ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

В учебном пособии излагаются основные базовые теоретические вопросы, касающиеся организации ЭВМ как вычислительной машины. В нем рассмотрены основные сведения о представлении информации в ЭВМ, показаны различные варианты ее представления, рассмотрены наиболее распространенные варианты систем счисления, применяемые в ЭВМ. Основная часть пособия посвящена организации арифметических вычислений в ЭВМ и различным способам сложения, вычитания, умножения и деления, используемым в современных компьютерах.

Пособие предназначено для студентов младших курсов, обучающихся по специальности 230101.65 и направления 230100.62, может быть полезно при изучении других дисциплин по специальности 230101: например, «Теория автоматов», «Схемотехника ЭВМ», «Организация ЭВМ, комплексов и систем», при выполнении курсового проектирования по дисциплине «Организация ЭВМ, комплексов и систем», а также может быть использовано студентами старших курсов, магистрантами, аспирантами и специалистами–проектировщиками вычислительной техники.

Несмотря на то, что в последнее время вышло достаточно много различных изданий по вопросам теории информации, способам и алгоритмам выполнения арифметических операций, применяемым в ЭВМ, информация в них слишком разрознена и иногда не совсем очевидна. В предлагаемом учебном пособии собраны сведения из различных изданий, касающихся способов и принципов выполнения арифметических операций, а также рассмотрены интегрированные прикладные программы, наиболее необходимые при работе на персональных ЭВМ. Для изучения предлагаются такие прикладные программы пакета Microsoft Office: Microsoft Word и Microsoft Excel. Приведены общие сведения о представлении информации в ЭВМ. Рассмотрены различные системы счисления, преобразование чисел, формы представления чисел в ЭВМ и др.

При подготовке пособия авторы старались отразить по возможности полный спектр вариантов организации каждой из основных арифметических операций (сложения, вычитания, умножения и деления). Для каждого из вариантов арифметических операций приведены конкретные практические примеры, особенно необходимые при изучении данной дисциплины, так как многие из алгоритмов и способов выполнения операций достаточно сложно понять из теоретических сведений.

В конце каждой главы предлагаются контрольные вопросы и упражнения.

Авторы выражает искреннюю признательность рецензентам – кандидату технических наук, доценту А.П. Жмакину и кандидату технических наук Д.В. Тюпину за детальное изучение рукописи и ценные замечания, которые позволили улучшить содержание пособия. Также авторы благодарны сотрудникам редакционно-издательского отдела Курского государственного технического университета за нелегкий труд, связанный с редактированием рукописи.

Авторы приносят извинения читателям данного учебного пособия за допущенные ошибки и неточности, и просят направлять свои отзывы и пожелания на кафедру вычислительной техники КурскГТУ.

ВВЕДЕНИЕ

Дисциплина цикла общих математических и естественнонаучных дисциплин ЕН.Ф.2 «Информатика» является обязательной при подготовке специалиста по направлению 230100.62 Информатика и вычислительная техника и специальности 230101.65 Вычислительные машины, комплексы, системы и сети и занимает важное место в ряду других фундаментальных наук. Эта включает в себя изучение персонального компьютера и его возможностей, и является базовой для дисциплин «Теория автоматов», «Организация, ЭВМ и систем». Обязательным является использование компьютера как помощника при оформлении текстовой документации и использовании обработки чисел в электронных таблицах; ознакомление со способами задания чисел в ЭВМ; умение выполнять арифметические операции, используемые в вычислительной технике.

Учебное пособие состоит из трех глав.

Первая глава посвящена прикладным аспектам использования ЭВМ. В ней рассмотрены основные вопросы по применению и использованию программ пакета Microsoft Office: Microsoft Word и Microsoft Excel. Эта глава особенно полезна студентам первого и второго курсов, начинающим общение с компьютером. В ней рассмотрены подготовка, редактирование и оформление текстовой документации, графиков, диаграмм и рисунков; обработка числовых данных в электронных таблицах.

Вторая глава рассматривает общие сведения о представлении информации в ЭВМ. В ней обсуждаются основные вопросы, касающиеся информации, систем счисления и чисел, такие как представление информации в цифровых автоматах (ЦА); позиционные системы счисления; методы перевода чисел; форматы представления чисел с плавающей запятой.

Третья глава учебного пособия обсуждает вопросы арифметики ЭВМ. В ней рассматриваются различные варианты и способы организации операций сложения, вычитания, умножения и деления, применяемых в компьютерах.

1. ВЫЧИСЛИТЕЛЬНАЯ ПРАКТИКА

Одним из условий эффективного использования вычислительной техники является создание специализированных прикладных программ [1,2], среди которых особая роль отводится офисным прикладным программам (рис. 1.1).

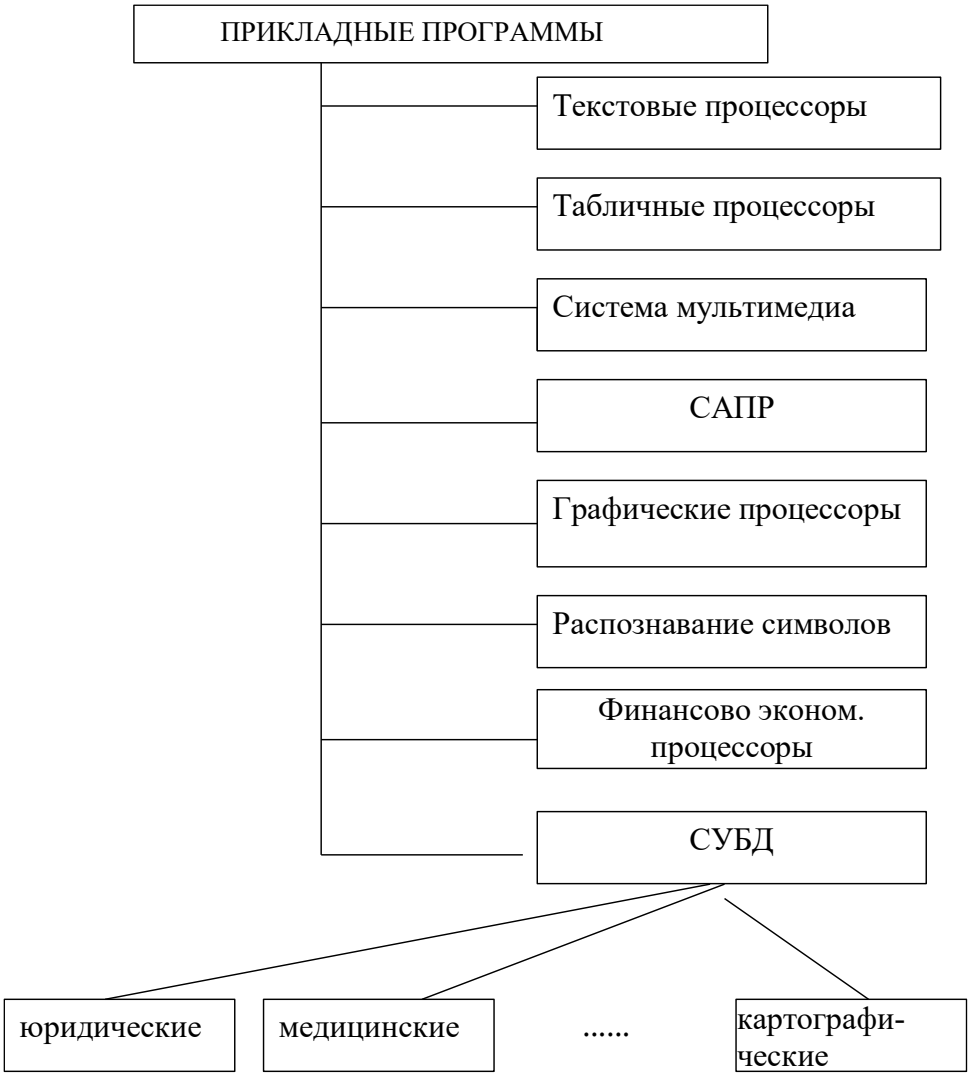


Рис. 1.1. Структура прикладных программ Windows

Кроме прикладных программ, ориентированных на конкретную предметную область деятельности, интенсивно проектируются интегрированные прикладные программы, соединяющие в себе различные функции отдельных предметных программ. Наиболее известным примером интегрированных

прикладных программ является офисный пакет MS Office. Он содержит такие прикладные программы, как Microsoft Word, Microsoft Excel, Microsoft Access, Microsoft PowerPoint, Microsoft Visio и т.д. Рассмотрим в данном учебном пособии две прикладные программы Microsoft Word и Microsoft Excel.

1.1. Текстовый редактор документов Microsoft Word

Интегрированная программа *Microsoft Word* [2] предназначена для создания документов, включающих форматированный текст, таблицы, списки, формулы и другие объекты (рис. 1.2).

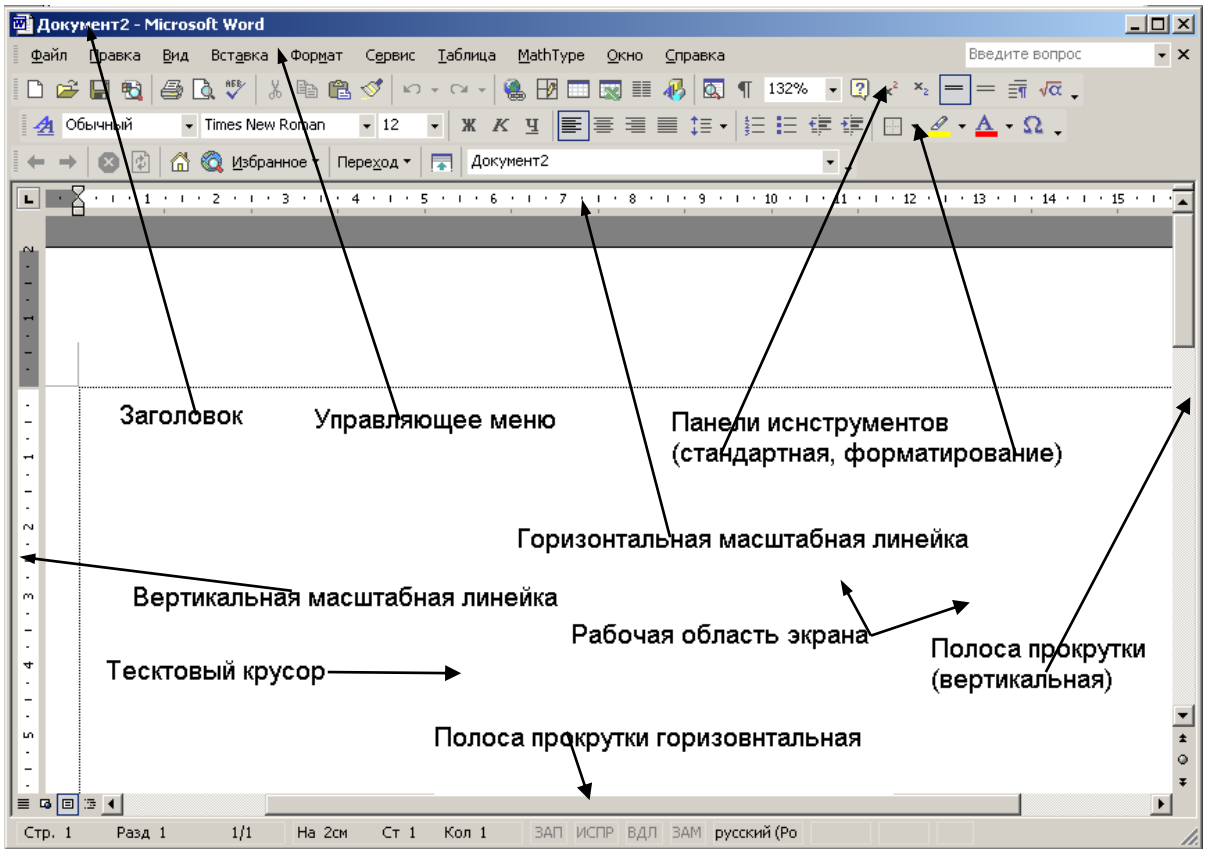


Рис. 1.2. Внешний вид программы Microsoft Word

Внешний вид экрана *Word*, как и любого окна Windows, включает рабочую область экрана, заголовок, управляющее меню, панели инструментов, полосы прокрутки и масштабные линейки. Основную площадь окна занимает рабочая область экрана, в которой выводится редактируемый документ.

Основные команды по редактированию документа собраны на панелях инструментов. Различают стандартную панель, панели форматирования, рисования, таблиц и границ и др. Для выбора и подключения необходимой панели инструментов используется вкладка управляющего меню **Вид**, категории **Панели Инструментов** (рис. 1.3).

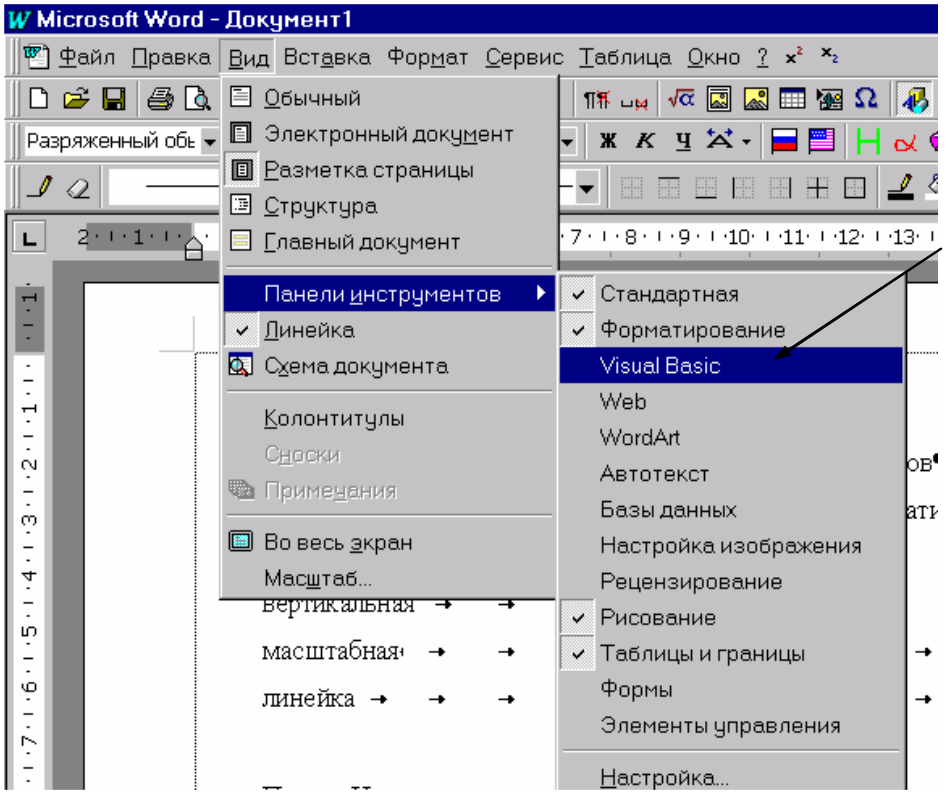


Рис. 1.3. Подключение панелей инструментов

Содержимое **Панели Инструментов** можно настраивать добавлением или удалением кнопок. Например, для вынесения команд нижний (x_2) и верхний (x^2) индекс необходимо проделать следующую последовательность действий. В категории **Вид**, вкладке **Панели Инструментов** выбрать пункт **Настройка**. В появившемся окошке выбрать вкладку **Команды.** Появится окошко (рис. 1.4), которое разделено на два поля: слева представлены все категории меню редактора Word, а справа команды, соответствующие выбранной категории. Для настройки панели инструментов (добавление команд) необходимо левой кнопкой мыши выбрать требуемую команду и, удерживая кнопку нажатой, переместить команду на панель

инструментов. Для удаления команды с **Панели Инструментов** необходимо проделать обратную операцию, то есть перетащить удаляемую команду на окошко Настройка и команда удаляется.

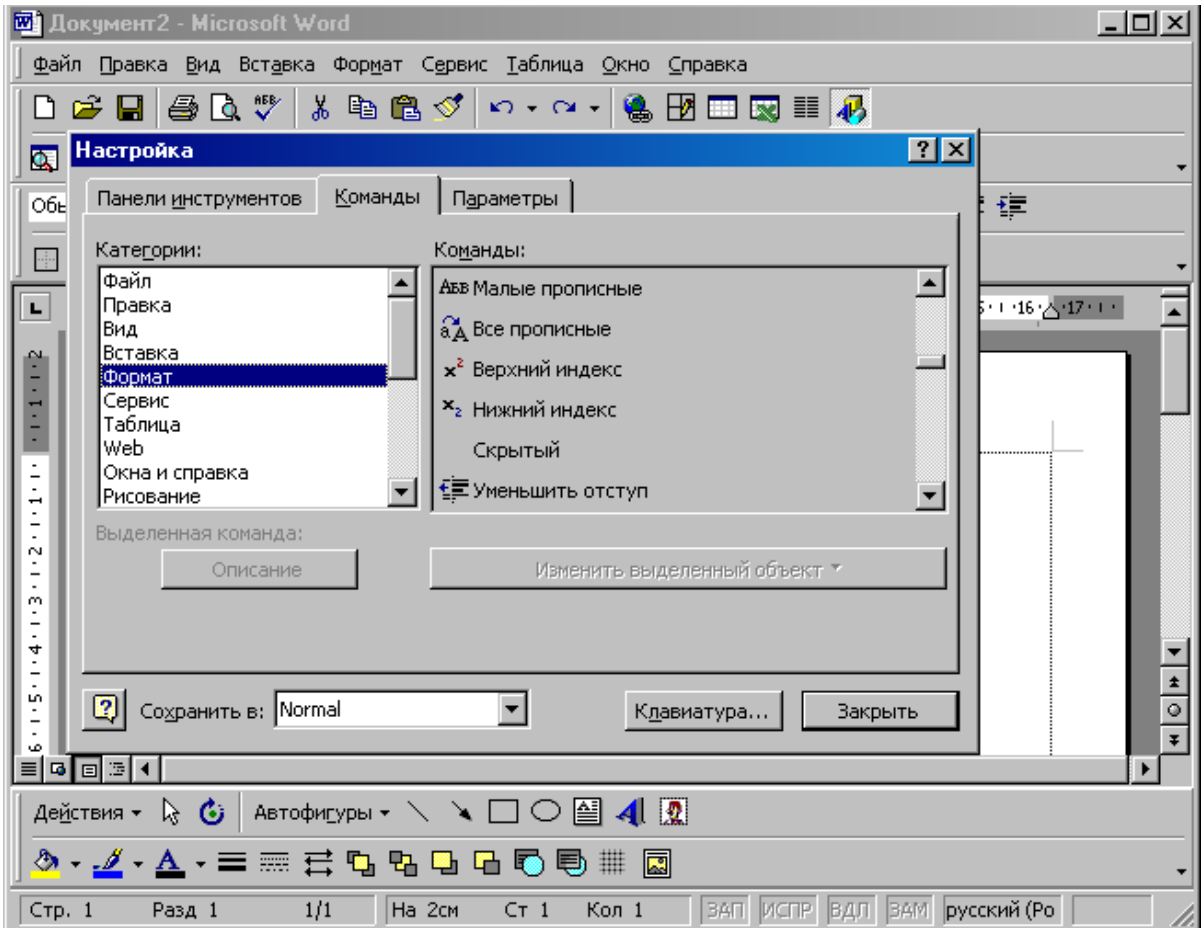


Рис. 1.4. Настройка панели инструментов

Word – это многооконный текстовый редактор, позволяющий редактировать несколько документов. Переключение между документами выполняется через вкладку **Окно** управляющего меню, в котором отображаются все имена файлов типа .doc, открытые на данный момент.

Ввод текста в Word осуществляется построчно с автоматическим переходом на новую строку. После нажатия клавиши «Enter» завершается текущий абзац и начинается новый абзац. Конец абзаца отмечается специальным служебным символом ¶ (рис. 1.5), который не отображается в документе при печати, то есть является непечатным символом. Word использует набор служебных непечатаемых символов для организации документа и облегчения ввода и редактирования текста.

Просмотр непечатных символов осуществляется нажатием кнопки ¶ на панели инструментов либо нажатием сочетания клавиш «Ctrl–*».

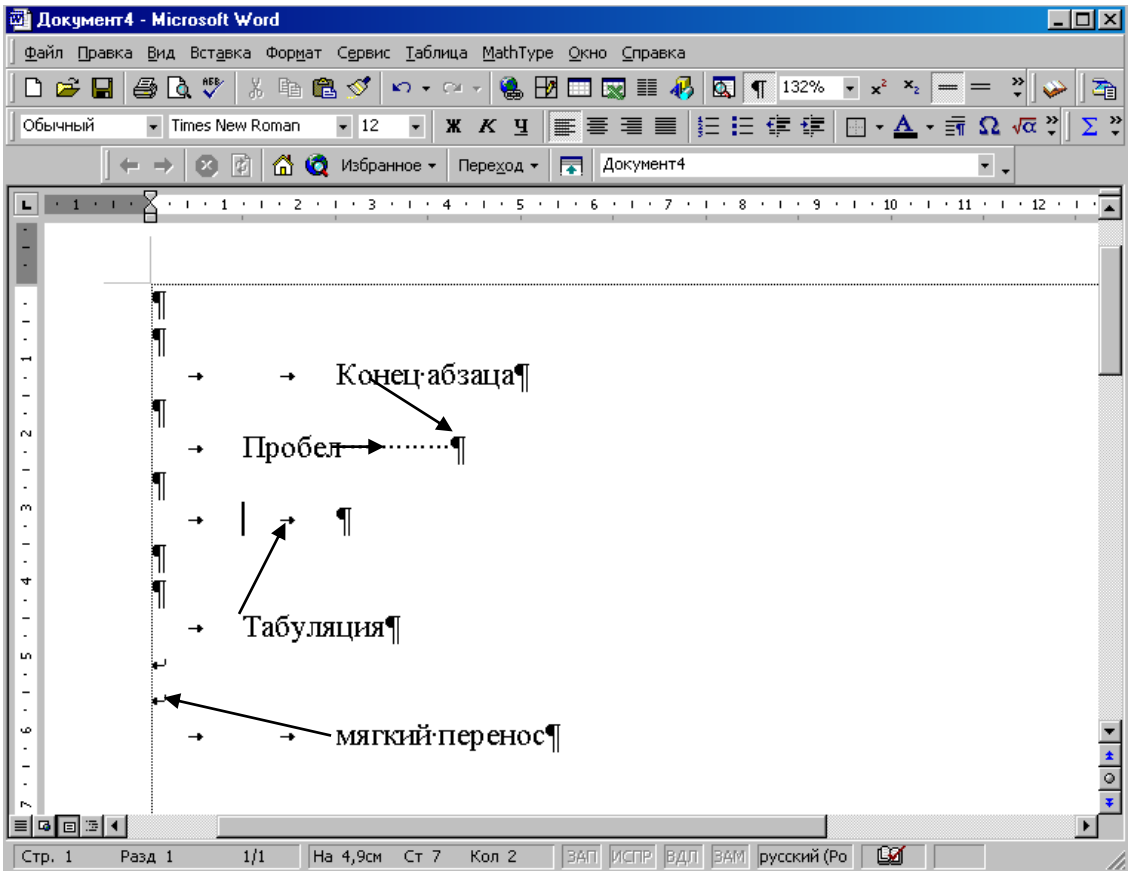


Рис 1.5. Пример непечатаемых символов

Основное назначение Word – создание форматируемых документов. Word предоставляет для организации работы несколько возможностей, а именно: использование шаблонов документов, создание и использование стилей, установление параметров форматирования вручную на текущий сеанс работы и т.д.

Шаблон – это файл, содержимым которого является образец оформления текущего документа. В шаблон входят набор панелей инструментов вместе со стилями, пользовательское меню, макросы (набор команд для выполнения некоторого множества операций). *Стиль* – это часть шаблона, регламентирующая применение совокупности форматов. Любой текст вводится на основе стиля, принятого по умолчанию. *Формат* – это составляющая стиля, регламентирующая правила

создания объекта (символ, слово, строка, абзац и т.д.). В простейшем случае создание форматированного документа связано с установкой форматов:

– размер шрифта;
– тип шрифта;
– способ начертания;
– тип выравнивания;
– отступы левой и правой границ области ввода текста;
– отступ первой строки абзаца;
– межстрочное расстояние;
– ориентация бумаги;
– уровень вводимого текста.

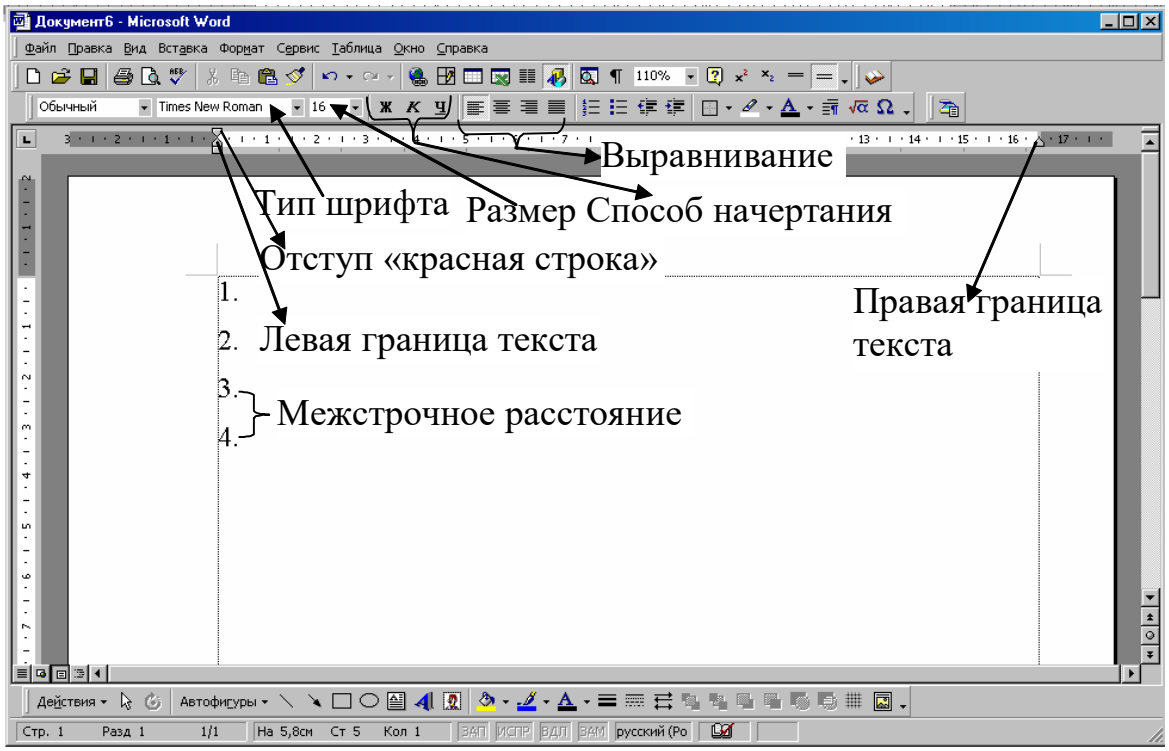


Рис. 1.6. Команды для форматирования текста

Большинство форматов имеет графическое представление в виде значков, расположенных на панели инструментов **Форматирование** (рис. 1.6). Формат **Тип шрифта** определяет написание символов. Стандартным шрифтом, соответствующим шрифту печатной машинки, считается Times New Roman. Стандартный **размер шрифта** составляет 14 или 16 пунктов. Формат **Способ начертания** (представлен тремя значками ***Ж, К,***

Ч) определяет выделенное (**Ж**), наклонное (***К***) начертание или начертание с подчеркиванием (**Ч**). Формат **Тип выравнивания** определяет вид вводимого текста по правой и левой границам рабочей области экрана.

Вместе с тем отдельные форматы описываются в управляющем меню во вкладках **Формат** и **Файл**. Вкладка **Формат** в категории **Абзац** содержит такие способы оформления текста, как **межстрочное расстояние**, **первая строка**, **уровень текста**. В категории **Параметры** страницы вкладки **Файл** содержатся форматы **отступов полей печати** и варианты **ориентации бумаги** (рис. 1.7). Вкладка **Шрифт** категории **Формат** содержит способы оформления текста, такие как вид шрифта, размер, цвет, интервалы, варианты видоизменения (верхний/нижний индекс, зачеркнутый, контур и т.д.).

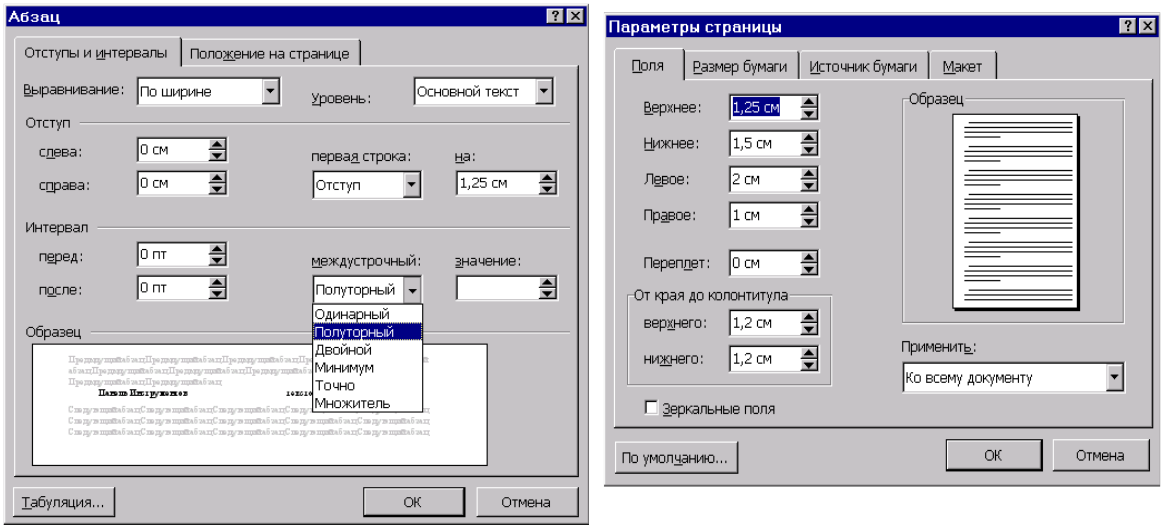


Рис. 1.7. Форматирование абзаца

Особенность задания форматов определяется тем, что действие форматов распространяется либо на вводимый текущий текст, либо на выделенный текст, введенный раньше.

Для работы с файлами в управляющем меню существует вкладка **Файл**, в которой расположены команды **Создать**, **Открыть файл**, **Сохранить**, **Сохранить как.** Первая команда предназначена для создания нового файла. Команда **Открыть файл** осуществляет выбор существующего файла для последующего редактирования. Выбор выполняется обычным

способом, принятым в Windows. По умолчанию выбор файла ограничен установленным фильтром (тип файлов) *документы Word*, который может быть изменен (рис. 1.8).

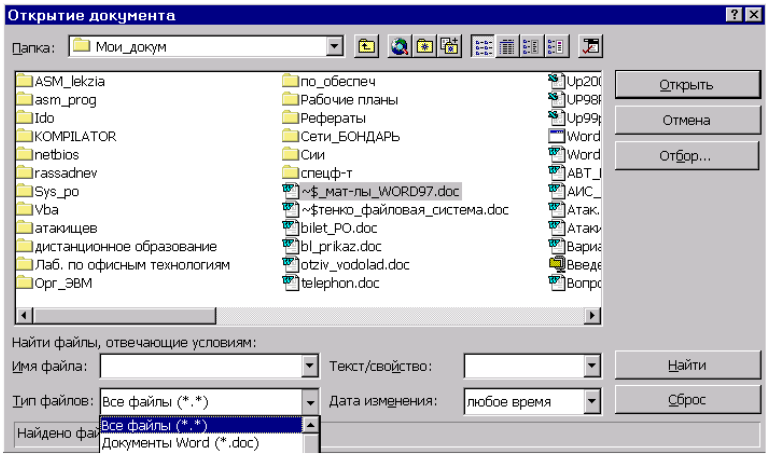


Рис. 1.8. Операция открытия документа типа Word

Команды **Сохранить**, **Сохранить как** предназначены для сохранения редактируемого документа в файл (рис. 1.9). Отличие между ними состоит в том, что последняя команда позволяет сохранить документ под новым именем, причем исходная версия сохранится под прежним именем.

