов образует двоичное число.

Свойство кодов Хэминга таково, что контрольное число указывает номер позиции, где произошла ошибка. При отсутствии ошибки в данной позиции последовательность будет содержать только нули. Полученное число таким образом описывает $n=(m+k+1)$ событий.

Следовательно, справедливо неравенство

$$2^k \geqslant (m+k+1). \tag{6.1}$$

Определить максимальное значение m для данного n можно из следующего:

n	1	2	3	4...	8...15	16...31	32...63	64
m	0	0	1	1...	4...11	11...26	26...57	57
k	1	2	2	3...	4...4	5...5	6...6	7

Определим теперь позиции, которые надлежит проверить в каждой из k проверок. Если в кодовой комбинации ошибок нет, контрольное число содержит только нули. Если в первом разряде контрольного числа стоит 1, это означает, что в результате первой проверки обнаружена ошибка. Имея таблицу двоичных эквивалентов для десятичных чисел, можно сказать, что, например, первая проверка охватывает позиции 1, 3, 5, 7, 9 и т. д., вторая проверка — позиции 2, 3, 6, 7, 10.

Проверка	Проверяемые разряды
1...	1,3,5,7,9,11,13,15...
2...	2,3,6,7,10,11,14,15,18,19,22,23.
3...	4,5,6,7,12,13,14,15,20,21,22,23.
4...	8,9,10,11,12,13,14,15,24... .
...	...

Теперь нужно решить, какие из позиций целесообразнее применить для передачи информации, а какие — для ее контроля. Преимущество использования позиций 1, 2, 4, 8, ... для контроля в том, что данные позиции встречаются только в одной проверяемой группе символов.

В табл. 6.2 представлены примеры кодирования информации по методу Хэминга для семиразрядного кода.

Таблица 6.2

Разряды двоичного кода							Кодируемая десятичная информация
1 k_1	2 k_2	3 m_1	4 k_3	5 m_2	6 m_3	7 m_4	
0	0	0	0	0	0	0	0
1	1	0	1	0	0	1	1
0	1	0	1	0	1	0	2
1	0	0	0	0	1	1	3
1	0	0	1	1	0	0	4
0	1	0	0	1	0	1	5
1	1	0	0	1	1	0	6
0	0	0	1	1	1	1	7
1	1	1	0	0	0	0	8
0	0	1	1	0	0	1	9
1	0	1	1	0	1	0	10
0	1	1	0	0	1	1	11
0	1	1	1	1	0	0	12
1	0	1	0	1	0	1	13
0	0	1	0	1	1	0	14
1	1	1	1	1	1	1	15

Как видно из табл. 6.2, в этом случае $n=7$, $m=4$, $k=3$ и контрольными будут разряды 1, 2, 4.

По методу Хэминга могут быть построены коды разной длины. При этом чем больше длина кода, тем меньше относительная избыточность. Например, для контроля числа, имеющего 48 двоичных разрядов, потребуется только шесть дополнительных (избыточных) разрядов. Коды Хэминга используют в основном для контроля передачи информации по каналам связи, что имеет место в вычислительных системах с телеобработкой данных или в системах коллективного пользования.

Пример 6.3. Определить правильность передачи информации $A=0{,}111000$ по каналу, если для контроля использован метод Хэминга.

Решение. Прежде всего определяем контрольное число, производя проверки по правилам, указанным на с. 113:

$$k_1=1;\quad k_2=0;\quad k_3=1.$$

Контрольное число говорит о том, что произошла ошибка в разряде 5.

Ответ: правильная информация $A=0{,}111100$.

§ 6.5. Контроль по модулю

Все рассмотренные выше методы контроля предназначены для решения частных задач.

Более разнообразные задачи позволяет решить метод контроля, основанный на свойствах сравнений. Развитые на этой основе методы контроля арифметических и логических операций называют **контролем по модулю.**

Рассмотрим основные положения из теории сравнений.

Если целым числам A и B соответствует один и тот же остаток от деления на третье число P, то числа A и B равноостаточны друг другу по модулю P или сравнимы по модулю P:

$$A \equiv B \pmod{P}. \tag{6.2}$$

Сравнения — *уравнения типа* (6.2).

Сравнимость двух чисел равносильна возможности представить их в алгебраическом виде

$$A = B + Pl. \tag{6.3}$$

Сравнения обладают рядом свойств:

1. Сравнения можно почленно складывать. Если

$$A_1 \equiv B_1 \pmod{P};$$

$$A_2 \equiv B_2 \pmod{P};$$

$$\cdots\cdots\quad\cdots$$

$$A_n \equiv B_n \pmod{P},$$

то $A_1 + A_2 + \ldots + A_n \equiv B_1 + B_2 + \ldots + B_n \pmod{P}$.

Отсюда следует, что слагаемое, стоящее в какой-либо части сравнения, можно переносить в другую часть, поменяв при этом его знак, т. е.

$$A + B \equiv C \pmod{P}$$

или

$$A \equiv C - B \pmod{P}.$$

2. Два числа, сравнимые с третьим числом, сравнимы и между собой:

если

$$A \equiv B \pmod{P};$$

$$C \equiv B \pmod{P},$$

то

$$A \equiv C \pmod{P}.$$

3. Сравнения можно почленно перемножить. Пусть

$$A_1 \equiv B_1 \pmod{P};$$

$$A_2 \equiv B_2 \pmod{P}.$$

Тогда на основании (6.3)

$$A_1 = B_1 + l_1 P;$$

$$A_2 = B_2 + l_2 P.$$

После умножения получаем

$$A_1A_2 = B_1B_2 + \underbrace{B_1l_2P + B_2l_1P + l_1l_2PP}_{NP}.$$

Следовательно,

$$A_1A_2 = B_1B_2 + NP,$$

или в общем случае:

$$A_1A_2A_3 \ldots A_m \equiv B_1B_2B_3 \ldots B_m \pmod{P}.$$

Из свойства 3 также следует, что обе части сравнения можно умножить на одно и то же целое число.

Пусть

$$A \equiv B \pmod{P};$$

$$K \equiv K \pmod{P}.$$

Тогда

$$AK \equiv BK \pmod{P}.$$

4. Обе части сравнения и модуль можно умножить на одно и то же число:

$$A = B + lP;$$

$$Am = Bm + mlP,$$

т. е.

$$Am \equiv Bm \pmod{mP}.$$

5. Обе части сравнения и модуль можно разделить на любой общий делитель.

Пусть

$$A \equiv B \pmod{P},$$

где $A = ad$; $B = bd$; $P = P_1d$.

Тогда

$$A = B + lP.$$

Подставив в это выражение значения A, B и P, получим

$$ad = bd + lP_1d.$$

Разделив уравнение на d, имеем

$$a = b + lP_1,$$

т. е.

$$a \equiv b \pmod{P_1}.$$

6. Обе части сравнения можно возвести в степень. Если

$$A \equiv B \pmod{P},$$

то

$$A^n \equiv B^n \pmod{P}.$$

Из свойства 6 следует, что над сравнениями можно произвести операцию извлечения корня n-й степени.

Рассмотренные выше свойства сравнений используются для осуществления операции контроля.

Существуют два метода получения контрольного кода:

числовой

$$r_A = f(A);$$

цифровой

$$r_A = f\left(\sum_i a_i\right),$$

где A — контролируемое число ($A = \sum_{i=0}^{i=n} a_i q^i$ для целых чисел); a_i — разряды числа A.

Рассмотрим эти методы подробно.

Числовой метод контроля. При числовом методе контроля код заданного числа определяется как наименьший положительный остаток от деления числа на выбранный модуль P:

$$r_A = A - \{A/P\} P, \tag{6.4}$$

где $\{\ \}$ — целая часть от деления числа.

При этом надо иметь в виду, что величина модуля P существенно влияет на качество контроля; если $P=q$ (q — основание системы счисления, в которой выражено число) и имеет место числовой контроль, то контролируется только младший разряд числа и контроль как таковой не имеет смысла; для $P=q^m$ справедливы аналогичные соображения, так как опять не все разряды числа (если $m<n$) участвуют в контроле и ошибки в разрядах старше m вообще не воспринимаются.

При числовом методе контроля по модулю P для определения остатка используют операцию деления, требующую больших затрат машинного времени. Для числового метода контроля справедливы основные свойства сравнений (сложение, умножение сравнений и т. д.). Поэтому, если

$$A \equiv r_A \pmod{P};$$

$$B \equiv r_B \pmod{P},$$

где $0 \leqslant r_A \leqslant P-1$; $0 \leqslant r_B \leqslant P-1$, то

$$A + B \equiv r_A + r_B \pmod{P}.$$

Отсюда

$$r_{A+B} \equiv r_A + r_B \pmod{P}. \qquad (6.5)$$

Аналогичным образом доказывается справедливость и следующих соотношений:

$$r_{A-B} \equiv r_A - r_B \pmod{P}; \qquad (6.6)$$

$$r_{AB} \equiv r_A r_B \pmod{P}. \qquad (6.7)$$

Пример 6.4. Для заданных чисел $A=125$ и $B=89$ определить контрольные коды самих чисел, их суммы и разности, если модуль $P=11$.

Решение. Контрольные коды чисел определяем по (6.4):

$$r_A = 125 - \{125/11\}\,11 = 4; \quad r_B = 89 - \{89/11\}\,11 = 1.$$

Аналогично находим контрольные коды для суммы и разности:

$$A+B=214, \quad r_{A+B} = 214 - \{214/11\}\,11 = 5;$$

$$A-B=36, \quad r_{A-B} = 36 - \{36/11\}\,11 = 3.$$

Проверку правильности определения контрольных кодов суммы и разности можно произвести на основании (6.5) и (6.6):

$$r_{A+B} = 4 + 1 \equiv 5 \pmod{11};$$

$$r_{A-B} = 4 - 1 \equiv 3 \pmod{11}.$$

Ответ: $r_A=4$, $r_B=1$, $r_{A+B}=5$, $r_{A-B}=3$.

Цифровой метод контроля. При цифровом методе контроля контрольный код числа образуется делением суммы цифр числа на выбранный модуль при выполнении условий

$$\left.\begin{aligned} r'_A &= \sum_i a_i - \left\{\frac{\sum_i a_i}{P}\right\} P \\ &\text{или} \\ r'_A &\equiv \sum_i a_i \pmod{P}. \end{aligned}\right\} \qquad (6.8)$$

Возможны два пути получения контрольного кода: 1) непосредственное деление суммы цифр на модуль P; 2) суммирование цифр по модулю P.

Второй путь весьма привлекателен, так как если $a_i < P$, то контрольный код получается только операцией суммирования. Это существенное преимущество цифрового метода контроля.

Пример 6.5. Определить контрольные коды чисел $A=153$ и $B=41$, их суммы и разности, если $P=11$.

Решение. Контрольные коды исходных чисел определяем по (6.8). Для этого находим суммы цифр и делим их на модуль:

$$\sum a_i = 9; \quad \sum b_i = 5.$$

Следовательно,

$$r'_A = 9; \quad r'_B = 5.$$

Аналогично определяем контрольные коды суммы:

$$C = A + B = 194; \quad \sum c_i = 14; \quad r'_C \equiv 3 \pmod{11}$$

и разности:

$$D = A - B = 112; \quad \sum d_i = 4; \quad r'_D \equiv 4 \pmod{11}.$$

Ответ: $r'_A = 9$, $r'_B = 5$, $r'_{A+B} = 3$, $r'_{A-B} = 4$.

Однако при цифровом методе свойства сравнений не всегда справедливы, и происходит это из-за наличия переносов (заемов) при выполнении арифметических действий над числами. Поэтому нахождение контрольного кода результата операции происходит обязательно с коррекцией.

Пусть заданы числа A и B и соответственно их контрольные коды $r_A' \equiv \sum\limits_i a_i \pmod{P}$; $r_B' = \sum\limits_i b_i \pmod{P}$; $C = A + B$.

Найти контрольный код r_C'.

Видимо, когда есть результат операции, то найти r'_C методом суммирования цифр по модулю не сложно.

Какова будет возможность получения r'_C через контрольные коды слагаемых?

Сумму цифр c_i числа можно найти, зная цифры a_i и b_i и количество переносов в каждом разряде. Каждый перенос уносит из данного разряда q единиц и добавляет одну единицу в следующий разряд, т. е. сумма цифр уменьшится на величину $q-1$ на каждый перенос.

Тогда

$$\sum_{i=1}^{n} c_i = \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i - l(q-1), \tag{6.9}$$

где l — количество переносов, возникших при сложении.

Так как

$$r'_A \equiv \sum_i a_i \pmod{P};$$

$$r'_B \equiv \sum_i b_i \pmod{P},$$

то

$$r'_C \equiv \sum_i c_i \pmod{P}.$$

Подставив эти значения в (6.9), получим

$$r'_C \equiv [r'_A + r'_B - l(q-1)] \pmod{P}. \tag{6.10}$$

Аналогичными рассуждениями можно показать, что для разности чисел $C = A - B$

$$r'_C \equiv [r'_A - r'_B + S(q-1)] \pmod{P}, \tag{6.11}$$

где S — количество заемов при выполнении операции.

Пример 6.6. Определить контрольные коды чисел $A=589$ и $B=195$, их суммы и разности, если $P=11$.

Решение. Контрольные коды исходных чисел определяем по (6.8). При этом используем второй путь, т. е. нахождение контрольного кода суммированием цифр по модулю:

$$\sum a_i = 5 \oplus 8 \oplus 9 = 0 \pmod{11};$$

$$r'_A \equiv 0 \pmod{11};$$

$$\sum b_i = 1 \oplus 9 \oplus 5 \equiv 4 \pmod{11};$$

$$r'_B \equiv 4 \pmod{11},$$

где символ $\oplus$ означает суммирование по указанному модулю.

Контрольный код суммы определяем по (6.10) — в этом случае $l=2$:

$$A+B=784;$$

$$r'_{A+B} \equiv 0+4-2(10-1) \equiv 8 \pmod{11}.$$

Примечание. В случае, когда имеет место отрицательный остаток, к сравнению надо добавить модуль P, столько раз, сколько необходимо для получения ближайшего положительного остатка.

Контрольный код разности получим по (6.11) — в этом случае $S=1$:

$$A-B=394;$$

$$r'_{A-B} \equiv 0-4+1(10-1) \equiv 5 \pmod{11}.$$

Ответ: $r'_A=0,\ r'_B=4,\ r'_{A+B}=8,\ r'_{A-B}=5.$

§ 6.6. Выбор модуля для контроля

Достоинство числового метода контроля — в справедливости свойств сравнений для контрольных кодов, что облегчает контроль арифметических операций.

Достоинство цифрового метода контроля — в возможности достаточно просто получать контрольные коды без значительных затрат времени. Чтобы сохранить эти достоинства, необходимо выполнение условия $r_A=r'_A$.

Так как $r_A \equiv A \pmod{P}$; $r'_A = \Sigma a_i \pmod{P}$, то

$$\sum_i a_i q^i \equiv \sum_i a_i \pmod{P}.$$

Это равенство возможно тогда, когда почленно обе части выражения равны:

$$a_i q^i \equiv a_i \pmod{P},$$

или

$$q^i \equiv 1 \pmod{P}.$$

Последнее выражение можно получить, если в сравнении $q \equiv 1 \pmod{P}$ возводить обе части в одну и ту же степень. Следовательно,

$$q \equiv 1 \pmod{P},$$

или

$$q = mP + 1, \qquad (6.12)$$

где m — целое число.

Из (6.12) следует, что

$$P = (q-1)/m. \qquad (6.13)$$

В результате получено, что для сохранения условия $r_A = r'_A$ необходимо наложить ограничения на модуль P.

Анализ (6.13) показывает, что для двоичной системы счисления нет целочисленного решения. Это значит, что контролируемую информацию надо представлять в некоторой промежуточной системе счисления. Выбор промежуточной системы счисления определяется величиной модуля P.

К модулю P предъявляют следующие общие требования:

1) величина модуля P должна быть такой, чтобы возникновение любой арифметической или логической ошибки нарушало сравнимость контрольных кодов;

2) образование контрольного кода должно осуществляться по возможности простыми средствами;

3) величина модуля P должна быть по возможности небольшой, так как необходимость выполнения контрольных операций ведет к увеличению вспомогательного оборудования.

Ввиду того что цифровая информация в ЭВМ должна представляться символами двоичного алфавита, для контроля целесообразно перейти к системам счисления с основанием $q=2^s$, где s — некоторое целое положительное число ($s \geqslant 2$). Переход от двоичного представления исходной информации к новому представлению с основанием $q=2^s$ осуществляется разбиением информации на группы по s разрядов с последующим суммированием этих групп по модулю $P=(2^s-1)/m$ или при $m=1$, $P=2^s-1$.

В самом деле, если $s=2$, то исходная информация разбивается на диады, при $s=3$ — на триады, при $s=4$ — на тетрады и т. д.

Свертывание — *процесс разбиения кодовой комбинации на группы и получения контрольного кода.*

Как правило, свертки (свернутые коды) образуются в результате суммирования выделенных групп (диад, триад и т. п.) по модулю P.

В теории кодирования показано, что модуль можно выбирать из условия

$$P = (2^s \pm 1)/m. \qquad (6.14)$$

Рассмотрим частные случаи образования сверток при разных значениях модуля P.

1. Контроль по модулю 3 ($m=1$, $s=2$, $P=3$). Здесь контролируемая информация представляется символами четверичной системы и свертки образуются суммированием диад по модулю 3. Так

как $2^s \equiv 1 \pmod 3$, то потребуется двухразрядный двоичный сумматор с цепью циклического переноса из старшего разряда в младший.

Пример 6.7. Найти контрольные коды для чисел $A=46=101110_{(2)}$, $B=29=$ $=011101_{(2)}$, если $P=3$.

Решение. Контрольные коды для чисел определяем по формуле (6.8) и цифры представляем диадами:

$$r_A = 10 \oplus 11 \oplus 10 \equiv 01 \pmod 3;$$

$$r_B = 01 \oplus 11 \oplus 01 \equiv 10 \pmod 3.$$

Ответ: $r_A=01$, $r_B=10$.

2. Контроль по модулю 7 ($m=1$, $s=3$, $P=7$). Здесь контролируемая информация разбивается на триады и представляется символами восьмеричной системы. Так как $2^3 \equiv 1 \pmod 7$, то для получения свертки нужно иметь трехразрядный двоичный сумматор с цепью циклического переноса.

Пример 6.8. Найти контрольный код для числа $C=153=011111001_{(2)}$ при $P=7$.

Решение. Исходное число разбиваем на триады, которые суммируются по mod 7:

$$r_C = 011 \oplus 111 \oplus 001 \equiv 100 \pmod 7.$$

Ответ: $r_C=100$.

3. Контроль по модулю 5 ($m=1$, $s=2$, $p=5$). Из теории чисел известно, что для того, чтобы число, выраженное в системе с основанием q, делилось на число $q+1$, необходимо и достаточно, чтобы разность между суммой цифр, стоящих на четных и нечетных местах или наоборот, делилась на величину $q+1$ без остатка.

Из этого правила можно сделать следующий вывод: контрольный код по mod $(q+1)$ определяется по формуле

$$r_A \equiv \sum_i (-1)^i b_i \bmod (q+1), \tag{6.15}$$

где b_i — двоичное изображение цифр в системе с основанием 2^s.

Так как, по условию, $r_A \leqslant P-1$, то для получения свертки потребуется трехразрядный двоичный сумматор, работающий по модулю 5.

Пример 6.9. Найти контрольный код для числа $A=0101101110$ при $P=5$.

Решение. Сначала исходное число разбивается на диады:

$$A = \underbrace{01}_{b_5}\ \underbrace{01}_{b_4}\ \underbrace{10}_{b_3}\ \underbrace{11}_{b_2}\ \underbrace{10}_{b_1}$$

Затем диады суммируем по правилу (6.15):

$$r_A = b_1 \oplus b_3 \oplus b_5 \ominus b_2 \ominus b_4 = 10 \oplus 10 \oplus 01 \ominus 01 \ominus 11 = 001 \pmod 5.$$

Если получается отрицательный остаток, то его надо заменить на дополнение до модуля.

Ответ: $r_A=001$.

§ 6.7. Контроль логических операций

К логическим операциям относятся операции сдвига, логического сложения и умножения, выполняемые по правилам, описанным в других главах (более подробно см. гл. 9).

Несмотря на кажущуюся простоту этих правил, осуществление операций контроля сталкивается с рядом трудностей, объясняемых тем, что логические операции являются поразрядными операциями.

Операция сдвига. Пусть задано число $A = a_n a_{n-1} \ldots a_1 a_0$, имеющее контрольный код $r_A = a_{k_s} \ldots a_{k_1}$.

Обозначим код числа A, сдвинутый влево $\overleftarrow{A}$ (без циклического переноса) и $\overleftarrow{A}_ц$ (с циклическим переносом) (при сдвиге вправо стрелка в обозначении будет повернута направо). Соответствующим образом обозначим и контрольный код: $A \equiv r_A \pmod P$; $\overleftarrow{A} \equiv r_{\overleftarrow{A}} \pmod P$; $\overrightarrow{A} \equiv r_{\overrightarrow{A}} \pmod P$; $\overleftarrow{A}_ц \equiv r_{\overleftarrow{A}_ц} \pmod P$.

Сдвиг влево двоичного числа эквивалентен умножению на 2. Так как при сдвиге числа происходит потеря некоторых его разрядов, то можно предполагать, что контрольный код сдвинутого числа изменится на величину Δ:

$$r_{\overleftarrow{A}} \equiv \overleftarrow{r}_A + \Delta \pmod P, \tag{6.16}$$

где $\overleftarrow{r}_A = 2r_A$ — сдвинутый влево контрольный код.

Очевидно, что величина Δ зависит от значений a_n и a_{k_s}, которые при сдвиге выходят за пределы разрядной сетки.

Если при сдвиге n-разрядного числа старшая единица выйдет за пределы разрядной сетки, то это эквивалентно вычитанию $a_n \sigma_{n+1}$ единиц из контрольного кода сдвинутого числа (где σ_{n+1} — вес $(n+1)$-го разряда).

Если при сдвиге контрольного кода выходит за пределы разрядной сетки разряд $a_{k_s} = 1$, то это эквивалентно уменьшению контрольного кода на $2^s = 1$. Такую потерю надо восстановить прибавлением к контрольному коду единицы.

В общем случае (6.16) принимает вид

$$r_{\overleftarrow{A}} \equiv (\overleftarrow{r}_A - a_n \sigma_{n+1} + a_{k_s}) \bmod (2^s - 1). \tag{6.17}$$

Веса разрядов кодовой комбинации, представленной в системе с основанием 2^s, назначаются следующим образом:

$$\begin{matrix} s = 3 \\ \text{веса } \sigma_i \end{matrix} \left\{ \begin{matrix} a_n & a_{n-1} & a_{n-2} & a_{n-3} & \ldots & a_3 & a_2 & a_1; \\ 2^2 & 2^1 & 2^0 & 2^2 & \ldots & 2^2 & 2^1 & 2^0. \end{matrix} \right.$$

В результате значения поправок Δ для контроля выполнения левого сдвига по модулю будут:

Значение a_n	0	1	0	1
Значение a_{k_s}	0	0	1	1
Поправка Δ	0	—1	+1	0

Значение поправок $\Delta=-1$ можно заменить ее дополнением до модуля.

Для выполнения сдвига влево с циклическим переносом из старшего разряда в младший разряд необходимо уменьшить контрольный код на величину $a_n\ (\sigma_{n+1}-1)$; так как $\sigma_{n+1}=1$, то этот член равен 0. Следовательно, формула (6.17) изменяется:

$$r_{\overleftarrow{A}_{\text{ц}}} \equiv \overleftarrow{r_A} + a_{k_s} \bmod (2^s-1). \tag{6.18}$$

Пример 6.10. Найти контрольные коды для числа $A=1{,}01011010$, сдвигаемого влево, при $P=7 (S=3)$.

Решение. Сначала определяем контрольный код исходного числа путем сложения триад по модулю 7:

$$r_A \equiv 101 \oplus 011 \oplus 010 \equiv 011 \pmod 7.$$

Затем сдвигается влево число $\overleftarrow{A}=0{,}10110100$ и его контрольный код $\overleftarrow{r_A}=110$.

На основании (6.17) при $a_n=1$, $a_{k_s}=0$ определяем контрольный код сдвинутого числа:

$$r_{\overleftarrow{A}} \equiv 110-1+000 \equiv 101 \pmod 7.$$

Производится сдвиг с циклическим переносом:

$$\overleftarrow{A}_{\text{ц}} = 0{,}10110101,$$

для которого контрольный код

$$r_{\overleftarrow{A}_{\text{ц}}} = 110+000 \equiv 110 \pmod 7.$$

Ответ: $r_{\overleftarrow{A}}=101$, $r_{\overleftarrow{A}_{\text{ц}}}=110$.

При сдвиге вправо происходит потеря младших разрядов числа и контрольного кода этого числа. Так как сдвиг вправо эквивалентен делению на 2, то

$$\overrightarrow{A}=(A-a_1)/2; \quad \overrightarrow{r_A}=(r_A-a_{k_1})/2. \tag{6.19}$$

Эти потери надо компенсировать. Это означает, что контрольный код сдвинутого вправо числа можно найти по формуле

$$r_{\overrightarrow{A}} = \overrightarrow{r_A} + \Delta \pmod{2^s-1}. \tag{6.20}$$

В зависимости от модуля поправка к контрольному коду в случае простого сдвига принимает следующее значение:

Значение a_1	0	0	1	1
Значение a_k	0	1	0	1
Поправка Δ для:				
$P=3$	00	10	01	00
$P=7$	000	100	011	000

При модифицированном сдвиге вправо, который выполняется по правилу $A=1,\ a_{n-1}a_{n-2}\ \dots\ a_2a_1$; $\vec{A}_{\text{м}}=1,\ 1a_{n-1}\ \dots\ a_3a_2$, происходит также потеря младших разрядов кодовой комбинации числа и контрольного кода. Для этого случая формула (6.20) сохраняет свой вид, но поправки должны быть следующими:

Значение a_1	0	0	1	1
Значение a_{k_1}	0	1	0	1
Поправка $\Delta_{\text{м}}$ для:				
$P=3$	10	01	00	10
$P=7$	100	001	000	100

Пример 6.11. Найти контрольные коды для числа $A=1{,}01110111101$, сдвигаемого вправо, при $P=7$.

Решение. Сначала определяем контрольный код для исходного числа путем сложения триад по модулю 7:

$$r_A=101\oplus 110\oplus 111\oplus 101\equiv 010\,(\mathrm{mod}\ 7).$$

Затем сдвигается вправо число $\vec{A}=0{,}10111011110$ и его контрольный код $\vec{r}_A=001$.

На основании (6.20) при $a_1=1$, $a_{k_1}=0$, поправки $\Delta=011$ определяем контрольный код:

$$r_{\vec{A}}\equiv 001+011\equiv 100\,(\mathrm{mod}\ 7).$$

Производится модифицированный сдвиг числа $\vec{A}_{\text{м}}=1{,}10111011110$, для которого контрольный код находим при $\Delta=000$:

$$r_{\vec{A}_{\text{м}}}\equiv 001+000\equiv 001\,(\mathrm{mod}\ 7).$$

Ответ: $r_A=010$, $r_{\vec{A}}=100$, $r_{\vec{A}_{\text{м}}}=001$.

Операция сложения по модулю 2. Операцию сложения по модулю 2 можно выразить через другие арифметические операции (см. § 9.3), например

$$A\oplus B=A+B-2(A\wedge B).$$

Если применить к этому выражению уже известные приемы, то получим

$$A\oplus B=(A+B)+\overline{A\wedge B}_{\text{сдв}}.$$

Тогда, используя переход от арифметических выражений к сравнениям, получим следующую формулу для образования контрольного кода:

$$r_{\oplus}=r_{A+B}+\bar{r}_{\wedge}\,(\mathrm{mod}\ P), \tag{6.21}$$

где r_{A+B} — контрольный код суммы двух чисел; $\bar{r}_{\wedge}$ — инверсия контрольного кода логического произведения двух чисел со сдвигом влево на один разряд.

Пример 6.12. Найти контрольный код логической суммы чисел $A=010000111$ и $B=101110011$ по модулю 7.

Решение. Прежде всего по изложенным выше правилам определим контрольные коды для исходных чисел:

$$r_A=010, \quad r_B=000, \quad r_{A+B}=010.$$

Затем вычислим следующие величины:

$$A\wedge B=000000011, \quad r_{\wedge}=011;$$

$$A\wedge B_{\text{сдв}}=000000110, \quad \overleftarrow{r}_{\wedge}=110.$$

После этого определим инверсное значение $\bar{r}_{\wedge}=001$.

По формуле (6.21) находим контрольный код:

$$r_{\oplus}=010+001\equiv 011 \pmod 7.$$

Ответ: $r_{\oplus}=011$.

Операция логического умножения. Операцию логического умножения двух чисел можно выразить через другие арифметические и логические операции:

$$A\wedge B=2^{-1}(A+B)-2^{-1}(A\oplus B).$$

Умножение на 2^{-1} означает сдвиг кода числа, стоящего в скобках, вправо на один разряд. После перехода к сравнениям контрольный код для логического умножения получается следующим образом:

$$r_{\wedge}\equiv r_{\overrightarrow{A+B}}+\bar{r}_{\overrightarrow{\oplus}} \pmod P, \tag{6.22}$$

где $r_{\overrightarrow{A+B}}$ — контрольный код суммы, сдвинутый вправо на один разряд; $\bar{r}_{\overrightarrow{\oplus}}$ — инверсия контрольного кода логической суммы чисел, сдвинутой на разряд вправо.

Примечание. При выполнении сдвигов необходима коррекция контрольных кодов в соответствии с изложенными выше правилами.

Пример 6.13. Найти контрольный код логического произведения чисел по модулю 3:

$$A=10011001, \quad r_A=00,$$

$$B=0.001111, \quad r_B=01.$$

Решение. Прежде всего находим сумму чисел и контрольный код:

$$A+B=11101000, \quad r_{A+B}=01.$$

Затем по (6.20) вычисляем

$$r_{\overrightarrow{A+B}}=\overrightarrow{r}_{A+B}+\Delta=10.$$

Определяем:

$$A \oplus B = 11010110, \quad r_{\oplus} = 01,$$

$$r_{\underset{\oplus}{\rightarrow}} = \vec{r}_{\oplus} + \Delta = 10,$$

$$\overline{r}_{\underset{\oplus}{\rightarrow}} = 01.$$

Следовательно, контрольный код логического произведения

$$r_{\wedge} \equiv 10 + 01 \equiv 00 \,(\mathrm{mod}\, 3).$$

Ответ: $r_{\wedge} = 00$.

§ 6.8. Контроль арифметических операций

Арифметические операции выполняются на сумматорах прямого, обратного и дополнительного кодов. Предполагая, что изображения чисел (операнды) хранятся в машине в соответствующем коде, т. е. операция преобразования в заданный код или обратно производится на входе или выходе машины, можно представить себе следующую методику реализации операций контроля.

Прежде всего рассматривают изображение числа в соответствующем коде как единую кодовую комбинацию, к которой можно приложить все сформулированные выше правила получения сверток. При этом требуется только обязательная кратность общего числа разрядов избранному модулю.

Рассмотрим последовательность действий на примере сумматора прямого кода.

Поскольку на сумматоре прямого кода складываются только цифровые части изображений чисел, а знак сохраняется, то контроль можно осуществить двумя способами:

1) раздельный контроль знаковой и цифровой частей изображений результата;

2) обобщенный контроль всего изображения.

При раздельном способе для контроля знаковых разрядов можно использовать средства для обнаружения переполнения, так как в случае модифицированного кода появление ошибок в знаковых разрядах приведет к несовпадению информации в них. При проверке правильности обработки цифровых частей изображений также не возникнет особых трудностей.

При обобщенном способе контроля требуется коррекция контрольного кода результата из-за того, что знак результата при сложении повторяет знак слагаемых. Следовательно, можно констатировать, что контрольный код суммы чисел должен быть

$$r_{(A+B)_{\text{пр}}} \equiv r_A + r_B - \mathrm{Sg}\,\sigma_s \,(\mathrm{mod}\, p), \tag{6.23}$$

где Sg — значение знакового разряда операндов; σ_s — вес старшего разряда свертки.

Пример 6.14. Произвести контроль операции сложения чисел на сумматоре прямого кода:

$$[A]_{пр} = 1{,}01101011,$$

$$[B]_{пр} = 1{,}00110010,$$

$$P = 7.$$

Решение. Прежде всего определим по (6.8) контрольные коды исходных чисел:

$$r_A = 110, \quad r_B = 101.$$

Результат операции

$$[A + B]_{пр} = 1{,}10011101.$$

Тогда контрольный код результата по (6.8)

$$r_{(A+B)} \equiv 110 \oplus 011 \oplus 101 \pmod 7,$$

или

$$r_{(A+B)} \equiv 000.$$

На основании (6.23) находим

$$r_{(A+B)_{пр}} \equiv 110 + 101 - 1 \cdot 100 \pmod 7,$$

или

$$r_{(A+B)_{пр}} \equiv 000.$$

Контрольные коды совпадают, что свидетельствуют о правильном выполнении операции.

Ответ: $r_{A+B} = 000$.

Обобщенный способ контроля может быть применен и для сумматоров обратного и дополнительного кодов.

Пример 6.15. Произвести контроль операции сложения кодов для сумматора обратного кода:

$$[A]_{об} = 1{,}011001001;$$

$$[B]_{об} = 0{,}110001111, \quad P = 3.$$

Решение. Определяем на основании (6.8):

$$r_{A_{об}} \equiv 10 \oplus 11 \oplus 00 \oplus 10 \oplus 01 \pmod 3; \quad r_{A_{об}} = 10,$$

$$r_{B_{об}} = 01 \oplus 10 \oplus C0 \oplus 11 \oplus 11 \pmod 3; \quad r_{B_{об}} = 00.$$

Результат:

$$(A + B)_{об} = 0{,}001011001 \quad и \quad r_{(A+B)_{об}} = 10.$$

Проверка:

$$r_{(A+B)_{об}} = r_{A_{об}} + r_{B_{об}} = 10 \pmod 3.$$

Ответ: $r_{A+B} = 10$.

При сложении чисел на сумматоре дополнительного кода потребуется коррекция контрольного кода в случае, если знаковые разряды изображений содержат единицу, так как при этом возникает единица переноса из знакового разряда. Очевидно, что контрольный код суммы будет равен

$$r_{(A+B)}=r_{A_д}+r_{B_д}-\alpha, \tag{6.24}$$

где α — коррекция ($\alpha=1$, если возник перенос из знакового разряда, и $\alpha=0$ — если переноса нет).

Пример 6.16. Произвести контроль операции сложення на сумматоре дополнительного кода:

$$[A]_д=1,00110001110;$$
$$[B]_д=1,11101110111,$$
$$P=15.$$

Решение: Определяем:

$$r_{A_д}\equiv 1001\oplus 1000\oplus 1110\ (\mathrm{mod}\ 15),\quad r_{A_д}=0001,$$

$$r_{B_д}\equiv 1111\oplus 0111\oplus 0111\ (\mathrm{mod}\ 15),\quad r_{B_д}=1110.$$

$$[A+B]_д=1,00100000101\quad \text{и}\quad r_{(A+B)}\equiv 1001\oplus 0000\oplus 0101\ (\mathrm{mod}\ 15),$$

$$r_{(A+B)}=1110.$$

Проверка:

$$r_{(A+B)_д}=r_{A_д}+r_{B_д}-\alpha=0001+1110-0001=1110.$$

Ответ: $r_{(A+B)}=1110$.

В случаях, когда операция преобразования в обратный или дополнительный код производится в процессе вычислений, целесообразна проверка этих преобразований.

Рассмотрим операцию преобразования из прямого кода в обратный.

Пусть $[A]_{пр}=a_na_{n-1}...a_1a_0$ — исходное число, представленное в прямом коде, для которого определяется остаток $r_A\equiv[A]_{пр}\ (\mathrm{mod}\ P)$. Подобные выражения можно записать для обратного кода $r_{A\,об}\equiv\equiv[A]_{об}\ (\mathrm{mod}\ P)$ и инвертированного изображения $r_{\overline{A}}\equiv[\overline{A}](\mathrm{mod}\ P)$.

Исходное число представляется следующим алгебраическим выражением:

$$[A]_{пр}=a_n2^n+A',$$

где a_n — знаковый разряд; A' — цифровая часть изображения.

Тогда

$$[\overline{A}]=\overline{a}_n2^n+\overline{A}'; \tag{6.25}$$

$$[A]_{об}=a_n2^n+\overline{A}', \tag{6.26}$$

где $\overline{A}'$ — инвертированное изображение цифровой части числа.

Из (6.26) вычтем (6.25):

$$[A]_{об}-[\overline{A}]=(a_n-\overline{a}_n)\,2^n. \tag{6.27}$$

Левую часть равенства (6.27) заменим сравнениями:

$$r_{A_{об}}-r_{\overline{A}}\equiv(a_n-\overline{a}_n)\,2^n \pmod P.$$

Так как $a_n=1$, то $\overline{a}_n=0$, следовательно,

$$r_{A_{об}}-r_{\overline{A}}\equiv 2^n \pmod P.$$

Пусть $2^n\equiv k \pmod P$.

Тогда

$$r_{A_{об}}\equiv r_{\overline{A}}+k \pmod P. \tag{6.28}$$

Выражение (6.28) дает возможность найти контрольное число для обратного кода при известных значениях величины P:

n	5	8	10	15	18	20	25	30	40
$P=3$ . . .	2	1	1	1	1	1	2	1	1
$P=7$ . . .	4	4	2	1	1	4	2	1	2

Пример 6.17. Осуществить контроль преобразования числа A в обратный код:

$$[A]_{пр}=1{,}01110010,\quad P=7.$$

Решение. Определяем следующие величины и соответствующие им контрольные коды:

$$[A]_{обр}=1{,}10001101,\quad r_{A_{об}}=101;$$

$$[\overline{A}]=0{,}10001101,\quad r_{\overline{A}}=001.$$

По (6.28) находим

$$r_{A_{об}}=r_{\overline{A}}+k=001+100=101,$$

где $k=100$, так как $n=8$.

Ответ: $r_{A_{об}}=101$.

Операцию преобразования в дополнительный код можно осуществить, используя соотношение (3.11) между обратным и дополнительным кодами: дополнительный код отличается от обратного на единицу младшего разряда. Следовательно, контрольное число можно получить из выражения

$$r_{A_{доп}}\equiv r_{\overline{A}}+k+1 \pmod P. \tag{6.29}$$

Рассмотренную выше методику определения контрольных чисел для таких операций, как умножение, или таких операций, как деление, можно применить для контроля работы ЭВМ, используя:

1) операционный контроль (контролем охватываются исходные числа и конечный результат);

2) шаговый контроль (проверяются все элементарные действия автомата при выполнении операции).

Задание для самоконтроля

1. Определить вес и кодовые расстояния для: а) $A=01101111$; $B=11101101$; б) $A=01010101$; $B=10101010$.

2. Произвести проверку правильности кода по методу Хэминга для чисел (номера разрядов идут слева направо) $A=101011001101010$; $B=101011001101110$; $C=101011110101010$.

3. Произвести контроль операций сложения $A+B$, умножения AB, вычитания $A-B$, используя методы числового контроля, если $A=122$, $B=93$, $P=11$.

4. Определить контрольные коды для чисел $A=10010001_{(2)} (P=7)$; $B=10101101_{(2)} (P=5)$; $C=0{,}01001001_{(2)} (P=7)$.

5. Определить контрольные коды числа $A=0{,}10010001$, которое сдвигается влево и вправо при $P=7$.

6. Найти контрольный код для логической суммы и логического произведения чисел $A=110011010$ и $B=010011011$.

7. Провести контроль операций сложения и вычитания чисел $A=0{,}100110011$ $(P=3)$ и $B=-0{,}011110010 (P=3)$ на сумматоре обратного кода.

8. Провести контроль операции умножения чисел $A=0{,}10011110 (P=7)$ и $B=-0{,}11001101 (P=7)$ на сумматоре прямого кода.

9. Определить правильность выполнения операции сложения чисел $A=-0{,}100011011$ и $B=0{,}11100011$ (при $P=3$) на сумматоре дополнительного кода.

10. Проверить операцию умножения чисел (из п. 9) на сумматоре прямого кода.

11. Возможно ли использовать контроль по модулю для определения места, где произошла ошибка?

12. Возможно ли применить коды Хэминга для контроля операции сложения и умножения кодов? Если да, то как это осуществить?

7 ГЛАВА

ВЫПОЛНЕНИЕ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ В НЕПОЗИЦИОННЫХ СИСТЕМАХ СЧИСЛЕНИЯ

§ 7.1. Основные сведения о системе остаточных классов

Пусть задан набор целых положительных чисел $p_1, p_2, ..., p_k$, которые в дальнейшем будут называться основаниями или модулями. Тогда любое положительное число можно представить в виде

$$A(P) = \alpha_1 B_1 + \alpha_2 B_2 + \ldots + \alpha_k B_k, \tag{7.1}$$

где $A(P)$ — представление числа A в системе счисления с основаниями p_i; B_i — целые положительные числа; α_i — наименьший положительный остаток от деления числа A на модуль p_i:

$$\alpha_i \equiv \text{res}(A) \bmod p_i; \quad 0 \leqslant \alpha_i \leqslant p_{i-1}. \tag{7.2}$$

Примечание. Обозначение res (A) происходит от слова residuum — остаток.

Остаток выбирается таким образом, чтобы $\alpha_i = A - s_i p_i$, $i = 1, 2, ..., k$, по аналогии с записью сравнений вида $A = \alpha_i + s_i p_i$ или $A \equiv \alpha_i (\bmod p_i)$.

Запись положительных чисел в системе остаточных классов — *представление этих чисел в виде* (7.1), *с выполнением условия* (7.2).

В качестве оснований системы остаточных классов выбран набор модулей $p_1, p_2, ..., p_k$. Само число изображается набором остатков по каждому из модулей, т. е. $A = (\alpha_1, \alpha_2, ..., \alpha_k)$. Отсюда видно, что образование остатков α_i (или символов для изображения числа) производится независимо друг от друга. Следовательно, такая система представления чисел — непозиционная. В теории чисел (см. [1]) доказывается, что если модули $p_1, p_2, ..., p_k$ являются взаимно-простыми целыми числами, то для каждого целого положительного числа A его изображение в виде остатков α_i единственное.

Рассмотрим кодирование чисел в системе остаточных классов на конкретном примере. Пусть $p_1 = 2$, $p_2 = 3$, $p_3 = 5$.

Изображение чисел в системе оснований p_1, p_2, p_3 представим в виде табл. 7.1.

Из таблицы видно, что изображения чисел повторяются начиная с $A = 30$.

Таблица 7.1

Десятичное число	Изображение десятичного числа $(\alpha_1, \alpha_2, \alpha_3)$	Десятичное число	Изображение десятичного числа $(\alpha_1, \alpha_2, \alpha_3)$	Десятичное число	Изображение десятичного числа $(\alpha_1, \alpha_2, \alpha_3)$
0	(0,0,0)	10	(0,1,0)	20	(0,2,0)
1	(1,1,1)	11	(1,2,1)	21	(1,0,1)
2	(0,2,2)	12	(0,0,2)	22	(0,1,2)
3	(1,0,3)	13	(1,1,3)	23	(1,2,3)
4	(0,1,4)	14	(0,2,4)	24	(0,0,4)
5	(1,2,0)	15	(1,0,0)	25	(1,1,0)
6	(0,0,1)	16	(0,1,1)	26	(0,2,1)
7	(1,1,2)	17	(1,2,2)	27	(1,0,2)
8	(0,2,3)	18	(0,0,3)	28	(0,1,3)
9	(1,0,4)	19	(1,1,4)	29	(1,2,4)
30	(0,0,0)	31	(1,1,1)	32	(0,2,2)

Диапазон представления чисел в системе остаточных классов — *произведение всех оснований системы*, т. е.

$$P = p_1 p_2 \ldots p_k = \prod_{i=1}^{i=k} p_i. \tag{7.3}$$

Следовательно, для того чтобы обеспечить однозначность изображения чисел, необходимо потребовать выполнения условия

$$A < P \quad \text{или} \quad 0 \leqslant A \leqslant P-1. \tag{7.4}$$

Система остаточных классов допускает расширение или сокращение набора оснований, не искажая при этом исходное число. В самом деле, пусть для набора чисел p_1, p_2, p_3 число изображается в виде $A=(\alpha_1, \alpha_2, \alpha_3)$. Введем новые основания p_i, p_j. Тогда изображение числа изменится:

$$A=(\alpha_1, \ \alpha_2, \ \alpha_3, \ \alpha_i, \ \alpha_j).$$

Аналогичным образом можно сократить набор оснований. Естественно, что расширение оснований увеличивает диапазон представления, а сокращение — уменьшает его.

§ 7.2. Формальные правила выполнения арифметических операций

Изображение чисел в системе остаточных классов основано на свойствах сравнений.

