Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

О.Г. Аврунін, О.В. Запорожець, Т.В. Носова, О.Г. Руденко, І.В. Руженцев,
В.В. Семенець, В.В Токарєв

МІКРОКОПРОЦЕСОРИ В ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНИХ СИСТЕМАХ

Харків 2015

УДК 621.391.1

Аврунін О.Г., О.В. Запорожець, Носова Т.В., Руденко О.Г., Руженцев І.В., Семенець В.В., Токарєв В.В. Мікропроцесори в інформаційно-вимірювальних систмах: Навч. посібник. – Харків: ХНУРЕ, 2015. – 185с.

ISBN 000-000-000-0

У даному навчальному посібнику розглянуто основні відомості про інформаційно-вимірювальні системи (ІВС), визначено місце ІВС в сучасній вимірювальні техніці та інформаційних технологіях, розглянута класифікація ІВС, принципи організації та основні структурні схеми ІВС, функції та переваги мікропроцесорних ІВС. Викладено аспекти теорії систем числення, алгоритми перетворення чисел з однієї системи числення в іншу, форми і способи подання чисел, алгоритми виконання арифметичних операцій в мікроконтролерних системах над числами з фіксованою і плаваючою крапкою. Розглянуто основи мікропроцесорних метрологічних систем на базі мікропроцесора i80x86 в реальному режимі, принципи зберігання масивів в пам'яті ПЕОМ, механізм базово-індексної адресації даних в МП i80x86.

Рекомендується студентам спеціальностей «Метрологія та інформаційно-вимірювальні технології», «Метрологія, стандартизація та сертифікація», «Комп'ютерна інженерія», «Біомедична інженерія» для студентів денної та заочної форми навчання.

Іл. 86. Табл. 61. Бібліогр. наймен. 31.

Рецензент:

С.І. Кондрашов, д-р техн. наук, проф., зав. каф. «Інформаційно-вимірювальних технологій і систем» НТУ «ХПІ».

ISBN 000-000-000-0

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Данное учебное пособие будет полезно студентам, обучающимся по направлениям: «Метрология и информационно-измерительные технологии», «Метрология, стандартизация и сертификация», «Компьютерная инженерия», «Биомедицинская инженерия» дневной и заочной формы обучения, а так же может быть использовано как электронный ресурс для дистанционного обучения.

В первом разделе учебного пособия рассмотрены основные сведения об информационно-измерительных системах (ИИС), определено место ИИС в современной измерительной технике и информационных технологиях, рассмотрена классификация ИИС, принципы организации и основные структурные схемы ИИС, функции и преимущества микропроцессорных ИИС.

Во втором разделе изложены аспекты теории систем счисления, алгоритмы перевода чисел из одной системы счисления в другую, формы и способы представления чисел, алгоритмы выполнения арифметических операций в микропроцессорных системах над числами с фиксированной и плавающей точкой.

В третьем разделе рассмотрены основы микропроцессорных метрологических систем на базе микропроцессора i80x86 в реальном режиме. Принципы хранения массивов в памяти ПЭВМ, механизм базово-индексной адресации данных в МП i80x86, применяемый в информационно-измерительных системах.

В современном обществе возникают новые научные и технические идеи, которые приводят к постановке актуальных измерительных задач. Для решения подобных задач необходимы высокотехнологичные микропроцессорные информационно-измерительные системы, которые значительно повысят качество измерений.

Говоря о развитии измерительной техники, необходимо, прежде всего, подчеркнуть качественные изменения средств измерений вследствие внедрения микропроцессоров и микропроцессорных систем, которые стали органической частью многих электронных измерительных приборов. Проникновение микропроцессоров в измерительную технику улучшило многие характеристики средств измерений, придало им новые свойства, открыло пути решения задач, которые ранее вообще не ставились. Применение микропроцессоров преобразует измерительные приборы в интеллектуальные устройства, способные проводить необходимую математическую обработку измерительной информации и представлять ее в наиболее удобном для восприятия виде.

С помощью микропроцессорных систем, встроенных в измерительные приборы достигаются многофункциональность приборов, упрощение управления измерительной процедурой, автоматизация регулировок, самокалибровка и автоматическая поверка, улучшение метрологических характеристик, выполнение вычислительных операций, статистическая обработка результатов наблюдений, создание программируемых, полностью

автоматизированных приборов. Трудно переоценить значение микропроцессоров для построения измерительно-вычислительных комплексов – автоматизированных средств измерений, предназначенных для исследования, контроля, испытания сложных объектов.

Широкое и всевозрастающее внедрение автоматизации практически во все сферы деятельности привело к коренной перестройке измерительной техники. Наряду с измерениями в автоматизированную систему входит также информационное обслуживание исследуемого (контролируемого) объекта, которое включает автоматический сбор, представление, доставку, запоминание, регистрацию, отображение, обработку и анализ информации, полученной в результате отдельных измерений. Зачастую приходится сталкиваться с целыми потоками измерительной информации. Поэтому основой современной измерительной техники является не отдельный, пусть даже автоматический прибор, а информационно-измерительная система, которая и решает поставленную задачу.

В случаях, когда измерительные приборы выполняются в виде отдельных устройств, не связанных с информационно-измерительной системой, микропроцессоры обеспечивают весь необходимый комплекс обработки информации. В случаях, когда прибор входит в качестве звена в информационно-измерительную систему, микропроцессор производит либо полную обработку информации, либо предварительную обработку данных, оставляя задачи полной обработки информации за вычислительной частью информационно-измерительной системы.

Кроме задач математической обработки измерительных параметров микропроцессор выполняет функции управляющего устройства, обеспечивающего подключение необходимых элементов приборов, прием командных сигналов, передачу выходных данных в широком кругу областей науки, техники и образования – от чисто измерительной техники до компьютерной, медицинской, электронной и т.д.

В учебном пособии каждый раздел содержит вопросы для самопроверки, которые помогут студенту определить степень освоения материала. В конце пособия приведены приложения с подробным программным кодом для лучшей иллюстрации решения стандартных задач, а также алфавитный указатель терминов для удобства ориентирования по тексту пособия.

1 ОСНОВНЫЕ СВЕДЕНИЯ ОБ ИНФОРМАЦИОННО-ИЗМЕРИТЕЛЬНЫХ СИСТЕМАХ

1.1 Место информационно-измерительных систем в современной измерительной технике и информационных технологиях

Измерения являются одним из основных источников количественной информации об исследуемых объектах самой различной природы. Измерительная техника развивалась и совершенствовалась на протяжении всей истории человечества, ее уровень определялся уровнем и потребностями производства, в свою очередь влияя на технологический уровень. По мере развития общественно-производственных отношений и научных исследований расширялся круг измеряемых физических величин. Если во времена античности существовала потребность в измерении всего нескольких величин (время, масса, длина, площадь, объем), то сейчас этот перечень насчитывает сотни наименований. Одновременно с расширением номенклатуры измеряемых величин на порядки возросли диапазоны измерений и уменьшились погрешности измерения. Кроме улучшения метрологических показателей средств измерений (СИ), существенно расширяются их функциональные возможности и повышаются эргономические свойства. Растет удельный вес автоматизированных СИ, увеличивается объем получаемой и обрабатываемой измерительной информации. Автоматизированные СИ встраиваются в системы автоматического управления различного уровня и становятся составными частями автоматизированного производства наряду с обрабатывающим и другим технологическим оборудованием.

В соответствии со сложившейся классификацией СИ подразделяются на следующие виды [1]:

1) меры – СИ, предназначенные для воспроизведения физической величины заданного размера с определенной точностью. Существуют однозначные, многозначные меры, а также наборы мер и магазины мер (гири, концевые меры длины, магазины сопротивлений, линейки, генераторы сигналов и т.д.);

2) измерительные преобразователи – СИ, предназначенные для выработки сигнала измерительной информации в форме, удобной для передачи, дальнейшего преобразования, обработки и хранения, но не поддающейся непосредственному восприятию наблюдателя. Примеры измерительных преобразователей: термопара, индукционный датчик, измерительный усилитель, аналого-цифровой преобразователь;

3) измерительные приборы – СИ, предназначенные для выработки сигнала измерительной информации в форме, удобной для восприятия наблюдателем. Измерительный прибор оснащен отсчетными устройствами (шкала со стрелкой, цифровой индикатор и т.п.), с помощью которых человек-оператор непосредственно считывает результат измерения. К этой категории

СИ относятся вольтметры, омметры, частотомеры, манометры, термометры и др.;

4) измерительные установки – совокупность функционально объединенных СИ (мер, преобразователей, приборов) и вспомогательных устройств, предназначенная для выработки сигналов измерительной информации в форме, удобной для восприятия наблюдателем, и расположенная в одном месте. Установка для косвенного измерения сопротивления на основе закона Ома будет состоять из амперметра, вольтметра, источника питания и соединительных проводников;

5) измерительные информационные системы – совокупность СИ и вспомогательных устройств, соединенных между собой каналами связи, предназначенная для выработки сигналов измерительной информации в форме, удобной для автоматической обработки, передачи и использования в автоматических системах управления.

Примерно до середины XX века измерительная аппаратура выдавала результаты измерений, воспринимаемые только органами чувств человека, причем процесс измерения не был автоматизирован. Это нашло свое отражение и в функциях различных видов СИ. Измерительные преобразователи, предназначенные для использования в неавтоматизированных СИ, обычно преобразуют измеряемую величину в угловое или линейное перемещение, например электромеханические преобразователи в электроизмерительных приборах, пружинные преобразователи в динамометрах. С развитием электрических методов измерения неэлектрических величин стали активно использоваться измерительные преобразователи, преобразующие различные физические величины в электрические величины, более удобные для дальнейшей автоматизированной обработки.

Общим для всех вышеупомянутых неавтоматизированных СИ является малый объем воспринимаемой, обрабатываемой и отображаемой измерительной информации. Эти СИ практически не могут хранить измерительную информацию. С появлением электронных цифровых измерительных приборов появилась возможность долговременно хранить в памяти результаты нескольких измерений.

Параллельно с развитием измерительной техники шло интенсивное развитие других важнейших составляющих современного технического прогресса – информационных технологий, являющихся основой автоматизации управления и производства. Информационная технология – совокупность методов, производственных и программно-технологических средств, объединенных в технологическую цепочку, обеспечивающую сбор, хранение, обработку, вывод и представление информации. Информационные технологии предназначены для снижения трудоемкости процессов использования информационных ресурсов. Они включают в себя широкий класс дисциплин и областей деятельности, относящихся к технологиям управления и обработки данных, в первую очередь числовых, с применением вычислительной техники. Таким образом, информационные технологии – это прежде всего компьютерные технологии, основной технической базой которых является

вычислительная техника. Другим техническим компонентом информационных технологий являются системы и каналы связи, обеспечивающие быструю и достоверную передачу информации на большие расстояния. Важную роль в применении информационных технологий для решения разнообразных технических задач играет программно-математическое обеспечение, включающее в себя алгоритмы преобразования информации и программы для реализации этих алгоритмов.

Совместное применение измерительной техники и методов информационных технологий в одних и тех же областях не могло не привести к их взаимопроникновению. Потребности современного производства и научных исследований все чаще ставят перед измерительной техникой задачи автоматической регистрации, хранения и математической обработки больших массивов измерительной информации, передачи ее на расстояние, использование для автоматического управления различными технологическими процессами. Эти проблемы, аналогичные проблемам информационных технологий, оказали существенное влияние на организацию процесса измерений, что привело к появлению измерительных информационных технологий.

Возможность и необходимость решения принципиально новых задач требуют использования соответствующих математических методов. Поэтому теоретической базой информационных измерительных технологий наряду с классической метрологией являются теория вероятностей, математическая статистика, интервальный анализ, нечеткая логика и ряд других наук. Технической базой измерительных информационных технологий являются автоматизированные СИ. Наиболее перспективными и интенсивно развивающимися автоматизированными средствами измерения являются информационные измерительные системы (ИИС), которые отличаются от традиционных средств измерения тремя принципиальными моментами:

1) большие объемы измерительной информации, подлежащие сбору, обработке и хранению;

2) автоматизация процессов сбора и обработки измерительной информации;

3) возможность изменения и наращивания решаемых измерительных задач, что придает ИИС существенную гибкость.

Возможность хранения значительных объемов измерительной информации также существенно расширяет возможности пользователя. При ручном сборе результатов измерения их регистрация и обработка также могли производиться только вручную, что затрудняло обработку больших массивов данных и применение сложных алгоритмов. Ручной ввод результатов в ЭВМ мог преодолеть эти трудности для лабораторных исследований, но явно не подходил для производственных условий. ИИС может хранить сотни и даже тысячи результатов измерений, и на этом массиве экспериментальных данных можно ставить новые задачи более высокого уровня.

Гибкость ИИС позволяет существенно уменьшить номенклатуру СИ, используемых для исследований в определенной области. Однако более важной

является возможность быстрой перестройки имеющейся ИИС для решения новой измерительной задачи, что практически недоступно для других видов СИ.

Перечисленные выше возможности ИИС привели к тому, что они широко используются в самых различных отраслях производства и научных исследований. Этому способствует прежде всего резкое возрастание возможностей и снижение стоимости средств вычислительной техники.

1.2 Классификация ИИС

Классификация ИИС производится в соответствии с различными классификационными признаками, отражающими область применения, функции и конструкцию ИИС [2]:

- функциональное назначение;
- вид и характер входных величин;
- вид выходной информации;
- вид структурно-функциональной схемы ИИС;
- принцип построения.

Первый классификационный признак представляется наиболее важным, так как он в первую очередь интересует потребителя (пользователя) ИИС. Этот признак не зависит от технических средств реализации измерительной системы. По функциональному назначению ИИС делятся на:

1) измерительные ИИС, которые выполняют прямые, косвенные, совместные и совокупные измерения с соответствующей математической обработкой и выдачей числового значения физической величины (телеизмерительные ИИС, если объект измерений находится на очень большом расстоянии);

2) ИИС автоматического контроля, предназначенные для установления соответствия между состоянием (свойством) объекта контроля и заданной нормой, которая определяет качественно разные области его состояния. ИИС выдает информацию о состоянии объекта контроля и об отклонении от заданной нормы;

3) ИИС технической диагностики, собирающие информацию о неисправностях и повреждении какой-либо системы, на основании которой решается задача отыскания места повреждений и установление причин этих повреждений и неисправностей, выявление дефектных элементов и восстановление нормальной работы объекта;

4) ИИС идентификации (распознавания образов), предназначенные для установления соответствия между объектом и заданным образом. При этом образ может быть задан в виде образцового изделия или в виде перечня определенных свойств и значений параметров с указанием полей допуска.

Если проанализировать функциональную структуру любой ИИС (рис. 1.1), то можно выделить следующие функциональные блоки:

– устройство измерения;
– устройство обработки информации;
– устройство хранения информации;
– устройство представления информации;
– устройство управления;
– устройство воздействия на объект.

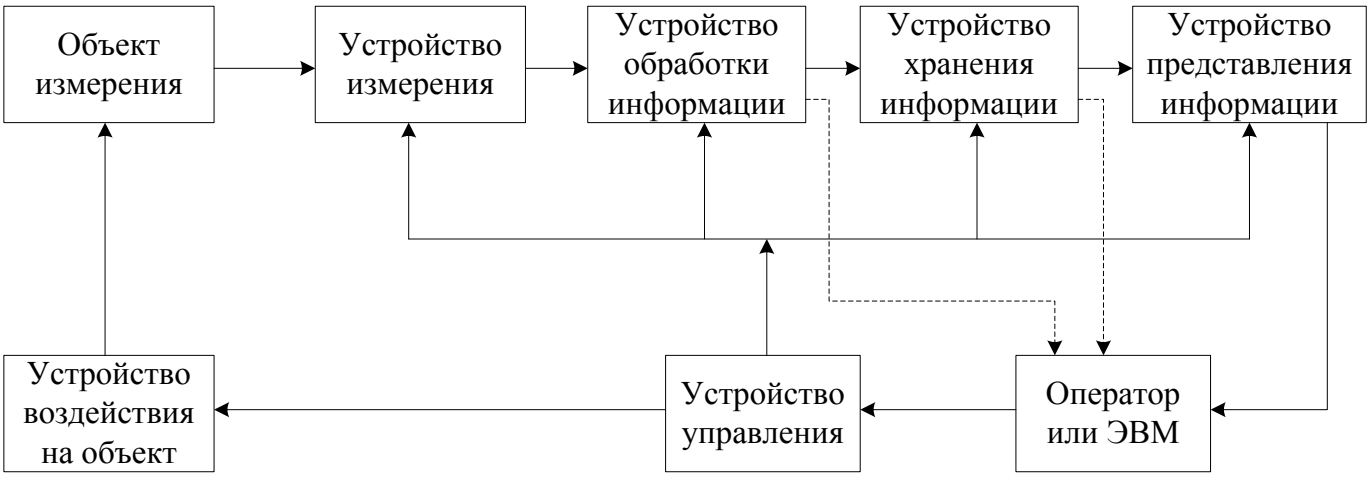


Рисунок 1.1 – Обобщенная структурная схема ИИС

Наиболее перспективным методом проектирования ИИС на сегодняшний день есть принцип агрегатно-модульного построения разных систем из сравнительно ограниченного набора унифицированных узлов, которые выпускаются промышленностью.

Агрегатно-модульный принцип построения ИИС предусматривает применение стандартных интерфейсов, под которыми понимают как совокупность правил, протоколов и программного обеспечения процесса обмена информацией, так и технические средства соединения модулей в системе.

В зависимости от способа организации передачи информации между функциональными узлами, которые являются приемниками и передатчиками информации, различают три типа структуры:

1) последовательную (цепочечную);
2) радиальную;
3) магистральную.

Последовательная (цепочечная) структура представляет собой последовательное или каскадное соединение функциональных узлов (ФУ), осуществляющих преобразование измерительного сигнала от первичного измерительного преобразователя (датчика) до выдачи результата измерения. Этот тип структуры ИИС представлен на рис. 1.2.

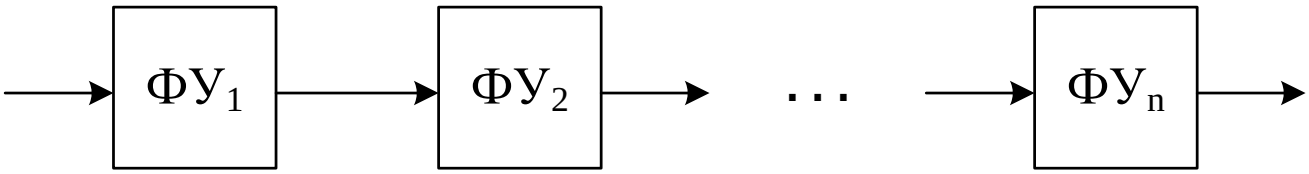


Рисунок 1.2 – Последовательная структура ИИС

К достоинствам последовательной структуры можно отнести простоту, а к недостаткам – низкую надежность и ограниченные функциональные возможности. Действительно, отказ любого функционального узла приведет к разрыву измерительного канала и, как следствие, к неработоспособности всей измерительной системы в целом.

Радиальная структура ИИС представлена на рис. 1.3. Центральным устройством такой системы является контроллер, который осуществляет функции управления отдельными функциональными узлами и распределяет потоки информации между ними. Отказ одного из функциональных блоков в радиальной структуре в общем случае не приводит к отказу всей системы, речь может идти только о частичной потере функциональности ИИС.

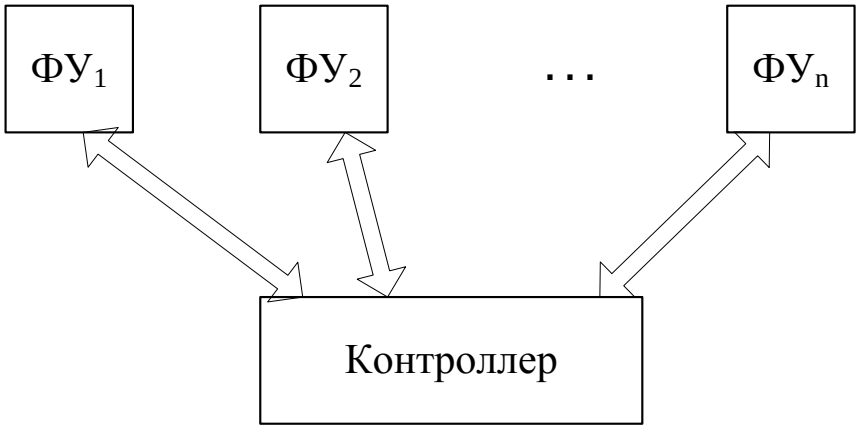


Рисунок 1.3 – Радиальная структура ИИС

Достоинствами радиальной структуры являются повышенная надежность и более широкие функциональные возможности в сравнении с последовательной структурой, а недостатками – ограниченное количество подключаемых функциональных узлов (определяется количеством портов ввода-вывода контроллера) и сложность контроллера.

Магистральная структура (другое ее название – общая шина) изображена на рис. 1.4. Главное преимущество магистральной структуры – возможность легко изменять конфигурацию системы, подключая либо отключая отдельные функциональные узлы. Максимальное количество подключаемых устройств будет определяться нагрузочной способностью магистрали. Надежностные характеристики такой структуры тоже определяются магистралью.

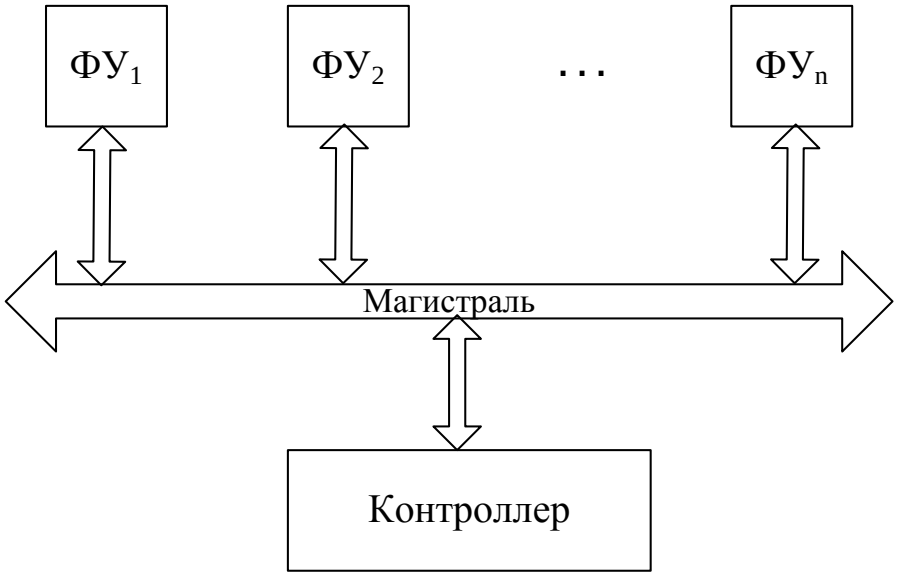


Рисунок 1.4 – Магистральная структура ИИС

Недостатки магистральной структуры вытекают из необходимости использования разделяемого ресурса – общей шины. Пропускная способность магистрали делится на количество подключаемых функциональных узлов. Следовательно, чем больше устройств будет подключено к магистрали, тем медленнее будет работать система в целом.