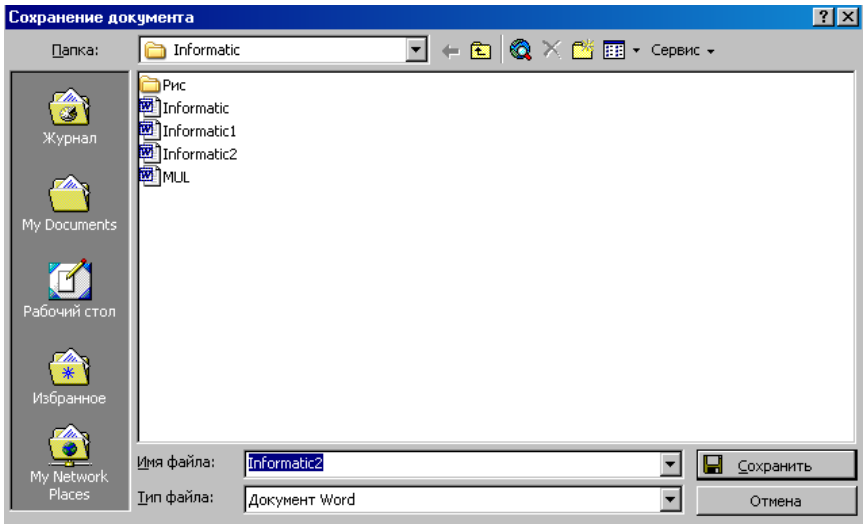


Рис. 1.9. Операция сохранения файлов

Создание списков облегчает восприятие перечислений. Списки подразделяются на нумерованные, маркированные и многоуровневые. Для создания списков можно использовать команду управляющего меню **Формат\Список** или значки

панели инструментов  . Причем, при использовании значков создания списков, список формируется ранее созданного формата. Используя окошко **Список** (рис. 1.10) пользователь может изменять параметры списка.

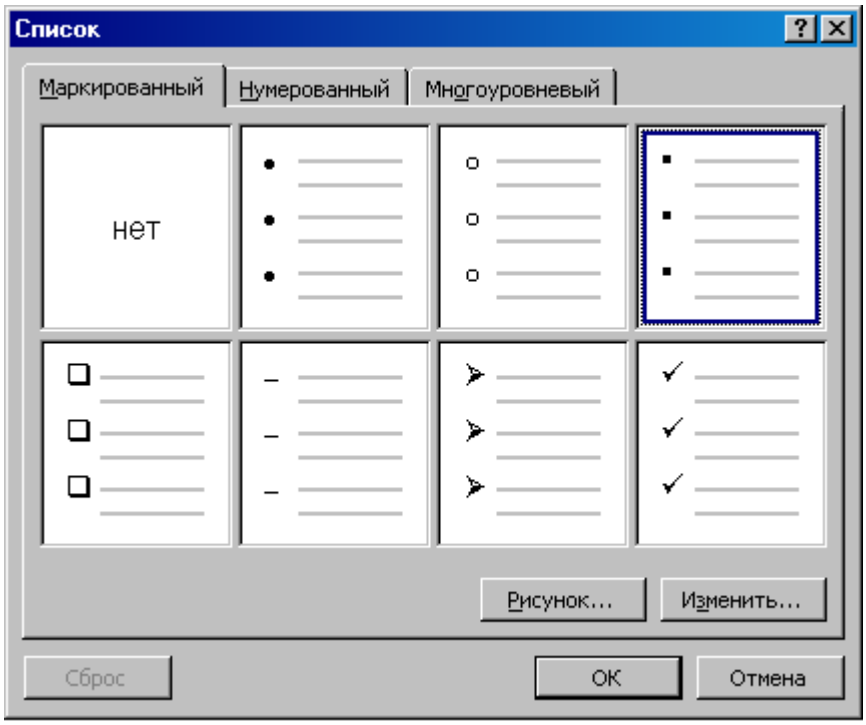


Рис. 1.10. Создание списков

Создание многоуровневого списка (рис. 1.11) выполняется в 2 этапа: сначала задается шаблон будущего списка; затем вводятся значения списка. Создание шаблона выполняется командой **Формат\Список\Многоуровневый\Изменить** и состоит в определении маркера списка для каждого уровня, его положения и отступа текста. Место элемента списка в многоуровневой иерархии определяется величиной отступа, которая задается командой **Увеличить отступ**, **Уменьшить отступ**  .

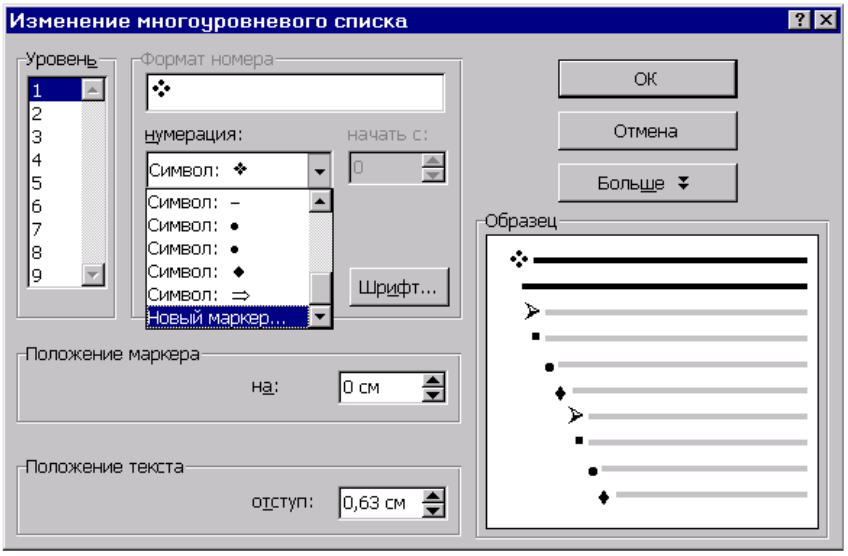


Рис. 1.11. Создание многоуровнего списка

Для работы с таблицами в управляющем меню имеется вкладка **Таблица,** содержащая все команды по созданию таблиц, редактированию данных и форматированию ячеек. Существует 2 способа создания таблиц (рис. 1.12):

1) создание стандартной таблицы командой **Добавить таблицу;**

2) создание нестандартной таблицы командой **Нарисовать таблицу.**

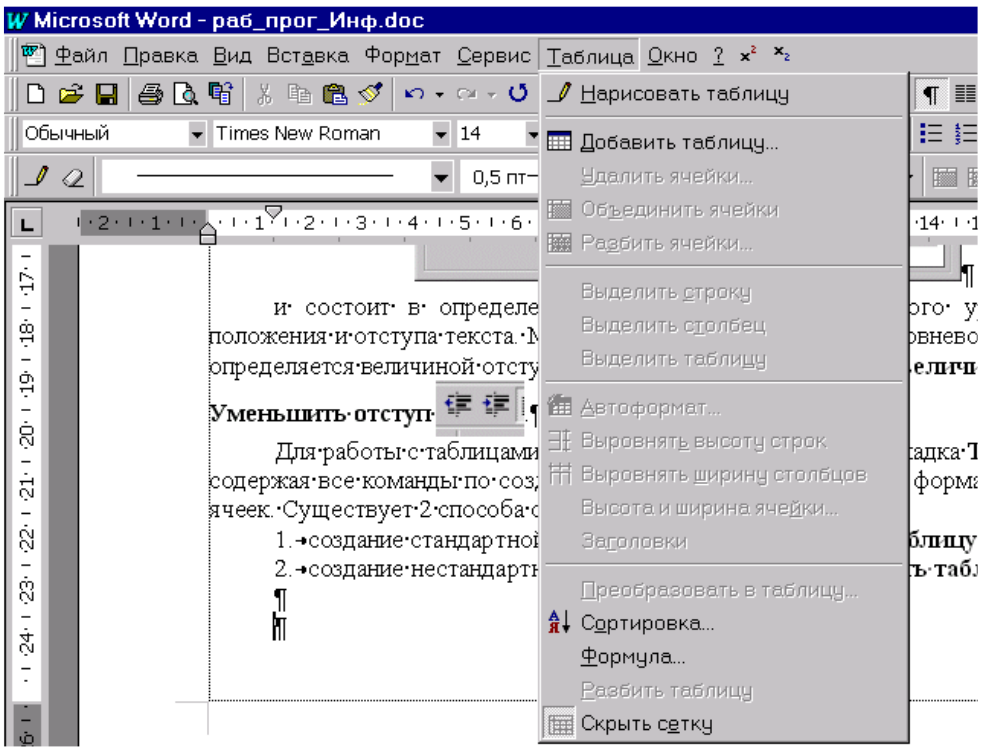


Рис. 1.12. Создание таблицы

Под *стандартной* таблицей условимся понимать таблицу из фиксированного количества ячеек (рис. 1.13).

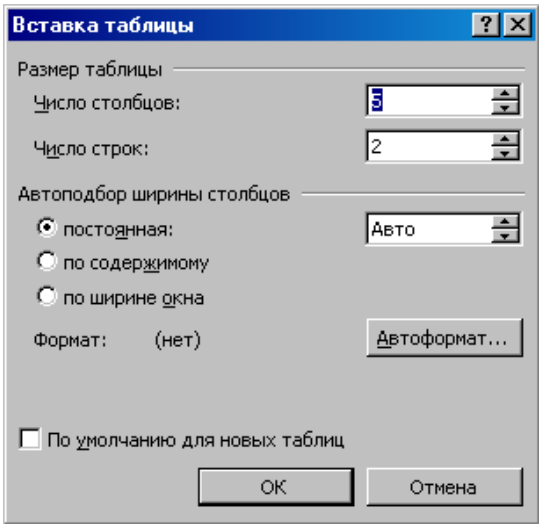


Рис. 1.13. Добавление стандартной таблицы

Под *нестандартной* таблицей понимают таблицу без фиксированного числа горизонтальных и вертикальных ячеек. Выбрав команду **Нарисовать таблицу,** указатель мыши приобретает форму карандаша, с помощью которого сначала рисуются внешние границы таблицы (прямоугольник), а затем произвольные вертикальные и горизонтальные разграничительные линии (рис. 1.14).

Рис. 1.14. Создание нестандартной таблицы

Для работы с нестандартной таблицей используется панель инструментов **Таблицы и границы** (рис. 1.15), в которой, в частности, имеются команды вертикального выравнивания текста в ячейках и изменения направления ввода текста.

Рис. 1.15. Панель инструментов Таблицы и границы

Таблица состоит из ячеек, которые можно заполнять текстом или графикой. Текст вводится внутри в каждой ячейки с автоматическим переходом на новую строку точно так же, как и при вводе в обычный документ. Переход между ячейками таблицы осуществляется с помощью клавиши **<Tab>** для перемещения слева–направо, либо **<Shift+Tab>** для перемещения справа–налево. Добавление (удаление) строк (столбцов) производится с помощью меню **Таблица** вкладки **Добавить** (**Удалить**). В этих вкладках необходимо выбрать определенное действие.

Работа с графикой в Word сводится к импорту графических изображений или созданию рисунков с помощью встроенного графического редактора. Вставка рисунка из файла выполняется командой **Вставка\Рисунок\Из файла.** В открывшемся диалоговом окне с помощью обычных средств следует вывести список графических файлов и выбрать нужный файл. В состав Word входят стандартные комплекты графических файлов, располагаемых в каталоге CLIPART.

Для создания графических рисунков используется панель инструментов **Рисование** (рис. 1.16).

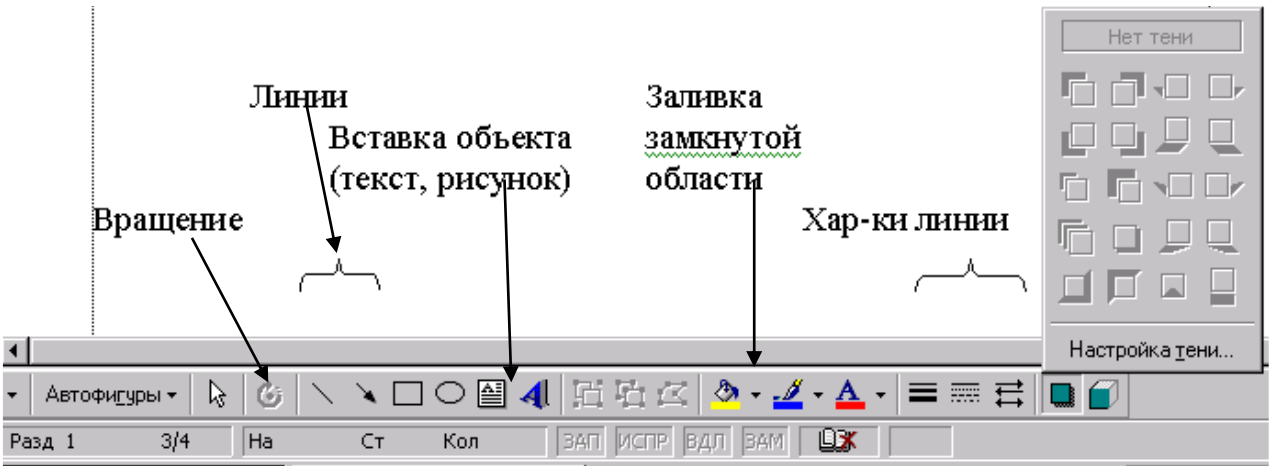


Рис. 1.16. Панель инструментов Рисование

Элементом панели рисование является инструмент WordArt (рис. 1.17), предназначенный для оформления заголовков и

создания спецэффектов со строками. Вызов WordArt связан с выбором из коллекции надписей нужного варианта.

Рис. 1.17. Инструмент WordArt

В дальнейшем необходимо задать текст и отформатировать его. Например, применение WordArt позволит получить например, заголовок, изображенный на рис. 1.18.

Рис. 1.18. Текст, полученный с помощью инструмента WordArt

Таким образом, в настоящем пункте учебного пособия были рассмотрены основные функции и возможности текстового редактора *Microsoft Word*, позволяющие подготавливать, редактировать и оформлять текстовую документацию с использованием таблиц, списков, графики и рисунков.

Выводы?????

1.2. Электронные таблицы Microsoft Excel

Программа обработки данных, обеспечивающая взаимодействие с пользователем при помощи выводимых на экран дисплея прямоугольных таблиц и работающая в

диалоговом режиме называется **электронной таблицей**[4,5].

Данные, входящие в таблицы, можно автоматически представлять в виде графиков, диаграмм, гистограмм и т.д.

Пользователь работает в диалоге со специальной программой, которая позволяет заполнять ячейки нужным ему содержимым (текстами, числами или формулами для расчетов), очищать их, копировать и удалять, сортировать (т.е. располагать клетки, а также строки и столбцы в определенном порядке), производить вычисления над всей таблицей или ее частью, сохранять таблицу на диске и распечатывать частично или полностью на бумагу и т.д.

После запуска Excel экран содержит пять областей: окно книги, которое занимает большую часть экрана, строку меню, две или несколько панелей инструментов, строку формул и строку состояния. Вместе эти пять областей называются *рабочей областью Excel*.

Окно книги (рис. 1.19) составляет основную часть рабочей области. **Рабочая книга** – документ приложения Excel. Книга состоит из нескольких рабочих листов, каждый из которых является электронной таблицей с широкими возможностями обработки данных. В нижней части окна книги размещаются ярлычки листов и кнопки их прокрутки, а в верхней части – строка заголовка. Кроме того, окно содержит листы и полосы прокрутки. Новая книга, показанная на рисунке, первоначально содержит три отдельных листа.

В нижней части окна книги находится несколько кнопок, с помощью которых вы можете переходить от одного листа к другому.

Когда видны не все ярлычки, для просмотра содержания книги можно использовать четыре кнопки, расположенные в нижнем левом углу окна. Две средние кнопки служат для прокрутки на один ярлычок влево или вправо. Крайние кнопки выполняют прокрутку к первому или последнему ярлычку книги. Можно изменить количество видимых на экране ярлычков, перетащив маркер разделения ярлычков. Чтобы восстановить исходное положение маркера разделения ярлычков, достаточно дважды щелкнуть на нем.

Перечисленные кнопки прокрутки и маркер разделения ярлычков не активизируют листы книги. Чтобы сделать лист активным, вы должны после прокрутки ярлычков щелкнуть на ярлычке нужного листа.

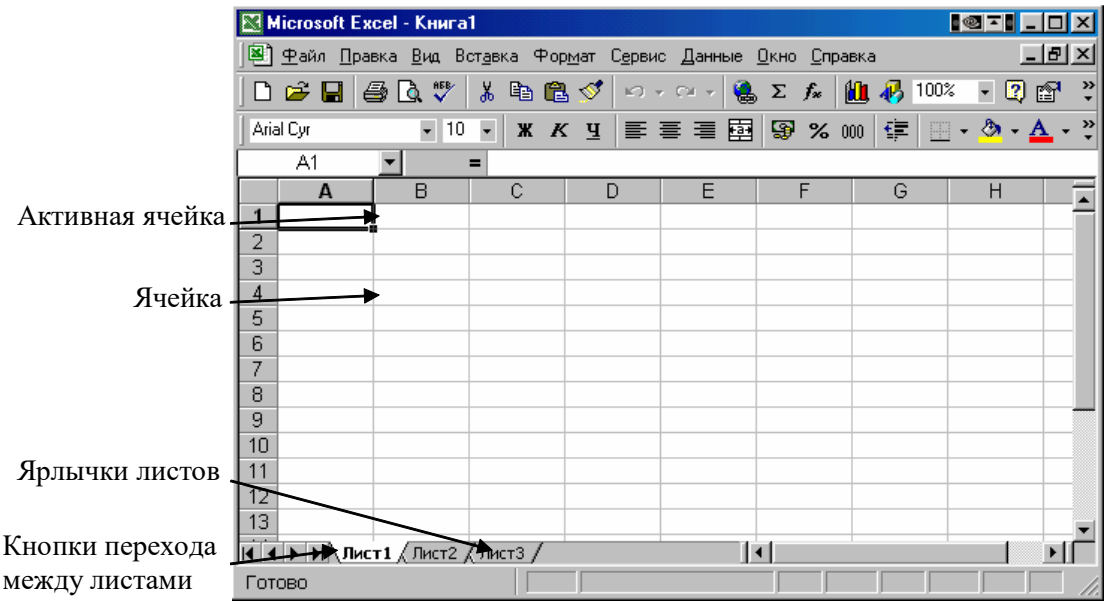


Рис. 1.19. Внешний вид окна Microsoft Excel

Рабочий лист книги разделен линиями сетки на строки и столбцы. Каждому столбцу соответствует определенная буква, которая появляется в качестве его заголовка. Заголовки столбцов могут принимать значения в диапазоне от A до IV (после столбца Z идет столбец AA, после AZ идет BA и так далее вплоть до IV). Каждой строке назначается целое число, которое выводится слева от сетки листа в качестве заголовка строки. Номера строк могут изменяться от 1 до 65 536.

На пересечении строки и столбца находится *ячейка (cell).* Ячейки являются основными строительными блоками любого рабочего листа. Каждая ячейка занимает свое место на листе, где можно хранить и отображать информацию, и имеет уникальные координаты, которые называются *адресом ячейки* или *ссылкой*. Например, ячейка, находящаяся на пересечении столбца A и строки 1, имеет адрес A1. Ячейка на пересечении столбца Z и строки 100, имеет адрес Z100. Выделенную ячейку называют *активной ячейкой (active cell).* Адрес активной ячейки выводится в поле имени, которое находится в левом конце строки формул.

Полосы прокрутки

Для просмотра содержимого листа можно использовать полосы прокрутки, расположенные вдоль правой и нижней сторон окна книги. Полосы прокрутки имеются только у активного окна книги, то есть у окна, в котором вы работаете.

Стрелки прокрутки на концах полос прокрутки позволяют за один раз перемещать лист на одну строку или один столбец. Щелчок на стрелке ▲ или стрелке ▼ на вертикальной полосе прокрутки сдвигает лист на одну строку вверх или вниз соответственно. Аналогично, щелчок по стрелке ► или стрелке ◄ на горизонтальной полосе прокрутки перемещает лист на один столбец вправо или влево. Прокрутка при нажатой клавише <*Shift*> позволяет в большом листе быстро перемещаться к столбцам или строкам, находящимся за пределами активной области.

Строка формул

Информацию можно вводить непосредственно в ячейку или с помощью строки формул. Содержимое активной ячейки всегда появляется в строке формул, это особенно важно, когда ячейка содержит формулу, как это показано на рис. 1.20.

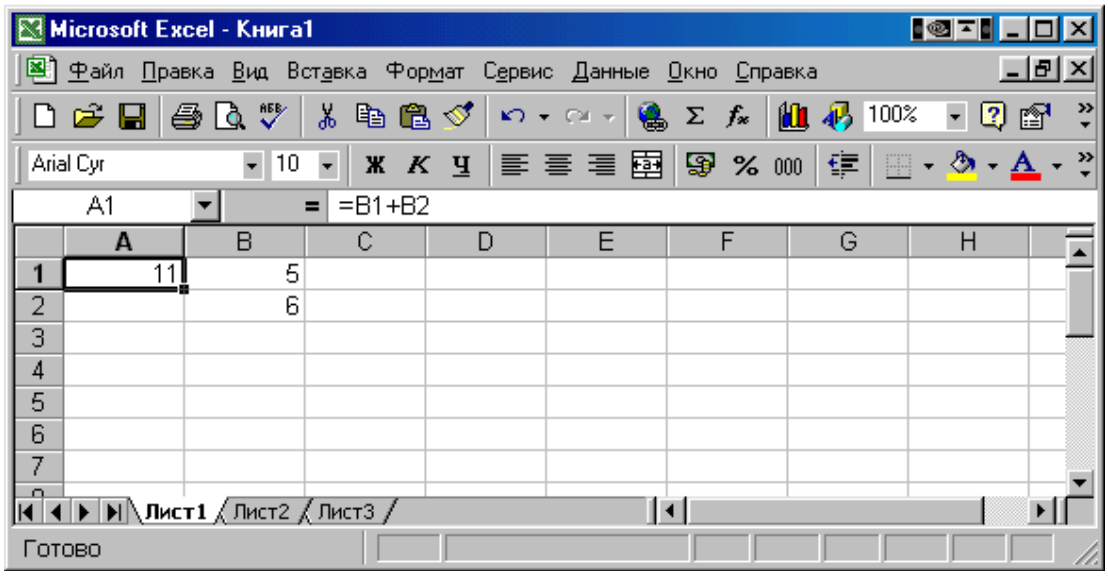


Рис. 1.20. Операции с формулами в Microsoft Excel

Кнопка = (Изменить формулу) отображается в строке формул только в случае ввода или редактирования данных в ячейке.

Хотя вы можете вводить информацию непосредственно в ячейку, использование строки формул имеет некоторые преимущества. Когда вы устанавливаете указатель на строке формул и нажимаете кнопку мыши, в строке появляются три кнопки. Кнопка с символом × называется кнопкой отмены, а кнопка ✔ - кнопкой ввода. При нажатии кнопки ввода Excel «фиксирует» введенную в строке формул информацию и переносит ее в лист. Нажатие кнопки ввода аналогично нажатию клавиши <Enter> за исключением того, что нажатие <Enter> обычно дополнительно активизирует ячейку, находящуюся непосредственно ниже той, в которую вы вводили данные. Для удаления ошибочно введенной информации нажмите кнопку отмены или клавишу <Esc>.

Прежде чем выполнять какие-либо операции с ячейками, необходимо выделить ячейку или группу ячеек. Для выделения одной ячейки необходимо установить на ней указатель и нажать левую кнопку мыши. Вокруг ячейки появится рамка, показывающая, что данная ячейка является активной, а в поле имени будет выведен ее адрес.

Выделение диапазонов осуществляется несколькими способами. Вместо перетаскивания указателя мыши по ячейкам, с которыми предстоит работать, необходимо указать любые два противоположных (по диагонали) угла диапазона. Этот прием известен как *расширение выделения.* В окне можно увидеть только часть рабочего листа, величина которой зависит от размеров экрана и его разрешения. Если необходимо выделить диапазон, выходящий за пределы окна книги, необходимо перетащить указатель за границы окна.

При необходимости можно перетащить указатель влево или вправо за границы окна книги, чтобы вывести на экран дополнительные столбцы и сделать нужное выделение.

Выделение больших диапазонов путем перетаскивания указателя занимает много времени. Существует другой способ выделения диапазона. Чтобы выделить, к примеру, диапазон A1:M38, проделайте следующие действия:

1) щелкните на ячейке A1;
2) пользуясь полосой прокрутки, выведите на экран ячейку

M38;

3) нажмите клавишу <F8> (или <Shift>);

4) щелкните на ячейке M38.

При выделении нескольких диапазонов с помощью мыши используется клавиша <Ctrl>.

Чтобы выделить столбец или строку целиком, следует щелкнуть на заголовке столбца или строки. Первая видимая ячейка становится активной.

Чтобы ввести числовое значение, следует выделить ячейку и ввести с клавиатуры число. Вводимые цифры отображаются в строке формул и в активной ячейке. Мигающая вертикальная черта, которая появляется в строке формул и в активной ячейке, называется *точкой вставки* (insertion point).

По окончании ввода значения необходимо его зафиксировать, чтобы оно постоянно хранилось в ячейке. Простейшим способом фиксации ввода является нажатие клавиши <Enter>. После этого точка вставки исчезает, и Excel сохраняет введенное значение в ячейке.

Ниже дается описание символов, которые в Excel имеют специальное значение.

1. Если вы начинаете ввод числа со знака «плюс» (+) или «минус» (–), Excel опускает «плюс» и сохраняет «минус», в последнем случае интерпретируя введенное значение как отрицательное число.

2. Символ E или e используется при вводе чисел в экспоненциальном представлении. Например, Excel интерпретирует 1E6 как 1000000 (единица, умноженная на десять в шестой степени).

3. Числовые значения, заключенные в круглые скобки, Excel интерпретирует как отрицательные. (Такая запись отрицательных чисел используется в бухгалтерском учете.) Например, значение (100) воспринимается Excel как -100.

4. Можно использовать десятичную запятую, как обычно. Кроме того, позволяется вставлять пробел для отделения сотен от тысяч, тысяч от миллионов и т.д. Если ввести числа с пробелами, разделяющими группы разрядов, то в ячейках они появляются вместе с пробелами, а в строке формул - без

пробелов.

5. Если начать ввод числа со знака доллара ($), Excel применяет к ячейке денежный формат.

6. Если ввод числового значения заканчивается знаком процента (%), Excel применяет к ячейке процентный формат.

7. Если при вводе числового значения используется наклонная черта (/) и введенная строка символов не может быть интерпретирована как дата, то Excel рассматривает введенное значение как дробь.

Числовое значение отображается в ячейке не более чем 15 цифрами. Если вводится длинное числовое значение, которое не может быть выведено в ячейке, то Excel использует экспоненциальное представление числа. При этом точность значения выбирается такой, чтобы число можно было отобразить в ячейке.

Значения, которые появляются в ячейке, называются *выводимыми* или *отображаемыми* значениями (displayed values). Значения, которые хранятся в ячейках и появляются в строке формул, называются *хранимыми* значениями (underlying values). Количество выводимых цифр зависит от ширины столбца. Если эта ширина недостаточна для вывода числа, то в зависимости от используемого формата отображения Excel может вывести либо округленное значение, либо строку символов #. Для отображения числа необходимо расширить столбец.

Ввод текста аналогичен вводу числовых значений. Чтобы ввести текстовое значение, необходимо выделить ячейку, набрать на клавиатуре текст и зафиксировать ввод, нажав клавишу <Enter> или кнопку ввода. Чтобы отказаться от ввода, следует нажать клавишу <Esc> или кнопку отмены.

Если текст не может быть полностью отображен в одной ячейке, Excel может вывести его, перекрывая соседние ячейки. Но текст при этом хранится в одной ячейке. Когда вводится текст в ячейку, которая перекрыта содержимым другой ячейки, то перекрывающий текст выглядит обрезанным.

Для переноса текста следует выделить ячейку, выбрать в меню **Формат (Format)** команду **Ячейки (Cells)** и на вкладке

Выравнивание (Alignment) открывшегося окна диалога установите флажок **Переносить по словам (Wrap Text)**, затем нажать кнопку ОК.

Использование формул

Для ввода формул необходимо сначала выделить пустую ячейку, затем ввести знак “=”, после чего ввести требуемую формулу, например “=10+5*2”. Далее следует нажать клавишу <Enter>. В ячейке A10 появится значение 20, но при выделении ячейки A10 в строке формул будет выведена только что введенная формула.

Приоритет операторов

Термин «приоритет» (precedence) обозначает последовательность, в которой Excel обрабатывает операторы в формуле. Excel руководствуется следующими правилами:

• в первую очередь вычисляются выражения внутри круглых скобок;

• умножение и деление выполняются раньше сложения и вычитания;

• операторы с одинаковым приоритетом выполняются слева направо.

Использование ссылок в формулах

Ссылка (cell reference) является идентификатором ячейки или группы ячеек в книге. Создавая формулу, содержащую ссылки на ячейки, вы связываете формулу с ячейками книги. Значение формулы зависит от содержимого ячеек, на которые указывают ссылки, и оно изменяется при изменении содержимого этих ячеек.

В качестве упражнения введите формулу, которая содержит ссылку на ячейку. Выделите ячейку A1 и введите следующее:

= 10*2

Теперь выделите ячейку A2 и введите формулу:

=A1

Значения в обеих ячейках равны 20. Если вы измените значение в ячейке A1, значение в ячейке A2 также изменится.

Теперь выделите ячейку A3 и введите:

=A1+A2

Excel выведет значение 40. В дальнейшем вы убедитесь,

что ссылки чрезвычайно полезны при создании и использовании сложных формул.

Ввод ссылок с использованием мыши

Например, чтобы ввести в ячейку B10 формулу со ссылками на ячейки A9 и A10, выполните следующие действия:

1) выделите ячейку B10 и введите знак равенства;

2) щелкните на ячейке A9 и введите знак «плюс»;

3) щелкните на A10 и нажмите клавишу <Enter>.

Вводить значение в активную ячейку можно даже тогда, когда она не видна на экране. При вводе формулы можно прокручивать лист без изменения активной ячейки и щелкать на ячейках, расположенных в дальних областях листа. Какая бы часть листа ни была видна на экране, в строке формул выводится содержимое активной ячейки.

Относительная ссылка (Relative reference) указывает на ячейку, основываясь на ее положении относительно ячейки, в которой находится формула, например, «на две строки выше». Ссылки именно такого типа мы использовали в предыдущих примерах. **Абсолютная ссылка** использует для указания на ячейку ее фиксированное положение на листе, например «ячейка находится в столбце A строки 2». **Смешанная ссылка** содержит относительную и абсолютную ссылку: например, «ячейка находится в столбце A и выше на две строки». Абсолютные и смешанные ссылки особенно полезны при копировании формулы из одного места листа в другое.

Относительная ссылка на ячейку A1 записывается так: =A1, а абсолютная ссылка на ячейку A1 имеет следующий вид: =A1.

Комбинируя абсолютные и относительные ссылки на ячейку A1, можно создать следующие смешанные ссылки: =$A1, =A$1.

Если символ доллара стоит перед буквой, то координата столбца абсолютная, а строки – относительная. Если символ доллара стоит перед числом, то, напротив, координата столбца относительная, а строки – абсолютная.

Формулы и функции

Все вычисляемые функции делятся на формулы и

встроенные функции (рис. 1.21). Как уже отмечалось ранее, ввод формул и функций выполняется в строке формул или непосредственно в ячейке и всегда должен начинаться с «=».

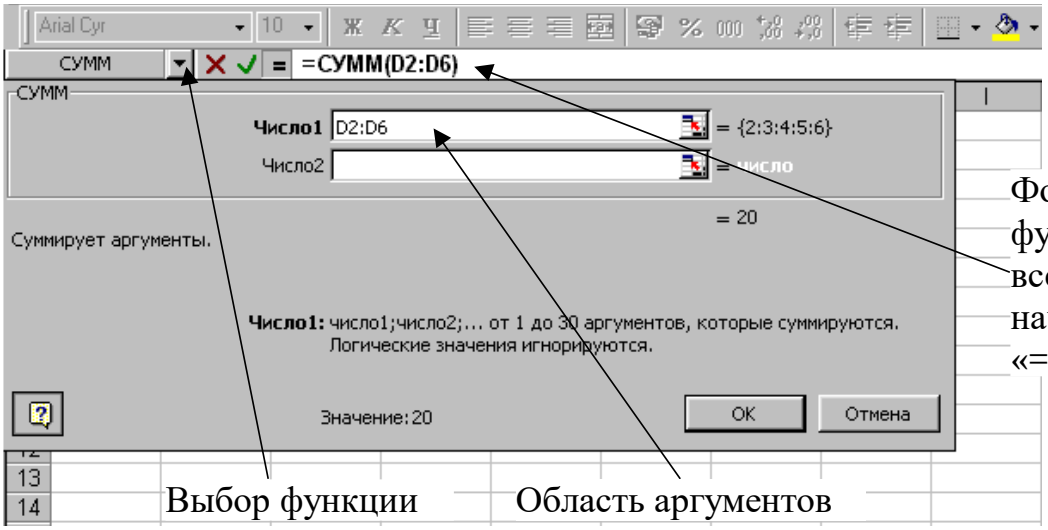


Рис. 1.21. Использование формул и функций в Excel

Операции, используемые в формулах

Основные операции, которые используются при создании формул, приведены в таблице 1.1.