Пусть для заданного набора оснований $p_1, p_2, \ldots, p_k$ числа A и B представлены в виде остатков:

$$A=(\alpha_1,\ \alpha_2, \ldots,\ \alpha_k); \quad B=(\beta_1,\ \beta_2, \ldots,\ \beta_k);$$

$$0 \leqslant \alpha_i \leqslant p_i-1; \quad 0 \leqslant \beta_i \leqslant p_i-1; \quad A<P; \quad B<P.$$

Тогда на основании свойств сравнений можно написать, что

$$\left.\begin{aligned} A+B&=(\alpha_1+\beta_1,\ \alpha_2+\beta_2, \ldots,\ \alpha_k+\beta_k); \\ A-B&=(\alpha_1-\beta_1,\ \alpha_2-\beta_2, \ldots,\ \alpha_k-\beta_k); \\ AB&=(\alpha_1\beta_1,\ \alpha_2\beta_2, \ldots,\ \alpha_k\beta_k). \end{aligned}\right\} \tag{7.5}$$

При этом надо иметь в виду, что если $\alpha_i+\beta_i \geqslant p_i$ и $\alpha_i\beta_i \geqslant p_i$, то в качестве цифры i-го разряда берутся величины из условия (7.2), или $(\alpha_i+\beta_i)-s_ip_i \geqslant 0$; $(\alpha_i\beta_i)-m_ip_i \geqslant 0$, где s_i, m_i — целые положительные числа.

Если $\beta_i>\alpha_i$, то вычитание остатков выполняется таким образом, что $p_i+\alpha_i-\beta_i \geqslant 0$.

Пример 7.1. Пусть $p_1=2$, $p_2=3$, $p_3=5$, $A_1=27$, $A_2=5$, $A_3=4$. Найти A_2+A_3; A_1-A_2; A_3A_2.

Решение. Сначала все числа записываем в системе остаточных классов с основаниями p_1, p_2, p_3:

$$A_1=(1,\ 0,\ 2); \quad A_2=(1,\ 2,\ 0); \quad A_3=(0,\ 1,\ 4).$$

В соответствии с формальными правилами

$$A_2+A_3=(1+0;\ 2+1;\ 0+4)=(1,\ 0,\ 4)=9;$$

$$A_1-A_2=(1-1;\ 0-2;\ 2-0)=(0,\ 1,\ 2)=22;$$

$$A_2A_3=(1\cdot 0;\ 2\cdot 1;\ 0\cdot 4)=(0,\ 2,\ 0)=20.$$

Ответ: $A_2+A_3=(1,\ 0,\ 4)$; $A_1-A_2=(0,\ 1,\ 2)$; $A_2A_3=(0,\ 2,\ 0)$.

Операция деления в системе остаточных классов может выполняться только в случае деления без остатка.

Пусть $C=A/B=(\gamma_1, \gamma_2, \ldots, \gamma_k)$,
где

$$\gamma_i=[\alpha_i/\beta_i]=\alpha_i\,[1/\beta_i]\,p_i. \tag{7.6}$$

Таким образом, деление без остатка двух чисел может осуществляться умножением остатков делимого на обратную величину остатков делителя.

Величина $x=[1/\beta_i]p_i$ есть решение сравнения $x\beta_i \equiv 1 \pmod{p_i}$, которое однозначно определено, если β_i и p_i — взаимно-простые числа.

Предположим, что $\beta_i=0$. Это означает, что модуль p_i является делителем числа B. Но так как, по условию, число A делится на число B без остатка, то модуль должен быть делителем и числа A, т. е. $\alpha_i=0$ и соответствующая цифра частного становится неопределенной, так как $\gamma_i=\left(\dfrac{0}{0}\right)p_i$. Тогда можно сократить это основание p_i и вычислить цифры частного по значениям других остатков или путем расширения набора оснований.

Положительное качество систем остаточных классов — сравнительная простота выполнения арифметических операций. Так как основаниями системы являются, как правило, небольшие числа, то это позволяет арифметические действия описывать в виде таблиц, которые закладываются в память машины. В этом случае выполнение операции сводится к выборке результатов по заданным остаткам операндов. К положительным качествам систем остаточных классов относят также отсутствие связей между разрядами изображения числа.

Однако представление чисел в системах остаточных классов исключает возможность их сравнения, что затрудняет выявление переполнения разрядной сетки (переполнение разрядной сетки может иметь место при выполнении операций сложения чисел одного знака, умножения и деления), представление отрицательных и дробных чисел, для изображения которых приходится прибегать к различным приемам.

Рассмотренные выше формальные правила выполнения операций в системе остаточных классов позволяют существенно повысить скорость работы вычислительных устройств. Однако в настоящее время для систем остаточных классов существует больше нерешенных проблем, что ограничивает их широкое практическое использование. Вследствие этого они могут быть использованы либо в малых вычислительных автоматах без программного управления, либо в специализированных машинах с короткой программой, где диапазоны чисел заранее определены для исходных данных, промежуточных и окончательных результатов.

§ 7.3. Перевод чисел из позиционной системы в систему остаточных классов

Выше уже был рассмотрен один из методов перевода чисел в систему остаточных классов. Этот метод заключается в том, что исходное число поочередно делится на модули и определяется наименьший положительный остаток, который и представляет цифру числа по данному основанию. При этом порядок записи цифр не имеет большого значения, если указано соответствие положения остатков их основаниям.

Примечание. Для дальнейшего изложения примем, что положение оснований соответствует меньшему по значению модулю слева с увеличением модулей вправо до наибольшего.

Например, для модулей $p_1=2$, $p_2=7$; $p_3=11$; $p_4=19$; $p_5=23$ получаем, для числа $A=199=(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)$, $\alpha_1=(199/2)=1$; $\alpha_2==(199/7)=3$; $\alpha_3=(199/11)=1$; $\alpha_4=(199/19)=9$; $\alpha_5=(199/23)=15$, или $A=199=(1, 3, 1, 9, 15)$.

Примечание. Здесь и в дальнейшем в скобках даны остатки от деления.

Однако из-за наличия операции деления такой метод перевода не эффективен для машинного использования.

Для машинного перевода используется следующий метод перевода чисел в систему остаточных классов. Имеется позиционная запись числа

$$A_q = a_k q^k + a_{k-1} q^{k-1} + \ldots + a_1 q^1 + a_0 q^0. \tag{7.7}$$

Пусть для набора модулей $p_1, p_2, \ldots, p_n$ любая степень основания представляется в виде остатков

$$q^i = (\eta_{1i}, \eta_{2i}, \ldots, \eta_{ni}). \tag{7.8}$$

Подставив (7.8) в (7.7), получим

$$\begin{gathered} A = a_k(\eta_{1k}, \eta_{2k}, \eta_{3k}, \ldots, \eta_{nk}) + a_{k-1}(\eta_{1,k-1}, \eta_{2,k-1}, \ldots, \eta_{n,k-1}) + \ldots \\ \ldots + a_1(\eta_{1,1}, \eta_{2,1}, \ldots, \eta_{n,1}) + a_0(\eta_{1,0}, \eta_{2,0}, \ldots, \eta_{n,0}) = \\ = (a_k\eta_{1,k} + a_{k-1}\eta_{1,k-1} + a_1\eta_{1,1} + a_0\eta_{1,0}) + (a_k\eta_{2,k} + a_{k-1}\eta_{2,k-1} + \\ + a_1\eta_{2,1} + a_0\eta_{2,0}) + \ldots + (a_k\eta_{n,k} + a_{k-1}\eta_{n,k-1} + \ldots + a_1\eta_{n,1} + a_0\eta_{n,0}). \end{gathered} \tag{7.9}$$

Если предположить, что выражение (7.9) представляет число A в соответствии с (7.1), то

$$\left.\begin{aligned} \alpha_1 &\equiv (a_k\eta_{1,k} + a_{k-1}\eta_{1,k-1} + \ldots + a_1\eta_{1,1} + a_0\eta_{1,0}) \bmod p_1; \\ \alpha_2 &\equiv (a_k\eta_{2,k} + a_{k-1}\eta_{2,k-1} + \ldots + a_1\eta_{2,1} + a_0\eta_{2,0}) \bmod p_2; \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \alpha_n &\equiv (a_k\eta_{n,k} + a_{k-1}\eta_{n,k-1} + \ldots + a_1\eta_{n,1} + a_0\eta_{n,0}) \bmod p_n. \end{aligned}\right\} \tag{7.10}$$

Следовательно,

$$A = (\alpha_1, \alpha_2, \ldots, \alpha_n)$$

и перевод в конечном итоге осуществляется по правилам (7.10).

Пример 7.2. Перевести число $A=97$ в систему остаточных классов с основаниями $p_1=3$; $p_2=5$; $p_3=7$.

Решение. Прежде всего запишем следующие числа:

$$10^0 = (1, 1, 1);$$

$$10^1 = (1, 0, 3).$$

Применяя правила (7.10), получим:

$$\alpha_1 \equiv 9\cdot 1 + 7\cdot 1 \equiv 16 \equiv 1 \pmod 3;$$

$$\alpha_2 \equiv 9\cdot 0 + 7\cdot 1 \equiv 7 \equiv 2 \pmod 5;$$

$$\alpha_3 \equiv 9\cdot 3 + 7\cdot 1 \equiv 34 \equiv 6 \pmod 7.$$

В результате имеем $A=97=(1, 2, 6)$.

Ответ: $A=(1, 2, 6)$.

§ 7.4. Перевод из системы остаточных классов в позиционную систему

Задача перевода остаточных классов в позиционную систему формулируется следующим образом: *для заданного набора модулей* $p_1, p_2, \ldots, p_n$ *перевести изображение* $A=(\alpha_1, \alpha_2, \ldots, \alpha_n)$ *в систему с основанием* q *так, чтобы*

$$A=\mu_1 B_1+\mu_2 B_2+\ldots+\mu_n B_n, \tag{7.11}$$

где $B_i=(\beta_{1,i}, \beta_{2,i}, \ldots, \beta_{n,i})$ — базисы.

Раскроем (7.11), подставив значения базисов B_i:

$$\begin{gathered}
A=(\alpha_1, \alpha_2, \ldots, \alpha_n)=\mu_1(\beta_{1,1}, \beta_{2,1}, \ldots, \beta_{n,1})+\\
+\mu_2(\beta_{1,2}, \beta_{2,2}, \ldots, \beta_{n,2})+\ldots+\mu_n(\beta_{1,n}, \beta_{2,n}, \ldots, \beta_{n,n})=\\
=(\mu_1\beta_{1,1}+\mu_2\beta_{1,2}+\ldots+\mu_n\beta_{1,n})+(\mu_1\beta_{2,1}+\mu_2\beta_{2,2}+\ldots+\mu_n\beta_{2,n})+\\
+\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
+(\mu_1\beta_{n,1}+\mu_2\beta_{n,2}+\ldots+\mu_n\beta_{n,n}).
\end{gathered}$$

Величины, записанные в скобках, приравниваются цифрам α_i изображения числа в остаточных классах:

$$\left.\begin{array}{l}
\mu_1\beta_{1,1}+\mu_2\beta_{1,2}+\ldots+\mu_n\beta_{1,n}=\alpha_1;\\
\mu_1\beta_{2,1}+\mu_2\beta_{2,2}+\ldots+\mu_n\beta_{2,n}=\alpha_2;\\
\ldots\ldots\ldots\ldots\ldots\ldots\\
\mu_1\beta_{n,1}+\mu_2\beta_{n,2}+\ldots+\mu_n\beta_{n,n}=\alpha_n.
\end{array}\right\} \tag{7.12}$$

Система уравнений (7.12) имеет целочисленное решение при условии, что определитель системы равен ± 1:

$$\begin{vmatrix}
\beta_{1,1} & \beta_{1,2} & \ldots\ldots & \beta_{1,n}\\
\beta_{2,1} & \beta_{2,2} & \ldots\ldots & \beta_{2,n}\\
\ldots & \ldots & \ldots & \ldots\\
\beta_{n,1} & \beta_{n,2} & \ldots\ldots & \beta_{n,n}
\end{vmatrix}=\pm 1.$$

При этом получается большое количество возможных решений. Однозначное решение будет только в случае, если в определителе все члены равны 0, за исключением диагональных, которые должны быть равны 1. Отсюда следует, что

$$\left.\begin{array}{l}
B_1=(1,\ 0,\ 0\ldots\ldots 0);\\
B_2=(0,\ 1,\ 0\ldots\ldots 0);\\
B_3=(0,\ 0,\ 1\ldots\ldots 0);\\
\ldots\ldots\ldots\ldots\ldots\\
B_n=(0,\ 0,\ 0\ldots\ldots 1).
\end{array}\right\} \tag{7.13}$$

В этом случае говорят, что величины B_i обладают свойствами ортогональности.

Тогда в случае ортогональных базисов получаем, что

$$\mu_i = \alpha_i, \quad i = 1, 2, \ldots, n.$$

Из (7.13) следует, что базисы должны определяться по формуле

$$B_i = \varphi_{p_i} P/p_i, \tag{7.14}$$

где φ_{pi} — целое положительное число, называемое весом базиса.

Обычно

$$\varphi_{pi} P/p_i \equiv 1 \pmod{p_i}. \tag{7.15}$$

Это означает, что

$$\varphi_{pi} P/p_i = l_i p_i + 1,$$

где l_i — целое положительное число.

Для определения значений φ_{p_i} нужно решить систему уравнений вида (7.15). Обозначим $P/p_i = P_i'$. Пусть величина P_i' вычислена. Так как P_i' составлена из множителей, взаимно простых с p_i, то в результате деления P_i' на p_i получится остаток δ_i. Тогда в соответствии с (7.15) значение φ_{p_i} можно определить как решение сравнений вида

$$\varphi_{p_i}\delta_i \equiv 1 \quad \text{или} \quad \varphi_{p_i}\delta_i - 1 = kp_i.$$

Примечание. Решения подобных уравнений уже приведены, и результаты их можно найти в соответствующей литературе (см., например, [1]).

Рассмотрим методику определения базисов на конкретном примере.

Пример 7.3. Определить базисы для системы оснований $p_1=2$; $p_2=7$; $p_3=11$; $p_4=19$; $p_5=23$.

Решение. Диапазоном представления чисел является величина

$$P = \prod_{i=1}^{i=5} p_i = 67\,298.$$

Как следует из (7.14),

$$B_1 = \varphi_{p_1} P/p_1 = \varphi_{p_i} 7 \cdot 11 \cdot 19 \cdot 23;$$

$$\delta_i = P_1'/p_1 = (7 \cdot 11 \cdot 19 \cdot 23)/2 \equiv 1 \pmod 2; \quad \varphi_{p_1}\delta_1 \equiv 1 \pmod 2.$$

Отсюда $\varphi_{p_1} = 1$ и $B_1 = 1 \cdot 7 \cdot 11 \cdot 19 \cdot 23 = 33649$.

Для определения величины B_2 воспользуемся уравнением

$$B_2 = \varphi_{p_2} P_2'; \quad \delta_2 = (2 \cdot 11 \cdot 19 \cdot 23)/7 \equiv 3 \pmod 7,$$

или

$$\varphi_{p_2} 3 - 1 = k \cdot 7; \quad \varphi_{p_2} = (7k + 1)/3.$$

Последнее уравнение решаем методом подбора: при $k=0$; $k=1$; $k=2$ соответственно $\varphi_{p_2}=1/3$; $\varphi_{p_2}=8/3$; $\varphi_{p_2}=5$.

По условию, φ_{p_i} должно быть целым положительным числом. Значит, $\varphi_{p_2}=5$. Отсюда $B_2=5\cdot 2\cdot 11\cdot 19\cdot 23=48070$.

Величину B_3 определяем из условия, что $B_3=\varphi_{p_3}P_3'=\varphi_{p_3}2\cdot 7\cdot 19\cdot 23$; $\delta_3=(2\cdot 7\cdot 19\cdot 23)/11\equiv 2 \pmod{11}$.

Отсюда $2\varphi_{p_3}\equiv 1 \pmod{11}$, $\varphi_{p_3}=(11k+1)/2$. При $k=1$ $\varphi_{p_3}=6$ и $B_3=6\cdot 2\cdot 7\times \times 19\cdot 23=36\,708$.

Находим:

$$B_4=\varphi_{p_4}P_4'=\varphi_{p_4}2\cdot 7\cdot 11\cdot 23; \quad \delta_4=(2\cdot 7\cdot 11\cdot 23)/19\equiv 8 \pmod{19}.$$

Отсюда $\varphi_{p_4}=(19k+1)/8$. При $k=5$ $\varphi_{p_4}=12$, $B_4=42\,504$.

Величину B_5 определяем из уравнений

$$B_5=\varphi_{p_5}P_5'=\varphi_{p_5}2\cdot 7\cdot 11\cdot 19; \quad \delta_5=(2\cdot 7\cdot 11\cdot 19)/23\equiv 5 \pmod{23}.$$

Отсюда $\varphi_{p_5}5\equiv 1 \pmod{23}$; $\varphi_{p_5}=(23\,k+1)/5$. При $k=3$ $\varphi_{p_5}=14$, $B_5=40\,964$.

Переведем число $A=234=(0, 3, 3, 6, 4)$ в позиционную десятичную систему:

$$A=0\cdot 33\,649+3\cdot 48\,070+3\cdot 36\,708+6\cdot 42\,504+4\cdot 40\,964=673\,214.$$

Полученный результат превышает диапазон представления чисел во много раз. Чтобы перейти к истинному значению числа, необходимо вычесть из этого результата целое число раз величину P, т. е. $A=673\,214-10\cdot 67\,298=234$.

Ответ: $B_1=33\,649$, $B_2=48\,070$, $B_3=36\,708$, $B_4=42\,504$, $B_5=40\,964$.

В приведенном выше примере для вычисления величины числа A было использовано число 10, показывающее, во сколько раз превышен диапазон представления чисел, и называемое рангом числа (r_A).

Следовательно, (7.11) необходимо записать в виде

$$A=\alpha_1B_1+\alpha_2B_2+\ldots+\alpha_nB_n-r_AP,$$

или

$$A\equiv\alpha_1B_1+\alpha_2B_2+\ldots+\alpha_nB_n \pmod{P}. \tag{7.16}$$

§ 7.5. Формы представления чисел для системы остаточных классов

Одним из ограничений для системы остаточных классов, является то, что в этих системах можно представлять только целые положительные числа. Однако на практике почти не встречаются задачи, в которых информация ограничена указанной областью. Поэтому возникает необходимость представлять не только целые, но и дробные числа, для чего в системе остаточных классов вводится специальная запись дроби.

Для целых положительных чисел диапазон представления $P=p_1p_2...p_n$. В случаях, когда необходимо представлять не только положительные, но и отрицательные целые числа, целесообразно сделать так, чтобы положительные числа располагались в диапазоне от $P/2$ до P, а отрицательные — в диапазоне от 0 до $P/2$. Это означает условное перенесение начала отсчета числовой оси в точку с координатой $P/2$. Так как в качестве оснований, как правило, выбирают нечетные простые числа, то в этом случае получают неравнозначные пространства для изображения положительных и отрицательных чисел. Чтобы уравнять их, обычно начало отсчета переносят в точку с координатой $(P-1)/2$. Тогда говорят об искусственной форме представления чисел A', определяемой следующим образом:

если $A>0$, то $A'=(P-1)/2+|A|$;

если $A<0$, то $A'=(P-1)/2-|A|$.

Как было установлено ранее, для чисел, представленных в системе остаточных классов, справедливо соотношение (7.16). Преобразуем его, разделив обе части на величину P:

$$\frac{A}{P}=\frac{\alpha_1\varphi_{p_1}}{p_1}+\frac{\alpha_2\varphi_{p_2}}{p_2}+\ldots+\frac{\alpha_n\varphi_{p_n}}{p_n}-r_A=a. \tag{7.18}$$

Из (7.18) видно, что ранг числа r_A должен выбираться так, чтобы число входило в диапазон представления. В работах А. Свободы доказано, что каждому числу A' в искусственной форме соответствует изображение A^*, определяемое соотношением

$$\left.\begin{aligned} &A^*\equiv\Phi A' \,(\mathrm{mod}\, P);\\ &A_i^*\equiv\varphi_{p_i}\,\alpha_i' \,(\mathrm{mod}\, p_i), \end{aligned}\right\} \tag{7.19}$$

где $\Phi=(\varphi_{p_i},\varphi_{p_{2i}},\ldots,\varphi_{p_n})$ — преобразующая функция.

Соотношения (7.19) позволяют представить правильные дроби в системах остаточных классов, так как можно произвести замену остатков в выражении (7.18):

$$\frac{A^*}{P}=a^*=\frac{\alpha_1^*}{p_1}+\frac{\alpha_2^*}{p_2}+\ldots+\frac{\alpha_n^*}{p_n}-r_A, \tag{7.20}$$

где $0\leqslant a^*<1$, или для положительных и отрицательных дробей $-0{,}5\leqslant a^*<0{,}5$.

Таким образом, можно говорить о представлении правильных дробей в форме с фиксированной запятой.

Пример 7.4. Для заданной системы оснований $p_1=2$; $p_2=7$; $p_3=11$; $p_4=19$; $p_5=23$ представить число $A=-234=-(0,3,3,6,4)$ в форме с фиксированной запятой.

Решение. Прежде всего, пользуясь соотношением (7.19), вычисляем $A^*\equiv\Phi A'$ (mod P), где $A'=P-A=(0,4,8,13,19)$. Величина $\Phi=(1,5,6,12,14)$ находится из примера 7.3.

Следовательно,

$$A^*=(1,5,6,12,14)\cdot(0,4,8,13,19)=(0,6,4,4,13).$$

На основании (7.20) определяем

$$a^*=\frac{0}{2}+\frac{6}{7}+\frac{4}{11}+\frac{4}{19}+\frac{13}{23}-r_A.$$

Для того чтобы величина a^* входила в диапазон от 0,5 до —0,5, необходимо ранг числа принять равным $r_A=2$. Тогда $a^* \approx -0{,}00348$.

Проверка:

$$A = Pa^* = 67\,298\,(-0{,}00348) = -234.$$

Ответ: $A=(0,\ 6,\ 4,\ 4,\ 13)$.

§ 7.6. Выполнение элементарных операций в системе остаточных классов

Рассмотрим правила выполнения элементарных арифметических операций в системе остаточных классов над числами, представленными в искусственной форме.

Операция сложения. Пусть $A'=A+P$; $B'=B+P$ для $P=p_1p_2...p_n$.

Необходимо получить

$$(A+B)'=P+(A+B). \tag{7.21}$$

Если сложить числа, представленные в искусственной форме, то

$$A'+B'=A+B+2P. \tag{7.22}$$

Расширим систему оснований введением $p_0=2$. Вследствие того, что числа $p_1,\ p_2,\ ...,\ p_n$ — взаимно-простые, величина P изобразится в расширенной системе оснований:

$$P=(1,\ 0,\ 0\ldots 0) \quad \text{для} \quad p_0,\ p_1,\ p_2,\ldots,\ p_n.$$

Следовательно, $2P=(0,\ 0,\ 0,\ ...,\ 0)$, что позволяет упростить выражение (7.22):

$$A'+B'=A+B. \tag{7.23}$$

Сравнивая (7.23) и (7.21), находим, что

$$(A+B)'=A'+B'+P. \tag{7.24}$$

Выражение (7.24) позволяет получить результат сложения чисел в искусственной форме.

Пример 7.5. Для набора оснований (см. пример 7.4) определить сумму чисел $A=234$ и $B=-199$.

Решение. Запишем числа в системе остаточных классов:

$$A=(0,\ 3,\ 3,\ 6,\ 4);$$
$$B=-(1;\ 3,\ 19,\ 15);$$
$$P=7\cdot 11\cdot 19\cdot 23=33\,649.$$

Представим числа в искусственных формах:

$$A'=(0,\ 3,\ 3,\ 6,\ 4)+(1,\ 0,\ 0,\ 0,\ 0)=(1,\ 3,\ 3,\ 6,\ 4);$$
$$B'=(1,\ 0,\ 0,\ 0,\ 0)-(1,\ 3,\ 1,\ 9,\ 15)=(0,\ 4,\ 10,\ 10,\ 8).$$

Получим

$$C'=A'+B'+P=(1,\ 3,\ 3,\ 6,\ 4)+(0,\ 4,\ 10,\ 10,\ 8)+(1,\ 0,\ 0,\ 0,\ 0)=$$
$$=(0,\ 0,\ 2,\ 16,\ 12)=33\,649-35=33\,614.$$

Ответ: $A+B=(0,\ 0,\ 2,\ 16,\ 12)$.

Операция вычитания. Предположим, что нужно получить разность чисел:

$$C = A - B.$$

Тогда, пользуясь уже известным приемом, произведем замену знака числа B; вместо операции вычитания будет осуществляться операция сложения:

$$C = A + \overline{B}.$$

При переходе к искусственным формам необходимо учесть следующее:

если величина B была отрицательной, то после перемены знака она становится положительной, т. е. $\overline{B}' = P + B$;

если величина B была положительной, то после перемены знака она становится отрицательной, т. е. $\overline{B}' = P - B$.

Для машинной реализации это проще всего осуществить таким образом:

$$(-B') = 2P - (B').$$

что по смыслу представляет собой дополнительный код вычитаемого.

Пример 7.6. Пусть $A = 13$; $B = 8$; $p_0 = 2$; $p_1 = 3$; $p_3 = 7$; $P = 3 \cdot 5 \cdot 7 = 105 = (1, 0, 0, 0, 0)$. Найти $A - B$.

Решение. Сначала находим искусственные формы чисел:

$$A' = (0, 1, 3, 6); \quad B' = (1, 2, 3, 1).$$

Определяем дополнительный код для вычитаемого:

$$(-B') = 2P - (B') = (2, 3, 5, 7) = (1, 2, 3, 1) = (1, 1, 2, 6).$$

Результат: $A' + (-B') = (1, 2, 0, 5)$, откуда в силу соотношения (7.24)

$$(A - B)' = A'(-B') + P = (0, 2, 0, 5) = 105 + 5.$$

Ответ: $(A - B)' = (0, 2, 0, 5)$.

Операция умножения. Пусть для $p_1, p_2, \ldots, p_n$, где $p_1 = 2$, и $P = \prod\limits_{l=2}^{n} p_l$ заданы $A' = P + A$ и $B' = P + B$.

Тогда

$$A'B' = PP + PA + PB + AB = P(P + A + B) + AB. \qquad (7.25)$$

Результат должен быть равен

$$(AB)' = P + AB. \qquad (7.2$$

Из (7.25) и (7.26)

$$(AB)' = A'B' + P - P(P + A + B). \qquad (7.27$$

Так как величина P нечетная, то значение последнего члена выражения (7.27) зависит от четности A и B: если A и B одинаковой четности, то остатки от модуля p_1 у них одинаковы и при их сложении всегда будем получать первый остаток для величины в скобках, равный единице. В противоположном случае остаток будет равен нулю. Отсюда следует, что при умножении двух чисел окончательный результат определяется так:

$$(AB)' = \begin{cases} A'B' + P, & \text{если } A \text{ и } B \text{ разной четности;} \\ A'B', & \text{если } A \text{ и } B \text{ одинаковой четности.} \end{cases} \tag{7.28}$$

Задание для самоконтроля

1. Перевести десятичные числа $A=49$, $B=117$, $C=-264$, $D=352$ в систему остаточных классов при модулях $p_1=7$, $p_2=11$, $p_3=19$.

2. Перевести число $A=(0, 3, 4, 4, 6)$ из системы остаточных классов с основаниями $p_0=2$, $p_1=5$, $p_2=7$, $p_3=11$, $p_4=19$ в десятичную систему счисления.

3. Для оснований $p_0=2$, $p_1=5$, $p_2=7$, $p_3=11$ выполнить следующие операции над числами $A=55$, $B=-38$: а) $A+B$, б) $A-B$, в) AB.

4. Каков признак переполнения разрядной сетки при действиях над числами, представленными в системе остаточных классов?

5. Как изображается знак в числах, представленных в системе остаточных классов?

6. Можно ли для системы остаточных классов использовать табличный метод выполнения арифметических операций?

7. Найти базисы для системы остаточных классов с основаниями $p_1=2$, $p_2=11$, $p_3=19$, $p_4=23$, $p_5=29$.

8. Перевести в десятичную систему числа $A=(1, 6, 15, 20, 24)$, $B=(0, 3, 5, 6, 7)$, $C=(1, 10, 18, 22, 28)$, $D=(1, 1, 1; 1, 1)$, представленные в системе остаточных классов с основаниями $p_1=2$, $p_2=7$, $p_3=11$, $p_4=19$, $p_5=23$ (см. примеры 7.3).

9. Выполнить операции над числами $A+B$; $C-D$; AD; BD (значение чисел см п. 8).

8 ГЛАВА

ДЕСЯТИЧНАЯ АРИФМЕТИКА

§ 8.1. Д-коды

Д-код (двоично-кодированное представление) десятичного числа — *такое его представление, в котором каждая десятичная цифра изображается тетрадой из двоичных символов:*

$$A_{\text{Д}}=\{\alpha_4^{(1)}\,\alpha_3^{(1)}\,\alpha_2^{(1)}\,\alpha_1^{(1)}\}_1\,\{\alpha_4^{(2)}\,\alpha_3^{(2)}\,\alpha_2^{(2)}\,\alpha_1^{(2)}\}_2\ldots\{\alpha_4^{(n)}\,\alpha_3^{(n)}\,\alpha_2^{(n)}\,\alpha_1^{(n)}\}_n, \tag{8.1}$$

где $\alpha_i^{(j)}$ — двоичные разряды тетрады j; n — количество десятичных разрядов.

Примечание. Название «Д-код» образовано от полного названия «десятичный код».

Существуют различные Д-коды, что определяется количеством возможных сочетаний по 10 из 16 комбинаций, которые допускает тетрада.

При образовании Д-кода следует исходить из общих требований, предъявляемых к системам счисления:

различным десятичным цифрам должны соответствовать различные тетрады;

бóльшая десятичная цифра должна изображаться большей тетрадой (если разряды тетрады имеют веса по двоичной системе счисления);

для десятичных цифр $a=\{\alpha_4, \alpha_3, \alpha_2, \alpha_1\}$ и $b=\{\beta_4, \beta_3, \beta_2, \beta_1\}$, связанных соотношением $a+b=9$, должно удовлетворяться условие

$$\beta_i=\begin{cases}0, & \text{если } \alpha_i=1\\ 1, & \text{если } \alpha_i=0\end{cases}\quad (i=1,\ 2,\ 3,\ 4). \tag{8.2}$$

Для однозначности перевода чисел в Д-код и обратно желательно, чтобы разряды тетрад имели определенные веса. Тогда значение десятичной цифры a_i соответствует выражению

$$a_i=\alpha_4\sigma_4+\alpha_3\sigma_3+\alpha_2\sigma_2+\alpha_1\sigma_1, \tag{8.3}$$

где σ_i — вес разряда тетрады.

В табл. 8.1 представлено кодирование десятичных цифр в различных Д-кодах.

Таблица 8.1

Десятичные цифры	Эквиваленты в кодах						
	Д₁ (система 8421)	Д₂ (система 2421)	Д₃ (система 5121)	Д₄ (система 8421+3)	Д₅ (система 53—21)	Д₆ (система 75—31)	Д₇ (система 5421)
0	0000	0000	0000	0011	0000	0000	0000
1	0001	0001	0001	0100	0001	0001	0001
2	0010	0010	0010	0101	0111	0110	0010
3	0011	0011	0011	0110	1010	0111	0011
4	0100	0100	0111	0111	0101	1010	0100
5	0101	1011	1000	1000	1000	0100	1000
6	0110	1100	1001	1001	1001	0101	1001
7	0111	1101	1010	1010	1111	1000	1010
8	1000	1110	1011	1011	1100	1001	1011
9	1001	1111	1111	1100	1101	1110	1100

В таблице для каждого Д-кода указаны разрешенные комбинации. Все другие комбинации — запрещенные.

Наличие разрешенных и запрещенных комбинаций — очень важное свойство Д-кодов. Оно отличает их от обычных позиционных систем счисления, в которых все комбинации — разрешенные.

Рассмотрим наиболее распространенные Д-коды.

Код $Д_1$ прямого замещения (система 8421). В коде $Д_1$ разрешенные комбинации соответствуют двоичным эквивалентам десятичных цифр с весами разрядов, равных степеням основания 2. Для этого кода не выполняется условие (8.2), так как цифры, являющиеся дополнением до 9, не получаются простым инвертированием наборов тетрад.

Код $Д_2$ (система 2421). Для кода $Д_2$ веса разрядов тетрады соответственно равны 2, 4, 2, 1; таблица кодирования делится на две части: от 0 до 4 — тетрады повторяют двоичные эквиваленты; от 5 до 9 — по сравнению с двоичной системой каждая тетрада содержит избыток +0110. Это дает возможность любую цифру одной части таблицы превратить в ее дополнение до 9 простым инвертированием.

Код $Д_4$ (система 8421+3). Для этого кода все тетрады имеют значения, на три единицы большие по сравнению с кодом $Д_1$ (отсюда название кода), и для него не существует целочисленных значений весов, которые удовлетворяли бы (8.3).

Коды $Д_5$ (система 53—21) и $Д_6$ (система 75—31). Эти коды отличаются от вышеназванных кодов тем, что для них некоторые веса имеют отрицательное значение.

В вычислительных машинах разного назначения чаще всего используются коды $Д_1$ и $Д_4$.

§ 8.2. Правила сложения в Д-кодах

При сложении чисел в Д-коде возникают некоторые трудности, обусловленные необходимостью вырабатывать десятичный перенос и производить коррекцию результата, из-за того, что тетрада дает 16 различных комбинаций, а в любом Д-коде используется только 10 из них.

Пусть заданы числа

$$A_Д = a_n a_{n-1} \dots a_1 a_0 \quad \text{и} \quad B_Д = b_n b_{n-1} \dots b_1 b_0,$$

где a_i и b_i — десятичные цифры, представленные тетрадами.

Тогда

$$C_Д = A_Д + B_Д \text{ и } c_i = a_i + b_i + П_{i-1} - П_i q,$$

где $П_i = \{0, 1\}$, $П_{i-1}$ — десятичные переносы; $q = 10$ — основание системы счисления.

Дальнейший вывод правил сложения надо рассматривать применительно к Д-кодам.

Код $Д_1$. При сложении чисел в коде $Д_1$ могут возникнуть следующие случаи:

1. Пусть $a_i + b_i + П_{i-1} < 10$, где a_i, b_i — тетрады для кода $Д_1$. При сложении в данном разряде числа образуется сумма меньше 10. Если действия над разрядами тетрады производят по правилам двоичной арифметики, то правильный результат получают без коррекции.

Пример 8.1. Сложить тетрады $a_i = 0100$ и $b_i = 0101$ при значении $П_{i-1} = 0$.
Решение.

$$c_i = a_i + b_i + П_{i-1} = 1001.$$

Ответ: $c_i = 1001$.

2. Пусть $a_i + b_i + П_{i-1} \geqslant 10$, т. е. возникает десятичный перенос и сумма должна быть равна $a_i + b_i + П_{i-1} - П_i \cdot 10$, где $П_i = 1$.

Свидетельством того, что результат неправильный, является либо появление запрещенной комбинации, если $15 \geqslant a_i + b_i + П_{i-1} \geqslant 10$, либо появление потетрадного переноса $П_i' = 16$, что превышает значение десятичного переноса на 6. Следовательно, требуется коррекция результата в данной тетраде введением поправки, равной +0110.

Пример 8.2. Сложить тетрады $a_i = 0101$ и $b_i = 1001$, при значении $П_{i-1} = 1$.
Решение. $c_i' = a_i + b_i + П_{i-1} = 1111$.
Величина $c_i' = 1111$ — запрещенная комбинация. Следовательно, надо ввести поправку:

$$\begin{array}{r} + \begin{array}{r} 1111 \\ 0110 \end{array} \\ \hline \underset{\substack{\uparrow \\ П_i}}{\boxed{1}}\ 0101, \end{array}$$

т. е. результат равен 0101 в даниой тетраде и образовал перенос в старшую тетраду.

Ответ: $C=0101$, $\Pi_i=1$.

Пример 8.3. Сложить тетрады $a_i=0111$, $b_i=1001$ при значении $\Pi_{i-1}=1$.

Решение.

$$c_i' = a_i + b_i + \Pi_{i-1} = \underset{\substack{\uparrow \\ \Pi_i'}}{\boxed{1}}\ 0001.$$

Появление потетрадного переноса требует коррекции результата:

$$c_i = 0001 + 0110 = 0111.$$

Ответ: $c_i=0111$, $\Pi_i=1$.

Таким образом, *если в данной тетраде сумма цифр с переносом из младшей соседней тетрады меньше* 10, *то сложение производится без поправок; если же сумма цифр с переносом равна или больше* 10, *то происходит коррекция результата введением поправки* 0110.

Пример 8.4. Сложить числа $A=279=0010\,0111\,1001$ и $B=581=0101\,1000\,0001$.

Решение. Прежде всего производится потетрадное суммирование, а затем коррекция там, где это необходимо:

$$\begin{array}{rcccl} A= & 0010 & 0111 & 1001 & \\ & + & & & \\ B= & 0101 & 1000 & 0001 & \\ \hline & 0111 & 1111 & 1010 & \\ & + & & & \\ & & 0110 & 0110 & \text{поправки} \\ \hline C= & 1000 \leftarrow & 0110 \leftarrow & 0000 & \end{array}$$

Здесь стрелка указывает передачу единицы десятичного переноса.

Ответ: $C=860=1000\,0110\,0000$.

Код Д$_2$. При сложении чисел в коде Д$_2$ могут возникнуть следующие случаи:

1. Пусть $a_i'<5$ и $b_i'<5$, где a_i', b_i' — тетрады для кода Д$_2$:

если $a_i'+b_i'+\Pi_{i-1}<5$ — результат сложения не требует коррекции;

если $a_i'+b_i'+\Pi_{i-1}\geqslant 5$ — результат попадает во вторую часть таблицы кодирования, где $c_i'=c_i+6$. Следовательно, здесь необходима коррекция результата введением поправки 0110. Признаком этого является появление запрещенных комбинаций.

2. Пусть $a_i'\geqslant 5$ и $b_i'<5$; $15\geqslant a_i'+b_i'+\Pi_{i-1}\geqslant 5$.

Так как $a_i'=a_i+6$, то при суммировании поправка не требуется.

3. Пусть $a_i'\geqslant 5$ и $b_i'\geqslant 5$:

если $10\leqslant a_i'+b_i'+\Pi_{i-1}\leqslant 15$ — результат требует коррекции путем введения поправки — 0110;

если же $a_i'+b_i'+\Pi_{i-1}>15$ — результат не требует коррекции.

Пример 8.5. Сложить числа $A=137=0001\ 0011\ 1101$ и $B=457=0100\ 1011\ 1101$.

Решение. Сиачала производится потетрадиое суммирование, а затем коррекция:

$$
\begin{array}{rcccl}
A= & 0001 & 0011 & 1101 & \\
 & + & & & \\
B= & 0100 & 1011 & 1101 & \\
\hline
 & 0101 & 1111 \leftarrow & 1010 & \\
 & + & + & + & \\
 & 0110 & 0000 & 0110 & \text{поправки} \\
\hline
C= & 1011 & 1111 & 0100 &
\end{array}
$$

Ответ: $C=594=0011\ 1111\ 0100$.

Примечание. Коррекция в коде $Д_2$ осуществляется потетрадно с блокировкой цепей переноса между ними.

Код $Д_4$. При сложении чисел в коде $Д_4$ возможны следующие случаи.

Пусть $a_i''=a_i+3$, $b_i''=b_i+3$, где a_i'' и b_i'' — тетрады для кода $Д_4$:

если $a_i''+b_i''+П_{i-1}<10$, то

$$c_i''=a_i+3+b_i+3+П_{i-1}=\underbrace{a_i+b_i+П_{i-1}+3}_{c_i''}+3-$$

результат требует коррекции путем введения поправки — 0011;

если $a_i''+b_i''+П_{i-1}\geqslant 10$, то $c_i''=\underbrace{a_i+b_i+П_{i-1}+3}_{c_i''}+3$.

Здесь возникает десятичный перенос, который, по условию, «уносит» с собой шесть избыточных комбинаций. Следовательно, в данном случае требуется коррекция результата путем введения поправки +0011.

Пример 8.6. Сложить числа $A=35=0110\ 1000$ и $B=28=0101\ 1011$.

Решение. Сначала производится потетрадиое сложеиие, а затем введеине поправок:

$$
\begin{array}{rccl}
A= & 0110 & 1000 & \\
 & + & & \\
B= & 0101 & 1011 & \\
\hline
 & 1100 \leftarrow & 0011 & \\
 & - & + & \\
 & 0011 & 0011 & \text{поправки} \\
\hline
C= & 1001 & 0110 &
\end{array}
$$

Ответ: $C=63=1001\ 0110$.

Здесь поправки вводятся при блокировке цепей потетрадного переноса.

Правила введения поправок можно сформулировать так:

если при сложении тетрад не возникает переноса ($П_i=0$), *то поправка равна* —0011 (*или дополнение* +1101); *если же возникает потетрадный перенос* ($П_i=1$), *то поправка равна* +0011.

Аналогичным образом можно рассмотреть правила суммирования для других Д-кодов.

§ 8.3. Представление отрицательных чисел в Д-кодах

Представление Д-кода в разрядной сетке машины может осуществляться в форме либо с фиксированной, либо с плавающей запятой. При этом отрицательные числа должны представляться в прямом, обратном или дополнительном кодах.

Поэтому, если $A = -0, a_1 a_2 \dots a_n$,
где a_i — тетрады, то

$$\left.\begin{aligned} [A]_{пр} &= 1,\ a_1 a_2 \dots a_n; \\ [A]_{об} &= 1,\ \bar{\bar{a}}_1 \bar{\bar{a}}_2 \dots \bar{\bar{a}}_n; \\ [A]_{д} &= 1,\ \bar{a}_1 \bar{a}_2 \dots \bar{a}_n, \end{aligned}\right\} \tag{8.4}$$

где $\bar{\bar{a}}_i$ — дополнение до $q-1$ во всех тетрадах; $\bar{a}_i$ — дополнение до $q-1$ во всех тетрадах, за исключением младшей, для которой это дополнение до $q=10$.

На основании правил преобразования (8.4) следует, что

$$\bar{\bar{a}}_i + a_i = q - 1. \tag{8.5}$$

Это означает, что для Д-кодов, для которых выполняется условие (8.2), обратный код получается простым инвертированием набора тетрад.

Пример 8.7. Найти обратный и дополнительный коды в коде $Д_2$ для числа $A = -0{,}127 = -0{,}0001\ 0010\ 1101$.

Решение. На основании (8.4)

$$[A]_{об} = 1{,}1110 \quad 1101 \quad 0010.$$

Используя соотношение $[A]_{об} + q^{-n} = [A]_{д}$, находим дополнительный код:

$$[A]_{д} = 1{,}1110 \quad 1101 \quad 0011.$$

Ответ $[A]_{об} = 1{,}1110\ 1101\ 0010.$
$[A]_{д} = 1{,}\ 1110\ 1101\ 0011.$

Примечание. Прибавление единицы в младшую тетраду при образовании дополнительного кода в коде $Д_2$ осуществляется по правилам сложения для этого кода.

Пример 8.8. Найти обратный и дополнительный коды в коде $Д_4$ для числа $A = -0{,}4591 = 0{,}0111\ 1000\ 1100\ 0100$.

Решение. На основании (8.4)

$$\left.\begin{aligned} [A]_{об} &= 1{,}1000 \quad 0111 \quad 0011 \quad 1011; \\ [A]_{д} &= 1{,}1000 \quad 0111 \quad 0011 \quad 1100. \end{aligned}\right\} \tag{a}$$

Ответ: см. формулу (а) данного примера.

Примечание. Прибавление единицы в младшую тетраду при образовании дополнительного кода в коде $Д_4$ не требует коррекции.

Код $Д_1$ отличается тем, что для него не выполняется условие (8.2). Эта особенность кода влияет на образование обратного или дополнительного кода, так как инвертирование набора тетрад означает получение дополнения до $2^4 - 1 = 15$. Следовательно, необходимо убрать разницу. Один из используемых при этом приемов состоит в том, что во все цифровые тетрады числа в коде $Д_1$ добавляется $+0110$ и после этого производится инвертирование набора. Полученное изображение представляет собой обратный код числа.

Пример 8.9. Получить обратный код в коде $Д_1$ для числа $A=-0{,}256=-0{,}0010\ 0101\ 0110$.

Решение. Сначала во все тетрады добавляется 0110:

$$\begin{array}{r} 0{,}0010 \\ +\ 0110 \\ \hline 0{,}1000 \end{array}\quad \begin{array}{r} 0101 \\ +\ 0110 \\ \hline 1011 \end{array}\quad \begin{array}{r} 0110 \\ +\ 0110 \\ \hline 1100 \end{array}.$$

После инвертирования этого набора получаем

$$[A]_{об}=1{,}0111\quad 0100\quad 0011.$$

Ответ: $[A]_{об}=1{,}0111\ 0100\ 0011$.

§ 8.4. Выполнение операций сложения и вычитания в Д-кодах

Все арифметические действия в Д-кодах выполняются над операндами по формальным правилам двоичной арифметики, сформулированным выше.

Возникающие при этом специфические особенности целесообразнее рассмотреть на конкретных примерах.

Пример 8.10. Сложить в коде $Д_1$ на сумматоре дополнительного кода числа $A=-0{,}1000\ 0010\ 0101$ и $B=0{,}1001\ 0100\ 0110$.