Уместно будет упомянуть и еще одну классификацию ИИС. В соответствии с организацией алгоритма функционирования различают:

1) системы с жестким заранее заданным алгоритмом функционирования. В системах этого типа, называемых еще системами с жесткой логикой, алгоритм работы ИИС не изменяется, поэтому такие системы применяются в основном для исследования объектов, которые работают в определенном установившемся режиме;

2) программированные системы. В программированных системах алгоритм работы изменяется согласно заранее заданной программе, которая составляется в зависимости от условий функционирования объекта исследования. Это позволяет значительно увеличить гибкость системы за счет возможности обеспечивать быструю переналадку с одних измерительных задач на другие программным способом. Достаточно изменить управляющую программу такой системы и конфигурацию коммутирующих соединений, чтобы полностью поменять функциональность измерительной системы;

3) адаптивные системы. В адаптивных системах алгоритм работы, а иногда и структура ИИС изменяются, приспосабливаясь к изменениям измеренных величин и условий работы объекта исследования. При построении адаптивной ИИС требуется меньшее количество априорной информации, которая имеет большое значение при исследовании новых объектов, характеристики которых еще мало известны.

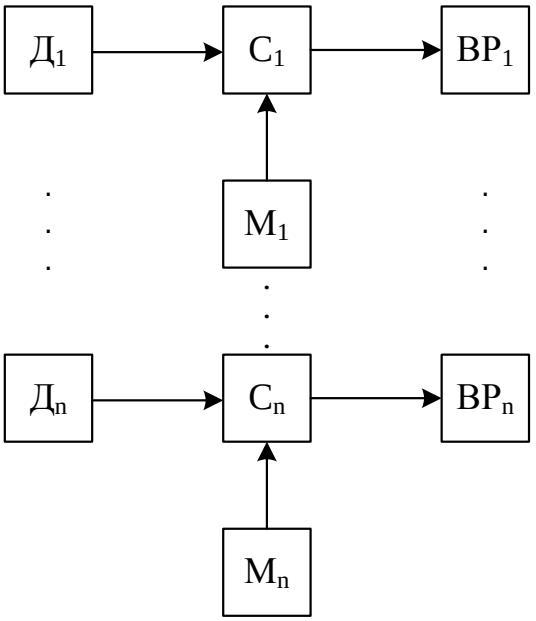
1.3 Принципы организации и основные структурные схемы ИИС

1.3.1 Измерительные системы

Измерительные ИИС характеризуются преимуществом функций измерения, функции обработки и хранения незначительные или отсутствуют. В зависимости от вида и числа различных элементов в структуре измерительных систем их делят на следующие категории:

1) многоканальные (системы с параллельной структурой);
2) сканирующие (системы с последовательной структурой);
3) мультиплицированные (системы с развертывающим уравновешиванием);
4) многоточечные.

Упрощенная структура многоканальной измерительной системы представлена на рис. 1.5. В каждом измерительном канале есть полный набор основных функциональных элементов: датчик Д (первичный измерительный преобразователь), мера М, элемент сравнения С (компаратор) и блок выдачи результата измерения ВР. Преимуществами многоканальной измерительной системы являются высокая надежность и быстродействие, обусловленные параллельной структурой. Вместе с тем такие системы характеризуются аппаратной избыточностью, следствием которой являются повышенная сложность и стоимость.

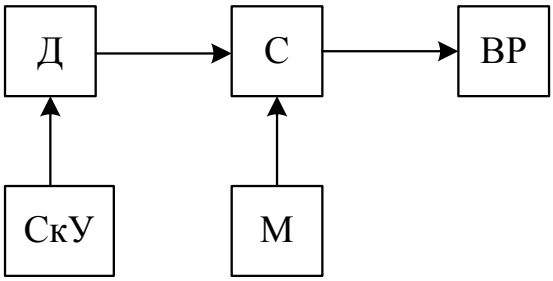


Д – датчик, первичный измерительный преобразователь,
С – элемент сравнения,
М – мера,
ВР – выдача результата

Рисунок 1.5 – Многоканальная измерительная система

Сканирующая измерительная система изображена на рис. 1.6. Она имеет один измерительный канал, а отличительной особенностью является наличие

сканирующего устройства СкУ, осуществляющего перемещение датчика в различные точки объекта измерения.



СкУ – сканирующее устройство

Рисунок 1.6 – Сканирующая измерительная система

Эти системы последовательно во времени выполняют измерение многих величин с помощью одного канала измерения. Сканирующее устройство перемещает датчик в пространстве, причем траектория движения может быть заранее запрограммирована (так называемое пассивное сканирование) или может изменяться в зависимости от полученной в процессе сканирования информации (активное сканирование).

При исследованиях параметрических полей (температур, давлений, механических напряжений, магнитных и др.) такие ИИС дают количественную оценку значений параметров полей в заданных точках. Иногда с помощью сканирующих измерительных систем определяют экстремальные значения параметров исследуемых полей или эквипотенциальные области.

К недостаткам сканирующих измерительных систем можно отнести относительно малое быстродействие из-за последовательного выполнения операций измерения для всех измеряемых величин.

Структура мультиплицированной измерительной системы, которую еще называют системой с развертывающим уравновешиванием, представлена на рис. 1.7. Она содержит несколько измерительных каналов, но в отличие от многоканальной структуры мера является общей для всех каналов. Эти системы разрешают на протяжении цикла изменения известной величины (развертки) выполнить сравнение со всеми измеряемыми величинами. Как правило, в таких системах измеренная величина x сравнивается с линейно изменяющейся величиной x_k. В момент равенства x и x_k формируется интервал времени T_x, пропорциональный измеряемой величине.

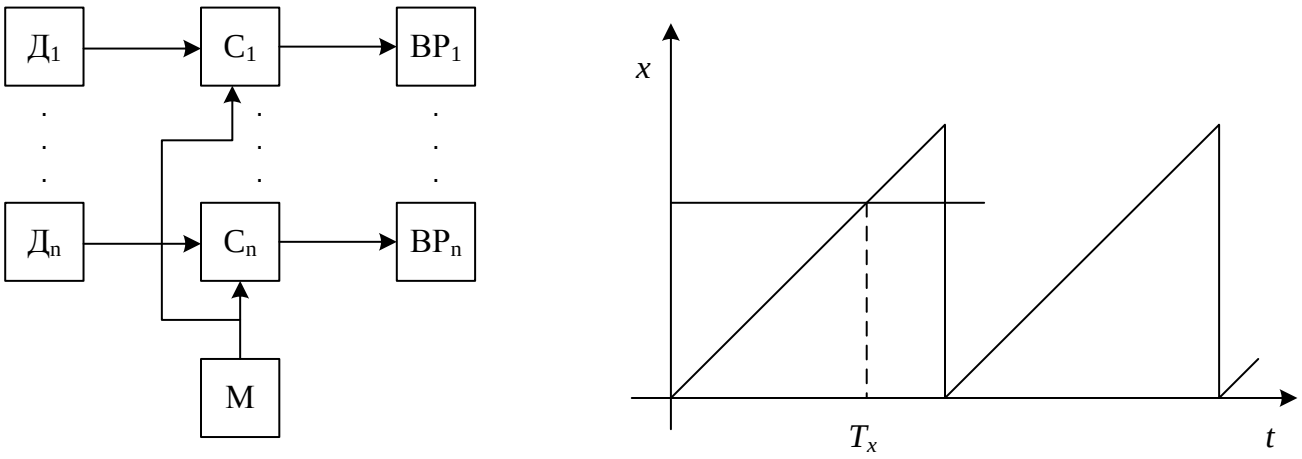
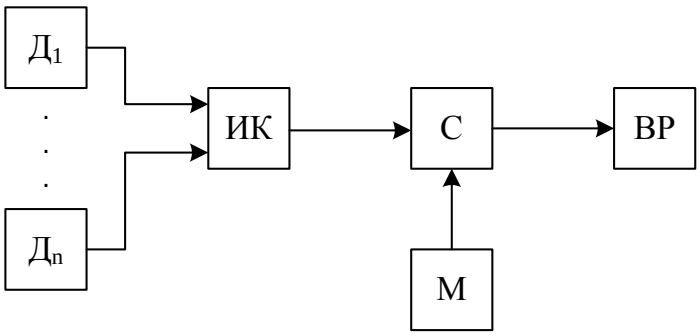


Рисунок 1.7 – Мультиплицированная измерительная система

Мультиплицированные измерительные системы имеют меньшее число элементов по сравнению с многоканальными системами, а следовательно и меньшую стоимость при обеспечении такого же быстродействия. Вместе с тем они содержат большое количество элементов сравнения, которые значительно усложняются при измерении сигналов низкого уровня.

Многоточечные измерительные системы (рис. 1.8) применяются для исследования сложных объектов с большим числом измеряемых величин. Число измерительных каналов в таких системах может достигать нескольких тысяч за счет использования измерительных коммутаторов ИК, которые предназначены для последовательного подключения большого количества датчиков ко входу одного измерительного канала.



ИК - измерительный коммутатор

Рисунок 1.8 – Многоточечная измерительная система

Достоинства таких систем очевидны: меньшее количество оборудования в сравнении с многоканальными системами, возможность наращивания числа измерительных каналов за счет коммутатора. Недостатками многоточечных измерительных систем являются пониженное быстродействие при большом количестве опрашиваемых датчиков и некоторое снижение точности за счет остаточных параметров ключей коммутации.

1.3.2 Телеизмерительные системы

Отдельной категорией ИИС являются телеизмерительные системы. Применяются они в тех случаях, когда необходимо выполнять измерения на больших расстояниях от объекта: измерение параметров движущихся объектов, измерение параметров объектов, рассредоточенных по площади (большие промышленные предприятия, газо- и нефтепроводы), измерение параметров объектов, непосредственное нахождение человека возле которых невозможно из-за требований безопасности (объекты атомной энергетики).

Отличием телеизмерительных систем является наличие в них канала связи. В качестве среды передачи могут использоваться проводные линии, радиолинии и оптические линии связи. Для передачи информации от нескольких источников по одной линии связи применяют разные принципы разделения каналов. Чаще всего используют временное и частотное разделение каналов.

В зависимости от информативного параметра сигнала, которым передается значения измеряемой величины по линии связи, телеизмерительные системы бывают:

1) токовые;
2) частотные,
3) время-импульсные;
4) цифровые.

В токовых телеизмерительных системах размер измеряемой величины передается по проводным линиям связи сигналом постоянного тока (0...5 мА). Дальность действия токовых телеизмерительных систем составляет 7-10 км для воздушных линий, 20-25 км для кабельных каналов.

В частотных телеизмерительных системах размер измеряемой величины передается по линиям связи частотой синусоидального или импульсного сигнала, может использоваться проводная и беспроводная связь. Недостатками таких систем являются перекрестные наводки и помехи. Дальность действия может достигать сотен километров.

Во время-импульсных телеизмерительных системах размер измеряемой величины передается по линиям связи длительностью импульсов или интервалами между импульсами. Дальность действия – сотни и тысячи километров.

В цифровых телеизмерительных системах размер измеряемой величины передается по линиям связи двоичной кодовой комбинацией в виде комбинации импульсов. Достоинствами цифровых телеизмерительных систем являются высокие метрологические характеристики, возможность работы с различными каналами связи, высокая помехозащищенность и возможность вывода информации в ЭВМ.

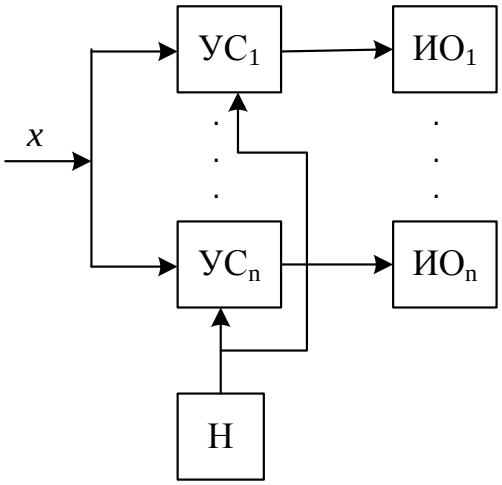
1.3.3 Системы автоматического контроля

Современные системы автоматического контроля (САК) делят на системы, в которых осуществляется непрерывный контроль параметров объекта (рис. 1.9), и системы с дискретным последовательным контролем этих параметров (рис. 1.10).

Устройство воспроизведения и хранения норм может быть общим для нескольких каналов или индивидуальным.

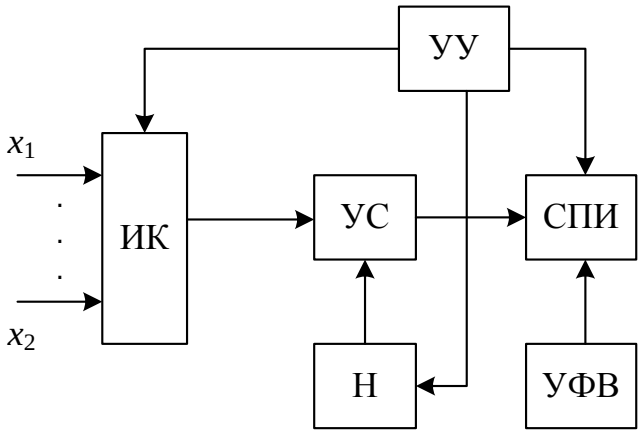
Системы с непрерывным контролем требуют большого количества оборудования и потому применяются только для контроля наиболее ответственных параметров.

Недостатки систем с дискретным контролем: большая избыточность операций контроля, возможность пропуска аварийных ситуаций.



УС – устройство сравнения,
ИО – индикация отклонений,
Н – набор норм

Рисунок 1.9 – Структурная схема одного канала САК с непрерывным контролем



УУ – устройство управления
СПИ – средства представления информации
УФВ – устройство формирования сигналов времени

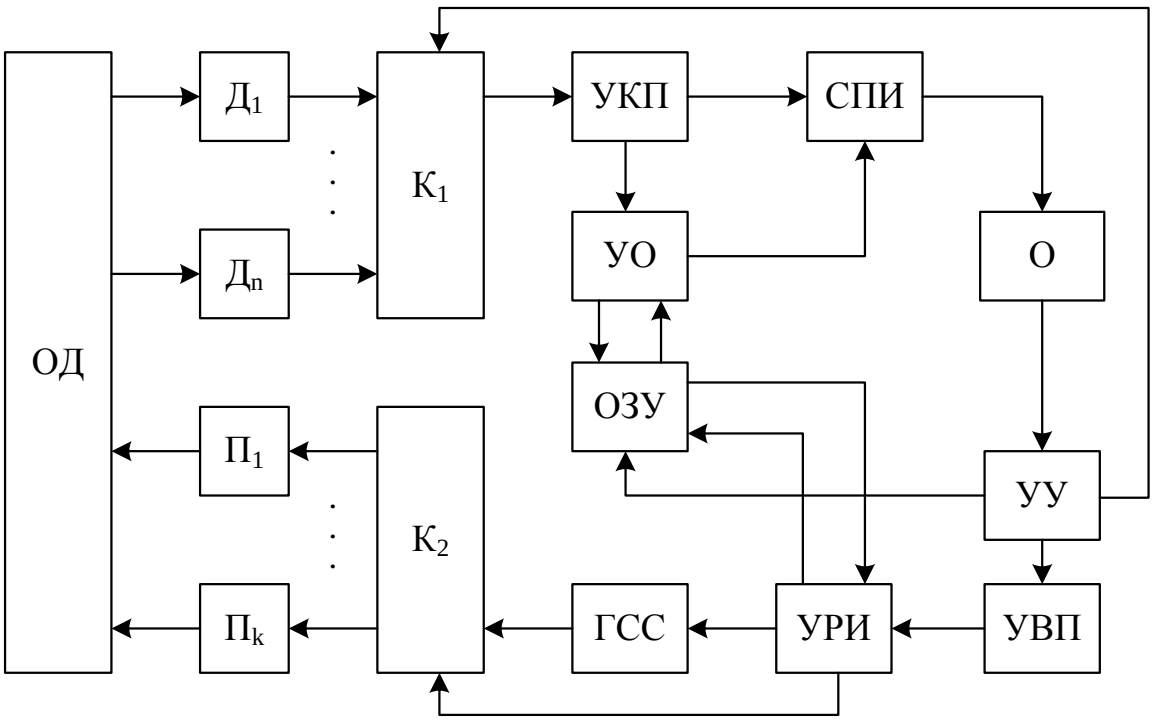
Рисунок 1.10 – Структурная схема САК с дискретным контролем

1.3.4 Системы технической диагностики

Системы технической диагностики разделяют на собственное диагностические и прогнозирующие.

По характеру выработки оценки состояния объекта диагностики системы технической диагностики разделяют на статистические и детерминированные. При статистической оценке состояния объекта решение принимается на основании измерений или проверок сигналов, которые характеризуют объект, а при детерминированной – параметры объекта, который проверяется, сравнивают с параметрами объекта, принятого за образцовый.

Существуют такие виды проверок: функциональная, алгоритмическая и логически-комбинационная. При функциональной проверке обнаруживают наличие сигнала на выходе объекта при поступлении сигнала на его вход; отсутствие выходного сигнала является отказом. При алгоритмической проверке согласно алгоритму работы объекта проверяется последовательность выполнения функций. Логически-комбинационная проверка разрешает обнаруживать неисправности на любом уровне. На вход объекта, который проверяется, в этом случае подают специальный диагностический тест, специальные стимулирующие сигналы.



ОД – объект диагностики, Д – датчик, К – коммутатор, П – преобразователь, УКП – устройство контроля параметров, УО – устройство обработки, ОЗУ – оперативное запоминающее устройство, УУ – управляющее устройство, УВП – устройство ввода программы, УРИ – устройство распределения информации, ГСС – генератор стимулирующих сигналов, СПИ – средства представления информации, О – оператор

Рисунок 1.11 – Структурная схема системы технической диагностики

1.4 Функции микропроцессоров в ИИС

Основное отличие измерительных систем, использующих микропроцессоры, от аналоговых измерительных средств заключается в том, что в таких системах часть измерительной процедуры выполняется в числовой форме с помощью программированной вычислительной мощности, которая вводится в измерительную цепь.

Изучая принципы построения и особенности функционирования измерительных информационных систем с микропроцессорами, нужно обратить внимание на следующее: эти системы состоят из двух нераздельно связанных частей – аппаратной и программной; аппаратная часть содержит в себе измерительные, вычислительные и вспомогательные средства, программная – системное, инструментальное и прикладное программное обеспечения. Функционирование микропроцессорных измерительных ситем опирается на принятую совокупность интерфейсов, которые обеспечивают взаимодействие всех компонентов.

Введение микропроцессора в состав измерительной цепи и соответственно числовых измерительных преобразований в измерительную процедуру радикально изменяет функциональные и метрологические возможности средств измерений. Появляется возможность выполнять сложные косвенные, совокупные и совместные измерения. Изменяются принципы реализации статистических измерений. Расширяются возможности по коррекции погрешностей и применению адаптивных и итеративных измерительных процедур.

К типичным измерительным преобразованиям, которые реализуются в числовой форме, принадлежат: масштабирование, функциональные преобразования (включая усреднение), формирование корректирующих влияний, обеспечение адаптации и итеративных измерений.

Естественно, что при этом изменяется структура полной погрешности – появляются составляющие, обусловленные числовыми измерительными преобразованиями, и возникают проблемы метрологического анализа в связи с усложнением алгоритмов измерений.

Для процессорных преобразований характерное изменение (уменьшение) удельного веса инструментальных погрешностей, которые определяются сбоями в функционировании микропроцессора.

Активное участие микропроцессора в реализации измерительного алгоритма приводит к необходимости более строго, с позиции метрологии, подходить к погрешностям измерений, которые становятся составной частью общей погрешности результата измерения. Это особенно важно для тех случаев, когда используются сравнительно малоразрядные микропроцессоры. К измерительным средствам, которые используются в автономных электроизмерительных средствах, выдвигаются требования минимальных массы и габаритов, которые также ограничивают разрядность процессоров.

С другой стороны, нет необходимости существенным образом завышать требования к точности измерений сравнительно с той погрешностью, которую вносят первичные измерительные преобразователи. Относительно рациональности применения микропроцессора, нужно стремиться к тому, чтобы погрешность вычислений незначительно увеличивала общую погрешность измерений.

Можно построить следующую классификацию функций, которые выполняются микропроцессором в измерительных системах [3, 4]:

1) контроллерные функции, связанные с управлением со стороны микропроцессора элементами аппаратного обеспечения;

2) вычислительные функции, связанные с обработкой и анализом данных;

3) тестовые функции, связанные с определением работоспособности измерительной системы, диагностикой и локализацией неисправности;

4) сервисные функции, связанные со взаимодействием оператора и информационно-измерительной системы.

К контроллерным относятся функции непосредственного управления элементами аппаратного обеспечения измерительной системы, в частности функции программ-драйверов. По типу аппаратного обеспечения контроллерные функции можно в свою очередь разбить на ряд подклассов:

а) управление измерительной цепью. Сюда входят управления переключателями каналов и диапазонов, подключение образцовых мер в процессе калибровки, управление измерительными усилителями. Как правило, эти функции выполняют чисто программными методами, иногда с использованием таймера – с помощью микропроцессора и портов ввода-вывода;

б) управление аналого-цифровым преобразованием. Эта функция может выполняться как чисто программно – микропроцессором, так и специализированным аппаратным обеспечением, в частности однокристальными аналого-цифровыми преобразователями (АЦП) и таймером, или комбинированными аппаратно-программными методами;

в) управление средствами общения с оператором. Сюда входит большая группа функций управления клавиатурой, индикаторами, звуковой сигнализацией и дисплеем;

г) управление регистраторами. Сюда относятся функции управления печатающими устройствами, самописцами, внешними накопителями, накопителями на магнитной ленте в режиме двустороннего обмена, управление накопителями на магнитных дисках и дополнительными внешними модулями памяти.

К вычислительным относятся функции предварительной и основной обработки данных. Для осуществления предварительной обработки сигналов применяются такие процедуры:

а) калибровка, линеаризация, нормализация. Эти виды предварительной обработки связаны с коррекцией погрешностей, внесенных первичными измерительными преобразователями и усилительным трактом;

б) масштабирование. Этот вид предварительной обработки служит для учета коэффициента передачи и напряжения сдвига узлов в приемоусилительном тракте;

в) цифровая фильтрация. Использование разнообразных алгоритмов цифровой фильтрации измерительных сигналов дает значительно более широкие возможности, чем аналоговая фильтрация;

г) сжатие данных. Для экономии памяти применяется сжатие данных без потери информации;

д) распознавание и устранение артефактов. Сущность процедуры распознавания артефактов – это классификация входного сигнала на два класса: “подлежит анализу” и “не подлежит анализу”. Не подлежащий анализу сигнал считается “артефактом”, имеется в виду, что он обязан своим происхождением факторам, которые не являются предметом исследования: случайным нарушением измерительной цепи, побочным влиянием на первичные преобразователи, наводками от внешних цепей и т.п.

Для основной обработки сигнала применяются такие процедуры:

а) статистическая обработка. Сюда входит ряд процедур для вычисления функции и плотности распределения, математического ожидания, дисперсии, среднего квадратичного отклонения, вариации, асимметрии и т.п.;

б) вычисление вторичных параметров по формулам. В частности, сюда относится реализация процедур различных косвенных измерений;

в) амплитудно-временной и контурный анализ. К этим процедурам относятся определение параметров экстремальных точек, длительности фронта и среза импульса, выделение характерных точек, точек пересечения заданного порога и т.п.

Самотестирование цифровых измерительных устройств и систем – характерная особенность современных устройств со встроенными микропроцессорами. Можно разделить все типы тестирования устройств со встроенными микропроцессорами на два основных класса: тестирование с применением внешних микропроцессорных средств и полностью автономное тестирование.