Таблица 1.1.

Основные операции создания формул

Операция	Обозначение	Назначение
Арифметические	+, -, *, /, %, ^	
Текстовые	&	сцепление
Сравнения	=, <, >, <=, >=, <>	
Адресные	двоеточие (:), запятая (,) пробел ( )	: - диапазон , - объединение СУММ (A12, F4)

Условное форматирование

Под «условным форматированием» (рис. 1.21) понимается написание данных от их значения (ФОРМАТ\УСЛОВИЕ–ФОРМАТИРОВАНИЕ). Преимущества условного форматирования следующие:

– не требует знания числового значения;

– содержит большое количество элементов форматирования;

– имеет возможность задания трех логических условий;

– в качестве условия использует логические формулы.

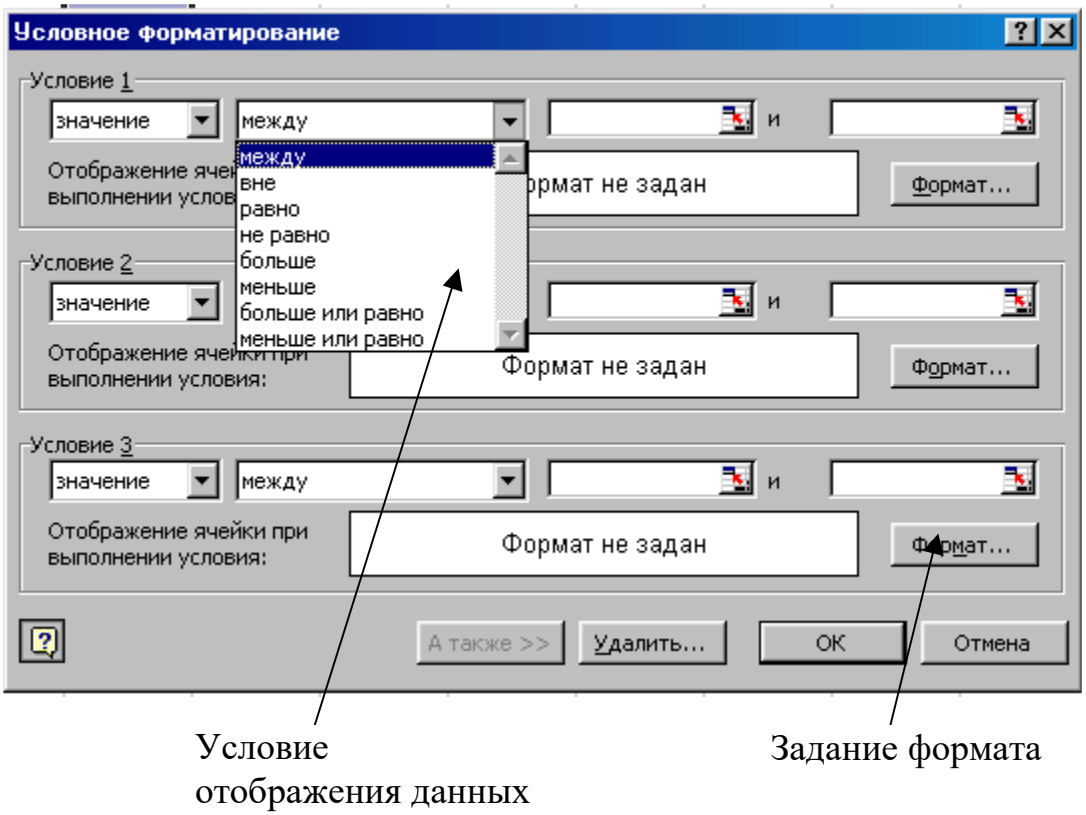


Рис. 1.21. Использование мастера условного форматирования в Excel

Список некоторых функций

ЕСЛИ (условие, знач. ДА, знач. НЕТ)

СУММ (ЧИСЛО 1, число2, ...)

СУММЕСЛИ (диапазон, условие) - суммирование ячеек при выполнение условия

СРЗНАЧ (число1, число2, ...) - вычисление среднеарифметического аргумента

СЧЕТ (значение1, значение2, ...) - подсчет количества чисел среди аргументов

СЧЕТА (значение1, значение2, ...) - подсчет количества непустых значений среди аргументов

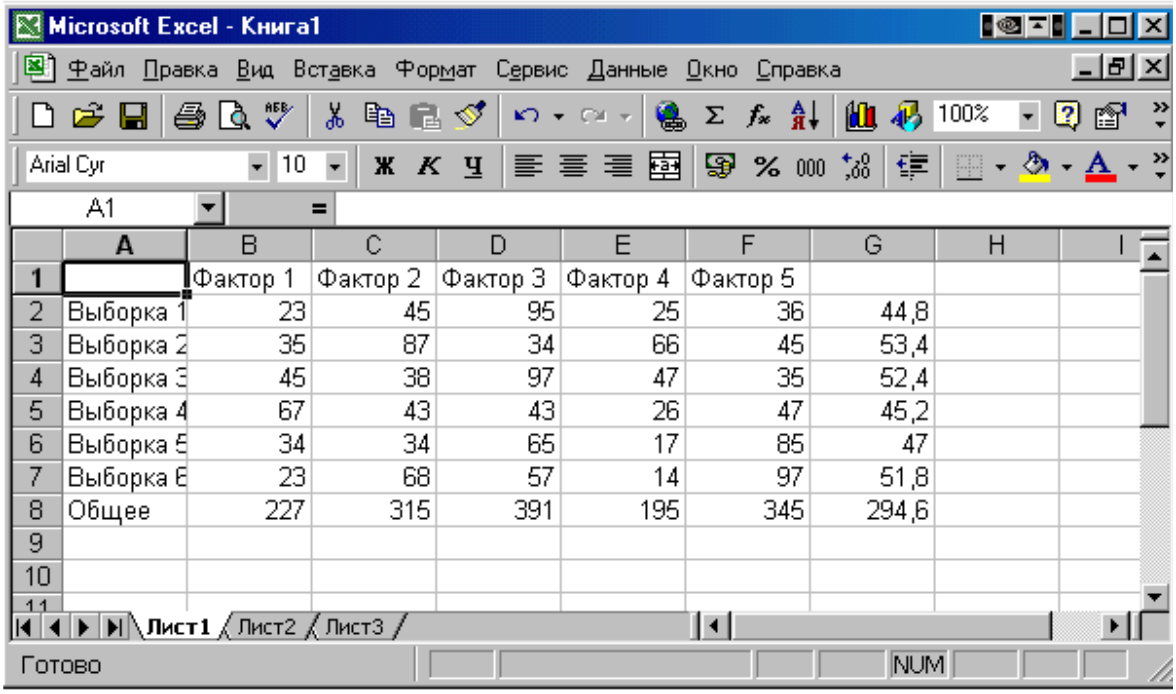
СЧЕТЕСЛИ (диапазон, условие) - подсчет кол-ва ячеек в диапазоне, соответствующем условию

ТЕКСТ (значение, строка_формат) - преобразование числового значения в текст с указанным форматом

ДЛСТР (текст) - подсчет длины строки
МАКС (ЧИСЛО 1, число2, ...)
МИН (ЧИСЛО 1, число2, ...),
НД() - возврат значения ошибки #Н/Д, трактуемое как «нет доступного значения». Эту функцию следует использовать в ячейках, от которых зависят вычисления. В случае отсутствия значения вместо формулы будет выведено #н/д.

Форматирование листов

Команда **Ячейки (Cells)** меню **Формат (Format)** контролирует большинство форматов, применяемых к ячейкам рабочего листа. Выполнить форматирование ячеек очень легко: просто выделите ячейку или диапазон и выберите соответствующую команду в меню **Формат**. Например, для форматирования ячеек B2:G8, показанных на рис. 1.22, следует выполнить описанные ниже действия.



	A	B	C	D	E	F	G	H	I
1		Фактор 1	Фактор 2	Фактор 3	Фактор 4	Фактор 5			
2	Выборка 1	23	45	95	25	36	44,8		
3	Выборка 2	35	87	34	66	45	53,4		
4	Выборка 3	45	38	97	47	35	52,4		
5	Выборка 4	67	43	43	26	47	45,2		
6	Выборка 5	34	34	65	17	85	47		
7	Выборка 6	23	68	57	14	97	51,8		
8	Общее	227	315	391	195	345	294,6		
9									
10									

Рис. 1.22. Пример форматирования ячеек при создании таблиц

1. Выделите ячейки B2:G8.
2. В меню **Формат** выберите команду **Ячейки**.
3. В открывшемся окне диалога щелкните на вкладке **Число (Number)**, если она не является активной.
4. В списке **Числовые форматы** выберите числовой формат.
5. Установите в поле **Число десятичных знаков** значение 1.

6. Нажмите кнопку **ОК**, чтобы вернуться в лист.

Форматирование с помощью кнопки <Формат> по образцу

Чтобы скопировать форматы в другое место листа, выполните описанные ниже действия.

1. Выделите ячейку или ячейки, из которых вы хотите скопировать форматы.

2. Нажмите кнопку  (Формат по образцу). Рядом с указателем появится значок маленькой кисти.

3. Выделите ячейку или ячейки, в которые вы хотите скопировать форматы.

Чтобы применить формат с помощью кнопки  панели инструментов, выделите ячейку или диапазон и затем нажмите кнопку, пользуясь мышью. Для удаления формата нажмите эту кнопку снова.

Форматирование чисел и текста

Команды меню **Формат** позволяют управлять отображением числовых значений и изменять вывод текста. Выберите в этом меню команду **Ячейки** (или просто нажмите клавиши *<Ctrl+I>)* и затем в открывшемся окне диалога **Формат ячеек** перейдите на вкладку **Число**, показанную на рис. 1.23.

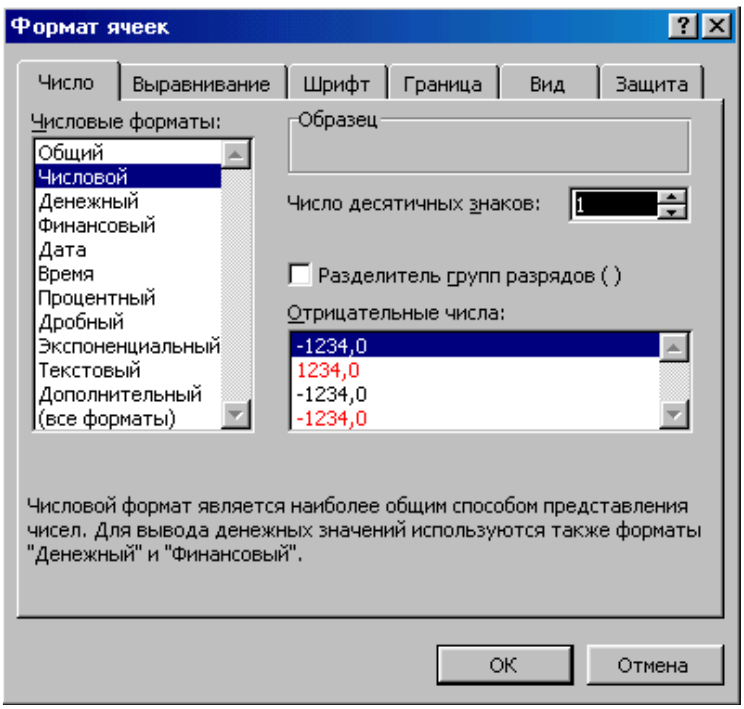


Рис. 1.23. Форматирование ячеек

Никогда не забывайте о различии между хранимыми и отображаемыми значениями. На хранимые числовые или текстовые значения в ячейках форматы не воздействуют. Например, при вводе числа с шестью десятичными знаками в ячейку, которая отформатирована с двумя десятичными знаками, число отображается только с двумя десятичными знаками, но хранимое значение, используемое в вычислениях, не изменяется.

Формат **Общий** - это первая категория в списке **Числовые форматы**. Если вы явно не измените формат ячейки, Excel отобразит любое введенное текстовое или числовое значение в формате **Общий**. За тремя исключениями формат **Общий** отображает точно то, что вы ввели в ячейку. Например, при вводе 123,45 в ячейке будет выведено 123,45. Ниже приведены три исключения:

1) длинные числовые значения отображаются в экспоненциальной записи или округляются. Например, в ячейке со стандартной шириной формат **Общий** отобразит целое число *12345678901234* как 1.23457E+13. Если ввести значение *123456,7812345* в ячейку со стандартной шириной и применить формат **Общий**, то будет выведено число 123456,8;

2) формат **Общий** не отображает незначащие нули. Например, число 723,0 выводится как 123;

3) десятичная дробь, введенная без числа слева от десятичной запятой, выводится с нулем. Например, ,123 выводится как 0,123.

Выравнивание содержимого ячеек

Вкладка **Выравнивание** окна диалога **Формат ячеек** (рис. 1.24), контролирует расположение текста и чисел в ячейках. Эту вкладку можно также использовать для создания многострочных надписей, повторения ряда символов в одной или нескольких ячейках, изменения ориентации текста.

При выравнивании по левому краю вы можете изменять величину отступа, которая по умолчанию принимается равной нулю. При увеличении отступа на одну единицу значение в ячейке смещается на ширину одного символа вправо. Максимальная величина отступа составляет 15 единиц.

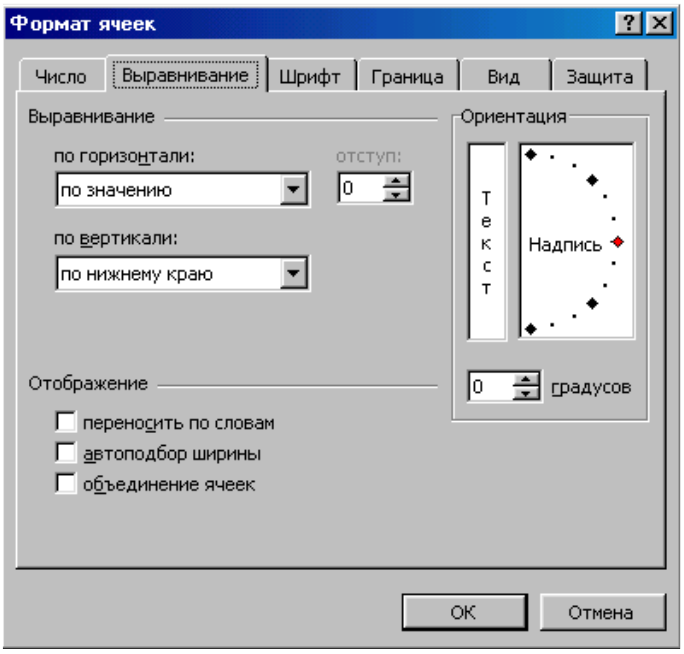


Рис. 1.24. Вкладка **Выравнивание** окна **Формат ячеек**

Формат **По центру выделения** позволяет центрировать текст от ячейки вдоль всех выделенных справа пустых ячеек до края выделения или до следующей непустой ячейки в выделении.

Область Ориентация позволяет размещать значения горизонтально (по умолчанию), вертикально сверху вниз или наклонно под углом до 90 градусов по часовой или против часовой стрелки. Excel автоматически настраивает высоту строки при вертикальной ориентации в том случае, если вы сами ранее или впоследствии не установите высоту строки вручную. Чтобы возвратить выделенные ячейки к их обычной ориентации, установите в горизонтальное положение ползунок **Надпись** в области **Ориентация**.

Флажок **Автоподбор ширины** в области **Отображение** на вкладке **Выравнивание** уменьшает размер символов в выделенной ячейке так, чтобы ее содержимое полностью помещалось в столбце. Это может быть полезно при работе с листом, в котором настройка ширины столбца по длинному значению имеет нежелательный эффект для остальных данных, или в том случае, когда использование вертикального или наклонного текста, переноса по словам является неприемлемым решением.

Изменение шрифта

Понятие *шрифт* включает в себя *гарнитуру,* например Helvetica, и ряд других атрибутов (размер, цвет и т. д.). В Excel выбор шрифта осуществляется на вкладке **Шрифт** окна диалога **Формат ячеек**. В рабочем листе шрифты используются для наглядного оформления информации различного типа и, в частности, для выделения заголовков. Чтобы задать шрифт для ячейки или диапазона, сначала выделите эту ячейку или диапазон. В меню **Формат** выберите команду **Ячейки** (или нажмите клавиши *<Ctrl+I>)* и затем перейдите на вкладку **Шрифт**.

Таким образом, применяя представленную информацию из первой главы, вы сможете качественно оформлять текстовую документацию на компьютере, используя при этом основные возможности прикладной программы Microsoft Word, а также приобретете навыки обработки чисел в электронных таблицах Microsoft Excel и представлять полученную информацию наглядно в виде графиков различных типов.

Выводы - ???

Контрольные вопросы

???

2. ОБЩИЕ СВЕДЕНИЯ О ПРЕДСТАВЛЕНИИ ИНФОРМАЦИИ В ЭВМ

Объектом передачи и преобразования в ЭВМ является *дискретная информация*. Для представления её применяется так называемый алфавитный способ, основой которого является использование фиксированного конечного набора символов любой природы, называемого алфавитом [6].

Примерами таких алфавитов могут служить и алфавиты естественных человеческих языков, совокупность десятичных цифр, любая другая упорядоченная совокупность знаков, предназначенная для образования и передачи сообщений. Символы из набора алфавита называются *буквами*, а любая конечная последовательность букв – *словом* в алфавите. При этом не требуется, чтобы слово обязательно имело языковое смысловое значение. Например, словами являются последовательности символов, составленные из алфавита, включающего буквы и цифры:

2006 AAAA CARCASS SENTENCED ПАСКАЛЬ

Все процессы, происходящие в вычислительной системе, связаны непосредственно с различными физическими *носителями информационных сообщений*, а все узлы и блоки этой системы являются физической средой, в которой осуществляются информационные процессы. Особенности носителя информации накладывают определённые ограничения на используемый для её представления алфавит. При подготовке к решению на ЭВМ исходная формулировка, описание метода решения, задание конкретных исходных данных осуществляются на математическом языке, алфавит которого, наряду с буквами других языков, составляют специальные символы знаков математических операций и другие знаки.

В процессе ввода, хранения, вывода и обработки информации в ЭВМ осуществляется неоднократное её преобразование из одной формы представления в другую. При этом с каждой из используемых форм представления информации связаны различные алфавиты. Для представления

информации на перфокартах и перфолентах используется, например, алфавит, включающий всего две буквы (есть пробивка, нет пробивки). Таким образом, процесс преобразования информации часто требует представлять буквы одного алфавита средствами (буквами, словами) другого алфавита. Такое представление называется *кодированием*.

Декодированием называется процесс обратного преобразования информации относительно ранее выполненного кодирования.

Для представления информации в ЭВМ преимущественное распространение получило *двоичное кодирование*, при котором символы вводимой в ЭВМ информации представляются средствами *двоичного алфавита*, состоящего из двух букв.

Двоичный алфавит по числу входящих в него символов является минимальным, поэтому при двоичном кодировании алфавита, включающего большее число букв, каждой букве ставится в соответствие последовательность нескольких двоичных знаков или двоичное слово. Такие последовательности называют *кодовыми комбинациями*.

Полный набор кодовых комбинаций, соответствующих двоичному представлению всех букв кодируемого алфавита, называется *кодом*.

Различают коды равномерные и неравномерные. Кодовые комбинации равномерных двоичных кодов содержат одинаковое число двоичных знаков, неравномерных – не одинаковое.

Примером неравномерного двоичного кода может служить азбука Морзе, в которой для каждой буквы и цифры определена двоичная последовательность коротких и длинных сигналов. В азбуке Морзе буква Е, например, соответствует один короткий сигнал (точка), а букве Ш – четыре длинных сигнала (четыре тире). Неравномерное кодирование позволяет повысить скорость передачи сообщений за счёт того, что наиболее часто встречающимся в передаваемых текстах символам (к ним относится и буква Е) назначается для её представления более короткая комбинация.

В технике наибольшее применение нашли равномерные коды, как более удобные для реализации. Например, во втором

международном телеграфном коде символы передаваемого алфавита кодируются последовательностями из пяти «токовых или безтоковых посылок».

В математике широко используется две формы записи чисел: *естественная* и *нормальная*.

При *естественной* форме число записывается в естественном натуральном виде, например: 12 560 – целое число, 0.003572 – правильная дробь, 4.89760 – неправильная дробь.

При *нормальной* форме запись одного числа может быть различной в зависимости от ограничений, накладываемых на её форму. Например, число 12560 может быть записано так:

$12\ 560 = 1.256*10^4 = 0.1256*10^5 = 125\ 600*10^1$ и т.д.

Так как числа бывают положительные и отрицательные, то в разрядной сетке при их машинном представлении один разряд отводится под знак числа, а остальные образуют поле числа. В знаковый разряд, который может располагаться как в начале, так и в конце числа, записывается информация о знаке числа. Примем, что знак положительного числа «+» изображается символом 0, а знак отрицательного числа «–» – символом 1. Если поле числа включает n разрядов, то диапазон представления целых чисел в этом случае ограничивается значениями $–(2n–1)$ и $(2n–1)$.

Представление чисел в ЭВМ в нормальной форме называют представлением числа в форме с плавающей запятой, так как положение запятой в записи числа в этом случае не является однозначным.

2.1. Числа конечной точности

Когда люди выполняют какие-либо арифметические действия, их не волнует вопрос, сколько десятичных разрядов занимает то или иное число. Физики, к примеру, могут вычислить, что во Вселенной присутствует 10^{78} электронов, и их не волнует тот факт, что полная запись этого числа потребует 79 десятичных разрядов. Никогда не возникает проблема нехватки бумаги для записи числа.

С компьютерами дело обстоит иначе. В большинстве компьютеров количество доступной памяти для хранения чисел

фиксированно и зависит от того, когда был разработан этот компьютер. Если приложить усилия, программист сможет представлять числа в два, три и более раз большие, чем позволяет размер памяти, но это не меняет природы данной проблемы. Память компьютера ограничена, поэтому мы можем иметь дело только с такими числами, которые можно представить в фиксированном количестве разрядов. Такие числа называются *числами конечной точности*.

Рассмотрим ряд положительных целых чисел, которые можно записать тремя десятичными разрядами без десятичной запятой и без знака. В этот ряд входит ровно 1000 чисел: 000, 001, 002, 003,..., 999. При таком ограничении невозможно выразить определенные типы чисел. Сюда входят:

1) числа больше 999;
2) отрицательные числа;
3) дроби;
4) иррациональные числа;
5) комплексные числа.

Одно из свойств набора всех целых чисел – замкнутость по отношению к операциям сложения, вычитания и умножения. Другими словами, для каждой пары целых чисел i и j числа $i+j$, $i-j$ и $i \times j$ – то же целые числа. Ряд целых чисел не замкнут относительно деления, поскольку существуют такие значения i и j, для которых i/j не выражается в виде целого числа (например, 7/2 или 1/3).

Числа конечной точности не замкнуты относительно всех четырех операций. Ниже приведены примеры операций над трехразрядными десятичными числами:

600+600=1200 (слишком большое число);
003-005=-2 (отрицательное число);
050x050=2500 (слишком большое число);
007/002=3,5 (не целое число).

Отклонения можно разделить на два класса: 1) операции, результат которых превышает самое большое число ряда (ошибка переполнения) или меньше, чем самое маленькое число ряда (ошибка из-за потери значимости), и 2) операции, результат которых не является слишком маленьким или слишком

большим, а просто не является членом ряда. Из четырех примеров, приведенных выше, первые три относятся к первому классу, а четвертый – ко второму классу.

Поскольку размер памяти компьютера ограничен и он должен выполнять арифметические действия над числами конечной точности, результаты определенных вычислений будут неправильными с точки зрения классической математики. Вычислительное устройство, которое выдает неправильный ответ, может показаться странным на первый взгляд, но ошибка в данном случае – это только следствие его конечной природы. Некоторые компьютеры содержат специальное аппаратное обеспечение, которое обнаруживает ошибки переполнения.

Алгебра чисел конечной точности отличается от обычной алгебры. В качестве примера рассмотрим ассоциативный закон

$a+(b-c)=(a+b)-c$.

Вычислим обе части выражения для $a=700$, $b=400$ и $c=300$. В левой части сначала вычислим значение $(b-c)$. Оно равно 100. Затем прибавим это число к a и получим 800. Чтобы вычислить правую часть, сначала вычислим $(a+b)$. Для трехразрядных целых чисел получится переполнение. Результат будет зависеть от компьютера, но он не будет равен 1100. Вычитание 300 из какого-то числа, отличного от 1100, не даст результата 800. Ассоциативный закон не имеет силы. Порядок операций важен.

Другой пример – дистрибутивный закон:

$a\times(b-c)=a\times b-a\times c$.

Сосчитаем обе части выражения для $a=5$, $b=210$ и $c=195$. В левой части $5\times15=75$. В правой части 75 не получается, поскольку $a\times b$ выходит за пределы ряда.

Система счисления, в которой значение цифры не зависит от ее положения в числе, называется *непозиционной*. Здесь веса всех одноименных цифр, где бы они не располагались в числе, будут одинаковы.

Наиболее известными представителями непозиционных систем счисления являются иероглифические и алфавитные. Иероглифические – это такие системы счисления, у которых каждая цифра представлена своим символом, значком или

иероглифом. Наиболее известной из них является римская система счисления.

Значение числа, записанного в римской системе счисления, определяется как сумма записанных подряд цифр, причем, если слева от цифры стоит меньшая цифра, то значение последней принимается со знаком минус, например: $IX=9_{(10)}$; $XI=11_{(10)}$, то есть здесь существует отклонение от правила независимости значения цифры от положения в числе. В настоящее время римская система используется в основном для целей нумерации. Запись числа в алфавитных системах строится по такому же принципу.

К основным недостаткам непозиционных систем счисления можно отнести:

1) отсутствие нуля;
2) необходимость содержания бесконечного количества символов;
3) сложность арифметических действий над числами.

Систему счисления называют *позиционной*, если значение цифры определяется ее положением в числе. Веса цифр в позиционной системе счисления различны и значение веса цифры зависит от номера ее позиции в числе. В общем случае вес $G_i=f(p,i)$, где i – номер позиции в числе, p – целое число, отличное от нуля и называемое *основанием системы счисления*.

Десятичная система счисления наиболее распространена в вычислительной практике. Своему распространению она обязана наличию у человека на руках десяти пальцев.

2.2. Диапазоны представления чисел

Арифметические действия над числами приводят к взаимодействию между разрядами, поэтому используются позиционные системы счисления с простейшим постоянным соотношением между весами соседних разрядов. Веса разрядов подчиняются закону геометрической прогрессии. Знаменатель прогрессии $p=const$ определяется основанием позиционной системы счисления (ССЧ).

Основание p определяет возможное количество цифр в любом из разрядов, то есть одно и то же число может быть

представлено различным количеством разрядов. Чем меньше основание P, тем больше разрядов требуется для изображения одного и того же числа. Например, $31_{(10)}= 11111_{(2)}=37_{(8)}$.

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

Пусть длина разрядной сетки равна любому положительному числу, например N. Тогда максимальное число, которое можно представить в разрядной сетке заданной длины:

$$A_{(p)max} = p^N - 1. \quad (2.1)$$

Однако, если задано максимальное абсолютное значение числа, то длина разрядной сетки

$$N=log_{(p)} (A_{(p)max}+1). \quad (2.2)$$

Интервал числовой оси, заключенный между максимальным и минимальным числами, называют *диапазоном представления (ДП) чисел* в данной системе счисления для заданного ограничения на длину разрядной сетки

$$-A_{(p)max} <= ДП <= A_{(p)max}. \quad (2.3)$$

Если предположить, что затраты на каждый разряд эквивалентны затратам на изображение одной цифры, то наименьшие затраты окажутся при использовании троичной системы счисления (p=3). Десятичная система по затратам не оптимальна, и для уменьшения затрат основание P следует уменьшить. В ВМ используются в основном двоичная ССЧ с основанием p=2. Она позволяет наиболее простым способом изображать и хранить в разрядах любую из двух цифр на основе двухпозиционных схем и передавать информацию по линиям связи в форме наличия (одна цифра) или отсутствия (другая цифра) сигнала. Кроме того, двоичная ССЧ самым оптимальным способом изображает логику высказывания: истина – 1, ложь – 0. Логика высказывания является основой построения логических устройств и широко используется в программах.

В двоичных ССЧ могут быть использованы как положительные, так и отрицательные цифры, например комбинации цифр: 00, 1$\bar{1}$, 0$\bar{1}$, где $\bar{1}$ – отрицательная цифра. Но в подавляющем большинстве случаев используются цифры 01.

В зависимости от принятых весов разрядов, изображаемые числа могут быть целыми, дробными или смешанными, содержащими целую и дробную части. Обычно ограничиваются или только целыми, или только дробными. Веса разрядов нигде не записываются, а только подразумеваются. У дробных чисел вес самого младшего разряда равен 2^{-m} (m – число разрядов). Вес самого старшего разряда $2^{-m}*2^{m-1}=2^{-1}$. У целых чисел соответственно равен 2^0 – вес младшего разряда; 2^{m-1} – вес старшего разряда.

Максимальное значение изображаемого дробного числа

$$\sum_{i=-m}^{-1} 2^i = 2^0 - 2^{-m}, \qquad (2.4)$$

целого числа

$$\sum_{i=0}^{m-1} 2^i = 2^m - 2^0. \qquad (2.5)$$

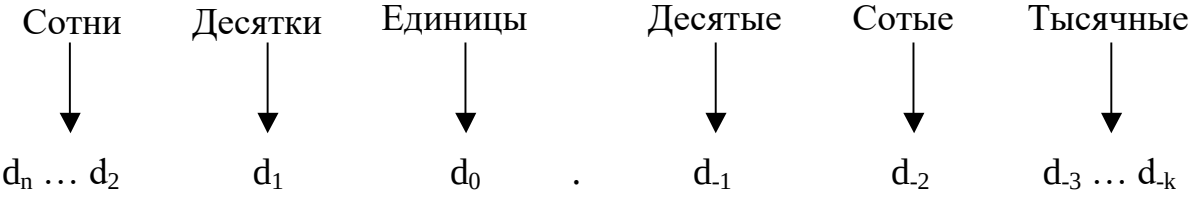
Если к максимальному значению изображаемого числа добавить одно минимальное дискретное значение – единицу младшего разряда, то изображение станет нулевым, то есть арифметические действия в вычислительных устройствах выполняются из условия сравнимости по модулю 2^m или 2^0 (это неизбежно, потому что существуют ограничения разрядности чисел).

2.3. Позиционные системы счисления

Любая позиционная система счисления характеризуется основанием. *Основание (базис)* естественной позиционной системы счисления – число знаков или символов, используемых для изображения цифр в данной системе. Таким образом, основание позиционной системы счисления – это количество различных знаков или символов, используемых для изображения цифр в данной системе. За основание системы можно принять любое натуральное число – два, три, четыре и т.д. Возможно бесчисленное множество позиционных систем, так как за основание можно принять любое число, образовав новую систему. Например, возможна шестнадцатеричная система счисления, запись чисел в которой может производиться с

помощью следующих знаков: 0, 1, ..., 9, A, B, C, Д, E, F.