Решение. Исходные числа представляются в дополнительном коде и их изображения складываются:

$$\begin{array}{rcccc} [A]_{д}= & 1, & 0001 & 0111 & 0101 \\ + & & & & \\ [B]_{д}= & 0, & 1001 & 0100 & 0110 \\ \hline & 1, & 1010 & 1011 & 1011 \\ + & & & & \\ & & 0110 & 0110 & 0110 \\ \hline & 0, \leftarrow & 0001 \leftarrow & 0010 \leftarrow & 0001. \end{array}\quad \text{поправки}$$

Ответ: $C=0{,}0001\ 0010\ 0001$.

Пример 8.11. Сложить в коде $Д_1$ на сумматоре обратного кода числа $A=-0{,}0100\ 1000\ 0111$ и $B=-0{,}0010\ 0011\ 0110$.

Решение. Исходные числа представляются в обратном коде, и их изображения складываются:

$$\begin{array}{rcccc} [A]_{об}= & 1, & 0101 & 0001 & 0010 \\ + & & & & \\ [B]_{об}= & 1, & 0111 & 0110 & 0011 \\ \hline & 0, & 1100 & 0111 & 0110 \\ + & & & & \\ & & 0110 & 0000 & 0000 \\ \hline & 1, \leftarrow & 0010 & 0111 & 0110. \end{array}\quad \text{поправки}$$

Ответ получаем после преобразования результата из обратного кода.
Ответ: $A+B=-0{,}0111\ 0010\ 0011$.

Примечание. При коррекции в коде $Д_1$ цепи межтетрадного переноса в сумматорах не блокируются.

Пример 8.12. Сложить в коде $Д_2$ на сумматоре дополнительного кода числа $A = -0,0000\ 1101\ 0011\ 1011$ и $B = -0,0000\ 1111\ 0010\ 1011$.

Решение. Получение дополнительного кода здесь представляет определенный интерес. Если все делать по правилам, то

$$[A]_д = 1,\quad 1111\quad 0010\quad 1100\quad 0101.$$

В последней тетраде получена запрещенная комбинация, что свидетельствует о необходимости коррекции путем введеиия поправки 0110:

$$\begin{array}{r} 0101 \\ +\ 0110 \\ \hline 1011. \end{array}$$

Аналогично для второго числа получаем в последней тетраде 1011.

Следовательно,

$$\begin{array}{rcccccl}
[A]_д = & 1, & 1111 & 0010 & 1100 & 1011 & \\
+ & & & & & & \\
[B]_д = & 1, & 1111 & 0000 & 1101 & 1011 & \\ \hline
& 1, \leftarrow & 1110 & 0011 \leftarrow & 1010 \leftarrow & 0110 & \\
& + & & + & + & + & \\
& & 0000 & 0000 & 1010 & 1010 & \text{поправки} \\ \hline
[A+B]_д = & 1, & 1110 & 0011 & 0100 & 0000. &
\end{array}$$

Производится обратное преобразование дополнительного кода.

Ответ: $A+B = -0,0001\ 1100\ 1100\ 0000$.

Пример 8.13. Сложить на сумматоре обратного кода (система $Д_2$) числа $A=0,1110\ 1011\ 0011$ и $B= -0,1100\ 0100\ 1011$.

Решение. Сначала числа записываются в обратном коде и их изображения складываются:

$$\begin{array}{rccccl}
[B]_{об} = & 1, & 0011 & 1011 & 0100 & \\
+ & & & & & \\
[A]_{об} = & 0, & 1110 & 1011 & 0011 & \\ \hline
& 0, \leftarrow & 0010 \leftarrow & 1\ 0110 & 1000 & \\
& + & & + & + & \\
& & 0000 & 1010 & 0110 & \text{поправки} \\ \hline
A+B = & 0, & 0010 & 0000 & 1110. &
\end{array}$$

Ответ: $A+B=0,0010\ 0000\ 1110$.

Пример 8.14. Сложить в коде $Д_4$ на сумматоре обратного кода числа $A= -0,1011\ 1100\ 0111\ 1001$ и $B=0,0011\ 1000\ 1001\ 1010$.

Решение. Сначала числа записываются в обратном коде и затем складываются изображения:

$$\begin{array}{rcccccl}
[A]_{об} = & 1, & 0100 & 0011 & 1000 & 0110 & \\
+ & & & & & & \\
[B]_{об} = & 0, & 0011 & 1000 & 1001 & 1010 & \\ \hline
& 1, & 0111 & 1100 \leftarrow & 0010 \leftarrow & 0000 & \\
& & + & + & + & + & \\
& & 1101 & 1101 & 0011 & 0011 & \text{поправки} \\ \hline
[A+B]_{об} = & 1, & 0100 & 1001 & 0100 & 0011. &
\end{array}$$

Ответ: $C= -0,1011\ 0110\ 1011\ 1100$.

Примечание. При введении поправок цепи межтетрадного переноса блокируются и отрицательные поправки вносятся в виде дополнения (1101).

Рассмотренные выше примеры выполнения операций сложения в Д-кодах позволяют сделать ряд общих замечаний: коррекция результата может осуществляться либо автоматически программным путем, либо с помощью аппаратных средств. Первый метод потребует разработки специального блока управления, а второй — усложнения схемы собственно сумматора. Эту задачу разработчик решает в конкретной постановке в зависимости от требований. На практике чаще используют схемный метод коррекции.

По изложенным выше правилам реализуются алгоритмы сложения (вычитания) чисел, представленных в форме с фиксированной запятой на специальных десятичных сумматорах.

Для десятичных чисел с плавающей запятой используют ту же методику, что и для двоичных чисел: порядки чисел перед сложением выравниваются (меньшему числу присваивается порядок большего числа) и по окончании операции производится нормализация результата. При этом со стороны младших разрядов отводится дополнительная тетрада, используемая при сдвигах вправо.

§ 8.5. Умножение чисел в Д-кодах

Выполнение операций умножения в Д-кодах принципиально производится по классической схеме:

умножение чисел сводится к последовательному суммированию частных произведений, полученных при умножении множимого на очередную цифру множителя. Так как каждая цифра множителя представляется в виде $(\beta_4, \beta_3, \beta_2, \beta_1)_i$, где i — номер разряда, то умножение сопровождается расшифровкой значения очередной тетрады множителя и сдвигом на четыре разряда сразу. Расшифровку можно осуществить разными способами. Простейшим приемом является последовательное вычитание единицы из значения тетрады до получения нуля и соответственно прибавление множимого в сумматор на каждом такте. При умножении на сумматоре прямого кода надо предусмотреть дополнительную тетраду на случай местного переполнения.

Пример 8.15. Умножить на сумматоре прямого кода (код $Д_2$) числа

$$[A]_{пр} = 1,\quad 0011\quad 1101;$$

$$[B]_{пр} = 0,\quad 0010\quad 0100.$$

Решение. При умножении используются сумматор прямого кода для кода $Д_2$ на три тетрады и регистры на две тетрады.

Последовательность выполнения операции показана в табл. 8.2.

Ответ: $[AB]_{пр} = 1,0000\ 1110\ 1110\ 1110.$

В некоторых машинах единой системы (например, ЕС-1020) при умножении десятичных чисел в коде $Д_1$ используется метод ускорения операции, при котором вся операция сводится к последовательному выполнению сложения или вычитания на сумматоре до-

Таблица 8.2

Сумматор ($\overrightarrow{СМ}$)			Регистр ($\overrightarrow{Рг\,B}$)		Примечание
0000	0000	0000	0010	0100	И.П. СМ:=0; Рг B:=$[B]_{пр}$; Рг A:=$[\dot{A}]_{пр}$
	+ 0011	1101		− 0001	
	0011	1101		0011	
	+ 0011	1101		− 0001	
	1101	0100		0010	Анализ тетрады b_1
	+ 0011	1101		− 0001	Вычитание единицы из содержимого тетрады
0001	0001	0001		0001	
	+ 0011	1101		− 0001	
				0000	Конец анализа
0001	0100	1110			Сдвиг на четыре разряда
0000	0001	0100	1110	0010	
	+ 0011	1101		− 0001	Анализ тетрады b_2
	1011	0001		0001	
	+ 0011	1101		− 0001	
				0000	Конец анализа
0000	0000	1110	1110	1110	Сдвиг на четыре разряда
					Конец

Примечание. При умножении в коде Д$_2$ анализ тетрад можно осуществлять с помощью реверсивного счетчика.

полнительного кода в зависимости от величины очередной цифры множителя:

Очередная цифра множителя	Вид операции	Очередная цифра множителя	Вид операции
0	0	5	$+2A+2A+1A$
1	$+1A$	6	$-2A-2A$
2	$+2A$	7	$-2A-1A$
3	$+2A+1A$	8	$-2A$
4	$+2A+2A$	9	$-1A$

Таким образом, предварительно должны быть подготовлены множимое и удвоенное множимое. При этом если предыдущая цифра была больше пяти, то действие на очередном шаге надо увеличить на $+1A$.

При умножении десятичных чисел часто используют другие приемы, позволяющие ускорить эту операцию. Например,

$$\left.\begin{aligned} AB&=2A\frac{B}{2}, && \text{если } B \text{ — четное число;}\\ AB&=2A\frac{B-1}{2}+A, && \text{если } B \text{ — нечетное число.}\end{aligned}\right\} \tag{8.6}$$

Действия в соответствии с (8.6) можно рассмотреть на примере десятичных чисел: $35\cdot42=2\cdot35\cdot\frac{42}{5}=70\cdot21=140\cdot10+70=280\times$ $\times5+70=560\cdot2+280+70=1120\cdot1+280+70=1470.$

При двоично-десятичном представлении чисел прием (8.6) может быть упрощен, так как удвоение числа означает сдвиг влево, а деление на 2 — сдвиг вправо. Поскольку сдвиги производятся над двоичными кодами, то требуется коррекция тетрад на каждом шаге. Корректирующие поправки определяются для каждого Д-кода. Коррекция выполняется тогда, когда происходит сдвиг единицы из крайнего разряда данной тетрады в соседнюю тетраду. Например, для кода $Д_1$ корректирующая поправка равна +0110 для тетрад множимого и —0011 (или +1101) для тетрад множителя.

Пример 8.16. (см. [7]). Умножить числа $A=0010\,0100$ и $B=0100\,0011$ методом (8.6) в коде $Д_1$.

Р е ш е н и е. Схема умножения выглядит следующим образом:

	B			*A*				*C*			
	0100	0011	0000	0010	0100		0000	0000	0010	0100	
(→)	0010	0001	0000	0100	1000	(←) +	0000	0000	0100	1000	сдвиг
(→)	0001	0000	0000	1001	0000	(←)	0000	0000	0110	1100	сдвиг
					+0110					+0110	поправки
				1010	0110		0000	0000	0111	0010	
(→)	0000	1000	0001	0010	1100	(←)+					сдвиг
		+1101		+0110	+0110						поправки
	0000	0101	0001	1001	0010		0000	0001	1001	0010	
							0000	0010	0000	0100	
									+0110		поправки
							0000	0010	0110	0100	
(→)	0000	0010	0011	0010	0100	(←)					сдвиг
				+0110							поправки
			0011	1000	0100						

```
(→) 0000   0001   0111   0000   1000 (←)  +                          сдвиг
                        +
                         0110                                         поправки
                  ------------------------
                  0111   0110   1000          0000   0111   0110   1000
(→) 0000   0000   1110   1101   0000 (←)      -------------------------
                                              0000   1001   1100   1100  сдвиг
                                                          +      +
                                                          0110   0110  поправки
                                              -------------------------

                                              0000   1010   0011   0010
                                                   +
                                                     0110                 поправки
                                              -------------------------
                                              0001   0000   0011   0010
```

Конец.

Ответ: AB=0001 0000 0011 0010.

§ 8.6. Деление чисел в Д-кодах

Деление десятичных чисел в Д-кодах выполняется методом последовательного вычитания делителя из делимого на первом шаге и из остатков — на последующих шагах. Вычитание на каждом шаге производится до тех пор, пока не получится отрицательный остаток. Каждый раз при получении положительного остатка добавляется единица в специальный счетчик, где накапливается очередная цифра частного. Затем осуществляется сдвиг на четыре двоичных разряда и прибавление делителя до тех пор, пока не получится положительный остаток. Количество сложений (без последнего) является дополнением соответствующей цифры частного до 9, что заносится в счетчик очередной цифры частного.

Таким образом, процесс деления состоит из ряда последовательно чередующихся циклов сложения и вычитания со сдвигами. Знак частного получается как логическая сумма по модулю 2 знаков чисел.

Все действия при выполнении операции деления должны осуществляться на сумматоре дополнительного (обратного) кода, работающего по правилам сложения — вычитания соответствующего Д-кода.

Для простоты рассмотрим пример деления в десятичной системе счисления, не прибегая к представлению чисел в виде тетрад.

Пример 8.17. Разделить числа A=0,154675 (делимое) и B=0,550 (делитель).

Р е ш е н и е. Установка исходного положения.

Ш а г 1. Осуществляется пробное вычитание: если результат будет положительный или равен 0, то вырабатывается сигнал прерывания, если — отрицательный, то производится сдвиг на одну тетраду.

В данном случае получается отрицательный остаток (рассматривается только цифровая часть чисел).

Шаг 2:

$$
\begin{array}{lrl}
 & -\ 154\,675 & \text{СЧ}:=0 \\
 & 55\,000 & \\ \hline
 & 99\,675 & \text{СЧ}:=0+1 \\
 & -55\,000 & \\ \hline
 & 44\,675 & \text{СЧ}:=1+1 \\
 & -55\,000 & \\ \hline
\text{Остаток}<0 & 89\,675 & c_1=2
\end{array}
$$

Шаг 3:

$$
\begin{array}{lrl}
 & +\ 89\,675 & \text{СЧ}:=9 \\
 & 5\,500 & \\ \hline
 & +\ 95\,175 & \text{СЧ}:=9-1 \\
 & 5\,500 & \\ \hline
\text{Остаток}>0 & 00\,675 & c_2=8
\end{array}
$$

Шаг 4:

$$
\begin{array}{lrl}
 & -\ 00\,675 & \text{СЧ}:=0 \\
 & 550 & \\ \hline
 & 00\,125 & \text{СЧ}:=0+1 \\
 & -\ 550 & c_3=1 \text{ и т. д.} \\ \hline
\text{Остаток}<0 & 99\,575 &
\end{array}
$$

Ответ: $C=0{,}281$.

Алгоритм десятичного деления, подобный рассмотренному, используют в машинах ЕС ЭВМ.

Для ускорения операции деления применяют приемы, аналогичные приемам, употребляемым для ускорения выполнения операции умножения.

Пусть A — делимое, B — делитель, C — частное. Предположим, что удалось отыскать частное в виде

$$C=0,\quad c_1c_2\ldots c_n.$$

Тогда

$$A=B(c_1 2^{-1}+c_2 2^{-2}+\ldots+c_n 2^{-n})+R_n, \tag{8.7}$$

где R_n — остаток от деления.

Пусть $R_n=0$. Положим, что $c_1=1$, $c_2=c_3=\ldots=0$. Тогда остаток на первом шаге

$$R_1=A-2^{-1}B.$$

Если $R_1\geqslant 0$, то $c_1=1$; если $R_1<0$, то $c_1=0$. В последнем случае восстанавливается предыдущий положительный остаток. Затем принимаем, что $c_2=1$, а остальные $c_i=0$ и т. д.

Остаток на любом шаге

$$R_l = A - B\sum_{i=1}^{n} c_i \cdot 2^{-l}. \tag{8.8}$$

Таким образом, при делении десятичных чисел, заданных в Д-коде, результат получается в двоичной системе счисления.

Пример 8.18. Найти частное от деления $A=0{,}2425$ (делимое) на $B=0{,}5200$ (делитель).

Решение. Для наглядности выполним этот пример в десятичной системе счисления с записью результата в двоичном коде, предполагая, что при переходе к определенному Д-коду все операции над десятичными числами будут выполняться по правилам этого Д-кода.

Последовательность выполнения операции деления дана в табл. 8.3.

Ответ: $C=0{,}011101$.

Таблица 8.3

Делитель (B) на i-м шаге	Сумматор (СМ)	Примечание	Цифры c_i
0,5200	0,2425	И. П.	
0,2600	− 0,2600	$B2^{-1}$; СМ:=[СМ]−$B2^{-1}$	
	9,9825	остаток отрицательный	$c_1=0$
	+ 0,2600	восстановление R_1	
	0,2425		
0,1300	− 0,1300	$B2^{-2}$; СМ:=[СМ]−$B2^{-2}$	
	0,1125	остаток положительный	$c_2=1$
0,0650	− 0,0650	$B2^{-3}$; СМ:=[СМ]−$B2^{-3}$	
	0,0475	остаток положительный	$c_3=1$
0,0325	− 0,0325	$B2^{-4}$; СМ:=[СМ]−$B2^{-4}$	
	0,0150	остаток положительный	$c_4=1$
0,0162	− 0,0162	$B2^{-5}$; СМ: =[СМ]−$B2^{-5}$	
	9,9938	остаток отрицательный	$c_5=0$
	+ 0,0162	восстановление R_5	
0,0081	0,0150		
	− 0,0081	$B2^{-6}$; СМ:=[СМ]−$B2^{-6}$	
	0,0069	остаток положительный	$c_6=1$
	и т. д.	Конец	

Так как при сдвиге чисел, представленных в Д-коде, приходится вводить поправки, то для хранения делителя целесообразно иметь самостоятельный сумматор, в котором при передаче единицы из одной тетрады в другую автоматически вносится поправка.

§ 8.7. Извлечение квадратного корня в Д-кодах

В основу алгоритмов извлечения квадратного корня в десятичной системе может быть с некоторыми уточнениями положена формула (5.8). Если для двоичной системы счисления в качестве нулевого приближения берут величину $y_0=2^{E(m/2)}$, где $E(m/2)$ — целая часть числа $m/2$, m — наибольший показатель степени основания системы, присутствующего в заданном числе, то для десятичной системы правильный выбор нулевого приближения определяет число итераций, необходимых для получения заданной точности. Как известно, точность результата ограничивается числом разрядов сумматора, на котором производится действие. Так, например, если имеется шестиразрядный сумматор и на шаге i промежуточное значение результата y_i имеет одну верную цифру, то можно предположить, что y_{i+1} будет иметь по крайней мере две верных и одну сомнительную цифры, для получения которых используется пять разрядов сумматора. Для проверки и выявления правильных цифр необходимо получить их на $(i+2)$-м шаге. Но из-за ограниченности разрядной сетки на этом шаге можно получить только три цифры. Таким образом, на шестиразрядном сумматоре в результате можно получить две верные и одну сомнительную цифры. Из-за этого обстоятельства итерационный метод нахождения корня является нецелесообразным для двоично-десятичного представления чисел. Рассмотрим несколько иной подход.

В десятичной системе счисления целое число можно представить в виде степенного полинома по основанию 10:

$$A=a_n\cdot 10^n=a_{n-1}\cdot 10^{n-1}+a_{n-2}\cdot 10^{n-2}+\ldots+a_1\cdot 10+a_0.$$

Квадрат числа A можно записать так:

$$\begin{gathered}A^2=a_n^2\cdot 10^{2n}+2a_n\cdot 10^n(a_{n-1}\cdot 10^{n-1}+a_{n-2}\cdot 10^{n-2}+\ldots\\ \ldots+a_1\cdot 10+a_0)+(a_{n-1}\cdot 10^{n-1}+a_{n-2}\cdot 10^{n-2}+\ldots+a_1\cdot 10+a_0)^2=\\ =a_n^2\cdot 10^{2n}+2a_n\cdot 10^n(a_{n-1}\cdot 10^{n-1}+a_{n-2}\cdot 10^{n-2}+\ldots\\ \ldots+a_1\cdot 10+a_0)+a_{n-1}^2\cdot 10^{2(n-1)}+2a_{n-1}\cdot 10^{n-1}(a_{n-2}\cdot 10^{n-2}+\\ +a_{n-3}\cdot 10^{n-3}+\ldots+a_1\cdot 10+a_0)+\ldots+a_3^2\cdot 10^{3\cdot 2}+\\ +2a_3\cdot 10^3(a_2\cdot 10^2+a_1\cdot 10+a_0)+a_2^2\cdot 10^4+\\ +2a_2\cdot 10^2(a_1\cdot 10+a_0)+a_1^2\cdot 10^2\cdot 10^2+2a_1\cdot 10\cdot a_0+a_0^2,\end{gathered}$$

где n — любое целое число.

Раскроем в этом выражении скобки и сгруппируем члены по убывающим степеням:

$$A^2=a_n^2\cdot 10^{2n}+a_{(n-1)}\cdot 10^{n-1}(2a_n\cdot 10^n+a_{n-1}\cdot 10^{n-1})+$$
$$+a_{n-2}\cdot 10^{n-2}(2a_n\cdot 10^n+2a_{n-1}\cdot 10^{n-1}+a_{n-2}\cdot 10^{n-2})+$$
$$+a_{n-3}\cdot 10^{n-3}(2a_n\cdot 10^n+2a_{n-1}\cdot 10^{n-1}+2a_{n-2}\cdot 10^{n-2}+a_{n-3}\cdot 10^{n-3})+\ldots$$
$$\ldots+a_0(2a_n\cdot 10^n+2a_{n-1}\cdot 10^{n-1}+2a_{n-2}\cdot 10^{n-2}+\ldots+2a_1\cdot 10+a_0).$$

Здесь на каждом последующем этапе содержимое в i-й скобке отличается от содержимого скобки на предыдущем этапе тем, что имеет дополнительные слагаемые $a_{n-i+1}\cdot 10^{n-i+1}$ и $a_{n-i}\cdot 10^{n-1}$.

Обозначим скобку на каждом этапе через b_i и примем $b_{n+1}=0$ и $a_{n+1}=0$, тогда

$$A^2=(b_{n+1}+a_{n+1}\cdot 10^{n+1}+a_n\cdot 10^n)\,a_n\cdot 10^n+$$
$$+(b_n+a_n\cdot 10^n+a_{n-1}\cdot 10^{n-1})\,a_{n-1}\cdot 10^{n-1}+$$
$$+(b_{n-1}+a_{n-1}\cdot 10^{n-1}+a_{n-2}\cdot 10^{n-2})\,a_{n-2}\cdot 10^{n-2}+$$
$$+b_{n-2}+a_{n-2}\cdot 10^{n-2}+a_{n-3}\,10^{n-3})\,a_{n-3}\cdot 10^{n-3}+\ldots$$
$$\ldots+(b_2+a_2\cdot 10^2+a_1\cdot 10)\,a_1\cdot 10+(b_1+a_1\cdot 10+a_0)\,a_0.$$

Данное допущение возможно, так как первоначальная запись числа A в виде степенного полинома предполагает, что число A не имеет степеней больше, чем n.

Вынося из скобок b_i общие для скобки множители, получим

$$A^2=(b'_{n+1}\cdot 10+a_{n+1}\cdot 10+a_n)\,a_n\cdot 10^{2n}+$$
$$+(b'_n\cdot 10+a_n\cdot 10+a_{n-1})\,a_{n-1}\cdot 10^{2(n-1)}+$$
$$+(b'_{n-1}\cdot 10+a_{n-1}\cdot 10+a_{n-2})\,a_{n-2}\cdot 10^{2(n-2)}+\ldots+(b'_1\cdot 10+a_1\cdot 10+a_0)\,a_0.$$

Окончательно

$$A^2=\sum_i\{b''_{(i+1)}\cdot 10+a_{(i+1)}\cdot 10+a_i\}\,a_i\cdot 10^{2i}, \tag{8.9}$$

где a_i — цифра числа A в соответствующем разряде i; b'_{i+1} — коэффициент при $a_{i+1}\cdot 10^{2(i+1)}$, т. е. полученный на предыдущем этапе; n — наибольшая степень полинома при разложении числа A.

Алгоритм ручного вычисления квадратного корня по данной формуле можно представить так:

1. Произвести анализ двух старших разрядов числа A^2, найти число a_n, квадрат которого наиболее близко подходит к двум старшим разрядам числа A^2, оставаясь меньше последнего.

2. Произвести вычитание из старших разрядов A^2 квадрата числа a_n.

3. Удвоить число a_n.

4. Сдвинуть остаток от вычитания на два разряда влево, а величину $2a_n$ — на один разряд влево.

5. Приписать справа от остатка вычитания два следующих старших разряда числа A^2.

6. Произвести анализ полученного числа на равенство нулю.

7. Если полученное число не равно нулю, то, анализируя его, найти такое a_{n-1}, которое, будучи умноженным на $(2a_n \cdot 10 + a_{n-1})$, даст в результате число, меньше полученного на шаге 5, но наиболее близкое к нему по значению. Перейти к п. 3.

8. Если при анализе в п. 6 получено равенство, перейти к п. 4, предварительно приписав справа от a_n нуль.

9. После получения количества цифр, равного $n/2$, прекратить вычисление.

При анализе в п. 1 и 7 можно использовать следующее соображение: если последовательно рассматривать квадраты чисел от 0 до 9, то переход от квадрата одного числа можно представить как прибавление к уже известному квадрату определенного числа, которое можно определить из формулы

$$c = a^2 - b^2 = (a - b)(a + b);$$

если $a = b - 1$, то

$$a^2 = b^2 + (a - b)(a + b) = b^2 + (a + b),$$

где a — предыдущее число, возведенное в квадрат.

Данный метод позволяет повысить точность результата. На шестиразрядном сумматоре можно получить в результате пять точных цифр, так как число во время выполнения над ним действий не все располагается на сумматоре.

К недостаткам метода относят довольно длительное время, необходимое для получения результата.

§ 8.8. Перевод чисел в Д-кодах

Рассмотрим некоторые вопросы перевода десятичных чисел, представленных в Д-коде, в двоичную систему счисления.

Пусть задано десятичное число $A = a_4a_3a_2a_1$, где a_i — десятичная цифра, которая должна быть представлена в Д-коде в виде $a_i = \{\alpha_4^i\,\alpha_3^i\,\alpha_2^i\,\alpha_1^i\}$.

Используя равенство $10 = 8 + 2 = 2^3 + 2^1$, любое десятичное целое число можно записать как

$$A_{\text{Д}} = (\ldots(\alpha_4^{''''}\,\alpha_3^{''''}\,\alpha_2^{''''}\,\alpha_1^{''''}\,(2^3 + 2^1) + \alpha_4^{'''}\,\alpha_3^{'''}\,\alpha_2^{'''}\,\alpha_1^{'''})\,(2^3 + 2^1) +$$

$$+ \alpha_4^{''}\,\alpha_3^{''}\,\alpha_2^{''}\alpha_1^{''})\,(2^3 + 2^{+1}) + \alpha_4^{'}\,\alpha_3^{'}\,\alpha_2^{'}\,\alpha_1^{'}.$$

Умножение на 2^k означает сдвиг двоичного кода на k разрядов влево. Следовательно, перевод сводится к сдвигу соответствующих тетрад и их последующему суммированию. Это суммирование может быть выполнено по следующей схеме (для четырехразрядного числа):

α_4''''	α_3''''	α_2''''	α_1''''	α_2'''	α_1''''	α_1'''	α_1''''	α_1'''	α_1''	α_1'''	α_1''	α_1'
		α_4''''	α_3''''	α_3'''	α_2'''	α_2''''	α_2'''	α_2''	α_1''''	α_2''	α_2'	
		α_4''''	α_4'''	α_2''''	α_1''''	α_3'''	α_3''	α_1'''	α_2'''	α_3'		
		α_4''''	α_3''''	α_4''''	α_3''''	α_4''	α_1''''	α_2''''	α_3''			
			α_3''''	α_2''''	α_4'''	α_2''''	α_2'''	α_3'''	α_4'			
				α_4''''	α_1''''	α_3'''	α_1''''	α_4''				
				α_4''''	α_3''''	α_2''''	α_3''''					
					α_4'''	α_4''''	α_4'''					
					α_3''''							
12	11	10	9	8	7	6	5	4	3	2	1	0

Степень двойки

В схеме некоторые символы встречаются многократно. Так как при переводе осуществляется суммирование по столбцам, то пары одинаковых символов дадут единицы переноса в соседние разряды. Все это можно учесть заранее, и схема, приведенная выше, преобразуется в следующую схему:

α_4''''	α_3''''	α_2''''	α_1''''	α_1''''	α_1''''	α_1'''	α_1''''	α_2'''	α_1''	α_1'''	α_1''	α_1'
	α_4''''	α_3''''	α_2''''	α_2''''	α_2''''	α_1'''	α_2''	α_1''	α_1''''	α_2''	α_2'	
		α_4''''	α_3''''	α_3''''	α_2'''	α_2''''	α_3'''	α_4'''	α_2'''	α_3'		
			α_4''''	α_3'''	α_3''''	α_2'''	α_3''	α_4''	α_3''			
			α_4'''	α_4''''	α_3'''	α_4''''	α_4'''		α_4'			
				α_4'''		α_4''						
12	11	10	9	8	7	6	5	4	3	2	1	0

Степень двойки

Таким образом, перевод числа в Д-коде осуществляется путем суммирования элементов тетрад по столбцам с передачей соответствующих переносов.

Подобные способы перевода реализованы в машинах единой системы ЭВМ, машинах фирмы «IBM» и т. д. При разработке схем перевода приходится решать вопросы создания суммирующего устройства на много входов. В самом деле, при переводе, например, восьмиразрядного десятичного числа количество слагаемых в столбце оказывается равным 13. Значит, надо иметь сумматор на 13 входов. Реализовать такую схему можно с помощью многоступенчатых схем. Естественно, при этом возникают дополнительные задержки сигнала, что снижает скорость перевода чисел.

Перевод чисел из двоичной системы счисления в Д-код может осуществляться разными способами. В некоторых случаях для ряда последовательных операций над двоичным изображением числа может быть использована сама вычислительная машина (например, деление на число 1010 целых двоичных чисел; десятичные цифры получаются последовательно одна за другой. При дробных числах эта операция видоизменяется таким образом, чтобы при умножении на число 1010 можно было получить соответствующие цифры десятичных дробей).

Алгоритмы перевода чисел из двоичной системы счисления в Д-код могут быть реализованы схемными или программными способами. Схемные способы перевода десятичных чисел в Д-код или из Д-кода в двоичную систему счисления и обратно весьма перспективны.

Задание для самоконтроля

1. Какие комбиинации являются запрещеиными для кодов $Д_1$, $Д_2$, $Д_5$?

2. Преобразовать число $A = -0{,}6315$ в дополнительный код в кодах $Д_1$ и $Д_2$.

3. Преобразовать число $B = -0{,}1234$ в обратиый код в кодах $Д_2$ и $Д_4$.

4. Сложить числа $A = -0{,}6315$ и $B = 0{,}1234$ иа сумматоре дополнительного кода в коде $Д_1$.

5. Сложить числа $A = 0{,}6315$ и $B = -0{,}1234$ на сумматоре обратного кода в коде $Д_2$.

6. Сложить числа $A = 0{,}1245$ и $B = -0{,}1246$ на сумматоре дополнительного кода в коде $Д_4$.

7. Умножить числа $A = 0{,}12$ и $B = 0{,}13$ иа сумматоре прямого кода в коде $Д_1$.

8. Разделить числа $A = 0{,}1246$ и $B = 0{,}13$ иа сумматоре дополнительного кода в коде $Д_2$.

9. Умножить числа $A = 0{,}146$ и $B = 0{,}178$ ускореиным методом по формуле (8.6) в коде $Д_1$ (сумматор обратного кода).

10. Извлечь квадратный корень из числа $A = 0{,}14412$ на сумматоре обратного кода в коде $Д_1$.

Раздел 2

x_1	x_2	$x_1 \vee x_2$
0	0	0
0	1	1
1	0	1
1	1	1

Логические основы цифровых автоматов

$$x_1 \wedge (x_2 \vee x_3) = (x_1 \wedge x_2) \vee (x_1 \wedge x_3)$$

$$x = x \vee x$$

$$x = x \wedge x$$

9 ГЛАВА

ОСНОВЫ АЛГЕБРЫ ЛОГИКИ

§ 9.1. Основные понятия алгебры логики

Для формального описания цифрового автомата широко применяют аппарат алгебры логики, являющийся одним из важных разделов математической логики.

Примечание. Создателем алгебры логики является английский математик Дж. Буль (1815—1864). Поэтому алгебру логики называют также алгеброй Буля. В последние годы алгебра Буля получила значительное развитие благодаря работам таких ученых, как Э. Пост, К. Шеннон, Г. Шестаков, В. Глушков, С. Яблонский и др.

Основным понятием алгебры логики является высказывание.

Высказывание — *некоторое предложение, о котором можно утверждать, что оно истинно или ложно.*

Например, высказывание «Земля — это планета Солнечной системы» истинно, а о высказывании «на улице идет дождь» можно сказать, истинно оно или ложно, если указаны дополнительные сведения о погоде в данный момент.

Любое высказывание можно обозначить символом x и считать, что $x=1$, если высказывание истинно, а $x=0$ — если высказывание ложно.

Логическая (булева) переменная — *такая величина x, которая может принимать только два значения* (0 или 1):

$$x=\{0,\ 1\}.$$

Высказывание абсолютно истинно, *если соответствующая ей логическая величина принимает значение $x=1$ при любых условиях.*

Примером абсолютно истинного высказывания является высказывание «Земля — это планета Солнечной системы».

Высказывание абсолютно ложно, *если соответствующая ей логическая величина принимает значение $x=0$ при любых условиях.*

Например, высказывание «Земля — спутник Марса» — абсолютно ложное.

Логическая функция (функция алгебры логики) — *функция $f(x_1, x_2, ..., x_n)$, принимающая значение, равное 0 или 1 на наборе логических переменных $x_1, x_2, ..., x_n$.*

Логические функции от одной переменной (табл. 9.1).

Таблица 9.1

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
0	1	0	0	1
1	1	0	1	0

В соответствии с введенными определениями функция $f_1(x)$ является абсолютно истинной (константа единицы), а функция $f_2(x)$ — абсолютно ложной функцией (константа нуля).

Функция $f_3(x)$, повторяющая значения логической переменной,— **тождественная функция** ($f_3(x) \equiv x$), а функция $f_4(x)$, противоположная по своим значениям x,— логическое отрицание, или функция НЕ ($f_4(x) = \neg x = \bar{x}$).

Логические функции от двух переменных (табл. 9.2).

Дизъюнкция (логическое сложение) — *функция* $f_8(x_1, x_2)$, *которая истинна тогда, когда истинны или* x_1, *или* x_2, *или обе переменные.*

Дизъюнкцию часто называют также функцией ИЛИ и условно обозначают так:

$$f_8(x_1, x_2) = x_1 + x_2 = x_1 \vee x_2.$$

От дизъюнкции следует отличать функцию $f_7(x_1, x_2)$, которая называется **функцией сложения по модулю 2 (функцией разноименности)** *и является истинной, когда истинны или* x_1, *или* x_2 *в отдельности.* Условное обозначение этой функции

$$f_7(x_1, x_2) = x_1 \oplus x_2.$$

Конъюнкция (логическое умножение) — *функция* $f_2(x_1, x_2)$, *которая истинна только тогда, когда* x_1 *и* x_2 *истинны.*

Конъюнкцию часто условно называют также функцией И; условно обозначают так:

$$f_2(x_1, x_2) = x_1 x_2 = x_1 \wedge x_2.$$

Пример 9.1. Имеются два высказывания: «Завтра будет холодная погода», «Завтра пойдет снег».

Дизъюнкцией этих высказываний будет новое высказываие: «Завтра будет холодпая погода или пойдет снег».

Соединительным союзом, который образовал новое предложеиие, является ИЛИ.

Конъюнкция образуется следующим образом: «Завтра будет холодная погода и пойдет снег».

Это высказываиие образоваио с помощью союза И.

Штрих Шеффера — *функция* $f_{15}(x_1, x_2)$, *которая ложна только тогда, когда* x_1 *и* x_2 *истинны.*

Функция	x_1x_2				Примечание
	00	01	10	11	
f_1	0	0	0	0	f_0
f_2	0	0	0	1	$x_1 \wedge x_2$ (конъюнкция)
f_3	0	0	1	0	$x_1 \wedge \overline{x_2}$ (запрет x_2)
f_4	0	0	1	1	$x_1\overline{x_2} \vee x_1x_2 = x_1$
f_5	0	1	0	0	$\overline{x_1}x_2$ (запрет x_1)
f_6	0	1	0	1	$\overline{x_1}x_2 \vee x_1x_2 = x_2$
f_7	0	1	1	0	$x_1 \oplus x_2$ (сложеиие по модулю 2)
f_8	0	1	1	1	$\underline{x_1 \vee x_2}$ (дизъюнкция)
f_9	1	0	0	0	$x_1 \wedge x_2 = x_1 \downarrow x_2$ (функция Пирса)
f_{10}	1	0	0	1	$x_1 \equiv x_2$ (равнозиачиость)
f_{11}	1	0	1	0	$\overline{x_1}\overline{x_2} \vee x_1\overline{x_2} = \overline{x_2}$
f_{12}	1	0	1	1	$x_2 \longrightarrow x_1$ (импликация)
f_{13}	1	1	0	0	$\overline{x_1}\overline{x_2} \vee \overline{x_1}x_2 = \overline{x_1}$
f_{14}	1	1	0	1	$x_1 \longrightarrow x_2$ (импликация)
f_{15}	1	1	1	0	x_1/x_2 (фуикция Шеффера)
f_{16}	1	1	1	1	f_1

Условное обозначение функции Шеффера:

$$f_{15}(x_1,\ x_2) = x_1/x_2.$$

П р и м е ч а и и е. Немецкий математик Шеффер иа основе этой функции создал алгебру, иазваиную алгеброй Шеффера.

Функция Пирса (Вебба) — *функция $f_9(x_1,\ x_2)$, которая истинна только тогда, когда x_1 и x_2 ложны.*

Условное обозначение этой функции:

$$f_9(x_1,\ x_2) = x_1 \downarrow x_2 = x_1 \bigcirc x_2.$$

П р и м е ч а н и е. Математики Пирс и Вебб, независимо друг от друга изучавшие свойства этой фуикции, создали алгебру, иазваиную алгеброй Пирса (Вебба).

Импликация — *функция $f_{14}(x_1,\ x_2)$, которая ложна тогда и только тогда, когда x_1 истинно и x_2 ложно.* Условное обозначение $f_{14}(\bar{x}_1,\ \bar{x}_2) = x_1 \rightarrow x_2$.

Все рассмотренные выше логические функции являются элементарными.

Две функции равносильны друг другу, *если принимают на всех возможных наборах переменных одни и те же значения* $f_1(x_1, x_2, \ldots, x_n) = f_2(x_1, x_2, \ldots, x_n)$.

Булевы переменные могут быть действительными или фиктивными.

Переменная x_i действительна, *если значение функции* $f(x_1, x_2, \ldots, x_i, \ldots, x_n)$ *существенно изменяется при изменении* x_i.

Переменная x_i фиктивна, *если значение функции* $f(x_1, \ldots, x_i, \ldots, x_n)$ *не изменяется при изменении* x_i.

Логическая функция от трех переменных (табл. 9.3).

Таблица 9.3

x_1	x_2	x_3	$f(x_1, x_2, x_3)$	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0	1	0	0	1
0	0	1	0	1	0	1	1
0	1	0	1	1	1	0	0
0	1	1	1	1	1	1	0

Из таблицы видно, что переменные x_1 и x_2 — действительные, а переменная x_3 — фиктивная, так как $f(x_1, x_2, 0) = f(x_1, x_2, 1)$ для всех наборов x_1, x_2.

Таким образом, появляется возможность сокращать или расширять количество переменных для логических функций удалением или введением фиктивных переменных.

Так как число значений переменных x_i ограничено, то можно определить количество N функций от любого числа переменных: $N = 2^{2^n}$, где n — количество переменных x_i.

Рассмотрим некоторые практические примеры использования алгебры логики.

Пример 9.2. В школе произошла неприятная история: разбито окно в одном из классов. Подозревают четырех учеников: Леню, Диму, Толю и Мишу [12].

При опросе каждый из детей сделал по три заявления:

Леня:

1) я не виноват — $Л_1$;
2) я не подходил к окну — $Л_2$;
3) Миша знает, кто разбил, — $Л_3$.

Дима:

1) стекло разбил не я — $Д_1$;
2) с Мишей я не был знаком до поступления в школу — $Д_2$;
3) это сделал Толя — $Д_3$.

Толя:

1) я не виноват — T_1;
2) это сделал Миша — T_2;
3) Дима говорит неправду, утверждая, что я разбил окно, — T_3.

Миша:

1) я не виноват — M_1;
2) стекло разбил Леня — M_2;

3) Дима может поручиться за меня, так как знает меня со дня рождения, — M_3.

В дальнейшем все признали, что одно из трех заявлений является неверным. Это пригодится при построении более сложных формул, поскольку показания каждого ученика в целом истинны только при условии, что два заявления истинны, а одно ложно. Используя элементарные логические функции, можно описать показания всех учеников в таком виде:

$$\left.\begin{aligned}
&\text{Л} = \text{Л}_1\text{Л}_2\overline{\text{Л}}_3 + \text{Л}_1\overline{\text{Л}}_2\text{Л}_3 + \overline{\text{Л}}_1\text{Л}_2\text{Л}_3;\\
&\text{Д} = \text{Д}_1\text{Д}_2\overline{\text{Д}}_3 + \text{Д}_1\overline{\text{Д}}_2\text{Д}_3 + \overline{\text{Д}}_1\text{Д}_2\text{Д}_3;\\
&T = T_1T_2\overline{T}_3 + T_1\overline{T}_2\overline{T}_3 + \overline{T}_1T_2T_3;\\
&M = M_1M_2\overline{M}_3 + M_1\overline{M}_2M_3 + \overline{M}_1M_2M_3.
\end{aligned}\right\}$$

Теперь остается решить эту систему уравнений и определить, какие показания истинны. Для этого надо упростить выражения, используя аксиомы. Рассмотрим третье уравнение. По условию, $T_1=T_3$, а значит, $\overline{T}_1=\overline{T}_3$, но $T_1\overline{T}_1=0$, $T_1T_1=T_1$ или $T=T_1\overline{T}_2$.

Поэтому оно верно тогда, когда $T_1=1$; $T_2=0$.

Значит, **Толя не виноват и Миша не виноват.**

Отсюда следует, что Д_3 ложно, т. е. $\text{Д}_3=0$ ($\overline{\text{Д}}_3=1$).

Следовательно,

$$\text{Д} = \text{Д}_1\text{Д}_2\overline{\text{Д}}_3.$$

Отсюда $\text{Д}_1=1$; $\text{Д}_2=1$.

Дима не виноват.

Д_2 противоположно M_3, т. е.

$$\overline{\text{Д}}_2 = M_3.$$

Значит, $M_3=0$, или $M=M_1M_2\overline{M}_3$.

Оно верно только тогда, когда $M_1=1$; $M_2=1$.

Следовательно, **стекло разбил Леня.**

Пример 9.3. Предположим, что имеется система кондиционирования воздуха для помещения, где установлена ЭВМ, состоящая из двух кондиционеров малой и большой мощности и работающая при таких условиях:

1) кондиционер малой мощности включается, если температура воздуха в помещении **достигает 19° С;**

2) кондиционер большой мощности включается, если температура воздуха достигает 22° С (малый кондиционер при этом отключается);

3) оба кондиционера включаются при температуре воздуха 30° С.

Пусть информация о температуре воздуха поступает от датчиков, которые соответственно срабатывают при достижении температуры 19, 22, 30° С. Каждый из этих датчиков выдает входную информацию для устройства управления кондиционерами. Первые три датчика определяют рабочие режимы, и их можно представить как входы управляющего автомата. Используя двоичный алфавит для задания состояний датчика (0 — нет сигнала о достижении заданного уровня температуры, 1 — есть сигнал), функционирование системы управления кондиционерами можно описать следующим образом:

z_3	z_2	z_1	ω_2	ω_1
0	0	0	0	0
0	0	1	0	1
0	1	1	1	0
1	1	1	1	1

Здесь z_1 — датчик, срабатывающий при $t=19°$ С; $z_1=0$, если температура меньше 19° С; $z_1=1$, если температура равна или больше 19° С; z_2 — датчик, срабатывающий при $t=22°$ С, $z_2=0$, если $t<22°$ С, $z_2=1$ при $t\geqslant 22°$ С; z_3 — датчик, срабатывающий при $t=30°$ С; $z_3=0$ при $t<30°$ С, $z_3=1$ при $t\geqslant 30°$ С; ω_1 и ω_2 — соответственно управление маломощным и мощным кондиционерами ($\omega=0$ — кондиционер выключен, $\omega=1$ — кондиционер включен).

Таблица описывает функционирование системы управления без нарушений работы.

Впервые теория Дж. Буля была применена П. С. Эренфестом к анализу контактных цепей (1910). Возможность описания переключательных схем с помощью логических формул оказалась весьма ценной по двум причинам. Во-первых, с помощью формул удобнее проверять работу схем. Во-вторых, задание условий работы любой переключательной схемы в виде формул упрощает процесс построения самой переключательной схемы, так как оказалось, что существует ряд эквивалентных преобразований, в результате которых логические формулы упрощаются. При описании переключательных схем замкнутый контакт принимается за истинное высказывание, а разомкнутый — за ложное, поэтому последовательное соединение контактов можно рассматривать как функцию И, а параллельное — как функцию ИЛИ.