В меру усложнения измерительного прибора или системы все чаще применяется режим интерактивной обработки – многостадийного диалога оператора с микропроцессорным устройством. При этом весь процесс измерения и обработки разбивается на ряд последовательных процедур. В качестве сервисных функций можно привести следующие:

а) ввод задачи;

б) сбор и предыдущая обработка первичной информации;

в) основная обработка и интерпретация результатов;

г) ввод результатов исследования для интерпретации, документирования и архивации.

1.5 Преимущества микропроцессорных измерительных систем

К преимуществам средств измерения со встроенными микропроцессорами можно отнести:

1) многофункциональность. До применения микропроцессоров многофункциональные измерительные системы представляли собой совокупность нескольких функциональных узлов, объединенных в одно конструктивное целое. При эксплуатации таких систем переход от одной функции к другой осуществляется с помощью коммутирующих устройств, а сами эти системы выполнены по схеме с жесткой логикой. Введение микропроцессорной системы в ИИС позволило превратить ее в программно-управляемое устройство. Функциональные возможности такой системы определяются выполняемой программой и могут быть легко видоизменены путем перехода к другой программе, хранимой в ПЗУ. Это обеспечивает гибкость и разрешает наращивать функции при модернизации измерительной системы без существенных изменений в ее схеме;

2) повышение точности измерений. Повышение точности достигается за счет автоматической компенсации систематической погрешности, в частности, автоматической установки нуля перед началом измерений, автоматического выполнения операции градуировки (самокалибровка), выполнение самоконтроля, уменьшение влияния случайных погрешностей путем проведения многократных измерений с последующим усреднением результатов, выявление и исключение грубых погрешностей;

3) расширение измерительных возможностей системы. Применение микропроцессоров позволяет существенно расширить возможности измерений широкого перечня параметров сигналов и характеристик устройств. Это связано прежде всего с использованием косвенных и совокупных измерений.

Существуют определенные сложности при выполнении косвенных измерений: снимаются показания разных приборов, нужно внимание экспериментатора, проведение вычислений, сложная процедура оценки погрешностей. Микропроцессорная система разрешает автоматизировать все эти процессы и экспериментатор сразу получает результат измерения;

4) упрощение и облегчение управления системой. Одним из критериев высокого уровня программного обеспечения измерительного прибора является степень сложности его передней панели. Для современных микропроцессорных приборов и систем характерная кнопочная система управления, конструктивно выполняемая в виде клавиатуры. Радикально уменьшает число органов управления автоматизация выбора границ измерения, интервала дискретизации напряжения исследуемого сигнала и других режимов работы;

5) возможность получения математических функций измеренных значений, например:

- умножение измеренного значения на константу;
- получение отклонений результата измерения от номинального значения (абсолютное и относительное (в %) отклонение);
- сдвиг, который предполагает вычитание константы из результата измерения;

- вычисление отношений: деление на константу, нахождение частного от деления;
- представление результата измерения в логарифмических единицах;
- линеаризация зависимостей;

6) получение статистических характеристик. Измерительные системы, в составе которых есть микропроцессорная система, разрешают формировать оценки таких вероятностных характеристик анализируемой случайной величины, как среднее значение, средняя мощность, среднеквадратичное значение, дисперсия, стандартное отклонение, коэффициент корреляции;

7) миниатюризация и экономичность аппаратуры. Резкое уменьшение числа компонентов в схеме измерительной системы вследствие выполнения ряда функций микропроцессорной системой, их относительно невысокая стоимость, значительное снижение потребляемой мощности разрешают строить малогабаритные и эргономичные устройства;

8) повышение надежности. Оно обусловлено уменьшением числа элементов схем, осуществлением самодиагностики, применением узлов с некалиброванными характеристиками, возможностью выполнения коррекции погрешностей, которая повышает метрологическую надежность;

9) сокращение продолжительности разработки. Часто для получения новых свойств измерительной системы, выполненной с использованием микропроцессоров, не нужно больших изменений в схеме или конструкции прибора. Для широко применяемых микропроцессоров уже разработаны библиотеки типичных прикладных программ, из которых можно выбрать необходимые компоненты;

10)возможность взаимодействия с другими измерительными системами. Одной из функций, выполняемой микропроцессором в измерительной системе, является поддержка одного или нескольких стандартных интерфейсов. Это дает возможность ИИС взаимодействовать с ЭВМ и другими измерительными системами.

11)На основании вышесказанного можно сделать вывод, что применение микропроцессора вместо традиционного схемного решения в рамках жесткой логики разрешает:

- сделать измерительную систему многофункциональной, функционально-гибкой;
- развивать и наращивать в дальнейшем измерительную систему;
- измерительной системе взаимодействовать с большим количеством входных и выходных устройств;
- запоминать и хранить большие объемы данных;
- использовать алгоритмы косвенных и совокупных измерений при автоматизации вычислительных процедур;
- достичь высоких метрологических характеристик, трудно доступных или недосягаемых обычными путями;
- проводить самокалибровку и самодиагностику с локализацией неисправностей вплоть до модуля;

- проводить статистическую обработку результатов измерений в автоматическом режиме, как органическую часть измерительной процедуры;
- оценивать значение погрешностей измерений и отображать их по ходу измерений на дисплее;
- выполнять математические функциональные преобразования, такие как линеаризация зависимости, вознесение в квадрат, нахождение отношений значений двух величин, выражение результатов измерения в децибелах и т.п.;
- увеличить объем измерений при сохранении высокой производительности.

Вопросы для самопроверки

1. Приведите классификацию средств измерений по функциональному назначению.
2. Что такое измерительная информационная система?
3. Приведите классификацию ИИС по функциональному назначению.
4. Назовите основные функциональные блоки ИИС, укажите их назначение.
5. Изобразите основные структуры ИИС, укажите достоинства и недостатки каждой из них.
6. Назовите основные типы организации алгоритма функционирования ИИС. Какие особенности присущи этим типам систем?
7. Приведите основные типы структур измерительных ИИС, укажите их преимущества и недостатки.
8. Что такое телеизмерительная система? Какие существуют типы телеизмерительных систем?
9. Какие функции выполняют системы автоматического контроля? Что такое непрерывный и дискретный контроль параметров?
10. Укажите назначение и основные функции систем технической диагностики.
11. Какие функции выполняют микропроцессоры в измерительных системах и устройствах?
12. Какие преимущества дает использование микропроцессоров в ИИС?

2 БАЗОВЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ МИКРОПРОЦЕССОРНОЙ АРИФМЕТИКИ

Микропроцессорная арифметика – совокупность принципов и форм представления числовой информации, методов и алгоритмов выполнения арифметических операций и вычисления элементарных функций, рассматриваемых на уровне внутренней структурной организации технических средств микропроцессорных систем (МПС). Это часть вычислительной математики, ориентированной на логический уровень описания вычислительных структур и процессов в них.

Что такое информация вообще, в общем понимании? Термин **информация** происходит от латинского слова **information,** что означает:

- сведения;
- разъяснения;
- изложение.

Алгоритм – способ преобразования информации, который задается с помощью конечной системы правил. Информация представляется в виде совокупности цифр (чисел) в некоторой системе счисления, сами же цифры отображаются сигналами, имеющими конечное число уровней квантования.

Система счисления (СС) – совокупность приемов и правил для установления однозначного соответствия между любым числом и его представлением в виде некоторой совокупности знаков (символов).

Количественный эквивалент числа (КЭЧ) – некоторое количество, однозначно соответствующее числу. (На абстрактно-интуитивном уровне безотносительно к системам счисления и измерения). Каждой цифре в записи числа сопоставляется некоторое количество, выражаемое этой цифрой и называемое количественным эквивалентом цифры (КЭЦ) .

Длина числа – количество позиций (разрядов) в записи числа. В техническом аспекте длина числа интерпретируется как длина разрядной сетки (ДРС) . Для разных (СС) характерна различная ДРС, необходимая для записи одного и того же числа.

Все операции в МПС выполняются как последовательности в пространстве и во времени некоторых простейших, элементарных операций, называемых микрооперациями. К числу основных классов микроопераций относятся:

1) передача (прием, выдача) операнда;
2) сдвиг (арифметический, циклический, логический, модифицированный) операнда на заданное число разрядов;
3) прибавление к операнду или вычитание из него единицы (в более общем случае - некоторой постоянной величины);
4) сравнение операндов (по принципу "больше - меньше - равно");
5) поразрядные логические операции (дизъюнкции, конъюнкции, равнозначности, сложения по модулю 2);

6) арифметическое сложение двух операндов, соответствующих числам в одной и той же системе счисления;

7) преобразование кодов операндов (включая инверсию, дополнение, шифрацию, дешифрацию и др.).

2.1 Системы счисления

Как было отмечено выше **система счисления** – совокупность приемов и правил для установления однозначного соответствия между любым числом и его представлением в виде некоторой совокупности знаков (символов). Запись числа в некоторой системе счисления называют кодом числа . Кратко число записывается следующим образом:

$$A = a_n a_{n-1} .... a_2 a_1 a_0, a_{-1} .... a_{-m} \ .$$

где A – количественный эквивалент числа (A); $(a_n a_{n-1} .... a_2 a_1 a_0, a_{-1} .... a_{-m})$ – цифры из множества, с помощью которых можно представить число (A).

Отдельную позицию в изображении числа принято называть разрядом, а номер позиции – номером разряда. Число разрядов в записи числа называется разрядностью и совпадает с его длиной. В техническом аспекте длина числа интерпретируется как длина разрядной сетки. Если алфавит имеет (p) различных значений, то разряд (a_i) в числе рассматривается как (p-ичная) цифра, которой может быть присвоено каждое из (p) значений. Каждой цифре (a_i) числа (A) однозначно соответствует ее количественный (числовой) эквивалент – $(K(a_i))$. Количественный эквивалент числа – (КЭЧ) – (A), заданного в определенной системе счисления, является некоторой функцией числовых эквивалентов всех его цифр, т.е.:

$$K(A) = f[K(a_n) ..... K(a_0)].$$

где $K(A)$ – количественный эквивалент числа (A); $K(a_n)$ – максимальный количественный (числовой) эквивалент цифры числа (A), находящийся в крайнем левом разряде; $K(a_0)$ – минимальный количественный (числовой) эквивалент цифры числа (A), находящийся в крайнем правом разряде.

Тогда при любой конечной разрядной сетке КЭЧ (A) будет принимать в зависимости от количественных эквивалентов отдельных разрядов значения от $K(A)_{\min}$ до $K(A)_{\max}$.

Диапазон представления (D) чисел в данной системе счисления – это интервал числовой оси, заключенный между максимальными и минимальными числами, представленными заданной разрядностью (длинной разрядной сетки):

$$D = K(A)_{(p)\max} ..... K(A)_{(p)\min} .$$

где D – диапазон представимых чисел в определенной системе счисления;

$K(A)_{(p)\max}$ – максимальный количественный эквивалент числа (A) по основанию (p);

$K(A)_{(p)\min}$ – минимальный количественный эквивалент числа (A) по основанию (p).

Любая система счисления, предназначенная для практического использования, должна обеспечивать:

- возможность представления любого числа в заданном диапазоне чисел;
- однозначность представления;
- краткость и простоту записи чисел;
- легкость овладения системой, а также простоту и удобство оперирования ею.

2.1.1 Классификация систем счисления

В настоящее время различают **позиционные** и **непозиционные** системы счисления. Классификация систем счисления приведена на рис. 2.1.

Рисунок 2.1 – Классификация систем счисления

Непозиционная система счисления – это такая система счисления, в которой каждой цифре на любом ее месте в записи числа однозначно соответствует один и тот же количественный эквивалент. Наиболее известным примером такой системы является **римская система счисления**, рис. 2.2, например:

Десятичные числа:	1	5	10	50	100	500	1000
Римские цифры:	I	V	X	L	C	D	M

Рисунок 2.2 – Соответствие десятичных чисел римским цифрам

В римской системе счисления несколько стоящих рядом одинаковых цифр суммируются рис.2.3:

$$XXX = X + X + X = 30_{(10)}.$$

Рисунок 2.3 – Пример записи числа с основанием (10) в Римской системе счисления

Если рядом стоят разные цифры, причем младшая – справа от старшей, то они также суммируются, рис.2.4:

$$XVI = X + V + I = 16_{(10)}.$$

Рисунок 2.4 – Пример записи числа с основанием (10) в Римской системе счисления

Если же младшая цифра находится слева от старшей, то она вычитается из этой старшей цифры рис.2.5:

$$IX = X - I = 9_{(10)}.$$

Рисунок 2.5 – Пример записи числа с основанием (10) в Римской сстеме счисления

Недостатки римской системы счисления заключаются в следующем:
- в пределе, теоретически, она имеет бесконечное количество цифр;
- арифметические действия над числами очень сложны;
- отсутствует цифра $\{0\}$.

Позиционная система счисления - это такая система счисления, в которой одной и той же цифре в зависимости от ее местоположения в записи числа соответствуют различные количественные эквиваленты. Наиболее известным примером такой системы является **десятичная система счисления,**

например: цифры **(**1**)** и **(**2**)** в зависимости от местоположения этих цифр в числе изменяется значение самого числа рис.2.6:

Разряды:	Десятки	Единицы
Цифры:	1	2

Рисунок 2.6 – Пример записи числа в десятичной системе счисления

При таком положении цифр получается число **двенадцать** ($12_{(10)}$). Нижний индекс при записи числа обозначает основание системы счисления, в данном случае $_{(10)}$ означает десятичную систему счисления. Если поменять местами цифры (1) и (2), рис.2.7:

Разряды:	Десятки	Единицы
Цифры:	2	1

Рисунок 2.7 – Пример записи числа в десятичной системе счисления

получается число **двадцать один** ($21_{(10)}$). Для определения количественного эквивалента полной записи числа в позиционной системе счисления используется некоторая функция от количественных эквивалентов цифр. Если этой функцией является функция сложения, то систему называют **аддитивной ,** если же используется функция умножения – систему называют **мультипликативной .**

Любое число в позиционной системе счисления может быть записано в виде:

$$A_{кэч} \Rightarrow a_n p_{n-1} \dots p_1 + a_{n-1} p_{n-2} \dots p_1 + \dots + a_2 p_1 + a_1, \qquad (2.1)$$

где $A_{кэч}$ – количественный эквивалент числа (A), состоящего из (n) цифр; a – цифра, $0 \le a \le \Rightarrow p_{i-1}$; p – основание системы счисления; n – количество разрядов в числе.

В левой части равенства записано символическое изображение числа. В правой части равенства показано, что все цифры числа в разных позициях имеют разный вес, при этом каждая позиция с присвоенными ей номером и весом называется – разрядом числа.

Правило: Количественный эквивалент числа в позиционной системе счисления равен сумме произведений количественных значений цифр и степеней основания, показатели которых равны номерам разрядов, причем нумерация разрядов начинается с (0).

Например: $A = 4872$, $n = 4, p = 10$, тогда можно записать:

$$A_p = a_4 p_3 p_2 p_1 + a_3 p_2 p_1 + a_2 p_1 + a_1 .$$

тогда:

$$A_{10} = 4 \cdot 10 \cdot 10 \cdot 10 + 8 \cdot 10 \cdot 10 + 7 \cdot 10 + 2 .$$

Однородность системы счисления означает, что во всех разрядах числа, записанного в такой системе, используют цифры из одного и того же множества.

Например, в обычной десятичной системе счисления во всех разрядах числа используются цифры из множества рис.2.8:

$$\{0,1,2,3,4,5,6,7,8,9\} .$$

Рисунок 2.8 – Множество цифр использующихся в десятичной системе счисления

В двоичной системе счисления используются цифры из множества рис.2.9:

$$\{0,1\} .$$

Рисунок 2.9 – Множество цифр использующихся в двоичной системе счисления

в троичной системе счисления используются цифры из множества рис.2.10:

$$\{0,1,2\} .$$

Рисунок 2.10 – Множество цифр использующихся в троичной системе счисления

в пятеричной системе счисления используются цифры из множества рис.2.11:

$$\{0,1,2,3,4\} .$$

Рисунок 2.11 – Множество цифр использующихся в пятеричной системе счисления

в восьмеричной системе счисления используются цифры из множества рис.2.12:

$$\{0,1,2,3,4,5,6,7\} .$$

Рисунок 2.12 – Множество цифр использующихся в восьмеричной системе счисления

в шестнадцатеричной системе счисления при записи числа используются цифры и буквы рис.2.13:

$$\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}.$$

Рисунок 2.13 – Множество цифр и букв использующихся в шестнадцатеричной системе счисления

Если позиционная система счисления **однородная с непосредственным представлением цифр и с естественным порядком весов**, то любое число может быть представлено в виде суммы попарных произведений:

$$A=\sum_{i=-m}^{n-1} a_i \cdot p^i .$$

где A – количественный эквивалент числа (A); a – цифра, $0 \le a \le \Rightarrow p_{i-1}$; p – основание системы счисления; n – количество разрядов в целой части числа слева от запятой; m – количество разрядов в дробной части числа справа от запятой.

Исходя из выше сказанного можно записать:

$$123_{10} = 1\cdot 10^2 + 2\cdot 10^1 + 3\cdot 10^0 = 100 + 20 + 3 = 123.$$

или $118{,}375 = 1\cdot 10^2 + 1\cdot 10^1 + 8\cdot 10^0 + 3\cdot 10^{-1} + 7\cdot 10^{-2} + 5\cdot 10^{-3} = 118{,}375.$

Соответствие чисел в (10-ой), (16-ой), (8-ой) и (2-ой) системах счисления приведены в табл. 2.1.

Таблица 2.1 – Соответствие чисел в (10-ой), (16-ой), (8-ой) и (2-ой) системах счисления

Десятеричная $X_{(10)}$	Шестнадцатеричная $X_{(16)}$	Восьмеричная $X_{(8)}$	Двоичная $X_{(2)}$
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100

5	5	5	0101
6	6	6	0110
7	7	7	0111
8	8	10	1000
9	9	11	1001
10	A	12	1010
11	B	13	1011
12	C	14	1100
13	D	15	1101
14	E	16	1110
15	F	17	1111

Помимо **позиционных однородных систем** известны также **позиционные неоднородные (смешанные) системы счисления**.

В таких системах цифры в разных разрядах могут принимать значения из различных множеств.

Задают неоднородные системы с помощью двухстрочных матриц вида:

$$M = \left| \begin{array}{c} t_m \;\; t_{m\text{-}1} \ldots t_i \ldots t_1 \\ k_m \;\; k_{m\text{-}1} \ldots k_i \ldots k_1 \end{array} \right|.$$

Здесь в первой строке матрицы указано число разрядов (t_i), отводимых в (i-й) группе разрядов ($i = \overline{1,m}$) представления числа для записи цифр по основанию (k_i), которое указано во второй строке того же столбца.

Неоднородные системы счисления, так же как и однородные, могут быть с **непосредственным** и с **кодированным** представлением цифр.

Примером смешанной системы с кодированным представлением цифр является система измерения времени (в годах, месяцах, неделях, сутках, часах, минутах и секундах).

Например: надо выразить время в 2 – года, 25 – суток, 14 – часов, 35 – минут и 48 – секунд, в секундах. Тогда можно записать, что основание в каждом разряде равно:

$$p_1 = 60 \textit{сек};$$
$$p_2 = 60 \textit{мин};$$
$$p_3 = 24 \textit{часа};$$
$$p_4 = 365 \textit{суток};$$

цифры имеют следующие значения:

$$a_1 = 48;$$
$$a_2 = 35;$$
$$a_3 = 14;$$
$$a_4 = 25;$$
$$a_5 = 2.$$

По формуле (2.1) можно записать:

$$A = 2 \cdot 365 \cdot 24 \cdot 60 \cdot 60 + 25 \cdot 24 \cdot 60 \cdot 60 + 14 \cdot 60 \cdot 60 + 35 \cdot 60 + 48.$$

Существует так же неоднородная двоично-пятиричная система счисления, в которой в нечетных разрядах основание $p_1 = 5, (a_i = 0 \div 4)$, а в четных разрядах основание $p_2 = 2, (a_i = 0 \div 1)$.

Так как произведение весов двух соседних (четного и нечетного) разрядов равно десяти, то двумя двоично-пятиричными разрядами можно кодировать одну десятичную цифру.

Пример записи десятичных цифр от 0 до 9 в двоично-пятиричной системе приведен в табл. 2.2.

Таблица 2.2 – Пример записи десятичных цифр в двоично-пятиричной 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

Например: записать число $853_{(10)}$ в двоично-пятиричной системе счисления.

Решение: исходя из значений, представленных в табл. 2.2 имеем:

$$8_{(10)} = 13_{(2\text{-}5)}$$
$$5_{(10)} = 10_{(2\text{-}5)}$$
$$3_{(10)} = 03_{(2\text{-}5)}$$

тогда: $A_{(10)} = 131003_{(2-5)}$.

Существует так же **кодированные системы счисления** – это позиционные системы счисления, в которых цифры одной системы счисления кодируются при помощи цифр другой системы счисления, а число в общем виде записывается следующим образом:

$$A = (a_{n,k} p^n + a_{n-1,k} p^{n-1} + ... + a_{1,k} p^1 + a_{0,k} p^0) P^k +$$
$$+ (a_{n,k-1} p^n + ... a_{0,k-1} p^0) P^{k-1} + ... + (a_{n,0} p^n + ... + a_{1,0} p^1 + a_{0,0} p^0) P^0.$$

где: A – число; a – цифра; p – основание системы счисления, символами которой кодируются цифры; P – основание исходной системы счисления.

Классическим примером кодированной системы счисления есть – двоично-десятичная система счисления.

При двоично-десятичном кодировании каждая десятичная цифра заменяется тетрадой (четверкой) двоичных цифр, а сами тетрады записываются последовательно в соответствии с порядком следования десятичных цифр.

При обратном преобразовании двоично-десятичного кода в десятичный исходный код разбивается на тетрады вправо и влево от запятой, которые затем заменяются десятичными цифрами.

Таким образом, при двоично-десятичном кодировании фактически не производится перевод числа в новую систему счисления, а мы имеем дело с двоично-кодированной десятичной системой счисления.

Например: десятичное число $12_{(10)}$ записать в двоично-десятичной системе счисления $= 00010010_{(2-10)}$.

При построении кодированных позиционных систем счисления в качестве весов разрядов могут быть выбраны как члены геометрической прогрессии, так и произвольные числа.

В зависимости от этого кодированные системы счисления делятся на кодированные системы счисления с естественными разрядами весов и на кодированные системы счисления с искусственными разрядами весов.

Кодированные системы счисления с естественными разрядами весов – это позиционная система счисления, в которой в качестве весов разрядов используются члены геометрической прогрессии.

Примером системы счисления с естественными разрядами весов может служить двоично-десятичная система с весами $(8-4-2-1)$.

Кодированные системы счисления с искусственными разрядами весов – это позиционная система счисления, в которой в качестве весов разрядов используются произвольные числа.

Примером системы счисления с искусственными разрядами весов может служить двоично-десятичная система с весами $(2-4-2-1,\ 4-2-2-1)$.

Искусственный порядок весов широко применяется в аналогово-цифровых и цифро-аналоговых преобразователях.

В табл. 2.3 представлены числа, записанные в десятичной системе счисления и в кодированной системе счисления с естественным и искусственным порядком весов.