Обычное десятичное число состоит из цепочки десятичных разрядов и иногда десятичной запятой. Общая форма записи показана на рис. 2.1. Десятка выбрана в качестве основы возведения в степень (основание системы счисления), поскольку мы используем 10 цифр.



$$\text{Число} = \sum_{i=-k}^{n} d_i \times 10^i$$

Рис. 2.1. Общая форма записи чисел

В компьютерах удобнее иметь дело с другими основаниями системы счисления. Самые важные из них – 2, 8 и 16. Соответствующие системы счисления называются двоичной, восьмеричной и шестнадцатеричной соответственно.

k-ричная система требует *k* различных символов для записи разрядов с 0 по *k*-*i*. Десятичные числа строятся из 10 десятичных цифр:

0123456789

Двоичные числа, напротив, строятся только из двух двоичных цифр:

01

Восьмеричные числа состоят из восьми цифр:

01234567

Шестнадцатеричные числа строятся из следующих цифр и символов:

0123456789ABCDEF

Двоичный разряд (то есть 1 или 0) обычно называют *битом*. На рисунке 2.2 десятичное число 2001 представлено в двоичной, восьмеричной и шестнадцатеричной системе. Число 7B9 очевидно шестнадцатеричное, поскольку символ B встречается только в шестнадцатеричных числах. А число 111 может быть в любой из четырех систем счисления. Чтобы

избежать двусмысленности, нужно использовать индекс для указания основания системы счисления.

Двоичное число	1 1 1 1 1 0 1 0 0 0 1 $1\times2^{10}+1+1\times2^{9}+1\times2^{8}+1\times2^{7}+1\times2^{6}+0\times2^{5}+1\times2^{4}+0\times2^{3}+0\times2^{2}+0\times2^{1}+1\times2^{0}$ 1024 + 512 + 256 + 128+ 64 + 0 + 16 + 0 + 0 + 0 + 1
Восьмеричное число	3 7 2 1 $3\times8^{3}+7\times8^{2}+2\times8^{1}+1\times8^{0}$ 1536+448 + 16 + 1
Десятичное число	2 0 0 1 $2\times10^{3}+0\times10^{2}+0\times10^{1}+1\times10^{0}$ 2000 + 0 + 0 + 1
Шестнадцатеричное число	7 D 1 $7\times16^{2}+13\times16^{1}+1\times16^{0}$ 1792 + 208 + 1

Рис. 2.2. Примеры представления чисел в двоичной, восьмеричной и шестнадцатеричной системах счисления

В таблице 2.1 в качестве примера ряд неотрицательных целых чисел представлен в каждой из четырех систем счисления.

Таблица. 2.1

Десятичные числа и их двоичные, восьмеричные и шестнадцатеричные эквиваленты

Десятичная$_{10}$	Двоичная$_{2}$	Восьмеричная$_{8}$	Шестнадцатеричная$_{16}$
0	00000	000	00

1	00001	001	01
2	00010	002	02
3	00011	003	03
4	00100	004	04
5	00101	005	05
6	00110	006	06
7	00111	007	07
8	01000	010	08
9	01001	011	09
10	01010	012	A
11	01011	013	B
12	01100	014	C
13	01101	015	D
14	01110	016	E
15	01111	017	F
16	10000	020	10
17	10001	021	11
18	10010	022	12
19	10011	023	13
20	10100	024	14
21	10101	025	15
22	10110	026	16
23	10111	027	17
24	11000	030	18
25	11001	031	19
26	11010	032	1A
27	11011	033	1B
28	11100	034	1C
29	11101	035	1D
30	11110	036	1E
31	11111	037	1F
40	101000	50	28
50	110010	62	32
60	111100	74	3C
70	1000110	106	46
80	1010000	120	50

90	1011010	132	5A
100	11001000	144	64
1000	1111101000	1750	3E8
2989	101110101101	5655	BAD

Запись чисел в каждой из систем счисления с основанием p означает сокращенную запись выражения

$$a_{n-1}p^{n-1}+a_{n-2}p^{n-2}+..\ +a_1p^1+a_0p^0+a_{-1}p^{-1}+\ ...+a_{-m}p^{-m}, \qquad (2.6)$$

где a_i – цифры системы счисления; n и m – число целых и дробных разрядов соответственно.

В каждой системе счисления цифры упорядочены в соответствии с их значениями: 1 больше 0, 2 больше 1 и т.д.

В любой системе счисления число изображается в виде набора знаков (цифр)

$$a_{m-1}, a_{m-2}, \ldots, a_1, a_0. \qquad (2.7)$$

В этих наборах цифра a_i определяет значение разряда i. При ручном счете количество разрядов не регламентируется; в вычислительных устройствах оно заранее устанавливается исходя из требований точности вычислений и допустимому объему аппаратуры. Память компьютера ограничена, поэтому мы можем иметь дело только с такими числами, которые можно представить в фиксированном количестве разрядов. Напомним, что такие числа называют *числами конечной точности*.

Каждый разряд i характеризуется своим весом p_i

$$A = a_{m-1}p_{m-1} + a_{m-2}p_{m-2} + ... + a_1p_1 + a_0p_0 = \sum_{i=0}^{m-1} a_i p_i\,, \qquad (2.8)$$

где p_i – целочисленное значение весов разрядов; a_i – значение цифр разрядов, определяющих сколько раз надо повторить веса, чтобы определить долю данного разряда в значении числа.

В вычислительных устройствах, как и при ручном счете, используются только позиционные системы счисления, позволяющие строго разграничивать аппаратуру по разрядам, что позволяет ее унифицировать.

2.4. Однородные и неоднородные системы счисления

Различают однородные и неоднородные системы счисления. *В неоднородных* системах счисления G_i не зависят

друг от друга и могут принимать любые значения. Эти системы еще называют системами со смешанным основанием. В неоднородных системах счисления в каждом i-м разряде количество допустимых символов может быть различно, при этом $0 \le a_i \le p_{i-1}$, где p_i – основание системы в i-м разряде.

В общем виде число A может быть представлено следующим образом:

$$A = a_n p_{n-1} \dots p_1 + a_{n-1} p_{n-2} \dots p_1 + \dots + a_2 p_1 + a_1, \qquad (2.9)$$

где a_i – цифра i-го разряда числа, причем $a_i = \overline{0, p_{j-1}}$ есть база системы счисления;

$p_i = \prod_0^i p_j$ – вес i-го разряда числа.

Примером неоднородной системы счисления может служить система счисления времени, для которой p_0=1 с; p_1=60 с; p_2=60 мин; p_3 =24 ч; p_4 =365 сут. Например, время в 2 года 25 суток 14 часов 35 мин 48 секунд, выраженное в единицах младшего разряда – секундах, определится по формуле (2.9):

A = 2·365·24·60·60·1+25·24·60·60·1+14·60·60·1+35·60·1+48·1.

Специально для применения в ЭВМ была создана неоднородная двоично-пятеричная система, в которой в нечетных разрядах основание p_1=5 (a_i=0–4), а в четных разрядах основание p_2=2 (a_i=0,1). Так как произведение весов двух соседних (четного и нечетного) разрядов равно десяти, то двумя двоично-десятичными разрядами можно кодировать одну десятичную цифру (табл. 2.2).

Таблица. 2.2

Кодирование десятичных цифр в двоично-пятеричной системе

$a_{(10)}$	$a_{(2\text{-}5)}$	$a_{(10)}$	$a_{(2\text{-}5)}$
0	00	5	10
1	01	6	11
2	02	7	12
3	03	8	13
4	04	9	14

Пример. Записать число $748_{(10)}$ в двоично-пятеричной системе счисления.

Решение. Исходя из значений $7_{10}=12_{(2–5)}$, $4_{10}=04_{(2–5)}$, $8_{10}=13_{(2–5)}$, представленных в таблице 2.2, получим $A_{(10)} = 12\ 04\ 13_{(2–5)}$.

Произведем обратное преобразование. Здесь $n = 6$, основания $p_1 = 5$; $p_2 = 2$; $p_3 = 5$; $p_4 = 2$; $p_5 = 5$; $p_6 = 2$, а цифры $a_1 = 3$; $a_2 = 1$; $a_3 = 4$; $a_4 = 0$; $a_5 = 2$; $a_6 = 1$. Для вычисления количественного эквивалента числа A подставим эти значения в формулу (2.9).

$A = 1\cdot 5\cdot 2\cdot 5\cdot 2\cdot 5 + 2\cdot 2\cdot 5\cdot 2\cdot 5 + 0\cdot 5\cdot 2\cdot 5 + 4\cdot 2\cdot 5 + 1\cdot 5 + 3 = 500 + 200 + 0 + 40 + 5 + 3 = 748_{(10)}$.

На практике используют сокращенную запись чисел

$$A_{(a)} = a_{n-1} \ldots a_1 a_0 a_{-1} \ldots a_{-m}. \qquad (2.10)$$

В настоящее время, в основном используются позиционные системы счисления, в которых $p_i = p_j$ при всех i и j. В них веса отдельных разрядов представляют собой степень основания, равную номеру позиции, то есть однородные системы счисления.

2.5. Свойства систем счисления

Системы счисления имеют следующие свойства:

1.Отношение весов соседних разрядов равно основанию системы счисления, то есть основание системы счисления p показывает, во сколько раз значение одной цифры $(i+1)$-го разряда превышает значение одной цифры i-го разряда.

2.Для любой системы счисления справедливо, что ее основание изображается символами 10 в своей системе, то есть любое число в своей системе p можно записать символами этой системы в виде

$$A_{(p)} = a_{n-1}\cdot 10^{n-1} + a_{n-2}\cdot 10^{n-2} + \ldots + a_0\cdot 10^0 + a_{-1}\cdot 10^{-1} + \ldots + a_{-m}\cdot 10^{-m}.$$

3.Умножение (деление) числа на целую степень, равную k, основания системы счисления p приводит к сдвигу запятой вправо (влево) на k разрядов, например:

$$A\cdot p = a_{n-1}\, a_{n-2}\ldots a_1 a_0, a_{-1}\, a_{-2}\ldots a_{-m\,(p)} \cdot p^1 = a_{n-1}\, a_{n-2}\ldots a_1 a_0\, a_{-1}, a_{-2}\ldots a =$$

$$= a_{n-1}\, a_{n-2}\ldots a_1, a_0\, a_{-1}\, a_{-2}\ldots a_{-m\,(p)}.$$

4.Общее количество различных n-разрядных p-ричных чисел равно $N = p^n$.

5.Определение наибольшего значения числа *A*, осуществляется следующим путем: в формулу (2.9) подставляют максимальное значение цифры:

$$A_{max} = (p\text{-}1)\cdot p^{n\text{-}1}+(p\text{-}1)\cdot p^{n\text{-}2}+\ldots+(p\text{-}1)\cdot p+(p\text{-}1)+(p\text{-}1)\cdot p^{-1}+\ldots$$
$$+(p\text{-}1)\cdot p^{-m} = p^{n}\text{-}1+1\text{-}p^{-m} = p^{n}\text{-}p^{-m}.$$

2.6. Выбор системы счисления

Выбор числа десять в качестве основания системы счисления исторически связан с числом пальцев на руках человека. Однако десятичная система счисления не является наиболее удобной с точки зрения ее реализации в ЭВМ. При анализе систем счисления на предмет их применения в ЭВМ учитываются следующие факторы:

1.Наличие физических элементов, способных изобразить символы системы.

2.Экономичность системы, то есть количество элементов, необходимое для представления многоразрядных чисел.

3.Трудоемкость выполнения арифметических операций в ЭВМ.

4.Быстродействие вычислительных устройств.

5.Наличие формального математического аппарата для анализа и синтеза вычислительных устройств.

6.Удобство работы человека с машиной.

7.Наибольшую помехоустойчивость кодирования цифр на носителях информации.

Наличие физических элементов, способных изобразить символы системы. Любой из символов, применяемых для записи чисел, должен в ЭВМ изображаться в виде одного или нескольких состояний какого-то физического элемента. Очевидно, что элемент будет тем проще, чем меньше состояний ему требуется иметь, то есть чем меньше основание системы счисления. Например, для реализации двоичной системы можно использовать любой простой элемент с двумя устойчивыми состояниями. Такими элементами являются: реле, конденсаторы, магнитные, полупроводниковые элементы, триггерные схемы и т.п. В настоящее время имеющиеся элементы с более чем двумя устойчивыми состояниями имеют существенные недостатки по

основным параметрам (надежность, быстродействие, габариты, стоимость). Таким образом, по этому критерию наиболее пригодной для ЭВМ является двоичная система счисления.

Экономичность системы счисления оценивается числом цифроразрядов, необходимых для изображения чисел в машине. Для представления в ЭВМ любого n-разрядного числа в системе счисления с основанием p можно использовать:

- n физических элементов с p устойчивыми состояниями;
- nk физических элементов с двумя устойчивыми состояниями, где k – минимально необходимое число двоичных разрядов, необходимых для кодирования любой p-ичной цифры.

Оценка экономичности той или иной системы счисления показала, что по критерию экономичности наиболее приемлемой является система счисления с основанием $p = 3$. Затем следуют системы с $p = 2$ и $p = 4$, которые уступают ей на 5,8%. Однако ввиду того, что троичный элемент менее надежен, чем двоичный, приходится оборудование для одного троичного разряда, как правило, увеличивать в два раза, то есть хранить троичный разряд в двух двоичных. С учетом этого наиболее экономичной оказывается двоичная система.

Трудоемкость выполнения арифметических операций. По этому критерию наиболее эффективной является двоичная система, так как чем меньше цифр участвуют в арифметических операций, тем проще их выполнять.

Быстродействие вычислительных устройств. Этот критерий находится в прямой зависимости от простоты арифметических операций. Очевидно также, что с увеличением числа цифр в системе счисления быстродействие ЭВМ при прочих равных условиях будет падать. Исследование показали, что ЭВМ, работающая в двоичной системе счисления, характеризуется более высоким быстродействием относительно троичной на 26,2% и относительно десятичной – в 2,7 раза.

Наличие формального математического аппарата для анализа и синтеза вычислительных устройств. Таким аппаратом является алгебра логики. Наибольшее развитие и законченность изучения, вследствие своей простоты и широкого практического применения получила двоичная логика.

Удобство работы человека с машиной. Безусловно, самой удобной по этому критерию является десятичная система счисления. Но решить, какая система находится на втором месте, сложнее, так как все они требуют перевода чисел. Очевидно, наиболее удобной для человека будет система, в которой проще всего выполняются арифметические действия, то есть двоичная.

Наибольшая помехоустойчивость кодирования цифр. Исходя из условия равных технических возможностей при реализации любой системы счисления, будем считать, что диапазон изменения носителя информации для всех систем остается одинаковым. Это значит, что при наложении некоторой помехи на основной сигнал, изображающий цифру, наибольшая ошибка возможна в устройстве, использующим систему счисления с самым большим основанием. Следовательно, с позиций наибольшей помехоустойчивости предпочтение следует отдать двоичной системе счисления.

Таким образом, исходя из перечисленных критериев, наиболее приемлемой для ЭВМ является *двоичная система счисления*. Однако в некоторых случаях при синтезе вычислительного устройства какому-либо критерию придается большее значение, чем остальным. Тогда для применения выбирается система счисления, оптимальная по выбранному критерию.

При использовании двоичной системы счисления необходимо выполнить преобразование десятичных чисел в двоичные. Однако, учитывая то обстоятельство, что многие математические задачи требуют сравнительно малый объем исходных данных по сравнению с объемом вычислений, этот недостаток становится несущественным.

Существенным недостатком двоичной системы счисления является то, что для записи двоичного числа требуется примерно в 3,3 раза больше разрядов, чем для записи того же числа в десятичной системе. Поэтому двоичную систему применяют, как правило, «для внутренних» нужд машины, а для целей коммуникации человека с машиной выбирают двоично-кодированные системы счисления: восьмеричную, шестнадцатеричную и двоично-кодированную десятичную.

2.7. Преобразование чисел в системах счисления

При подготовке вычислений на ЭВМ необходимы прямые и обратные преобразования между двоичными и десятичными числами. Эти преобразования различны для целых и дробных чисел.

Пусть целое десятичное число А разложено по другому основанию ССЧ *p*:

$$A = a_{m-1}p^{m-1} + a_{m-2}p^{m-2} + \ldots + a_1 p^1 + a_0 p^0 . \qquad (2.11)$$

Если A разделить на p, то получится число $A_1 = Ap^{-1}$, и в остатке a_0, равное значению младшего разряда числа A и записанное в новой *p*–ричной ССЧ.

При этом

$$A_1 = a_{m-1}p^{m-2} + a_{m-2}p^{m-3} + \ldots + a_1 p^0 . \qquad (2.12)$$

Структура A_1 такая же, как A в формуле (2.11), поэтому путем повторных преобразований можно последовательно определить значения разрядов $a_1, a_2, a_3, \ldots , a_{m-1}$.

Таким образом, первый способ перевода основан на делении числа на 2. Частное записывается непосредственно под исходным числом, а остаток (0 или 1) записывается рядом с частным. То же проделывается с полученным частным. Процесс повторяется до тех пор, пока не останется 0. В результате должны получиться две колонки чисел – частных и остатков. Двоичное число можно считать из колонки остатков снизу вверх. На рисунке 2.3 показано, как происходит преобразование из десятичной в двоичную систему с помощью операции деления.

Преобразование десятичного числа 1492 в двоичное производится путем последовательного деления (сверху вниз). Например, 93 делится на 2, получается 46 и остаток 1. Остаток записывается в строку внизу.

Второй способ вытекает из определения двоичных чисел. Самая большая степень двойки, меньшая, чем число, вычитается из этого числа. Та же операция проделывается с полученной разностью. Когда число разложено по степеням двойки, двоичное число может быть получено следующим образом. Единички ставятся в тех позициях, которые соответствуют

полученным степеням двойки, а нули – во всех остальных позициях.

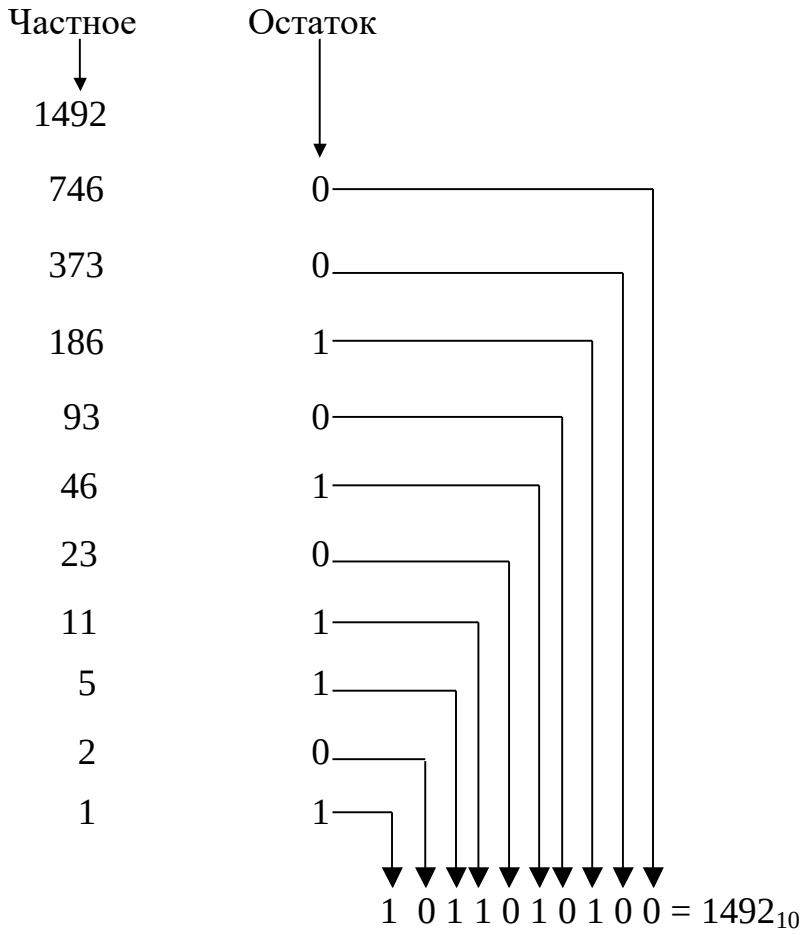


Рис. 2.3. Преобразование десятичного числа в двоичное

Соответствующие степени числа два представлены следующим рядом:

$2^0 = 1 \quad 2^2 = 4 \quad 2^4 = 16 \quad 2^6 = 64 \quad 2^8 = 256 \quad 2^{10} = 1024$

$2^1 = 2 \quad 2^3 = 8 \quad 2^5 = 32 \quad 2^7 = 128 \quad 2^9 = 512$ и т.д.

Например, с помощью подбора целых степени двойки перевести десятичное число 293 в соответствующее двоичное можно следующим образом:

$A^1 = 293 - 2^8 = 37$

$A^2 = 37 - 2^5 = 5$

$A^3 = 5 - 2^2 = 1$

$A^4 = 1 - 2^0 = 0$

$293_{10} = [2^8 + 2^5 + 2^2 + 2^0 + 2^8] = 100100101_2$

Чтобы преобразовать двоичное число в восьмеричное, нужно разделить его на группы по три бита, причем три бита непосредственно слева от двоичной запятой формируют одну группу, следующие три бита слева от этой группы формируют вторую группу и т. д. Каждую группу по три бита можно преобразовать в один восьмеричный разряд со значением от 0 до 7 (см. табл. 2.1). Чтобы дополнить группу до трех битов, нужно спереди приписать один или два нуля. Преобразование из восьмеричной системы в двоичную тоже тривиально. Каждый восьмеричный разряд просто заменяется эквивалентным 3-битным числом. Преобразование из 16-ричной в двоичную систему, по сути, сходно с преобразованием из 8-ричной в двоичную систему, только каждый 16-ричный разряд соответствует группе из четырех битов, а не из трех. На рисунке 2.4 приведены примеры преобразований из одной системы в другую.

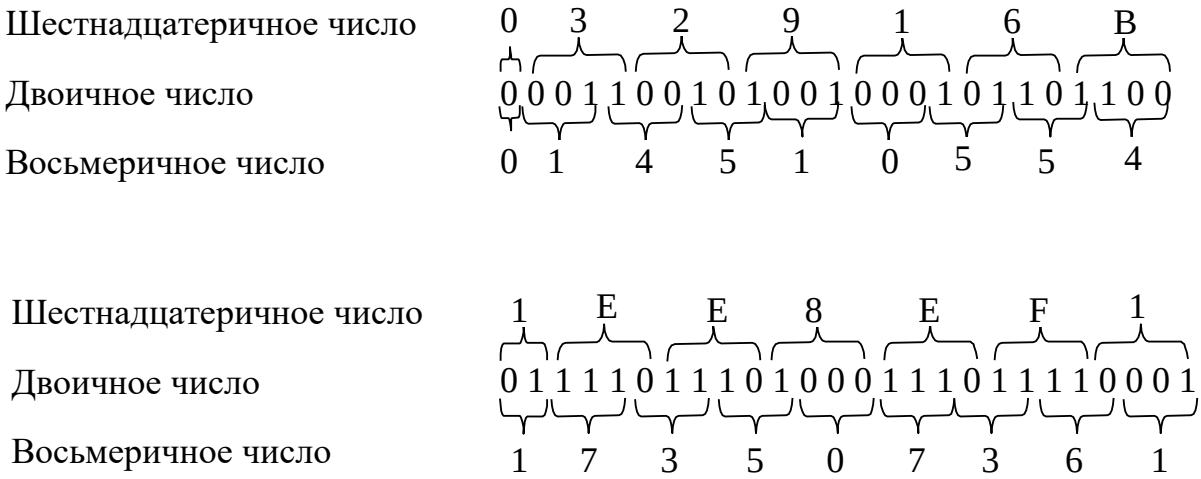


Рис. 2.4. Примеры преобразования двоичных чисел

Двоичные числа можно преобразовывать в десятичные двумя способами. Первый способ – суммирование степеней двойки, которые соответствуют биту 1 в двоичном числе. Например:

10110=24+22+21=16+4+2=22

Второй способ. Двоичное число записывается вертикально по одному биту в строке, крайний левый бит находится внизу. Самая нижняя строка — это строка 1, затем идет строка 2 и т. д. Десятичное число строится напротив этой колонки. Сначала

обозначим строку 1. Элемент строки *n* состоит из удвоенного элемента строки n-1 плюс бит строки n (0 или 1). Элемент, полученный в самой верхней строке, и будет ответом. Метод проиллюстрирован на рис. 2.5.

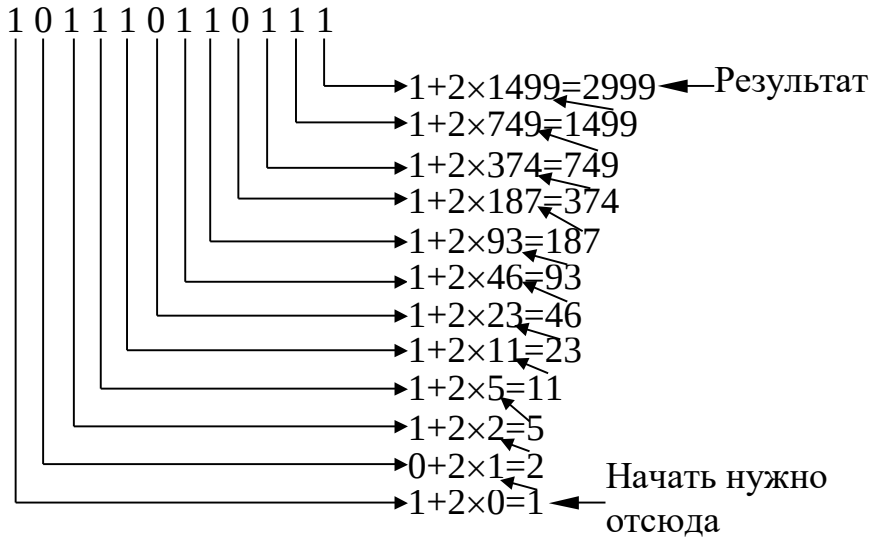


Рис. 2.5. Преобразование двоичного числа в десятичное с помощью последовательного умножения

На рис. 2.5 двоичное число 101110110111 преобразуется в десятичное путем последовательного удваивания снизу вверх. В каждой следующей строке удваивается значение предыдущей строки и прибавляется соответствующий бит. Число 374 умножается на 2 и прибавляется бит соответствующей строки (в данном случае 1). В результате получается 749 и т.д.

Преобразование из десятичной в восьмеричную или 16-ричную систему можно выполнить либо путем преобразования сначала в двоичную, а затем в нужную нам систему, либо путем вычитания степеней 8 или 16.

Для перевода правильных дробей из системы счисления с основанием p_1 в систему с основанием p_2 используется метод, базирующийся на умножении переводимой правильной дроби на основание p_2 новой системы счисления. Правильная дробь $A_{(p1)}$ в системе с основанием p_2 может быть записана в виде

$$A_{(p2)} = a_{-1} \cdot p_2^{-1} + a_{-2} \cdot p_2^{-2} + \ldots + a_{-m} \cdot p_2^{-m}. \qquad (2.13)$$

Если правую часть выражения умножить на p_2, то найдем неправильную дробь, в целой части которой будет число a_{-1}.

Умножив затем оставшуюся дробную часть на величину основания p_2, получим дробь, в целой части которой будет a_{-2} и т.д. Повторяя процесс умножения m раз, найдем все m цифр дробной части числа в новой системе счисления. При этом все действия должны выполняться по правилам арифметики и, следовательно, в целой части получающихся дробей будут появляться эквиваленты цифр новой системы счисления, записанные в исходной системе счисления.

При p=2 алгоритм преобразования десятичного дробного в двоичное число сводится к последовательному умножению дробных частей произведений на два и к последовательному формированию двоичных разрядов начиная со старших по значению частей произведений.

Пример. A_{10}=0.752 перевести с точностью до 6 двоичных разрядов.

$$0.752 \cdot 2 = 1.504 \; A_{-1} = 1$$

$$0.504 \cdot 2 = 1.008 \; A_{-2} = 1$$

$$0.008 \cdot 2 = 0.016 \; A_{-3} = 0$$

$$0.016 \cdot 2 = 0.032 \; A_{-4} = 0$$

$$0.032 \cdot 2 = 0.064 \; A_{-5} = 0$$

$$0.064 \cdot 2 = 0.0128 \; A_{-6} = 0$$

$0.752_{10} \approx 0.110000_2$

При переводе правильных дробей из одной системы счисления в другую можно получить дробь в виде бесконечного числа или расходящегося ряда. Процесс перевода можно закончить, если появится дробная часть, имеющая во всех разрядах нули, или будет достигнута заданная точность перевода (получено требуемое число разрядов результата). Последнее означает, что при переводе дроби необходимо указать число разрядов в случае ее представления в новой системе счисления. Естественно, что при этом возникает погрешность перевода чисел. В ЭВМ точность перевода обычно ограничивается длиной разрядной сетки, отведенной для представления чисел.

Двоичные числа неудобны в обращении из-за большой разрядности, особенно при вводе в ЭВМ. Поэтому в ВТ находят

широкое применение в качестве промежуточных 2^p-ричные ССЧ (p=2, p=3, p=4), в каждой из которых могут быть записаны любые из 2^p цифр. В свою очередь, любая 2^p–ричная цифра может быть изображена комбинацией из p двоичных цифр. Таким образом, группируя двоичное число по p разрядам, можно элементарным способом переходить из двоичной системы в 2^p-ричную и наоборот. Наиболее широко используются восьмеричная (p=3)

p=3 0,1,2,3,4,5,6,7

и шестнадцатеричная (p=4)

p=4 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

системы счисления, которые очень удобны в качестве промежуточных при переводе десятичных чисел в двоичные.

Представление чисел в восьмеричной и шестнадцатеричной системе счисления мы рассмотрим в следующем пункте.

2.8. Восьмеричная и шестнадцатеричная системы

Напомним, что восьмеричная и шестнадцатеричная системы счисления относятся к двоично-кодированным системам, когда основание системы счисления p представляет целую степень двойки:

$p = 8 = 2^3$ – для восьмеричной,

$p = 16 = 2^4$ – для шестнадцатеричной.

База *восьмеричной системы счисления* использует для изображения чисел восемь цифр: 0, 1, 2, 3, 4, 5, 6, 7, то есть $a_i = \overline{0,7}$. Основание $p = 8_{(10)} = 10_{(8)}$. Если восьмеричное число записать в развернутом виде в виде суммы значений цифр и выполнить арифметические действия по правилам десятичной системы, то получим десятичный эквивалент восьмеричного числа. Например, $125,4_{(8)} = 1\cdot 8^2 + 2\cdot 8^1 + 5\cdot 8^0 + 4\cdot 8^{-1} = 85,5_{(10)}$.

В *шестнадцатеричной системе* для записи чисел используются цифры 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 и прописные латинские буквы *A, B, C, D, E, F*, имеющие значение десятичных чисел 10, 11, 12, 13, 14, 15 соответственно. Поэтому шестнадцатеричное число может иметь, например, вид $3E5,C_{(16)}$. Представляя это число в развернутом виде, получим

$3E5,C_{(16)} = 3\cdot 16^2 + E\cdot 16^1 + 5\cdot 16^0 + C\cdot 16^{-1}$.

Выполняя арифметические операции по правилам десятичной системы и принимая во внимание, что E=14, C=12, получим $3E5,C_{(16)} = 560,75_{(10)}$.

Большим достоинством восьмеричной и шестнадцатеричной систем счисления является, во-первых, возможность более компактно представить запись двоичного числа, во-вторых, сравнительно просто осуществлять преобразование чисел из двоичной в восьмеричную и шестнадцатеричную системы, и наоборот. Действительно, так как для восьмеричного числа каждый разряд представляется группой из трех двоичных разрядов, а для шестнадцатеричного – группой из четырех двоичных разрядов, то для такого преобразования достаточно объединить двоичные цифры в группы по 3 и 4 бита соответственно, продвигаясь от разделяющей запятой вправо и влево. При этом в случае необходимости добавляют нули в начале и в конце числа и каждую такую группу – триаду или тетраду – заменяют эквивалентной восьмеричной или шестнадцатеричной цифрой.

Приведем примеры:

а) перевод двоичного числа 1101111001,1101 в восьмеричное:

$$\begin{array}{cccccc} 001 & 101 & 111 & 001, & 110 & 100_{(2)} \\ 1 & 5 & 7 & 1 & 6 & 4 \end{array} = 1571,64_{(8)}$$

б) перевод двоичного числа 11111111011,100111 в шестнадцатеричное:

$$\begin{array}{cccccc} 0111 & 1111 & 1011, & 1001 & 1100_{(2)} \\ 7 & F & B & 9 & C \end{array} = 7FB,9C_{(16)}$$

Восьмеричная и шестнадцатеричная системы счисления используются в текстах программ для более короткой и удобной записи двоичных кодов команд, адресов и операндов. Особенно удобно использовать шестнадцатеричную систему, когда разрядность чисел и команд выбрана кратной байту, при этом каждый двоичный код байта записывается в виде двухразрядного шестнадцатеричного числа. Кроме того, эти системы применяются в ЭВМ при некоторых формах представления чисел.