Использование логических функций оказалось особенно полезным для описания работы логических элементов ЭВМ.

§ 9.2. Свойства элементарных функций алгебры логики

Из табл. 9.2 видно, что элементарные функции типа отрицания, дизъюнкции, конъюнкции, Шеффера, Пирса, импликации и т. д. находятся в определенной связи друг с другом. Рассмотрим эти связи и свойства исходных функций.

Конъюнкция, дизъюнкция, отрицание (функции И, ИЛИ, НЕ). Используя основные положения алгебры логики, нетрудно убедиться в справедливости следующих аксиом. Пусть x — некоторая логическая переменная. Тогда

1) $x=\overline{\overline{x}}$, что означает возможность исключения из логического выражения всех членов, имеющих двойное отрицание, заменив их исходной величиной;

2) $\left.\begin{array}{l} x+x=x \\ xx=x \end{array}\right\}$ — правила подобных преобразований, которые позволяют сокращать длину логических выражений:

3) $x+0=x$; 6) $x1=x$;

4) $x+1=1$; 7) $x\overline{x}=0$;

5) $x0=0$; 8) $x+\overline{x}=1$ (логическая истина).

Дизъюнкция и конъюнкция обладают рядом свойств, аналогичных свойствам обычных арифметических операций сложения и умножения:

1) свойство ассоциативности (сочетательный закон):

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

2) свойство коммутативности (переместительный закон):

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

3) свойство дистрибутивности (распределительный закон): для конъюнкции относительно дизъюнкции

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

для дизъюнкции относительно конъюнкции

$$x_1+x_2x_3=(x_1+x_2)(x_1+x_3).$$

Свойство дистрибутивности фактически определяет правила раскрытия скобок или взятия в скобки логических выражений.

Справедливость указанных свойств легко доказывается с помощью вышеизложенных аксиом.

Докажем, например, что

$$x_1+x_2x_3=(x_1+x_2)(x_1+x_3).$$

В самом деле,

$$(x_1+x_2)\wedge(x_1+x_3)=x_1x_1+x_1x_3+x_1x_2+x_2x_3=$$
$$=x_1+x_1x_3+x_1x_2+x_2x_3=x_1(1+x_2+x_3)+x_2x_3=x_1+x_2x_3.$$

Аналогичным образом можно доказать и другие законы.

Несложно установить правильность соотношений, известных как **законы де Моргана:**

$$\left.\begin{aligned}\overline{x_1x_2}&=\bar{x}_1+\bar{x}_2;\\ \overline{x_1+x_2}&=\bar{x}_1\bar{x}_2.\end{aligned}\right\} \tag{9.1}$$

Из законов де Моргана вытекают следствия

$$\left.\begin{aligned}x_1x_2&=\overline{\bar{x}_1+\bar{x}_2};\\ x_1+x_2&=\overline{\bar{x}_1\bar{x}_2},\end{aligned}\right\} \tag{9.2}$$

с помощью которых появляется возможность выражать конъюнкцию через дизъюнкцию и отрицание или дизъюнкцию через конъюнкцию и отрицание.

Законы де Моргана и следствия из них справедливы для любого числа переменных:

$$\left.\begin{aligned}\overline{x_1+x_2+\ldots+x_n}&=\bar{x}_1\bar{x}_2\ldots\bar{x}_n;\\ \overline{x_1x_2\ldots x_n}&=\bar{x}_1+\bar{x}_2+\ldots+\bar{x}_n.\end{aligned}\right\} \tag{9.3}$$

Для логических функций устанавливаются соотношения, известные как **законы поглощения:**

$$\left.\begin{aligned}x_1+(x_1x_2)&=x_1;\\ x_1(x_1+x_2)&=x_1.\end{aligned}\right\} \tag{9.4}$$

В табл. 9.4 показана справедливость законов поглощения.

Таблица 9.4

x_1	x_2	x_1+x_2	x_1x_2	$x_1+(x_1x_2)$	$x_1\,(x_1+x_2)$
0	0	0	0	0	0
0	1	1	0	0	0
1	0	1	0	1	1
1	1	1	1	1	1

Функция сложения по модулю 2 — *функция, выражаемая следующим образом:*

$$x_1 \oplus x_2 = x_1\bar{x}_2 + \bar{x}_1x_2 = (x_1 + x_2)\,(\bar{x}_1 + \bar{x}_2). \tag{9.5}$$

Функция сложения по модулю 2 обладает следующими свойствами:

коммутативности (переместительный закон)

$$x_1 \oplus x_2 = x_2 \oplus x_1;$$

ассоциативности (сочетательный закон)

$$x_1 \oplus (x_2 \oplus x_3) = (x_1 \oplus x_2) \oplus x_3;$$

дистрибутивности (распределительный закон)

$$x_1\,(x_2 \oplus x_3) = (x_1x_2) \oplus (x_1x_3).$$

Для этой функции справедливы аксиомы:

$$x \oplus x = 0; \quad x \oplus 1 = \bar{x};$$

$$x \oplus \bar{x} = 1; \quad x \oplus 0 = x.$$

На основании аксиом и свойств можно вывести правила перевода функций И, ИЛИ, НЕ через функцию сложения по модулю 2 и наоборот:

$$\left.\begin{aligned} &\bar{x}_1 = x_1 \oplus 1; \\ &x_1 + x_2 = x_1 \oplus x_2 \oplus x_1x_2; \\ &x_1\,x_2 = (x_1 \oplus x_2) \oplus (x_1 + x_2). \end{aligned}\right\} \tag{9.6}$$

Функция импликации (→) — *функция, выражаемая следующим образом:*

$$x_1 \to x_2 = \bar{x}_1 + x_2.$$

Для функции импликации справедливы аксиомы:

$$x \to x = 1; \quad x \to \bar{x} = \bar{x};$$

$$x \to 1 = 1; \quad 1 \to \bar{x} = x;$$

$$x \to 0 = \bar{x}; \quad 0 \to x = 1.$$

Из аксиом следует, что импликация обладает только свойством коммутативности (переместительный закон) в измененном виде:

$$x_1 \to x_2 = \overline{x}_2 \to \overline{x}_1.$$

Свойство ассоциативности для этой функции не справедливо (табл. 9.5).

Таблица 9.5

x_1	x_2	x_3	$x_1\to(x_2\to x_3)$	$(x_1\to x_2)\to x_3$
0	0	0	1	0
0	1	0	1	0
1	0	0	1	1
1	1	0	0	0
0	0	1	1	1
0	1	1	1	1
1	0	1	1	1
1	1	1	1	1

Функции И, ИЛИ, НЕ через импликацию выражаются так:

$$x_1 + x_2 = \overline{x}_1 \to x_2;$$

$$x_1x_2 = \overline{\overline{x_1\,x_2}} = \overline{x_1 \to \overline{x}_2}; \tag{9.7}$$

$$\overline{x}_1 = x_1 \to 0.$$

Функция Шеффера (/) — *функция, которая может быть выражена следующим соотношением*

$$x_1/x_2 = \overline{x_1\,x_2}.$$

Для нее характерны аксиомы:

$$x/x = \overline{x};\quad x/1 = \overline{x};$$

$$x/\overline{x} = 1;\quad \overline{x}/0 = 1;$$

$$x/0 = 1;\quad \overline{x}/1 = x.$$

Для функции Шеффера справедливо только свойство коммутативности

$$x_1/x_2 = x_2/x_1,$$

но

$$x_1/(x_2/x_3) \neq (x_1/x_2)/x_3.$$

Из основных свойств можно получить формулы преобразования:

$$\left.\begin{aligned}&x_1x_2=\overline{x_1/x_2}=x_1/x_2/x_1/x_2;\\&\overline{x}=x/x;\\&x_1+x_2=\overline{\overline{x_1+x_2}}=\overline{\overline{x}_1\overline{x}_2}=\overline{x}_1/\overline{x}_2=x_1/x_1/x_2/x_2.\end{aligned}\right\}\qquad(9.8)$$

Функция Пирса (Вебба) ($\uparrow$) — *функция, которая описывается выражением*

$$x_1\uparrow x_2=\overline{x_1+x_2}=\overline{x}_1\wedge\overline{x}_2.$$

Для этой функции легко доказываются аксиомы:

$$x\uparrow x=\overline{x};\quad x\uparrow 0=\overline{x};$$

$$x\uparrow\overline{x}=0;\quad x\uparrow 1=0.$$

На основании аксиом можно показать, что для функции Пирса (Вебба) справедливо только свойство коммутативности

$$x_1\uparrow x_2=x_2\uparrow x_1.$$

Функции И, ИЛИ, НЕ выражаются через функцию Пирса (Вебба) следующим образом:

$$\left.\begin{aligned}&x_1x_2=(x_1\uparrow x_1)\uparrow(x_2\uparrow x_2);\\&x_1+x_2=(x_1\uparrow x_2)\uparrow(x_1\uparrow x_2);\\&\overline{x}=x\uparrow x.\end{aligned}\right\}\qquad(9.9)$$

§ 9.3. Аналитическое представление функций алгебры логики

Существует много способов задания логических функций. Ранее был рассмотрен табличный способ, при котором каждому набору значений переменных в таблице истинности указывается значение самой логической функции. Этот способ показателен и может быть применен для записи функций от любого количества переменных. Однако при анализе свойств ФАЛ такая запись не является компактной. Проще выглядит аналитическая запись в виде формул.

Рассмотрим фиксированный набор переменных $\{x_1, x_2, ..., x_n\}$, на котором задана функция алгебры логики. Так как любая переменная $x_i=\{0, 1\}$, то набор значений переменных фактически представляет собой некоторое двоичное число. Представим, что номером набора будет произвольное двоичное число i, получаемое следующим образом:

$$i=x_1\cdot 2^{n-1}+x_2\cdot 2^{n-2}+\ldots+x_{n-1}\cdot 2^1+x_n.\qquad(9.10)$$

Пусть имеется функция $\Phi_i(x_1, x_2, ..., x_n)$, равная

$$\Phi_i=\begin{cases}0, & \text{если номер набора равен } i,\\ 1, & \text{если номер набора не равен } i.\end{cases}$$

Функцию Φ_i называют **термом.**

Дизъюнктивный терм (макстерм) — *терм, связывающий все переменные, представленные в прямой или инверсной форме, знаком дизъюнкции* (иногда в литературе используется термин «конституэнта нуля»).

Например,

$$\Phi_1 = \bar{x}_1 \vee x_2 \vee x_3 \vee x_4;$$

$$\Phi_2 = x_1 \vee \bar{x}_2.$$

Конъюнктивный терм (минтерм) — *терм, связывающий переменные, представленные в прямой или инверсной форме, знаком конъюнкции* (иногда в литературе используется термин «конституэнта единицы»).

Например,

$$F_1 = x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4;$$

$$F_2 = \bar{x}_1 \bar{x}_3 \bar{x}_4.$$

Ранг терма r *определяется количеством переменных, входящих в данный терм.*

Например, для минтерма

$$F_1 = \bar{x}_1 x_2 \bar{x}_3 x_4 \bar{x}_5, \quad r = 5$$

и для макстерма

$$\Phi_1 = \bar{x}_1 + x_2 + \bar{x}_3, \quad r = 3.$$

На основании вышесказанного можно сформулировать следующую теорему.

Теорема. *Любая таблично-заданная ФАЛ может быть представлена аналитическим способом в виде*

$$f(x_1, x_2, \ldots, x_n) = F_1 \vee F_2 \vee \ldots \vee F_n = \bigvee_1 F_i, \qquad (9.11)$$

где i — номера наборов, на которых функция равна 1; $\bigvee_1$ — знак дизъюнкции, объединяющий все термы F_i, равные 1.

В самом деле, если на каком-либо наборе функция $f(x_1{}^*, x_2{}^*, \ldots, x_n{}^*) = 1$, то, вследствие того что $x \vee 1 = 1$, в правой части выражения (9.11) всегда найдется элемент, равный 1; если же на наборе i функция $f(x_1{}^*, x_2{}^*, \ldots, x_n{}^*) = 0$, то в правой части не найдется ни одного элемента, равного 1, так как $0 \vee 0 \vee \ldots \vee 0 = 0$.

Таким образом, каждому набору i, для которого $f_i = 1$, соответствует элемент $F_i = 1$, а наборам i, на которых $f_i = 0$, не соответствует ни одного элемента $F_i = 1$. Поэтому таблица истинности однозначно отображается аналитической записью вида (9.11), которую в дальнейшем будем называть **объединением термов.**

Нормальная дизъюнктивная форма (НДФ) — *объединение термов, включающее минтермы переменного ранга.*

Количество всех термов, входящих в состав (9.11), равно количеству единичных строк таблицы.

Пример 9.4. Записать в аналитическом виде функцию, заданную табл. 9.6.

Таблица 9.6

x_1	x_2	x_3	$f(x_1, x_2, x_3)$	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1	1	0	0	1
0	0	1	0	1	0	1	0
0	1	0	0	1	1	0	0
0	1	1	1	1	1	1	0

Решение. На основании теоремы эту функцию можно записать в аналитической форме:

$$f(x_1, x_2, x_3) = F(0, 0, 0) + F(0, 1, 1) + F(1, 0, 0) =$$

$$= \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1 x_2 x_3 + x_1\overline{x}_2\overline{x}_3.$$

Ответ: $f(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1 x_2 x_3 + x_1\overline{x}_2\overline{x}_3$.

Для представления ФАЛ в (9.11) используется совокупность термов, объединенных знаками дизъюнкции ($\vee$ или $+$). Можно использовать также другую элементарную логическую операцию. Сформулируем основные требования к этой операции.

Требование 1: *если какой-либо терм $F_i = 1$, то функция f должна быть равна* 1.

Требование 2: *если какой-либо терм $F_i = 0$, то функция f может быть равна* 1.

Необходимо, чтобы при всех термах $F_i = 0$ функция f была равна 0.

Табличное представление искомой логической операции Δ имеет вид табл. 9.7 и 9.8.

Таблица 9.7

F_l	F_j	$\Delta = \vee$
0	0	0
0	1	1
1	0	1
1	1	1

Таблица 9.8

F_l	F_j	$\Delta = \oplus$
0	0	0
0	1	1
1	0	1
1	1	0

Таким образом, получили, что искомой функцией кроме функции ИЛИ может быть функция разноименности и при этом справедливым становится такое **следствие из теоремы (9.11)**:

Любая таблично заданная ФАЛ может быть представлена в следующей аналитической форме:

$$f(x_1, x_2, \ldots, x_n) = F_1 \Delta F_2 \Delta \ldots \Delta F_k, \tag{9.12}$$

где знак Δ обозначает операции $\vee$, $\oplus$.

Требования 1 и 2 можно обобщить и потребовать, чтобы аналитическое представление нулевых и единичных строчек таблицы различалось и чтобы выполнялось взаимно-однозначное соответствие между нулевой (единичной) строкой и термом.

Требование 3: *если какой-либо терм $\Phi_i = 0$, то и функция f должна быть равна* 0.

Требование 4: *если все термы $\Phi_i = 1$, то функция $f = 1$.*

Выполняя эти требования, можно прийти к двум другим возможным функциям: конъюнкции и равнозначности (табл. 9.9 и 9.10).

Таблица 9.9

Φ_i	Φ_j	$\Delta = \wedge$
0	0	0
0	1	0
1	0	0
1	1	1

Таблица 9.10

Φ_i	Φ_j	$\Delta = \equiv$
0	0	1
0	1	0
1	0	0
1	1	1

Теорема: *любая таблично заданная ФАЛ может быть задана в аналитической форме:*

$$f(x_1, x_2, \ldots, x_n) = \Phi_1 \wedge \Phi_2 \wedge \ldots \wedge \Phi_k, \tag{9.13}$$

где k — количество двоичных наборов, для которых $f = 0$.

Нормальная конъюнктивная форма (НКФ) — *объединение термов (9.13), включающее в себя макстермы разных рангов.*

Из **теоремы (9.13)** вытекает такое **следствие:**

любая таблично заданная ФАЛ может быть представлена в аналитической форме:

$$f(x_1, x_2, \ldots, x_n) = \Phi_1 \equiv \Phi_2 \equiv \ldots \equiv \Phi_k, \tag{9.14}$$

где k — количество нулевых значений функции.

§ 9.4. Совершенные нормальные формы (СНФ)

Нормальные (конъюнктивная, дизъюнктивная) формы не дают однозначного представления функции. Такое представление получается только при совершенных нормальных формах.

Введем обозначения

$$x^0 = \bar{x}; \quad x^1 = x.$$

Тогда в общем виде переменная может быть задана как некоторая функция

$$x^{\alpha}=\begin{cases} x, & \text{если} \quad \alpha=1; \\ \bar{x}, & \text{если} \quad \alpha=0, \end{cases} \tag{9.15}$$

при этом

$$x^{\alpha}=\alpha x+\bar{\alpha}\bar{x}, \tag{9.16}$$

где α — двоичная переменная.

Рассмотрим конъюнкцию вида $x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$, где $\{\alpha_1\alpha_2 \ldots \alpha_n\}$ представляет собой двоичный набор; число наборов равно 2^n, т. е.

$$\begin{matrix} 0\;0 \ldots 0\;0\;0 \\ 0\;0 \ldots 0\;0\;1 \\ 0\;0 \ldots 0\;1\;0 \\ 0\;0 \ldots 0\;1\;1 \end{matrix}$$

и т. д.

Если задать всем α_i значение 0 и 1, то можно получить, например, дизъюнкцию вида

$$\bigvee_1 x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}=x_1^0 x_2^0 \ldots x_n^0 \vee x_1^0 x_2^0 \ldots x_n^1 \vee x_1^0 x_2^0 \ldots$$

$$\ldots x_{n-1}' x_n^0 \vee x_1^0 x_2^0 \ldots x_{n-1}^1 x_n^1 \vee \ldots \vee x_1^1 x_2^1 \ldots x_n^1, \tag{9.17}$$

где $\bigvee_1$ — символ обобщенной дизъюнкции по единичным строкам.

Тогда справедлива следующая теорема.

Теорема: *любая ФАЛ может быть представлена в виде*

$$f(x_1x_2 \ldots x_kx_{k+1} \ldots x_n)=\bigvee_1 x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_k^{\alpha_k} f(\alpha, \ldots \alpha_k x_{k+1} \ldots x_n). \tag{9.18}$$

Выражение (9.18) называют **разложением функции алгебры логики по k переменным.**

Докажем теорему: прежде всего следует определить, при каких условиях выполняется равенство $x^{\alpha}=1$.

Очевидно, что это имеет место при $x=\alpha$. В самом деле, если $x=\alpha$, то $\alpha^{\alpha}=\alpha\alpha+\bar{\alpha}\bar{\alpha}=\alpha+\bar{\alpha}=1$; если $x=\bar{\alpha}$, то $(\bar{\alpha})^{\alpha}=\bar{\alpha}\alpha+\bar{\bar{\alpha}}\bar{\alpha}=\alpha\bar{\alpha}=0$.

С учетом того, что $x^{\alpha}=1$ при $x=\alpha$, можно утверждать, что конъюнкция вида $x_1^{\alpha_1} x_2^{\alpha_2} x_3^{\alpha_3} \ldots x_k^{\alpha_k}$ равна 1 при $x_1=\alpha_1$, $x_2=\alpha_2, \ldots, x_k=\alpha_k$.

Эти равенства можно подставить в правую часть выражения (9.18). В результате

$$\bigvee_1 \underbrace{x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_k^{\alpha_k}}_{1} f(x_1x_2 \ldots x_kx_{k+1} \ldots x_n)=f(x_1x_2 \ldots x_kx_{k+1} \ldots x_n),$$

что и требовалось доказать.

Из **теоремы (9.18)** можно вывести два основных **следствия:**

1) *если $k=1$, то функция алгебры логики представляется в виде*

$$f(x_1, x_2, \ldots, x_n)=x_1 f(1, x_2, \ldots, x_n) \vee \bar{x}_1 f(0, x_2, \ldots, x_n); \tag{9.19}$$

2) *если $k=n$, то любая ФАЛ может быть представлена в виде*

$$f(x_1, x_2, \ldots, x_n)= \bigvee_{1} x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}. \tag{9.20}$$

Совершенная нормальная дизъюнктивная форма (СНДФ) — *ФАЛ, заданная в виде* (9.20).

Учитывая вышеизложенное, рассмотрим основные свойства СНДФ:

в СНДФ нет двух одинаковых минтермов;

в СНДФ ни один минтерм не содержит двух одинаковых множителей (переменных);

в СНДФ ни один минтерм не содержит вместе с переменной и ее отрицание.

На основании этих свойств можно предложить следующий алгоритм получения СНДФ из таблицы истинности:

1. Положить номер строки в таблице $i=1$, номер элемента в строке $j=1$.
2. Выбрать из таблицы набор с номером i. Если $f=1$, то перейти к п. 3, иначе к п. 5.
3. Сформировать терм F_i. Выбрать элемент строки с номером j. Если

$$x_j=\begin{cases} 0, & \text{то} \quad F_i:=F_i \wedge \bar{x}_j, \\ 1, & \text{то} \quad F_i:=F_i \wedge x_j. \end{cases}$$

4. Вычислить $j:=j+1$. Если $i<n$, то перейти к п. 3, иначе к п. 5.
5. Вычислить $i:=i+1$. Если $i<2^n$, то перейти к п. 2, иначе к п. 6.
6. Конец.

Пример 9.5. Представить функцию, заданную табл. 9.11 в СНДФ.

Таблица 9.11

x_1	x_2	x_3	f	x_1	x_2	x_3	f
0	0	0	0	1	0	0	1
0	0	1	0	1	0	1	0
0	1	0	0	1	1	0	1
0	1	1	1	1	1	1	0

Решение. В соответствии с алгоритмом

$$f(x_1, x_2, x_3)= \bar{x}_1 x_2 x_3 \vee x_1 \bar{x}_2 \bar{x}_3 \vee x_1 x_2 \bar{x}_3.$$

Рассмотрим СНФ для других функций.

Путь имеется терм вида

$$\Phi_i = x_1^{\overline{\alpha}_1} \vee x_2^{\overline{\alpha}_2} \vee \ldots \vee x_n^{\overline{\alpha}_n},$$

где

$$\Phi_i = \begin{cases} 0, \text{ если номер набора равен } i; \\ 1, \text{ если номер набора не равен } i. \end{cases}$$

Если $x_i = \alpha_i$ и α_i — текущий элемент двоичного набора, то тогда возможны случаи:

1) $x_i = 0$	2) $x_i = 0$	3) $x_i = 1$	4) $x_i = 1$
$\alpha_i = 0$	$\alpha_i = 1$	$\alpha_i = 0$	$\alpha_i = 1$
$\overline{\alpha}_i = 1$	$\overline{\alpha}_i = 0$	$\overline{\alpha}_i = 1$	$\overline{\alpha}_i = 0$
$x_i^{\overline{\alpha}_i} = x_i = 0$	$x_i^{\overline{\alpha}_i} = \overline{x}_i = 1$	$x_i^{\overline{\alpha}_i} = x_i = 1$	$x_i^{\overline{\alpha}_i} = \overline{x}_1 = 0$

$x_i^{\overline{\alpha}_1} = 0$ для $x_i = \alpha_i$ и тогда терм Φ_i можно использовать в представлении (9.13) и (9.14).

Теорема: *любая ФАЛ, кроме абсолютно истинной функции, может быть представлена в совершенной конъюнктивной нормальной форме* (СКНФ):

$$f(x_1, x_2, \ldots, x_n) = \bigwedge_{0} (x_1^{\overline{\alpha}_1} \vee x_2^{\overline{\alpha}_2} \vee \ldots \vee x_n^{\overline{\alpha}_n}). \qquad (9.21)$$

Для представления логической функции используются операции И, ИЛИ, НЕ ($\wedge$, $\vee$, $\urcorner$).

Следствием из теоремы (9.21) является:

любая ФАЛ, кроме конституэнты 1, *может быть представлена в виде*

$$f(x_1, x_2, \ldots, x_n) = \underset{0}{\equiv} (x_1^{\overline{\alpha}_1} \vee x_2^{\overline{\alpha}_2} \vee \ldots \vee x_n^{\overline{\alpha}_n}). \qquad (9.22)$$

Здесь используются операции неравнозначности, дизъюнкции, отрицания ($\equiv$, $\vee$, $\urcorner$).

Пример 9.6. Найти СКНФ для функции (см. табл. 9.11).

Решение.

$$f_{\text{СНКФ}}(x_1, x_2, x_3) = (x_1 \vee x_2 \vee x_3)(x_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee x_2 \vee \overline{x}_3) \times$$

$$\times (\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3).$$

Это представление более громоздкое, так как в таблице много строк, для которых $f=0$.

В рассмотренных ранее объединениях термов были использованы для записи термов F_{ij} и Φ_{ij} операции ИЛИ, НЕ, а также И, НЕ. Можно использовать также набор из импликации ($\rightarrow$) и отрицания (НЕ).

С п о с о б ы   п р е о б р а з о в а н и я   НФ в СНФ. Совершенная нормальная форма отличается от нормальной формы (НФ) тем, что всегда содержит термы только максимального ранга и дает однозначное представление функции.

Произвольная нормальная дизъюнктивная форма (НДФ) переводится в СНДФ следующим образом.

Пусть $f_{\text{НФ}} = F_1$. Тогда

$$f_{\text{СНФ}} = F_1 x_i \vee F_1 \bar{x}_i, \tag{9.23}$$

где x_i — переменная, которая не входит в данный терм F_1.

Пример 9.7. Логическую функцию, заданную в НДФ:

$$f(x_1, x_2, x_3, x_4) = \underset{F_1}{x_1\bar{x}_2} \vee \underset{F_2}{x_2\bar{x}_3\bar{x}_4} \vee \underset{F_3}{\bar{x}_1\bar{x}_3 x_4} \vee \underset{F_4}{x_1 x_2 x_3 x_4},$$

преобразовать в СНДФ.

Р е ш е н и е. Воспользуемся приемом преобразования (9.23) поочередно к термам:

$$F_1 = x_1\bar{x}_2(x_3 \vee \bar{x}_3) = x_1\bar{x}_2 x_3 \vee x_1\bar{x}_2\bar{x}_3.$$

Оба члена полученного выражения умножаем на $(x_4 \vee \bar{x}_4)$. В результате получаем

$$F_1 = (x_1\bar{x}_2 x_3 \vee x_1\bar{x}_2\bar{x}_3)(x_4 \vee \bar{x}_4) = x_1\bar{x}_2 x_3 x_4 \vee$$

$$\vee x_1\bar{x}_2 x_3\bar{x}_4 \vee x_1\bar{x}_2\bar{x}_3 x_4 \vee x_1\bar{x}_2\bar{x}_3\bar{\bar{x}}_4.$$

Аналогично,

$$F_2 = x_2\bar{x}_3 x_4(x_1 \vee \bar{x}_1) = x_1 x_2\bar{x}_3 x_4 \vee \bar{x}_1 x_2\bar{x}_3 x_4;$$

$$F_3 = \bar{x}_1\bar{x}_3 x_4(x_2 \vee \bar{x}_2) = \bar{x}_1 x_2\bar{x}_3 x_4 \vee \bar{x}_1\bar{x}_2\bar{x}_3 x_4.$$

После приведения подобных членов определяем СНДФ:

$$f_{\text{СНДФ}}(x_1, x_2, x_3, x_4) = x_1\bar{x}_2 x_3 x_4 \vee x_1\bar{x}_2 x_3\bar{x}_4 \vee x_1\bar{x}_2\bar{x}_3 x_4 \vee x_1\bar{x}_2\bar{x}_3 x_4 \vee$$

$$\vee x_1 x_2\bar{x}_3 x_4 \vee \bar{x}_1 x_2\bar{x}_3 x_4 \vee \bar{x}_1\bar{x}_2\bar{x}_3 x_4 \vee x_1 x_2 x_3 x_4.$$

Если максимальный ранг для функции равен r, а минимальный ранг j-го терма равен k, то преобразование (9.23) необходимо применить к j-му терму $r-k$ раз.

Произвольная нормальная конъюнктивная форма (НКФ) переводится в СКНФ путем следующего преобразования:

пусть $f_{\text{НКФ}} = \Phi_1$. Тогда

$$f_{\text{СКНФ}} = \Phi_1 \vee x_i\bar{x}_i = (\Phi_1 \vee x_i)(\Phi_1 \vee \bar{x}_i). \tag{9.24}$$

Пример 9.8. Преобразовать в СКНФ логическую функцию

$$f(x_1, x_2, x_3) = (\underset{\Phi_1}{x_1 \vee x_2})(\underset{\Phi_2}{x_2 \vee \bar{x}_3})(\underset{\Phi_3}{x_1 \vee x_2 \vee x_3}).$$

Решение. Применяем правило преобразований (9.24) поочередно к термам Φ_1 и Φ_2:

$$\Phi_1 = (x_1 \vee x_2) \vee x_3 \overline{x}_3 = (x_1 \vee x_2 \vee x_3)(x_1 \vee x_2 \vee \overline{x}_3);$$

$$\Phi_2 = (x_2 \vee \overline{x}_3) \vee x_1 \overline{x}_1 = (x_1 \vee x_2 \vee \overline{x}_3)(\overline{x}_1 \vee x_2 \vee \overline{x}_3).$$

После окончательных упрощений СКНФ имеет вид

$$f_{\text{СКНФ}}(x_1, x_2, x_3) = (x_1 \vee x_2 \vee x_3)(x_1 \vee x_2 \vee \overline{x}_3)(\overline{x}_1 \vee x_2 \vee \overline{x}_3).$$

§ 9.5. Полные системы функций алгебры логики

Одни логические функции, как было рассмотрено выше, можно выражать через другие логические функции.

Базис — *полная система ФАЛ, с помощью которой любая ФАЛ может быть представлена суперпозицией исходных функций.*

К базису относится система функций И, ИЛИ, НЕ (базис 1), свойства которых были изучены Дж. Булем. Поэтому алгебра высказываний, построенная на основе этих функций, названа булевой алгеброй. Базисами являются также системы, содержащие функции И, НЕ (базис 2), ИЛИ, НЕ (базис 3), состоящие из функции Шеффера (базис 4) и функции Пирса (Вебба) (базис 5). Это перечисление показывает, что базисы могут быть избыточными (базис 1) и минимальными (базисы 4 и 5).

Базис минимальный, *если удаление хотя бы одной функции превращает систему ФАЛ в неполную.*

Проблема простейшего представления логических функций сводится к выбору не только базиса, но и формы наиболее экономного представления этих функций.

Базис И, ИЛИ, НЕ является избыточной системой, так как возможно удаление из него некоторых функций. Например, используя законы де Моргана, можно удалить либо функцию И, заменив ее на функции ИЛИ и НЕ, либо функцию ИЛИ, заменив ее на функции И и НЕ.

Если сравнить в смысле минимальности различные формы представления ФАЛ, то очевидно, что нормальные формы экономичнее совершенных нормальных форм. Но с другой стороны, нормальные формы не дают однозначного представления.

Минимальная форма представления ФАЛ — *форма представления ФАЛ, которая содержит минимальное количество термов и переменных в термах, т. е. минимальная форма не допускает никаких упрощений.*

Например, функция $f(x_1, x_2, \ldots, x_n) = x_1 + x_2$ является минимальной формой, и наоборот, функция $x_1 + \overline{x}_1 x_2$ может быть упрощена, если к этому выражению применить распределительный закон т. е. $x_1 + \overline{x}_1 x_2 = (x_1 + \overline{x}_1)(x_1 + x_2) = x_1 + x_2$.

Следовательно, упрощение сложных логических выражений может быть осуществлено по основным законам и аксиомам, изложенным выше.

Пример 9.9. Упростить функцию $f(A,\ B,\ C)=A\overline{B}+B\overline{C}+\overline{B}C+\overline{A}B$ в базисе 1.

Решение. Сначала применим правило (9.23), а затем упростим функцию.

$$f(A,\ B,\ C)=A\overline{B}+B\overline{C}+B\overline{C}(A+\overline{A})+\overline{A}B(C+\overline{C})=$$
$$=A\overline{B}+B\overline{C}+AB\overline{C}+\overline{A}B\overline{C}+\overline{A}BC+\overline{A}B\overline{C}=A\overline{B}(1+C)+$$
$$+B\overline{C}(1+\overline{A})+\overline{A}C(B+\overline{B})=A\overline{B}+B\overline{C}+\overline{A}C.$$

Ответ: $f(A,\ B,\ C)=A\overline{B}+B\overline{C}+\overline{A}C.$

Пример 9.10. Упростить функцию $f(A,\ B,\ C,\ D)=A\overline{B}+C+\overline{A}\overline{C}D+B\overline{C}D$ в базисе 1.

Решение. Нетрудно доказать, что $x+\overline{x}y=x+y$. Используя также теорему де Моргана, получим

$$x+y=\overline{\overline{x}\,\overline{y}}.$$

Тогда

$$C+\overline{A}\overline{C}D=C+\overline{A}D;\quad C+B\overline{C}D=C+BD.$$

Следовательно,

$$f(A,\ B,\ C,\ D)=A\overline{B}+C+\overline{A}D+BD=A\overline{B}+D(\overline{A}+B)+C=$$
$$=A\overline{B}+D\overline{A\overline{B}}+C=A\overline{B}+C+D.$$

Ответ: $f(A,\ B,\ C,\ D)=A\overline{B}+C+D.$

§ 9.6. Числовое и геометрическое представление функций алгебры логики

Часто для упрощения записи функций алгебры логики вместо полного перечисления термов используют номера наборов, для которых функция принимает единичное значение. Например, функция, заданная табл. 9.5, может быть записана в виде $f(x_1,\ x_2,\ x_3)=\underset{1}{\bigvee} F(0,\ 3,\ 4)$; это означает, что функция принимает значение 1 на наборах, номера которых равны 0, 3 и 4. Такую форму записи называют **числовой.**

Многие преобразования, выполняемые над булевыми функциями, удобно интерпретируются с использованием их геометрических представлений. Так, функцию двух переменных можно интерпретировать как некоторую плоскость, заданную в системе координат x_1, x_2 (рис. 9.1). Отложим по каждой оси единичные отрезки x_1 и x_2. Получится квадрат, вершины которого соответствуют комбинациям переменных. Из такого геометрического представления функций двух переменных следует: две вершины, принадлежащие одному и тому же ребру и называемые соседними, «склеиваются» по переменной, меняющейся вдоль этого ребра.

Таким образом, **правило склеивания** для минтермов можно записать для функции трех переменных в следующем виде

$$x_1\overline{x}_2\overline{x}_3+x_1\overline{x}_2x_3=x_1\overline{x}_2.$$

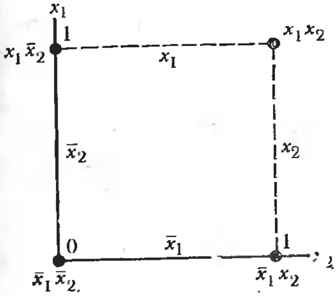


Рис. 9.1. Геометрическое представление функции двух переменных

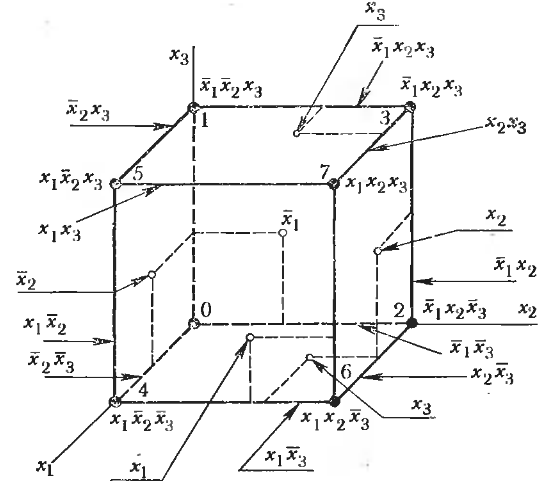


Рис. 9.2. Геометрическое представление функции трех переменных

Пример 9.11. Определить соседние минтермы и применить правило склеиваиия:

$$f_1(x_1, x_2, x_3) = x_1x_2x_3; \quad f_4(x_1, x_2, x_3) = \bar{x}_1x_2\bar{x}_3;$$

$$f_2(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2\bar{x}_3; \quad f_5(x_1, x_2, x_3) = x_1\bar{x}_2x_3.$$

$$f_3(x_1, x_2, x_3) = x_1x_2\bar{x}_3;$$

Р е ш е н и е. В соответствии с определением выпишем пары соседних термов:

$$f_1 \text{ и } f_3; \quad f_1 \text{ и } f_5; \quad f_2 \text{ и } f_4; \quad f_3 \text{ н } f_4.$$

Применив правило склеивания к этим парам, получим новые термы:

$$x_1x_2x_3 + x_1x_2\bar{x}_3 = x_1x_2; \quad \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 = \bar{x}_1\bar{x}_3;$$

$$x_1x_2x_3 + x_1\bar{x}_2x_3 = \bar{x}_1x_3; \quad x_1x_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 = x_2\bar{x}_3.$$

Для функций трех переменных геометрическое представление выполняют в виде куба (рис. 9.2). Ребра куба поглощают вершины. Грани куба поглощают свои ребра и, следовательно, вершины.

Функция четырех переменных представляется уже в виде четырехмерного куба (рис. 9.3). В геометрическом смысле каждый набор $x_1x_2x_3\ldots x_n$ может рассматриваться как n-мерный вектор, определяющий точку n-мерного пространства. Исходя из этого все множество наборов, на которых определена функция n переменных, представляется в виде вершин n-мерного куба. Координаты вершин куба должны быть указаны в порядке, соответствующем порядку перечисления переменных в записи функций. Отмечая точками вершины, в которых функция принимает значение, равное 1, получаем геометрическое представление ФАЛ.

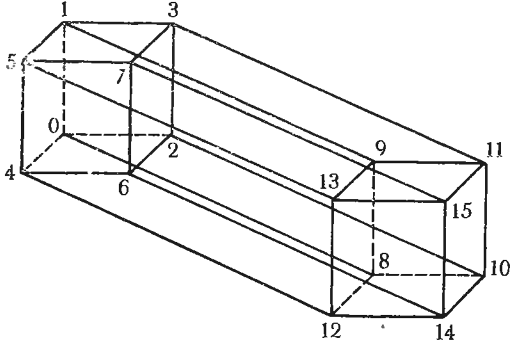


Рис. 9 3. Геометрическое представление функции четырех переменных

Терм максимального ранга принято называть 0-кубом (точка) и обозначать K^0.

Например, для $f(x_1x_2x_3) = \bigvee_1 (0, 4, 7)$

$$K^0 = \begin{Bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{Bmatrix}.$$

Если два 0-куба из комплекса K^0 различаются только по одной координате, то они образуют 1-куб (отрезок):

$$K^1 = \{x\ 0\ 0\},$$

где x — независимая координата.

Если два 1-куба имеют общую независимую компоненту и различаются только по одной координате, то они образуют 2-куб.

Таким образом, для построения одномерного единичного куба берут два 0-куба (точки) и соединяют отрезком прямой. Двумерный куб (грань) получается, если вершины двух 1-кубов соединить параллельными отрезками. Трехмерный куб получается при соединении соответствующих вершин двух двумерных кубов отрезками единичной длины. Геометрическое представление будет использовано в гл. 10 при разработке методов минимизации с использованием минимизирующих карт.

Задание для самоконтроля

1. Написать произвольную конъюнкцию для функции: а) трех переменных, б) n переменных.
2. Написать произвольную функцию Шеффера для функции:
а) трех переменных, б) n переменных.
3. Написать законы де Моргана для функции трех переменных.

4. Доказать, что $f_1(x_1, x_2, x_3) = f_2(x_1, x_2, x_3)$, где $f_1(x_1, x_2, x_3) = \bar{x}_1x_2x_3 + x_1\bar{x}_2x_3 + x_1x_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 + x_1\bar{x}_2\bar{x}_3 + x_1x_2x_3$; $f_2(x_1, x_2, x_3) = x_1 + x_2$.

5. Преобразовать логические функции к нормальной форме:

$$f_1(x_1, x_2, x_3) = \overline{(x_1x_2 + \bar{x}_2x_3)}\,\overline{x_1x_2};$$

$$f_2(x_1, x_2, x_3) = (\overline{\overline{x_1x_2x_3} + \overline{x_2x_3}}) + \overline{x_1\bar{x}_3},$$

используя свойства элементарных функций.

6. Преобразовать в СНДФ и СКНФ функции вида: а) $f(x_1, x_2, x_3) = x_1 + \bar{x}_2x_3 + \bar{x}_1x_2x_3 + x_1\bar{x}_3$; б) $f(x_1, x_2, x_3) = x_1 + \bar{x}_2 + x_3$.

7. Представить в базисе «импликация, отрицание» функцию, числовое представление которой имеет вид: а) $f_1(x_1, x_2, x_3) = \bigwedge\limits_0(3, 4, 6)$; б) $f_2(x_1, x_2, x_3) = \bigvee\limits_1(1, 2, 5)$.

8. Представить в базисе Шеффера функцию $f(x_1, x_2, x_3) = \bar{x}_1\bar{x}_3 + \bar{x}_2\bar{x}_3 + x_1x_2x_3$.

9. Представить в базисе Пирса (Вебба) функцию $f(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2 + \bar{x}_2\bar{x}_3 + x_1x_2x_3$.

10. Упростить логические выражения, используя свойства элементарных функций: а) $f(x_1, x_2, x_3) = \bar{x}_1x_2x_3 + x_1\bar{x}_2x_3 + \bar{x}_1x_2x_3 + x_1x_2x_3 + \bar{x}_1x_2x_3$ б) $f(x_1, x_2, x_3) = \bigvee\limits_1(1, 3, 5, 6, 7)$.

10 ГЛАВА

МИНИМИЗАЦИЯ ФУНКЦИЙ АЛГЕБРЫ ЛОГИКИ

§ 10.1. Метод неопределенных коэффициентов для базиса И — ИЛИ — НЕ

На основании (9.18) любую логическую функцию можно представить в нормальной форме. Например, пусть функция $f(x_1, x_2, x_3)$ записана в виде следующей нормальной дизъюнктивной формы (НДФ):

$$\left.\begin{aligned}
f(x_1, x_2, x_3) = {} & K_1^1 x_1 + K_1^0 \bar{x}_1 + K_2^1 x_2 + K_2^0 \bar{x}_2 + K_3^1 x_3 + \\
& + K_3^0 \bar{x}_3 + K_{12}^{11} x_1 x_2 + K_{12}^{10} x_1 \bar{x}_2 + K_{12}^{01} \bar{x}_1 x_2 + K_{12}^{00} \bar{x}_1 \bar{x}_2 + \\
& + K_{13}^{11} x_1 x_3 + K_{13}^{10} x_1 \bar{x}_3 + K_{13}^{01} \bar{x}_1 x_3 + K_{13}^{00} \bar{x}_1 \bar{x}_3 + \\
& + K_{23}^{11} x_2 x_3 + K_{23}^{10} x_2 \bar{x}_3 + K_{23}^{01} \bar{x}_2 x_3 + K_{23}^{00} \bar{x}_2 \bar{x}_3 + \\
& + K_{123}^{111} x_1 x_2 x_3 + K_{123}^{110} x_1 x_2 \bar{x}_3 + K_{123}^{101} x_1 \bar{x}_2 x_3 + K_{123}^{100} x_1 \bar{x}_2 \bar{x}_3 + \\
& + K_{123}^{011} \bar{x}_1 x_2 x_3 + K_{123}^{010} \bar{x}_1 x_2 \bar{x}_3 + K_{123}^{001} \bar{x}_1 \bar{x}_2 x_3 + K_{123}^{000} \bar{x}_1 \bar{x}_2 \bar{x}_3,
\end{aligned}\right\} \quad (10.1)$$

где K_{ijk}^{lmn} — неопределенные коэффициенты, принимающие значение 0 или 1 и подбираемые так, чтобы получающаяся после этого НДФ была минимальной.

Критерием минимальности является минимальное количество букв в записи НДФ. При определении НДФ пользуются следующими свойствами: $x_1 + x_2 + \ldots + x_n = 0$, если $x_1 = x_2 = \ldots = x_n = 0$, $x_1 + x_2 \ldots$ $\ldots + x_n = 1$, если хотя бы один член уравнения равен 1.

Так как функция может принимать на наборах значения 0 и 1, то на основании уравнения (10.1) можно получить:

$$\left.\begin{aligned}
& K_1^0 + K_2^0 + K_3^0 + K_{12}^{00} + K_{13}^{00} + K_{23}^{00} + K_{123}^{000} = f_0(0, 0, 0); \\
& K_1^0 + K_2^0 + K_3^1 + K_{12}^{00} + K_{13}^{01} + K_{23}^{01} + K_{123}^{001} = f_1(0, 0, 1); \\
& K_1^0 + K_2^1 + K_3^0 + K_{12}^{01} + K_{13}^{00} + K_{23}^{10} + K_{123}^{010} = f_2(0, 1, 0); \\
& K_1^0 + K_2^1 + K_3^1 + K_{12}^{01} + K_{13}^{01} + K_{23}^{11} + K_{123}^{011} = f_3(0, 1, 1); \\
& K_1^0 + K_2^0 + K_3^0 + K_{12}^{10} + K_{13}^{10} + K_{23}^{00} + K_{123}^{100} = f_4(1, 0, 0); \\
& K_1^1 + K_2^0 + K_3^1 + K_{12}^{10} + K_{13}^{11} + K_{23}^{01} + K_{123}^{101} = f_5(1, 0, 1); \\
& K_1^1 + K_2^1 + K_3^0 + K_{12}^{11} + K_{13}^{10} + K_{23}^{10} + K_{123}^{110} = f_6(1, 1, 0); \\
& K_1^1 + K_2^1 + K_3^1 + K_{12}^{11} + K_{13}^{11} + K_{23}^{11} + K_{123}^{111} = f_7(1, 1, 1).
\end{aligned}\right\} \quad (10.2)$$

Если $f_i=0$ на соответствующем наборе переменных, то все коэффициенты, входящие в данное уравнение, равны 0. Тогда в остальных уравнениях (10.2) надо вычеркнуть члены, содержащие нулевые коэффициенты, а из оставшихся уравнений, равных 1, найти коэффициенты, определяющие конъюнкцию наименьшего ранга в каждом из уравнений.