Таблица 2.3 – Числа десятичной системы счисления и кодированной системе счисления с естественным и искусственным порядком весов

Десятичная СС	Кодированная СС 8-4-2-1	Кодированная СС 4-2-2-1	Кодированная СС 2-4-2-1
0	0000	0000	0000
1	0001	0001	0001

2	0010	0010	0010
3	0011	0011	0011
4	0100	0110	0100
5	0101	0111	0101
6	0110	1010	0110
7	0111	1011	0111
8	1000	1110	1110
9	1001	1111	1111

Например: десятичное число $1593_{(10)}$ в двоично-десятичной системе счисления с естественными разрядами весов 8-4-2-1 имеет вид: 0001 0101 1001 0011, а в двоично-десятичной системе счисления с искусственными разрядами весов 2-4-2-1 имеет вид: 0001 0101 1111 0011.

В современных микропроцессорных системах помимо рассмотренных систем счисления встречаются и системы счисления с непостоянными разрядами весов. Наиболее известным примером таких систем является **код Грея**.

Кодом Грея порядка (n) называется любая циклическая последовательность всех наборов из (0) и (1) длины (n), в которой два соседних набора отличаются ровно в одной компоненте.

Код Грея является одношаговым кодом, т.е. при переходе от одного числа к другому всегда меняется лишь какой то один из всех битов. Соответствие десятичных чисел в диапазоне от 0 до 15 двоичным числам и коду Грея приведено в табл. 2.4.

В двоичном коде при переходе от изображения одного числа к изображению соседнего числа может происходить одновременное изменение цифр в нескольких разрядах, что может явиться источником ошибок, в работе аппаратуры в некоторых случаях например при переходе от 7 к 8.

В коде Грея два соседних значения отличаются только в одном разряде.

Двоичные разряды в коде Грея не имеют постоянного веса. Код Грея изначально предназначался для защиты от ложного срабатывания электрических переключателей. Сегодня код Грея широко используется для упрощения выявлений и исправления ошибок в системах связи.

Таблица 2.4 – Соответствие десятичных чисел в диапазоне от 0 до 15 двоичным числам и коду Грея

Десятичные числа	Двоичные числа	Код Грея
1	2	3
0	0000	0000
1	0001	0001
2	0010	0011

Продолжене таблицы 2.4

1	2	3
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

2.1.2 Выбор системы счисления для использования в микропроцессорных системах

Непозиционные системы счисления непригодны в силу своей громоздкости и трудности выполнения арифметических операций.

Из позиционных систем счисления наиболее удобны однородные системы счисления, так как одинаковое основание, т.е. одинаковое количество символов во всех разрядах приводит к наиболее рациональному использованию оборудования и наиболее простым алгоритмам выполнения арифметических операций. Поэтому проанализируем однородные позиционные системы счисления на предмет их применения в микропроцессорных системах. При этом будем учитывать следующие факторы:

- **простота технической реализации;**

  очевидно, что (p – позиционный) запоминающий элемент будет тем проще, чем меньше состояний (позиций) ему требуется иметь, т.е. чем меньше основание системы счисления.

- **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. Если увеличить диапазон некоторого носителя информации при реализации одной системы счисления, то его можно увеличить и при реализации другой системы. Но тогда очевидно преимущество систем с малыми основаниями, так, как представления цифр в этих системах отличаются друг от друга в более значительной степени, чем в системах с большими основаниями. Это значит, что при наложении на основной сигнал,

изображающий цифру, некоторой помехи произвольной формы, наибольшая ошибка возможна в устройстве, использующем систему счисления с самым большим основанием.

– **минимум оборудования;**

чтобы решить, какой системе отдать предпочтение с точки зрения получения минимума расхода оборудования при прочих равных условиях, сделаем предположение, что все МПС оперируют с одним и тем же количеством чисел. Если (b_i) – количество цифр, с которыми оперирует (i-я) МПС; (n_i) – количество разрядов, в каждом числе (i-ой) МПС, то произведение:

$$D_i = b_i \cdot n_i, \tag{2.2}$$

где D_i – количество цифроразрядов, приходящихся на 1 число; b_i – количество цифр;

n_i – количество разрядов.

Таким образом, задача в данном случае сводится к нахождению такой системы счисления, которая имеет самое малое количество цифроразрядов при заданном количестве чисел – операндов (N):

$$N = b_i^{n_i}, \tag{2.3}$$

формулу (2.3) можно переписать в другом виде:

$$n_i = \log_{b_i} N, \tag{2.4}$$

подставив это значение (n_i) в соотношение (2.2), получим:

$$D = b_i \cdot \log_{b_i} N = \frac{b_i}{\log_N b_i}, \tag{2.5}$$

будим считать, что основание системы счисления может принимать любые значения.

Тогда количество цифроразрядов также будет величиной непрерывной, связанной с основанием системы счисления логарифмической функциональной зависимостью:

$$D(b) = \frac{b_i}{\log_N b_i}, \tag{2.6}$$

теперь задача нахождения $D(b)_{\min}$ сводится к обыкновенному исследованию функции (2.6) на экстремум:

$$\frac{dD}{db} = \frac{1 \cdot \log_N b - b \cdot \frac{1}{b} \log_N e}{(\log_N b)^2} = 0, \qquad (2.7)$$

откуда $b_{opt} = e \approx 2,718 \Rightarrow k_{np.opt} = 3$. Выясним теперь, насколько каждое из целочисленных оснований (b_i) уступает (b_{opt}). Для этого оценим каждое основание (b_i) не абсолютной величиной (D_i), а его относительным значением:

$$D_{i(отн.)} = \frac{D_i}{D_{\min}}, \qquad (2.8)$$

где:

$$D_{\min} = b_{opt} \cdot \log_{b_{opt}} N = e \cdot \ln N, \qquad (2.9)$$

Подставив выражение (2.9) в формулу (2.8), получим выражение, не зависящее от величины (N):

$$D_{i(отн.)} = \frac{b_i \log_{b_i} N}{e \times \ln N} = \frac{b_i}{e \times \ln b_i}, \qquad (2.10)$$

Произведя расчеты по формуле (2.10) для некоторых оснований системы счисления, сведем полученные результаты в табл. 2.5:

Таблица 2.5 – Соотношение оснований систем счисления и эффективности

b_i	2	3	4	5	6	7	8	9	10
$D_{i(отн.)}$	1,062	1,004	1,062	1,143	1,232	1,3	1,416	1,507	1,597

- **простота арифметических действий**;
  чем меньше цифр в системе счисления, тем проще арифметические действия. Таблицы сложения, вычитания, умножения и деления будут усложняться с увеличением основания системы счисления.
- **наибольшее быстродействие**;
  более сложные операции такие как умножение и деление, расчленяются на более простые операции такие как операции сложения и сдвига. С увеличением основания системы счисления быстродействие будет падать.
- **простота формального аппарата для синтеза цифровых устройств**;

математическим аппаратом, позволяющим относительно просто и экономично строить цифровые схемы, узлы, блоки, является алгебра логики. Наибольшее развитие и законченность вследствие своей простоты и широкого практического распространения двухпозиционных элементов в настоящее время получила двузначная логика.

– **удобство работы;**

безусловно наилучшей системой счисления с точки зрения удобства работы человека является десятичная, так как в любом другом случае потребуется переводить исходные числа из десятичной системы в принятую, а затем обратно в десятичную. Поэтому можно утверждать, что все системы счисления с основаниями превышающими десятичную систему счисления, являются менее удобными, чем системы с малыми основаниями.

В соответствии с рассмотренными критериями формируется комплексный показатель. При этом максимальную оценку получает система счисления с основанием два. Это позиционная система счисления, в которой для изображения чисел используются два символа, а веса разрядов меняются по закону ($2^{\pm l}$).

Большой практический интерес представляют двоично-кодированные системы счисления: двоично-десятичная Binary Code Decimal (BCD), восьмеричная (octal), шестнадцатеричная (hexadecimal). В них каждый разряд записывается с помощью нескольких двоичных разрядов, имеющих определенные веса. Цифры восьмеричной системы представляются тремя двоичными разрядами (триадами), десятичной и шестнадцатеричной – четырьмя (тетрадами). Также следует отметить, что восьми- и шестнадцатиричные системы относятся к особому классу систем с основанием кратным (2). Эта особенность обуславливает их широкое применение в процессе взаимодействия с пользователем микропроцессорной системы. С одной стороны, они облегчают восприятие двоичной информации, так как любое двоичное представление числа может быть разбито на триады или тетрады, которые проще для визуального восприятия человеком. С другой стороны, они часто используются в качестве промежуточных систем счисления при переводе из десятичной системы счисления в двоичную систему счисления и наоборот.

Двоичная система счисления вводит понятие основной структурной единицы информации, которой является бит (bit – **bi**nary digi**t**). Бит соответствует одному двоичному разряду числа, представленного в двоичной системе счисления.

Однако бит является слишком мелкой единицей и не всегда удобен для практических приложений. Поэтому наряду с битом в микропроцессорной системе получила распространение производная структурная единица информации – машинное слово, под которым понимают упорядоченную совокупность битов, имеющую некоторый смысл, т.е. воспринимаемую

основными устройствами микропроцессора как единое целое. Длина машинного слова является одной из определяющих характеристик микропроцессорной системы и выбирается исходя, прежде всего, из соображений точности. Таким образом, длина машинного слова является различной для микропроцессорных систем разных классов. Для унифицированного представления информации используют машинно-независимую структурную единицу – байт – byte – (кусок, часть) – совокупность из восьми смежных двоичных разрядов. Длину машинного слова обычно выбирают равной целому числу байт. Восьмиразрядный байт удобен тем, что полностью покрывает всю символьную информацию и удобен для записи десятичных и 16-ричных тетрад.

2.1.3 Перевод чисел из одной позиционной системы счисления в другую

Задача перевода чисел из одной позиционной системы счисления в другую является одной из главных в микропроцессорной арифметике. Ее можно сформулировать следующим образом: требуется перевести некоторое число (X), записанное в позиционной системе счисления с основанием (k_1), в такую же систему счисления, имеющую основание (k_2).

Другими словами: по изображению операнда (X) в системе счисления с основанием (k_1) найти изображение (Y) того же операнда в системе с основанием (k_2).

$$X_{(k_1)} \Rightarrow n = s + m.$$

$$Y_{(k_2)} \Rightarrow q = r + l.$$

Существуют 2 группы методов перевода: табличные и расчетные.

1. Табличные методы.

В простейшем случае в памяти микропроцессорной системы хранится таблица соответствия между всеми числами в системах счисления с основаниями (k_1) и (k_2), а сама процедура перевода сводится к обращению к этой таблице. Плюс табличных методов перевода заключается в высокой скорости перевода. Минус табличных методов перевода заключается в том, что размеры такой таблицы и, следовательно, занимаемый ею объем памяти, часто оказываются технически неприемлемыми. Поэтому с целью уменьшения занимаемого объема памяти в ней хранят только таблицы соответствия цифр заданных систем счисления и весов их разрядов. Перевод чисел осуществляется путем обращения к этим таблицам и выполнения операций умножения и сложения в соответствии с выражением для КЭЧ. Если, например, числа в системе с основанием (k_1) представлены (n – разрядами), то по первому

варианту размерность таблицы, сохраняемой в памяти, определяется (k^n) строками, а по второму варианту – ($k_1 + n + 1$) строками.

2. Расчетные методы.

В общем виде решение задачи перевода можно представить как нахождение коэффициентов (y_j) нового ряда, изображающего число в системе счисления с основанием (k_2).

Все действия должны выполняться по правилам (k_1-арифметики), т.е. по правилам исходной арифметики. После нахождения максимальной степени основания проверяют «вхождение» в заданное число всех степеней нового основания, меньших максимального.

Каждая из отмеченных степеней может входить в ряд не более ($k_2 - 1$) раз, что определяется условием:

$$0 \le x_i \le (k_1 - 1)$$
$$0 \le y_j \le (k_2 - 1).$$

Для перевода операнда (X) в систему с основанием (k_2) необходимо записать (X) в форме для вычисления количественного эквивалента, далее заменить цифры (x_i) и основания (k_1) их эквивалентами в системе с основанием (k_2), а потом вычислить полученное выражение по правилам арифметики в системе с основанием (k_2). Этот алгоритм удобно использовать в случае, когда ($k_1 < k_2$), причем (k_2) соответствует системе счисления, где просто и "привычно" выполняются операции сложения и умножения (например, десятичной системе). Для упрощения вычислений при этом используют схему Горнера, в соответствии с которой формула для КЭЧ преобразуется путем многократного вынесения за скобки:

$$Y = (...((x_{s-1}k_1 + x_{s-2})k_1 + x_{s-3})k_1 + ... + x_1)k_1 + x_0 + ... + (..(((0 + x_m)k_1^{-1} + \\ + x_{m+1})k_1^{-1} + x_{m+2})k_1^{-1} + ... + x_1)k_1^{-1}, \quad (2.11)$$

Отсюда видно, что для получения целой части числа необходимо выполнить (s) шагов вычислений, а для получения дробной – (m) шагов вычислений. Таким образом вышеописанный алгоритм, состоит из двух алгоритмов, а именно: перевода целой части числа, выполняемого в соответствии с рекуррентной формулой (2.12):

$$A_i = A_{i-1}k_1 + x_{s-i},\ i = \overline{1, s} + 1, \quad (2.12)$$

где $A_0 = 0$, A_s – целая часть исходного числа в системе счисления с основанием (k_2); и переводе дробной части числа, выполняемого в соответствии с рекуррентной формулой(2.12):

$$B_j = (B_{j\text{-}1} + x_{\text{-}1+j})k^{\text{-}1},\ j = \overline{1,m}\ , \qquad (2.13)$$

где $B_0 = 0$, B_m – дробная часть исходного числа в системе с основанием (k_2).

Рассмотренный алгоритм не имеет каких-либо теоретических ограничений на область своего применения.

Однако при переводе целых чисел в системы с "непривычными" основаниями, особенно в случае ($k_1 > k_2$), использование этого алгоритма связано с весьма громоздкими вычислениями.

Поэтому на практике используют отдельные алгоритмы перевода целых чисел и правильных дробей.

В случае ($k_1 > k_2$). Целое число (X) запишем в системе с основанием (k_2) с использованием схемы Горнера:

$$Y = (...(((y_{r-1}k_2 + y_{r-2})k_2 + y_{r-3})k_2 + ... + y_1)k_2 + y_0\,.$$

Правую часть выражения разделим на величину основания (k_2). В результате получим первый остаток (y_0) и целую часть:

$$Y = (...(((y_{r-1}k_2 + y_{r-2})k_2 + y_{r-3})k_2 + ... + y_1)\,.$$

Разделив целую часть на (k_2), найдем второй остаток(y_1).

Повторяя процесс деления r раз, получим последнее целое частное, которое, по условию, меньше основания системы счисления (k_2) и является старшей цифрой числа, представленного в системе с основанием (k_2).

Правило перевода целых чисел из одной позиционной системы счисления в другую:

- для перевода целого числа в новую систему счисления его надо последовательно делить на основание новой системы счисления до тех пор, пока не получится частное, у которого целая часть равна (0).

Число в новой системе счисления записывается из остатков от последовательного деления, причем последний остаток будет старшей цифрой нового числа.

Алгоритм перевода целых чисел из одной позиционной системы счисления в другую можно сформулировать следующим образом: операнд (X) необходимо делить на (k_2) по правилам целочисленного деления в исходной системе сисления с основанием (k_1) до получения остатка.

Если частное от такого деления не нуль, то далее частное рассматривается как делимое и процесс деления на (k_2) продолжают.

Как только очередное частное станет равным нулю, процесс деления прекращают.

Остаток, полученный при первом делении на (k_2), представляет собой цифру результата с весом (k_2^0), остаток от второго деления – цифру результата с весом (k_2^1) и т.д.

Последний остаток является старшей цифрой результата.

Например: необходимо перевести из десятичной системы счисления в двоичную систему счисления целое число $A = 53_{(10)}$.

Решение: произведем последовательное деление исходного числа ($53_{(10)}$) на основание новой системы счисления (2). Перевод целого числа $A = 53_{(10)}$ из десятичной системы счисления в двоичную систему счисления, приведен на рис. 2.14:

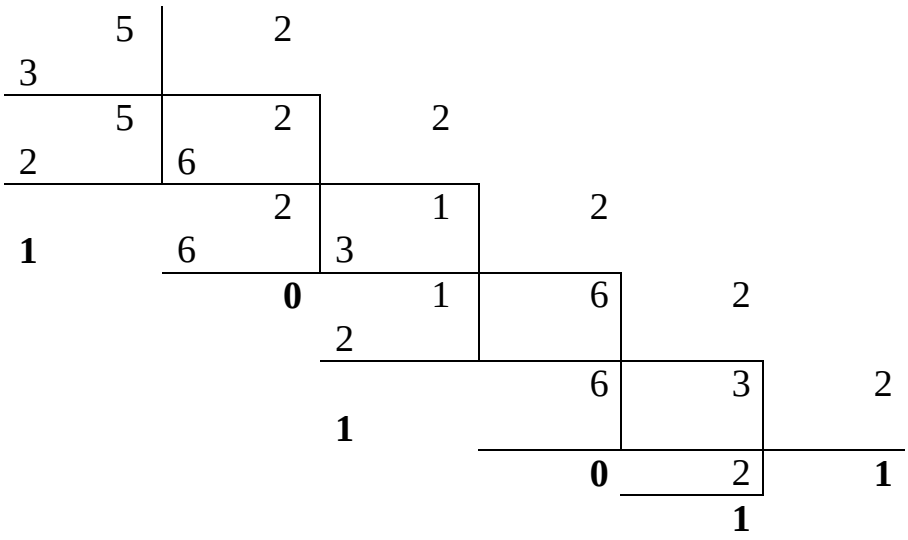


Рисунок 2.14 – Перевод целого числа $A = 53_{(10)}$ из десятичной системы счисления в двоичную систему счисления

Из сформулированного выше правила при переводе целого числа $A = 53_{(10)}$ из десятичной системы счисления в двоичную систему счисления получим:

$$53_{(10)} - 110101_{(2)}.$$

Например: необходимо перевести из десятичной системы счисления в троичную систему счисления целое число $A = 53_{(10)}$.

Решение: произведем последовательное деление исходного числа $53_{(10)}$ на основание новой системы счисления (3). Перевод целого числа $A = 53_{(10)}$ из десятичной системы счисления в троичную систему счисления, приведен на рис. 2.15:

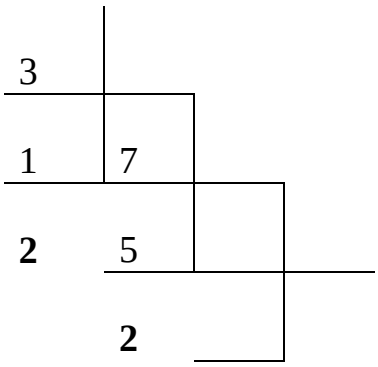


Рисунок 2.15 – Перевод целого числа $A = 53_{(10)}$ из десятичной системы счисления в троичную систему счисления

Из сформулированного выше правила при переводе целого числа $A = 53_{(10)}$ из десятичной системы счисления в троичную систему счисления получим:

$$53_{(10)} - 1222_{(3)}.$$

Например: необходимо перевести из десятичной системы счисления в пятеричную систему счисления целое число $A = 53_{(10)}$.

Решение: произведем последовательное деление исходного числа $(53_{(10)})$ на основание новой системы счисления (5). Перевод целого числа $A = 53_{(10)}$ из десятичной системы счисления в пятеричную систему счисления, приведен на рис. 2.16:

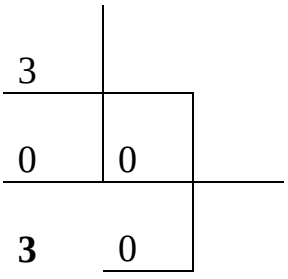


Рисунок 2.16 – Перевод целого числа $A = 53_{(10)}$ из десятичной системы счисления в пятеричную систему счисления

Из сформулированного выше правила при переводе целого числа $A = 53_{(10)}$ из десятичной системы счисления в пятеричную систему счисления получим:

$$53_{(10)} - 203_{(5)}.$$

Например: необходимо перевести из десятичной системы счисления в восьмеричную систему счисления целое число $A = 53_{(10)}$.

Решение: произведем последовательное деление исходного числа $(53_{(10)})$ на основание новой системы счисления (8). Перевод целого числа $A = 53_{(10)}$ из десятичной системы счисления в восьмеричную систему счисления, приведен на рис. 2.17:

53	8
48	**6**
5	

Рисунок 2.17 – Перевод целого числа $A = 53_{(10)}$ из десятичной системы счисления в восьмеричную систему счисления

Из сформулированного выше правила при переводе целого числа $A = 53_{(10)}$ из десятичной системы счисления в восьмеричную систему счисления получим:

$$53_{(10)} - 65_{(8)}.$$

Например: необходимо перевести из десятичной системы счисления в шестнадцатеричную систему счисления целое число $A = 53_{(10)}$.

Решение: произведем последовательное деление исходного числа $(53_{(10)})$ на основание новой системы счисления (16). Перевод целого числа $A = 53_{(10)}$ из десятичной системы счисления в шестнадцатеричную систему счисления, приведен на рис. 2.18:

3	6
8	

5

Рисунок 2.18 – Перевод целого числа $A = 53_{(10)}$ из десятичной системы счисления в шестнадцатеричную систему счисления

Из сформулированного выше правила при переводе целого числа $A = 53_{(10)}$ из десятичной системы счисления в шестнадцатеричную систему счисления получим:

$$53_{(10)} - 35_{(16)}.$$

При переводе из не десятичной системы счисления в десятичную систему счисления, ввиду ее не привычности для человека производство в ней арифметических действий значительно затруднено. В этом случае для преобразования чисел необходимо воспользоваться формулой:

$$A = \sum_{i=-m}^{s-1} a_i \cdot p^i .$$

Например: перевести из двоичной системы счисления в десятичную систему счисления целое число $A = 110101_{(2)}$.

Решение: Запишем число (A) в виде суммы произведений степеней основания на соответствующую цифру в десятичной системе счисления:

$$A = 110101_{(2)} = 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 1 \cdot 32 + 1 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 = 32 + 16 + 0 + 4 + 0 + 1 = 53_{(10)}.$$

Таким образом, получаем: $110101_{(2)} - 53_{(10)}$.

Например: перевести из троичной системы счисления в десятичную систему счисления целое число $A = 1222_{(3)}$.

Решение: Запишем число (A) в виде суммы произведений степеней основания на соответствующую цифру в десятичной системе счисления:

$$A = 1222_{(3)} = 1 \cdot 3^3 + 2 \cdot 3^2 + 2 \cdot 3^1 + 2 \cdot 3^0 = 1 \cdot 27 + 2 \cdot 9 + 2 \cdot 3 + 2 \cdot 1 = 27 + 18 + 6 + 2 = 53_{(10)}.$$

Таким образом, получаем: $1222_{(3)} - 53_{(10)}$.

Например: перевести из пятеричной системы счисления в десятичную систему счисления целое число $A = 203_{(5)}$.

Решение: Запишем число (A) в виде суммы произведений степеней основания на соответствующую цифру в десятичной системе счисления:

$$A = 203_{(5)} = 2 \cdot 5^2 + 0 \cdot 5^1 + 3 \cdot 5^0 = 2 \cdot 25 + 0 \cdot 5 + 3 \cdot 1 = 50 + 0 + 3 = 53_{(10)}.$$

Таким образом, получаем: $203_{(5)} - 53_{(10)}$.

Например: перевести из восьмеричной системы счисления в десятичную систему счисления целое число $A = 65_{(8)}$.