2.9. Отрицательные двоичные числа

На протяжении всей истории цифровых компьютеров для репрезентации отрицательных чисел использовались четыре различные системы. Первая из них называется системой со знаком. В такой системе крайний левый бит – это знаковый бит (0 – это «+», а 1 – это «-»), а оставшиеся биты показывают абсолютное значение числа.

Во второй системе, которая называется дополнением до единицы, тоже присутствует знаковый бит (0 – это плюс, а 1 – это минус). Чтобы сделать число отрицательным, нужно заменить каждую 1 на 0 и каждый 0 на 1. Это относится и к знаковому биту. Система дополнения до единицы уже устарела.

Третья система, дополнение до двух, содержит знаковый бит (0 – это «+», а 1 – это «-»). Отрицание числа происходит в два этапа. Сначала каждая единица меняется на 0, а каждый 0 – на 1 (как и в системе дополнения до единицы). Затем к полученному результату прибавляется 1. Двоичное сложение происходит точно так же, как и десятичное, только перенос совершается в том случае, если сумма больше 1, а не больше 9. Например, рассмотрим преобразование числа 6 в форму с дополнением до двух:

00000110 (+6);

11111001 (-6 в системе с дополнением до единицы);

11111010 (-6 в системе с дополнением до двух).

Если нужно совершить перенос из крайнего левого бита, он просто отбрасывается.

В четвертой системе, которая для m-битных чисел называется *excess* 2^{m-1}, число представляется как сумма этого числа и 2^{m-1}. Например, для 8-битного числа (m=8) система называется excess 128, а число сохраняется в виде суммы исходного числа и 128. Следовательно, –3 превращается в –3+128=125, и это число (–3) представляется 8-битным двоичным числом 125 (01111101). Числа от –128 до +127 выражаются числами от 0 до 255 (все их можно записать в виде 8-битного положительного числа). Отметим, что эта система соответствует системе с дополнением до двух с обращенным знаковым битом.

В таблице 2.3 представлены примеры отрицательных чисел во всех четырех системах.

Таблица 2.3
Отрицательные 8-битные числа
в четырех различных системах

N-десятичное	N двоичное	-N в системе со знаком	-N дополнение до единицы	-N дополнение до двух	-N excess 128
1	00000001	10000001	11111110	11111111	01111111
2	00000010	10000010	11111101	11111110	01111110
3	0000001 1	10000011	11111100	11111101	01111101
4	00000100	10000100	11111011	11111100	01111100
5	00000101	10000101	11111010	11111011	01111011
6	00000110	10000110	11111001	11111010	01111010
7	00000111	10000111	11111000	11111001	01111001
8	00001000	10001000	11110111	11111000	01111000
9	00001001	10001001	11110110	11110111	01110111
10	00001010	10001010	11110101	11110110	01110110
20	00010100	10010100	11101011	11101100	01101100
30	00011110	10011110	11100001	11100010	01100010
40	10101000	10101000	11010111	11011000	01011000
50	00110010	10110010	11001101	11001110	01001110
60	00111100	10111100	11000011	11000100	01000100
70	01000110	11000110	10111001	10111010	00111010
80	01010000	11010000	10101111	10110000	00110000
90	01011010	11011010	10100101	10100110	00100110
100	01100100	11100100	10011011	10011100	00011100
127	01111111	11111111	10000000	10000001	00000001
128	Не сущест.	Не сущест.	Не сущест.	10000000	00000000

В системах со знаком и с дополнением до единицы есть два представления нуля: +0 и –0. Такая ситуация нежелательна. В системе с дополнением до двух такой проблемы нет, поскольку здесь плюс нуль это всегда плюс нуль. Но зато в этой системе есть другая особенность. Набор битов, состоящий из 1, за которым следуют все нули, является дополнением самого себя. В результате ряд положительных и отрицательных чисел несимметричен – существует одно отрицательное число без соответствующего ему положительного.

2.10. Двоично-кодированная десятичная система счисления (*D*-коды)

Непосредственное изображение десятичных чисел приводит к необходимости двоичного кодирования десятичных цифр. Устройствам, выполняющим арифметические преобразования с десятичными числами, присваивается специальный термин «десятичная арифметика». Такие устройства должны иметь максимальное сходство с обычными двоичными устройствами.

Десятичная арифметика включается в состав аппаратурных средств высокопроизводительных систем с целью исключения преобразований исходных данных в двоичную форму и результатов в десятичную.

Двоично-кодированная десятичная система является комбинированной системой счисления, которая обладает достоинствами двоичной и удобством десятичной системы.

D-код – это двоично-кодированное представление десятичного числа, в котором каждая десятичная цифра представляется тетрадой из двоичных символов.

Количество различных двоичных тетрад $N = 2^4 = 16$. Для кодирования двоичных цифр из них используется только десять. Наличие избыточных комбинаций позволяет иметь различные *D*-коды. В ЭВМ наибольшее применение нашли системы кодирования 8421 – D_1, 2421 – D_2, (8421+3) – D_4. Появляющаяся избыточность приводит к множеству кодирования десятичных цифр, из которых следует выбирать оптимальную.

Код 8421 (табл. 2.4) называется *кодом с естественными весами*, где цифры 8,4,2,1 – веса двоичных разрядов тетрад. Любая десятичная цифра в этом коде изображается ее эквивалентом в двоичной системе счисления. Этот код нашел наибольшее применение при кодировании десятичных чисел в устройствах ввода-вывода и при построении операционных устройств десятичной арифметики.

Особенность кодов D_2 и D_4 (8421+3) или кода с избытком 3 в том, что кодирование любой десятичной цифры и дополнительной к ней цифры до 9 осуществляется взаимно дополняющими тетрадами. Эта особенность дает простой способ

получения дополнения до 9 путем инвертирования двоичных цифр тетрады. Такие коды удобно использовать для организации операции вычитания при построении десятичных сумматоров.

Таблица 2.4

Примеры кодирования десятичных цифр тетрадами

Десятичная цифра	Эквиваленты в *D*-кодах		
	D_1 (8421)	D_2 (2421)	D_4 (8421+3)
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0011	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100

Приведем пример кодирования десятичного числа A = 8371 в двоично-кодированной десятичной системе счисления:

D_1: A = $1000\ 0011\ 0111\ 0001_{(2/10)}$;

D_2: A = $1110\ 0011\ 1101\ 0001_{(2/10)}$;

D_4: A = $1011\ 0110\ 1010\ 0100_{(2/10).}$

Оптимальность кодирования определяется шестью требованиями, которым должен удовлетворять десятичный код.

1. *Однозначность.* Каждой десятичной цифре должен соответствовать определенный, отличающийся от других, двоичный код.

Невыполнение данного требования приводит к неоднозначности результатов.

2. *Упорядоченность.* Большим десятичным цифрам должны соответствовать большие тетрады десятичного кода и, наоборот, меньшим – меньшие тетрады.

Выполнение данного требования необходимо для организации количественного сравнения цифр в десятичных разрядах.

3. *Четность.* Четным цифрам должны соответствовать четные тетрады, нечетным цифрам – нечетные тетрады. Соответствие может быть отмечено любым способом.

Выполнение данного требования необходимо для выполнения округления результата.

4. *Дополнительность.* Если x_1 и x_2 – такие две цифры, для которых $x_1+x_2 = 9$ и цифре x_1 сопоставляется тетрада $\alpha_3\alpha_2\alpha_1\alpha_0$, то цифре x_2, если удовлетворяется требование дополнительности, должна сопоставляться тетрада $\alpha_3\alpha_2\alpha_1\alpha_0$, получаемая путем инверсии двоичных разрядов кода цифры x_1.

Требование дополнительности необходимо для упрощения реализации дополнительных и обратных кодов десятичных чисел.

5. *Весомозначность.* Должны существовать четыре целых положительных числа: p_3,p_2,p_1,p_0, называемых весами, с помощью которых можно определить десятичную цифру x по значению двоичной тетрады $\alpha_3\alpha_2\alpha_1\alpha_0$, сопоставленной x, по формуле

$$x = \alpha_3 p_3 + \alpha_2 p_2 + \alpha_1 p_1 + \alpha_0 p_0 .$$

Выполнение данного требования способствует декодированию.

6. *Непрерывность.* Непрерывной последовательности изменений значения цифр должна соответствовать непрерывная последовательность изменений значения тетрад.

Ни один из десятичных кодов не удовлетворяет одновременно всем шести перечисленным требованиям.

Наибольшее распространение в ВТ нашел код прямого замещения с весом разрядов 8421. Этот код самый наглядный и удобный, так как в соответствии с названием кода десятичная цифра в нем соответствующим значением двоичного кода. Однако код 8421 не удовлетворяет требованию дополнительности, поэтому действия в этом коде с изменением знака десятичного числа связаны с инверсией разрядов или взятия дополнения, то есть требуют дополнительных коррекций и/или временных затрат.

Достоинствами двоично-кодированной десятичной системы счисления относительно двоичной являются:

- отсутствие необходимости перевода исходных данных и результатов из одной системы счисления в другую;
- удобство контроля промежуточных результатов путем вывода их на индикацию для внутреннего наблюдения;
- более широкие возможности для автоматического контроля из-за наличия в *D*-кодах избыточных комбинаций.

D-коды применяют для решения экономических задач, которые характеризуются большим объемом исходных данных, сравнительной простотой и малым объемом выполняемых над ними преобразований и большим количеством результатов вычислений. Эта система широко используется в калькуляторах и персональных микроЭВМ.

2.11. Формы представления чисел в ЭВМ

ЭВМ оперирует с числами, содержащими конечное число разрядов. Количество разрядов ограничено длиной разрядной сетки машины.

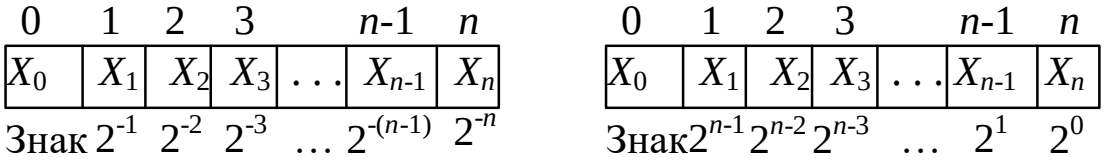
Под разрядной сеткой понимается совокупность двоичных разрядов, предназначенных для хранения и обработки машинных слов (*двоичных кодов*).

Для представления чисел используются две формы с фиксированной запятой (естественная) и с плавающей запятой (нормальная). Для любой формы представления чисел их знаки – плюс и минус – кодируются соответственно цифрами 0 и 1. При этом знак числа располагается перед старшим разрядом кода числа.

Форма представления чисел с фиксированной запятой предполагает, что положение запятой, отделяющей целую часть от дробной, фиксировано в разрядной сетке машины. (Хотя запятая и фиксируется, но никак не выделяется в коде числа, а только подразумевается.)

Количество двоичных разрядов и положение запятой определяют такие важные характеристики ЭВМ, как точность и диапазон представления чисел.

Обычно в ЭВМ используются два способа расположения запятой (рис. 2.6): перед старшим разрядом (дробные числа) или после младшего (целые числа).



а) б)

Рис. 2.6. Разрядные сетки ЭВМ для представления чисел с фиксированной запятой: а – для правильных дробей; б – для целых чисел

Диапазоны представления модулей для этих разрядных сеток имеют вид:

а) для дробных чисел:

$|X_{max}| = 0{,}111\ldots11_{(2)} = 1\text{-}2^{-n}$;

$|X_{min}| = 0{,}000\ldots01_{(2)} = 2^{-n}$;

$$1\text{-}2^{-n} \geq |X| \geq 2^{-n}\ ; \qquad (2.14)$$

б) для целых чисел:

$|X_{max}| = 111\ldots11_{(2)} = 2^n\text{-}1$;

$|X_{min}| = 000\ldots01_{(2)} = 1$;

$$2^n\text{-}1 \geq |X| \geq 1. \qquad (2.15)$$

Важной характеристикой является *динамический диапазон* модулей представляемых чисел:

$$d = \frac{|X_{\max}|}{|X_{\min}|}. \qquad (2.15)$$

Для дробных чисел

$$d = \frac{1-2^n}{2^{-n}} = 2^n - 1; \qquad (2.16)$$

для целых чисел

$$d = \frac{2^n - 1}{2^{-n}} = 2^n - 1. \qquad (2.17)$$

Величина динамического диапазона d не зависит от положения запятой, а определяется только длиной разрядной сетки. В обоих случаях диапазон представления чисел в машине с фиксированной запятой

$$d = \frac{|X_{\max}|}{|X_{\min}|} = \frac{1-2^{-n}}{2^{-n}} = \frac{2^n - 1}{1} \cong 2^n. \qquad (2.18)$$

Если при выполнении вычислений значения чисел выйдут за пределы допустимого диапазона, то возникнет ошибка. Чтобы этого избежать, исходные данные необходимо масштабировать:

$$X = [X]K_x, \qquad (2.19)$$

где K_x – масштабный коэффициент.

Длину разрядной сетки с фиксированной запятой в операционных устройствах (ОУ) современных ЭВМ принято выбирать кратной байту (8 бит). Для мини- и микроЭВМ использует 1, 2 или 4 байта, для ЭВМ общего назначения длина разрядной сетки составляет 4 байта (32 бита) или 8 байт (64 бита). Длина слова в памяти ОУ также кратна 1 байту.

По сложившимся в вычислительной технике традициям нумерация разрядов в разрядной сетке в машинах общего назначения ведётся слева направо, а в мини- и микроЭВМ, микропроцессорах – справа налево.

Форма представления чисел с плавающей запятой позволяет избежать трудоёмкого масштабирования исходных чисел и значительно увеличить диапазон и точность представляемых чисел [9,10].

Представление чисел в форме с плавающей запятой в общем виде определяются выражением

$$X = \pm M_x Q^p, \qquad (2.20)$$

где M_x – мантисса числа;

Q^p – характеристика числа;

Q – основание системы счисления ($Q = 2$ или $Q = 16$);

p – порядок числа.

Мантисса и порядок задаются в системе счисления с основанием Q. Знак числа совпадает со знаком числа мантиссы.

Мантисса в ЭВМ обычно представляется правильной дробью в нормализованном виде, то есть первая цифра справа от запятой должна быть отличной от нуля:

$$M_x = x_0, x_1, x_2, \ldots, x_m,$$

где x_0 – код знака числа.

Для $Q = 2$: $M_x = x_0, 1, x_2, \ldots, x_m$; для $Q = 16$: $M = x_0, x_1^*, x_2^* \ldots x_m^* x_{m/4}^*$, где x_i^* – шестнадцатеричная цифра, $x_1^* > 0$.

Порядок числа p – целое число со знаком – имеет смысл указателя истинного положения запятой в числе.

Форматы чисел (разрядная сетка) для плавающей запятой в двоичной системе счисления с указанием веса разрядов представлены на рис. 2.7.

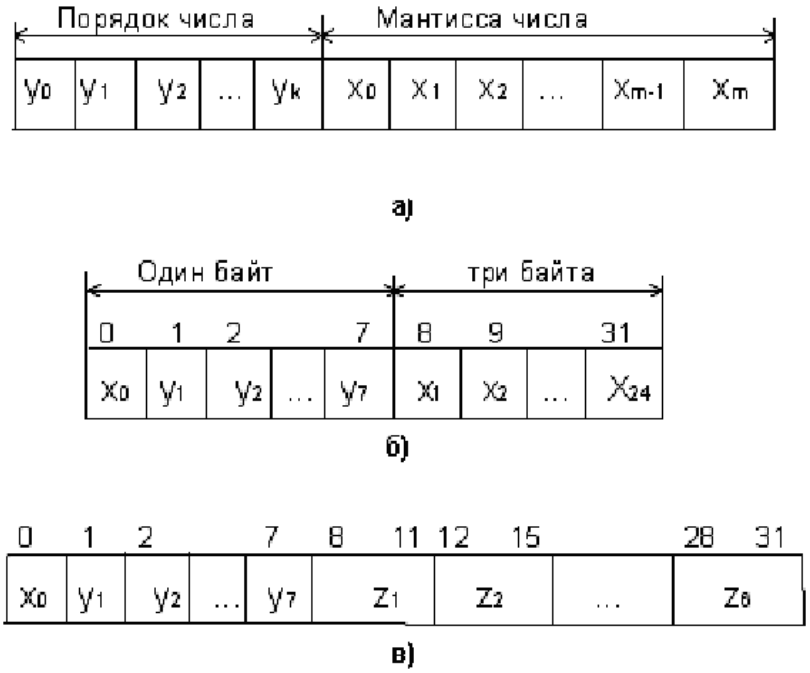


Рис. 2.7. Формат чисел с плавающей запятой:
а – изображение порядков положительными и отрицательными числами; y_0 – код знака порядка; б – изображение кода числа со смещенным порядком в формате слова; в – изображение шестнадцатеричного числа в формате слова со смещенным порядком

Диапазон представления нормализованных чисел с плавающей запятой, когда порядок может принимать как положительное, так и отрицательное значение, определяется так (для $Q = 2$):

$$P_{\max} = 2^k - 1; \quad |\mathrm{M}_{\max}| = 1 - 2^{-m};$$

$$|\mathrm{X}_{\max}| = 2^{P_{max}} |M_{\max}| = 2^{2^1-1} \left(1 - 2^{-m;}\right);$$

$$P_{\min} = -\left(2^k - 1\right); \; |\mathrm{M}_{\min}| = 2^{-1}; \; |\mathrm{X}_{\min}| = 2^{-\left(2^1-1\right)} \cdot 2^{-1} = 2^{-2^1};$$

$$2^{2^1-1}\left(1 - 2^{-m}\right) \geq |X| \geq 2^{-2^1}. \qquad (2.21)$$

Динамический диапазон

$$d = \frac{|X_{\max}|}{[X_{\min}]} = \frac{2^{2^1-1}}{2^{-2^1}} \cong 2^{2^{1+1}-1}. \qquad (2.22)$$

Сравнивая выражения (2.18) и (2.22) можно показать, что при одной и той же длине разрядной сетки диапазон представляемых чисел с плавающей запятой значительно шире, а относительная погрешность представления e=1/*d* значительно меньше, чем для чисел с фиксированной запятой. Однако устройства, реализующие операции с числами, представленными в форме с плавающей запятой, характеризует большая сложность и меньшее быстродействие.

В некоторых ЭВМ для упрощения операций над порядками используют форму с плавающей запятой со смещенным порядком. В этом случае порядок представляется целым *k*-разрядным положительным числом, а разряд знака порядка добавляется к мантиссе (см. рис. 2.7). При таком кодировании число, равное нулю, будет содержать нули в разрядной сетки как для мантиссы, так и для порядка.

Для представления чисел с плавающей запятой широко используется шестнадцатеричная система счисления (Q = 16), при этом смещенный порядок числа представляется двоичным целым числом, а мантисса – шестнадцатеричной дробью (см. рис. 2.7в). Тогда выражение (2.20) примет вид

$$X = \pm 16^p M_x. \qquad (2.23)$$

В этом случае при одинаковом числе двоичных разрядов разрядной сетки несколько уменьшается точность представления чисел, по сравнению с точностью представления чисел в двоичной системе счисления, однако увеличивается диапазон представляемых чисел в машине.

2.12. Машинный нуль и переполнение разрядной сетки

В общем случае при любом способе представления чисел при решении задачи на машине возможны два вида выхода результатов за пределы диапазона представления.

В первом случае числа по абсолютной величине становятся меньше наименьшего значащего числа, которое можно записать

в разрядную сетку: $|X| < X_{min}$, а во втором случае они могут стать больше наибольшего представимого числа $|X| > X_{max}$.

При ситуации, соответствующей первому случаю, получаем число, состоящее в пределах разрядной сетки из одних нулей (машинный нуль). При выполнении операций над числами с плавающей запятой возможно получение числа, имеющего порядок меньше допустимого и нормализованную мантиссу, или числа, имеющего нулевую мантиссу и допустимый порядок. Эти числа рассматриваются как машинные нули.

Во втором случае описывается возникновение переполнения разрядной сетки. Различают два вида переполнения: положительное, когда $X > + X_{max}$ и отрицательное, если $X < -X_{max}$.

В случае представления чисел с плавающей запятой переполнение разрядной сетки выражается в том, что результат какой-либо операции имеет порядок больше допустимого.

При любой форме представления чисел переполнение разрядной сетки приводит к аварийной ситуации, требующей останова решения задачи.

2.13. Точность представления чисел в ЭВМ

Представление чисел характеризуется абсолютной и относительной погрешностями.

Абсолютная погрешность – это разность между истинным значением входной величины X и её значением, полученным из машинного изображения $[X]$, то есть

$$\Delta[X] = X - [X]. \qquad (2.24)$$

Усреднённая абсолютная погрешность представления чисел в машине с фиксированной запятой определяется как среднее арифметическое между минимальным представимым числом и его минимальной потерей:

$$\Delta = \frac{X_{min} + 0}{2} = \frac{2^{-n} + 0}{2} = 2^{-(n+1)}, \qquad (2.25)$$

то есть в машинах с фиксированной запятой абсолютная погрешность постоянна и равна половине младшего разряда.

Относительная погрешность представления определяется как отношение усреднённой абсолютной погрешности к самому числу $\varepsilon = |\ X\ |$.

Так как само число X меняется в пределах $X_{\min} = 2^{-n} \le |X| \le 1 - 2^{-n} = X_{\max}$, то и относительная погрешность является величиной переменной, меняющейся соответственно в пределах $\varepsilon_{\min} \le \varepsilon \le \varepsilon_{\max}$.

Для машин с фиксированной запятой она определятся следующим образом:

$$\varepsilon_{\min} = \left|\frac{\Delta}{X_{\max}}\right| = \frac{2^{-(n+1)}}{1-2^{-n}} \approx 2^{-(n+1)};$$
$$\varepsilon_{\max} = \left|\frac{\Delta}{X_{\min}}\right| = \frac{2^{-(n+1)}}{2^{-n}} \approx 2^{-1}. \qquad (2.25)$$

Таким образом, относительная погрешность для машин с фиксированной запятой зависит от величины числа и колеблется в пределах от $2^{-(n+1)}$ для больших чисел до 2^{-1} – для малых; Причём при $A \to 2^{-n}$ относительная погрешность может достигать 100%.

В машинах с плавающей запятой абсолютная погрешность представления числа определяется следующим образом:

$$\Delta = \Delta M \cdot 2^{p}, \qquad (2.26)$$

где ΔM – погрешность представления мантиссы, которая определяется так же, как абсолютная погрешность в машинах с фиксированной запятой;

$\Delta M = 2^{-(n+1)}$;

p – порядок числа, который изменяется в пределах

$$-\left(2^{k}-1\right) \le p \le \left(2^{k}-1\right). \qquad (2.27)$$

Следовательно, в отличие от машин с фиксированной запятой в машинах с плавающей запятой абсолютная погрешность представления чисел зависит от порядка числа: минимальная при наибольшем отрицательном p и максимальная при наибольшем положительном p. $\Delta_{\min}$ и $\Delta_{\max}$ определяются следующим образом:

$$\Delta_{\min} = 2^{-(n+1)} \cdot 2^{-(2^1-1)} = 2^{-(n+2^1)};$$
$$\Delta_{\max} = 2^{-(n+1)} \cdot 2^{2^1-1} = 2^{-n+2^1-2}. \quad (2.28)$$

Относительная погрешность представления чисел в машинах с плавающей запятой определяется по общему правилу

$$\varepsilon = \left|\frac{\Delta}{X}\right| = \frac{\Delta M \cdot 2^p}{M \cdot 2^p} = \frac{\Delta M}{M}, \quad (2.29)$$

то есть не зависит от порядка числа и изменяется в следующих пределах:

$$\varepsilon_{\min} = \frac{\Delta X}{X_{\max}} = \frac{2^{-(n+1)}}{1-2^{-n}} = 2^{-(n+1)};$$
$$\varepsilon_{\max} = \frac{\Delta X}{X_{\min}} = \frac{2^{-(n+1)}}{2^{-1}} = 2^{-n}. \quad (2.30)$$

Следовательно, в машинах с плавающей запятой, в отличие от машин с фиксированной запятой, относительная погрешность изображения чисел во всём диапазоне представления практически постоянна и для чисел с нормализованной мантиссой зависит от количества разрядов мантиссы: чем их больше, тем меньше погрешность представления.

2.14. Формы представления двоичных и десятичных чисел в ЭВМ

Знаковый разряд двоичных чисел весом 2^m для целых и 2^0 для дробных чисел участвует совместно с числовыми разрядами в арифметических операциях. Знаковый разряд так же, как цифровые разряды, принимает значение 1 (это знак «–») и 0 (это знак «+»).

Для машинного представления отрицательных чисел используют прямой, обратный и дополнительный коды. При этом знаки чисел кодируются двоичными цифрами: «+» цифрой 0, а «–» цифрой 1.

Любой код положительного числа совпадает с самим числом.

Прямой код (ПК) числа – простейший код, в котором к абсолютной величине числа слева приписывается знаковый признак.

Для целого n-разрядного двоичного числа *X* связь между числом *X* и его изображением в прямом коде имеет вид

$$[X]_n = \begin{cases} X, & \text{если } X \geq 0; \\ 2^n + |X|, & \text{если } X \leq 0. \end{cases} \quad (2.31)$$

Для правильной двоичной дроби прямой код получим исходя из следующего соотношения:

$$[X]_n = \begin{cases} X, & \text{если } X \geq 0; \\ 1 + |X|, & \text{если } X \leq 0. \end{cases} \quad (2.32)$$

Пример. Найти прямой код для отрицательного числа –3 (1011) и –0.3 (0.0011) Пусть разрядная сетка имеет 8 разрядов и один разряд отводится для знака.

Примечание. Здесь и далее для целых чисел условно точкой будем отделять знаковый разряд от цифровой части числа.

На основании (2.30) и (2.31) получаем:

$X_{ПК}$=1.0000011 – для целых чисел;

$X_{ПК}$=1,0000011 – для правильных дробей.

Важная особенность прямого кода в том, что цифру знакового разряда и цифровую часть числа нельзя рассматривать как единое целое. Поэтому для этого кода при умножении и делении чисел операции со знаком и модулями исходных чисел выполняются отдельно. Выполнять операции сложения и вычитания в прямом коде возможно в том случае, если оба числа, участвующие в арифметических операциях имеют одинаковые знаковые разряды (оба положительные либо оба отрицательные). Операция сложения и вычитания в прямом коде неудобна, поскольку прямой код не обеспечивает замену вычитания чисел сложением их кодов. Поэтому перед выполнением операции алгебраического сложения числа преобразуются в один из инверсных кодов (обратный или дополнительный). Прямой код широко используется в ЗУ и устройствах ввода-вывода.

Обратный код (ОК) является дополнением модуля исходного числа до наибольшего числа без знака,

помещающегося в разрядную сетку. Обратный код обладает определенными преимуществами. Он полностью симметричен, так как изображению максимального по абсолютному значению отрицательного числа соответствует $2^m + 2^0 / 2^0 + 2^{-m}$ сопоставляется изображение такого же максимального положительного числа $2^m - 2^0 / 2^0 - 2^{-m}$. Изображение положительных и отрицательных чисел взаимно дополняют друг друга, то есть до последовательности единиц во всех двоичных разрядах, то есть для получения ОК отрицательного числа необходимо взять инверсию всех двоичных разрядов.

Для n – разрядной сетки имеем *для целых чисел*:

$$\left[\overline{X}\right] = \begin{cases} \overline{X}, & \text{при } \overline{X} \geq 0; \\ 2^n - 1 - \left|\overline{X}\right|, & \text{при } \overline{X} \leq 0. \end{cases} \qquad (2.33)$$

Для правильной двоичной дроби

$$\left[\overline{X}\right] = \begin{cases} \overline{X}, & \text{при } \overline{X} \geq 0; \\ 2 - 2^{-n} - \left|\overline{X}\right|, & \text{при } \overline{X} \leq 0. \end{cases} \qquad (2.34)$$

Из соотношений (2.33) и (2.34) видно, что положительное число не меняет своего изображения в обратном коде, а обратный код отрицательного *n*-разрядного двоичного числа определяется из равенства:

$$[X]_0 = 1.\ \overline{x_1}\,\overline{x_2}\ldots\overline{x_n} \text{ – для целых чисел;}$$
$$[X]_0 = 1,\ \overline{x_1}\,\overline{x_2}\ldots\overline{x_n} \text{ – для правильных дробей,} \qquad (2.35)$$

где $\overline{X} = 1 - x_i$.

Пример. Найти обратный код для отрицательных чисел X= –1011 и Y= –0,1011. Результат представить 8-битным числом.

Решение. На основании выражений (2.35) для X и Y получим:

$X_{ОК}$ = 1.1110100; $Y_{ОК}$ = 1,0100111.

В обратном коде можно изображать максимальное положительное число X_{max} = 0,11...11 = $1–2^{-n}$ и наибольшее отрицательное число X_{min} = – 0,11...11 = – $(1–2^{-n})$:

$X_{maxОК}$ = 0,11…11; $X_{minОК}$ = 1,00…00.

Нуль в обратном коде имеет два изображения:

$[+ 0]_0$ = 0,00...00;

$[-0]_0 = 1{,}11...11$.

Изображение чисел в *дополнительном* коде (ДК) наиболее распространенное и не требует каких-либо дополнительных аппаратурных дополнений. Изображение положительных чисел равно значению самих чисел. Например, изображение положительного числа +3 в дополнительном коде будет выглядеть следующим образом:

$X_{ДК}=0.0000011$

Изображение отрицательных чисел представляет собой дополнение до 2^{m-1} для целых чисел и до 2^1 – для дробных. Поэтому сумма изображений двух одинаковых по абсолютному значению чисел разного знака равна $2^{m-1}/2^1$. В общем виде функцию изображения ДК для целых чисел можно представить следующим образом:

$$A_{ДК} = \begin{cases} A, \text{ при } 0 \le A < 2^m; \\ 2^{m+1} - |A|, \text{ при } -2^m \le A < 0; \end{cases} \qquad (2.36)$$

для дробных

$$A_{ДК} = \begin{cases} A, \text{ при } 0 \le A < 2^0; \\ 2^1 - |A|, \text{ при } -2^0 \le A < 0. \end{cases} \qquad (2.37)$$

Следует обратить внимание на несимметричность ДК в связи с тем, что изображению $A = -2^m / -2^0$ не сопоставляется изображение $A = +2^m / +2^0$. Необходимо помнить, что чем больше изображение отрицательного числа, тем меньше абсолютное значение самого изображаемого числа.

Существует два способа перевода чисел из прямого кода в дополнительный. Первый способ заключается в следующем:

1) записать число в прямом коде;

2) для ПК найти соответствующий обратный код;

3) к ОК числа прибавить к младшему разряду единицу;

4) полученное число будет дополнительным кодом заданного числа.

Например, найдем дополнительный код для отрицательного числа –6 (1.0110).

Для перевода числа по первому способу необходимо проделать следующую последовательность действий:

$X_{ПК}$=1.0000110 – прямой код числа –6;

$X_{ОК}$=1.1111001 – обратный код числа –6;

+1;

$X_{ДК}$=1.1111010 – дополнительный код числа –6.

Второй способ перевода заключается в следующем:

1) записать число в прямом коде;

2) найти, просматривая с младших разрядов, первую встретившуюся единицу и все разряды слева от нее перевести в обратный код (за исключением знаковой);

3) все разряды справа от найденной единицы, включая найденную, оставить в прежнем виде.

По второму способу для того же числа получаем следующую последовательность действий:

$X_{ПК}$=1.00001|10

$X_{ДК}$=1.1111010

Важное достоинство дополнительного и обратного кодов заключается в том, что при выполнении арифметических операций сложения и вычитания цифру знакового разряда и цифровую часть числа можно рассматривать как единое целое и обращаться со знаковым разрядом так же, как и с разрядами цифровой части числа.

Во второй главе учебного пособия рассмотрели общие сведения о представлении информации в ЭВМ, основные вопросы, касающиеся информации, систем счисления и чисел, такие как представление информации в цифровых автоматах (ЦА); позиционные системы счисления; методы перевода чисел; форматы представления чисел с плавающей запятой.

Контрольные вопросы

1. Какие из операций с плавающей запятой считаются наиболее сложными? Ответ обоснуйте на конкретных примерах.
2. По каким причинам наибольшее распространение в ЭВМ получила двоичная система счисления?
3. Переведите числа 329; 701; 0,280 из десятичной системы счисления в двоичную, троичную и пятеричную.
4. В чем суть организации циклического переноса и циклического займа при выполнении арифметических операций

в обратном коде?

5. Обоснуйте необходимость игнорирования выходных переносов и входных займов при арифметических операциях в дополнительном коде.