На основании изложенного можно сформулировать следующий алгоритм нахождения неопределенных коэффициентов:

1. Выбрать очередную строку, в которой $f_i=0$. Все коэффициенты этой строки приравнять 0.
2. Если все нулевые строки просмотрены, то перейти к п. 3, если нет, то к п. 1.
3. Просмотреть строки, в которых $f_i=1$, и вычеркнуть из них все коэффициенты, встречающиеся в строках, где $f_i=0$.
4. Переписать модифицированные уравнения.
5. Выбрать очередную строку $f_i=1$ и вычеркнуть максимально возможное количество коэффициентов так, чтобы ранг остающихся членов был минимальным.

Метод неопределенных коэффициентов наиболее применим для дизъюнктивной формы и практически непригоден для конъюнктивной формы.

Пример 10.1. Найти минимальную форму для функции, заданной в виде

$$f(x_1, x_2, x_3)=\vee(0, 2, 4, 7)=\overline{x_1}\,\overline{x_2}\,\overline{x_3}+\overline{x}_1x_2\overline{x}_3+x_1\overline{x}_2\overline{x}_3+x_1x_2x_3.$$

Решение. Составим систему уравнений (10.2), которую удобнее всего записать в виде таблицы (табл. 10.1). После вычеркивания нулевых коэффициентов уравнения (10.2) принимают такой вид:

$$K_{123}^{111}=1;$$

$$K_{23}^{00}+K_{123}^{100}=1;$$

$$K_{13}^{00}+K_{123}^{010}=1;$$

$$K_{13}^{00}+K_{23}^{00}+K_{123}^{000}=1.$$

Результат: $K_{13}^{00}=1$; $K_{23}^{00}=1$; $K_{123}^{111}=1$.

Ответ: $f(x_1, x_2, x_3)=\overline{x}_1\overline{x}_3+\overline{x}_2\overline{x}_3+x_1x_2x_3.$

§ 10.2. Метод Квайна

При минимизации по методу Квайна (базис 1) предполагается, что исходная функция задана в совершенной нормальной дизъюнктивной форме (СНДФ).

Импликанта функции f — *некоторая логическая функция* φ, *обращаемая в* 0 *при наборе переменных, на котором сама функция также равна* 0.

Поэтому любой конъюнктивный терм, входящий в состав СНДФ, или группа термов, соединенных знаками дизъюнкции, являются импликантами исходной НДФ.

Таблица 10.1.

$\cancel{K_1^1}$	$\cancel{K_2^1}$	$\cancel{K_3^1}$	$\cancel{K_{12}^{11}}$	$\cancel{K_{13}^{11}}$	$\cancel{K_{23}^{11}}$	K_{123}^{111}	1
$\cancel{K_1^1}$	$\cancel{K_2^1}$	$\cancel{K_3^0}$	$\cancel{K_{12}^{11}}$	$\cancel{K_{13}^{10}}$	$\cancel{K_{23}^{10}}$	K_{123}^{110}	0
$\cancel{K_1^1}$	$\cancel{K_2^0}$	$\cancel{K_3^1}$	$\cancel{K_{12}^{10}}$	$\cancel{K_{13}^{11}}$	$\cancel{K_{23}^{01}}$	$\cancel{K_{123}^{101}}$	0
$\cancel{K_1^1}$	$\cancel{K_2^0}$	$\cancel{K_3^0}$	$\cancel{K_{12}^{10}}$	$\cancel{K_{13}^{10}}$	K_{23}^{00}	K_{123}^{100}	1
$\cancel{K_1^0}$	$\cancel{K_2^1}$	$\cancel{K_3^1}$	$\cancel{K_{12}^{01}}$	$\cancel{K_{13}^{01}}$	$\cancel{K_{23}^{11}}$	$\cancel{K_{123}^{011}}$	0
$\cancel{K_1^0}$	$\cancel{K_2^1}$	$\cancel{K_3^0}$	$\cancel{K_{12}^{01}}$	$\cancel{K_{13}^{00}}$	$\cancel{K_{23}^{10}}$	K_{123}^{010}	1
$\cancel{K_1^0}$	$\cancel{K_2^0}$	$\cancel{K_3^1}$	$\cancel{K_{12}^{00}}$	$\cancel{K_{13}^{01}}$	$\cancel{K_{23}^{01}}$	$\cancel{K_{123}^{001}}$	0
$\cancel{K_1^0}$	$\cancel{K_2^0}$	$\cancel{K_3^0}$	$\cancel{K_{12}^{00}}$	K_{13}^{00}	K_{23}^{00}	K_{123}^{000}	1

Первичная импликанта функции — *импликанта типа элементарной конъюнкции некоторых переменных, никакая часть которой уже не является импликантой.*

Задача минимизации по методу Квайна состоит в попарном сравнении всех импликант, входящих в СНДФ, с целью выявления возможности поглощения какой-то переменной:

$$Fx_i \vee F\bar{x}_i = F. \tag{10.3}$$

Таким образом удается снизить ранг термов. Эта процедура проводится до тех пор, пока не останется ни одного члена, допускающего поглощение с каким-либо другим термом. Термы, подвергшиеся поглощению, отмечаются. Неотмеченные термы представляют собой первичные импликанты.

Полученное логическое выражение не всегда оказывается минимальным. Поэтому исследуется возможность дальнейшего упрощения. Для этого составляется таблица, в строках которой записываются найденные первичные импликанты, а в столбцах указываются термы исходного уравнения. Клетки этой таблицы отмечаются в случае, если первичная импликанта входит в состав какого-либо терма. После этого задача упрощения сводится к тому, чтобы найти такое минимальное количество первичных импликант, которые покрывают все столбцы.

Таким образом, метод Квайна выполняется в несколько этапов. Рассмотрим это на конкретном примере.

Таблица 10.2

Исходные термы	1 $\bar{x}_1\bar{x}_2x_3x_4$	2 $\bar{x}_1x_2\bar{x}_3\bar{x}_4$	3 $\bar{x}_1x_2\bar{x}_3x_4$	4 $\bar{x}_1x_2x_3x_4$	5 $x_1\bar{x}_2\bar{x}_3x_4$	6 $x_1\bar{x}_2x_3x_4$	7 $x_1x_2\bar{x}_3\bar{x}_4$	8 $x_1x_2\bar{x}_3x_4$
$\bar{x}_1\bar{x}_2x_3x_4$	1			$\boxed{\bar{x}_1x_3x_4}$		$\boxed{\bar{x}_2x_3x_4}$		
$\bar{x}_1x_2\bar{x}_3\bar{x}_4$		1	$\boxed{\bar{x}_1x_2\bar{x}_3}$				$\boxed{x_2\bar{x}_3\bar{x}_4}$	
$\bar{x}_1x_2\bar{x}_3x_4$		$\bar{x}_1x_2\bar{x}_3$	1	$\boxed{\bar{x}_1x_2x_4}$				$\boxed{x_2\bar{x}_3x_4}$
$\bar{x}_1x_2x_3x_4$	$\bar{x}_1x_3x_4$		$\bar{x}_1x_2x_4$	1				
$x_1\bar{x}_2\bar{x}_3x_4$					1	$\boxed{x_1\bar{x}_2x_4}$		$\boxed{x_1\bar{x}_3x_4}$
$x_1\bar{x}_2x_3x_4$	$\bar{x}_2x_3x_4$				$x_1\bar{x}_2x_4$	1		
$x_1x_2\bar{x}_3\bar{x}_4$		$x_2\bar{x}_3\bar{x}_4$					1	$\boxed{x_1x_2\bar{x}_3}$
$x_1x_2\bar{x}_3x_4$			$x_2\bar{x}_3x_4$		$x_1\bar{x}_3x_4$		$x_1x_2\bar{x}_3$	1

Таблица 10.3

Первичные импликанты ранга 3	$\boxed{\bar{x}_1x_3x_4}$	$\boxed{\bar{x}_2x_3x_4}$	$\bar{x}_1x_2\bar{x}_3$	$x_2\bar{x}_3\bar{x}_4$	$\boxed{\bar{x}_1x_2x_4}$	$x_2\bar{x}_1x_4$	$\boxed{x_1\bar{x}_2x_4}$	$\boxed{x_1\bar{x}_3x_4}$	$x_1x_2\bar{x}_3$
$\bar{x}_1x_3x_4$	1								
$\bar{x}_2x_3x_4$		1							
$\bar{x}_1x_2\bar{x}_3$			1						$\boxed{x_2\bar{x}_3}$
$x_2\bar{x}_3\bar{x}_4$				1		$\boxed{x_2\bar{x}_3}$			
$\bar{x}_1x_2x_4$					1				
$x_2\bar{x}_3x_4$				$x_2\bar{x}_3$		1			
$x_1\bar{x}_2x_4$							1		
$x_1\bar{x}_3x_4$								1	
$x_1x_2\bar{x}_3$			$x_2\bar{x}_3$						1

Пусть необходимо минимизировать логическую функцию, заданную в виде

$$f(x_1,\ x_2,\ x_3,\ x_4)=\bigvee_1(3,\ 4,\ 5,\ 7,\ 9,\ 11,\ 12,\ 13)=$$

$$=\overline{x}_1\overline{x}_2x_3x_4\vee\overline{x}_1x_2\overline{x}_3\overline{x}_4\vee\overline{x}_1x_2\overline{x}_3x_4\vee\overline{x}_1x_2x_3x_4\vee$$

$$\vee\, x_1\overline{x}_2\overline{x}_3x_4\vee x_1\overline{x}_2x_3x_4\vee x_1\overline{x}_2\overline{x}_3x_4\vee x_1\overline{x}_2x_3\overline{x}_4.$$

Задача решается в несколько этапов.

Этап 1. Нахождение первичных импликант. Прежде всего составляется таблица (табл. 10.2) и находятся импликанты четвертого и третьего ранга, т. е. снижается ранг членов, входящих в СНДФ. Затем составляется другая таблица (табл. 10.3), которая включает все термы, не подвергшиеся поглощению, а также первичные импликанты третьего ранга. Составление таблиц продолжается до тех пор, пока нельзя будет применить правило (10.3). В рассматриваемом примере можно дойти до первичной импликанты второго ранга (табл. 10.3) —$x_2\bar{x}_3$.

Таким образом, найдены первичные импликанты наименьшего ранга (отмечены прямоугольником в табл. 10.3).

Этап 2. Расстановка меток. Составляется таблица, число строк которой равно числу полученных первичных импликант, а число столбцов совпадает с числом минтермов СНДФ. Если в некоторый минтерм СНДФ входит какая-либо из первичных импликант, то на пересечении соответствующего столбца и строки ставится метка (табл. 10.4).

Таблица 10.4

Первичные импликанты	1	2	3	4	5	6	7	8
	Исходные термы							
	$\bar{x}_1\bar{x}_2x_3x_4$	$\bar{x}_1x_2\bar{x}_3\bar{x}_4$	$\bar{x}_1x_2\bar{x}_3x_4$	$\bar{x}_1x_2x_3x_4$	$x_1\bar{x}_2\bar{x}_3x_4$	$x_1\bar{x}_2x_3x_4$	$x_1x_2\bar{x}_3\bar{x}_4$	$x_1x_2\bar{x}_3x_4$
$\bar{x}_1x_3x_4$	$\vee$			$\vee$				
$\bar{x}_2x_3x_4$	$\vee$					$\vee$		
$\bar{x}_1x_2x_4$			$\vee$	$\vee$				
$x_1\bar{x}_2x_4$					$\vee$	$\vee$		
$x_1\bar{x}_3x_4$					$\vee$		$\vee$	$\vee$
$\boxed{x_2\bar{x}_3}$		$\vee$	$\vee$				$\vee$	$\vee$

Этап 3. Нахождение существенных импликант. Если в каком-либо из столбцов табл. 10.4 имеется только одна метка, то первичная импликанта в соответствующей строке является существенной, так как без нее не будет получено все множество заданных минтермов. В табл. 10.4 существенной импликантой является терм $x_2\bar{x}_3$. Столбцы, соответствующие существенным импликантам, из таблицы вычеркиваются.

Этап 4. Вычеркивание лишних столбцов. После третьего этапа в результате вычеркивания столбцов 2, 3, 7 и 8 получается табл. 10.5. Если в таблице есть два столбца, в которых имеются метки в одинаковых строках, то один из них вычеркивается. Покрытие оставшегося столбца будет осуществлять отброшенный минтерм. В примере такого случая нет.

Таблица 10.5

Первичные импликанты	1	4	5	6
	Исходные термы			
	$\bar{x}_1\bar{x}_2x_3x_4$	$\bar{x}_1x_2x_3x_4$	$x_1\bar{x}_2\bar{x}_3x_4$	$x_1\bar{x}_2x_3x_4$
$\boxed{\bar{x}_1x_3x_4}$	$\vee$	$\vee$		
$\bar{x}_2x_3x_4$	$\vee$			$\vee$
$\bar{x}_1x_2x_4$		$\vee$		
$\boxed{x_1\bar{x}_2x_4}$			$\vee$	$\vee$
$x_1\bar{x}_3x_4$			$\vee$	

Этап 5. Вычеркивание лишних первичных импликант. Если после отбрасывания некоторых столбцов на этапе 4 в табл. 10.5 появляются строки, в которых нет ни одной метки, то первичные импликанты, соответствующие этим строкам, исключаются из дальнейших рассмотрений, так как они не покрывают оставшиеся в рассмотрении минтермы.

Этап 6. Выбор минимального покрытия. Выбирается в табл. 10.5 такая совокупность первичных импликант, которая включает метки во всех столбцах (по крайней мере по одной метке в каждом столбце). При нескольких возможных вариантах такого выбора отдается предпочтение варианту покрытия с минимальным суммарным числом букв в импликантах, образующих покрытие. Этому требованию удовлетворяют первичные импликанты $\bar{x}_1x_3x_4$ и $x_1\bar{x}_2x_4$.

Таким образом, минимальная форма заданной функции будет складываться из суммы существенных импликант (этап 3) и первичных импликант, покрывающих оставшиеся минтермы (этап 6):

$$f(x_1, x_2, x_3, x_4)=x_2\bar{x}_3 \vee \bar{x}_1x_3x_4 \vee x_1\bar{x}_2x_4.$$

§ 10.3. Метод Квайна — Мак-Класки

Недостатком метода Квайна является необходимость полного попарного сравнения всех минтермов на этапе нахождения первичных импликант. С ростом числа минтермов увеличивается количество попарных сравнений. Числовое представление функций алгебры логики позволяет упростить этап 1 (см. § 10.2). Все минтермы записываются в виде их двоичных номеров, а все номера разбиваются по числу единиц на непересекающиеся группы, так как условием образования r-куба является наличие расхождения в $(r-1)$-кубах только по одной координате (в одном двоичном разряде) и наличие общих независимых координат. Поэтому группы, которые различаются в двух разрядах или более, просто не имеет смысла сравнивать. При этом в i-группу войдут все номера (наборы), имеющие в своей двоичной записи i единиц. Попарное сравнение можно производить только между соседними по номеру группами.

Пример 10.2. Пусть задана функцня

$$f(x_1, x_2, x_3, x_4) = \vee(3, 4, 5, 7, 9, 11, 12, 13).$$

Решение. Сначала выпишем 0-кубы:

$$K^0 = \{0011, 0100, 0101, 0111, 1001, 1011, 1100, 1101\}.$$

Разобьем 0-кубы на три группы по количеству едииц в каждом двоичном наборе:

$$K_1^0 = \{0100\}; \quad K_2^0 = \begin{Bmatrix} 0011 \\ 0101 \\ 1001 \\ 1100 \end{Bmatrix}; \qquad K_3^0 = \begin{Bmatrix} 0111 \\ 1011 \\ 1101 \end{Bmatrix}.$$

Этап 1. Нахождение первичных импликант:

а) сравнение K_1^0 и K_2^0:

$$\begin{array}{ll} 0100^* & 0011 \\ & 0101^* \\ & 1001 \\ & 1100^* \end{array}$$

Здесь и ниже символом * отмечены наборы, которые склеиваются.

На основании сравнения строим куб K_1^1, в котором поглощениая координата заменяется символом x:

$$K_1^1 = \begin{Bmatrix} 010x \\ x100 \end{Bmatrix};$$

б) сравнение K_2^0 и K_3^0:

$$\begin{array}{ll} 0011^* & 0111^* \\ 0101^* & 1011^* \\ 1001^* & 1101^* \\ 1100^* & \end{array}$$

Строим куб K_2^1 :

$$K_2^1 = \begin{Bmatrix} 0x11 \\ x011 \\ 01x1 \\ 10x1 \\ 1x01 \\ 110x \\ x101 \end{Bmatrix}$$

Первичных импликант ранга 4 нет;

в) разобьем все 1-кубы на четыре группы в зависимости от положения независимой координаты x:

$$K_1^2=\begin{Bmatrix}010x\\110x\end{Bmatrix};\qquad K_1^3=\begin{Bmatrix}01x1\\10x1\end{Bmatrix};\qquad K_1^4=\begin{Bmatrix}0x11\\1x01\end{Bmatrix};\qquad K_1^5=\begin{Bmatrix}x100\\x011\\x101\end{Bmatrix};$$

г) сравнение K_1^2 и K_1^3, K_1^4 и K_1^5 внутри каждой группы:

$$\begin{matrix}010x & 01x1^* & 0x11^* & x100\\ 110x & 10x1^* & 1x01^* & x011^*\\ & & & x101\end{matrix}$$

На основании сравнения строим кубы $K_1^{2'}=x10x$; $K_1^{2^{IV}}=x10x$; K_1^3 и $K_1^{\bar{4}}$ не сравниваются.

Следовательно, символом $*$ отмечены первичные импликанты ранга 3:

$$K^1=\{01x1;\quad 10x1;\quad 0x11;\quad 1x01;\quad x011\};$$

д) сравнение $K_1^{2'}$ и $K_1^{2^{IV}}$:

$$K_1^{2'}=K_1^{2^{IV}}.$$

Следовательно, получаем первичную импликанту ранга 2:

$$K^2=\{x10x\}.$$

Этап 2. Расстановка меток (табл. 10.6).

Таблица 10.6

Первичные импликанты	Исходные термы							
	0011	0100	0101	0111	1001	1011	1100	1101
01x1				*				
10x1					*	*		
0x11	*			*				
1x01					*		*	*
x011	*							
$\boxed{x10x}$		*	*				*	*

Этап 3. Нахождение существенных импликант.

Существенной импликантой ранга 2 будет терм

$$\{x10x\}=x_2\overline{x}_3.$$

Этапы 4 и 5. Отсутствуют.

Этап 6. Выбирается минимальное покрытие оставшихся термов $\{10x1\}$ и $\{0x11\}$ (табл. 10.7).

Первичные импликанты	Исходные термы			
	0011	0111	1001	1011
01x1		*		
$\boxed{10x1}$			*	*
$\boxed{0x11}$	*	*		
1x01			*	
x011	*			*

Результат равен

$$f(x_1, x_2, x_3, x_4) = x_2\bar{x}_3 \vee x_1\bar{x}_2x_4 \vee \bar{x}_1x_3x_4.$$

При использовании метода Квайна для СКНФ необходимо рассматривать значение функции $f=0$ и термы, соответствующие этим значениям. В результате получим $\bar{f} = \bigvee\limits_1 (x_1, x_2, \ldots, x_n)$. Далее необходимо воспользоваться соотношениями де Моргана, с тем чтобы привести функцию к СНДФ. Все дальнейшие действия аналогичны вышеизложенным.

§ 10.4. Метод минимизирующих карт

Одним из способов графического представления булевых функций от небольшого числа переменных являются карты Карно. Их разновидность — карты Вейча, которые строят как развертки кубов на плоскости. При этом вершины куба представляются клетками карты, координаты которых совпадают с координатами соответствующих вершин куба. Карта заполняется так же, как таблица истинности: значение 1 указывается в клетке, соответствующей набору, на котором функция имеет значение 1. Значение 0 обычно на картах не отражается. На рис. 10.1 показаны примеры условного размещения переменных на минимизирующих картах для двух (рис. 10.1, *а*), трех (рис. 10.1, *б*) и четырех (рис. 10.1, *в*) переменных. Примеры использования карт Карно для минимизации указанных ниже функций даны на рис. 10.2:

$$f_1 = x_1\bar{x}_2 \vee x_1x_2 \text{ (рис. 10.2, } a);$$

$$f_2 = \bar{x}_1\bar{x}_2\bar{x}_3 \vee x_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1x_2\bar{x}_3 \vee x_1x_2x_3 \text{ (рис. 10.2, } б);$$

$$f_3 = \bar{x}_1\bar{x}_2x_3x_4 \vee \bar{x}_1x_2\bar{x}_3x_4 \vee \bar{x}_1x_2\bar{x}_3x_4 \vee \bar{x}_1x_2x_3x_4 \vee$$

$$\vee x_1\bar{x}_2\bar{x}_3x_4 \vee x_1\bar{x}_2x_3x_4 \vee x_1x_2\bar{x}_3x_4 \vee x_1x_2\bar{x}_3\bar{x}_4 \text{ (рис. 10.2, } в).$$

Карты Карно, как правило, используют для ручной минимизации булевых функций при небольшом числе переменных.

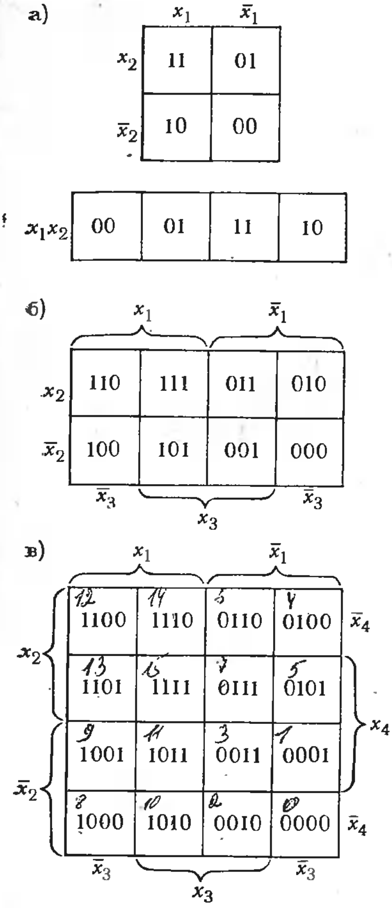


Рис. 10.1. Минимизирующие карты для функции двух (a), трех ($б$), четырех ($в$) переменных

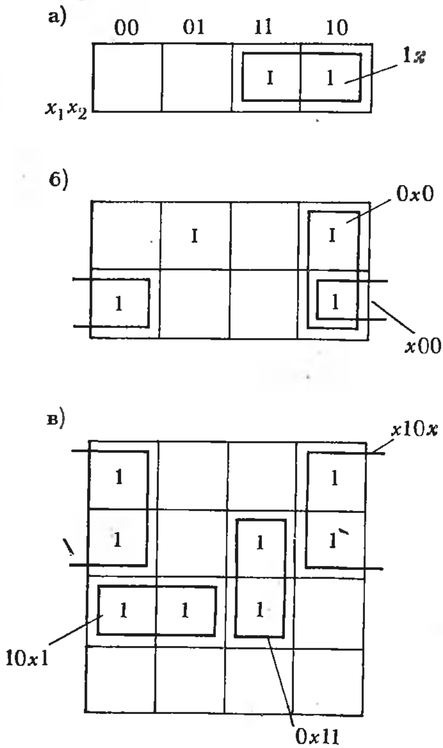


Рис. 10.2. Минимизация логических функций двух (a), трех ($б$) и четырех ($в$) переменных с помощью карт

Правила минимизации следующие.

1. Две соседние клетки (два 0-куба) образуют один 1-куб. При этом имеется в виду, что клетки, лежащие на границах карты, также являются соседними по отношению друг к другу (пример образования 1-куба см. на рис. 10.2, a. Независимая координата обозначена символом x).

2. Четыре вершины могут объединяться, образуя 2-куб, содержащий две независимые координаты (пример образования 2-куба изображен на рис. 10.2, $в$).

3. Восемь вершин могут объединяться, образуя один 3-куб.

4 Шестнадцать вершин, объединяясь, образуют один 4-куб и т. д.

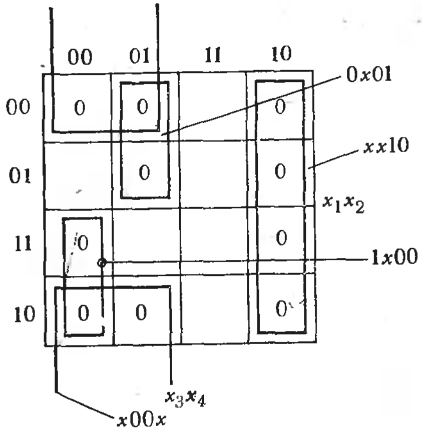


Рис. 10.3. Пример миниимизации для конъюнктивной формы

Таким образом, при числе переменных, равном или большем пяти, отобразить графически функцию в виде единой плоской карты невозможно. В таких случаях строят комбинированную карту, состоящую из совокупности более простых карт, например четырехмерных. Тогда процедура минимизации будет состоять в том, что сначала находят минимальные формы внутри четырехмерных кубов, а затем, расширяя понятие соседних клеток, отыскивают минимальные термы для совокупности карт. Соседними клетками являются клетки, совпадающие при совмещении карт поворотом вокруг общего ребра.

При получении минимальной формы для СКНФ функция задается термами, принимающими нулевое значение на соответствующих наборах. Поэтому в клетках минимизирующей карты пишут нули.

Пример 10.3. Найти минимальную конъюнктивную форму для функции (рис. 10.3) $f(x_1, x_2, x_3, x_4) = \underset{0}{\mathrm{V}}\ (0, 1, 2, 5, 6, 8, 9, 10, 12, 14)$ методом минимизирующих карт.

Решение. С помощью минимизирующих карт находим

$$\overline{f}\,(x_1,\ x_2,\ x_3,\ x_4) = \overline{x}_1\overline{x}_3x_4 \vee x_3\overline{x}_4 \vee x_1\overline{x}_3\overline{x}_4 \vee \overline{x}_2\overline{x}_3.$$

Применяя правила де Моргана, получаем

$$f = \overline{\overline{x}_1\overline{x}_3x_4 \vee x_3\overline{x}_4 \vee x_1\overline{x}_3\overline{x}_4 \vee \overline{x}_2\overline{x}_3} = \overline{\overline{x}_1\overline{x}_3x} \wedge \overline{x_3\overline{x}_4} \wedge \overline{x_1\overline{x}_3\overline{x}_4} \wedge \overline{\overline{x}_2\overline{x}_3} =$$

$$= (x_1 \vee x_3 \vee \overline{x}_4)\,(\overline{x}_3 \vee x_4)\,(\overline{x}_1 \vee x_3 \vee x_4)\,(x_2 \vee x_3).$$

Ответ: $f\,(x_1,\ x_2,\ x_3,\ x_4) = (x_1 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_3 \vee x_4) \wedge (\overline{x}_1 \vee x_3 \vee x_4) \wedge (x_2 \vee x_3)$.

§ 10.5. Минимизация логических функций, заданных в базисе ($\oplus$, $\wedge$, $\neg$)

Метод неопределенных коэффициентов может быть применен для минимизации функций, заданных в разных базисах.

Рассмотрим применение метода неопределенных коэффициентов на примере базиса ($\oplus$, $\wedge$, $\neg$). Функцию $f(x_1, x_2, x_3)$ представим в виде, аналогичном нормальной дизъюнктивной форме, где вместо дизъюнкции стоит знак операции сложения $\oplus$ по модулю 2. Эта операция имеет особенности, отличающие ее от операции дизъюнкции:

$$0 = 0 \oplus 0 \oplus 0 \oplus \ldots \oplus 0, \tag{1}$$

или

$$0=\underbrace{1\oplus 1\oplus 1\oplus \ldots \oplus 1}_{m}, \qquad (2)$$

где $m=2k$ — четное количество единиц;

$$1=\underbrace{1\oplus 1\oplus 1\oplus \ldots \oplus 1}_{m}, \qquad (3)$$

где $m=2k+1$ — нечетное количество единиц.

Для операции дизъюнкции всегда $1=1\vee 1\vee 1\ldots$. Наличие свойств (2) и (3) операции сложения по модулю 2 усложняет минимизацию. Так как основными критериями минимизации по-прежнему являются минимальный ранг каждого терма и минимальное количество термов, то при минимизации в базисе $(\oplus, \wedge, \neg)$ целесообразно приравнять нулю все коэффициенты на наборах, где $f=0$, так как тогда в единичных строках могут остаться термы высокого ранга. Поэтому особой разницы между выбором очередной строки (нулевой или единичной) нет. Количество коэффициентов, остающихся в нулевых строках, должно быть четным, а в единичных — нечетным. Лучше начать с единичных строк и оставлять те коэффициенты минимального ранга, которые чаще повторяются в этих строках.

Пример 10.4. Задана функция $f(x_1, x_2, x_3)=\oplus(0, 1, 5, 6)$.
Найти минимальное представление в базисе $(\oplus, \wedge, \neg)$.
Решение. Составим табл. 10.8.

Таблица 10.8

K_1^1	K_2^1	K_3^1	K_{12}^{11}	K_{13}^{11}	K_{23}^{11}	K_{123}^{111}	0	111
K_1^1	K_2^1	K_3^0	K_{12}^{10}	K_{13}^{10}	K_{23}^{10}	K_{123}^{110}	1	110
K_1^1	K_2^0	K_3^1	K_{12}^{10}	K_{13}^{11}	K_{23}^{01}	K_{123}^{101}	1	101
K_1^1	K_2^0	K_3^0	K_{12}^{10}	K_{13}^{10}	K_{23}^{00}	K_{123}^{100}	0	100
K_1^0	K_2^1	K_3^1	K_{12}^{01}	K_{13}^{01}	K_{23}^{11}	K_{123}^{011}	0	011
K_1^0	K_2^1	K_3^0	K_{12}^{01}	K_{13}^{00}	K_{23}^{10}	K_{123}^{010}	0	010
K_1^0	K_2^0	K_3^1	K_{12}^{00}	K_{13}^{01}	K_{23}^{01}	K_{123}^{001}	1	001
K_1^0	K_2^0	K_3^0	K_{12}^{00}	K_{13}^{00}	K_{23}^{00}	K_{123}^{000}	1	000

В таблице коэффициент K_2^0 повторяется четыре раза и его целесообразно оставить. В нулевой строке надо оставить еще какой-нибудь коэффициент минимального ранга, который также повторится в тех единичных строках, в которых еще не было оставлено ни одного коэффициента. Минимальная форма будет получена после выполнения следующих действий:

1. Посчитаем, сколько раз встречаются в единичных строках термы первого (минимального) ранга, и оставим те из них, которые встречаются максимальное число раз.

2. Найдем нулевые строки, в которых встречаются оставленные термы, и оставим в строках эти термы.

3. Рассмотрим нулевую строку, в которой остался хотя бы один единичный терм, и найдем в ней еще единичный терм ($1\oplus 1=0$), встречающийся максимальное число раз в единичных строках, в которых еще не было оставлено ни одного терма (строки 4, 6).

Результат получаем в виде уравнений:

$$K_{23}^{10}\oplus K_{13}^{10}\oplus K_{123}^{110}=1;$$

$$K_{2}^{0}\oplus K_{23}^{01}\oplus K_{123}^{101}=1;$$

$$K_{2}^{0}\oplus K_{12}^{00}\oplus K_{23}^{01}\oplus K_{123}^{001}=1;$$

$$K_{2}^{0}\oplus K_{12}^{00}\oplus K_{123}^{000}=1.$$

Ответ: $f(x_1, x_2, x_3)=\bar{x}_2\oplus x_1\bar{x}_3$.

Метод Квайна — Мак-Класки также может быть применен для минимизации в базисе ($\oplus$, $\wedge$, $\neg$). В этом случае кроме минтермов, на которых заданная функция принимает значение 1 (A-минтермы), в таблицу импликант включаются и некоторые минтермы, на которых функция принимает значение 0 (B-минтермы). Последние имеют отличие от A-минтермов в двух двоичных разрядах. B-минтермы включаются для того, чтобы обеспечить минимально возможный ранг термов в минимальной форме. Для функции трех переменных это вершины куба, которые расположены по диагонали к вершинам, на которых $f=1$. Так как $\underbrace{1\oplus 1}\oplus\underbrace{1\oplus 1}\oplus\ldots=0$, то в дальнейшем их можно исключить, используя дважды в минимальном покрытии функции f. В СНДФ этого сделать никогда невозможно, так как $1=1\vee 1$.

Для определения B-минтермов производится попарное сравнение всех A-минтермов. Если в сумме A_1 и A_2 имеются две единицы, то берется любой из минтермов A_1, A_2 и пишутся четыре минтерма, которые получаются из A_1 или A_2 подстановкой на местах, соответствующих единицам, всех возможных комбинаций из 0 и 1. Тем самым определяется множество покрытия, ранг которого на две единицы меньше ранга исходного множества. Если некоторые из полученных минтермов не A-минтермы, то они будут искомыми минтермами.

Пример 10.5. Пусть задана функция $f(x_1, x_2, x_3)=\oplus\ (0,\ 2,\ 4,\ 7)=\bar{x}_1\bar{x}_2\bar{x}_3\oplus$ $\oplus\ \bar{x}_1x_2\bar{x}_3\oplus x_1\bar{x}_2\bar{x}_3\oplus x_1x_2x_3$.

Исходные A-минтермы {000, 010, 100, 111}.

Найти минимальную форму.

Решение.

Этап 1. Нахождение B-минтермов.

Производится попарное сравнение:

а) $\oplus\begin{array}{r}000\\010\\\hline 010\end{array}$ б) $\oplus\begin{array}{r}000\\100\\\hline 100\end{array}$ в) $\oplus\begin{array}{r}000\\111\\\hline 111\end{array}$ г) $\oplus\begin{array}{r}100\\010\\\hline 110\end{array}$ д) $\oplus\begin{array}{r}100\\111\\\hline 011\end{array}$ е) $\oplus\begin{array}{r}010\\111\\\hline 101\end{array}$

В случаях г), д), е) сумма имеет по две единицы. В результате

$$B = \{011,\ 101,\ 110\}.$$

Этап 2. Нахождение первичных импликант:

$$K^0 = \{000,\ 010,\ 011,\ 100,\ 101,\ 110,\ 111\}.$$

А. Нахождение первичных импликант ранга 3:

$$K_1^0 = \{000\}; \quad K_2^0 = \begin{Bmatrix} 010 \\ 100 \end{Bmatrix}; \quad K_3^0 = \begin{Bmatrix} 011 \\ 101 \\ 110 \end{Bmatrix}; \quad K_4^0 = \{111\};$$

а) сравнение K_1^0 и K_2^0:

$$\begin{array}{ll} 000^* & 010^* \\ & 100^* \end{array}$$

$$K_1^1 = \{0x0,\ x00\};$$

б) сравнение K_2^0 и K_3^0:

$$\begin{array}{ll} 010^* & 011^* \\ 100^* & 101^* \\ & 110^* \end{array}$$

$$K_2' = \{01x,\ x10,\ 10x,\ 1x0\};$$

в) сравнение K_3^0 и K_4^0:

$$\begin{array}{ll} 011^* & 111^* \\ 101^* & \\ 110^* & \end{array}$$

$$K_3^1 = \{x11,\ 1x1,\ 11x\}.$$

Первичных импликант ранга 3 нет.

Б. Нахождение первичных импликант ранга 2.

Разобьем все множество импликант на три группы в зависимости от положения независимой координаты x:

$$K_1'' = \begin{Bmatrix} 10x \\ 01x \\ 11x \end{Bmatrix}; \quad K_2'' = \begin{Bmatrix} 0x0 \\ 1x0 \\ 1x1 \end{Bmatrix}; \quad K_3'' = \begin{Bmatrix} x00 \\ x10 \\ x11 \end{Bmatrix};$$

а) сравнение K_1'', K_2'' и K_3'':

$$K_1^2 = \{1xx,\ x1x\};$$

$$K_2^2 = \{xx0,\ 1xx\};$$

$$K_3^2 = \{xx0,\ x1x\}.$$

Таким образом,

$$K^2 = \{1xx,\ x1x,\ xx0\}.$$

Этап 3. Расстановка меток, выбор покрытия (табл. 10.9).

Таблица 10.9

Первичные импликанты	Минтермы						
	A				B		
	000	010	100	111	101	011	110
000	*						
010		*					
100			*				
111				*			
101					*		
011						*	
110							*
$x00$	*		*				
$1x0$			*				
$x10$		*					
$0x0$	*	*					
$11x$				*			*
$01x$		*				*	
$10x$			*		*		
$xx0$	*	*	*				*
$1xx$			*	*	*		*
$x1x$		*		*		*	*

На основании табл. 10.9 получим минимальную форму в виде

$$f(x_1,\ x_2,\ x_3) = x_1x_2 \oplus \bar{x}_3.$$

$$(\oplus,\ \wedge,\ \neg)$$

Для сравнения приведем выражение минимальной формы для СНДФ:

$$f(x_1,\ x_2,\ x_3) = \bar{x}_1\bar{x}_3 + \bar{x}_2\bar{x}_3 + x_1x_2x_3.$$

На рис. 10.4 показано графическое решение для обоих базисов.

Ответ: $f(x_1,\ x_2,\ x_3) = x_1x_2 \oplus \bar{x}_3$.

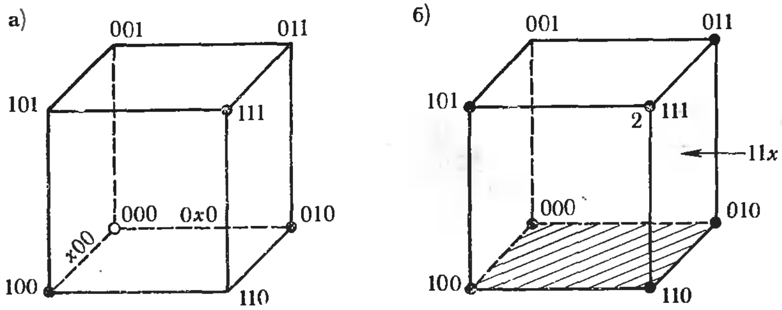


Рис. 10.4. Графическое решение задачи минимизации для примера 10.5: *а* — в случае СНДФ; *б* — для базиса $(\oplus,\ \wedge,\ \neg)$

§ 10.6. Минимизация в базисе Пирса (Вебба)

Функцией Пирса (Вебба) является следующая функция:

$$f(x_1,\ x_2)=x_1\downarrow x_2=\overline{x_1\vee x_2}=\bar{x}_1\bar{x}_2. \qquad (10.4)$$

На основании (10.4) осуществляется переход от дизъюнкций и конъюнкций к функциям Пирса (Вебба):

$$\left.\begin{aligned} &x_1+x_2+\ldots+x_n=\overline{\overline{x_1\downarrow x_2\downarrow\ldots\downarrow x_n}};\\ &x_1x_2\ldots x_n=\bar{x}_1\downarrow\bar{x}_2\downarrow\ldots\downarrow\bar{x}_n. \end{aligned}\right\} \qquad (10.5)$$

Тогда, если логическая функция задана в СКНФ, то на основании (10.5)

$$f(x_1,x_2,\ldots,x_n)=\bigwedge_0\Phi_i=\bigwedge_0(x_1^{\bar{\alpha}_1}\vee x_2^{\bar{\alpha}_2}\vee\ldots\vee x_n^{\bar{\alpha}_n})=$$

$$=\bigwedge_0\overline{\overline{(x_1^{\bar{\alpha}_1}\downarrow x_2^{\bar{\alpha}_2}\downarrow\ldots\downarrow x_n^{\bar{\alpha}_n})}}. \qquad (10.6)$$

Таким образом, из (10.6) можно получить совершенную нормальную форму для функции Пирса (Вебба) в виде

$$f(x_1,\ x_2,\ldots,\ x_n)=\mathop{\downarrow}_0\overline{(x_1^{\bar{\alpha}_1}\downarrow x_2^{\bar{\alpha}_2}\downarrow\ldots\downarrow x_{n-1}^{\bar{\alpha}_{n-1}}\downarrow x_n^{\bar{\alpha}_n}}. \qquad (10.7)$$

Если в (10.7) подставить значение $\bar{x}=x\downarrow x$, то получится совершенная нормальная форма.

Пример 10.6. Функция $f(x_1,\ x_2,\ x_3)$ задана табл. 10.10. Требуется дать аналитическое описание этой функции в виде (10.7).

Таблица 10.10

x_1	x_2	x_3	$f(x_1, x_2, x_3)$	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1	1	0	0	1
0	0	1	0	1	0	1	0
0	1	0	1	1	1	0	1
0	1	1	1	1	1	1	1

Решение. В соответствии с (10.6) получаем

$$f(x_1,x_2,\ x_3)=(x_1\vee x_2\vee\bar{x}_3)(\bar{x}_1\vee x_2\vee\bar{x}_3)=(\overline{\overline{x_1\downarrow x_2\downarrow\bar{x}_3}})(\overline{\overline{\bar{x}_1\downarrow x_2\downarrow\bar{x}_3}}).$$

На основании (10.5)

$$f(x_1,\ x_2,\ x_3)=(\overline{x_1\downarrow x_2\downarrow\bar{x}_3})\downarrow(\overline{\bar{x}_1\downarrow x_2\downarrow\bar{x}_3}).$$

Зная, что $\bar{x}=x\downarrow x$, окончательно получаем совершенную нормальную форму для функции Вебба:

$$f(x_1,\ x_2,\ x_3)=[(x_1\downarrow x_2)\downarrow(x_1\downarrow x_2)\downarrow x_3\downarrow x_3]\downarrow$$
$$\downarrow\{[(x_1\downarrow x_1)\downarrow x_2]\downarrow[(x_1\downarrow x_1)\downarrow x_2]\downarrow x_3\downarrow x_3\}.$$

Если за основу взята СНДФ, то преобразование осуществляется по формулам

$$
f(x_1, x_2, x_3) = \bigvee_1 x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n} = \bigvee_1 \overline{x_1^{\bar{\alpha}_1}} \downarrow \overline{x_2^{\bar{\alpha}_2}} \downarrow \ldots \downarrow \overline{x_n^{\bar{\alpha}_n}} =
$$

$$
= \downarrow_1 \overline{(\overline{x_1^{\bar{\alpha}_1}} \downarrow \overline{x_2^{\bar{\alpha}_2}} \downarrow \ldots \downarrow \overline{x_n^{\bar{\alpha}_n}})}. \tag{10.8}
$$

Пример 10.7. Пусть функция $f(x_1, x_2)$ задана табл. 10.11. Найти минимальную форму.

Таблица 10.11

x_1	x_2	$f(x_1, x_2)$	x_1	x_2	$f(x_1, x_2)$
0	0	1	1	0	0
0	1	1	1	1	1

Решение. Совершенные нормальные формы в базисе Пирса (Вебба) на основе дизъюнкции запишем так:

$$
f(x_1, x_2) = \overline{\overline{(x_1 \downarrow x_2) \downarrow (x_1 \downarrow \bar{x}_2)} \downarrow (\bar{x}_1 \downarrow \bar{x}_2)}.
$$

Преобразуя эту форму, получаем начальное выражение:

$$
f(x_1, x_2) = \overline{\overline{\overline{\bar{x}_1 \bar{x}_2} \downarrow \bar{x}_1 \downarrow x_2} \downarrow x_1 x_2} = \overline{\overline{\bar{x}_1 \bar{x}_2 \bar{x}_1 x_2} \downarrow x_1 x_2} =
$$

$$
= \overline{\overline{\overline{\bar{x}_1 \bar{x}_2}\,\overline{\bar{x}_1 x_2}\,\overline{x_1 x_2}}} = \bar{x}_1 \bar{x}_2 \vee \bar{x}_1 x_2 \vee x_1 x_2.
$$

Ответ: $f(x_1, x_2) = \bar{x}_1 x_2 \vee x_1 x_2$.

Рассмотрим возможность применения метода неопределенных коэффициентов для минимизации в базисе Пирса (Вебба).