Решение: Запишем число (A) в виде суммы произведений степеней основания на соответствующую цифру в десятичной системе счисления:

$$A = 65_{(8)} = 6 \cdot 8^1 + 5 \cdot 8^0 = 6 \cdot 8 + 5 \cdot 1 = 48 + 5 = 53_{(10)}.$$

Таким образом, получаем: $65_{(8)} - 53_{(10)}$.

Например: перевести из шестнадцатеричной системы счисления в десятичную систему счисления целое число $A = 35_{(16)}$.

Решение: Запишем число (A) в виде суммы произведений степеней основания на соответствующую цифру в десятичной системе счисления:

$$A = 35_{(16)} = 3 \cdot 16^1 + 5 \cdot 16^0 = 3 \cdot 16 + 5 \cdot 1 = 48 + 5 = 53_{(10)}.$$

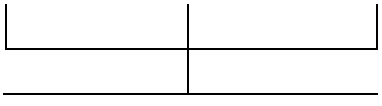
Таким образом, получаем: $35_{(16)} - 53_{(10)}$.

При переводе из недесятичной системы счисления в недесятичную систему счисления с основанием степени двойки, например если необходимо перевести число из двоичной системы счисления в систему счисления, основанием которой является степень двойки, достаточно объединить цифры двоичного числа в группы по столько цифр, каков показатель степени.

Например, если перевод осуществляется в восьмеричную систему счисления, то группы будут содержать три цифры $(8 = 2^3)$, такая группа называется **триадой.** Если перевод осуществляется в шестнадцатеричную систему счисления, то группы будут содержать четыре цифры $(16 = 2^4)$, такая группа называется **тетрадой.** В целой части числа группировка производится справа налево, в дробной части – слева направо. Если в последней группе недостает цифр, то дописываются нули: в целой части – слева, в дробной – справа. Затем каждая группа заменяется соответствующей цифрой новой системы счисления.

Например: необходимо из двоичной системы счисления перевести в восьмеричную систему счисления целое число: $A = 110101_{(2)}$.

Решение: исходя из вышесказанного, разобьем целое число: $A = 110101_{(2)}$ на триады и получим рис.2.19:



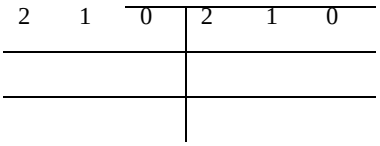


Рисунок 2.19 – Перевод целого числа $A=110101_{(2)}$ в восьмеричную систему счисления

Таким образом, получаем: $110101_{(2)}-65_{(8)}$.

Например: необходимо из двоичной системы счисления перевести в шестнадцатеричную систему счисления целое число: $A=110101_{(2)}$.

Решение: исходя из вышесказанного, разобьем целое число: $A=110101_{(2)}$ на тетрады и получим рис.2.20:

0	**0**	**1**	**1**	**0**	**1**	**0**	**1**
2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0
8	4	2	1	8	4	2	1
3				**5**			

Рисунок 2.20 – Перевод целого числа $A=110101_{(2)}$ в шестнадцатеричную систему счисления

Таким образом, получаем: $110101_{(2)}-35_{(16)}$.

При переводе из недесятичной системы счисления в недесятичную систему счисления с основанием степени двойки, например если необходимо перевести число из восьмеричной системы счисления в шестнадцатеричную систему счисления, необходим промежуточный перевод в двоичную систему счисления. Сначала сформировать триады, а затем тетрады. То же самое, если необходимо перевести число из шестнадцатеричной системы счисления в восьмеричную систему счисления необходим промежуточный перевод в двоичную систему счисления. Сначала сформировать тетрады, а затем триады.

Например: необходимо из восьмеричной системы счисления перевести в шестнадцатеричную систему счисления целое число: $A=65_{(8)}$.

Решение: исходя из вышесказанного, переведем сначала целое число $A=65_{(8)}$ в двоичную систему счисления, разбив его на триады, а затем триады преобразовав в тетрады получим число в шестнадцатеричной системе счисления рис.2.21:

6			**5**			
1	1	0	1	0	1	
0	0	1	1	0	1	0	1
3			**5**			

Рисунок 2.21 – Перевод из восьмеричной системы счисления в шестнадцатеричную систему счисления целого числа $A = 65_{(8)}$

Таким образом, получаем: $65_{(8)} - 35_{(16)}$.

Например: необходимо из шестнадцатеричной системы счисления перевести в восьмеричную систему счисления целое число: $A = 35_{(16)}$.

Решение: исходя из вышесказанного, переведем сначала целое число: $A = 35_{(16)}$ в двоичную систему счисления, разбив его на тетрады, а затем тетрады преобразовав в триады получим число в восьмеричной системе счисления рис.2.22:

			3				**5**			
			0	0	1	1	0	1	0	1
0	0	0	1		1	0	1		0	1
	0			**6**				**5**		

Рисунок 2.22 – Перевод из шестнадцатеричной системы счисления в восьмеричную систему счисления целого числа $A = 35_{(16)}$

Таким образом, получаем: $35_{(16)} - 65_{(8)}$.

При переводе из десятичной системы счисления в двоично-десятичную систему счисления подразумевают, что двоично-десятичная система счисления представляет собой систему с основанием (10), цифры которой закодированы в виде четырехразрядных двоичных чисел (тетрад), либо с естественным порядком весов (8–4–2–1), либо с искусственным порядком весов. **Например:** необходимо из десятичной системы счисления перевести в двоично-десятичную систему счисления целое число: $A = 118_{(10)}$. **Решение:** исходя из вышесказанного, при переводе целого десятичного числа: $A = 118_{(10)}$ в двоично-десятичную систему счисления, необходимо разбить его на тетрады рис. 2.23:

1				**1**				**8**			
0	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**0**
2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0
8	4	2	1	8	4	2	1	8	4	2	1

Рисунок 2.23 – Перевод из десятичной системы счисления в двоично-десятичную систему счисления целого числа $A = 118_{(10)}$.

Таким образом, получаем: $118_{(10)} - 100011000_{(2-10)}$. **Например:** необходимо из двоично-десятичной системы счисления перевести в десятичную систему счисления целое число: $A = 100011000_{(2-10)}$. **Решение:** исходя из вышесказанного, при переводе целого двоично-десятичного числа: $A = 100011000_{(2-10)}$ в десятичную систему счисления, необходимо разбить его на тетрады рис. 2.24:

0	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**0**
2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0
8	4	2	1	8	4	2	1	8	4	2	1
		1				**1**				**8**	

Рисунок 2.24 – Перевод из двоично-десятичной системы счисления в десятичную систему счисления целого числа $A = 100011000_{(2-10)}$

Таким образом, получаем: $100011000_{(2-10)} - 118_{(10)}$

При переводе из недесятичной системы счисления в недесятичную систему счисления например, если необходимо перевести число из пятеричной системы счисления в троичную систему счисления, необходим промежуточный перевод в десятичную систему счисления. **Например:** перевести из пятеричной системы счисления в троичную систему счисления целое число $A = 203_{(5)}$. **Решение:** Исходя из вышесказанного сначала необходимо целое число: $A = 203_{(5)}$ записанное в пятеричной системе счисления, перевести в десятичную систему счисления, а затем полученное число из десятичной системы счисления перевести в троичную систему счисления.

Действие №1: $A = 203_{(5)} - 10$.

$$A = 203_{(5)} = 2 \cdot 5^2 + 0 \cdot 5^1 + 3 \cdot 5^0 = 2 \cdot 25 + 0 \cdot 5 + 3 \cdot 1 = 50 + 0 + 3 = 53_{(10)}.$$

Действие №2: $A = 53_{(10)} - 3$, рис.2.25:

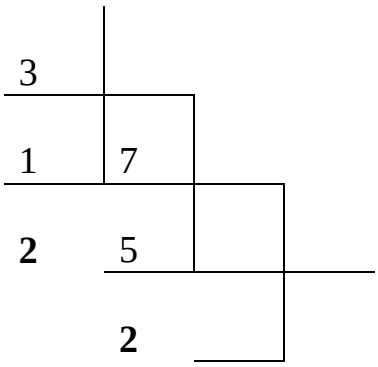


Рисунок 2.25 – Перевод целого числа $A = 53_{(10)}$ из десятичной системы счисления в троичную систему счисления

Таким образом, получаем: $203_{(5)} - 1222_{(3)}$.

Правило перевода правильной дроби из одной позиционной системы счисления в другую. Пусть правильная (p^x – ичная) дробь $A(p^x)$ уже переведена и представлена в новой системе счисления с основанием (p):

$$A(p^x) = a_1 \cdot p^{-1} + a_2 \cdot p^{-2} + a_3 \cdot p^{-3} + ... + a_n \cdot p^{-n}.$$

где a_1 – целая часть первого произведения;

$A(p^{x1}) = a_2 \cdot p^{-1} + a_3 \cdot p^{-2} + a_3 \cdot p^{-3} + ... + a_n \cdot p^{-n+1}$ – дробная часть первого произведения.

Умножив на (p) дробную часть $A(p^{x1})$ первого произведения, определим вторую цифру (a_2):

$$A(p^{x1}) = a_2 + a_3 \cdot p^{-1} + ... + a_n \cdot p^{(-n+2)}.$$

где a_1 – целая часть первого произведения.

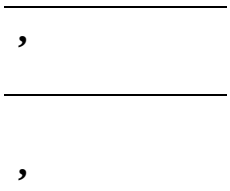
Для перевода правильной дроби из одной позиционной системы счисления в другую ее надо последовательно умножать на основание новой системы счисления (p) до тех пор, пока в новой дроби не будет нужного количества цифр, которое определяется требуемой точностью представления дроби или цифры последнего произведения должны равняться нулю.

Правильная дробь в новой системе счисления записывается из целых частей произведений, получающихся при последовательном умножении, причем первая целая часть будет старшей цифрой новой дроби. **Например**: необходимо перевести из десятичной системы счисления в двоичную систему счисления правильную дробь $A = 0,375_{(10)}$. **Решение:** исходя из вышесказанного произведем последовательное умножение $A = 0,375_{(10)}$ на новое основание системы счисления (2), рис.2.26:

,

Самая старшая цифра-

,



Самая младшая цифра -

Рисунок 2.26 – Перевод из десятичной системы счисления в двоичную систему счисления правильной дроби $A = 0,375_{(10)}$.

Таким образом, получаем: $0,375_{(10)} = 0,011_{(2)}$. **Например**: необходимо перевести из десятичной системы счисления в восьмеричную систему счисления правильную дробь $A = 0,8125_{(10)}$. **Решение:** исходя из вышесказанного произведем последовательное умножение $A = 0,8125_{(10)}$ на новое основание системы счисления (8), рис.2.27:

Самая старшая цифра-

Самая младшая цифра-

Рисунок 2.27 – Перевод из десятичной системы счисления в восьмеричную систему счисления правильной дроби $A = 0,8125_{(10)}$.

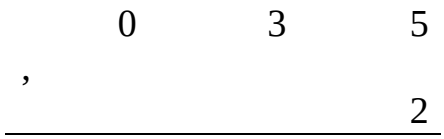
Таким образом, получаем: $0,8125_{(10)} = 0,64_{(8)}$.

Если при переводе правильной дроби из одной позиционной системы счисления в другую при последовательном умножении на основание новой системы счисления (p) цифры последнего произведения не равняются нулю, то процесс умножения заканчивается тогда, когда появляется период и в этом случае говорят: приблизительно равно.

Правильная дробь в новой системе счисления записывается из целых частей произведений, получающихся при последовательном умножении, причем первая целая часть будет старшей цифрой новой дроби.

Например: необходимо перевести из десятичной системы счисления в двоичную систему счисления правильную дробь $A = 0,35_{(10)}$.

Решение: исходя из вышесказанного произведем последовательное умножение $A = 0,35_{(10)}$ на новое основание системы счисления (2), рис.2.28:



Самая старшая цифра-

0	7	0
,		
		2
1	4	0
,		
		2
0	8	0
,		
		2
1	6	0
,		
		2
1	2	0
,		
		2
0	4	0
,		

Самая младшая цифра-

Рисунок 2.28 – Перевод из десятичной системы счисления в двоичную систему счисления правильной дроби $A = 0,35_{(10)}$

последнее произведение получилось равным (0,40), а ранее было уже получено произведение (1,40), следовательно считаем, что начался период.

Таким образом, получаем: $0,35_{(10)} \approx 0,010110_{(2)}$.

Например: необходимо перевести из двоичной системы счисления в десятичную систему счисления правильную дробь $A = 0,0110_{(2)}$.

Решение: в этом случае для преобразования чисел необходимо воспользоваться формулой:

$$A = \sum_{i=-m}^{s-1} a_i \cdot p^i .$$

Запишем число (A) в виде суммы произведений степеней основания на соответствующую цифру в десятичной системе счисления:

$$A = 0,0110_{(2)} = 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 0 \cdot 2^{-4} = 0 \cdot \frac{1}{2} + 1 \cdot \frac{1}{4} + 1 \cdot \frac{1}{8} + 0 \cdot \frac{1}{16} =$$

$$= \frac{1}{4} + \frac{1}{8} = \frac{2}{8} + \frac{1}{8} = \frac{3}{8} = 0,375_{(10)}$$

Таким образом, получаем: $0,0110_{(2)} = 0,375_{(10)}$.

Для перевода неправильной десятичной дроби в систему счисления с недесятичным основанием необходимо отдельно перевести целую часть и отдельно дробную.

Например: необходимо перевести из десятичной системы счисления в двоичную систему счисления неправильную дробь $A = 23{,}125_{(10)}$.

Решение: исходя из вышесказанного необходимо сначала перевести целую часть числа (A) путем последовательного деления на основание новой системы счисления (2) до тех пор, пока не получится частное, у которого целая часть равна (0). При этом число в новой системе счисления записывается из остатков от последовательного деления, причем последний остаток будет старшей цифрой нового числа. А затем перевести дробную часть путем последовательного умножения на основание новой системы счисления (2) до тех пор, пока в новой дроби не будет нужного количества цифр, которое определяется требуемой точностью представления дроби. Тогда дробная часть числа (A) в новой системе счисления записывается из целых частей произведений, получающихся при последовательном умножении, причем первая целая часть будет старшей цифрой новой дроби, рис.2.28:

23	2			
22	11	2		
1	10	5	2	
	1	4	2	2
		1	2	**1**
			0	

0,	1	2	5
			2
0,	2	5	0
			2
0,	5	0	0
			2
1,	0	0	0

Рисунок 2.28 – Перевод из десятичной системы счисления в двоичную систему счисления неправильной дроби $A = 23{,}125_{(10)}$.

Таким образом, получаем: $23{,}125_{(10)} - 10111{,}001_{(2)}$.

Для перевода восьмеричного (шестнадцатеричного) числа в двоичное необходимо каждую цифру исходного числа записать в виде эквивалентного ей трехбитного (четырехбитного) двоичного числа.

Например: необходимо перевести из восьмеричной системы счисления в двоичную систему счисления неправильную дробь $A = 273{,}5_{(8)}$.

Решение: исходя из вышесказанного разобьем каждую цифру неправильной восьмеричной дроби на триады и переведем каждую цифру отдельно:

$273.5_{(8)}$ = 0 1 0 1 1 1 0 1 1, 1 0 $1_{(2)}$ = $010111011{,}101_{(2)}$ или $10111011{,}101_{(2)}$

веса: 4-2-1 4-2-1 4-2-1 4-2-1

Например: необходимо перевести из шестнадцатеричной системы счисления в двоичную систему счисления неправильную дробь $A = 5AE{,}18_{(16)}$.

Решение: исходя из вышесказанного разобьем каждую цифру неправильной шестнадцатеричной дроби на тетрады и переведем каждую цифру отдельно:

$5AE.18_{(16)}$= 0 1 0 1 1 0 1 0 1 1 1 0, 0 0 0 1 1 0 0 0$_{(2)}$ =$10110101110,00011_{(2)}$

веса: 8-4-2-1 8-4-2-1 8-4-2-1 8-4-2-1 8-4-2-1

Все рассмотренные алгоритмы предназначены для программного перевода чисел. Известно также множество алгоритмов перевода, ориентированных на реализацию их аппаратными средствами. Однако изучать такие алгоритмы целесообразно вместе с изучением микропроцессорных систем.

2.2 Представление числовых данных в микропроцессорных системах

Система вещественных чисел, используемая в ручных расчетах, предполагается бесконечной и непрерывной, т.е. отсутствуют ограничения на диапазон используемых чисел и точность (количество значащих цифр).

Пояснение: для любого вещественного числа существует бесконечно много чисел, которые больше или меньше его, и между любыми двумя вещественными числами находится также бесконечно много чисел.

В микропроцессорной технике размеры устройств ограничены. Эти ограничения касаются диапазона чисел и точности их представления. Поэтому система машинных чисел является конечной и дискретной, образуя подмножество системы вещественных чисел.

На рис. 2.29 схематически представлена система микропроцессорных чисел где ($X_{\max}$) и ($X_{\min}$) – соответственно максимально и минимально представимые числа, между которыми находится конечное множество допустимых чисел.

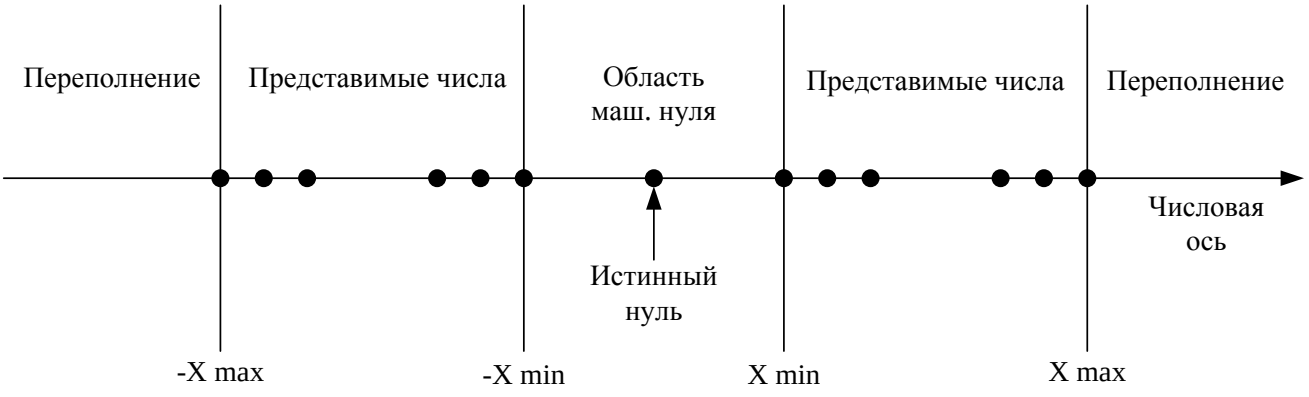


Рисунок 2.29 – Система микропроцессоных чисел

Если результат операции по модулю превышает $|X_{\max}|$, то возникает переполнение (overflow). Если модуль результата $|X| < |X_{\min}|$, фиксируется антипереполнение (underflow).

В этом случае большинство микропроцессоров возвращают в качестве результата операции (0). Поэтому область чисел от ($-X_{\min}$ до $X_{\min}$), за исключением истинного нуля, называют машинным нулем. В современных микропроцессорных системах преимущественно используют две формы представления чисел:

- с фиксированной запятой (фиксированной точкой – fixed point) – естественная форма;
- с плавающей запятой (плавающей точкой – floating point) – альтернативные названия – нормальная, экспоненциальная, научная запись или полулогарифмическая.

Например, пусть задано число:

$$X \Rightarrow x_{s-1}...x_1x_0, x_{-1}x_{-2}...x_{-m}.$$

где: X – некоторое число; $x_{s-1}...x_1x_0$ – цифры слева от запятой; $x_{-1}x_{-2}...x_{-m}$ – цифры справа от запятой.

В позиционной системе счисления с основанием (2) его можно записать в виде:

$$X = M \cdot 2^P.$$

где: X – некоторое число; M – мантисса; 2 – основание системы счисления;

P – порядок.

Если каждому числу (X) в микропроцессорной системе однозначно соответствует мантисса (M), а порядок (P) фиксирован для всех чисел, то число (X) представлено в форме с фиксированной запятой. Порядок (P) в этом случае устанавливается заранее при подготовке задачи к решению, причем число (2^{-p}), по существу, является масштабным коэффициентом, на который необходимо умножить исходные данные для того, чтобы избежать переполнения разрядной сетки.

Естественная форма характеризуется тем, что положение разрядов (и запятой) в микропроцессорном представлении чисел остается всегда постоянным независимо от величин чисел, с которыми оперирует микропроцессорная система. С целью упрощения процедуры определения масштабных коэффициентов запятую обычно фиксируют перед старшим разрядом мантиссы или после ее младшего разряда.

В первом случае микропроцессорная система оперирует с числами, которые по абсолютной величине меньше единицы, то есть, с правильными

дробями. Если количество разрядов для представления мантиссы равно (n), то: ($|X_{\min}| = 2^{-n}, |X_{\max}| = 1 - 2^{-n}$).

Во втором случае микропрцессорная система выполняет операции с целыми числами, абсолютные величины которых находятся в пределах от (1 (минимального, не равного нулю числа) до $2^n - 1$).

Если каждому числу (X) однозначно соответствует пара (M) и (P), то такое представление называют формой с плавающей запятой. Следует иметь ввиду, что положение запятой в реальной разрядной сетке микропроцессорной системы физически никак не указывается, а только "подразумевается" и само число можно записать в виде:

$$X = M_X P_X .$$

$$X = \pm M_X E \pm P_X .$$

где: X – некоторое число; M – мантисса числа (X); E – разделитель мантиссы и порядка числа (X); P – порядок числа (X).

Для однозначного представления числа в нормальной форме принято, что мантисса должна удовлетворять условию:

$$2^{-1} \le |M_X| < 1.$$

то есть мантисса должна быть правильной дробью, а ее старшая цифра – отличной от (0). В таком случае представление числа является нормализованным. Помимо однозначного представления, это дает еще и сохранение максимального количества значащих цифр мантиссы при выполнении операций. Положение разрядов числа в форме с плавающей запятой не является постоянным, то есть запятая как бы плавает.

С учетом рассмотренных ранее структурных единиц информации, в микропроцессорных системах различают следующие форматы: слово, полуслово (байт), двойное слово (doubleword), учетверенное слово (quadword).

В соответствии с перечисленными форматами различают базовые типы целочисленных данных:

- слово – 8 бит;
- целое слово - 16 бит;
- короткое целое слово - 32 бит;
- длинное целое слово - 64 бит.

Для того, чтобы микропроцессорная система могла оперировать как положительными, так и отрицательными числами, в разрядную сетку микропроцессорного представления числа необходимо ввести знаковую часть.

В двоичной системе счисления принято знак положительного числа обозначать «0», а знак отрицательного числа - «1» (обычно это крайний левый бит в представлении числа), как показано на рис.2.30.