6. Можно ли предугадать одну из двух форм изображения нулевого результата арифметических операций в обратном коде?

7. Переведите числа 329; 701; 0,280 из десятичной системы счисления в двоичную в полулогарифмической форме с основанием порядка X = 16, изображая порядок в смещенном дополнительном коде, а мантиссу — в обычном дополнительном.

8. Объясните необходимость коррекции первичной суммы на плюс шесть и первичной разности на минус шесть при выполнении соответствующих операций с числами, представленными в двоично-десятичном коде прямого замещения (8421).

9. Какие арифметические действия (сдвиги, инкременты, декременты, сложения, вычитания, умножения, деления), в какой последовательности и взаимосвязи необходимо выполнить при реализации четырех основных арифметических операций над числами, представленными в двоичном коде и в полулогарифмической форме (с порядками)? Выполните примеры на сложение, умножение и деление.

10. Какие из следующих цепочек символов являются шестнадцатеричными числами: BED, CAB, DEAD, DECADE, ACCEDED, BAG, DAD?

11. Сколько различных положительных чисел можно выразить в k разрядах, используя числа с основанием системы счисления r?

12. Большинство людей с помощью пальцев на руках могут сосчитать до 10. Однако компьютерщики способны на большее. Представим, что каждый палец соответствует одному двоичному разряду. Пусть вытянутый палец означает 1, а загнутый - 0. До скольки вы сможете сосчитать, используя пальцы обеих рук? А если рассматривать пальцы на руках и ногах? Представим, что большой палец левой ноги - это знаковый бит для чисел с дополнением до двух. Сколько чисел можно выразить таким

способом?

13. Система с дополнением до десяти аналогична системе с дополнением до двух. Отрицательное число в системе с дополнением до десяти получается путем прибавления 1 к соответствующему числу с дополнением до девяти без учета переноса. По какому правилу происходит сложение в системе с дополнением до десяти?

3. ДВОИЧНАЯ АРИФМЕТИКА

Чтобы овладеть любой системой счисления, надо уметь складывать и умножать в ней любые числа. Арифметические действия в двоичной системе счисления выполняют по тем же правилам, что и в десятичной системе, с той лишь разницей, что основание системы равно двум [10].

3.1. Правила двоичной арифметики

Сложение и вычитание двоичных чисел основаны на правилах этих действий в пределах одного разряда и правилах учета межразрядных переносов и займов.

Для операций сложения, вычитания и умножения используются правила, приведенные в таблице 3.1.

Таблица 3.1

Правила арифметических операций

Сложение	Вычитание	Умножение
0+0=0	0-0=0	$0 \cdot 0 = 0$
0+1=1	заем$\rightarrow$ 0-1=1	$0 \cdot 1 = 0$
1+0=1	1-0=1	$1 \cdot 0 = 0$
1+1=1$\rightarrow$перенос	1-1=0	$1 \cdot 1 = 1$

Перенос, возникающий в *i*-м разряде, передается в следующий (*i*+1)-разряд с увеличенным вдвое весом и уменьшенным вдвое значением.

Заем из (*i*+1)-го разряда передается в *i-й* разряд с уменьшенным вдвое весом и увеличенным вдвое значением.

Приведем пример сложения двух двоичных чисел. Справа показано сложение тех же чисел в десятичной системе счисления. Следует обратить внимание на то, что перенос в соседний (старший) разряд возникает в том случае, если сумма цифр данного разряда больше или равна основанию системы счисления.

переносы	1 1 1 1	1 1 1
	1 0 0 1 1, 1 1	1 9, 7 5
+	1 1 0 0 1, 0 1	+ 2 5, 2 5
	1 0 1 1 0 1, 0 0	4 5, 0 0

При вычитании двоичных чисел (см. табл. 3.1) в данном разряде при необходимости занимается единица из соседнего

(старшего) разряда. Эта занимаемая единица равна двум единицам данного разряда. Заем производится каждый раз, когда цифра в разряде вычитаемого больше цифры в том же разряде уменьшаемого. Например, при вычитании:

$$\begin{array}{r} \overset{\curvearrowright}{0}\overset{}{1}0 \\ 10\cancel{1}00011 \\ + \quad\quad 1001 \\ \hline 1001010 \end{array}$$

единица из разряда с весом 2^4 была занята в разряд с весом 2^3; эта единица стала там двойкой, и в разряде с весом 2^3 выполнилось вычитание 10–1 = 1; на месте разряда с весом 2^4 в уменьшаемом фактически остался нуль.

Распространение займа сразу на несколько более старших разрядов можно проследить на примере вычитания чисел $101110{,}001_{(2)}$ и $101{,}011_{(2)}$. Записав числа друг под другом:

$$\begin{array}{r} 101110{,}001 \\ - \quad\quad 101{,}011 \\ \hline \end{array},$$

нетрудно заметить, что в разряде с весом 2^{-2} в результате вычитания должен произойти заем из разряда с весом 2^1. Перепишем пример с учетом фактического расположения цифр после заема и выполним вычитание. Вместо зачеркнутых цифр необходимо использовать в качестве уменьшаемого надписанные цифры. Окончательный результат (разность) составляет $101000{,}110_{(2)}$.

$$\begin{array}{r} \overset{}{\cdot}\overset{1}{1}\overset{}{\cdot}\overset{2}{1} \\ 1011\cancel{1}\cancel{0}{,}\cancel{0}\cancel{0}1 \\ - \quad\quad 101{,}011 \\ \hline 101000{,}110 \end{array}$$

Пример. Уменьшаемое $1000000_{(2)}$, вычитаемое $1_{(2)}$, разность составляет

$$\begin{array}{r} 1000000 \\ - \quad\quad\quad 1 \\ \hline 0111111 \end{array}$$

В соответствии с правилами можно эффективно организовать последовательное умножение множимого на разряды множителя. При каждом умножении на разряд множителя, равный 1, множимое передается в сумматор с

накапливающим регистром; если разряд множителя равен 0, передача множимого в сумматор блокируется. Каждый раз при передаче множимого в сумматор должен быть учтен вес очередного разряда множителя путем сдвига накапливаемого частичного произведения или множимого. Таким образом, основу устройства умножения составляет устройство сложения, к которому добавляются регистры множителя и множимого, а также цепи сдвига частичных произведений и множимого.

Операция деления выполняется путем последовательных вычитаний делителя из промежуточных остатков, а устройство деления состоит из вычитателя с накапливающим регистром, регистра частного и регистра делителя с цепями сдвига остатков или делителя.

В основном арифметические операции выполняются на одном общем устройстве, называемом арифметико-логическим устройством (АЛУ).

Старшие разряды сумматоров с наименьшими весами разрядов участвуют в операциях сложения как обычные числовые разряды, но дополнительно они выполняют функции знаковых разрядов.

3.2. Арифметические операции в двоичной системе счисления

Рассмотрим основные операции (сложение и вычитание) с использованием ОК и ДК. Для примера возьмем числа 25, 31 и проведем с ними все возможные операции сложения и вычитания:

1) 31+25=56

Данный вариант самый простой и не требует никаких преобразований перед сложением, поэтому необходимо просто перевести числа в двоичную систему счисления и сложить с помощью обычной операции сложения:

$31 \rightarrow$ 0.0011111
$25 \rightarrow$ 0.0011001

0.0111000 $\rightarrow$ 32+16+8=56

2) 31+(-25)=+6

В данном случае перед сложением необходимо отрицательное число перевести предварительно либо в ОК, либо в дополнительный код, после чего произвести обычную операцию сложения. В качестве примера переведем ПК в ДК:

$-25_{ПК} \rightarrow$ 1.0011001
$-25_{ДК}$ 1.1100111

После перевода числа в дополнительный код проводим основной цикл сложения:

$+31 \rightarrow$ 0.0011111
+
$-25_{дк} \rightarrow$ 1.1100111

0.0000110 $\rightarrow$ +6

так как «0» в знаковом разряде, то $С_{доп}=С_{ПК}$.

В случае появления знакового разряда со значением «1», результат представлен в ДК и необходим перевод в ПК для получения результата.

3) –31+25=-6

В данном случае отрицательное число необходимо предварительно перевести либо в ДК, либо в обратный код, после чего провести сложение. В случае получения отрицательного числа результат необходимо проинвертировать (заменить единицу на ноль, а ноль на единицу) для получения ПК. В примере воспользуемся переводом в ОК:

$-31_{ПК}$ 1.0011111
$-31_{ОК}$ 1.1100000
$-31 \rightarrow$ 1.1100000
+
$25 \rightarrow$ 0.0011001

1.1111001

После основного сложения, так как результат отрицательный, производим его преобразования в прямой код,

основываясь на методике, описанной в п. 2.14. В результате получаем ПК=1.0000110, то есть число –6.

Сумматор ОК отличается от других сумматоров тем, что в них имеет место цепь обратной связи (циклический перенос). При появлении переноса из знакового разряда к результату необходимо добавить «+1» в младший разряд.

4) 31+(–25)=+6

В данном случае отрицательное число необходимо перевести в обратный код. После перевода получим -$25_{ОК}$=1.1100110.

Производим операцию сложения в ОК:

```
31→              0.0011111
                +
-25ОК→           1.1100110
                 ------------
                ← 0.0000101
                 +
                       +1
                 ------------
                 0.0000110 → +6
```

5) 31-(-25)=+56

По правилам арифметики A-(-B) = A+B. В данном случае два отрицания превращаются в обычное сложение, и поэтому операция сложения в данном случае проводится аналогично примеру 1.

```
31 →             0.0011111
                +
25 →             0.0011001
                 ------------
                 0.0111000 → 32+16+8=56
```

6) –31+(–25)=–56

В данном случае перед сложением необходимо оба числа сначала перевести в обратный код, после чего проводить операцию сложения:

-31 → 1.1100000
-25→ 1.1100110

← 1.1000110
+1

1.0111000

Полученный результат представлен в обратном коде, поэтому для получения ПК его необходимо проинвертировать. Получим 1.0111000 → -56.

Таким образом, используя обратный и дополнительный коды можно выполнить операции сложения и вычитания чисел с различными знаками в двоичной системе счисления.

3.3. Сложение в *D*–кодах

3.3.1.Сложение двоичных чисел в коде прямого замещения (D_1)

Так как наибольшее десятичное одноразрядное число равно 9, то, с учетом переноса в данный разряд, значение результата разрядного суммирования лежит в пределах от 0 до 19. При этом единица во втором разряде представляет собой десятичный перенос в следующую тетраду, а сумма получается в двоичном коде, отличном от требуемого десятичного представления, то есть требует коррекции. В связи с этим при сложении могут возникнуть следующие случаи:

1) если $a_i+b_i+p_i < 10$, то при выполнении действий над разрядами тетрады по правилам двоичной арифметики сразу получается правильный результат;

2) если $a_i+b_i+p_i \geq 10$, то возникает тетрадный перенос, который следует рассматривать как десятичный. Сумму следует скорректировать на +6 (+0110), так как перенос передан в старший разряд с весом 16 вместо 10. Признаком необходимости коррекции является выходной перенос в старшую тетраду (межтетрадный перенос);

3) если $15 \geq a_i + b_i + p_i \geq 10$, то необходима такая же коррекция из-за превышения допустимого значения суммы (т.е. появляется запрещенная комбинация).

Запрещенные комбинации – это следующие значения:

10 = 1010
11 = 1011
12 = 1100
13 = 1101
14 = 1101
15 = 1111

Признаком необходимости коррекции является единица в старшем разряде и хотя бы по одной единице в 2-х соседних кодах первоначальной суммы.

На первом шаге проводим основной цикл сложения, после этого производим коррекцию в тетрадах, имеющих или тетрадный перенос, или в которых возникает в результате сложения запрещенная комбинация.

Пример: 184+298=482 (код 8421)

0001 1000 0100
+
0010 1001 1000

0100←0001 1100
+
0000 0110 0110

0100 1000 0010

В данном примере в младшей тетраде присутствует запрещенная комбинация 1100, значит ее необходимо скорректировать на +6. Из второй тетрады в третью был межтетрадный перенос, поэтому к тетраде, из которой был перенос, прибавим +6 для коррекции результата.

Прямое вычитание десятичных разрядов всегда меньше 10, поэтому разность необходимо скорректировать на минус шесть только при возникновении шестнадцатеричного займа, так как десятичный разряд приобретает в данном случае лишних шесть единиц.

Пример: произвести вычитание чисел в коде прямого замещения (код 8421)

615-396=219

0110 0001 0101
-
0011 1001 0110

0010→0111→1111
-
0000 0110 0110

0010 0001 1001

В данном случае поправки делаются для тех тетрад, для которых был сделан заем.

3.3.2.Сложение двоичных чисел в коде 8421+3 (D_4)

В данном коде перед сложением необходимо представить десятичное число в двоичном виде, основываясь на следующем правиле:

$$a_i' = a_i + 3; \qquad b_i' = b_i + 3,$$

то есть для получения разряда двоичного числа необходимо к исходному десятичному числу прибавить цифру 3 и записать результат в виде тетрады.

В данном коде при сложении могут возникнуть следующие случаи:

1) если $a_i+b_i+p_i \leq 15$, то результат необходимо скорректировать на величину –3 (–0011). Так как операция вычитания производится через операцию сложения, то –3 (–0011) заменяем на 1101 (+13) с блокировкой межтетрадного переноса;

2) если $a_i+b_i+p_i > 15$, то при переходе в старшую тетраду (возник межтетрадный перенос) меняет свой вес с 16 на 10. Поэтому требуется коррекция на величину +3 (0011).

В качестве примера рассмотрим сложение чисел 184 + 298= 482.

0100 1011 0111
0101 1100 1011

1010←1000←0010
-0011 +0011+0011

0111 1011 0101

В данном случае в двух младших тетрадах возникли межтетрадные переносы, поэтому их необходимо скорректировать на величину +3. В старшей тетраде появилось число меньше 15, поэтому эту тетраду необходимо скорректировать на величину –3 (0011).

3.3.3.Сложение в коде D_1 с использованием обратного кода

Как и в обычном двоичном сложении чисел со знаком, перед сложением двоично-десятичных чисел необходимы их предварительные преобразования. Рассмотрим способы сложения с применением обратного кода (*ОК*).

Существует два способа нахождения обратного кода. В первом случае для получения обратного кода числа, записанного в виде тетрад, необходимо для каждой тетрады найти ее дополнение до числа 9. После этого необходимо провести сложение по правилам сложения чисел в коде прямого замещения с учетом необходимых поправок. После получения результата, если число отрицательное, необходимо снова найти дополнение каждой тетрады до числа 9 для получения необходимого результата (кода ПК). В качестве примера рассмотрим сложение чисел:

-298+127=-171

Предварительно перед сложением представим отрицательное число в обратном коде.

$-298_{ПК}$: 1. 0010 1001 1000
$-298_{ОК}$: 1. 0111 0000 0001

В данном случае для дополнения младшей тетрады до числа 9 необходимо число 1, для дополнения второй тетрады необходим ноль, а для дополнения старшей тетрады – число 7. После перевода отрицательного числа в ОК проводим основной цикл сложения:

1. 0111 0000 0001
0. 0001 0010 0111
---
1. 1000 0010 1000

Получили результат, записанный в обратном коде. Для получения конечного результата находим его дополнение до числа 9. Аналогично, дополняя разряды до девяти, получаем:

1. 0001 0111 0001 или -171.

При этом необходимо учитывать, что если существует перенос из знакового разряда, то эту единицу прибавляем к младшей тетраде.

Во втором способе последовательность действий следующая. Первоначально необходимо записать исходное число в виде тетрад в прямом коде. После этого прибавляем к каждой тетраде число +6 (0110), после чего результат инвертируем. Это и будет обратный код исходного числа. После этого проводим обычное сложение, как и в первом способе, после чего при получении в результате сложения отрицательного числа переведем его в ПК путем добавления +6 и инверсии.

1. 0010 1001 1000
+
0. 0110 0110 0110
---
1. 1000 1111 1110
1 0111 0000 0001

Полученное число и будет обратным кодом. Как видно, оно полностью совпадает с обратным кодом, полученным в первом случае. Теперь проводим основной цикл сложения.

1. 0111 0000 0001
+
0. 0001 0010 0111
---
1. 1000 0010 1000
+
0110 0110 0110

1.1110 1000 1110
1.0001 0111 0001 → -171

После основного цикла сложения находим прибавление +6 с последующей инверсией.

3.3.4.Сложение в коде D_4 с использованием кода ДК

Для получения дополнительного кода, записанного в коде D_4, необходимо найти дополнение всех тетрад, кроме младшей, до числа 15, а младшую тетраду – до числа 16. После основного цикла сложения необходимо произвести коррекции. В случае возникновения межтетрадных переносов тетрады, из которых был перенос, необходимо скорректировать на число +3 (0011). Остальные тетрады корректируются на число +13 (1101) с блокировкой межтетрадных переносов. После коррекции, если число отрицательное, то для него также необходимо найти дополнения до 15- и 16-ти, как и в начале вычислений.

В качестве примера возьмем числа –298 и +127 и выполним операцию сложения.

Сначала необходимо отрицательное число перевести в дополнительный код с использованием описанной выше методики. Получаем:

$-298_{ПК}$ 1 0101 1100 1011
$-298_{ДК}$ 1 1010 0011 0101

Затем производим основной цикл сложения:

1. 1010 0011 0101
0. 0100 0101 1010
---
1. 1110 1000 1111
0. 1101 1101 1101 коррекция
---
1. 1011 0101 1100 результат

После основного цикла сложения получаем отрицательное число, которое представлено в дополнительном коде. Для получения прямого кода необходимо найти дополнения младшей тетрады до 16-ти, а всех остальных до 15-ти. Получаем следующий результат:

1. 0100 1010 0100

Учитывая, что полученное число записано в коде D_4, после вычитания из каждой тетрады цифры 3 получим значение –171.

3.3.5.Алгоритм сложения с избытком шесть

Данный алгоритм сложения использует избыточные на плюс шесть значения разрядов одного из слагаемых. Если в разряде суммы возникнет перенос, то значение суммы правильное, так как оно было скорректировано заранее. Если переноса нет, то предварительная коррекция была не нужна, и избыток на плюс шесть необходимо компенсировать вычитанием шести или эквивалентным ему сложением с десятью и с игнорированием выходного переноса (десять есть дополнение шести до шестнадцати).

Для пояснения данного алгоритма выполним сложение 184+298=482 в коде прямого замещения с предварительной коррекцией первого слагаемого на плюс шесть. Получаем следующие тетрады для числа 184:

0111 1110 1010

Второе слагаемое записываем в виде тетрад без изменений.

0010 1001 1000

Проводим основной цикл сложения:

0111 1110 1010
+
0010 1001 1000

1010 ←1000 ←0010

После основного цикла сложения видно, что в первой и второй тетрадах был перенос. Это означает, что эти тетрады были скорректированы удачно и коррекция не нужна. В третьей тетраде никаких переносов не было, и это означает, что предварительная коррекция была не нужна и необходимо выполнить коррекцию. Для этого прибавим к этой тетраде число 10_{10} с блокировкой выходного переноса:

1010 1000 0010
+
1010 0000 0000

0100 1000 0010 → 482

То есть для выполнения операции сложения с избытком шесть, при выполнении представленного выше примера, необходимо выполнить коррекцию только первой тетрады, так как при использовании традиционного алгоритма сложения требовалось бы корректировать две тетрады.

3.4. Выполнение операции умножения в двоичной системе счисления

По сравнению со сложением и вычитанием, умножение – более сложная операция, как при программном, так и при аппаратном воплощении. В ЭВМ применяются различные алгоритмы реализации операции умножения и, соответственно, несколько схем построения операционных блоков, обеспечивающих выполнение операции умножения.

При точном умножении двух чисел количество значащих цифр произведения может в пределе достичь двойного количества значащих цифр сомножителей. Правила приближенных вычислений рекомендуют оставлять в произведении столько же значащих цифр, сколько их содержится в наименее точном из сомножителей.

Наиболее просто операция умножения выполняется в ПК, при этом на первом этапе определяется знак произведения путем сложения знаковых цифр сомножителей по модулю два.

Традиционная схема умножения похожа на известную из школьного курса процедуру записи «в столбик». Вычисление произведения $P(p_{2n-1}, p_{2n-2}, \ldots, p_1, p_0)$ двух n-разрядных двоичных чисел без знака $A(a_{2n-1}, a_{2n-2}, \ldots, a_1, a_0)$ и $B(b_{2n-1}, b_{2n-2}, \ldots, b_1, b_0)$ сводится к формированию частичных произведений (ЧП) W_i (по одному на каждую цифру множителя) с последующим суммированием полученных ЧП. Перед суммированием каждое частичное произведение должно быть сдвинуто на один разряд относительно предыдущего согласно весу цифры множителя, которой это ЧП соответствует. Поскольку операндами являются двоичные числа, вычисление ЧП упрощается: если цифра множителя b_i равна 0, то W_i тоже равно 0, а при $b_i = 1$ частичное произведение равно множимому ($W_i=A$). Перемножение двух n–разрядных двоичных чисел $P = A \times B$ приводит к получению результата, содержащего $2n$ битов. Таким образом, алгоритм умножения предполагает последовательное выполнение двух операций – сложения и сдвига (рис. 3.1).

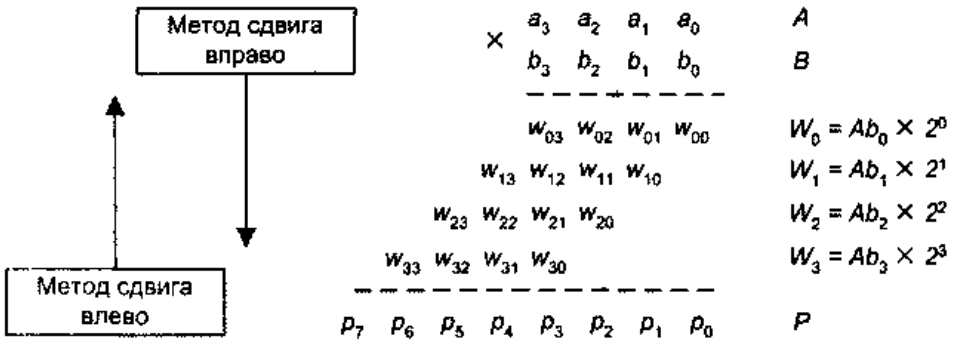


Рис. 3.1. Общая схема умножения со сдвигом суммы частичных произведений влево или вправо

Суммирование ЧП обычно производится не на завершающем этапе, а по мере их получения. Это позволяет избежать необходимости хранения всех ЧП, то есть сокращает аппаратурные издержки. Согласно данной схеме устройство умножения предполагает наличие регистров множимого, множителя и суммы частичных произведений, а также сумматора ЧП и, возможно, схем сдвига, если операция сдвига не реализована иным способом, например, за счет «косой» передачи данных между узлами умножителя.

В зависимости от способа получения суммы частичных произведений (СЧП) возможны четыре варианта реализации «традиционной» схемы умножения [10]:

1) начиная с младших разрядов множителя, со сдвигом суммы частичных произведений вправо и при неподвижном множимом;

2) начиная со старших разрядов множителя, при сдвиге суммы частичных произведений влево и неподвижном множимом;

3) начиная с младших разрядов множителя, при сдвиге множимого влево и неподвижной сумме частичных произведений;

4) начиная со старших разрядов множителя, со сдвигом множимого вправо и при неподвижной сумме частичных произведений.

3.4.1.Умножение младшими разрядами множителя со сдвигом СЧП вправо

Полученное выражение можно представить в виде схемы Горнера для вычисления полиномов

$$C = (((...((0 + A + Ab_n)2^{-1} + Ab_{n-1})2^{-1} + ... + Ab_{n-1})2^{-1} + \\ + ... + Ab_{n-1})2^{-1} + Ab_{n-1})2^{-1}. \quad (3.1)$$

Это выражение может быть сведено к n-кратному выполнению цикла $C_{i+1} = (C_i + Ab_{n-1})2^{-1}$ при начальных условиях i=0; C_0=0.

В каждом цикле множимое либо добавляется к СЧП (если b_i=1), либо нет (если b_i=0), после этого сумма умножается на 2^{-1}, то есть сдвигается на один разряд вправо. После окончания n-го цикла образуется искомое произведение, то есть $C_n=C=AB$.

Очередную цифру множителя, управляющую суммированием частичных произведений, удобнее всего снимать с младшего разряда регистра множителя, в котором в каждом цикле производится сдвиг содержимого на один разряд вправо.

12×13=156

0.0001100 множимое

0.0001101 множитель

0.0000000 СЧП

0|0001100

0000000

0001100 1 шаг

00001100 сдвиг

000001100 2 шаг + 0 → сдвиг

0001100

000111100 3 шаг + A

0000111100 сдвиг

0001100

0010011100 4 шаг + A

00010011100 сдвиг

000010011100 + 0 → сдвиг

0000010011100 + 0 → сдвиг

00000010011100 + 0 → сдвиг

128+16+8+4 =156

Знак результата получается путем сложения знаковых разрядов сомножителей (если знаковые разряды не участвуют в операции умножения) и присваивается результату до операций сложения и сдвига.

3.4.2.Умножение младшими разрядами множителя со сдвигом множимого влево

Выражение представим в следующем виде:

$$C = 2^{-n}\left(Ab_n + 2Ab_{n-1} + \ldots + 2^i Ab_{n-i} + \ldots + 2^{n-1} Ab_{n_1}\right). \qquad (3.2)$$

Вычисление этого выражения сводится к n-кратному выполнению цикла $C_{i+1} = C_i + A_i b_{n-i}$, где A_i=2A_{i-1} при начальных значениях i=0, C_o=0, A_0=A.

В каждом цикле умножения множимое сдвигается на один разряд влево и либо прибавляется к СЧП (при b_i=1), либо нет (при b_i=0).

Например, выполним вычисление 12×13=156:

0.0001100 множимое
0.0001101 множитель
0.0000000 СЧП

0001100
0000000 1 шаг СЧП + А
0001100 СЧП
00011000 сдвиг А на один разряд
000110000 сдвиг А на один разряд (b_i=0)
0001100 СЧП
000111100 СЧП
0001100000 сдвиг А на один разряд
0010011100 СЧП+А (b_i=1)
00011000000 сдвиг А на один разряд
000110000000 сдвиг А на один разряд (b_i=0)
0001100000000 сдвиг А на один разряд (b_i=0)
00011000000000 сдвиг А на один разряд (b_i=0)

В результате имеем 128+16+8+4 =156.

3.4.3.Умножение старшими разрядами множителя со сдвигом СЧП влево

Выражение преобразуется к виду:

$$C = 2^{-(n+1)}((\ldots(\ldots((0 + Ab_1)2 + Ab_2)2 + \ldots + Ab_i)2 + \\ + \ldots + Ab_{n-1})2 + Ab_n)2 \quad (3.3)$$

При этом умножение сводится к *n*-кратному повторению цикла $C_{i+1} = (C_i + Ab_{i+1})2, i = 0, C_0 = 0.$

Тогда управление умножением будет производиться цифрами множителя, начиная со старших разрядов. *СЧП* в каждом цикле будет сдвигаться на один разряд влево.

Например, выполним вычисления 12×13=156:

0.0001100 множимое
0.0001101 множитель
0000000 СЧП
00000000 сдвиг СЧП влево (b_i=0)
000000000 сдвиг СЧП влево (b_i=0)
0000000000 сдвиг СЧП влево (b_i=0)
0001100 + A, т.к. (b_i=1)
0000001100 СЧП
00000011000 сдвиг СЧП влево
0001100 + A, т.к. (b_i=1)
00000100100
000001001000 сдвиг СЧП влево
0000010010000 сдвиг СЧП влево (b_i=0)
0001100 + A, т.к. (b_i=1)
0000010011100 СЧП

Получаем результат 128+16+8+4 = 156.

3.4.4.Умножение старшими разрядами множителя со сдвигом множимого вправо

Выражение представляется в виде

$$C = A2^{-1}b_1 + A2^{-2}b_2 + \ldots + A2^{-i}b_i + \ldots + A2^{-n}b_n. \quad (3.4)$$

Тогда вычисление произведения может быть сведено к *n*-кратному выполнению цикла:

$$A_{i+1} = A_i 2^{-1};$$

$$C_{i+1} = C_i + A_{i+1}b_{i+1}, \qquad i = 0, A_0 = A, C_0 = 0,$$

то есть в каждом цикле множимое сдвигается на один разряд вправо и в зависимости от значения управляющего разряда множителя либо прибавляется к СЧП, либо нет.

Как и в предыдущих случаях, выполним вычисление 12×13=156. После выполнения преобразований в двоичную систему счисления получаем:

0001100 множимое
0001101 множитель
0000000 СЧП
00001100 сдвиг А на разряд вправо, т.к. разряд b_i=0
000001100 сдвиг А на разряд вправо, т.к. разряд b_i=0
0000001100 сдвиг А на разряд вправо, т.к. разряд b_i=0
0000000 + СЧП
0000001100 СЧП
00000001100 сдвиг А на разряд вправо
0000001100 СЧП
00000100100 СЧП
000000001100 сдвиг А на разряд вправо
0000000001100 сдвиг А на вправо, т.к. разряд b_i=0
00000100100
0000010011100 СЧП → 156.

3.5. Умножение чисел со знаком

Несколько сложнее обстоит дело с умножением чисел со знаком, когда *n*-разрядные сомножители содержат знак (в старшем разряде слова) и (*n*-1) значащую цифру. В дальнейшем условимся отделять знаковый разряд точкой, не забывая однако, что знаковый разряд участвует в операции наряду с цифровыми разрядами.

Наиболее очевидная мысль — получить абсолютные значения операндов и перемножить их как числа без знака. Справедливость такого решения видна из примера, приведенного на рис. 3.2 (показан процесс умножения чисел +13 и +10).

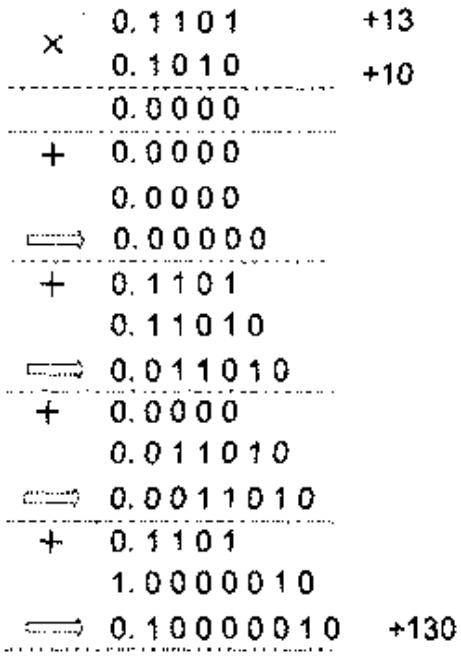


Рис. 3.2. Пример умножения

Во всех вычислительных машинах (ВМ) общепринято представлять числа со знаком в форме с фиксированной запятой в дополнительном коде (ДК). Положительные числа в этом представлении не отличаются от записи в прямом коде, а отрицательные записываются в виде 2^n-x, где x — фактическое значение числа. В двоичной системе запись отрицательного числа в дополнительном коде сводится к инвертированию всех цифровых разрядов числа, представленного в прямом коде, и прибавлению единицы к младшему разряду получившегося после инвертирования обратного кода.

По этой причине более предпочтительны варианты, не требующие преобразования сомножителей и обеспечивающие вычисления непосредственно в дополнительном коде. Первая из особенностей умножения проявляется при выполнении операции арифметического сдвига вправо для суммы частичных произведений – освободившиеся при сдвиге цифровые позиции должны заполняться не нулем, а значением знакового разряда сдвигаемого числа. Здесь, однако, следует учитывать, что это правило заполнения освободившихся цифровых разрядов начинает действовать лишь с момента, когда среди анализируемых разрядов множителя появляется первая единица.

3.5.1.Множимое произвольного знака. Множитель положительный

Пример для положительных сомножителей *(A* > 0, *B* > 0) был рассмотрен выше, в случае же отрицательного множимого процедура умножения протекает аналогично (рис. 3.3), но с учетом сделанного выше замечания об арифметическом сдвиге СЧП.