Любая логическая функция в базисе Пирса (Вебба) может быть записана в виде

$$
\left.
\begin{aligned}
f(x_1, x_2, x_3) = {} & K_1^0 \bar{x}_1 \downarrow K_2^0 \bar{x}_2 \downarrow K_3^0 \bar{x}_3 \downarrow K_1^1 x_1 \downarrow K_2^1 x_2 \downarrow K_3^1 x_3 \downarrow \\
& \downarrow K_{12}^{00}(\bar{x}_1 \downarrow \bar{x}_2) \downarrow K_{13}^{00}(\bar{x}_1 \downarrow \bar{x}_3) \downarrow K_{23}^{00}(\bar{x}_2 \downarrow \bar{x}_3) \downarrow K_{12}^{10}(x_1 \downarrow \bar{x}_2) \downarrow \\
& \downarrow K_{13}^{10}(x_1 \downarrow \bar{x}_3) \downarrow K_{23}^{10}(x_2 \downarrow \bar{x}_3) \downarrow K_{12}^{01}(\bar{x}_1 \downarrow x_2) \downarrow K_{13}^{01}(\bar{x}_1 \downarrow x_3) \downarrow \\
& \downarrow K_{23}^{01}(\bar{x}_2 \downarrow x_3) \downarrow K_{12}^{11}(x_1 \downarrow x_2) \downarrow K_{13}^{11}(x_1 \downarrow x_3) \downarrow K_{23}^{11}(x_2 \downarrow x_3) \downarrow \\
& \downarrow K_{123}^{000}(\bar{x}_1 \downarrow \bar{x}_2 \downarrow \bar{x}_3) \downarrow K_{123}^{001}(\bar{x}_1 \downarrow \bar{x}_2 \downarrow x_3) \downarrow \\
& \downarrow K_{123}^{010}(\bar{x}_1 \downarrow x_2 \downarrow \bar{x}_3) \downarrow K_{123}^{011}(\bar{x}_1 \downarrow x_2 \downarrow x_3) \downarrow \\
& \downarrow K_{123}^{100}(x_1 \downarrow \bar{x}_2 \downarrow \bar{x}_3) \downarrow K_{123}^{101}(x_1 \downarrow \bar{x}_2 \downarrow x_3) \downarrow \\
& \downarrow K_{123}^{110}(x_1 \downarrow x_2 \downarrow \bar{x}_3) \downarrow K_{123}^{111}(x_1 \downarrow x_2 \downarrow x_3).
\end{aligned}
\right\} \tag{10.9}
$$

Для функции Пирса (Вебба) справедливо:

$$0 \downarrow 0 = 1;$$
$$0 \downarrow 1 = 0;$$
$$1 \downarrow 1 = 0.$$

Следовательно, для системы уравнений, полученной на основе (10.9), можно в единичных строках все коэффициенты приравнять 0. Кроме того, при наличии хотя бы одной 1 в строке функция Вебба становится равной 0.

Таким образом, следует приравнять 0 коэффиценты при термах максимального ранга.

Пример 10.8. Пусть функция задана в виде $f(x_1, x_2, x_3) = \bigvee_1 (0, 2, 4, 7)$ (см. пример 10.6). Найти минимальную форму.

Решение. На основании (10.9) составим систему уравнений в виде табл. 10.12.

Таблица 10.12

$\cancel{K_1^0}$	$\cancel{K_2^0}$	$\cancel{K_3^0}$	$\cancel{K_{12}^{00}}$	$\cancel{K_{13}^{00}}$	$\cancel{K_{23}^{00}}$	$\cancel{K_{123}^{000}}$	1
$\cancel{K_1^0}$	$\cancel{K_2^0}$	$\cancel{K_3^1}$	$\cancel{K_{12}^{00}}$	$\boxed{K_{13}^{01}}$	$\boxed{K_{23}^{01}}$	K_{123}^{001}	0
$\cancel{K_1^0}$	$\cancel{K_2^1}$	$\cancel{K_3^0}$	$\cancel{K_{12}^{01}}$	$\cancel{K_{13}^{00}}$	$\cancel{K_{23}^{10}}$	$\cancel{K_{123}^{010}}$	1
$\cancel{K_1^0}$	$\cancel{K_2^1}$	$\cancel{K_3^1}$	$\cancel{K_{12}^{01}}$	$\boxed{K_{13}^{01}}$	$\cancel{K_{23}^{11}}$	K_{123}^{011}	0
$\cancel{K_1^1}$	$\cancel{K_2^0}$	$\cancel{K_3^0}$	$\cancel{K_{12}^{10}}$	$\cancel{K_{13}^{10}}$	$\cancel{K_{23}^{00}}$	$\cancel{K_{123}^{100}}$	1
$\cancel{K_1^1}$	$\cancel{K_2^0}$	$\cancel{K_3^1}$	$\cancel{K_{12}^{10}}$	$\cancel{K_{13}^{11}}$	$\boxed{K_{23}^{01}}$	K_{123}^{101}	0
$\cancel{K_1^1}$	$\cancel{K_2^1}$	$\cancel{K_3^0}$	$\cancel{K_{12}^{11}}$	$\cancel{K_{13}^{10}}$	$\cancel{K_{23}^{10}}$	$\boxed{K_{123}^{110}}$	0
$\cancel{K_1^1}$	$\cancel{K_2^1}$	$\cancel{K_3^1}$	$\cancel{K_{12}^{11}}$	$\cancel{K_{13}^{11}}$	$\cancel{K_{23}^{11}}$	$\cancel{K_{123}^{111}}$	1

Из оставшихся коэффициентов получаем следующне уравнения:

$$K_{13}^{10} \downarrow K_{23}^{10} \downarrow K_{123}^{110} = 0;$$

$$K_{13}^{10} \downarrow K_{123}^{100} = 0;$$

$$K_{23}^{10} \downarrow K_{123}^{010} = 0;$$

$$K_{123}^{001} = 0.$$

Прнмем $K_{123}^{110} = K_{123}^{100} = K_{123}^{010} = 0$. Тогда

$$f(x_1, x_2, x_3) = (\overline{x_1} \downarrow \overline{x_2} \downarrow x_3) \downarrow (x_2 \downarrow \overline{x_3}) \downarrow (x_1 \downarrow \overline{x_3}),$$

или

$$f(x_1, x_2, x_3) = x_1 x_2 \bar{x}_3 \downarrow \bar{x}_2 x_3 \downarrow \bar{x}_1 x_3 = \overline{x_1 x_2 \bar{x}_3} \wedge \overline{\bar{x}_2 x_3} \wedge \overline{\bar{x}_1 x_3} =$$

$$= (\bar{x}_1 + \bar{x}_2 + x_3)(x_2 + \bar{x}_3)(x_1 + \bar{x}_3) = x_1 x_2 x_3 + \bar{x}_2 \bar{x}_3 + \bar{x}_1 \bar{x}_3.$$

Ответ: $f(x_1, x_2, x_3) = x_1 x_2 x_3 + \bar{x}_2 \bar{x}_3 + \bar{x}_1 \bar{x}_3$.

Задание для самоконтроля

1. Найти минимальную форму, используя метод неопределенных коэффициентов для функции $f(x_1, x_2, x_3) = \bar{x}_1 \bar{x}_2 \bar{x}_3 + x_1 \bar{x}_2 \bar{x}_3 + \bar{x}_1 x_2 x_3 + x_1 x_2 \bar{x}_3 + \bar{x}_1 \bar{x}_2 x_3$.

2. Найти минимальную конъюнктивную нормальную форму с помощью карт Карно для функции $f(x_1, x_2, x_3, x_4) = \bigwedge\limits_0 (0, 1, 4, 5, 11, 12, 2, 6, 10, 14)$.

3. Найти минимальную форму с помощью геометрического представления для функцин $f(x_1, x_2, x_3, x_4) = \bigvee\limits_1 (0, 1, 2, 3, 4, 5, 6, 10, 12, 13, 14)$.

4. Найти минимальное представление в базисе $(\oplus, \wedge, \neg)$ методом неопределенных коэффициентов для функции $f(x_1, x_2, x_3) = \bigvee\limits_1 (0, 2, 4, 7)$.

5. Используя метод Квайна — Мак-Класкн, найти миннмальную форму в базнсе $(\oplus, \wedge, \neg)$ для функции $f = \bigoplus\limits_1 (0, 2, 3, 7)$.

6. Найти миннмальное представление в базисе Вебба для функции $f(x_1, x_2, x_3) = \bigvee\limits_1 (0, 1, 5, 6)$.

7. Найти простые импликанты для функции $f(x_1, x_2, x_3, x_4) = \bigvee\limits_1 (0, 1, 2, 6, 7, 9, 11, 14, 15)$.

8. Составить минимизнрующие карты для функции $f(x_1, x_2, x_3, x_4) = \bigwedge\limits_0 (1, 2, 3, 4, 6, 7, 10, 11, 12, 15)$.

11 ГЛАВА

АНАЛИЗ И СИНТЕЗ ЭЛЕКТРОННЫХ СХЕМ

§ 11.1. Логические операторы электронных схем

По зависимости выходного сигнала от входного все электронные схемы можно условно разбить на:

схемы первого рода, включающие **комбинационные схемы** — *схемы, выходной сигнал в которых зависит только от состояния входов (наличия входных сигналов) в каждый момент времени*;

схемы второго рода, включающие **накапливающие схемы (элементы с памятью)** — *схемы, выходной сигнал в которых зависит как от входных сигналов, так и от состояния схемы в предыдущие моменты времени.*

По количеству входов и выходов схемы бывают:

с одним входом и одним выходом,

с несколькими входами и одним выходом,

с одним входом и несколькими выходами,

с несколькими входами и выходами.

Практически любая ЭВМ состоит из комбинации схем первого и второго родов разной сложности.

Рассмотрим некоторые конкретные примеры.

На рис. 11.1, *а* показана схема, выполненная на транзисторе.

Схема работает следующим образом. В интервале времени от 0 до t_1 (рис. 11.1, *б*) на входе действует почти нулевое напряжение. За счет делителя R_1—R_2 и источника $E_б$ транзистор $ПТ_1$ закрыт и на выходе напряжение равно $—E_к$. В момент t_1 происходит изменение напряжения на входе (действует u_1), что изменяет потенциал

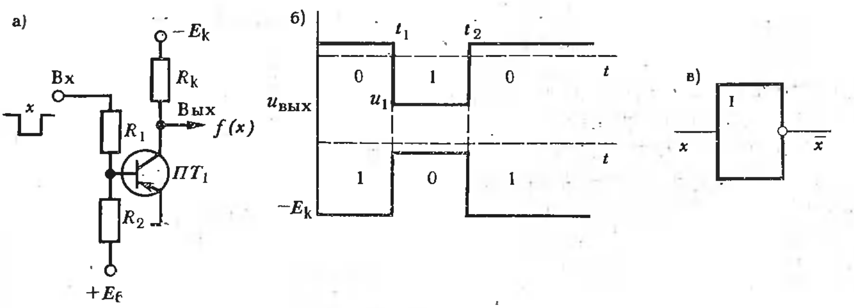


Рис. 11.1. Анализ работы элементарной электронной схемы

на базе транзистора $ПТ_1$ до такой степени, что транзистор открывается, через резистор $R_к$ течет ток, который изменяет напряжение на выходе: оно становится близким нулю. Так продолжается до момента t_2, когда напряжение на входе снова изменяется, вызывая соответствующее изменение на выходе. На рис. 11.1, *б* показана временна́я диаграмма изменений напряжений на входе и выходе схемы. Напряжения на выходе и входе принимают два значения (высокий уровень — 0, низкий уровень — 1). Тогда логически работу рассмотренной схемы можно описать с помощью функции НЕ (рис. 11.1, *в*), что подтверждается следующим:

Момент временн	$0—t_1$	$t_1—t_2$	$t_2—t_3$
Вход x	1	0	1
Выход $f(x)$	0	1	0

Логический оператор схемы — *элементарная логическая функция, с помощью которой описывается работа схемы.*

Таким образом, рассмотренная выше схема описывается функцией НЕ и называется **инвертором** (рис. 11.1, *в*).

На рис. 11.2, *а* показана схема дизъюнктора, описываемая логическим оператором ИЛИ (рис. 11.2, *в*). Временна́я диаграмма работы этой схемы представлена на рис. 11.2, *б*.

На рис. 11.3, *а* показана комбинация электронных схем дизъюнктора и инвертора, выполненная на транзисторах, а ее логический оператор ИЛИ-НЕ — на рис. 11.3, *в*. Временна́я диаграмма работы этой схемы дана на рис. 11.3, *б*.

Аналогичным образом проводится анализ работы схемы конъюнктора (рис. 11.4, *а*). Временна́я диаграмма работы схемы и ее логический оператор показаны соответственно на рис. 11.4, *б, в*.

На рис. 11.5, *а* показана комбинированная схема конъюнктора-инвертора, а на рис. 11.5, *б, в* — временна́я диаграмма работы схемы и ее логический оператор. Основной особенностью этих схем является то, что они полностью идентичны схемам, показанным на рис. 11.2, *а* и 11.3, *а*. Соответственно этим лишний раз подтверждается справедливость законов де Моргана, которые описывают двойственный характер наборов логических функций И-НЕ и ИЛИ-НЕ.

§ 11.2. Задачи анализа и синтеза электронных схем

На основании вышеизложенного можно прийти к заключению, что различные электронные схемы или их комбинации на логическом уровне могут быть описаны с помощью логических операторов. Такое операторное описание электронных схем позволяет абстрагироваться от физической природы конкретных электронных элементов и осуществлять их анализ. При этом оказывается, что для анализа совсем не обязательно иметь саму схему. Для того чтобы получить значение функции на выходе какой-либо схемы, достаточно записать эту зависимость в виде логических операторов, связанных между собой в соответствии с выполняемой функцией.

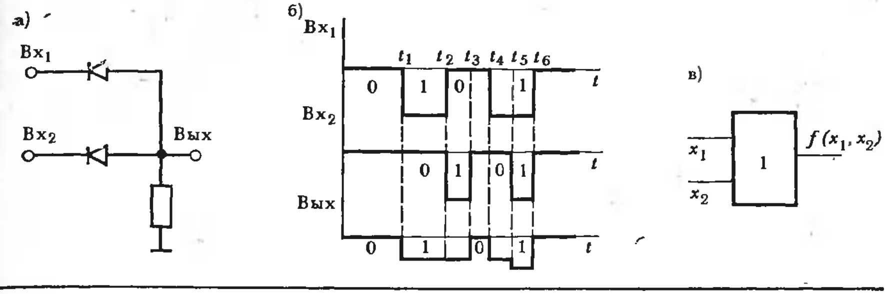


Рис. 11.2. Анализ схемы днзьюнктора

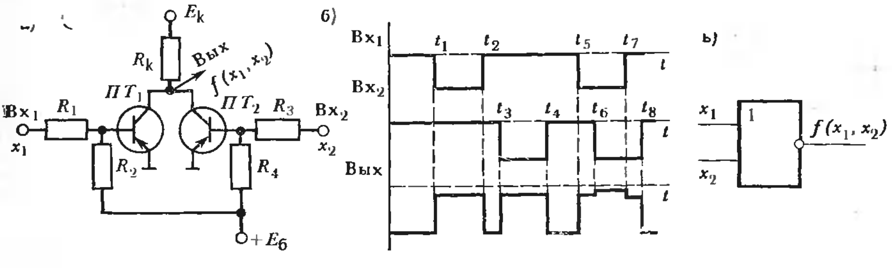
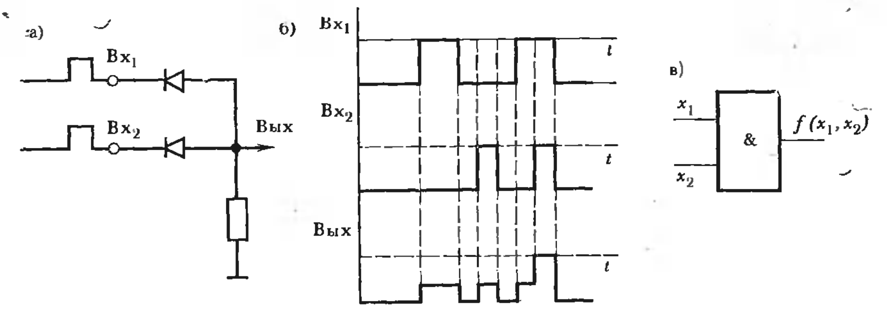


Рис. 11.3. Анализ схемы днзъюнктора с инверсией



Рнс. 11.4. Аналнз схемы конъюнктора

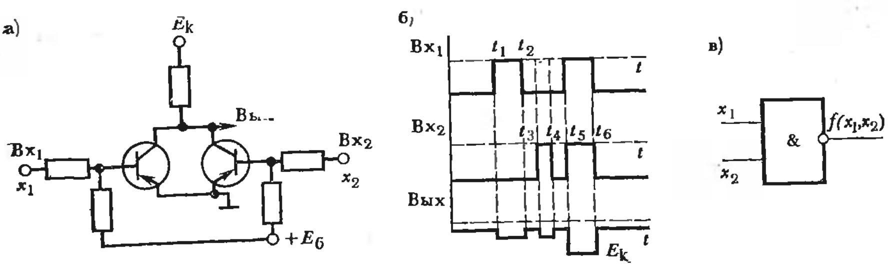


Рис. 11.5. Анализ схемы конъюнктора с нверсней

Задача анализа электронных схем с помощью аппарата алгебры логики может быть сформулирована как задача нахождения логической функции, описывающей работу заданной схемы.

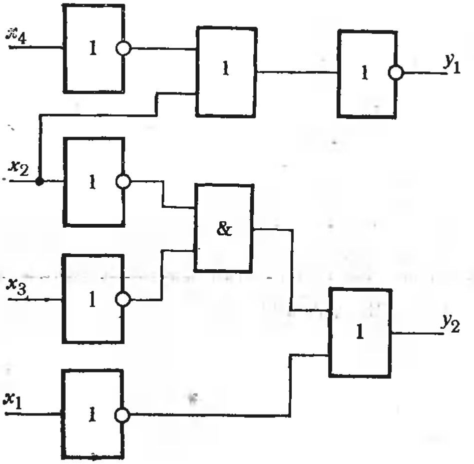


Рис. 11.6. Логическая схема с двумя выходами

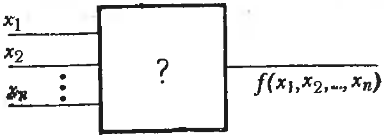


Рис. 11.7. Условное изображение задачи синтеза

При этом исходят из того, что каждому функциональному элементу электронной схемы можно поставить в соответствие логический оператор. Этим самым устанавливается однозначное соответствие между элементами схемы и ее математическим описанием.

Анализ электронной схемы проводится в два этапа:

1) из принципиальной схемы удаляются все несущественные вспомогательные элементы, которые не влияют на логику работы схемы;

2) через логические операторы выражают все элементы, получая логическое уравнение, которое является моделью функции, выполняемой заданной схемой. Проводится анализ этой функции с целью устранения лишних частей.

Например, схема, представленная на рис. 11.6, может быть описана логическими выражениями $y_1=\overline{x_2+x_4}$; $y_2=\bar{x}_1+\bar{x}_2\bar{x}_3$. Однако с точки зрения инженерного проектирования чаще приходится решать обратную задачу (рис. 11.7), называемую задачей синтеза электронных схем.

Задачу синтеза электронных схем можно сформулировать следующим образом: при заданных входных переменных и известной выходной функции необходимо спроектировать логическое устройство, которое реализует эту функцию (при этом могут быть наложены дополнительные ограничения либо в виде системы логических элементов, которые используются, либо в виде требований по количеству логических операторов).

Следовательно, в результате решения задачи синтеза возникает логическая схема, воспроизводящая заданную функцию.

Как правило, решая задачи анализа и синтеза, используют полные базисы функций. При этом каждую логическую функцию, входящую в базис, сопоставляют с некоторым физическим элементом. Значит, логическую схему можно заменить структурной схемой, состоящей из физических элементов.

Таким образом удается соединить математическую задачу синтеза логической схемы с инженерной задачей проектирования электронной схемы. При разработке электронной схемы за основные критерии принимают: минимум аппаратуры, минимум типов применяемых элементов, максимум надежности.

С точки зрения математической логики задача синтеза решается при обеспечении минимального числа логических операторов, минимального количества типов логических операторов. В результате можно сформулировать последовательные этапы решения задачи синтеза электронной схемы:

1) составление математического описания (система логических уравнений), адекватно отображающего процессы, происходящие в схеме;

2) анализ логических уравнений и получение минимальной формы для каждой из них в заданном базисе;

3) переход от логических уравнений к логической (структурной) схеме посредством применения логических операторов.

§ 11.3. Синтез электронных схем с одним выходом

Схемы с одним выходом и несколькими входами относятся к наиболее простым схемам. Основная сложность при синтезе этих схем состоит в том, чтобы найти выражение для выходной функции в заданном базисе.

Пример 11.1. Синтезировать схему в базисе «НЕ-импликация», если функция имеет вид $(x_1, x_2, x_3) = \overline{x_1} \to (x_1\overline{x_2} + x_3)$.

Решение. Перейдем от смешанной системы логических функций к системе «НЕ-импликация» на основе правил перехода:

$$x_1 \to x_2 = \overline{x_1} + x_2 = \overline{x_1\overline{x_2}},$$

$$x_1x_2 = \overline{x_1 \to x_2}. \tag{11.1}$$

Получаем $\varphi(x_1, x_2, x_3) = \overline{x_1} \to (\overline{x_1 \to x_2} + x_3) = \overline{x_1} \to ((x_1 \to x_2) \to x_3)$.

Функция $\varphi(x_1, x_2, x_3)$ может быть реализована на основе логических операторов «НЕ» и «импликация» (рис. 11.8).

Ответ: логическая схема на рис. 11.8.

Задача синтеза, как правило, имеет множество решений в зависимости от выбранной системы логических элементов. Однако для любой заданной функции алгебры логики почти всегда можно синтезировать схему, соответствующую этой функции. Получение оптимальной схемы с точки зрения минимального количества логических связок требует нахождения минимальной формы для функции алгебры логики.

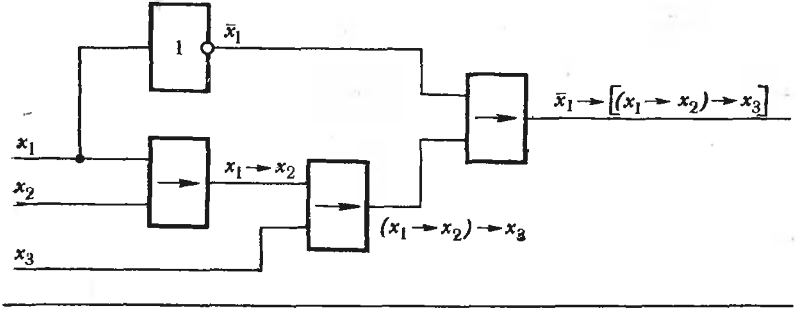


Рис. 11.8. Логическая схема, выполненная на элементах импликации и инверсии (к примеру 11.1)

Некоторые более сложные схемы, имеющие несколько выходов, могут быть сведены в частном случае к набору схем с одним выходом. В таких случаях синтез осуществляется путем декомпозиции для каждой выделяемой схемы. Рассмотрим в качестве примера синтез одноразрядного двоичного сумматора методом декомпозиции.

Пример 11.2. Синтезировать схему, заданную табл. 11.1 в базисе И—ИЛИ—НЕ.

Таблица 11.1

a_i	b_i	$П_{i-1}$	c_i	$П_i$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Здесь a_i, b_i — слагаемые i-го разряда, операндов a и b; c_i — сумма слагаемых i-го разряда; $П_i$ и $П_{i-1}$ — соответственно переносы из i-го и (i—1)-го разрядов.

Решение. Синтезируемую схему можно рассматривать как схему, состоящую из двух частей: 1) схемы для получения поразрядной суммы c_i (полусумматор); 2) схемы для получения переноса $П_i$.

На основе теоремы (9.20) запишем СНДФ для функции c_i и $П_i$.

Используя минимизирующие карты Карно, получаем минимальные формы для каждой из функций $П_i$ и c_i (рис. 11.9, *а*, *б*):

$$\left.\begin{aligned} c_i &= П_{i-1}(\overline{a_i}\overline{b_i} + a_i b_i) + \overline{П}_{i-1}(\overline{a}_i b_i + a_i\overline{b_i}; \\ П_i &= П_{i-1}(a_i + b_i) + a_i b_i. \end{aligned}\right\} \qquad (11.2)$$

Функции (11.2) могут быть реализованы схемой, представленной на рис. 11.10.

Ответ: логическая схема на рис. 11.10.

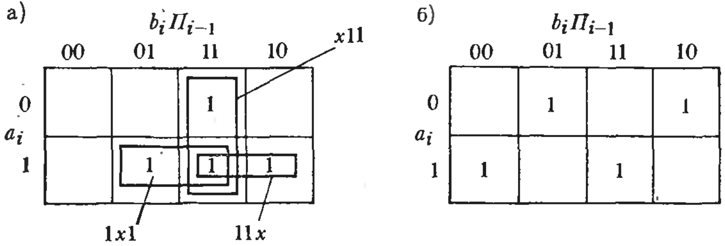


Рис. 11.9. Минимизация функций Π_i и c_i (к примеру 11.2)

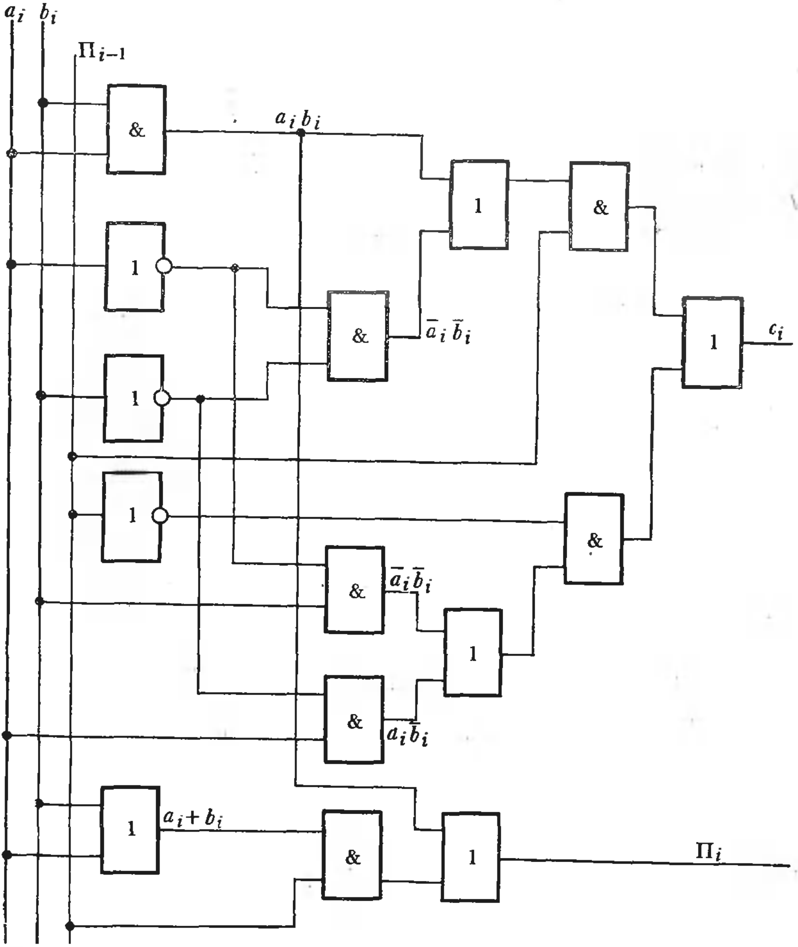


Рис. 11.10. Логическая схема двоичного сумматора (к примеру 11.2)

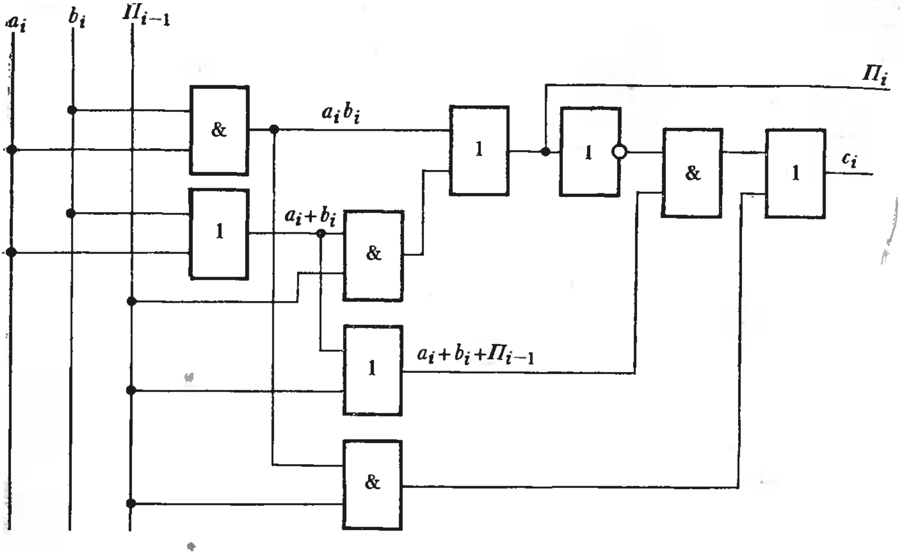


Рис. 11.11. Минимальная схема двоичного сумматора

Однако путь решения задачи, который показан в примере 11.2, не всегда дает минимальное решение. Например, исходное выражение для c_i может быть записано иначе. Из табл. 11.2 видно, что поразрядная сумма c_i равна 1 тогда, когда одно из слагаемых a_i, b_i или Π_{i-1} равно 1, а остальные слагаемые равны 0 и при этом $\Pi_i=0$ $(\overline{\Pi}_i=1)$ или когда все три слагаемых равны 1. Поэтому

$$c_i=(a_i+b_i+\Pi_{i-1})\,\Pi_i+a_ib_i\Pi_{i-1}.$$

Таким образом, окончательное выражение имеет вид

$$\left.\begin{aligned} c_i&=(a_i+b_i+\Pi_{i-1})\,\overline{\Pi}_i+a_ib_i\Pi_{i-1};\\ \Pi_i&=\Pi_{i-1}(a_i+b_i)+a_ib_i. \end{aligned}\right\} \qquad (11.3)$$

Функции (11.3) могут быть реализованы логической схемой, представленной на рис. 11.11.

§ 11.4. Синтез электронных схем с несколькими выходами

Задача синтеза схемы с n входами и k выходами отличается тем от задачи синтеза k схем с n входами и одним выходом, что при решении необходимо исключить дублирование в k схемах синтезируемых функций.

Примером схем с несколькими входами и несколькими выходами служит схема дешифратора (рис. 11.12). Принцип работы дешифратора прост: при заданном наборе входных сигналов на выходе возбуждается одна шина или несколько шин в соответствии с заданной зависимостью. В табл. 11.2 представлена работа дешифратора от трех переменных, у которого возбуждается только один из выходов.

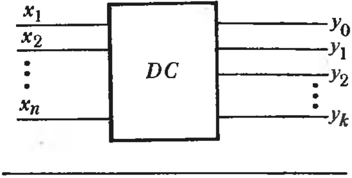


Рис. 11.12. Дешифратор

Таблица 11.2

Входы			Выходы							
x_1	x_2	x_3	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Синтез такой схемы может быть осуществлен, если рассматривать раздельно каждую выходную функцию:

$$y_0=\bar{x}_1\bar{x}_2\bar{x}_3; \quad y_4=x_1\bar{x}_2\bar{x}_3;$$

$$y_1=\bar{x}_1\bar{x}_2x_3; \quad y_5=x_1\bar{x}_2x_3;$$

$$y_2=\bar{x}_1x_2\bar{x}_3; \quad y_6=x_1x_2\bar{x}_3;$$

$$y_3=\bar{x}_1x_2x_3; \quad y_7=x_1x_2x_3.$$

Реализация этих выражений в виде конъюнкторов дает возможность создать логическую схему дешифратора (рис. 11.13).

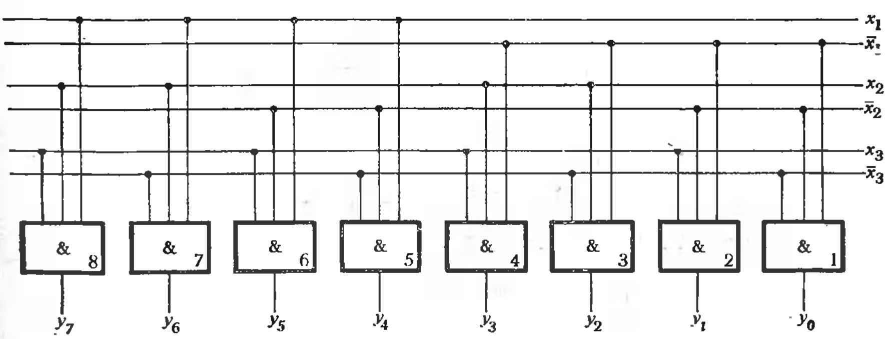


Рис. 11.13. Логическая схема дешифратора

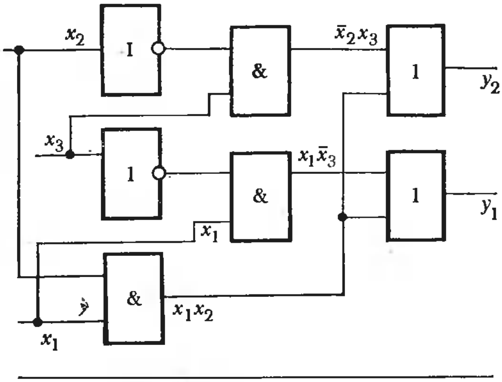


Рис. 11.14. Логическая схема (к примеру 11.3)

Однако несложно убедиться в том, что такой подход к построению схем не дает оптимального решения из-за указанных выше особенностей схем с несколькими выходами.

Рассмотрим два наиболее простых метода синтеза таких схем.

Первый метод (классический) основан на выделении простых импликант заданной системы функций подобно тому, как это делается в методе минимизации Квайна — Мак-Класки, и затем покрытии каждой заданной функции этими простыми импликантами. Далее синтез схемы идет на уровне простых импликант. При этом требуется:

1. Найти простые импликанты заданной системы функций.
2. Выразить каждую заданную функцию через простые импликанты.
3. Синтезировать схему, включающую только эти импликанты и связи между ними.

Пример 11.3. Синтезировать схему, функции на выходах которой имеют вид

$$y_1 = f_1(x_1,\ x_2,\ x_3) = x_1\bar{x}_2\bar{x}_3 + x_1x_2\bar{x}_3 + x_1x_2x_3;$$

$$y_2 = f_2(x_1,\ x_2,\ x_3) = x_1x_2 + \bar{x}_2x_3.$$

В качестве базиса взяты функции И, ИЛИ, НЕ.

Решение. Найдем простые импликанты, разбив множество y_1 на три группы в соответствии с количеством единиц в каждой группе:

$$K_1^0 = \{100\};\quad K_2^0 = \{110\};\quad K_3^0 = \{111\}.$$

В результате сравнения групп получим

$$K^1 = \{1x0,\ 11x\} = \{x_1\bar{x}_3,\ x_1x_2\}.$$

Простых импликант ранга 3 нет.
Простые импликанты ранга 2

$$K^1 = \{x_1x_2,\ \bar{x}_2x_3,\ x_1\bar{x}_3\}.$$

Окончательный вид выходных функций будет:

$$y_1 = \underline{x_1x_2} + x_1\bar{x}_3;\quad y_2 = \underline{x_1x_2} + \bar{x}_2x_3.$$

В полученных выражениях подчеркнут член, являющийся общим для обоих уравнеиий, что позволяет упростить окончательный вариант схемы, представленный иа рис. 11.14.

Ответ: логическая схема на рис. 11.14.

Второй метод (метод каскадов) основан на теореме разложения логической функции по k переменным и выглядит следующим образом:

$$\left.\begin{aligned}
&f(x_1,\ x_2, \ldots,\ x_n) = x_n f_1 \vee \overline{x}_n f_2;\\
&f_1(x_1,\ x_2, \ldots,\ x_{n-1}) = x_{n-1} f_{11} \vee \overline{x}_{n-1} f_{12};\\
&f_2(x_1,\ x_2, \ldots,\ x_{n-1}) = x_{n-1} f_{21} \vee \overline{x}_{n-1} f_{22};\\
&f_{11}(x_1,\ x_2, \ldots,\ x_{n-2}) = x_{n-2} f_{111} \vee \overline{x}_{n-2} f_{112};\\
&f_{22}(x_1,\ x_2, \ldots,\ x_{n-2}) = x_{n-2} f_{221} \vee \overline{x}_{n-2} f_{222};\\
&f_{21}(x_1,\ x_2, \ldots,\ x_{n-2}) = x_{n-2} f_{211} \vee \overline{x}_{n-2} f_{212};\\
&f_{12}(x_1,\ x_2, \ldots,\ x_{n-2}) = x_{n-2} f_{121} \vee \overline{x}_{n-2} f_{122};\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots
\end{aligned}\right\} \qquad (11.4)$$

Процесс разложения происходит до тех пор, пока не будут получены функции $f_{ijk\ldots l}$, зависящие только от двух аргументов. Далее синтезируется схема, соответствующая системе уравнений минимального ранга.

Пример 11.4. Синтезировать схему в базисе И—ИЛИ—НЕ, выходные функции которой заданы в виде уравнений:

$$\varphi_1 = x_1x_2x_3 \vee x_1x_2\overline{x}_3 \vee \overline{x}_1x_2\overline{x}_3;$$

$$\varphi_2 = \overline{x}_1\overline{x}_2x_3 \vee x_1\overline{x}_2\overline{x}_3 \vee \overline{x}_1\overline{x}_2\overline{x}_3;$$

$$\varphi_3 = x_1x_2\overline{x}_3 \vee \overline{x}_1x_2\overline{x}_3 \vee \overline{x}_1\overline{x}_2x_3.$$

Решение. Применим разложения (11.4) к заданным функциям:

$$\varphi_1 = \underbrace{x_1x_2x_3}_{f_{11}} \vee \overline{x}_3\,\underbrace{(x_1x_2 \vee \overline{x}_1x_2)}_{f_{12}=x_2};$$

$$\varphi_2 = \underbrace{\overline{x}_1\overline{x}_2}_{f_{21}}x_3 \vee \underbrace{(x_1\overline{x}_2 \vee \overline{x}_1\overline{x}_2)}_{f_{22}=\overline{x}_2}\,\overline{x}_3;$$

$$\varphi_3 = \overline{x}_3\,\underbrace{(x_1x_2 \vee \overline{x}_1x_2 \vee \overline{x}_1\overline{x}_2)}_{f_{32}};$$

$$f_{32} = x_1x_2 \vee \overline{x}_1x_2 \vee \overline{x}_1\overline{x}_2 = \overline{x}_1 \vee x_2.$$

После упрощений выходные уравнения можно записать так:

$$\varphi_1 = x_1x_2x_3 \vee x_2\overline{x}_3;$$

$$\varphi_2 = \overline{x}_1\overline{x}_2x_3 \vee \overline{x}_2\overline{x}_3;$$

$$\varphi_3 = \overline{x}_3\,(x_2 \vee \overline{x}_1).$$

На рис. 11.15 изображена соответствующая этим уравнениям логическая схема.

Ответ: логическая схема на рис. 11.15.

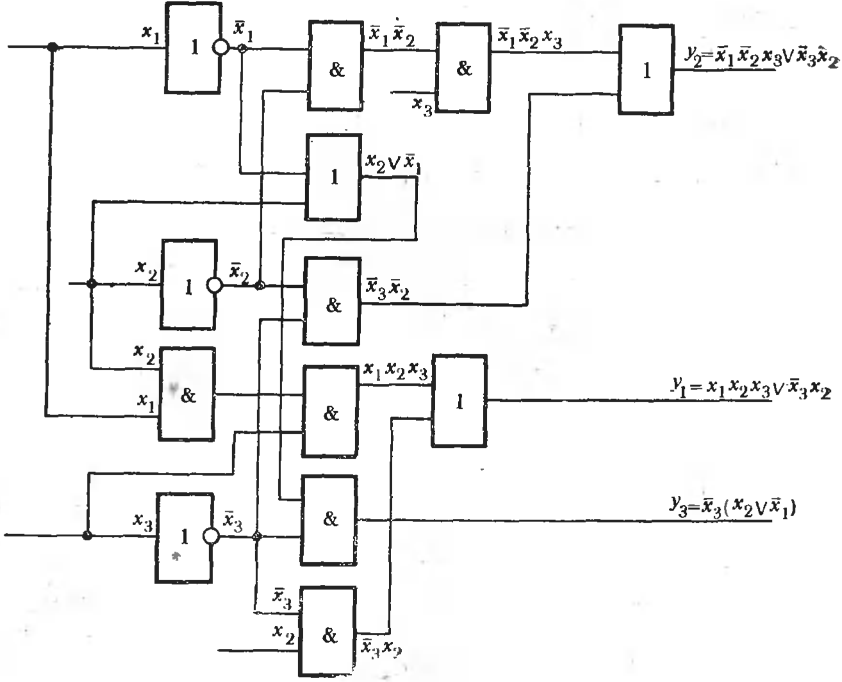


Рис. 11.15. Логическая схема (к примеру 11.4)

§ 11.5. Неполностью определенные функции алгебры логики

Неполностью определенная логическая функция n переменных — *функция, заданная на числе наборов, меньших 2^n.*

Следовательно, неполностью определенная функция алгебры логики содержит множество наборов, на которых она не определена. Встречаются такие функции достаточно часто. Если количество неопределенных наборов m, то путем различных доопределений можно получить 2^m различных функций.

Примечание. В дальнейшем набор, на котором функция $f(x_1, \ldots, x_n)$ не определена, будем отмечать звездочкой (*).

Для наборов, на которых функция не определена, значение логической функции может быть произвольным. Все зависит от некоторых других условий. Например, если взять какой-то десятичный код, пусть Д$_1$, то выходная функция определена на 10 из 16 возможных наборов. Остальные являются запрещенными. Значит, оставшиеся шесть наборов должны быть доопределены, так как иначе невозможно будет использовать аналитическое представление в виде совершенных нормальных форм (СНФ). Это доопределение очень важно, так как от него зависят действительные результаты.

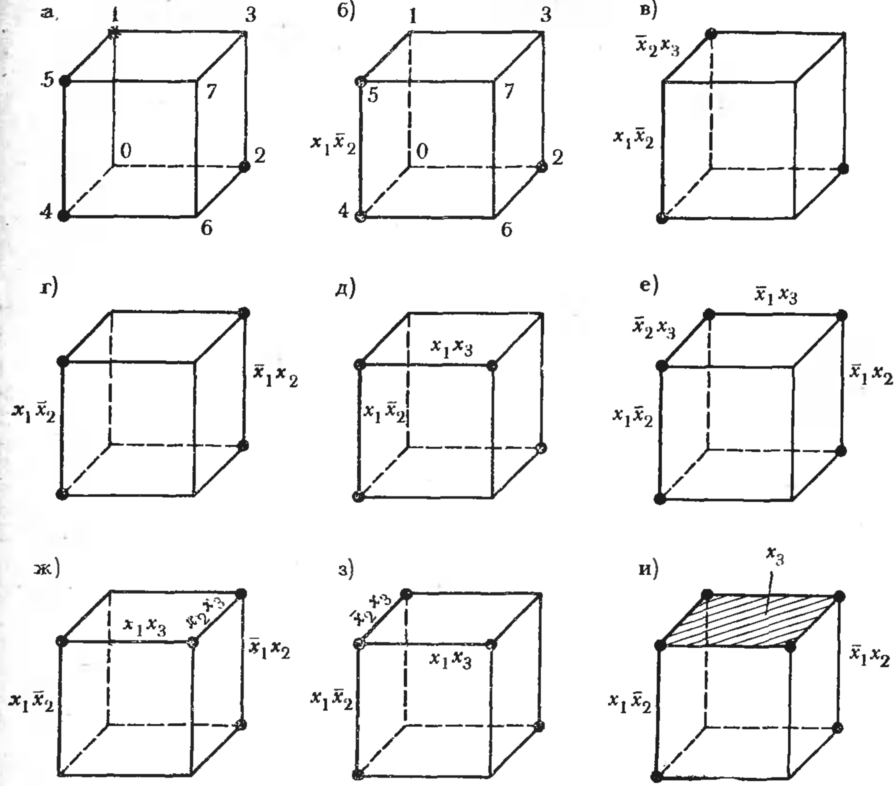


Рис. 11.16. Графическое решение задачи доопределения функций (к примеру 11.5):

а — графическое представление заданной функции; *б* — функция доопределена на отмеченных наборах нулями; *в* — *д* — функция доопределена на одном из отмеченных наборов единицей, на остальных — нулями; *е, ж* — функция доопределена на двух из отмеченных наборов единицами, на оставшемся наборе — нулем; *и* — функция доопределена на отмеченных наборах единицами

Пример 11.5. Доопределить следующую функцию:

$$f\ (x_1,\ x_2,\ x_3) = \bigvee_1 (1^*,\ 2,\ 3^*,\ 4,\ 5,\ 7^*).$$

Решение. Рассмотрим, какие функции получаются, если отмеченные наборы доопределить.