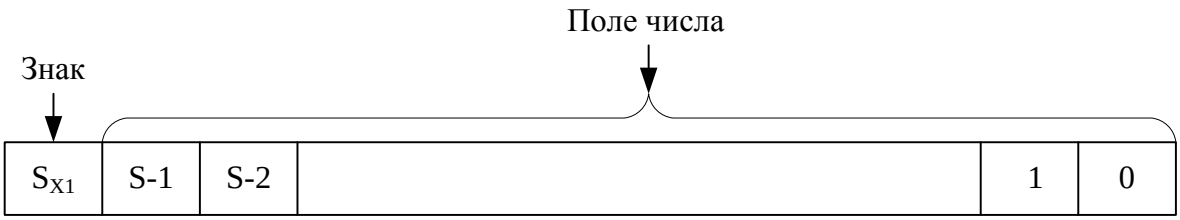


Рисунок 2.30 – Формат числа в двоичной системе счисления со знаковым битом

Будем обозначать целые знаковые числа в виде:

$$S_X, x_{s-1}...x_1x_0.$$

а правильные дроби:

$$S_X, x_{-1}x_{-2}...x_{-m}.$$

В связи с тем, что алгоритмы выполнения операций в микропроцессорных системах имеют свою специфику, возникает проблема представления отрицательных чисел. Ее решают за счет использования особых способов кодирования числовых данных.

Рассмотрим три наиболее распространенных кода, которые применяются на практике:

- прямой код;
- обратный код;
- дополнительный код.

Прямой код используется для представления отрицательных чисел в запоминающем устройстве, а также при умножении и делении.

Обратный и дополнительный коды используются для замены операции вычитания операцией сложения, что упрощает устройство арифметического блока микропроцессорной системы.

К кодам выдвигаются следующие требования:

а) разряды числа в коде жестко связаны с определенной разрядной сеткой;

б) для записи кода знака в разрядной сетке отводится фиксированный, строго определенный разряд.

1. Прямой код.

Наиболее естественный код. Формируется следующим образом: в знаковый бит (S_X) числа помещают знак числа (0 – если число положительное и 1 – если число отрицательное), а остальные биты используют для абсолютного значения (модуля) числа. Правило преобразования чисел в прямом коде можно записать так:

$$[X]_{ПК} = \begin{cases} X, & X \geq 0 \\ A + |X|, & X < 0 \end{cases}.$$

где A – вес старшего разряда в n – разрядной сетке, $A = 2^{s-1}$.

Отображение s-битных наборов двоичных чисел (верхняя числовая прямая) на числовую ось данных прямого кода (нижняя числовая прямая) для микропроцессорной системы показано на рис. 2.31.

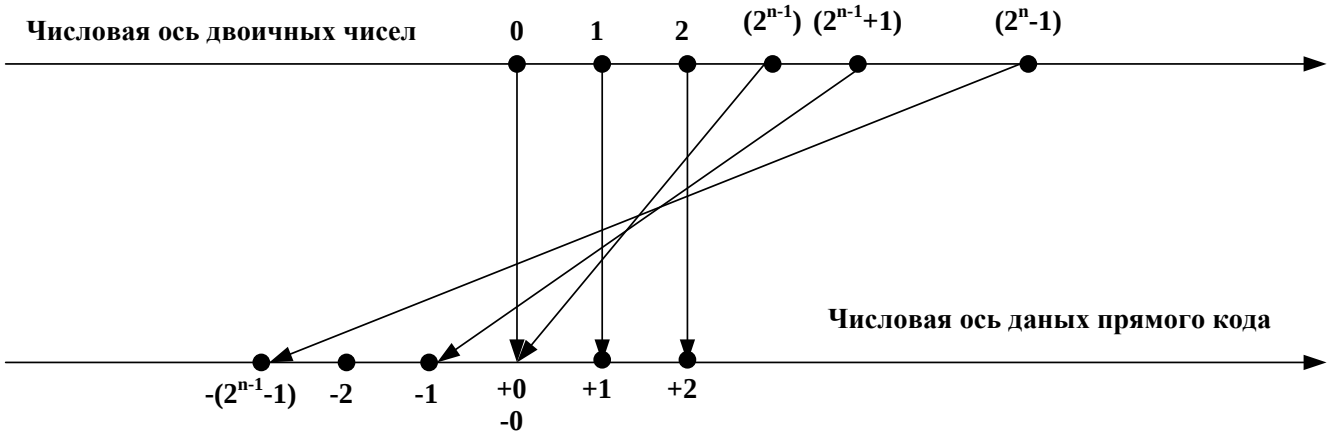


Рисунок 2.31 – Отображение s-битных наборов двоичных чисел на числовую ось данных прямого кода

Например:

Десятичное число	Двоичное число	Прямой код	Комментарии
$(0_{(10)})$	$0000_{(2)}$	0,000	положительный (0)
$(0_{(10)})$	$0000_{(2)}$	1,000	отрицательный (0)
$(4_{(10)})$	$0100_{(2)}$	0,100	положительная (4)
$(-4_{(10)})$	$1100_{(2)}$	1,100	отрицательная (4)
$(3_{(10)})$	$0011_{(2)}$	0,011	положительная (3)
$(-3_{(10)})$	$1011_{(2)}$	1,011	отрицательная (3)

Диапазон представимых чисел: $-(2^{s-1}-1)...(2^s-1)$.

Положительный момент заключается в удобстве операции ввода-вывода данных. Отрицательный момент в том, что существует два представления (0) и операция алгебраического сложения требует анализа знаков и модулей операндов, и выбора фактической операции сложения или вычитания.

2. Обратный код.

Обратный (n-разрядный) двоичный код положительного целого числа состоит из одноразрядного кода знака (0), за которым следует $(n-1)$ разрядное

двоичное представление модуля числа, то есть обратный код для положительного числа совпадает с прямым кодом.

Обратный (n-разрядный) двоичный код отрицательного целого числа состоит из одноразрядного кода знака (1), за которым следует ($n-1$) разрядное двоичное число, представляющее собой инвертированное ($n-1$) разрядное представление модуля числа, то есть для отрицательного числа все цифры числа заменяются на противоположные, а именно (1 на 0, а 0 на 1) и в знаковый разряд заносится единица.

Обратный код числа определяется соотношением:

$$[X]_{OK} = \begin{cases} X, & X \geq 0 \\ B - |X|, & X < 0 \end{cases}.$$

где $B = 2^s - 1 = X_{\max}$ – максимальное число в n-разрядной сетке.

Отображение s-битных наборов двоичных чисел (верхняя числовая прямая) на числовую ось данных обратного кода (нижняя числовая прямая) для микропроцессорной системы показано на рис. 2.32.

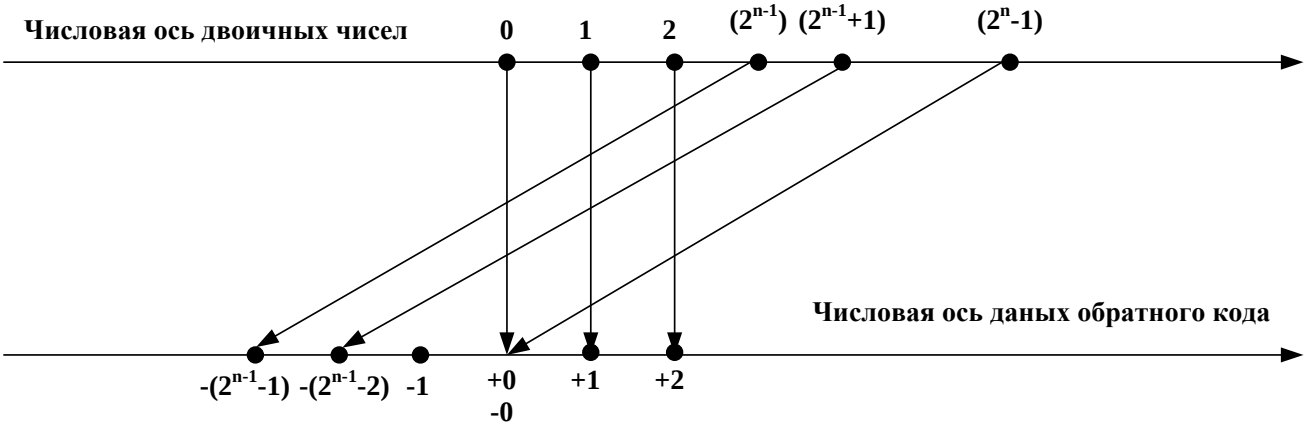


Рисунок 2.32 – Отображение s-битных наборов двоичных чисел на числовую ось данных обратного кода

Например:

Десятичное число	Двоичное число	Обратный код	Комментарии
$(0_{(10)})$	$0000_{(2)}$	0,000	положительный (0)
$(0_{(10)})$	$1111_{(2)}$	1,111	отрицательный (0)
$(4_{(10)})$	$0100_{(2)}$	0,100	положительная (4)
$(-4_{(10)})$	$1011_{(2)}$	1,011	отрицательная (4)
$(3_{(10)})$	$0011_{(2)}$	0,011	положительная (3)
$(-3_{(10)})$	$1100_{(2)}$	1,100	отрицательная (3)

Диапазон представимых чисел: $-(2^{s-1}-1)...(2^{s}-1)$.

Положительный момент заключается в том, что при сложении обратных кодов чисел как беззнаковых чисел получаем обратный код суммы.

Отрицательный момент заключается в том, что существует два представления (0) и при выполнении операций сложения в обратном коде требуется коррекция суммы с помощью циклического переноса единицы переполнения из знаковых разрядов суммы.

3. **Дополнительный код**.

Наиболее распространенный способ представления отрицательных целых чисел в микропроцессорной системе.

Он позволяет заменить операцию вычитания на операцию сложения, чем упрощает архитектуру микропроцессорной системы.

Дополнительный код является дополнением числа до некоторого граничного числа ($|X_{\max}-1|=2^{s}$).

Дополнительный код положительного числа совпадает с прямым кодом.

Для получения дополнительного кода отрицательного числа существует три способа:

- все цифры модуля исходного числа заменяются на взаимно обратные, то есть производится инверсия всех цифр числа, затем к полученному значению добавляется единица в младшем разряде;
- из модуля числа вычитается (1) младший бит, а затем инвертируются все разряды;
- необходимо записать n- битный модуль числа. Затем просматривать число справа налево, сохранить все младшие нули и первую встретившуюся (1), а остальные биты инвертировать.

Дополнительный код числа определяется соотношением:

$$[X]_{ДК}=\begin{cases}X, & X\geq 0\\ X_{обр.}+|X|, & X<0\end{cases}.$$

$$[X]_{ДК}=\begin{cases}X, & X\geq 0\\ C-|X|, & X<0\end{cases}.$$

где $-[X]=C-|X|$, $C=X_{гр.}$ равняется весу не существующего в данном числе разряда, расположенного слева от знаковой цифры.

Отображение s-битных наборов двоичных чисел (верхняя числовая прямая) на числовую ось данных дополнительного кода (нижняя числовая прямая) для микропроцессорной системы показано на рис. 2.33.

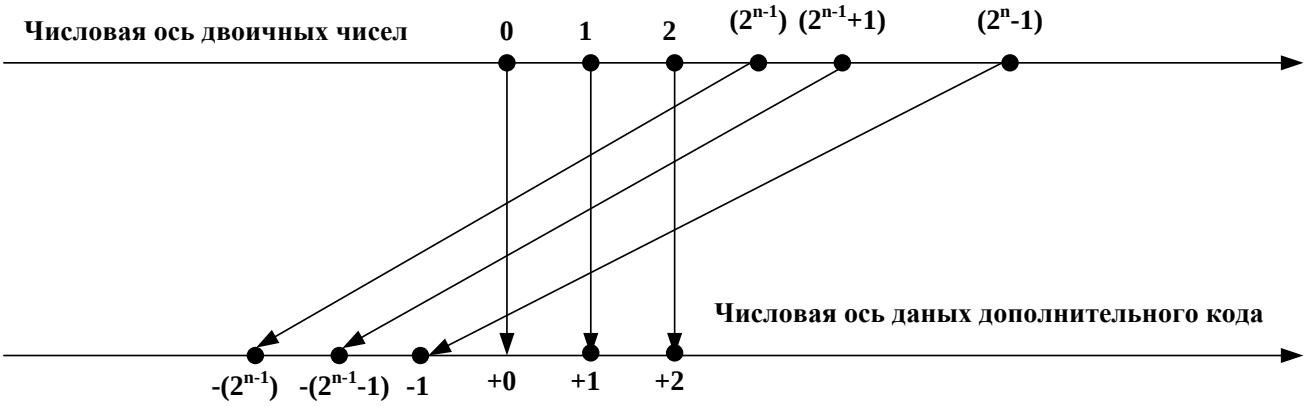


Рисунок 2.33 – Отображение s-битных наборов двоичных чисел на числовую ось данных дополнительного кода

Например:

Десятичное число	Двоичное число	Дополнительный код	Комментарии
$(0_{(10)})$	$0000_{(2)}$	0,000	положительный (0)
$(4_{(10)})$	$0100_{(2)}$	0,100	положительная (4)
$(-4_{(10)})$	$1100_{(2)}$	1,100	отрицательная (4)
$(3_{(10)})$	$0011_{(2)}$	0,011	положительная (3)
$(-3_{(10)})$	$1101_{(2)}$	1,101	отрицательная (3)

Диапазон представимых чисел: $-(2^{s-1}-1)...(2^s-1)$.

Свойства дополнительного кода:

- при сложении чисел в дополнительном коде как беззнаковых чисел получаем дополнительный код суммы. Знаковые биты суммируются обычным образом, а возникающий при их сложении перенос игнорируется;
- любое число в дополнительном коде можно считать младшими битами («хвостом») числа любой длины, если содержимое знакового бита копировать влево. Эта операция называется расширением знака. Используется, когда исходные операнды в операциях сложения и вычитания имеют разную длину.

Свойства кодов:

- операции должны выполняться над данными, представленными в одном и том же коде: ПК-ПК, ОК-ОК, ДК-ДК;
- положительные числа в ПК, ОК, ДК не меняют своего представления;
- результат выполнения операций сложения и вычитания над числами, представленными в ПК, ОК или ДК, являются ПК, ОК или ДК соответственно;

- двоичный набор, представляющий (-0) в ПК и ДК, является запрещенной комбинацией;
- в отличие от ПК, в ОК и ДК нельзя отбрасывать нули после знакового разряда в целой части и нули в конце дробной части отрицательного числа (разрешается отбрасывать 1).

2.2.1 Арифметические флажки

Арифметические флажки являются признаками, представляющими общую характеристику результата выполнения операции. Наиболее широко применяются следующие флажки:

– **флажок переноса (Flag Carry),** обозначаемый (FC).

Устанавливается в (1) в двух случаях:

1. Если при выполнении операции сложения имеет место перенос из старшего бита результата (представляет собой расширение результата на 1 бит влево);
2. Если при выполнении операции вычитания имеет место заем (borrow) в старший бит. Это возможно в случае, если уменьшаемое меньше вычитаемого. Если операнды интерпретируются как беззнаковые числа, данный флажок является признаком переполнения микропроцессорой системы.

В операциях над знаковыми числами самостоятельного значения не имеет.

– **флажок вспомогательного переноса или дополнительный перенос (Flag Auxiliary)**, обозначаемый (FA).

При сложении показывает перенос, а при вычитании - заем из младшей тетрады (бит 3) результата. Используется в операциях двоично-десятичной арифметики.

– **флажок нуля (Flag Zero),** обозначаемый (FZ).

Признак нулевого результата. Устанавливается в (1) когда результат выполнения операции равняется (0).

– **флажок знака (Flag Sign)**, обозначаемый (FS).

Повторяет состояние знакового бита.

– **флажок переполнения (Flag Overflow),** обозначаемый (FO).

В операциях над знаковыми числами показывает, находится ли результат внутри диапазона представимых чисел:

- $(FO) = 0$ – результат правильный;
- $(FO) = 1$ – возникло переполнение. Следует отметить, что (FO) устанавливается в (1), если перенос в старший разряд и из него не совпадают.

2.2.2 Контроль переполнения в микропроцессорных системах

Возможно только при сложении чисел с одинаковыми знаками, когда для представления результата недостаточно отведенного количество разрядов (требуется больше, чем для представления операндов). Это может приводить не только к неправильному (по абсолютной величине) результату, но и к неправильному его знаку. Независимо от кода для представления отрицательных чисел (обратный или дополнительный) переполнение разрядной сетки имеет место всегда, если только при сложении положительных чисел возникает перенос в знаковый разряд, а при сложении отрицательных чисел такой перенос равен нулю. Введение масштабных коэффициентов, на которые необходимо умножить все исходные данные перед решением задачи, позволяет предотвратить переполнение. В микропроцессорных системах предусматривают специальные средства для обнаружения переполнения разрядной сетки:

- программный способ; основан на том, что при сложении чисел с одинаковыми знаками знак суммы должен совпадать со знаками операндов;
- аппаратный способ; основан на использовании модифицированных кодов.

При использовании модифицированных кодов**.** знак числа изображается двумя одинаковыми цифрами. Эти цифры в процессе выполнения операций обрабатываются так же, как и числовые разряды. При этом появление в знаковых разрядах разных цифр (01 при сложении положительных и 10 при сложении отрицательных двоичных операндов) свидетельствует о переполнении разрядной сетки.

Модифицированный код позволяет формировать правильный знак результата даже при наличии переполнения.

Этот знак сохраняется во втором (старшем) знаковом разряде. Такой способ обнаружения переполнения разрядной сетки наиболее распространен в микропроцессорных системах, несмотря на некоторую избыточность в аппаратных средствах для представления чисел, поскольку позволяет оперативно (то есть, одновременно с результатом) получить информацию о переполнении.

На рис.2.34 представлен формат двоичного знакового числа с использованием модифицированного кода.

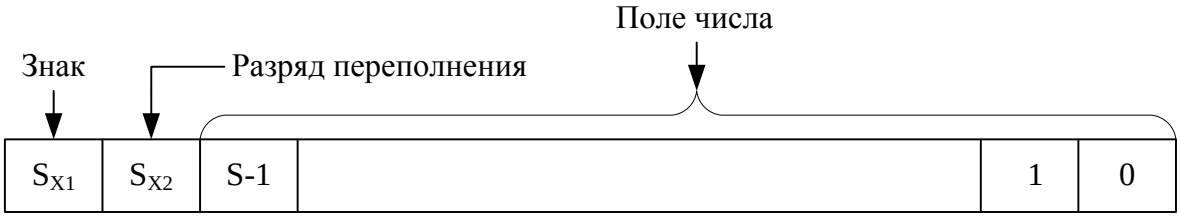


Рисунок 2.34 – Формат двоичного знакового числа с использованием модифицированного кода

При выполнении операций сложения и вычитания чисел в дополнительных кодах обрабатывающее устройство микропроцессора не делает различий между знаковыми и беззнаковыми операндами.

То есть операнды должны интерпретироваться самим пользователем как знаковые или беззнаковые. В операции вычитания используется дополнительный код вычитаемого. Если длина операндов превышает длину машинного слова, то сложение и вычитание выполняют в несколько приемов, организуя программный цикл.

Операцию начинают с младших частей операндов, «продвигаясь» далее в сторону старших частей. На каждом шаге должны обрабатываться возникающие переносы (или заемы).

2.3 Выполнение арифметических операций в микропроцессорных системах над двоичными числами с фиксированной точкой

2.3.1 Операция сложения и вычитания, двоичных беззнаковых чисел в микропроцессорных системах

Микропроцессорная система выполняет сложение и вычитание операндов по правилам сложения и вычитания двоичных беззнаковых чисел рис.2.35:

Правила сложения:	Правила вычитания:
1. 0 + 0 = 0	1. 0 - 0 = 0
2. 0 + 1 = 1	2. 1 - 1 = 0
3. 1 + 0 = 1	3. 1 - 0 = 1
4. 1 + 1 = 10	4. 10 - 1 = 1

Рисунок 2.35 – Правила сложения и вычитания, двоичных беззнаковых чисел

Проблем не возникает до тех пор, пока значение результата не превышает разрядной сетки операнда.

Например: необходимо перевести в двоичную систему счисления, а затем сложить два числа: $7_{(10)}$ и $5_{(10)}$ записанных в десятичной системе счисления (по правилам сложения двоичных беззнаковых чисел). Длина разрядной сетки операндов равна четырем битам.

Используя вышеописанные правила, мы находим сумму двух чисел следующим образом: сначала складываем числа в последнем столбце, записываем младший разряд полученной суммы под столбцом, а старший в следующий слева столбец, и продолжаем сложение, как показано на рис.2.36.

Например: необходимо перевести в двоичную систему счисления, а затем найти разность двух чисел: $9_{(10)}$ и $7_{(10)}$ записанных в десятичной

системе счисления (по правилам вычитания двоичных беззнаковых чисел). Длина разрядной сетки операндов равна четырем битам рис.2.37.

Проблема возникает тогда, когда результат выходит за пределы разрядной сетки операнда, как показано в следующем примере.

Например: необходимо перевести в двоичную систему счисления, а затем найти сумму двух чисел: $9_{(10)}$ и $7_{(10)}$ записанных в десятичной системе счисления (по правилам сложения двоичных беззнаковых чисел). Длина разрядной сетки операндов равна четырем битам рис.2.38.

Десятичное число	Двоичное число
$7_{(10)}$	$0111_{(2)}$
$5_{(10)}$	$0101_{(2)}$
$12_{(10)}$	$1100_{(2)}$

Рисунок 2.36 – Пример вычисления суммы двух чисел: $7_{(10)}$ и $5_{(10)}$ записанных в десятичной системе счисления (по правилам сложения двоичных беззнаковых чисел), при четырехразрядной сетке операндов

Десятичное число	Двоичное число
$9_{(10)}$	$1001_{(2)}$
$7_{(10)}$	$0111_{(2)}$
$2_{(10)}$	$0010_{(2)}$

Рисунок 2.37 – Пример вычисления разности двух чисел: $9_{(10)}$ и $7_{(10)}$ записанных в десятичной системе счисления (по правилам вычитания двоичных беззнаковых чисел), при четырехразрядной сетке операндов

Десятичное число		Двоичное число
$9_{(10)}$		$1001_{(2)}$
$7_{(10)}$		$0111_{(2)}$
$16_{(10)}$	**1**	$0000_{(2)}$

Рисунок 2.38 – Пример вычисления суммы двух чисел: $9_{(10)}$ и $7_{(10)}$ записанных в десятичной системе счисления (по правилам сложения двоичных беззнаковых чисел), при четырехразрядной сетке операндов

В примере, представленном на рис.2.38 видно, что (1) вышла за пределы разрядной сетки операнда.

При обработке полученной суммы она не учитывается и следовательно сумма получилась равной (0), а не (16), что является не верным результатом.

В таких случаях увеличивают разрядную сетку операнда, и результат принимает правильное значение.

Например: необходимо перевести в двоичную систему счисления, а затем найти сумму двух чисел: $9_{(10)}$ и $7_{(10)}$ записанных в десятичной системе счисления (по правилам сложения двоичных беззнаковых чисел). Длина разрядной сетки операндов равна восьми битам рис.2.39:

Десятичное число	Двоичное число
$9_{(10)}$	$00001001_{(2)}$
$7_{(10)}$	$00000111_{(2)}$
$16_{(10)}$	$00010000_{(2)}$

Рисунок 2.39 – Пример вычисления суммы двух чисел: $9_{(10)}$ и $7_{(10)}$ записанных в десятичной системе счисления (по правилам сложения двоичных беззнаковых чисел), при восьмиразрядной сетке операндов

В микропроцессорной системе этот исход сложения прогнозируется и предусмотрены специальные средства для фиксирования подобных ситуаций и их обработки.

Так, для фиксирования ситуации выхода за разрядную сетку результата, как в данном примере, предназначен флаг переноса (FC). Он располагается в бит – (0) регистра флагов eflags. Именно установкой этого флага фиксируется факт переноса (1) из старшего разряда операнда за пределы разрядной сетки.