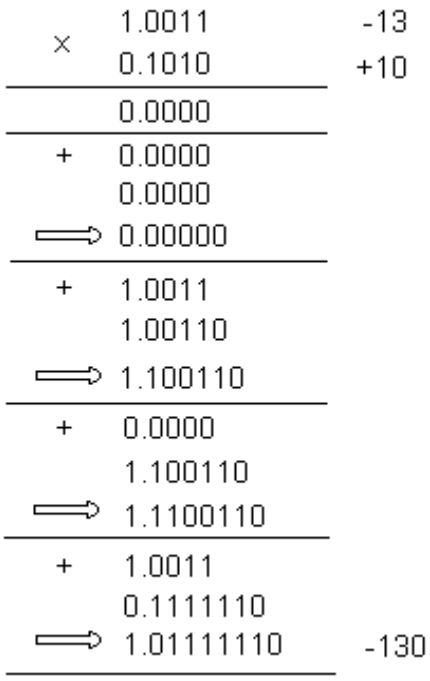


Рис. 3.3. Пример умножения с сомножителями разного знака

Поскольку результат умножения отрицательный, он получается в ДК, и для получения правильного результата его необходимо перевести в ПК.

3.5.2.Множимое произвольного знака. Множитель отрицательный

Так как множитель отрицателен, он записывается в дополнительном коде; $[B]_Д = 2^n - |B|$, и в цифровых разрядах кода будет представлено число $2^{n-1} - |B|$. При типовом умножении (как в случае $B > 0$) получим $P' = A \times \left(2^{n-1} - |B|\right) = -|B| \times A + A \times 2^{n-1}$.

Псевдопроизведение P' больше истинного произведения P на величину $A \times 2^{n-1}$, что и необходимо учитывать при формировании окончательного результата. Для этого перед последним сдвигом из полученного псевдопроизведения необходимо вычесть избыточный член. На рисунках 3.4 и 3.5 приведены примеры умножения положительного и отрицательного множимого на отрицательный множитель, в которых видна упомянутая коррекция результата.

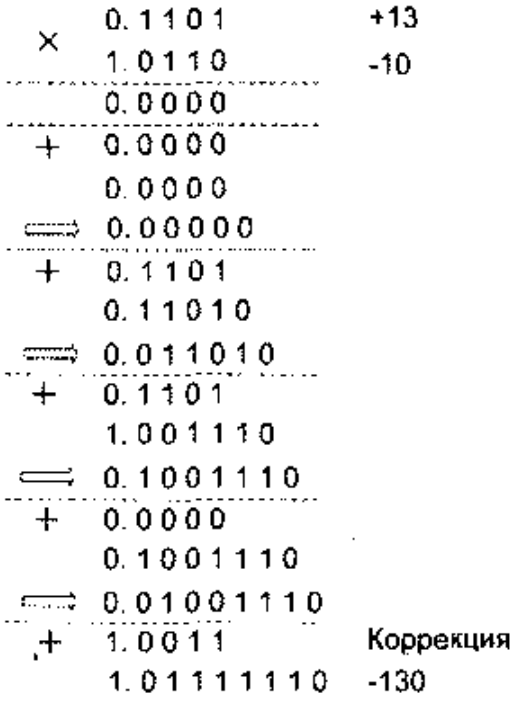


Рис. 3.4. Пример умножения положительного множимого на отрицательный множитель

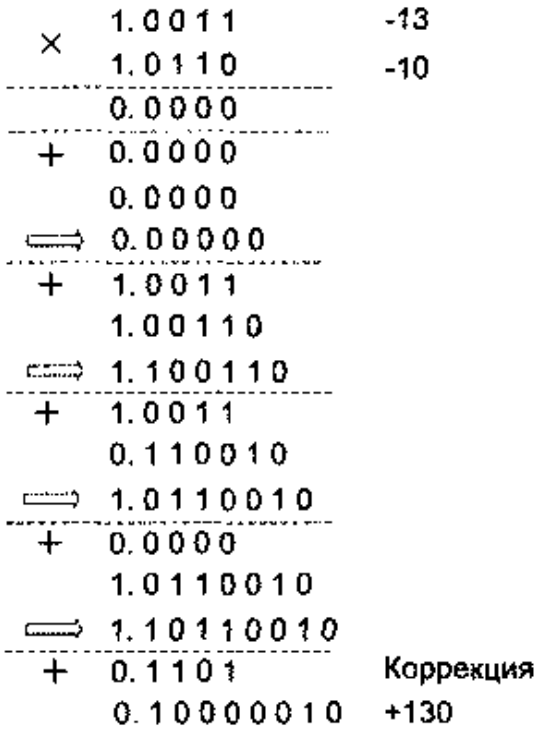


Рис. 3.5. Пример умножения отрицательного множимого на отрицательный множитель

Процесс умножения, представленный на рис. 3.4 и 3.5, протекает аналогично описанным в п/п 3.5 и 3.5.1, за исключением последнего этапа, где необходимо проводить коррекцию. Для коррекции необходимо вычесть избыточный член, которым является множимое.

В примере, представленном на рис. 3.4, множимое положительное, поэтому при вычитании оно имеет отрицательный знак. Для вычитания его необходимо перевести в дополнительный код. В результате перевода образуется число 1.0011, которое прибавляется к псевдопроизведению. После сложения образуется требуемое произведение.

В примере, показанном на рис. 3.5, исходное множимое отрицательное, поэтому при коррекции при вычитании образуется положительный знак, так как –(–13) по правилам арифметики дает положительный знак. Следовательно, необходимо прибавить число 0.1101. В результате сложения с псевдопроизведением получаем требуемое произведение.

3.5.3.Умножение целых чисел и правильных дробей

Рассмотренные алгоритмы относились к представлению чисел с фиксированной запятой, то есть, как это принято в

большинстве ВМ, к целым числам. При перемножении чисел со знаком необходимо принимать во внимание, что произведение двух *n*-разрядных чисел со знаком (знак и (*n*–1) значащий разряд) может иметь 2*(n*–1*)* значащих разрядов и для его хранения обычно используют регистр двойной длины *(*2*n* разрядов).

Поскольку число итераций в операции умножения определяется количеством цифровых разрядов множителя, окончательный результат может размещаться в разрядной сетке двойного слова неверно, что и имеет место при перемножении целых чисел (рис. 3.6).

а – умножение целых чисел

а)

Знак	2^3	2^2	2^1	2^0						
1	0	0	1	1						Множимое
Знак	2^3	2^2	2^1	2^0						
1	0	1	1	0						Множитель
Знак	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
0	1	0	0	0	0	0	1	0		Произведение

б – умножение правильных дробей

б)

Знак	2^{-1}	2^{-2}	2^{-3}	2^{-4}						
1	0	0	1	1						Множимое
Знак	2^{-1}	2^{-2}	2^{-3}	2^{-4}						
1	0	1	1	0						Множитель
Знак	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	
0	1	0	0	0	0	0	1	0		Произведение

Рис. 3.6. Представление произведения в разрядной сетке слова удвоенной разрядности: а – умножение целых чисел; б – умножение правильных дробей

Как видно, младший разряд произведения целых чисел, имеющий вес 2^0, размещается в позиции двойного слова, соответствующей весу 2^1. Таким образом, для правильного расположения произведения в разрядной сетке двойного слова необходим дополнительный сдвиг вправо. Такой сдвиг можно учесть как в аппаратуре умножителя, так и программным способом.

С другой стороны, при перемножении правильных дробей дополнительный сдвиг не нужен. Это обстоятельство необходимо принимать во внимание при построении умножителя для чисел в форме с плавающей запятой, где участвующие в операции мантиссы представлены в нормализованном виде, то есть правильными дробями.

При умножении правильных дробей часто ограничиваются результатом, имеющим одинарную длину. В этом случае может применяться либо отбрасывание лишних разрядов, либо округление.

3.6. Методы ускорения операции умножения

Условно методы ускорения умножения можно разделить на *аппаратные* и *логические*. Те и другие требуют дополнительных затрат оборудования, которые возрастают с увеличением разрядности сомножителей. Аппаратные способы приводят к усложнению схемы умножителя, но не затрагивают схемы управления. Дополнительные затраты оборудования при реализации логических методов не зависят от разрядности операндов, но схема управления умножителя при этом утяжеляется. На практике ускорение умножения часто достигается комбинацией аппаратных и логических методов.

Логические подходы к убыстрению умножения можно подразделить на две группы:

1)методы, позволяющие уменьшить количество сложений в ходе умножения;

2)методы, обеспечивающие обработку нескольких разрядов множителя за шаг.

Реализация и тех и других требует введения дополнительных цепей сдвига в регистры. Рассмотрим первую группу логических методов.

3.6.1.Алгоритм Бута

В основе алгоритма Бута [7,9] лежит следующее соотношение, характерное для последовательностей двоичных цифр:

$$2m + 2^{m-1} + \ldots + 2^k = 2^{m+1} - 2^k, \qquad (3.4)$$

где m и k — номера крайних разрядов в группе из последовательных единиц. Например, 011110 = 2^5 -2^1. Это означает, что при наличии в множителе групп из нескольких единиц (комбинаций вида 011,110), последовательное добавление к СЧП множимого с нарастающим весом (от 2^k до 2^m) можно заменить вычитанием из СЧП множимого с весом 2^k и прибавлением к СЧП множимого с весом 2^{m+1}.

Как видно, алгоритм предполагает три операции: сдвиг, сложение и вычитание. Помимо сокращения числа сложений (вычитаний), у алгоритма есть еще одно достоинство — он в равной степени применим к числам без знака и со знаком.

Алгоритм Бута сводится к перекодированию множителя из системы (0, 1) в избыточную систему (–1,0,1), из-за чего его часто называют перекодированием Бута (Booth recoding). В записи множителя в новой системе: 1 означает добавление множимого к сумме частичных произведений, –1 – вычитание множимого и 0 не предполагает никаких действий. Во всех случаях после очередной итерации производится сдвиг множимого влево или суммы частичных произведений вправо. Реализация алгоритма предполагает последовательный в направлении справа налево анализ пар разрядов множителя – текущего b_i и предшествующего b_{i-1} (b, b_{i-1}). Для младшего разряда множителя (i = 0) считается, что предшествующий разряд равен 0, то есть имеет место пара $b_0$0. На каждом шаге i (i = 0,1,..., n-1) анализируется текущая комбинация b,b_{i-1}.

Комбинация 10 означает начало цепочки последовательных единиц, и в этом случае производится вычитание множимого из СЧП.

Комбинация 01 соответствует завершению цепочки единиц, и здесь множимое прибавляется к СЧП.

Комбинация 00 свидетельствует об отсутствии цепочки единиц, а 11 — о нахождении внутри такой цепочки. В обоих случаях никакие арифметические операции не производятся.

По завершении описанных действий осуществляется сдвиг множимого влево либо суммы частичных произведений вправо, и цикл повторяется для следующей пары разрядов множителя.

Описанную процедуру рассмотрим на примерах (используется вариант со сдвигом множимого влево). В приведенных примерах операция вычитания, как это принято в реальных умножителях, выполняется путем сложения с множителем, взятым с противоположным знаком и представленным в дополнительном коде. Для удлинения кода до нужного числа разрядов в дополнительные позиции слева заносится значение знакового разряда.

Пример: 0110×0011 = 00010010 (в десятичном виде 6×3= 18). После перекодирования Бута множитель (0,0,1,1) приобретает вид (0,1,0,-1).

Вначале сумма частичных произведений принимается равной нулю: 00000000. Полагается, что младшему разряду множителя предшествовал 0. Дальнейший процесс поясняет рис. 3.7.

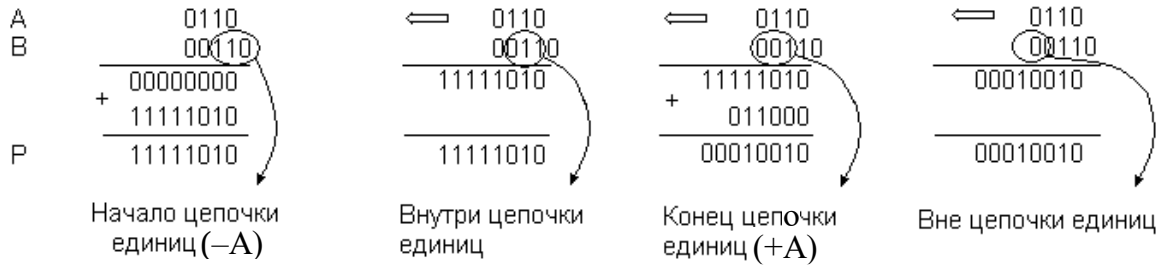


Рис. 3.7. Умножение (6×3) в соответствии с алгоритмом Бута

Пример: В двоичной системе провести умножение чисел 1100×0011=11110100 или в десятичной записи –4×3=–12. Процесс вычисления проиллюстрируем на рис. 3.8.

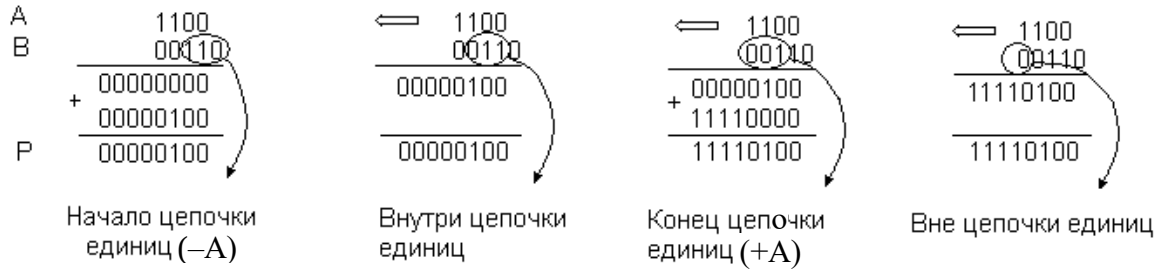


Рис. 3.8. Умножение (-4×3) в соответствии с алгоритмом Бута

Процесс умножения, представленного на рис. рис. 3.7 происходит следующим образом. Анализ разрядов множителя

начинается с младшего разряда, первая, встречающаяся комбинация в множителе является 10, поэтому производится вычитание множимого из суммы частичных произведений, после этого появляется цепочка единиц, поэтому никаких действий не производится. Следующей появляется комбинация 01, поэтому происходит прибавление множимого к сумме частных произведений. На последнем этапе появляется комбинация 00, поэтому никаких действий не происходит. В результате получается искомое произведение.

Умножение чисел, показанных на рис. 3.8 происходит аналогично умножению, показанному на рис. 3.7. Первоначально появляется комбинация 10 и происходит вычитание множимого из суммы частных произведений, затем следует цепь единиц и не производится никаких действий, после этого цепочка единиц заканчивается и производится прибавление множимого к последнему варианту суммы частных произведений. Последней остается комбинация нулей и поэтому не производится никаких действий. Последний вариант суммы частных произведений есть требуемое произведение. На рисунках представлены конечные результаты после выполнения всех промежуточных вычислений.

При наиболее благоприятном сочетании цифр множителя количество суммирований равно n/2, где n — число разрядов множителя.

3.6.2.Модифицированный алгоритм Бута

На практике большее распространение получила модификация алгоритма Бута, где количество операций сложения при любом сочетании единиц и нулей в множителе всегда равно $n/2$. В модифицированном алгоритме производится перекодировка цифр множителя из стандартной двоичной системы (0,1) в избыточную систему (-2,-1,0,1,2), где каждое число представляет собой коэффициент, на который умножается множимое перед добавлением к СЧП. Одновременно анализируются три разряда множителя $b_{i+1}b_ib_{-i}$ (два текущих и старший разряд из предыдущей тройки) и, в зависимости от комбинации 0 и 1 в этих разрядах, выполняется прибавление или

вычитание множимого, прибавление или вычитание удвоенного множимого, либо никакие действия не производятся (табл. 3.2).

Таблица 3.2

Логика модифицированного алгоритма Бута

x_{i+1}	x_i	x_{i-1}	Код (-2-1 0 1 2)	Выполняемые действия
0	0	0	0	Не выполнять никаких действий
0	0	1	1	Прибавить к СЧП множимое
0	1	0	1	Прибавить к СЧП множимое
0	1	1	2	Прибавить к СЧП удвоенное множимое
1	0	0	-2	Вычесть из СЧП удвоенное множимое
1	0	1	-1	Вычесть из СЧП удвоенное множимое
1	1	0	-1	Вычесть из СЧП удвоенное множимое
1	1	1	0	Не выполнять никаких действий

Пример вычисления произведения 011001×101110 = 011000111110 (в десятичном виде 25×(-18)=-450) показан на рис. 3.9.

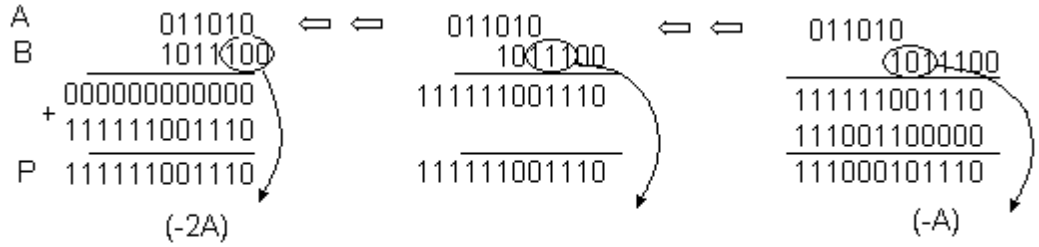


Рис. 3.9. Пример умножения (18× (-25)) в соответствии с модифицированным алгоритмом Бута

На рисунке 3.9 представлены примеры операций без представления перевода из ПК в ДК (алгоритм требует проведения операций вычитания множимого и необходим перевод в ДК). Первоначально производится вычитание из СЧП удвоенного произведения множимого (разряды множителя равны 100), после этого не происходит никаких действий, так как разряды множителя равны 111 и на последнем шаге снова

происходит вычитание из СЧП удвоенного произведения множимого (разряды множителя равны 101). Как видно из примера, ускорение операции умножения достижимо за счет логического анализ разрядов множителя и сокращения количества операций суммирования, что широко используется для сокращения времени операции умножения при проектировании различных вычислительных устройств.

3.6.3.Алгоритм Лемана

Еще большее сокращение количества сложений может дать модификация, предложенная м. Леманом [8]. Здесь, даже при наименее благоприятном сочетании цифр множителя, количество операций сложения не превышает величины *n*/2, а в среднем же оно составляет *n*/3. Суть модификации заключается в следующем:

- если две группы нулей разделены единицей, стоящей в *k-й* позиции, то вместо вычитания в *k-й* позиции и сложения в (k+1)-й позиции достаточно выполнить только сложение в *k-й* позиции;
- если две группы единиц разделены нулем, стоящим в *k-й* позиции, то вместо сложения в *k-й* позиции и вычитания в *(k*+1)-й позиции достаточно выполнить только вычитание в *k*-й позиции.

3.6.4.Обработка двух разрядов множителя за шаг

Из второй группы логических методов остановимся на умножении с обработкой за шаг двух разрядов множителя (IBM 360/370). Анализ множителя начинается с младших разрядов. В зависимости от входящей двухразрядной комбинации предусматриваются следующие действия:

- 00 − ростой сдвиг на два разряда вправо суммы частичных произведений (СЧП);
- 01 − к СЧП прибавляется одинарное множимое, после чего СЧП сдвигается на 2 разряда вправо;
- 10 − к СЧП прибавляется удвоенное множимое, и СЧП сдвигается на 2 разряда вправо;

• 11 − из СЧП вычитается одинарное множимое, и СЧП сдвигается на 2 разряда вправо. Полученный результат должен быть скорректирован на следующем шаге, что фиксируется в специальном триггере признака коррекции.

Так как в случае пары 11 из СЧП вычитается одинарное множимое вместо прибавления утроенного, то для корректировки результата к СЧП перед выполнением сдвига надо было бы прибавить учетверенное множимое. Но после сдвига на два разряда вправо СЧП уменьшается в четыре раза, так что на следующем шаге достаточно добавить одинарное множимое. Это учитывается при обработке следующей пары разрядов множителя: путем обработки пары 00 как 01, пары 01 – как 10, 10 — как 11, а 11 — как 00. В последних двух случаях фиксируется признак коррекции (табл. 3.3).

Таблица 3.3

Формирование признака коррекции

Пара разрядов	+1 из предыду-щей пары	+1 в следующую пару	Знак действия	Кратность множимому
00	0	0		0
01	0	0	+	1
10	0	0	+	2
11	0	1	-	1
00	1	0	+	1
01	1	0	+	2
10	1	1	-	1
11	1	1	+	0

После обработки каждой комбинации содержимое регистра множителя и сумматора частичных произведений сдвигается на 2 разряда вправо. Данный метод умножения требует корректировки результата, если старшая пара разрядов множителя равна 11 или 10 и состояние признака коррекции единичное. В этом случае к полученному произведению должно быть добавлено множимое.

3.7. Умножение чисел в *D*-кодах

Подобный вид умножения сводится к последовательному суммированию частных произведений, получаемых при

умножении множимого на очередную цифру множителя. При этом умножение сопровождается расшифровкой значения очередной 1-й тетрады множителя и сдвигом множителя на четыре разряда сразу. Самым простым приемом расшифровки тетрады является последовательное вычитание единицы из значения тетрады до получения нуля и, соответственно, прибавление множимого к СЧП на каждом такте. Так как при умножении множимого на тетраду возможно переполнение разрядной сетки СЧП, то в ней необходимо предусмотреть дополнительную тетраду для учета возникающих переносов. Применяется способ умножения младшими разрядами множителя со сдвигом СЧП вправо. Умножение обычно производится в прямом коде.

Пример. Умножить на сумматоре прямого кода (код D_1) числа

$$[A]_{пр} = 1{,}1000\ 0110;$$

$$[B]_{пр} = 1{,}0011\ 0011.$$

Решение: При умножении используются сумматор прямого кода для кода D_1 на три тетрады и регистры на две тетрады (табл. 3.4). СЧП на первом шаге алгоритма обнуляется. В приведенном примере при анализе младшей тетрады (0011) выполнено три прибавления множимого к СЧП. Затем при появлении в результате вычитания нуля производятся сдвиги СЧП и множителя на 4 разряда вправо.

Таблица 3.4

Пример операции умножения чисел в D-кодах

СЧП			Регистр В		Примечание
0000	0000	0000	0011	0011	Анализ младшей тетрады множителя. Результат на каждом шаге суммирования СЧП и множимого представлен после проведения коррекции
	+			−	
	1000	0110		1	
0000	1000	0110		0010	
	+			−	
	1000	0110		1	
0001	0111	0010		0001	
	+			−	
	1000	0110		1	
0010	0101	1000		0000	Конец анализа младшей тетрады. Сдвиг на четыре

					разряда
0000	0010	0101	1000	0011	
	+			−	Анализ следующей за младшей тетрады множителя
	1000	0110		1	
0001	0001	0001		0010	
	+			−	
	1000	0110		1	
0001	1001	0111		0001	
	+			−	
	1000	0110		1	
0010	1000	0011		0000	
0000	0010	1000	0011	1000	Конец анализа младшей тетрады. Сдвиг на четыре разряда

Если сомножители имеют по n десятичных разрядов, то в регистре множимого должно быть n разрядов (тетрад) справа от запятой, а в регистре СЧП должно быть n основных разрядов, одна тетрада переполнений и один $(n+1)$-*й* дополнительный десятичный разряд, который предназначен для округления результата. При необходимости сохранить все $2n$ разрядов произведения справа от регистра СЧП должен быть еще сдвиговый регистр для младших разрядов произведения. Регистр множителя должен иметь n тетрад справа от запятой. Причем, если младшая тетрада этого регистра выполнена в виде реверсивного счетчика, (т.е. счетчика, который может прибавлять или вычитать единицу), то операцию умножения можно ускорить в среднем почти в 2 раза. В этом случае необходимо иметь специальный разряд для записи 1, если при суммировании в счетчике–тетраде появится код 10_{10}.

Такой состав оборудования объясняется следующим образом. Если очередная цифра множителя, находящаяся в младшей тетраде регистра множителя, равна или меньше 5_{10}, то производится многократное прибавление множимого к СЧП. При каждом суммировании множимого вычитается 1 из счетчика. Такие действия выполняются до тех пор, пока на счетчике не появится код нуля. Если же очередная цифра множителя равна или больше 6_{10}, то производится многократное вычитание (сложение в дополнительном *D*-коде*)* множимого из СЧП. При каждом вычитании множимого к содержимому счетчика прибавляется 1. Вычитания продолжаются до тех пор, пока в счетчике не появится код 10_{10}, при этом в специальный

разряд записывается 1.

Вторая тетрада множителя при этом приписывается на место младшей тетрады. Затем производится анализ второй тетрады множителя путем вычитания единицы по алгоритму приведенному выше. После сдвигов СЧП и множителя вправо на 4 разряда на этапе анализа второй тетрады для приведенного примера нет больше тетрад множителя, которые необходимо анализировать. В результате использования алгоритма умножения получен результат равный 0, 0000 0010 1000 0011 1000 ($+2838_{10}$). Знак результата определяется с использованием логической функции «сумма по модулю 2».

При умножении в *D*-кодах также можно использовать еще один способ ускорения операции умножения, основанный на следующих формулах:

$$AB = 2A\frac{B-1}{2} + A, \quad \text{если } B \text{ – нечетное число;}$$

$$AB = 2A\frac{B}{2}, \quad \text{если } B \text{ – четное число.}$$

При двоично-десятичном представлении чисел удвоение числа означает сдвиг влево, а деление на 2 – сдвиг вправо, причем так как сдвиги производятся над двоичными кодами, требуется коррекция тетрад на каждом шаге. Корректирующие поправки определяются для каждого *D*-кода. Коррекция выполнятся тогда, когда происходит сдвиг единицы из крайнего разряда данной тетрады в соседнюю тетраду. Для кода D_1 корректирующая поправка равна +0110 для тетрад множимого и –0011 (или +1101) для тетрад множителя.

Пример. Умножить ускоренным способом (1) на сумматоре прямого кода числа (табл. 3.5):

$[A]_{пр} = 1{,}1000\ 0110;$

$[B]_{пр} = 1{,}0011\ 0011.$

На примере, представленном в табл. 3.5 приведена операция умножения числа -86 на -33, записанных потетрадно в двоичном коде. Здесь в колонке *B* показан множитель, то есть число -33, в колонке А записано множимое, то есть число -86,

колонка С иллюстрирует процесс накопления произведения в СЧП. В примере показаны числа после требуемых коррекций. На первом шаге множитель отрицательный (33), следовательно, его необходимо добавить к СЧП. После этого производится сдвиг множителя на разряд вправо, а множимого – на разряд влево, при этом в множителе происходит межтетрадный перенос, в младшей тетераде множимого появляется запрещенная комбинация (число 1100), а во второй – произошел межтетрадный перенос. Следовательно разряды А и В требуют коррекции, младшую тетраду В следует скорректировать на +13 с блокировкой межтетрадного переноса, а тетрады А на +0110. После коррекции число В положительно, поэтому добавления множимого к С не происходит, а производится очередная операция сдвигов множимого и множителя. На каждом шаге производится анализ необходимости коррекции в тетрадах множимого и множителя, и добавления скорректированного множимого к С. В С в случае необходимости необходимо производить коррекции на +0110. Операция сдвигов прекращается, когда тетрада множителя станет равной 0001, при этом происходит последнее добавления А к С (значение 0001 является нечетным).

Таблица 3.5

Пример ускоренной операции умножения

B	*A*	*C*	Примечание

0011	0011		0000	1000	0110	0000	0000	1000	0110	
0001	1001		0001	0000	1100					
+		+								Сдвиг
	1101			0110	0110					
0001	0110		0001	0111	0010					Поправки *B* – четное
0000	1011		0010	1110	0100					
+		+								Сдвиг
	1101			0110	.					
0000	1000		0011	0100	0100					Поправки *B* – четное
0000	0100		0110	1000	1000					Сдвиг *B* – четное
0000	0010		1101	0001	0000					
		+								Сдвиг
			0110	0110	0110					
		0001	0011	0111	0110					Поправки *B* – четное
										Сдвиг
0000	0001	0010	0100	1110	1100					
		+								Поправки *B* – нечетное
				0110	0110					
		0010	0111	0101	0010	+				
						0010	0111	0101	0010	
						0010	0111	1101	1000	
						+				Поправка
								0110		
						0010	1000	0011	1000	

3.8. Аппаратные методы ускорения умножения

Традиционный метод умножения за счет сдвигов и сложений, даже при его аппаратной реализации, не позволяет достичь высокой скорости выполнения операции умножения. Связано это, главным образом, с тем, что при добавлении к СЧП очередного частичного произведения перенос должен распространиться от младшего разряда СЧП к старшему. Задержка из-за распространения переноса относительно велика, причем она повторяется при добавлении каждого ЧП.

Один из способов ускорения умножения состоит в изменении системы кодирования сомножителей, за счет чего можно сократить количество суммируемых частичных произведений. Примером такого подхода может служить алгоритм Бута.

Еще один ресурс повышения производительности умножителя – использование более эффективных способов суммирования ЧП, исключающих затраты времени на распространение переносов. Достигается это за счет представления ЧП в избыточной форме, благодаря чему суммирование двух чисел не связано с распространением переноса вдоль всех разрядов числа. Наиболее употребительной формой такого избыточного кодирования является так называемая форма *с сохранением переноса.* В ней каждый разряд числа представляется двумя битами *cs*, известными как перенос (*c*) и сумма (*s*). При суммировании двух чисел в форме с сохранением переноса перенос распространяется не далее, чем на один разряд. Это делает процесс суммирования значительно более быстрым, чем в случае сложения с распространением переноса вдоль всех разрядов числа.

Наконец, третья возможность ускорения операции умножения заключается в параллельном вычислении всех частичных произведений. Если рассмотреть общую схему умножения (рис. 3.10), то нетрудно заметить, что отдельные разряды ЧП представляют собой произведения вида $a_i b_j$, то есть произведение определенного бита множимого на определенный бит множителя. Это позволяет вычислить все биты частичных произведений одновременно, с помощью n^2 схем И. При перемножении чисел в дополнительном коде отдельные разряды ЧП могут иметь вид $a_i \overline{b_j}$, $\overline{a_i} b_j$ или $\overline{a_i b_j}$. Тогда элементы И заменяются элементами, реализующими соответствующую логическую функцию.

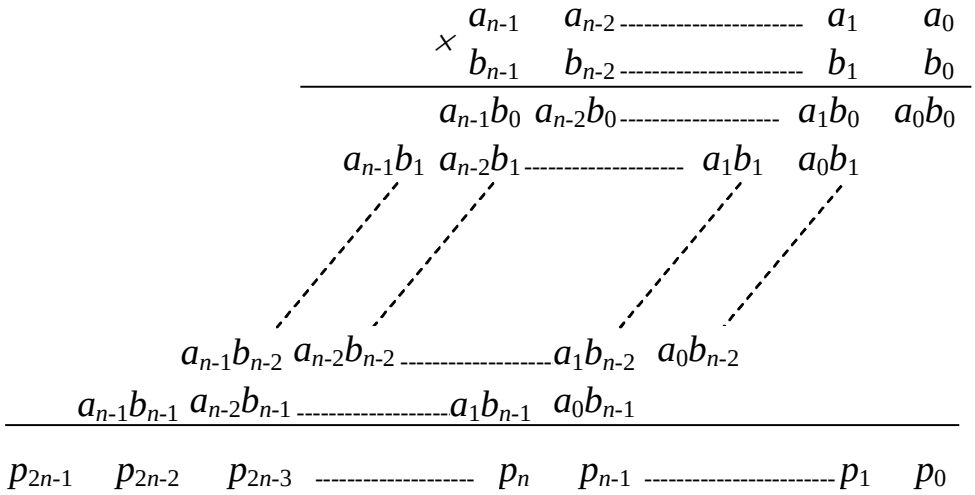


Рис. 3.10. Схема перемножения *n*-разрядных чисел без знака

Таким образом, аппаратные методы ускорения умножения сводятся:

- к параллельному вычислению частичных произведений;
- к сокращению количества операций сложения;
- к уменьшению времени распространения переносов при суммировании частичных произведений.

Все три подхода в любом их сочетании обычно реализуются с помощью комбинационных устройств.

Параллельное вычисление ЧП имеет место практически во всех рассматриваемых ниже схемах умножения. Различия проявляются в основном в способе суммирования полученных частичных произведений, и с этих позиций используемые схемы умножения можно подразделить на *матричные* и *с древовидной структурой* [9]. В обоих вариантах суммирование осуществляется с помощью массива взаимосвязанных одноразрядных сумматоров. В матричных умножителях сумматоры организованы в виде матрицы, а в древовидных они реализуются в виде дерева того или иного типа.

Различия в рамках каждой из этих групп выражаются в количестве используемых сумматоров, их виде и способе распространения переносов, возникающих в процессе суммирования.