На рис. 11.16 представлено множество определенных функций, соответствующих заданной функции $f(x_1,\ x_2,\ x_3)$. Если функцию $f(x_1,\ x_2,\ x_3)$ доопределить на отмеченных наборах нулями, то получится в результате минимизации новая функция:

$$f_б\,(x_1,\ x_2,\ x_3) = x_1\overline{x}_2 \vee \overline{x}_1 x_2 \overline{x}_3 \text{ (рис. 11.16, } б).$$

Если на отмеченных наборах задать значения для f, равные единице, то

$$f_и\,(x_1,\ x_2,\ x_3) = x_3 \vee x_1\overline{x}_2 \vee \overline{x}_1 x_2 \text{ (рис. 11.16, } и).$$

демонстрируют разные варианты доопределения. Минимальное решение для заданной функции:

Другие части рис. 11.16 демонстрируют разные варианты доопределения. Минимальное решение для заданной функции:

$$f_г(x_1,\ x_2,\ x_3) = x_1\bar{x}_2 \vee \bar{x}_1 x_2$$

(рис. 11.16, *г*).

Ответ: $f_г(x_1, x_2, x_3) = x_1\bar{x}_2 \vee \bar{x}_1 x_2$.

Пример 11.5 показывает, что доопределение функции существенно влияет на конечный результат минимизации.

При доопределении функций можно руководствоваться следующим правилом: *минимальная дизъюнктивная нормальная форма неполностью определенной функции $f(x_1, x_2, \ldots, x_n)$ получается как дизъюнкция наиболее коротких по числу букв импликант функции $\varphi_1(x_1, x_2, \ldots, x_n)$, принимающей значение, равное 1, на всех наборах, где функция не определена, которые в совокупности покрывают все импликанты в совершенной нормальной форме для функции $\varphi_0(x_1, \ldots, x_n)$, принимающей значение, равное 0, на всех наборах, где f не определена.*

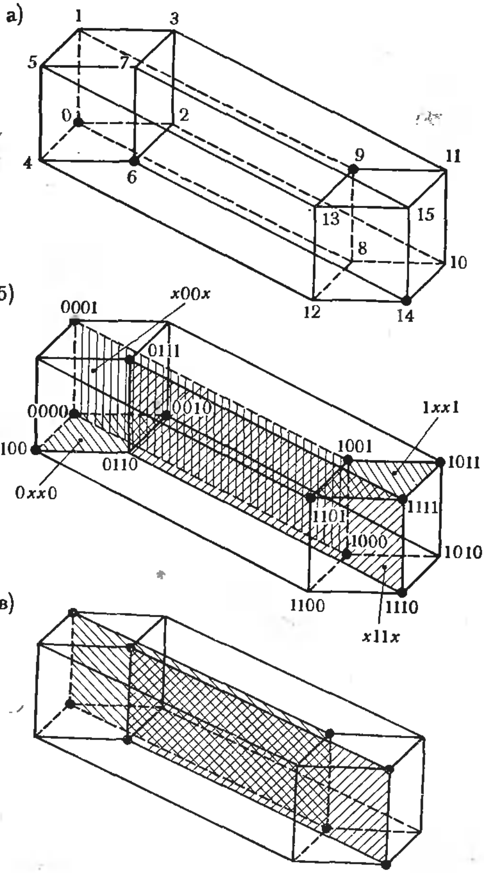


Рис. 11.17. Графический метод минимизации (к примеру 11.6):

а — графическое представление заданной функции; *б* — графическое представление функции $\varphi_1(x_1, x_2, x_3, x_4)$, доопределенной единицами на отмеченных наборах; *в* — оптимальное доопределение функции $f(x_1, x_2, x_3, x_4)$

В самом деле, функция $\varphi_1(x_1, \ldots, x_n)$ содержит все простые импликанты, которые могут встретиться в i-м покрытии f, а φ_0 содержит минимальное число простых импликант. Поэтому надо выбрать из этого набора те импликанты из φ_1, которые оптимально покрывают φ_0.

Пример 11.6. Найти минимальную форму для функции четырех переменных:

$$f(x_1, x_2, x_3, x_4) = (0,\ 1^*,\ 2^*,\ 4^*,\ 6,\ 7^*,\ 8^*,\ 9,\ 11^*,\ 13^*,\ 14,\ 15^*).$$

Решение. На рис. 11.17, *а*, представлена функция f, отображаемая также табл. 11.3. Звездочкой на рисунке отмечены неопределенные значения.

СНФ для функции $\varphi_0(x_1, x_2, x_3, x_4)$ может быть получена из табл. 11.3 при замене символов * на 0:

$$\varphi_0(x_1, x_2, x_3, x_4) = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1 x_2 x_3 \bar{x}_4 \vee x_1 \bar{x}_2 \bar{x}_3 x_4 \vee x_1 x_2 x_3 \bar{x}_4.$$

Таблица 11.3

x_1	x_2	x_3	x_4	f	x_1	x_2	x_3	x_4	f
0	0	0	0	1	1	0	0	0	*
0	0	0	1	*	1	0	0	1	1
0	0	1	0	*	1	0	1	0	0
0	0	1	1	0	1	0	1	1	*
0	1	0	0	*	1	1	0	0	0
0	1	0	1	0	1	1	0	1	*
0	1	1	0	1	1	1	1	0	1
0	1	1	1	*	1	1	1	1	*

Соответственно функция $\varphi_1(x_1, x_2, x_3, x_4)$ определяется, если в табл. 11.3 символ (*) заменить на 1. Этот случай изображен на рис. 11.17, *б*. Минимизируя показанную функцию, получаем:

$$\varphi_1(x_1, x_2, x_3, x_4) = \bar{x}_1\bar{x}_4 \vee \bar{x}_2\bar{x}_3 \vee x_2 x_3 \vee x_1 x_4.$$

Оптимальное доопределение функции, соответствующее минимальному покрытию, может быть найдено по методу Квайна (табл. 11.4).

Таблица 11.4

φ_1	$\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\bar{x}_1 x_2 x_3 \bar{x}_4$	$x_1\bar{x}_2\bar{x}_3 x_4$	$x_1 x_2 x_3 \bar{x}_4$
$\bar{x}_1\bar{x}_4$	$\vee$	$\vee$		
$\bar{x}_2\bar{x}_3$	$\vee$		$\vee$	
$x_2 x_3$		$\vee$		$\vee$
$x_1 x_4$			$\vee$	

Таким образом, минимальное покрытие

$$f(x_1, x_2, x_3, x_4) = x_2 x_3 \vee \bar{x}_2\bar{x}_3 \text{ (рис. 11.17, } в).$$

Ответ: $f(x_1, x_2, x_3, x_4) = x_2 x_3 \vee \bar{x}_2\bar{x}_3$.

§ 11.6. Синтез электронных схем с использованием свойств неполностью определенных функций

Пусть необходимо построить функциональную схему для системы, заданной уравнениями

$$y_1 = f_1(x_1, x_2, \ldots, x_n);$$
$$y_2 = f_2(x_1, x_2, \ldots, x_n). \quad (11.5)$$

Предположим, что функция y_1 уже построена. Тогда новая функция

$$y_2^*=f_2^*(y_1, x_1, x_2, \ldots, x_n).$$

Функция y_2^* — неполностью определенная функция и в таблице значений функции только половина значений определена. При этом y_1 используется в качестве $(n+1)$-го аргумента, хотя фактически f_2^* зависит только от n аргументов.

Последовательность этапов при синтезировании схемы будет такой.

Система уравнений вида (11.5) может быть задана либо аналитически, либо в виде таблицы. Теперь нужно построить таблицу значений функции y_2^*, если y_1 является аргументом, полагая, что значение y_2^* не определено для тех строк таблицы, которые отсутствуют в таблице значений y_2. Функция y_2^* записывается в СНДФ:

$$y_2^*=f_2^*(x_1, x_2, x_3, y_1)=\bigvee_1 x_1^{\alpha_1} x_2^{\alpha_2} x_3^{\alpha_3} y_1^{\sigma_1}, \tag{11.6}$$

где

$$\sigma_1=\begin{cases} 1, & \text{если} \quad y_1=1; \\ 0, & \text{если} \quad y_1=0. \end{cases}$$

Положим, что на неопределенных значениях $f_2^*=0$, тем самым находим функцию $\varphi_0(x_1, x_2, x_3, x_4)$ (см. § 11.5). Далее положим $y_2=1$ и запишем СНДФ функции $f_2^*(1)$. После этого найдем минимальную форму для $f_2^*(1)$ каким-либо методом (рис. 11.17, *б*). Затем составим таблицу покрытия $f_2^*(0)$ и выделим минимальное покрытие (рис. 11.17, *в*). В завершение определим оптимальный набор неопределенных значений f_2^*. При этом f_2^* становится определенной на всех наборах. В результате находится соответствующая МДНФ, на основе которой синтезируется схема.

Пример 11.7. Синтезировать одноразрядный двоичный сумматор с использованием свойств неполностью определенных функций.

Решение. Значения Π_i и c_i даны в табл. 11.1. Пусть схема, реализующая функцию Π_i, уже синтезирована. Тогда построим таблицу функции c_i^* (табл. 11.5).

Таблица 11.5

a_i	b_i	Π_{i-1}	Π_i	c_i^*	N	a_i	b_i	Π_{i-1}	Π_i	c_i^*	N
0	0	0	0	0	0	1	0	0	0	1	8
0	0	0	1	*	1	1	0	0	1	*	9
0	0	1	0	1	2	1	0	1	0	*	10
0	0	1	1	*	3	1	0	1	1	0	11
0	1	0	0	1	4	1	1	0	0	*	12
0	1	0	1	*	5	1	1	0	1	0	13
0	1	1	0	*	6	1	1	1	0	*	14
0	1	1	1	0	7	1	1	1	1	1	15

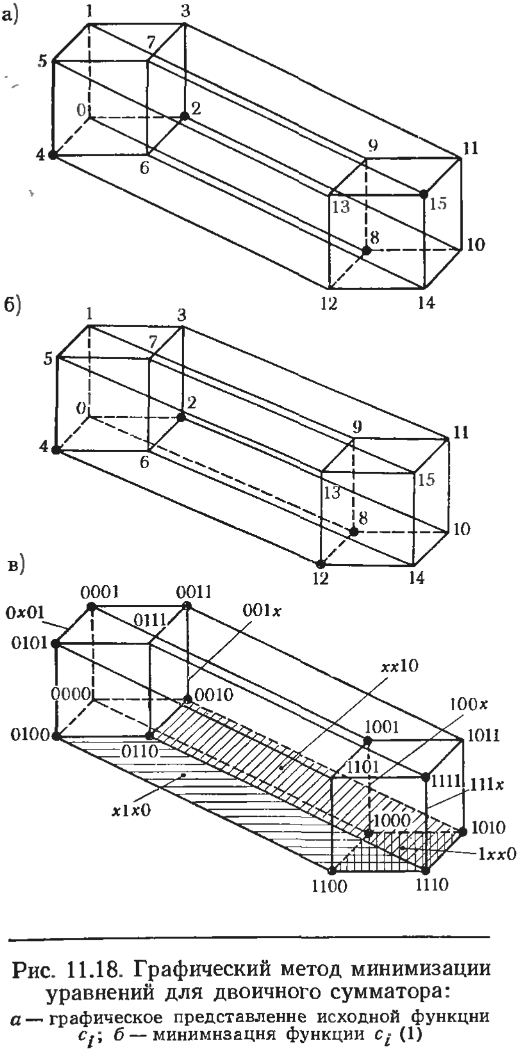


Рис. 11.18. Графический метод минимизации уравнений для двоичного сумматора:
а — графическое представленне исходной функцни c_i; *б* — минимнзацня функции c_i (1)

Введем обозначения $П_i = y_1$ и $c_i = y_2$. Тогда

$$c_i^* = y_2^* = f_2^*(a_i,\ b_i,\ П_{i-1},\ П_i).$$

На рис. 11.18, *а* показано графическое представление функции c_i^*. Доопределение функции c_i^*, записанное в СНДФ, будет

$$c_i^*(1) = \bigvee_1 (1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 8,\ 9,\ 10, 12,\ 14,\ 15) \text{ (рис. 11.18, } б);$$

$$c_i^*(0) = \bigvee_1 (2,\ 4,\ 8,\ 15).$$

Графическим методом находим минимальную форму для c_i^* (рис. 11.18, б);

$$c_i^*(1) = a_i\overline{\Pi}_i + b_i\overline{\Pi}_i + \Pi_{i-1}\overline{\Pi}_i + a_ib_i\Pi_{i-1} + a_i\overline{b}_i\overline{\Pi}_{i-1} + \overline{a}_i\overline{\Pi}_{i-1}\Pi_i + \overline{a}_i\overline{b}_i\Pi_{i-1}.$$

Составим таблицу покрытия для c_i^* (0) (табл. 11.6).

Таблица 11.6

Первичные импликанты	Исходные термы			
	$\overline{a}_i\overline{b}_i\Pi_{i-1}\overline{\Pi}_i$	$\overline{a}_ib_i\overline{\Pi}_{i-1}\overline{\Pi}_i$	$a_i\overline{b}_i\overline{\Pi}_{i-1}\overline{\Pi}_i$	$a_ib_i\Pi_{i-1}\Pi_i$
$a_i\overline{\Pi}_i$			V	
$b_i\overline{\Pi}_i$		V		
$\Pi_{i-1}\overline{\Pi}_i$	V			
$\overline{a}_i\overline{\Pi}_{i-1}\Pi_i$				
$\overline{a}_i\overline{b}_i\Pi_{i-1}$	V			
$a_ib_i\Pi_{i-1}$				V
$a_i\overline{b}_i\overline{\Pi}_{i-1}$			V	

Минимальное покрытие

$$a_i\overline{\Pi}_i;\quad b_i\overline{\Pi}_i;\quad \Pi_{i-1}\overline{\Pi}_i \quad \text{и} \quad a_ib_i\Pi_{i-1}.$$

Из табл. 11.7 находим оптимальное доопределение функции c_i^* (0), в соответствии с которым получаем конечный вид уравнений:

$$c_i = (a_i + b_i + \Pi_{i-1})\,\overline{\Pi}_i + a_ib_i\Pi_{i-1};$$

$$\Pi_i = (a_i + b_i)\,\Pi_{i-1} + a_ib_i.$$

§ 11.7. Временные булевы функции

Ранее были рассмотрены способы анализа и синтеза схем первого рода (комбинационных), которые невозможно применить для схем второго рода (схем с памятью). Основная особенность схем с памятью состоит в том, что их работа зависит от времени. Следовательно, в число переменных, от которых зависит выходная функция схемы с памятью, должно входить время t. Но время t не является двоичной переменной. Поэтому вводится понятие **автоматного времени**, принимающего дискретные целочисленные значения 0, 1, 2 и т. д. Это означает, что работа схемы с памятью распадается на ряд интервалов, в течение которых автоматное время условно принимает постоянное значение.

Временна́я булева функция (ВБФ) — *логическая функция* $y = \varphi(x_1, x_2, \ldots, x_n, t)$, *принимающая значение* $\{0, 1\}$ *при* $0 \leqslant t \leqslant s-1$, *где* s — *количество интервалов автоматного времени.*

Можно утверждать, что число различных ВБФ равно $2^{s\cdot 2^n}$. В самом деле, если время принимает s значений, т. е. $t = 0, 1, 2, \ldots, s-1$, и каждому интервалу времени соответствует 2^n различных двоичных наборов, то всего будет $s \cdot 2^n$ различных наборов. Следовательно, общее число ВБФ равно $2^{s\cdot 2^n}$.

Любая временна́я булева функция может быть представлена в виде

$$y=\varphi(x_1, x_2, \dots, x_n, t)=\varphi_0\tau_0 \vee \varphi_1\tau_1 \vee \dots \vee \varphi_{s-1}\tau_{s-1}, \quad (11.7)$$

где φ_i — конъюнктивный или дизъюнктивный терм от переменных $(x_1, x_2, ..., x_n)$; τ_i — вспомогательная функция, принимающая значение $\tau_i=\{0, 1\}$ в моменты времени t_i.

Фо́рма представления временных логических функций (11.7) позволяет применить к функциям y все методы упрощения и минимизации, рассмотренные ранее.

Пример 11.8. Преобразовать функцию (табл. 11.7) в вид (11.7).

Таблица 11.7

x_1	x_2	t	$\varphi(x_1, x_2, t)$	x_1	x_2	t	$\varphi(x_1, x_2, t)$
0	0	0	0	1	0	1	1
0	1	0	0	1	1	1	0
1	0	0	1	0	0	2	0
1	1	0	0	0	1	2	0
0	0	1	0	1	0	2	1
0	1	1	1	1	1	2	1

Решение. Функцию $y=\varphi(x_1, x_2, t)$ представляем совокупностью трех логических функций $\varphi_0(x_1, x_2)$; $\varphi_1(x_1, x_2)$; $\varphi_2(x_1, x_2)$, которые для табл. 11.7 имеют вид

$$\varphi_0(x_1, x_2) = x_1\overline{x_2};$$

$$\varphi_1(x_1, x_2) = \overline{x}_1x_2 \vee x_1\overline{x}_2;$$

$$\varphi_2(x_1, x_2) = x_1\overline{x}_2 \vee x_1x_2 = x_1.$$

На основании (11.7) записываем окончательный вид временно́й логической функции:

$$y = x_1\overline{x}_2\tau_0 \vee (\overline{x}_1x_2 \vee x_1\overline{x}_2)\,\tau_1 \vee x_1\tau_2. \quad (11.8)$$

Ответ: см. формулу (11.8).

Очевидно, что разложение (11.7) можно применить только к периодическим временным функциям. Переход к схеме от логического выражения вида (11.7) можно осуществить следующим образом.

Предположим, что на выходах некоторой схемы (дешифратора) в моменты времени t_i появляются сигналы:

если $t_1=0$, то на выходе *1* сигнал $\tau_0=1$ при $\tau_1=0$, $\tau_2=0$,

если $t_2=1$, то на выходе *2* сигнал $\tau_1=1$ при $\tau_0=0$, $\tau_2=0$,

если $t_3=2$, то на выходе *3* сигнал $\tau_2=1$ при $\tau_0=0$, $\tau_1=0$.

(рис. 11.19)

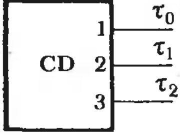


Рис. 11.19. Разновидность дешифратора

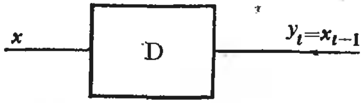


Рис. 11.20. Схема задержки

Для каждой функции φ_i строим соответствующую логическую схему, не зависящую от переменной t. После этого все схемы соединяем между собой в соответствии с (11.7).

Рекуррентная булева функция (РБФ) — *логическая функция, зависящая как от текущих значений* (x_{t_i}) *входных переменных, так и от предшествующих значений самой функции* (y_{t-i}).

Полная аналитическая запись такой функции имеет вид

$$y_t = \varphi(x_{t_1}, x_{t_2}, \ldots, x_{t_n}, y_{t-1}, y_{t-2}, \ldots, y_{t-k}), \tag{11.9}$$

$$y_t = \{0, 1\} \quad \text{при} \quad t > 0,$$

где x_{t_i} — текущие значения входных переменных, y_j — значения выходных функций в момент времени $j = t, t-1, t-2, \ldots$

Чтобы представить необходимость рекуррентных булевых функций, рассмотрим некоторый физический элемент, работа которого описывается так:

t	0	1	2	...	t_{i-1}	t_i
x	x_0	x_1	x_2	...	x_{i-1}	x_i
$y_t = f(x, t)$ . . .	0	x_0	x_1	...	x_{i-2}	x_{i-1}

Следовательно, $y_{t+1} = x_t$. Отсюда значение выходного сигнала в момент времени $t+1$ равно значению входного сигнала в момент времени t. Такой элемент называют **задержкой**; $D(t)$ — его логический оператор (рис. 11.20).

Рассмотрим схему, имеющую цепь обратной связи с включенной в нее схемой задержки (рис. 11.21). Предположим, что в качестве схемы с функцией $f(x, y)$ взята логическая схема ИЛИ. Тогда в совокупности эта схема работает так, как показано на временной диаграмме (рис. 11.21, *б*), т. е. $f(x, y) = x_{t+1} \vee y_t$.

В схеме рис. 11.21 выходной сигнал зависит как от входного сигнала в данный момент времени, так и от выходного сигнала в предшествующий момент времени. В самом общем случае при наличии n входов и k цепей обратной связи, в которых осуществляется равная задержка, такие схемы могут быть описаны с помощью рекуррентных временны́х логических функций.

Следовательно, любая реккурентная булева функция может быть реализована с помощью набора логических операторов функциональных элементов, представляющих обычные функции алгебры логики, и операторов схем задержки.

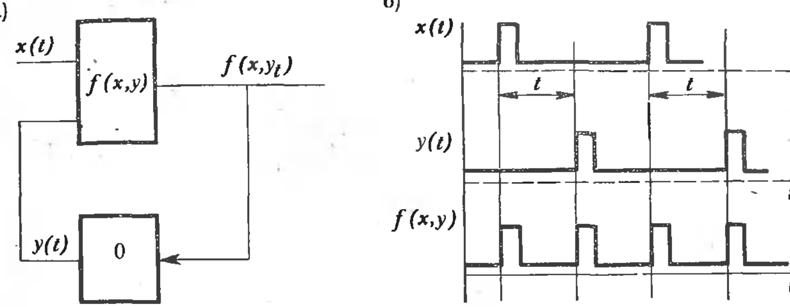


Рис. 11.21. Схема с обратной связью

§ 11.8. Последовательные автоматы

Рассмотрим частный случай рекуррентной временной логической функции. Допустим, что на вход функциональной схемы входные переменные не подаются, а поступают только сигналы по цепям обратных связей, т. е. все $x_i=0$ в момент времени $t\,(t\neq 0)$. Тогда рекуррентная булева функция (РБФ) примет вид

$$y_i(t+1)=f_i(y_{1t},\ y_{1(t-1)},\ \dots,\ y_{1(t-t_1)},\ y_{2t},\ \dots,\ \dots,\ y_{2(t-t_2)},\ \dots,\ y_{mt},\ \dots,\ y_{m(t-t_m)}),\quad i=1,\ 2,\ \dots,\ m. \tag{11.10}$$

Для таких функций всегда необходимо задавать нулевые значения (т. е. при $t=0$). Предположим, что обратная связь осуществляется только на один такт времени. Тогда

$$y_{i(t+1)}=f_i(y_{1t},\ y_{2t},\ \dots,\ y_{mt}),\quad i=1,\ 2,\ \dots,\ m. \tag{11.11}$$

На рис. 11.22 представлена логическая схема последовательного автомата, описываемая системой уравнений вида (11.11).

Последовательные автоматы — *схемы, описываемые системой уравнений вида* (11.11) *при заданных начальных условиях.*

Рассмотрим последовательный автомат, имеющий три выхода и начальные значения $y_{1,0}=1$, $y_{2,0}=1$, $y_{3,0}=1$. Для такой схемы в любой момент времени на входе может действовать одна из восьми возможных комбинаций входных сигналов,

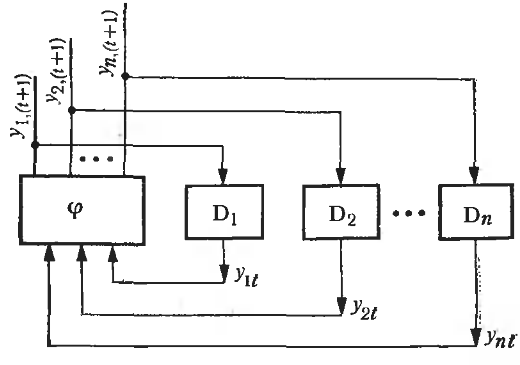


Рис. 11.22. Логическая схема последовательного автомата

При этом для каждого набора входных переменных можно определить значения выходных переменных, предполагая, что в этот момент цепи обратной связи условно разорваны. Предположим, что получилась следующая таблица состояний автомата (табл. 11.8).

Таблица 11.8

y_{1t}	y_{2t}	y_{3t}	$y_{1(t+1)}$	$y_{2(t+1)}$	$y_{3(t+1)}$
0	0	0	0	1	1
0	0	1	1	0	0
0	1	0	1	1	0
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	0	0	1
1	1	0	0	1	0
1	1	1	1	0	1

На основании данных этой таблицы записывается СНДФ для всех выходных параметров:

$$\left.\begin{aligned} y_{1(t+1)} &= \bar{y}_{1t}\bar{y}_{2t}y_{3t} \vee \bar{y}_{1t}y_{2t}\bar{y}_{3t} \vee y_{1t}\bar{y}_{2t}\bar{y}_{3t} \vee y_{1t}y_{2t}y_{3t};\\ y_{2(t+1)} &= \bar{y}_{1t}\bar{y}_{2t}\bar{y}_{3t} \vee \bar{y}_{1t}y_{2t}\bar{y}_{3t} \vee y_{1t}\bar{y}_{2t}\bar{y}_{3t} \vee y_{1t}y_{2t}\bar{y}_{3t};\\ y_{3(t+1)} &= \bar{y}_{1t}\bar{y}_{2t}\bar{y}_{3t} \vee y_{1t}\bar{y}_{2t}\bar{y}_{3t} \vee y_{1t}\bar{y}_{2t}y_{3t} \vee y_{1t}y_{2t}y_{3t}. \end{aligned}\right\} \quad (11.12)$$

Уравнения (11.12) справедливы для любого момента времени.

Используя методы минимизации, можно получить следующую минимальную форму:

$$\left.\begin{aligned} y_{1(t+1)} &= \bar{y}_{1t}\bar{y}_{2t}y_{3t} \vee \bar{y}_{1t}y_{2t}\bar{y}_{3t} \vee y_{1t}\bar{y}_{2t}\bar{y}_{3t} \vee y_{1t}y_{2t}y_{3t};\\ y_{2(t+1)} &= \bar{y}_{3t};\\ y_{3(t+1)} &= \bar{y}_{2t}\bar{y}_{3t} \vee y_{1t}y_{3t}. \end{aligned}\right\} \quad (11.13)$$

Теперь рассмотрим работу этого автомата последовательно в моменты времени начиная с $t=0$. По условию, при $t=0$ на входе действуют: $y_{10}=1$, $y_{20}=1$, $y_{30}=1$. Из табл. 11.9 видно, что такому набору входных переменных соответствуют следующие значения на выходах: $y_{1,1}=1$; $y_{2,1}=0$; $y_{3,1}=1$. В свою очередь, для момента времени $t=1$ этот набор переменных действует уже на входе автомата и вызывает соответствующие значения на выходах: $y_{1,2}=0$, $y_{2,2}=0$, $y_{3,2}=1$ и т. д.

Таким образом, можно провести анализ работы автомата в любой фиксированный момент времени и получить полную таблицу состояний (табл. 11.9).

Следовательно, состояния автомата повторяются с определенным периодом (в данном случае он равен четырем). Периодичность

Таблица 11.9

y_{1t}	y_{2t}	y_{3t}	$y_{1(t+1)}$	$y_{2(t+1)}$	$y_{3(t+1)}$
1	1	1	1	0	1
1	0	1	0	0	1
0	0	1	1	0	0
1	0	0	1	1	1
1	1	1	1	0	1
1	0	1	0	0	1
0	0	0	1	0	0
1	0	0	1	1	1
1	1	1	1	0	1
1	0	1	0	0	1
0	0	1	1	0	0
1	0	0	1	1	1
1	1	1	1	0	1

и значение выходов зависят также от начального состояния автомата, в чем нетрудно убедиться, если задавать различные начальные условия.

Формулы (11. 13), полученные в результате минимизации, позволяют построить схему автомата в выбранном базисе функций (И, ИЛИ, НЕ).

Таблицу состояний автомата можно также представить в виде **диаграммы переходов,** представляющей собой круг, разделенный на k равных частей, из которых каждая часть представляет состояние входов автомата (соответственно выходов). Стрелками внутри круга соединяются входные наборы с соответствующими им выходными наборами.

§ 11.9. Анализ электронных схем, описываемых вырожденными рекуррентными булевыми функциями

На основании ранее изложенного можно представить некоторую логическую схему, которая задается следующей системой функций:

$$\left.\begin{aligned} &y_{1(t+1)}=\varphi_1(y_{1t}, y_{2t}, \ldots, y_{st}, y_{1(t-1)}, \ldots, y_{1(t-k)}, \ldots \\ &\ldots, y_{s(t-1)}, \ldots, y_{s(t-k)}); \\ &y_{2(t+1)}=\varphi_2(y_{1t}, y_{2t}, \ldots, y_{st}, y_{1(t-1)}, \ldots, y_{1(t-k)}, \ldots \\ &\ldots, y_{s(t-1)}, \ldots, y_{s(t-k)}); \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &y_{s(t+1)}=\varphi_s(y_{1t}, y_{2t}, \ldots, y_{st}, y_{1(t-1)}, \ldots, y_{1(t-k)}, \ldots \\ &\ldots, y_{s(t-1)}, \ldots, y_{s(t-k)}). \end{aligned}\right\} \quad (11.14)$$

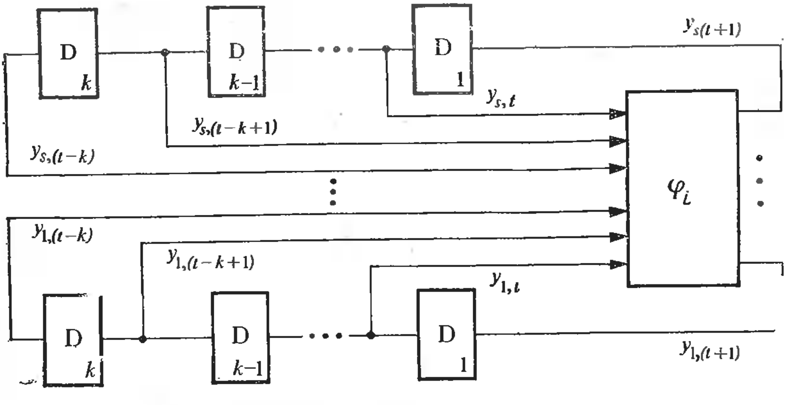
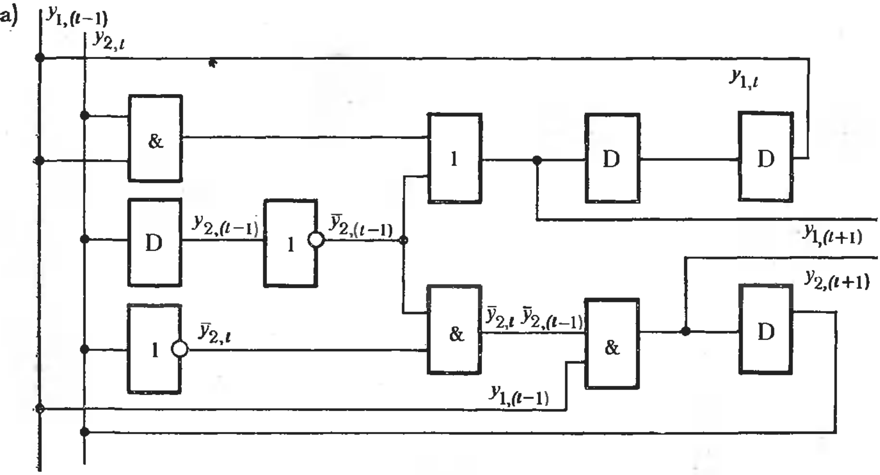


Рис. 11.23. Схема последовательного автомата, описываемого вырожденными рекуррентными функциями



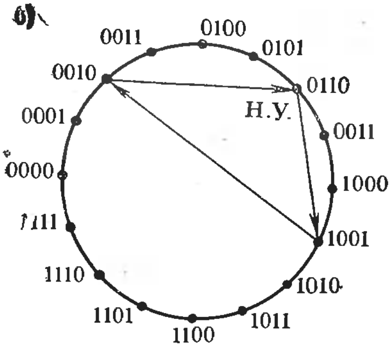


Рис. 11.24. Логическая схема (*а*) и диаграмма переходов (*б*) последовательного автомата (к примеру 11.9)

При этом задаются начальные условия $y_{10}, y_{20}, \ldots, y_{s0}$, где $y_{s0}=\{0, 1\}$.

Возможный вариант такой схемы изображен на рис. 11.23.

Если задана электронная схема последовательного автомата, то анализ ее должен осуществляться в такой последовательности:

1. Написать систему уравнений вида (11.14), описывающих работу схемы, и задать начальные условия.

2. Определить таблицу состояний или диаграмму переходов последовательного автомата на основе полученной системы уравнений.

Пример 11.9. Провести анализ электронной схемы последовательного автомата (рис. 11.24, *а*), описываемой системой уравнений

$$y_{1(t+1)} = y_{2,t}\, y_{2,(t-1)} \vee \overline{y}_{2,(t-1)};$$

$$y_{2(t+1)} = y_{1,(t-1)}\, \overline{y}_{2,t}\, \overline{y}_{2,(t-1)}.$$

Решение. Определяем таблицу состояний (табл. 11.10) в общем случае, так как начальные условия не заданы.

Таблица 11.10

y_{1t}	y_{2t}	$y_{1(t-1)}$	$y_{2(t-1)}$	$y_{1(t+1)}$	$y_{2(t+1)}$	y_{1t}	y_{2t}
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	1	1	0
0	0	1	1	0	0	0	0
0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1
0	1	1	0	1	0	0	1
0	1	1	1	1	0	0	1
1	0	0	0	1	0	1	0
1	0	0	1	0	0	1	0
1	0	1	0	1	1	1	0
1	0	1	1	0	0	1	0
1	1	0	0	1	0	1	1
1	1	0	1	0	0	1	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	1

Теперь в зависимости от различных обстоятельств можно задать начальные условия и получить конкретную таблицу состояний (табл. 11.11).

Пусть $y_{1,0}=0$, $y_{2,0}=1$, $y_{1,(-1)}=1$, $y_{2,(-1)}=0$.

Получился период повторения, равный 3.

Диаграмма переходов для данного случая представлена на рис. 11.24, *б*.

Таблица 11.11

y_{1t}	y_{2t}	$y_{1,(t-1)}$	$y_{2,(t-1)}$	$y_{1,(t+1)}$	$y_{2,(t+1)}$	$y_{1,t}$	$y_{2,t}$
0	1	1	0	1	0	0	1
1	0	0	1	0	0	1	0
0	0	1	0	0	1	1	0
0	1	1	0	1	0	0	1
1	0	0	1	0	0	1	0
0	0	1	0	0	1	1	0
0	1	1	0	1	0	0	1
1	0	0	1	0	0	1	0
0	0	1	0	0	1	1	0

§ 11.10. Анализ и синтез электронных схем с помощью рекуррентных булевых функций

Рассмотрим некоторую обобщенную схему последовательного автомата (рис. 11.25), которая описывается следующей системой уравнений:

$$\left.\begin{aligned}
y_{1(t+1)} &= \varphi_1(x_{1,(t+1)}, \ldots, x_{n,(t+1)};\ y_{1,t}, \ldots, y_{st}, \ldots, y_{1,(t-k)}, \ldots, y_{s(t-k)});\\
y_{2(t+1)} &= \varphi_2(x_{1,(t+1)}, \ldots, x_{n,(t+1)};\ y_{1,t}, \ldots, y_{st}, \ldots, y_{1,(t-k)}, \ldots, y_{s(t-k)});\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
y_{s(t+1)} &= \varphi_s(x_{1(t+1)}, \ldots, x_{n(t+1)};\ y_{1t}, \ldots, y_{st}, \ldots, y_{1(t-k)}, \ldots, y_{s(t-k)});\\
z_{1(t+1)} &= \Psi_1(x_{1(t+1)}, \ldots, x_{n(t+1)};\ y_{1t}, \ldots, y_{st}, \ldots, y_{1(t-k)}, \ldots, y_{s(t-k)});\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
z_{s(t+1)} &= \Psi_s(x_{1(t+1)}, \ldots, x_{n(t+1)};\ y_{1t}, \ldots, y_{st}, \ldots, y_{1(t-k)}, \ldots, y_{s(t-k)}),
\end{aligned}\right\} \tag{11.15}$$

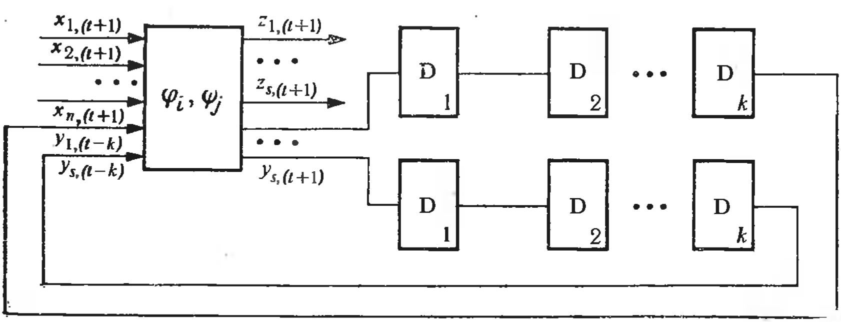


Рис. 11.25. Обобщенная схема последовательного автомата

где $x_{i,t}$ — входные переменные; $y_{i,t}$ — внутренние состояния схемы в момент времени t; $z_{i,t}$ — выходные переменные в момент времени t.

В этой схеме выходные переменные зависят как от входных переменных, так и от внутренних состояний.

Предположим, что конкретная схема описывается следующей таблицей состояний (табл. 11.12).

Таблица 11.12

$x_{1,(t+1)}$	$x_{2,(t+1)}$	y_t	$y_{(t+1)}$	$\overline{y}_{(t+1)}$
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

Выходная функция для этой схемы описывается уравнением

$$y_{(t+1)} = \overline{x}_{1,(t+1)} \overline{x}_{2(t+1)} y_t \vee x_{1(t+1)} \overline{x}_{2(t+1)} y_t \vee$$

$$\vee x_{1(t+1)} \overline{x}_{2(t+1)} y_t \vee x_{1(t+1)} x_{2(t+1)} \overline{y}_t,$$

которое после преобразования и минимизации имеет вид

$$y_{(t+1)} = \overline{x}_{2(t+1)} y_t \vee x_{1(t+1)} \overline{y}_t. \qquad (11.16)$$

Триггеры с двумя входами — *схемы, выходная функция для которых имеет вид* (11.16). Они обладают двумя устойчивыми внутренними состояниями. Переход триггера из одного состояния в другое осуществляется при выполнении следующих условий:

если $y_t = 0$ при $x_{1,(t+1)} = 1$, то $x_{2,(t+1)}$ не влияет;

если $y_t = 1$ при $x_{2(t+1)} = 1$, то $x_{1(t+1)}$ не влияет.

Анализируя табл. 11.12, можно сделать вывод, что при $x_{1(t+1)} = x_{2(t+1)} = 1$ переход триггера из одного состояния в другое зависит от значения y_t. Если $y_t = 0$, то $y_{(t+1)} = 1$, если $y_t = 1$, то $y_{(t+1)} = 0$. Отсюда следует, что не обязательно подавать на входы триггера две независимых переменных, а достаточно подавать один сигнал на два входа одновременно.

Выходная функция для этой схемы описывается уравнением

$$y_{(t+1)} = \overline{x}_{t+1} y_t \vee x_{t+1} \overline{y}_t. \qquad (11.17)$$

Триггеры со счетным входом — *схемы, выходная функция для которых имеет вид* (11.17).

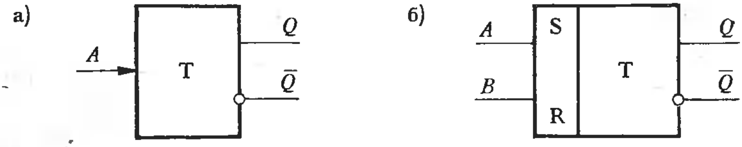


Рис. 11.26. Условное обозначение триггеров типа Т (*а*) и типа TRS (*б*)

Таблица состояний этой схемы имеет следующий вид (табл. 11.13).

Таблица 11.13

x_{t+1}	y_t	$y_{(t+1)}$	$\bar{y}_{(t+1)}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

На рис. 11.26, *а, б* соответственно представлены условные обозначения триггера с двумя входами и триггера со счетным входом. Таким образом, рассмотренные триггерные схемы являются частным случаем обобщенной схемы вида (11.15).

Если ввести следующие обозначения:

$$X_{(t+1)} = \{x_{1,(t+1)},\ x_{2,(t+1)}, \ldots,\ x_{n,(t+1)}\};$$

$$Y_t = \{y_{1,t},\ y_{2,t}, \ldots,\ y_{m,t}\};$$

$$Y_{(t+1)} = \{y_{1,(t+1)},\ y_{2,(t+1)}, \ldots,\ y_{m,(t+1)}\};$$

$$Z_{(t+1)} = \{z_{1,(t+1)},\ z_{2,(t+1)}, \ldots,\ z_{k,(t+1)}\},$$

то система уравнений (11.15) будет представлена в виде

$$\left.\begin{aligned} Y_{(t+1)} &= F(X_{(t+1)},\ Y_t); \\ Z_{(t+1)} &= \Phi(X_{(t+1)},\ Y_t). \end{aligned}\right\} \qquad (11.18)$$

Уравнения (11.18) называют **каноническими уравнениями.**

Пример 11.10. Уравнение $y_{1(t+1)} = x_{1(t+1)} y_{1t} \vee x_{2(t+1)}$ преобразовать так, чтобы его можно было реализовать с помощью триггера и других логических функций.

Решение. Так как уравнение триггера $y_{t+1} = \bar{x}_{2(t+1)} y_t \vee x_{1(t+1)} \bar{y}_t$, то, введя обозначения $\bar{x}_{2(t+1)} = y^1_{1(t+1)}$ и $x_{1(t+1)} = y^2_{1(t+1)}$ и зная, что $y^1_{1(t+1)} = y_{1(t+1)}$ при $y_t = 1$, получаем в результате подстановки $y_{1t} = 1$ в исходное уравнение

$$y_{1(t+1)} = x_{1(t+1)} \vee x_{2(t+1)}.$$

Однако $y^2_{1(t+1)}=y_{t+1}$ при $y_t=0$. Значит, подставляя $y_t=0$ в исходное уравнение, получим $y^2_{1(t+1)}=x_{2(t+1)}$.

Следовательно,

$$y_{1(t+1)}=(x_{1(t+1)}\vee x_{2(t+1)})\,y_t\vee x_{2(t+1)}\,\bar{y}_t.$$

Ответ: $y_{1(t+1)}=(x_{1(t+1)}\vee x_{2(t+1)})\,y_t\vee x_{2(t+1)}\,\bar{y}_t$.

Любую схему с памятью можно представить в виде совокупности схем И, ИЛИ, НЕ и триггеров. Докажем это.

Опираясь на теорему о разложении функции по k переменным, любое уравнение из системы уравнений (11.15) можно представить в виде

$$y_{t,(t+1)}=y_{1,t}\underbrace{f_1(x_{1,(t+1)},\ x_{2,(t+1)},\ \ldots,\ x_{n,(t+1)},1,\ y_{2,t},\ldots,\ y_{m,t})}_{\bar{u}_{2,(t+1)}}\vee;$$

$$\vee\,\bar{y}_{1,t}\underbrace{f_1(x_{1,(t+1)},\ x_{2,(t+1)},\ \ldots,\ x_{n,(t+1)},\ 0,\ y_{2,t},\ \ldots,\ y_{m,t}).}_{u_{1,(t+1)}} \qquad (11.19)$$

В сокращенной записи (11.19) имеет вид

$$y_{t,(t+1)}=\bar{u}_{2,(t+1)}\,y_{1,t}\vee u_{1,(t+1)}\,\bar{y}_{1,t},$$

что является уравнением триггера с двумя входами, при этом функции $u_{1(t+1)}$ и $u_{2(t+1)}$ — входы триггера.

Произведя аналогичную операцию с другими уравнениями, приходим к выводу, что все они могут быть записаны в виде (11.19).

Пример 11.11. Задана система уравнений:

$$y_{1(t+1)}=x_{1(t+1)}\,y_{1t}\vee x_{2(t+1)}\,y_{2t};$$

$$y_{2(t+1)}=x_{2(t+1)}\,y_{1t}\vee y_{2t};$$

$$z_{1(t+1)}=x_{1(t+1)}\,x_{2(t+1)}\,y_{2t}.$$

Построить схему на элементах И, ИЛИ, НЕ и триггер.

Решение. Прежде всего осуществим разложение первого и второго уравнений по (11.19):

$$y_{1(t+1)}=y_{1t}\underbrace{(x_{1(t+1)}\vee x_{2(t+1)}\,y_{2t})}_{\bar{u}_{2(t+1)}}\ \vee\bar{y}_{1t}\underbrace{(x_{2(t+1)}\,y_{2t})}_{u_{1(t+1)}},$$

$$y_{2(t+1)}=y_{2t}\underbrace{(1)}_{\bar{v}_{2(t+1)}}\vee\bar{y}_{2t}\underbrace{(y_{1t}x_{2(t+1)})}_{v_{1(t+1)}}.$$

Окончательная схема имеет вид, представленный на рис. 11.27.

Ответ: логическая схема на рис. 11.27.

Рассмотрим решение дополнительных примеров с целью закрепления изложенного материала.