Естественно, что программист должен предусматривать возможность такого исхода операции сложения и средства для корректировки. Это предполагает включение участков кода после операции сложения, в которых анализируется флаг (FC). Анализ этого флага можно провести различными способами. Самый простой и доступный использовать команду условного перехода. Эта команда в качестве операнда имеет имя метки в текущем

сегменте кода. Переход на эту метку осуществляется в случае если в результате работы предыдущей команды флаг (FC) установился в (1).

Для закрепления материала рекомендуется выполнить следующие упражнения:

1. Перевести в **двоичную** систему счисления и вычислить выражение:
$$\mathbf{15_{(16)} - 15_{(8)}.}$$
2. Перевести в **двоичную** систему счисления и вычислить выражение:
$$\mathbf{21_{(16)} + 21_{(10)}.}$$
3. Перевести в **двоичную** систему счисления и вычислить выражение:
$$\mathbf{(10_{(10)} + 10_{(8)}) - 11_{(16)}.}$$
4. Перевести в **двоичную** систему счисления и вычислить выражение:
$$\mathbf{17_{(16)} - 17_{(8)}.}$$
5. Перевести в **двоичную** систему счисления и вычислить выражение:
$$\mathbf{19_{(16)} + 19_{(10)}.}$$
6. Перевести в **двоичную** систему счисления и вычислить выражение:
$$\mathbf{(21_{(10)} + 21_{(8)}) - 21_{(16)}.}$$

2.3.2 Операция сложения и вычитания двоичных знаковых чисел в микропроцессорных системах

При сложении и вычитании знаковых двоичных чисел операция вычитания заменяется операцией сложения в дополнительном коде.

Докажем, что результат выполнения операций сложения и вычитания над числами, представленными в ДК, являются ПК, ОК или ДК соответственно.

Для этого рассмотрим отдельно формирование числовой и знаковой частей суммы.

Тогда операцию сложения проанализируем над «псевдомодулями» слагаемых - дополнительными кодами, у которых отброшен знаковый разряд.

В этом случае граничное число (C) станет равным весу знакового разряда, который становится отсутствующим.

Представим операцию сложения в общем виде:

$$Z = X + Y.$$

где Z – сумма; X – первое слагаемое; Y – второе слагаемое.

В зависимости от знаков слагаемых возможны следующие варианты:

1. $X > 0; Y > 0$**,** тогда:

$$Z = [X]_{ДК} + [Y]_{ДК} = |X| + |Y| = [X + Y]_{ДК} = [Z]_{ДК}.$$

Например: рассмотрим два случая **А** и **Б**.

Случай А: необходимо вычислить сумму двух чисел записанных в десятичной системе счисления $(5 + 2 = 7)$, для этого необходимо оба числа перевести в двоичную систему счисления, а затем представить в дополнительном коде, для удобства возьмем длину разрядной сетки четыре бита. Формат суммы представлен на рис. 2.40, а вычисление суммы и значения флагов на рис.2.41.

0,	1	1	1
D3	D2	D1	D0

Рисунок 2.40 – Формат суммы

Случай А

Десятичное число	Двоичное число	Значения флагов
$5_{(10)}$	$0{,}101_{(2)}$	FC=0
$2_{(10)}$	$0{,}010_{(2)}$	FS=0
$7_{(10)}$	$0{,}111_{(2)}$	FZ=0
		FO=0

Рисунок 2.41 – Пример вычисления суммы двух чисел и значение флагов после получения результата

Значение флагов:

$FC = 0$, потому, что нет перехода (1) из разряда $D2$ в знаковый разряд $D3$ и нет перехода из знакового разряда $D3$ за пределы разрядной сетки.

$FS = 0$, потому, что значение знакового бита $D3 = 0$.

$FZ = 0$, потому, что сумма не равняется (0).

$FO = 0$, потому, что результат правильный и переполнение разрядной сетки нет.

Случай Б: необходимо сложить два числа записанных в десятичной системе счисления $(7 + 7 = 14)$, для этого необходимо оба числа перевести в двоичную систему счисления, а затем представить в дополнительном коде, для удобства возьмем длину разрядной сетки четыре бита.

Формат суммы представлен на рис. 2.42, а вычисление суммы и значения флагов на рис.2.43.

1,	1	1	0
D3	D2	D1	D0

Рисунок 2.42 – Формат суммы

Случай Б

Десятичное число	Двоичное число	Значения флагов
$7_{(10)}$	$0{,}111_{(2)}$	FC=1
$7_{(10)}$	$0{,}111_{(2)}$	FS=1
$14_{(10)}$	$1{,}110_{(2)}$	FZ=0
		FO=1

Рисунок 2.43 – Пример вычисления суммы двух чисел и значение флагов после получения результата

Значение флагов:

$FC=1$, потому, что есть перехода (1) из разряда $D2$ в знаковый разряд $D3$ и нет перехода из знакового разряда $D3$ за пределы разрядной сетки.

$FS=1$, потому, что значение знакового бита $D3=1$, а это не правильно так как оба слагаемых положительные числа, значит и сумма должна быть положительной, а в данном случае сумма получается отрицательной так как знаковый бит равен (1).

$FZ=0$, потому, что сумма не равняется (0).

$FO=1$, потому, что результат не правильный и переполнение разрядной сетки есть.

2. $X>0; Y<0$ и $|X|>|Y|$, тогда:

$$Z=[X]_{ДК}+[Y]_{ДК}=|X|+(C-|Y|)=C+(|X|-|Y|)=C+\Delta.$$

Тогда (C) эквивалентно переносу в знаковый разряд, т.к. $C+\Delta>X_{\max}$. Тогда:

$$Z = [\Delta]_{ДК}.$$

Например: необходимо сложить два числа записанных в десятичной системе счисления $(5+(-2)=3)$, для этого необходимо оба числа перевести в двоичную систему счисления, а затем представить в дополнительном коде, для удобства возьмем длину разрядной сетки четыре бита.

Формат суммы представлен на рис.2.44, а вычисление суммы и значения флагов на рис.2.45.

1	0,	0	1	1
	D3	D2	D1	D0

Рисунок 2.44 – Формат суммы

Десятичное число	Двоичное число	Значения флагов
$5_{(10)}$	$0{,}101_{(2)}$	FC=0
$(-2)_{(10)}$	$1{,}110_{(2)}$	FS=0
$(3)_{(10)}$	$10{,}011_{(2)}$	FZ=0
		FO=0

Рисунок 2.45 – Пример вычисления суммы двух чисел и значение флагов после получения результата

Значение флагов:

$FC=0$, потому, что есть переход (1) из разряда $D2$ в знаковый разряд $D3$ и есть переход из знакового разряда $D3$ за пределы разрядной сетки.

$FS=0$, потому, что значение знакового бита $D3=0$.

$FZ=0$, потому, что сумма не равняется (0).

$FO=0$, потому, что результат правильный и переполнение разрядной сетки нет. Единица, которая вышла за пределы разрядной сетки не учитывается.

3. $X>0; Y<0$ и $|X|<|Y|$, тогда: $C-|\Delta|$, по сути, представляет собой дополнительный код отрицательной разности:

$$|X|-|Y| \Rightarrow Z=[|X|-|Y|]_{ДК},$$

так как

$$C-\Delta < X_{\max}.$$

Например: необходимо сложить два числа записанных в десятичной системе счисления ($2+(-5)=(-3)$), для этого необходимо оба числа перевести в двоичную систему счисления, а затем представить в дополнительном коде, для удобства возьмем длину разрядной сетки четыре бита.

Формат суммы представлен на рис.2.46, а вычисление суммы и значения флагов на рис.2.47.

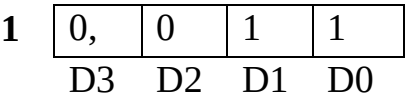


Рисунок 2.46 – Формат суммы

Десятичное число	Двоичное число	Значения флагов
$2_{(10)}$	$0{,}010_{(2)}$	FC=0
$(-5)_{(10)}$	$1{,}011_{(2)}$	FS=1
$(-3)_{(10)}$	$1{,}101_{(2)}$	FZ=0
		FO=0

Рисунок 2.47 – Пример вычисления суммы двух чисел и значение флагов после получения результата

Значение флагов:

$FC=0$, потому, что нет перехода (1) из разряда $D2$ в знаковый разряд $D3$ и нет перехода из знакового разряда $D3$ за пределы разрядной сетки.

$FS=1$, потому, что значение знакового бита $D3=1$.

$FZ=0$, потому, что сумма не равняется (0).

$FO=0$, потому, что результат правильный и переполнение разрядной сетки нет. Число 1,101 и есть (-3). Для того, чтобы перейти к прямому коду необходимо все биты проинвертировать, а затем к младшему биту прибавить (1), рис.2.48:

$1{,}101_{(2)}$	(-3)
$0{,}010_{(2)}$	инверсия
$1_{(2)}$	+1
$0{,}011_{(2)}$	(+3)

Рисунок 2.48 – Пример перехода от дополнительного кода числа к прямому коду

4. $X < 0;\ Y < 0$, тогда:

$$Z = [X]_{ДК} + [Y]_{ДК} = (C - |X|) + (C - |Y|) = C + C - (|X| + |Y|) > X_{\max}.$$

Первое (C) игнорируется, т.к. эквивалентно переносу в знакоый разряд, а $C - (|X| + |Y|)$ – ДК суммы модулей. При условии, что $|X| + |Y| \le X_{\max}$.

Например: рассмотрим два случая а) и б).

Случай А: необходимо сложить два числа записанных в десятичной системе счисления ($(-5) + (-2) = (-7)$), для этого необходимо оба числа перевести в двоичную систему счисления, а затем представить в дополнительном коде, для удобства возьмем длину разрядной сетки четыре бита. Формат суммы представлен на рис.2.49, а вычисление суммы и значения флагов на рис.2.50.

1	1,	0	0	1
	D3	D2	D1	D0

Рисунок 2.49 – Формат суммы

Случай А

Десятичное число	Двоичное число	Значения флагов
$(-5)_{(10)}$	$1{,}011_{(2)}$	FC=0
$(-2)_{(10)}$	$1{,}110_{(2)}$	FS=1
$(-7)_{(10)}$	$11{,}001_{(2)}$	FZ=0
		FO=0

Рисунок 2.50 – Пример вычисления суммы двух чисел и значение флагов после получения результата

Значение флагов:

$FC = 0$, потому, что есть переход (1) из разряда $D2$ в знаковый разряд $D3$ и есть перехода из знакового разряда $D3$ за пределы разрядной сетки.

$FS = 1$, потому, что значение знакового бита $D3 = 1$.

$FZ = 0$, потому, что сумма не равняется (0).

$FO = 0$, потому, что результат правильный и переполнение разрядной сетки нет. Единица, которая вышла за пределы разрядной сетки не учитывается.

Число 1,001 и есть (-7). Для того, чтобы перейти к прямому коду необходимо все биты проинвертировать, а затем к младшему биту прибавить (1), рис.2.51.

Случай Б: необходимо сложить два числа записанных в десятичной системе счисления $((-7)+(-7)=(-14))$, для этого необходимо оба числа перевести в двоичную систему счисления, а затем представить в дополнительном коде, для удобства возьмем длину разрядной сетки четыре бита.

Формат суммы представлен на рис.2.52, а вычисление суммы и значения флагов на рис.2.53.

$1{,}001_{(2)}$	(-7)
$0{,}110_{(2)}$	инверсия
$1_{(2)}$	+1
$0{,}111_{(2)}$	(+7)

Рисунок 2.51 – Пример перехода от дополнительного кода числа к прямому коду

1	0,	0	1	0
	D3	D2	D1	D0

Рисунок 2.52 – Формат суммы

Случай б)

Десятичное число	Двоичное число	Значения флагов
$(-7)_{(10)}$	$1{,}001_{(2)}$	FC=1
$(-7)_{(10)}$	$1{,}001_{(2)}$	FS=0
$(-14)_{(10)}$	$10{,}010_{(2)}$	FZ=0
		FO=1

Рисунок 2.53 – Пример вычисления суммы двух чисел и значение флагов после получения результата

Значение флагов:

$FC = 1$, потому, что нет перехода (1) из разряда $D2$ в знаковый разряд $D3$ и есть переход из знакового разряда $D3$ за пределы разрядной сетки.

$FS = 0$, потому, что значение знакового бита $D3 = 0$, а это не правильно так как оба слагаемых отрицательные числа, значит и сумма должна быть отрицательные, а в данном случае сумма получается положительной так как знаковый бит равен (0).

$FZ = 0$, потому, что сумма не равняется (0).

$FO = 1$, потому, что результат не правильный и переполнение разрядной сетки есть. Единица, которая вышла за пределы разрядной сетки не учитывается.

Из вышесказанного на примере (4^x-разрядной) сетки можно сделать следующие выводы:

- переполнение разрядной сетки возможно, когда оба операнда имеют одинаковые знаки;
- есть переход (1) из разряда $D2$ в знаковый разряд $D3$ и нет перехода из знакового разряда $D3$ за пределы разрядной сетки.
- нет перехода (1) из разряда $D2$ в знаковый разряд $D3$ и есть переход из знакового разряда $D3$ за пределы разрядной сетки.

При возникновении ситуации переполнения, увеличивается количество бит в разрядной сетке и результат становится правильным.

Для закрепления материала рекомендуется выполнить следующие упражнения:

1. Перевести в **двоичную** систему счисления (n=8), вычислить выражение, записать результат в **ДК** и значение флагов (FV,FC,FS,FZ):

$$\mathbf{(30_{(16)} - 15_{(8)}) + 25_{(10)}.}$$

2. Перевести в **двоичную** систему счисления (n=8), вычислить выражение, записать результат в **ДК** и значение флагов (FV,FC,FS,FZ):

$$\mathbf{36_{(8)} + (21_{(10)} - 21_{(16)}).}$$

3. Перевести в **двоичную** систему счисления (n=8), вычислить выражение, записать результат в **ДК** и значение флагов (FV,FC,FS,FZ):

$$\mathbf{(38_{(10)} - 1A_{(16)}) + 12_{(8)}.}$$

4. Перевести в **двоичную** систему счисления (n=8), вычислить выражение, записать результат в **ДК** и значение флагов (FV,FC,FS,FZ):

$$\mathbf{(34_{(16)} - 13_{(8)}) + 23_{(10)}.}$$

5. Перевести в **двоичную** систему счисления (n=8), вычислить выражение, записать результат в **ДК** и значение флагов (FV,FC,FS,FZ):

$$56_{(8)} + (21_{(10)} - 25_{(16)}).$$

6. Перевести в **двоичную** систему счисления (n=8), вычислить выражение, записать результат в **ДК** и значение флагов (FV,FC,FS,FZ):

$$(22_{(10)} - 3A_{(16)}) + 14_{(8)}.$$

2.3.3 Операции сдвига в микропроцессрных системах

Операция сдвига в микропроцессорных системах является одной из самых распространенных в микропрцессорной арифметике. В частности, она используется при выполнении умножения или деления двоичных чисел. Обычно выполняется в регистрах как встроенная микрооперация за счет специальной организации внутрирегистровых связей.

Существует несколько вариантов сдвига:

1. **Логический сдвиг.**

Сдвиг, который выполняется над всеми разрядами операнда, включая знаковый.

2. **Арифметический сдвиг.**

Не влияет на положение знака операнда.

Арифметический сдвиг двоичного операнда влево или вправо на (i) разрядов эквивалентно соответственно умножению и делению исходного операнда на (2^i).

При сдвиге вправо разряды исходного операнда, которые выходят за пределы разрядной сетки регистра, теряются, внося при этом ошибку, которая по абсолютной величине не превышает полученного после сдвига числа.

Сдвиг влево имеет смысл, пока не теряются значащие цифры операнда.

3. **Модифицированный сдвиг.**

Арифметический сдвиг операндов, представленных обратным или дополнительным кодом:

- обратный код – при правом и левом сдвигах в освобождающиеся разряды регистра записываются цифры знакового разряда;
- дополнительный код – при сдвиге вправо в разряды, которые освобождаются, записываются цифры знакового разряда, а при сдвиге влево в освобождающиеся разряды записываются нули.

4. **Циклический сдвиг.**

Операнд перемещается в регистре как в замкнутом контуре, поступая с выхода последнего разряда на вход первого.

Схемы выполнения операций сдвига приведены на рис.2.54.

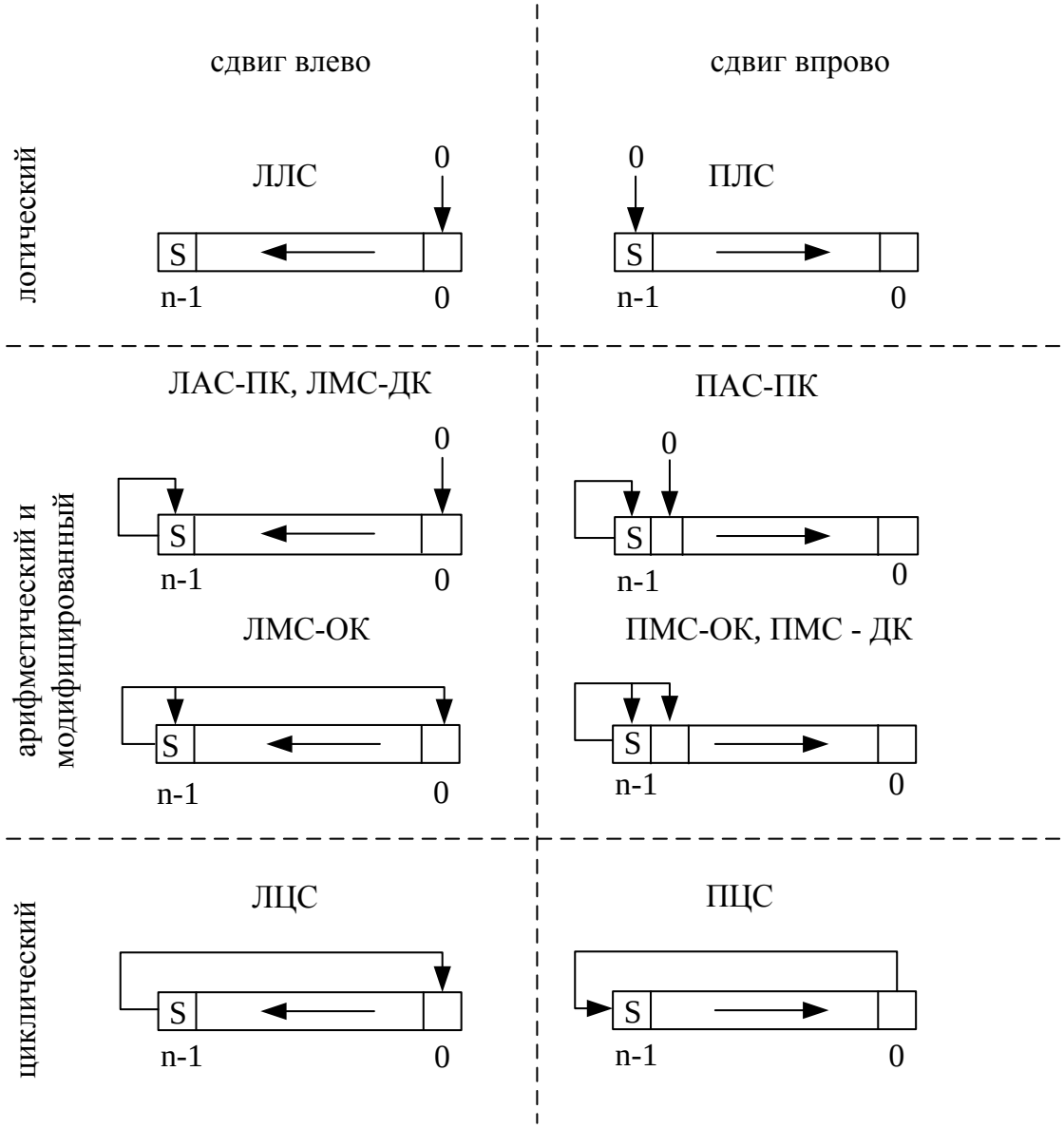


Рисунок 2.54 – Схемы выполнения операций сдвига

2.3.4 Умножение двоичных беззнаковых чисел в микропроцессоных системах

Пусть сомножителями X и Y являются s-битные целые числа без знака: где – (X) – множимое, (Y) – множитель, (Z) – произведение. Тогда:

$$Z = X \cdot Y .$$

где: $X = x_{s-1}x_{s-2}...x_1x_0$; $Y = y_{s-1}y_{s-2}...y_1y_0$.

Представим множитель Y в развернутой форме:

$$Y = y_{s-1} \cdot 2^{s-1} + y_{s-2} \cdot 2^{s-2} + ... + y_1 \cdot 2^1 + y_0 \cdot 2^0.$$

Тогда:

$$Z = X \cdot Y = (X \cdot y_{s-1}) \cdot 2^{s-1} + (X \cdot y_{s-2}) \cdot 2^{s-2} + ... +$$
$$+(X \cdot y_1) \cdot 2^1 + (X \cdot y_0) \cdot 2^0 = \sum_{i=0}^{s-1} X \cdot y_i \cdot 2^i \quad .$$

Произведение $z_i = (X \cdot y_i)2^i$ множимого на один бит множителя называется частичным произведением (ЧП). Полное произведение числа представляет собой сумму (s) частичных произведений (ЧП). При умножении вручную существует два способа:

- ✓ **способ № 1,** начиная с младшей цифры множителя и сдвиг частичных произведений влево;
- ✓ **способ № 2,** начиная со старшей цифры множителя и сдвиг частичных произведений вправо. **Например.** Необходимо перемножить два беззнаковых числа ($7 \cdot 3 = 21$). Для удобства возьмем длину разрядной сетки равную четырем битам. Правило умножения вручную заключается в следующем:
  - шаг 1. Анализируется младший (правый) бит множителя, если младшая цифра множителя равна (1), то в сумму частичных произведений записываются все биты множимого, если младшая цифра множителя равна (0), то в сумму частичных произведений записываются все (0);
  - шаг 2. Анализируется следующий после младшего бит множителя, если цифра множителя равна (1), то в сумму частичных произведений записываются все биты множимого сдвинутые на один разряд влево, если цифра множителя равна (0), то в сумму частичных произведений записываются все (0), сдвинутые на один разряд влево;
  - шаг 3. Шаги (1) и (2) повторяются до тех пор, пока не будут проанализированы все цифры множителя;

– шаг4. Производится сложение всех сумм частичных произведений.

Полученная сумма и будет равна произведению множимого на множитель.

Итак: необходимо перемножить два беззнаковых числа ($7 \cdot 3 = 21$)

где: (7) – множимое; (3) – множитель; (21) – произведение.

Необходимо вычислить сумму частичных произведений – (ЧП), а следовательно и произведение.

Способ №1

Десятичное число	Двоичное число	Комментарии
$7_{(10)}$	$0111_{(2)}$	множимое
$3_{(10)}$	$0011_{(2)}$	множитель
	4321	номера цифр множителя
	0111	1^{oe} – ЧП
	0111	2^{oe} – ЧП
	0000	3^{oe} – ЧП
	0000	4^{oe} – ЧП
$21_{(10)}$	$0010101_{(2)}$	сумма ЧП - произведение

Способ №2

Десятичное число	Двоичное число	Комментарии
$7_{(10)}$	$0111_{(2)}$	множимое
$3_{(10)}$	$0011_{(2)}$	множитель
	1234	номера цифр множителя
	0000	1^{oe} – ЧП
	0000	2^{oe} – ЧП
	0111	3^{oe} – ЧП
	0111	4^{oe} – ЧП
$21_{(10)}$	$0010101_{(2)}$	сумма ЧП - произведение

В отличие от ручного умножения, операционное устройство микропроцессора не может просуммировать сразу (s) частичных произведений, как это делает человек. Обычный сумматор, как правило, рассчитан на сложение только двух операндов. Если нужно получить сумму нескольких слагаемых, то происходит накопление суммы: сначала в сумматор

записывается первое слагаемое, к нему прибавляется второе, затем к полученной сумме прибавляется третье слагаемое и так до получения полной суммы.