В *матричных умножителях* суммирование осуществляется матрицей сумматоров, состоящей из последовательных линеек (строк) одноразрядных сумматоров с сохранением переноса (ССП). По мере движения данных вниз по массиву сумматоров каждая строка ССП добавляет к СЧП очередное частичное произведение. Поскольку промежуточные СЧП представлены в избыточной форме с сохранением переноса, во всех схемах, вплоть до последней строки, где формируется окончательный результат, распространения переноса не происходит. Это означает, что задержка в умножителях отталкивается только от «глубины» массива (числа строк сумматоров) и не зависит от разрядности операндов, если только в последней строке

матрицы, где формируется окончательная СЧП, не используется схема с последовательным переносом.

Наряду с высоким быстродействием важным достоинством матричных умножителей является их регулярность, что особенно существенно при реализации таких умножителей в виде интегральной микросхемы. С другой стороны, подобные схемы занимают большую площадь на кристалле микросхемы, причем с увеличением разрядности сомножителей эта площадь увеличивается пропорционально квадрату числа разрядов. Вторая проблема с матричными умножителями – это низкий уровень утилизации аппаратуры. По мере движения СЧП вниз каждая строка задействуется лишь однократно, когда ее пересекает активный фронт вычислений. Это обстоятельство, однако, может быть затребовано для повышения эффективности вычислений путем конвейеризации процесса умножения, при которой по мере освобождения строки сумматоров последняя может быть использована для умножения очередной пары чисел.

3.9. Выполнение операции деления

Деление несколько более сложная операция, чем умножение, но базируется на тех же принципах. Основу составляет общепринятый способ деления с помощью операций вычитания или сложения и сдвига (рис. 3.11).

z-делимое	$z_7\ z_6\ z_5\ z_4\ z_3\ z_2\ z_1\ z_0$	$d_3\ d_2\ d_1\ d_0$	*D–делимое*
$-Dq_3\times2^3$	$r_3\ r_2\ r_1\ r_0$	$q_3\ q_2\ q_1\ q_0$	*Q–частное*
$-Dq_3\times2^2$	$r_3\ r_2\ r_1\ r_0$		
$-Dq_3\times2^1$	$r_3\ r_2\ r_1\ r_0$		
$-Dq_3\times2^0$	$r_3\ r_2\ r_1\ r_0$		
	$s_3\ s_2\ s_1\ s_0$		*S–остаток*

Рис. 3.11. Общая схема операции деления

Задача сводится к вычислению частного Q и остатка S:

$$Q = \mathrm{int}\left(\frac{Z}{D}\right),\ S = Z - QD,\ S < D. \qquad (3.5)$$

Деление выражается как последовательность вычитаний делителя сначала из делимого, а затем из образующихся в

процессе деления частичных остатков (ЧО). Делимое $Z(z_{2n-1}z_{2n-2}\dots z_1z_0)$ обычно представляется двойным словом (*2n* разрядов), делитель $D(d_{2n-1}d_{2n-2}\dots d_1d_0)$, частное $Q(q_{2n-1}q_{2n-2}\dots q_1q_0)$ и остаток $S(s_{2n-1}s_{2n-2}\dots s_1s_0)$ имеют разрядность *n*.

Операция выполняется за *n* итераций и может быть описана следующим образом:

$$S^{(i)} = 2S^{(i-1)} - q_{n-i}\left(2^n D\right), \text{ при } S^{(0)} = Z \text{ и } S^{(n)} = 2^n S; \qquad (3.6)$$

$$q_{n-i} = \begin{cases} 1, \text{ если } \left(2S^{(i-1)} - 2^n D\right) \geq 0, \\ 0, \text{ если } \left(2S^{(i-1)} - 2^n D\right) < 0. \end{cases} \qquad (3.7)$$

После *n* итераций получается

$$S^{(n)} = 2^n S^{(0)} - Q\left(2^n D\right) - 2^n\left[Z - (Q \times D)\right] = 2^n S. \qquad (3.8)$$

Частное от деления 2*n*-разрядного числа на *n*-разрядное может содержать более, чем *n* разрядов. В этом случае возникает переполнение, из-за чего перед выполнением деления необходима проверка условия

$$Z < \left(2^n - 1\right)D + D = 2^n D. \qquad (3.9)$$

Из выражения следует, что переполнения не будет, если число, содержащееся в старших *n* разрядах делимого, меньше делителя.

Помимо этого требования, перед началом операции необходимо исключить возможность ситуации деления на 0.

Реализовать деление можно двумя основными способами:

1) с неподвижным делимым и сдвигаемым вправо делителем;

2) с неподвижным делителем и сдвигаемым влево делимым.

Недостатком первого способа является потребность иметь в устройстве деления сумматор и регистр двойной длины. Второй способ позволяет строить делитель с сумматором одинарной длины. Неподвижный делитель *D* хранится в регистре одинарной длины, а делимое Z, сдвигаемое относительно *D,* находится в двух таких же регистрах. Образующиеся цифры частного *Q* заносятся в освобождающиеся при сдвиге Z разряды одного из регистров Z.

Ниже на примере чисел без знака рассматриваются два основных алгоритма целочисленного деления.

Наиболее очевидный алгоритм носит название алгоритма *деления с неподвижным делителем и восстановлением остатка.* В учебном пособии он представлен в силу того, что очень похож на общепринятый способ деления столбиком. Данный алгоритм может быть описан следующим образом:

1) исходное значение частичного остатка полагается равным старшим разрядам делимого;
2) частичный остаток удваивается путем сдвига на один разряд влево. При этом в освобождающийся при сдвиге младший разряд ЧО заносится очередная цифра частного.
3) из сдвинутого ЧО вычитается делитель и анализируется знак результата вычитания;
4) очередная цифра модуля частного равна единице, когда результат вычитания положителен, и нулю, если отрицателен. В последнем случае значение остатка восстанавливается до того значения, которое было до вычитания;
5) пункты 2-4 последовательно выполняются для получения всех цифр модуля частного.

На рисунке 3.12 показан процесс деления с восстановлением остатка (здесь число 41 делится на 7).

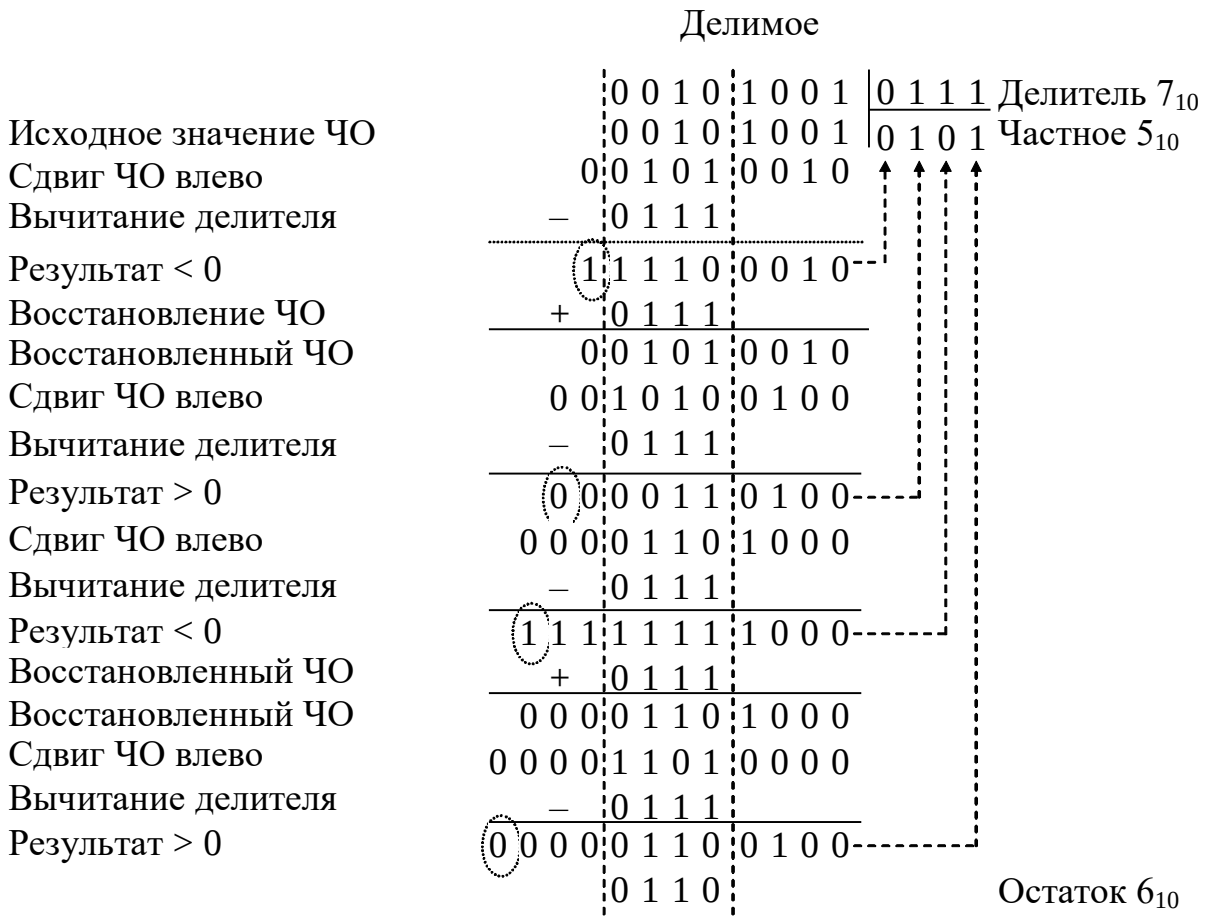


Рис. 3.12 Пример деления с восстановлением остатка

На первом шаге алгоритма деления производится операция сдвига делимого на один разряд влево (рис. 3.12), при этом в освобождающийся разряд записывается ноль. После этого происходит вычитания из сдвинутого делимого числа разрядов делителя. Затем анализируется старший разряд частичного остатка. Если он равен 1, то есть число меньше нуля, то в разряд частного записывается 0, если – 0 (число больше нуля), то в частное записываем 1. В нашем случае он равен 1, поэтому в частное записываем ноль и производим восстановление остатка с помощью добавления делителя. Результат суммирования сдвигаем на разряд влево. На этом первый шаг деления закончен и для получения очередной цифры частного необходимо снова вычесть разряды делителя. В примере на рис. 3.12 результат вычитания больше нуля (старший разряд равен 0), поэтому в частное записываем 1. операцию необходимо продолжить до получения количества разрядов, равных количеству разрядов

делителя. В итоге получаем значение частного равное 0101 (5_{10}) и в остатке 0110.

Недостаток алгоритма деления с восстановлением остатков заключается в необходимости выполнения на отдельных шагах дополнительных операций сложения для восстановления частичного остатка. Это увеличивает время выполнения деления, которое в этом случае может меняться в зависимости от конкретного сочетания кодов операндов. В силу указанных причин реальные делители строятся на основе алгоритма *деления с неподвижным делителем без восстановления остатка.* Приведем описание этого алгоритма:

1) исходное значение частичного остатка полагается равным старшим разрядам делимого;

2) частичный остаток удваивается путем сдвига на один разряд влево, при этом в освобождающийся при сдвиге младший разряд ЧО заносится очередная цифра частного;

3) из сдвинутого частичного остатка вычитается делитель, если остаток положителен, и к сдвинутому частичному остатку прибавляется делитель, если остаток отрицательный;

4) очередная цифра модуля частного равна единице, когда результат вычитания положителен, и нулю, если он отрицателен;

5) пп. 2-4 последовательно выполняются для получения всех цифр модуля частного.

Как видим, пп. 1, 2, 5 полностью совпадают с соответствующими пунктами предыдущего алгоритма деления.

Процесс деления без восстановления остатка для ранее рассмотренного примера демонстрируется на рис. 3.13.

Алгоритм деления без восстановления остатка аналогичен алгоритму деления с восстановлением остатков. Отличие состоит в том, что после появления отрицательного числа после вычитания производится сдвиг на единицу влево, после чего к сдвинутому остатку добавляется делитель. Полученный частичный остаток анализируем на значение больше нуля или меньше нуля, и в зависимости от этого аналогично предыдущему алгоритму записываем единицу или ноль в разряд частного. В остальном, процесс деления полностью аналогичен алгоритму деления с восстановления остатков.

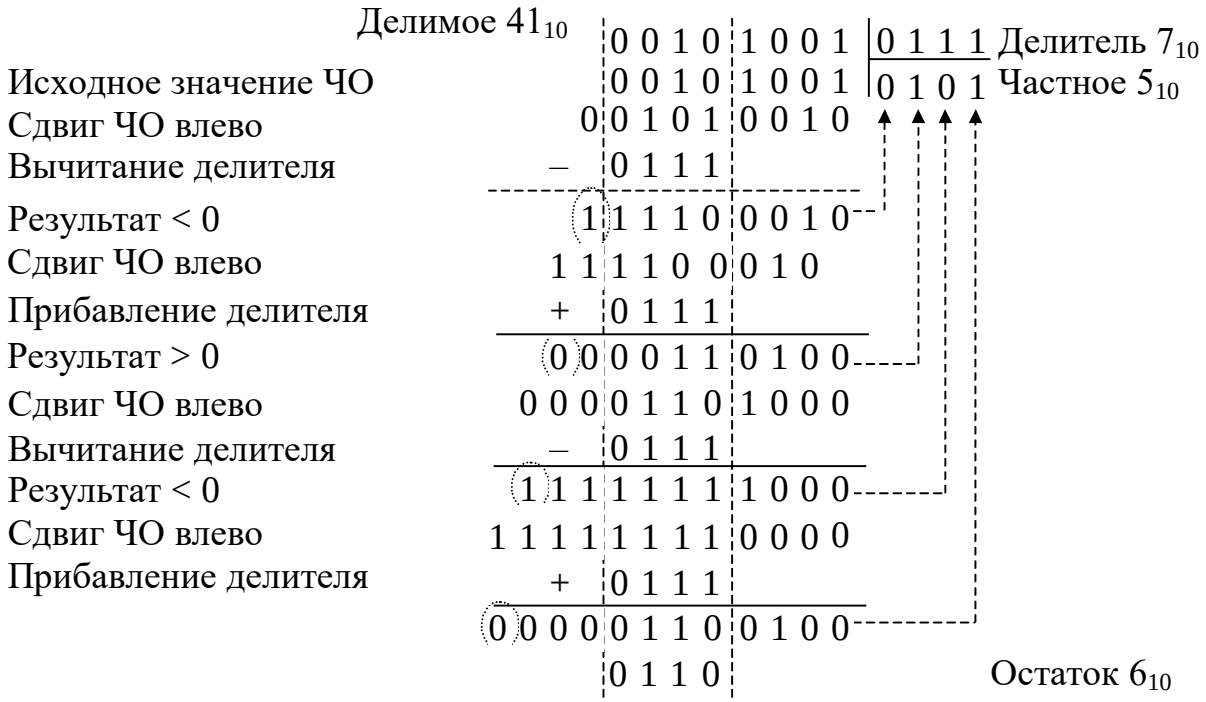


Рис. 3.13. Пример деления без восстановления остатка

3.10. Деление чисел со знаком

Как и в случае умножения, деление чисел со знаком может быть выполнено путем перехода к абсолютным значениям делимого и делителя с последующим присвоением частному знака «плюс» при совпадающих знаках делимого и делителя либо «минус» – в противном случае.

Деление чисел, представленных в дополнительном коде, можно осуществлять, не переходя к модулям. Рассмотрим необходимые для этого изменения в алгоритме без восстановления остатка.

Так как делимое и делитель не обязательно имеют одинаковые знаки, то действия с частичным остатком (прибавление или вычитание делителя) зависят от знаков остатка и делителя (табл. 3.4).

- если знак остатка совпадает со знаком делителя, то очередная цифра частного – 1, иначе – 0;
- если $Z > 0$ и $D < 0$, частное необходимо увеличить на 1;
- если $Z < 0$ и $D > 0$, то при ненулевом остатке от деления частное нужно увеличить на единицу;

• если Z < 0 и D < 0, то при нулевом остатке от деления частное нужно увеличить на единицу.

Таблица 3.4.

Операция, выполняемая в очередной итерации деления

Знак остатка	Знак делителя	Действие
+	+	Вычитание делителя
+	-	Прибавление делителя
-	+	Прибавление делителя
-	-	Вычитание делителя

Остаток всегда приводится к положительному числу, то есть, если по завершении деления он отрицателен, к нему следует прибавить модуль делителя.

3.11. Деление в избыточных системах счисления

Наиболее распространенные методы ускорения операции деления основаны на применении алгоритмов, где частное представляется в системе счисления, отличной от двоичной. Это означает, что цифры частного могут иметь больше, чем два значения, например {-1,0,1}, как это было в алгоритме умножения Бута, или {-2,-1,0,1,2}. В таких системах одно и то же число может быть записано несколькими способами, из-за чего системы называют избыточными. Очередная цифра частного в избыточной системе счисления, в зависимости от базы этой системы, соответствует двум или более цифрам в двоичном представлении частного, и для нужного количества двоичных цифр частного и остатка требуется меньше итераций. В то же время реализация такого подхода ведет к усложнению аппаратуры делителя, в частности, надстраивается логика определения операции, выполняемой в очередной итерации. Для этой цели в состав устройства деления включается специальная память, хранящая таблицу, определяющую необходимые действия в зависимости от текущей комбинации цифр в частичном остатке и делителе. Тем не менее выигрыш в быстродействии оказывается решающим моментом. Так, в микропроцессорах Pentium при делении мантисс чисел с плавающей запятой используется алгоритм SRT с базой 4, то есть частное сначала вычисляется с использованием цифр -2, -1,

0, 1, 2 с последующим преобразованием результата к стандартному двоичному представлению. В этом варианте выбор очередной цифры частного производится с помощью таблицы, состоящей из отдельных секций. Конкретную секцию определяют четыре старшие цифры делителя (после его нормализации). Входом в секцию служат шесть старших цифр частичного остатка. ЧО в каждой итерации сдвигается не на один, а на два разряда, то есть число итераций сокращается вдвое. Известны варианты делителей, где берется еще большее основание системы счисления, в частности 8 и 16. В этом случае логика работы устройства существенно усложняется.

3.12. Замена деления умножением на обратную величину

Операцию деления на D можно заменить умножением на

$$\frac{1}{D}:Q=\frac{Z}{D}=Z\times\frac{1}{D}, \qquad (3.10)$$

где D – это делитель

В этом случае проблема сводится к эффективному вычислению $1/D$. Обычно задача решается одним из двух методов: с помощью ряда Тейлора или метода Ньютона–Рафсона [9]. В обоих случаях основное время расходуется на умножение, поэтому рассматриваемый метод ускорения деления имеет смысл при наличии быстрых схем умножения.

При реализации первого метода делитель D представляется в виде: $D = 1 + X$. Тогда для двоичного представления D можно записать:

$$\frac{1}{D}=(1-X)\times\left(1+X^2\right)\times\left(1+X^4\right)\times\left(1+X^8\right)\times\left(1+X^{16}\right)...$$

Метод был использован в модели 91 вычислительной машины IBM 360 для вычисления 32-разрядной величины $1/D$. Возможные значения сомножителей в правой части выражения извлекались из таблицы емкостью 28 байт, хранящейся в памяти. Операция вычисления $1/D$ требует шести умножений.

Вычисление величины $1/D$ методом Ньютона–Рафсона сводится к нахождению корня уравнения

$$f(X)=\frac{1}{X}-D=0, \qquad (3.11)$$

то есть $X = 1/D$. Решение может быть получено с привлечением рекуррентного соотношения $X_{i+t} = X_i\ (2\ -X_iD)$. Количество итераций определяется требуемой точностью вычисления $1/D$. Реализация метода для n-разрядных чисел требует $2 \cdot \text{int}(\log_2 n)$-1 операций умножения.

В общем, замена операции деления на умножение более характерна для чисел с плавающей запятой.

3.13. Ускорение операции деления

В основе методов ускорения операции деления лежит так называемый алгоритм SRT [11,12]. Свое название алгоритм получил по фамилиям авторов (Sweney, Robertson, Tocher), разработавших его независимо друг от друга приблизительно в одно и то же время. Этот алгоритм представляет собой модификацию деления без восстановления остатка. В стандартной процедуре на каждом шаге помимо сдвига частичного остатка производится прибавление либо вычитание делителя. В SRT-алгоритме сдвиг частичных остатков (ЧО) также имеется в каждой итерации, однако сложение или вычитание, в зависимости от получающегося ЧО, на отдельных шагах может не выполняться, что, естественно, позитивно влияет на быстродействие деления.

Алгоритм был ориентирован на операции над мантиссами чисел с плавающей запятой и опирается на то обстоятельство, что мантиссы в таких числах нормализованы. Впервые SRT-алгоритм был реализован в модели 91 вычислительной машины IBM 360. В настоящее время он широко применяется в блоках обработки чисел с плавающей запятой, в частности в микропроцессорах фирмы Intel.

Сначала рассмотрим алгоритм применительно к положительным целым числам. Делимое представляется $(2n+1)$-разрядным числом, а делитель – n-разрядным. Процедура деления начинается с удаления в делителе всех нулей, предшествующих старшей единице, то есть с операции, аналогичной нормализации мантиссы в числах с плавающей запятой. Будем условно называть эту операцию нормализацией. Исключение k предшествующих нулей реализуется за счет

сдвига делителя влево на k разрядов. На аналогичное число разрядов влево сдвигается и делимое. Далее выполняются n итераций, в которых вычисляются цифры частного и частичные остатки. Действия, выполняемые на i-$й$ итерации, можно описать следующим образом:

$$q_i = \begin{cases} 1, \text{если } 2S^{(i-1)} \geq D; \\ 0, \text{если } -\mathrm{D} \leq 2S^{(i-1)} < D; \\ -1, \text{если } 2S^{(i-1)} < -D, \end{cases} \qquad (3.12)$$

$$S^{(i)} = 2S^{(i-1)} - q_i D.$$

Частное представляется в системе счисления, отличной от двоичной. Это означает, что цифры частного могут иметь больше чем два значения 0 и 1. В рассматриваемом случае 1,0,1.

По завершении всех n итераций, если последний остаток отрицателен, выполняется коррекция этого остатка и полученного частного, для чего к остатку прибавляется делитель, а из частного вычитается единица с весом младшего разряда.

Последний этап в алгоритме – преобразование частного из системы {-1 , 0, 1} в систему {0,1}, то есть в обычную двоичную систему.

На практике используется следующая процедура. Делимое и делитель, представленные в дополнительном коде, размещаются в регистре делимого (РДМ) и делителя (РДТ) соответственно. Дальнейшие действия можно описать следующим образом:

1) если в делителе D имеются k предшествующих нулей (при $D>0$) или предшествующих единиц (при $D<0$), то производится предварительный сдвиг содержимого РДМ и РДТ влево на k разрядов;

2) для i от 0 до n-1:

- если три старших цифры частичного остатка в РДМ совпадают, то $q_i=0$ и производится сдвиг содержимого РДМ на один разряд влево;
- если три старших цифры частичного остатка в РДМ не совпадают, а сам ЧО отрицателен, то q_i = -1, выполняется сдвиг

содержимого РДМ на один разряд влево и к ЧО прибавляется делитель;

- если три старших цифры частичного остатка в РДМ не совпадают, а сам ЧО положителен, то $q_i = 1$, выполняется сдвиг содержимого РДМ на разряд влево и из ЧО вычитается делитель;

3) если после завершения п. 2 остаток отрицателен, то производится коррекция (к остатку прибавляется делитель, а из частного вычитается единица);

4) остаток сдвигается вправо на k разрядов.

Рассмотрим пример реализации данного алгоритма (рис. 3.14).

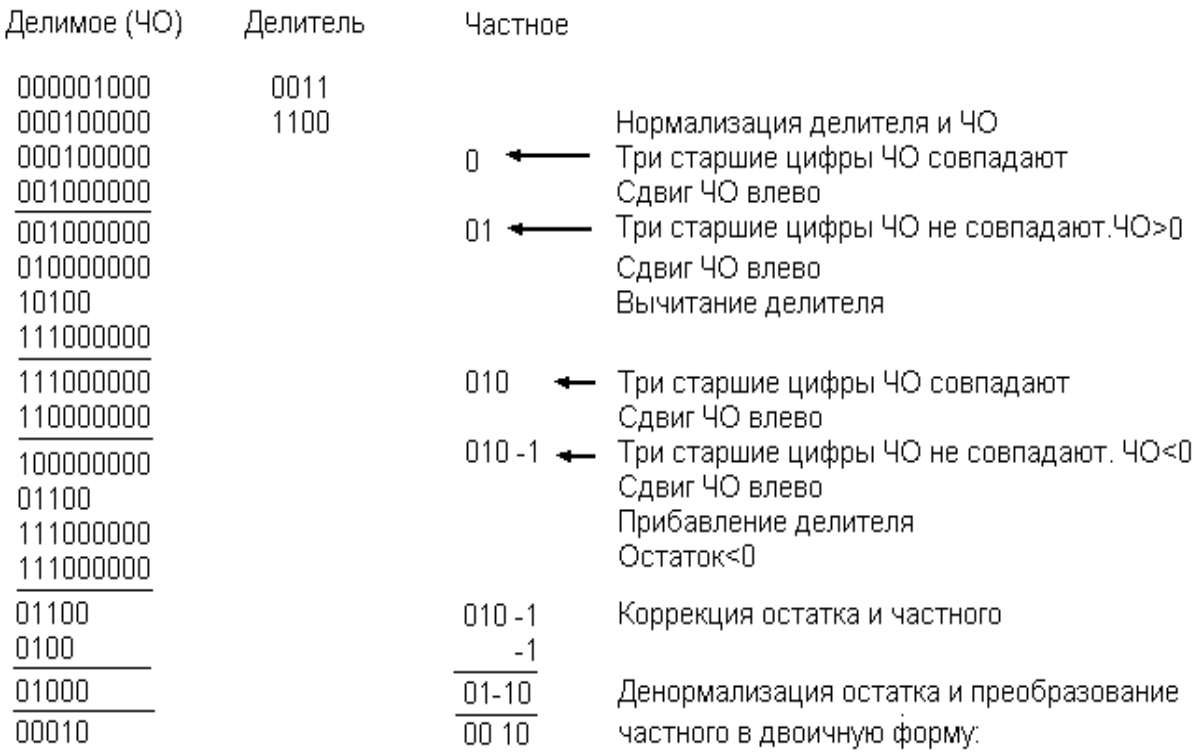


Рис. 3.14. Пример ускорения операции деления

На первом шаге для удаления предшествующих нулей делитель сдвигается на два разряда влево. Аналогично поступают и с ЧО, который вначале совпадает с делимым. Далее выполняется процедура, описанная выше в п. 2. Операция вычитания *D* обеспечивается прибавлением делителя с противоположным знаком. Поскольку по завершении операции остаток отрицателен, производится его коррекция путем прибавления *D*. Одновременно частное уменьшается на единицу

(эта операция показана в системе {-1, 0,1}, в которой представлено частное). Наконец, на последнем шаге форма представления частного меняется, и переходят к представлению в стандартной двоичной системе.

В стандартном алгоритме деления без восстановления остатка помимо сдвига в каждой итерации выполняется операция сложения или вычитания. В варианте SRT, в зависимости от кодов операндов в отдельных итерациях, достаточно только сдвига, что ускоряет процесс деления. Согласно статистическим данным, в среднем число сложений и вычитаний при использовании этого алгоритма сокращается в 2,67 раза.

В главе рассмотрели основные вопросы арифметики ЭВМ, различные варианты и способы организации операций сложения, вычитания, умножения и деления, применяемых в компьютерах в разных системах счисления. Способы выполнения арифметических операций подробно рассмотрены на примерах, приведенных в данной главе учебного пособия.

Контрольные вопросы

1.Чем обусловлена специфика целочисленного сложения и вычитания? Ответ поясните на примерах.

2.Какую роль играет в целочисленном сложении и вычитании дополнительный код? Ответ поясните на примерах.

3.К чему бы привел отказ от дополнительного кода при целочисленном сложении и вычитании? Ответ поясните на примерах.

4.Как выявляется переполнение при целочисленном сложении и вычитании?

5.Сформулируйте достоинства, недостатки и область применения четырех вариантов целочисленного «традиционного» умножения. Как учитываются знаки сомножителей?

6.Охарактеризуйте суть двух групп логических методов ускорения умножения.

7.Попарно сравните алгоритм Бута, модифицированный алгоритм Бута, алгоритм Лемана.

10. Разработайте алгоритм умножения с обработкой за шаг трех разрядов множителя.

11. Поясните суть аппаратных методов ускорения умножения, выделив три возможных подхода.

12. Сравните организацию целочисленного деления с восстановлением остатка и без восстановления остатка. Как учитываются при делении знаки операндов?

13. Обоснуйте возможность совмещения структур умножителя и делителя. Опишите объединенную структуру.

14. Сформулируйте четыре пути ускорения целочисленного деления. Сравните между собой их возможную реализацию.

15. Создайте структуру операционного блока для выполнения как сложения/вычитания, так и базового набора логических операций. Обоснуйте каждый элемент этой структуры.

16. Составьте таблицы умножения для чисел системы счисления с основанием 3.

17. Перемножьте двоичные числа 0111 и 0011.

18. Выполните следующие вычисления над 8-битными числами с использованием ДК:

00101101	11111111	00000000	11110111
+01101111	+11111111	-11111111	-11110111

19. Выполните те же вычисления с использованием ОК.

ЗАКЛЮЧЕНИЕ

В данном учебном пособии излагается основная информация, касающаяся практического применения ЭВМ, в виде базовых сведений о наиболее популярных в настоящее время программных пакетах Microsoft Office: Microsoft Word и Microsoft Excel, их основных возможностей по подготовке, редактированию и оформлению текстовой документации, графиков, диаграмм и рисунков; обработке числовых данных в электронных таблицах, позволяющих качественно оформлять отчетную документацию при обучении в университете

Рассмотренные в данном учебном пособии общие сведения о представлении информации в ЭВМ, такие как системы счисления, формы представления чисел в ЭВМ, преобразования чисел в системах счисления позволяют студентам разобраться в многообразии представлений информации в цифровой технике и научиться преобразовывать информацию, поступающую в ЭВМ. Также приведены примеры и пояснения по выполнению основных арифметических операций, таких как операции сложения, вычитания, умножения и деления в различных системах счисления. На их основе повышается качество изучения таких дисциплин как «Теория автоматов» и «Организация ЭВМ и систем», в частности, рассмотренная информация оказывает существенную помощь при проектировании процессора ЭВМ в курсовой работе по дисциплине «Организация ЭВМ и систем».

Данное пособие носит пояснительный характер и предназначено для ознакомления с базовыми основами функционирования ЭВМ.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Григорьев, В.Л. Самоучитель по операционной системе. М.: Энергоатомиздат. – 1992. 311 с.
2. Фигурнов, В.Э. IBM PC для пользователя [Текст] / В.Э. Фигурнов. Изд 6-е перераб. и доп. М.: ИНФРА-М. 1996. 287 с.
3. Жмакин, А.П., Бабкин Е.А. Текстовый редактор Word-97. [Текст] / А.П. Жмакин, Е.А. Бабкин; Курск. гос. тех. ун-т, 1999. 118 с.
4. Персон, Р. Microsoft Excel в подлиннике [Текст] Т.1 / Р. Персон. СПб.: BHV–С-Петербург. – 1997. 651 с.
5. Персон, Р. Microsoft Excel в подлиннике [Текст] Т.2. / Р. Персон. СПб.: BHV–С-Петербург. – 1997. 1272 с.
6. Острейковский, В.А. Информатика [Текст] / В.А. Острейковский. М.: Высшая школа. – 2000. 512 с.
7. Booth, A.D. A signal binary multiplication technique [Text] / A.D. Booth // Mech. Appl. Math. 1951. 4, path 2. P. 236-240.
8. Lehman, M. High-speed digital multiplication. [Text] / M. Lehman // IRE Transaction on electronic computers. 1957. V. EC-6–6, №3б.
9. Цилькер, Б.Я. Организация ЭВМ и систем [Текст]: учебник для вузов / Б.Я. Цилькер, С.А. Орлов. СПб.: Питер, 2004. 668 с.
10. Танэнбаум, Э. Архитектура компьютера [Текст] / Э. Таненбаум. 4-е изд. СПб.: Питер, 2003. 704 с.
11. Cocke J. High Speed Arithmetic in a Parallel Device [Text] / J. Cocke, D.W. Sweney // Technical Report IBM. 1957. feb.
12. Robertson, J.E. A new class of digital division methods [Text] / J.E. Robertson // IEEE Transactions on Computers, Electronics Computers, EC–7. 1958, Sep. p. 218-222.