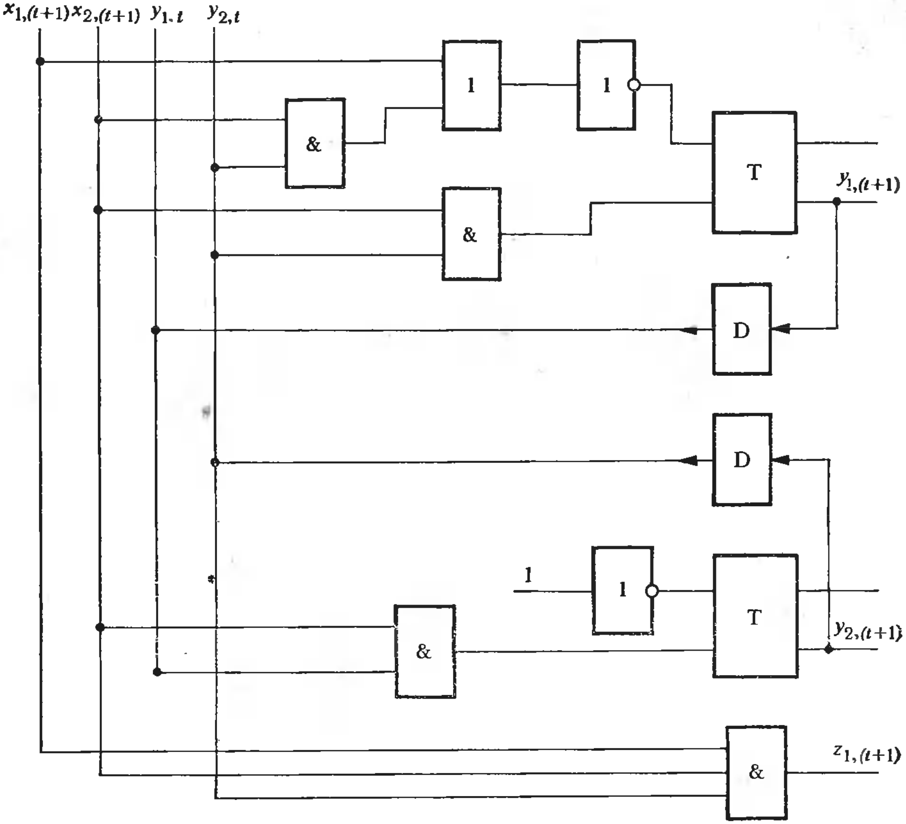


Рис. 11.27. Логическая схема устройства (к примеру 11.11)

Пример 11.12. Построить схему, соответствующую следующей системе уравнений:

$$\varphi_1 = x_1\bar{x}_2x_3\bar{x}_4 \vee \bar{x}_1\bar{x}_2x_3\bar{x}_4,$$

$$\varphi_2 = x_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1x_2x_3x_4 \vee \bar{x}_1x_2\bar{x}_3x_4.$$

Решение. Для решения используем метод нахождения простых импликант для базиса И, ИЛИ, НЕ (см. § 11.4):

$$\varphi_1 = \bigvee_1 \{1010,\ 0010\};$$

$$K_{11}^0 = \{1010\}_*, \quad K_{12}^0 = \{0010\}_*;$$

$$K_1^1 = \{x010\};$$

$$\varphi_2 = \bigvee_1 \{1000,\ 0111,\ 0101\};$$

$$K_{21}^0 = \{1000\},\ K_{22}^0 = \{0111\}_*, \quad K_{23}^0 = \{0101\}_*;$$

$$K_2^1 = \{01x1\};.$$

$$K^1 = \{x010,\ 01x1\};$$

$$K^0 = \{1000\}.$$

Здесь знак * означает, что отмеченные кубы покрываются.

Конечный вид уравнения для синтеза схемы

$$\varphi_1 = \bar{x}_2 x_3 \bar{x}_4;$$

$$\varphi_2 = \bar{x}_1 x_2 x_4 \vee x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4.$$

Полученную схему рекомендуем нарисовать самостоятельно.

Пример 11.13. Синтезировать схему, соответствующую системе уравнений

$$\varphi_1 = x_1 \bar{x}_2 x_3 \bar{x}_4 \vee \bar{x}_1 \bar{x}_2 x_3 \bar{x}_4;$$

$$\varphi_2 = x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \vee \bar{x}_1 x_2 x_3 x_4 \vee \bar{x}_1 x_2 \bar{x}_3 x_4,$$

используя метод каскадов и базисные функции И, ИЛИ, НЕ.

Решение (см. § 11.4). Произведем разложение:

$$\varphi_1 = \underbrace{(x_1 \bar{x}_2 x_3 \vee \bar{x}_1 \bar{x}_2 x_3)}_{f_{12}} \bar{x}_4;$$

$$f_{12} = (\underbrace{x_1 \bar{x}_2 \vee \bar{x}_1 \bar{x}_2}_{f_{121}})\, x_3;$$

$$f_{121} = \bar{x}_2;$$

$$\varphi_1 = \bar{x}_2 x_3 \bar{x}_4;$$

$$\varphi_2 = \underbrace{x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4}_{f_{22}} \vee \underbrace{(\bar{x}_1 x_2 x_3 \vee \bar{x}_1 x_2 \bar{x}_3)}_{f_{21}}\, x_4;$$

$$f_{21} = (\underbrace{\bar{x}_1 x_3 \vee \bar{x}_1 \bar{x}_3}_{f_{211}})\, x_2;$$

$$f_{211} = \bar{x}_1;$$

$$f_{21} = \bar{x}_1 x_2 x_4.$$

Конечный вид уравнений для синтеза схем

$$\left.\begin{aligned} \varphi_2 &= x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \vee \bar{x}_1 x_2 x_4; \\ \varphi_1 &= \bar{x}_2 x_3 \bar{x}_4. \end{aligned}\right\} \qquad (11.20)$$

Ответ: см. уравнение (11.20).

Пример 11.14. Найти оптимальное доопределение функции

$$f(x_1 x_2 x_3 x_4) = \bigvee_1 (1^*,\ 2,\ 5^*,\ 6^*,\ 8^*,\ 12,\ 15).$$

Решение (см. § 11.5 и § 11.6). Для решения примера используем геометрическое представление функции (рис. 11.28)

$$f_1 = \bigvee_1 (1,\ 2,\ 5,\ 6,\ 8,\ 12,\ 15). \qquad (11.21)$$

Ответ: см. уравнение (11.21).

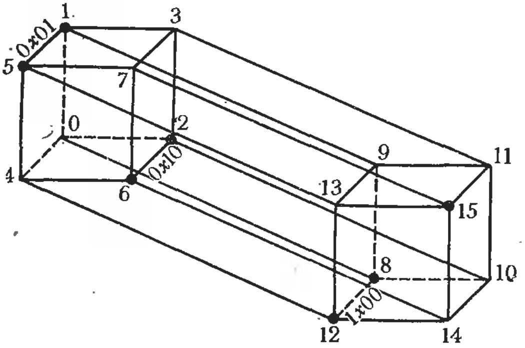


Рис. 11.28. Оптимальное доопределение логической функции (к примеру 11.14)

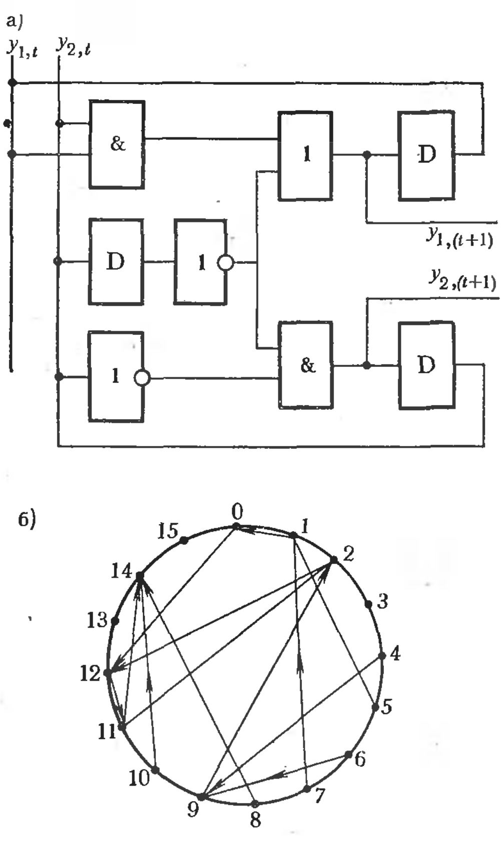


Рис. 11.29. Анализ работы логической схемы (*а*) и диаграмма переходов (*б*) (к примеру 11.15)

Пример 11.15. Провести анализ работы схемы, представленной на рис. 11.29, *а*.

Решение (см. § 11.9).

1. Система уравнений, описывающая работу схемы, имеет вид

$$y_{1(t+1)} = y_{1t} y_{2t} \vee \overline{y}_{2(t-1)};$$

$$y_{2(t+1)} = \overline{y}_{2(t-1)} \overline{y}_{2t}.$$

2. Построим, исходя из этой системы уравнений, таблицу состояний схемы (табл. 11.14).

Таблица 11.14

y_{1t}	y_{2t}	$y_{1(t-1)}$	$y_{2(t-1)}$	$y_{1(t+1)}$	$y_{2(t+1)}$	y_{1t}	y_{2t}
0	0	0 (1)	0	1	1	0	0
0	0	0 (1)	1	0	0	0	0
0	1	0 (1)	0	1	0	0	1
0	1	0 (1)	1	0	0	0	1
1	0	0 (1)	0	1	1	1	0
1	0	0 (1)	1	0	0	1	0
1	1	0 (1)	0	1	0	1	1
1	1	0 (1)	1	1	0	1	1

3. Диаграмма переходов имеет вид, показанный на рис. 11.29, *б*.

Пример 11.16. Синтезировать схему, выполненную на элементах И, ИЛИ, НЕ и триггерах и заданную системой уравнений

$$y_{1(t+1)} = x_{1(t+1)} \overline{x}_{2(t+1)} y_{1t} \vee y_{2t};$$

$$y_{2(t+1)} = x_{2(t+1)} \vee y_{1t}.$$

Решение (см. § 11.10). Уравнение триггера

$$y_{t+1} = \underbrace{\overline{x}_{2(t+1)}}_{y^1_{t+1}} y_t \vee \underbrace{x_{1(t+1)}}_{y^2_{t+1}} \overline{y}_t,$$

где

$$y^1_{t+1} = y_{t+1} \mid_{y_t = 1}; \quad y^2_{t+1} = y_{t+1} \mid_{y_t = 0}.$$

Отсюда

$$y^1_{1(t+1)} = y_{1(t+1)} \mid_{y_{1t} = 1} = x_{1(t+1)} \overline{x}_{2(t+1)} \vee y_{2t},$$

$$y^2_{1(t+1)} = y_{1(t+1)} \mid_{y_{1t} = 0} = y_{2t}.$$

Значит, $y_{1(t+1)} = (x_{1(t+1)} \overline{x}_{2(t+1)} \vee y_{2t}) y_{1t} \vee y_{2t} \overline{y}_{1t}.$

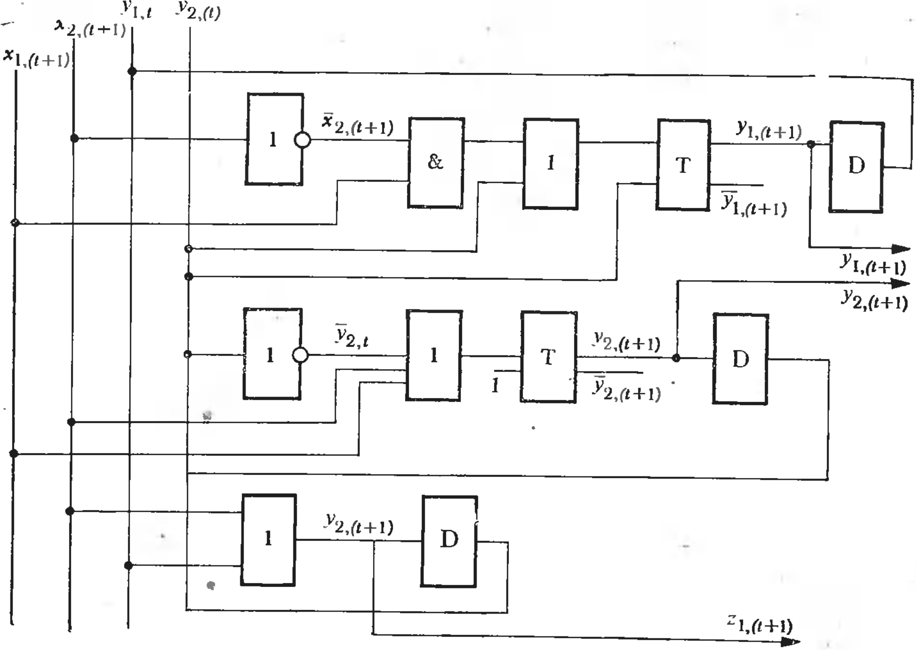


Рис. 11.30. Синтез схемы из примера 11.16

Аналогично преобразуем другие уравнения:

$$y_{2(t+1)} = x_{2(t+1)} \vee y_{1t},$$

$$y^1_{2(t+1)} = y_{2(t+1)}\,|_{\,y_{2t}=1} = x_{2(t+1)} \vee y_{1t},$$

$$y^2_{2(t+1)} = y_{2(t+1)}\,|_{\,y_{2t}=0} = x_{2(t+1)} \vee y_{1t},$$

т. е. от y_{2t} не зависит:

$$z_{1(t+1)} = x_{1(t+1)} \vee x_{2(t+1)} \vee \bar{y}_{2t},$$

$$z^1_{1(t+1)} = z_{1(t+1)}\,|_{\,y_{2t}=1} = x_{1(t+1)} \vee x_{2(t+1)},$$

$$z^2_{1(t+1)} = z_{1(t+1)}\,|_{\,y_{2t}=0} = 1,$$

$$z_{1(t+1)} = (x_{1(t+1)} \vee x_{2(t+1)} \vee \bar{y}_{2t})\, y_{2t} \vee \bar{y}_{2t}.$$

Ответ: окончательная схема представлена на рис. 11.30.

Пример 11.17. Синтезировать схему, заданную следующей таблицей состояний (табл. 11.15).

Решение (см. § 11.9). Уравнения, описывающие работу схемы, имеют вид

$$y_{1(t+1)} = \bar{y}_{1t}\bar{y}_{2t}\bar{y}_{2(t-1)} \vee \bar{y}_{1t}y_{2t}\bar{y}_{2(t-1)} \vee y_{1t}\bar{y}_{2t}\bar{y}_{2(t-1)} \vee y_{1t}y_{2t}\bar{y}_{2(t-1)} = \bar{y}_{2(t-1)}.$$

Таблица 11.15

y_{1t}	y_{2t}	$y_{2(t-1)}$	$y_{1(t+1)}$	$y_{2(t+1)}$	y_{2t}
0	0	0	1	1	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	0	0
1	0	1	0	0	0
1	1	0	1	0	1
1	1	1	0	0	1

Минимизация с помощью карт

	00	01	11	10	
0	1	1	1	1	$y_{2(t-1)}$
1					
	y_{1t}			y_{2t}	

$$y_{2(t+1)} = \overline{y}_{1t}\overline{y}_{2t}\overline{y}_{2(t-1)}.$$

Система уравнений

$$y_{1(t+1)} = \overline{y}_{2(t-1)};$$

$$y_{2(t+1)} = \overline{y}_{1t}\overline{y}_{2t}\overline{y}_{2(t-1)}.$$

Ответ: окончательная схема представлена на рис. 11.31.

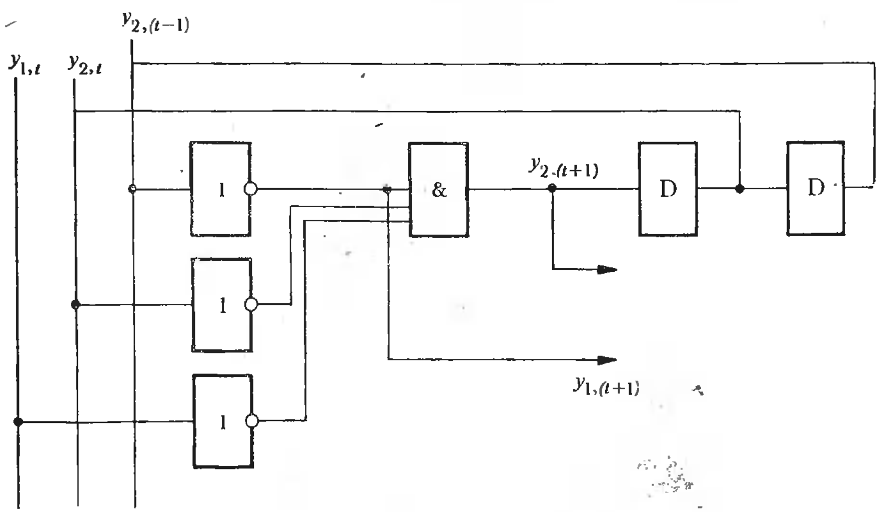


Рис. 11.31. Синтез схемы из примера 11.17

КУРСОВАЯ РАБОТА

§ П.1. Основные задачи курсовой работы

Цель курсовой работы — закрепление у студентов основных теоретических положений предмета, приобретение навыков практического решения технических задач логического проектирования узлов и блоков ЭВМ. Знания, полученные студентами в процессе выполнения курсовой работы, в дальнейшем послужат базой для изучения предметов «Расчет и проектирование элементов ЭВМ», «Основы теории и проектирования ЭВМ», а также будут использованы при курсовом и дипломном проектировании.

Для выполнения курсовой работы необходимо знание основных разделов предмета «Арифметические и логические основы цифровых автоматов».

Каждому студенту выдается индивидуальное ТЗ, содержащее следующие разделы:

1. Разработка машинного алгоритма выполнения арифметической операции, для которой определены система счисления, форма представления чисел, способ обработки разрядов или структурная схема самого устройства.
2. Проектирование методики контроля выполнения заданной операции.
3. Синтез логической схемы функционального узла (одноразрядного сумматора, свертки, дешифратора и т. п.) с использованием заданной системы логических элементов.

В разделе 1 задания на разработку выдаются алгоритмы выполнения логических операций:

а) сложение (вычитание) двух чисел, представленных в форме с фиксированной или плавающей запятой, на машине с параллельной, последовательной или параллельно-последовательной обработкой разрядов (отрицательные числа могут быть представлены в прямом, обратном и дополнительном кодах);

б) умножение двух чисел, представленных в форме с фиксированной или плавающей запятой, в прямом, обратном и дополнительном кодах с одновременным анализом одного, двух разрядов множителя и более начиная с младших или старших разрядов (в качестве варианта может быть задана конкретная структурная схема выполнения операции);

в) деление двух чисел, представленных в форме с фиксированной или плавающей запятой, на сумматоре обратного или дополнительного кода, с восстановлением или без восстановления остатков, с округлением или без округления частного;

г) извлечение квадратного корня из числа, заданного в форме с фиксированной или плавающей запятой, на сумматоре обратного или дополнительного кода;

д) прочие логические операции (в сочетании с арифметическими операциями).

Для иллюстрации работы алгоритма задаются два десятичных числа, которые необходимо перевести в заданную систему счисления и произвести оценку погрешностей, возникающих при переводе чисел из десятичной системы в другую, а также при выполнении самого алгоритма. В задании к первой части должно быть указано количество разрядов в разрядной сетке машины или точность представления информации.

Раздел 2 посвящен разработке методов контроля выполнения заданной операции. Эта часть задания базируется на выборе методики контроля арифметических и логических операций и методики осуществления контрольной операции.

В разделе 3 студенту предлагается синтезировать какое-то простейшее логическое устройство (или его часть), предназначенное для реализации некоторых функций разработанного в разделе 1 алгоритма выполнения операции.

Здесь можно предложить синтез одноразрядного сумматора (в заданной системе счисления); схемы шифратора-дешифратора; схемы свертки для образования контрольного кода; схемы анализатора разрядов множителя и т. п.

Исходными данными для синтезирования логического устройства являются:

система логических элементов, задаваемых преподавателем, таблица состояний.

В качестве логических элементов целесообразно задавать элементы типа И, ИЛИ, НЕ или их комбинации И—НЕ, ИЛИ—НЕ, И—ИЛИ—НЕ и др.

Работа над этим разделом заключается в логическом описании функционирования заданного устройства, минимизации логических функций и разработке самой логической схемы.

Курсовая работа оформляется в виде пояснительной записки объемом до 30 рукописных страниц и двух листов чертежей формата А1.

Пояснительная записка должна включать в себя:

титульный лист с названием работы, фамилией студента и консультанта;

техническое задание;

раздел первый: «Разработка машинного алгоритма выполнения операции» с обоснованием и описанием работы алгоритма;

раздел второй: «Разработка метода контроля операции» с обоснованием выбора метода и описанием его работы;

раздел третий: «Синтез логической схемы», содержащий логическое описание функционирования заданной схемы, методику минимизации состояний, окончательный вариант логической схемы;

заключение, в котором дается критический обзор выполненной работы, предложения по устранению недостатков работы алгоритма или схемы.

Графическая часть содержит чертежи на двух листах формата А1, на которых должны быть представлены:

лист 1 — структурная схема машинного алгоритма выполнения заданной операции;

лист 2 — логическая схема разработанного устройства.

Пояснительная записка и графическая часть должны быть оформлены в соответствии с требованиями ЕСКД. Условные обозначения приведены в приложениях.

Оформленная курсовая работа, подписанная консультантом, предъявляется комиссии, назначаемой заведующим кафедрой. В состав комиссии входят два человека. На защите студент должен показать умение кратко и грамотно излагать технические вопросы по проектированию, уметь обосновать выбор принятого решения, а также знать в подробностях все разработанные вопросы. В результате защиты студенту ставится оценка по четырехбалльной системе: «отлично», «хорошо», «удовлетворительно», «неудовлетворительно».

§ П.2. Методические указания по выполнению курсовой работы

Графическая часть работы состоит из двух листов чертежей формата А1. На листе 1 представляется алгоритм выполнения заданной операции, а на листе 2 — логическая схема устройства, синтез которой осуществляется в работе.

Для представления алгоритмов используется специальный граф-схемный язык, основными элементами которого являются функциональные и вспомогательные блоки, описывающие операции алгоритма.

Каждому отдельному действию, осуществляемому алгоритмом, придается значение микрооперации. Любая микрооперация изменяет состояние блоков разрабатываемого устройства. Всем блокам устройства присваиваются имена (условные обозначения) и порядковые номера: сумматор — СМ, сумматор мантисс — СММ, регистр — Рг, сумматор порядка — СМП, дешифратор — ДШ, регистр переносов — РгП, счетчик — СЧ.

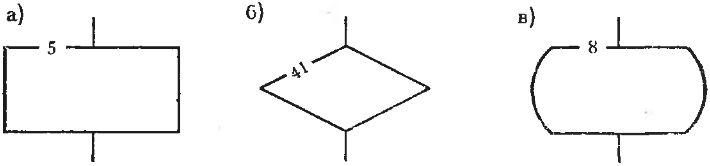


Рис. П.1. Условные обозначения функционального (*а*) и условного (*б*) блоков и блока комментариев (*в*)

На рис. П.1, *а — в* представлены условные обозначения функциональных и вспомогательиых блоков. Каждому блоку алгоритма присваивается номер, который записывается слева в разрыве линии обвода блока. Описание последовательности выполиения действий ведется с использованием номеров блоков. Блоки в алгоритме соединяются линиями связи. Как правило, линия связи входит в блок сверху и выходит снизу. Линия связи дается без стрелок, если она идет сверху вниз и слева направо. Во всех других случаях ставятся стрелки. Толщина линий связи в два раза меньше, чем толщина обводки блоков. Линии связи проводятся параллельно внешним краям рамки листа. Допускается пересечение их или изгиб под углом 90°. Расстояние между параллельными линиями должно быть не менее 8 мм. Размеры блоков выбираются из ряда: $a=10$, 15, 20, 30, ... мм, $b=1{,}5a$. Расстояние между блоками должно быть не менее 10 мм.

В каждый блок может войти и выйти только по одной линии связи. Из условного блока выходят две связи: снизу и из правого угла. Над связями, выходящими из условного блока, пишут слова: «Да», «Нет».

Например, на рис. П.2, *а* показан блок, который осуществляет проверку выполнения условия $\alpha=1$ и разветвление алгоритма по двум направлениям. На рис. П.2, *б* представлен условный блок-переключатель, употребляемый в случае большого количества разветвлений (для этого блока условия ветвления записываются на каждом из выходов).

Линии связи проводятся в горизонтальном, либо в вертикальном направлеиии. Несколько логических связей могут объединяться в одну линию (рис. П.3, *а*) или пересекаться под углом 90°. Для записи действий, протекающих параллельно во времени, используют условное обозначение параллельного процесса (рис. П.3, *б*). Это означает, что несколько процессов начинаются и заканчиваются одновременно. Линии связи можно разрывать, обозначив точки разрыва одинаковыми символами внутри соединителя (рис. П.4, *а, б*). Около места разрыва связи допускается в скобках запись комментария, определяющего, куда направляется связь и откуда приходит.

Для описания проходящих процессов кроме условных обозначений используется оператор присваивания «: =». Например, действие «передать в регистр *А* изображение числа *В* в обратном коде» запишется в виде Рг*А* : = $[B]_{\text{об}}$. Запись Рг*В* : = 0 означает установку регистра в нулевое положение.

Оператор присваивания указывает и устройство, на котором производнтся действие. Так, запись СМ*А*:= =[СМ*А*]+[Рг*В*] означает, что действие осуществляется на сумматоре *А*. Запись СМ*А*: =[СМ*В*]+[Рг*D*] недопустима, так как неизвестно, где производится действие.

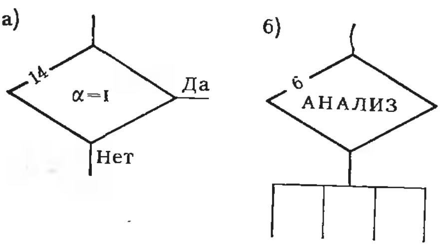


Рис. П.2. Блоки разветвлений

Выполнение сдвигов обозначается следующим образом: сдвиги сумматора вправо на *k* разрядов — *R* (*k*, СМ): сдвиги регистра *А* влево на *n* разрядов — *L* (*n*, Рг*А*).

Поэтому запись вида СМ*А* : = *R* (2, СМ*А*) означает, что в сумматоре *А* осуществляется сдвиг вправо на два разряда и сдвинутое число остается в сумматоре A. Модифицированный сдвиг обозначается следующим образом: Рг2 : $=R_м$ (2Рг2).

Операции преобразования в обратный и дополнительный коды можно рассматривать как самостоятельные микрооперации и для их записи использовать свои обозначения. Например, содержимое регистра A преобразуется в дополнительный код — ДК (РгA) или содержимое сумматора A преобразуется в обратный код — ОК (СМA). Операция инвертирования — Рг$\overline{A}$, что означает инвертирование содержимого регистра A во всех разрядах.

Двоичные разряды во всех блоках устройства нумеруются слева направо. Номер разряда какого-то блока указывается в квадратных скобках после нaвания блока: Рг1 [5] — 5-й разряд регистра 1; Рг2 [15÷÷2] — совокупность разрядов со 2-го по 15-й регистра 2.

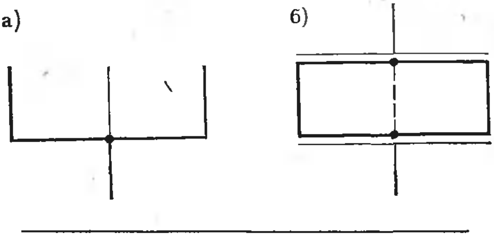


Рис. П.3. Логические связи

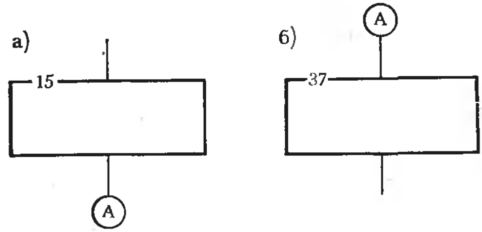


Рис. П.4. Разрыв соединительных линий

Ниже представлены основные обозначения, используемые для записи алгоритмов:

Выполняемая операция	Обозначение
Установка в нулевое положение	Рг1 : $=0$ СМ : $=0$
Инвертирование кода	Рг1 : $=\overline{\text{Рг1}}$
Передача информации из одного блока в другой:	
а) из Рг1 в Рг2 прямым кодом	Рг2 : = [Рг1]$_{пр}$
б) из Рг1 в СМ обратным кодом	СМ : = [Рг1]$_{ок}$
в) из сумматора в регистр 2 дополнительным кодом	Рг2 : = [СМ]$_{дк}$
Сдвиг регистра Рг1 на n разрядов:	
а) влево	Рг1 : = L (n, Рг1)
б) вправо (модифицированный)	Рг1 : $=R_м$ (n, Рг1)
Операция последовательного счета	СЧ : = СЧ + 1
Операция сложения кодов: содержимое СМ сложить с содержимым Рг1	СМ : = [СМ] + [Рг1]
Частичная передача информации из Рг2 с 3-го по 15-й в Рг1 с 1-го по 13-й разряды . .	Рг1 [13÷1] : = Рг2 [15÷3]

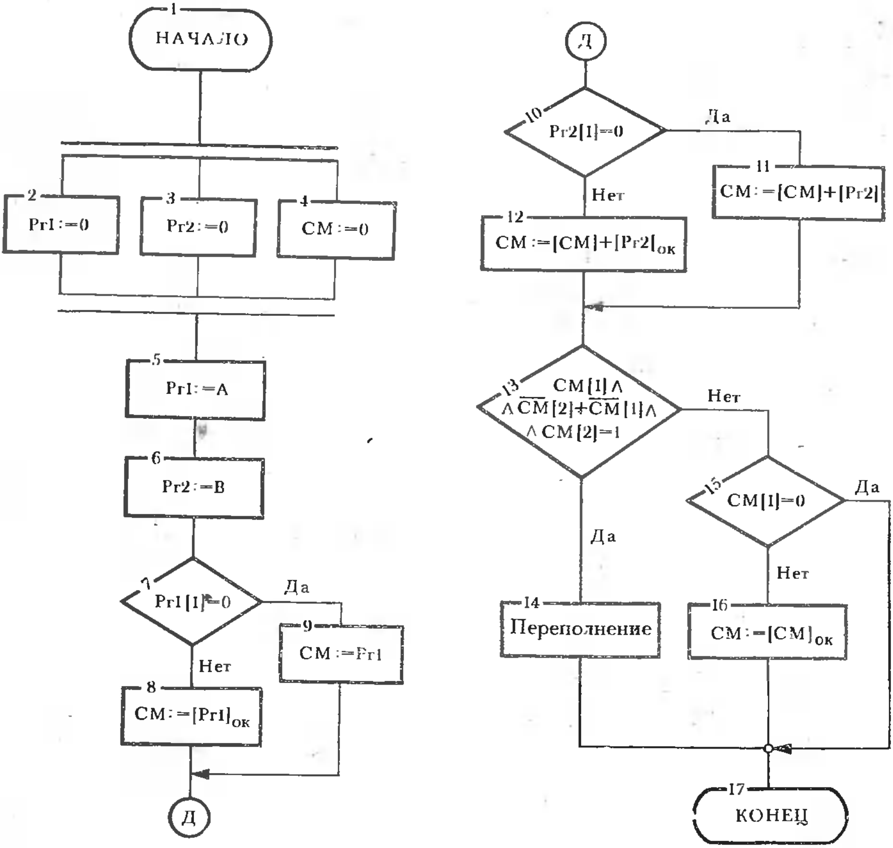


Рис. П.5. Алгоритм сложения чисел, представленных в форме с фиксированной запятой, на сумматоре обратного кода

Любой алгоритм начинается с блока комментария «Начало» и заканчивается блоком комментария «Конец». Внутри функциональных и вспомогательных блоков записывается содержательная часть процессов, которая выполняется данным блоком алгоритма.

Для описания повторяющихся действий можно ввести процедуры. Процедура оформляется в виде отдельного алгоритма, которому присваивается имя. Кроме того, могут быть даны также наборы формальных параметров, заменяемых на фактические при обращении к данной процедуре. Имя процедуры и перечень формальных параметров указываются в блоке «Начало» алгоритма процедуры. Обращение к процедуре осуществляется из общего алгоритма с помощью блока обращения, в котором указываются имя процедуры и в круглых скобках фактические параметры, соответствующие данной процедуре.

Пример оформления алгоритма сложения чисел с фиксированной запятой на сумматоре обратного кода приведен на рис. П.5.

На листе 2 курсовой работы располагается логическая схема устройства или часть устройства, синтез которого выполняется на основе заданной системы логических элементов. Этот лист оформляется в соответствии с требованиями ГОСТ 2743—72. Кроме логической схемы на листе можно представить таблицы истинности состояний или логические формулы.

§ П.3. Типовые задания

Задание 1.

1. Разработать алгоритмы выполнения операций сложения (вычитания) двоичных чисел для машины с плавающей запятой на сумматоре обратного кода. Разрядная сетка машины — 24 разряда:

разряды 1—17 — для мантиссы со знаком;

разряды 18—24 — для порядка со знаком.

На примере десятичных чисел $A = -2{,}862$ и $B = 42{,}26$ определить погрешности перевода исходных данных в двоичную систему счисления и обратно.

2. Разработать контроль операции сложения по модулю 5 с иллюстрацией действия метода контроля на числах A и B.

3. Произвести синтез логической схемы двоичного сумматора, работающего по модулю 5 на основе базисных функций И, НЕ.

Задание 2.

1. Разработать алгоритм выполнения операции умножения двоичных чисел, представленных в форме с фиксированной запятой. Все действия выполняются на сумматоре обратного кода с анализом цифр множителя начиная с младших разрядов. Разрядная сетка машины — 24 разряда. Результат операции получить с точностью в 24 двоичных разряда. Определить количество дополнительных разрядов сумматора на примере десятичных чисел $A = 35{,}24$, $B = -2{,}762$.

Сравнить ошибки округления для случаев:

а) округление в конце операции умножения;

б) округление в конце каждого такта суммирования.

2. Разработать систему контроля операции суммирования двух чисел, используя метод контроля по модулю 7.

Синтезировать логическую схему двоичного сумматора, работающего по модулю 7, на основе базисных функций ИЛИ, НЕ.

Задание 3.

1. Разработать машинный алгоритм выполнения операции деления двоичных чисел, представленных в форме с плавающей запятой, на сумматоре обратного кода (дополнительного кода). Разрядная сетка машины — 32 двоичных разряда:

разряды 1—24 — для мантиссы,

разряды 25—32 — для порядка.

Результаты операции нормализовать, округлить, определить погрешность выполнения операции иа примере десятичных чисел $A = 2636{,}4$ и $B = -0{,}24343$.

2. Разработать контроль операции суммирования машинных изображений со знаком, используя контроль по модулю 3.

3. Произвести синтез логической схемы двоичного сумматора на основе базисных функции И, ИЛИ, НЕ, осуществив минимизацию по методу Квайна.

Задание 4.

1. Разработать машинный алгоритм выполнения операции сложения (вычитания) десятичных чисел на сумматоре, работающем в системе с весами 2, 4, 2, 1. Форма представления чисел — фиксированная запятая перед старшим разрядом. Количество десятичных разрядов в разрядной сетке машины равно четырем. Отрицательные числа представляются дополнительным кодом. Иллюстрацию действия алгоритма провести на примере десятичных чисел $A = 26{,}783$, $B = -0{,}1225$.

Произвести оценку погрешностей.

2. Разработать контроль операции суммирования машинных изображений, используя метод контроля по модулю 9.

3. Синтезировать логическую схему одноразрядного десятичного сумматора в коде 2, 4, 2, 1, используя в качестве базисных функций И—НЕ.

З а д а н и е 5.

1. Разработать алгоритм выполнения операции умножения двоичных чисел, представленных в форме с плавающей запятой. Операция производится с одновременным анализом двух разрядов множителя начиная с младших разрядов. Количество разрядов разрядной сетки ЭВМ — 24 (из них 18 разрядов — мантисса и 6 разрядов — порядок). Определить точность полученного результата на примере десятичных чисел $A=284{,}7$, $B=-0{,}03626$.

Определить относительную и абсолютную погрешности выполнения операции с округлением и без округления (нормализация обязательна). Оценить эффективность ускорения операции.

2. Разработать метод контроля всей операции по модулю 3.

3. Разработать логическую схему устройства коррекции контрольного кода при сдвиге числа на два разряда, используя в качестве базиса функцию «стрелка Пирса».

З а д а н и е 6.

1. Разработать алгоритм выполнения операции деления десятичных чисел, представленных в двоично-кодированной системе с весами 8, 4, 2, 1. Форма представления чисел — фиксированная запятая, машинные изображения чисел по своей абсолютной величине меньше 1. Все действия производятся на сумматоре дополнительного кода. Диапазон представления исходных даниых $\pm 10^{\pm 3}$. Определить разрядную сетку машины на примере чисел (максимальное и минимальное) $A=19{,}110$, $B=-1{,}300$.

2. Используя систему элементов типа И, ИЛИ, НЕ, синтезировать логическую схему одноразрядного десятичного сумматора.

3. Разработать методику контроля операции суммирования машинных изображений без знаков, использовав контроль по модулю 15.

З а д а н и е 7.

1. Разработать машинный алгоритм выполнения операций сложения и умножения чисел, представленных в системе остаточных классов с основаниями

$$P_1=2,\quad P_2=7,\quad P_3=11,\quad P_4=19,\quad P_5=23.$$

Числа в машине представляются в форме с фиксированной запятой. Для перевода исходных десятичных чисел в систему в остаточных классах и обратно определить ортогональные базисы. Работу алгоритмов проверить на примерах сложения (числа $A=234$ и $B=-157$) и умножения (числа $A=135$ и $B=-24$).

2. Разработать методику контроля выполнения операций сложения и умножения в системе остаточных классов.

3. Разработать логическую схему сумматора по модулю 19.

Все дополнительные сведения, необходимые для выполнения задания, согласуются с преподавателем.

З а д а н и е 8.

1. Разработать машинный алгоритм выполнения арифметических операций сложения и деления на устройстве, структурная схема которого задается преподавателем. Форма представления — плавающая запятая. Отрицательные числа представляются в обратном коде. Разрядная сетка машины — 12 двоичных разрядов:

для маитиссы — 8 разрядов,

для порядка — 4 разряда.

Оценку точности и проверку работы алгоритмов провести на десятичных числах $A=0{,}153$, $B=-12{,}52$.

2. Разработать систему контроля операции сложения по модулю 5.

3. Разработать логическую схему свертки контрольных кодов (сумматор по модулю 5), используя в качестве базисных функций «стрелку Пирса».

Задание 9.

1. Разработать машинный алгоритм арифметических операций вычитания и умножения чисел $A=253,1$ и $B=0,691$. Операции производятся в арифметическом устройстве, структурная схема которого задается преподавателем. Форма представления чисел — с плавающей запятой. Отрицательные числа представляются в дополнительном коде. Сумматор мантисс имеет 24 двоичных разряда, сумматор порядка — 8 двоичных разрядов. Результаты операций перевести в десятичную систему счисления и дать оценку точности. Разработать контроль выполнения операций по модулю 3.

2. Разработать логическую схему для сравнения трехразрядных двоичных чисел на элементах И, ИЛИ, НЕ.

Задание 10.

1. Разработать алгоритм выполнения операции умножения чисел с помощью таблицы квадратов по формуле

$$AB \approx (A_1 + 2^{-m/2} A_2)(B_1 + 2^{-m/2} B_2) = A_1B_1 + 2^{-m/2}(A_2B_1 + A_1B_2),$$

где $A_jB_j=0,25\ [(A_i+B_j)^2-(A_i-B_j)^2]$.

Числа представлены в форме с фиксированной запятой. Действия производятся на сумматоре дополнительного кода. Количество разрядов — 16 двоичных разрядов без учета знака.

Дать оценку точности перевода десятичных чисел и самой операции умножения на примере чисел $A=-0,8724$, $B=0,4901$.

2. Проработать методику контроля операции суммирования чисел с учетом знаков по модулю 3.

3. Разработать логическую схему одноразрядного двоичного сумматора, используя в качестве базиса функцию Пирса (Вебба).

Задание 11.

1. Разработать алгоритм извлечения квадратного корня из двоичного числа с фиксированной запятой с использованием итерационной формулы

$$x_{i+1} = 0,5\,(x_i + a/x_i), \quad i = 1,\ 2,\ 3,\ 4,$$

где x_{i+1} — новое значение приближения к корню, x_i — текущее приближение к корню $x_1=10$, a — число, из которого извлекается корень.

Разрядность числа a, сумматора и результата — 20 двоичных разрядов.

Реализовать разработанный алгоритм на числе $a=0,3761967$.

Оценить точность выполнения операции для заданного числа.

2. Разработать метод контроля микроопераций сложеиия и сдвига на один разряд вправо по модулю 3.

3. Разработать логические схемы свертки первого и второго уровней по модулю 3 на базе логических функций ИЛИ, НЕ.

Задание 12.

1. Разработать алгоритмы выполнения операции сложения десятичных чисел, представленных в десятичном коде «с избытком три» и в форме с плавающей запятой. Разрядность мантиссы — 6 тетрад, разрядность порядков — 2 тетрады. Все операции производятся на пятиразрядном (однотетрадном) сумматоре, имеющем один двоичный разряд для фиксации переноса. Отрицательные числа представляются в дополнительном коде.

Работу алгоритма иллюстрировать на примере чисел $A=27,58367$, $B=-317,2598$.

Оценить для этих чисел погрешность перевода и погрешность выполнения операции.

2. Разработать метод контроля операции по модулю 9.

3. Синтезировать логическую схему однотетрадного сумматора в коде «с избытком три», используя в качестве базисной функции операцию Пирса (Вебба) (ИЛИ — НЕ).

Задание 13.

1. Разработать машинный алгоритм ускоренного умножения двоичных чисел, представленных в форме с фиксированной запятой, используя формулу:

$$AB=2A\frac{B}{2}, \text{ при } B \text{ — четном;}$$

$$AB=2A\frac{B-1}{2}+A, \text{ при } B \text{ — нечетном.}$$

Разрядная сетка — 16 двоичных разрядов. Проверить работу алгоритма на примере двух чисел: $A=-0{,}253$; $B=0{,}468$.

2. Разработать контроль выполнения операции по конечным результатам, используя контроль по модулю 15.

3. Синтезировать логическую схему двоичного сумматора для получения сверток, применив метод Квайна—Мак-Класки, на элементах И—НЕ.

Задание 14.

1. Разработать машинный алгоритм умножения десятичных чисел, представленных в форме с фиксированной точкой, на сумматоре обратного кода. Разрядность:

для операндов — 3 тетрады;

для сумматора — 4 тетрады.

Использовать код $Д_1$ и любой из методов ускорения.

Проверить работу алгоритма при $A=256$, $B=-362$.

2. Разработать контроль элементарной операции сдвига кодов, применив контроль по модулю 3.

3. Синтезировать схему одноразрядного десятичного сумматора на элементах ИЛИ—НЕ.

ЛИТЕРАТУРА

1. *Акушский И. Я., Юдицкий Д. А.* Машинная арифметика в остаточных классах. М., 1968.
2. *Алферова З. В.* Теория алгоритмов. М., 1973.
3. *Анисимов Б. В., Четвериков В. Н.* Основы теории и проектирования ЭЦВМ. М., 1970.
4. *Атстопас Ф. Ф.* Арифметические основы цифровых вычислительных машин. Каунас, 1970.
5. *Баранов С. И.* Синтез микропрограммных автоматов. Л., 1974.
6. *Глушков В. М.* Синтез цифровых автоматов. М., 1962.
7. *Голышев Л. К.* Структурная теория цифровых машин. М., 1971.
8. *Деге В.* ЭВМ думает, считает, управляет. М., 1974.
9. *Ершов А. П., Шура-Бура М. Р.* Пути развития программирования в СССР. — Кибернетика, 1976, № 6.
10. *Калужнин Л. А.* Что такое математическая логика. М., 1964.
11. *Карцев М. А.* Арифметика цифровых машин. М., 1969.
12. *Кобринский Н. Е., Пекелис В. Д.* Быстрее мысли. М., 1963.
13. *Лысиков Б. Г.* Арифметические и логические основы ЭЦВМ. Минск, 1974.
14. Решение математических задач на автоматических цифровых машинах/ *Люстерник Л. А., Абрамов А. А., Шестаков В. И., Шура-Бура М. Р.* М., 1952.
15. *Новиков П. С.* Элементы математической логики. М., 1959.
16. *Папернов А. А.* Логические основы ЦВМ. М., 1972.
17. *Питерсон У., Уэлдон Э.* Коды, исправляющие ошибки. М., 1976.
18. *Поспелов Д. А.* Арифметические основы вычислительных машин дискретного действия. М., 1970.
19. *Поспелов Д. А.* Логические методы анализа и синтеза схем. М. — Л., 1964.
20. *Путинцев Н. Д.* Аппаратный контроль управляющих цифровых вычислительных машин. М., 1966.
21. *Фистер М.* Логическое проектирование цифровых вычислительных машин. Киев, 1964.
22. Синтез вычислительных алгоритмов управления и контроля/*Кузьмин И. В., Березюк Н. Т., Фурманов К. К., Шаронов В. Б.* Киев, 1975.
23. Энциклопедия кибернетики. Тт. 1, 2. Киев, 1975.
24. *Евреинов Э. В., Прангишвили И. В.* Цифровые автоматы с настраиваемой структурой. М., 1974.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

ОГЛАВЛЕНИЕ