Длина произведения s-битных сомножителей равна $2s$ бит:

$$Z = X \cdot Y = X_{\max} \cdot Y_{\max} = (2^s - 1) \cdot (2^s - 1) = 2^{2s} - 2 \cdot 2^s + 1 = 2^{2s} - 2^{s+1} + 1.$$

Поскольку умножение на (2^i) эквивалентно сдвигу влево, то вычисление произведения (Z) сводится к формированию частичных произведений $(X \cdot y_i)$, их сдвигу и суммированию с учетом весов, определяемых величинами (2^i).

$$X \cdot y_i = \begin{cases} X, & y_i = 1 \\ 0, & y_i = 0 \end{cases}.$$

Умножение реализуется циклическим процессом, на каждом шаге которого:

- анализируется очередной бит y_i множителя;
- в зависимости от его значения происходит $(y_i = 1)$ или нет $(y_i = 0)$ прибавление множимого к предыдущей сумме частичных произведений;
- производится изменение взаимного положения множимого (X) и суммы частичных произведений с учетом веса (2^i).

Таким образом, умножение в двоичной системе счисления естественным образом сводится к двум операциям – сложению и сдвигу чисел.

В соответствии со способом формирования суммы частичных произведений – (ЧП), возможны четыре варианта умножения. Они различаются тем, с каких разрядов множителя (Y) (младших или старших) начинается умножение, и что сдвигается (множимое или сумма ЧП).

Варианты умножения, начиная с младших или старших разрядов множителя, называются еще умножением младшими и старшими разрядами вперед соответственно. Схемы выполнения операции умножения двоичных беззнаковых чисел представлены на рис.2.55. При умножении младшими разрядами вперед производятся последовательные сдвиги множителя вправо, вследствие чего в младшем разряде регистра множителя последовательно появляются все его цифры, начиная с младшей. Т.о., в специальной схеме

анализа значения текущей цифры множителя нужно анализировать только состояние младшего разряда соответствующего регистра.

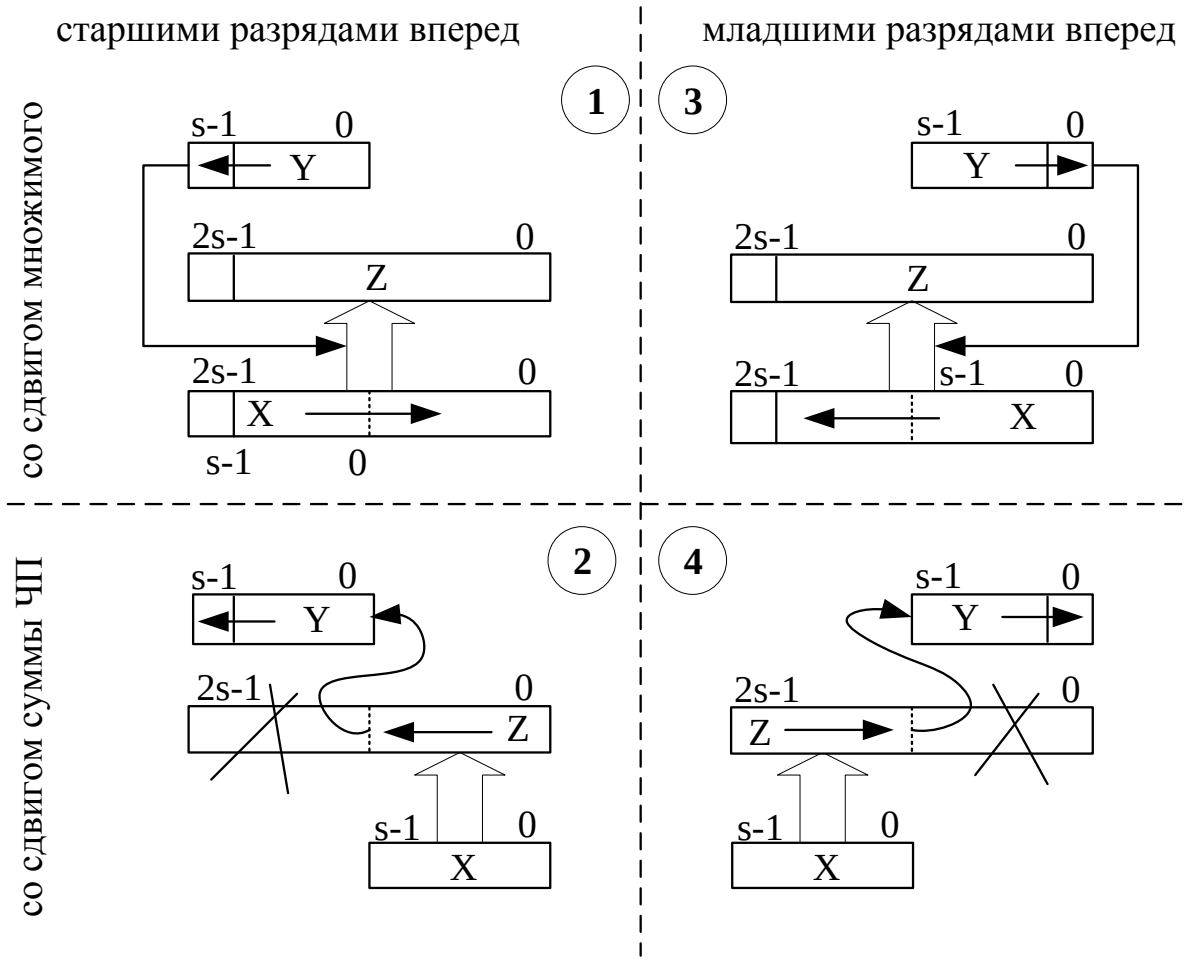


Рисунок 2.55 – Схемы выполнения операции умножения двоичных беззнаковых чисел

Соответственно, при умножении старшими разрядами вперед должен анализироваться старший разряд множителя.

Схема 1

$$Z = (...((0 + X \cdot y_{s-1}2^{s-1}) + X \cdot y_{s-2}2^{s-2}) + ... + X \cdot y_1 2^1) + X \cdot y_0 2^0.$$

$$Z = (...((0 + X \cdot y_{-1}2^{-1}) + X \cdot y_{-2}2^{-2}) + ... + X \cdot y_{-(m-1)}2^{-(m-1)}) + X \cdot y_{-m}2^{-m}.$$

Время выполнения операции умножения:

$$T = (s-1) \cdot (t_c + t_+) + t_+.$$

где (t_c) – время, затрачиваемое на выполнение операции сдвига на один разряд;

(t_+) – время, затрачиваемое на выполнение операции сложения.

Схема 2

$$Z = (...((0 \cdot 2^1 + X \cdot y_{s-1}) \cdot 2^1 + X \cdot y_{s-2}) \cdot 2^1 + ... + X \cdot y_1) \cdot 2^1 + X \cdot y_0.$$

$$Z = (...((0 \cdot 2^1 + X \cdot y_{-1} \cdot 2^{-m}) \cdot 2^1 + X \cdot y_{-2} \cdot 2^{-m}) \cdot 2^1 + ... + X \cdot y_{-(m-1)} \cdot 2^{-m}) \cdot 2^1 + X \cdot y_{-m} \cdot 2^{-m}.$$

$$T = (s-1) \cdot (t_c + t_+) + t_+.$$

Схема 3

$$Z = (...((0 + X \cdot y_0 2^0) + X \cdot y_1 2^1) + ... + X \cdot y_{s-2} 2^{s-2}) + X \cdot y_{s-1} 2^{s-1}.$$

$$Z = (...((0 + X \cdot y_{-m} 2^{-m}) + X \cdot y_{-(m-1)} 2^{-(m-1)}) + ... + X \cdot y_{-2} 2^{-2}) + X \cdot y_{-1} 2^{-1}.$$

$$T = (s-1) \cdot (t_c + t_+) + t_+.$$

Схема 4

$$Z = (...((0 + X \cdot y_0 \cdot 2^s) \cdot 2^{-1} + X \cdot y_1 \cdot 2^s) \cdot 2^{-1} + ... + X \cdot y_{s-1} \cdot 2^s) \cdot 2^{-1}.$$

$$Z = ((...((0 + X \cdot y_{-m}) \cdot 2^{-1} + X \cdot y_{-(m-1)}) \cdot 2^{-1} + ... + X \cdot y_{-2}) \cdot 2^{-1} + X \cdot y_{-1}) \cdot 2^{-1}.$$

$$T = s \cdot (t_c + t_+).$$

Рассмотрим на примере два базовых алгоритма умножения в микропроцессорных системах двоичных беззнаковых чисел:

Алгоритм №1.

Алгоритм умножения младшими разрядами вперед, со сдвигом суммы ЧП вправо.

1. Исходное значение суммы (ЧП) принимается равным (0), счетчику тактов – (Сч.Т) присваивается значение, равное числу разрядов множителя.
2. Анализируется младшая разрядная цифра множителя. Если она равна (1), то к сумме (ЧП) прибавляется множимое, совмещенное по старшим разрядам; если (0) – прибавление не производится.
3. Производится сдвиг множителя и суммы ЧП вправо на (1) разряд. Содержимое (Сч.Т) уменьшается на (1).
4. Анализируется содержимое (Сч.Т). Если оно не равно (0), то переход к (п.2), иначе – (п.5).
5. Умножение закончено, младшая часть произведения находится на месте множителя, а старшая – на месте суммы (ЧП). Например: необходимо перемножить два беззнаковых числа ($7 \cdot 3 = 21$). Для удобства возьмем длину разрядной сетки равную четырем битам, а именно: $X = 7$ – множимое, $Y = 3$ – множитель, $Z = 21$ – произведение. Если (X) и (Y) равняется четырем битам, то как было отмечено выше (Z) должно быть восьмиразрядным значением, т.е длина разрядной сетки произведения в два раза больше множимого и множителя. Алгоритм умножения приведен в табл. 2.6.

Таблица 2.6 – Алгоритм умножения со сдвигом вправо двоичных беззнаковых чисел

Регистр (B) множимое X	Регистр (C) множитель Y	Регистр (A) произведение Z	Счетчик тактов (Сч.Т)	Комментарии
0 1 1 1	0 0 1 1	00000000	4	
		0111		множимое
		01110000		$1^{я}$ СЧП
	0 0 1	00111000	3	$1^{ыи}$сдвиг СЧП
		0111		множимое
		10101000		$2^{я}$ СЧП
	0 0	01010100	2	$2^{ои}$сдвиг СЧП
	0	00101010	1	$3^{ии}$сдвиг СЧП
		00010101	0	$4^{ыи}$сдвиг СЧП
СТОП				

Алгоритм №2.

Алгоритм умножения старшими разрядами вперед, со сдвигом суммы ЧП влево.

1. Исходное значение суммы (ЧП) принимается равным (0), (Сч.Т) присваивается значение, равное числу разрядов множителя.
2. Производится сдвиг суммы (ЧП) влево на (1) разряд.
3. Анализируется старшая разрядная цифра множителя. Если она равна (1), то к сумме (ЧП) прибавляется множимое, совмещенное по младшим разрядам; если (0) – прибавление не производится.
4. Производится сдвиг множителя влево на (1) разряд. Содержимое (Сч.Т) уменьшается на (1).
5. Анализируется содержимое (Сч.Т). Если оно не равно (0), то переход к (п.2), иначе – (п.6).
6. Умножение закончено, произведения находится на месте суммы (ЧП), которая имеет удвоенную разрядность. Например: необходимо перемножить два беззнаковых числа ($7 \cdot 3 = 21$). Для удобства возьмем длину разрядной сетки равную четырем битам, а именно: $X = 7$ – множимое, $Y = 3$ – множитель, $Z = 21$ – произведение. Если (X) и (Y) равняется четырем битам, то как было отмечено выше (Z) должно быть восьмиразрядным значением, т.е длина разрядной сетки произведения в два раза больше множимого и множителя.

Алгоритм умножения приведен в табл. 2.7.

Таблица 2.7 – Алгоритм умножения со сдвигом влево двоичных беззнаковых чисел

Регистр (B) множимое X	Регистр (C) множитель Y	Регистр (A) произведение Z	Счетчик тактов (Сч.Т)	Комментарии
0 1 1 1	0 0 1 1	00000000	4	
		00000000		1ый сдвиг СЧП
	0 1 1	00000000	3	2 ои сдвиг СЧП
		00000000		3 ии сдвиг СЧП
	1 1	00000000	2	4ый сдвиг СЧП
		0111		множимое
		00000111		1я СЧП
	1	00001110	1	5ый сдвиг СЧП
		0111		множимое
		00010101	0	2я СЧП

		СТОП		

Для закрепления материала рекомендуется выполнить следующие упражнения:

1. Перевести в **двоичную** систему счисления и вычислить выражение:
((15$_{(10)}$ – 10$_{(10)}$) * (18$_{(10)}$ – 11$_{(10)}$)).
2. Перевести в **двоичную** систему счисления и вычислить выражение:
((21$_{(10)}$ – 15$_{(10)}$) * (18$_{(10)}$ – 11$_{(10)}$)).
3. Перевести в **двоичную** систему счисления и вычислить выражение:
((19$_{(10)}$ – 15$_{(10)}$) * (18$_{(10)}$ – 11$_{(10)}$)).
4. Перевести в **двоичную** систему счисления и вычислить выражение:
((18$_{(10)}$ – 11$_{(10)}$) * (18$_{(10)}$ – 14$_{(10)}$)).
5. Перевести в **двоичную** систему счисления и вычислить выражение:
((23$_{(10)}$ – 17$_{(10)}$) * (21$_{(10)}$ – 15$_{(10)}$)).
6. Перевести в **двоичную** систему счисления и вычислить выражение:
((14$_{(10)}$ – 11$_{(10)}$) * (18$_{(10)}$ – 10$_{(10)}$)).

2.3.5 Деление двоичных беззнаковых чисел в микропроцессорных системах

Деление мантисс чисел в форме с фиксированной запятой выполняется над абсолютными величинами операндов, представленными, чаще всего, прямым кодом.

Подобно умножению, деление в микропроцессорных системах реализуется как многошаговый процесс выполнения операций суммирования, сдвига и других элементарных операций.

В отличие от умножения этот процесс имеет итеративный характер, так как результат деления в большинстве случаев не может быть точно представлен числом конечной длины.

Как правило, формальным признаком окончания операции деления принимается количество сдвигов: при достижении числа сдвигов, равного количеству разрядов в частном, операция деления завершается.

Обозначим X – делимое, Y – делитель, а Z – частное. Тогда:

$$Z = \frac{X}{Y}.$$

Пусть (X), (Y) и (Z) являются беззнаковыми числами.

Операция деления характеризуется также дополнительным результатом (R) – остатком.

В практической реализации выделяют два типа операции деления:

- первый тип это когда делимое, делитель и частное имеют одну и ту же длину (s);
- второй тип это когда делимое имеет длину $(2s)$ – удвоенную по сравнению с делителем и частным. Рассмотрим случай 1.

$$X = x_{2s-1}x_{2s-2}...x_1x_0,\ X < 2^{2s}.$$

$$Y = y_{s-1}y_{s-2}...y_1y_0,\ Y < 2^{s}.$$

$$Z = z_{s-1}z_{s-2}...z_1z_0,\ Z < 2^{s}.$$

$$R = r_{s-1}r_{s-2}...r_1r_0,\ R < 2^{s}.$$

$$\frac{X}{Y} = Z(R).$$

Для того, чтобы деление было корректным, т.е. чтобы частное не превысило разрядную сетку, необходимо обеспечить выполнение условия:

$$(|X| - |Y|') < 0, \quad |Y|' = |Y| \cdot 2^{s}.$$

или

$$(|X|' - |Y|) < 0, \quad |X|' = |X| \cdot 2^{-s}.$$

В противном случае будет иметь место переполнение разрядной сетки.

Переполнение исключено, если делимое и делитель имеют одинаковую длину. Как особый случай переполнения рассматривают попытку деления на нуль.

По существу, деление сводится к последовательности вычитаний делителя вначале из делимого, а затем из остатков.

Цифра (z_{s-i+1}) частного определяется следующим образом: если текущий остаток (R_i) больше или равен делителю, цифра частного равна (1), если меньше, то цифра частного равна (0).

При этом операция сравнения реализуется посредством операции вычитания.

Так как частное можно определить только со старших разрядов, существует два варианта деления представленных на рис.2.56:

- с неподвижным делимым (частичным остатком) и сдвигаемым вправо делителем;
- с неподвижным делителем и сдвигаемым влево делимым (частичным остатком).

В зависимости от способа обработки отрицательного частичного остатка, различают два алгоритма деления:

- алгоритм №1, с восстановлением остатка;
- алгоритм №2 без восстановления остатка.

Деление с восстановлением остатка заключается в следующем, если на очередном шаге получен положительный остаток, то в очередную цифру частного записывается (1), а остаток становится «предыдущим» для следующего шага.

Данный шаг на этом заканчивается.

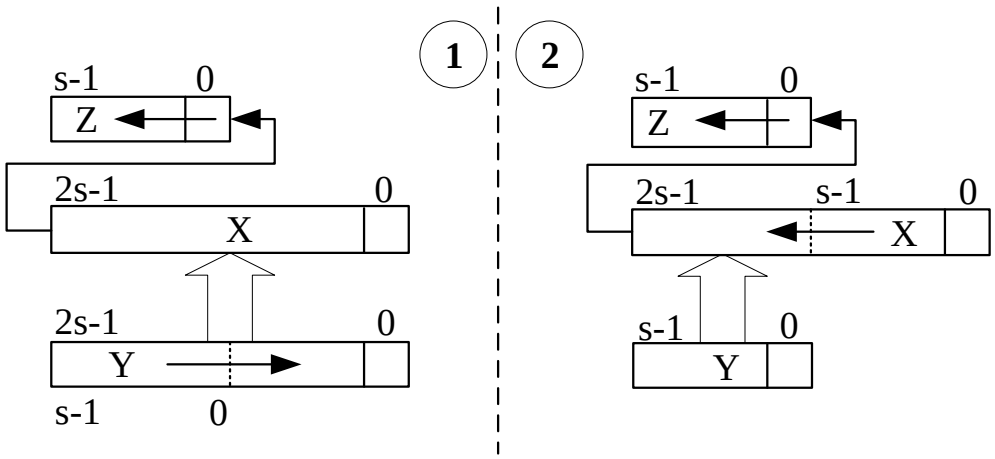


Рисунок 2.56 – Схемы деления двоичных беззнаковых чисел

Если текущий остаток отрицателен, то в очередную цифру частного записывается (0), а к остатку прибавляется делитель для восстановления предыдущего, сдвинутого влево остатка, который становится «предыдущим» для следующего шага.

Таким образом, отрицательный остаток аннулируется, поскольку он выполнил свою функцию сигнализатора о соотношении модулей остатка и делителя и больше не нужен.

Алгоритм №1.

Деление целых двоичных беззнаковых чисел методом с восстановлением остатка.

1. Исходное значение частного (Z) полагается равным (0). (Сч.Т) присваивается значение (s). Исходное значение частичного остатка (R_0) полагается равным (s) старшим разрядам делимого.
2. Выполняется пробное вычитание делителя (Y) из исходного значения частичного остатка – $(ЧО)$. Положительная разность указывает на то, что частное превысит (s-разрядную сетку), и будет выполнено прерывание. Если же результат вычитания отрицательный, то деление можно выполнять.
3. Восстанавливается исходное значение $(ЧО)$ прибавлением делителя к полученному отрицательному остатку.
4. $(ЧО)$ удваивается путем сдвига на один разряд влево.
5. Из $(ЧО)$ вычитается делитель и анализируется знак результата вычитания: если остаток положительный, то очередная цифра частного равна (1), иначе – (0).
6. Частное сдвигается влево с занесением очередной полученной цифры частного в освободившийся младший разряд. (Сч.Т) уменьшается на (1).
7. Если полученный остаток отрицателен, то восстанавливается предыдущий положительный остаток прибавлением делителя к отрицательному остатку.
8. (Пп. 4 – 7) последовательно выполняются до получения всех цифр частного, пока (Сч.Т) не станет равным (0).
9. Если последний остаток от деления отрицателен, то восстанавливается предыдущий положительный остаток, который будет окончательным остатком от деления.

Недостатки:

- нерегулярность выполнения операций, что усложняет устройство управления (для получения одной цифры частного необходимо выполнять либо одно вычитание и сдвиг, либо одно вычитание, одно сложение и сдвиг);
- относительно малая скорость деления, т.к. в среднем половина шагов будет содержать операцию восстановления остатка.

$T = (s+1)(1,5t_+ + t_c) - t_c$.

Например: необходимо разделить два беззнаковых числа $(21:7=3)$.

Для удобства возьмем длину разрядной сетки равную четырем битам, а именно:

$X = 21$ – делимое;

$Y = 7$ – делитель;

$Z = 3$ – частное.

Если (Z) и (Y) равняется четырем битам, то как было отмечено выше (X) должно быть восьмиразрядным значением, т.е длина разрядной сетки делимого в два раза больше делителя и частного.

Алгоритм деления приведен в табл. 2.8.

Таблица 2.8 – Алгоритм деление целых двоичных беззнаковых чисел методом с восстановлением остатка.

	Регистр (B) делимое X								Регистр (C) делитель Y				Регистр (A) частное Z				Счетчик тактов (Сч.Т)
	0	0	0	1	0	1	0	1	0	1	1	1	0	0	1	1	4
	1	0	0	1													
	1	**0**	**1**	**0**	<0												
	0	1	1	1													
1	**0**	**0**	**0**	**1**													
	0	0	1	0	1	0	1										
	1	0	0	1													
	1	**0**	**1**	**1**	<0												3
	0	1	1	1													
1	**0**	**0**	**1**	**0**													
	0	1	0	1	0	1											
	1	0	0	1													
	1	**1**	**1**	**0**	<0												2
	0	1	1	1													
1	**0**	**1**	**0**	**1**													
	1	0	1	0	1												
	1	0	0	1													
1	0	0	1	1	>0												1
	0	1	1	1													
	1	0	0	1													
1	**0**	**0**	**0**	**0**	>0												0
СТОП																	

Деления без восстановления остатка заключается в следующем, исходной предпосылкой для использования данного метода является желание избавиться от процедуры восстановления остатка.

Благодаря этому, длительность деления можно существенно уменьшить по сравнению с вышеприведенной оценкой (за счет исключения "лишней" операции восстановления остатка), используя алгоритм деления без восстановления остатка.

Проанализируем шаг деления, при котором текущий остаток оказался отрицательным. Обозначим:

- (R_{i-1}) – предыдущий остаток;
- (R_i) – текущий;
- (R_{i+1}) – последующий.

Пусть:

$$R_i = 2(R_{i-1}) - |Y| < 0 .$$

При этом, в соответствии с правилом деления, должны выполняться три действия:

- восстановление предыдущего (сдвинутого влево) остатка:

$$R_i + |Y| = 2R_{i-1} - |Y| + |Y| = 2R_{i-1}.$$

- сдвиг восстановленного остатка влево на один разряд:

$$2 \cdot 2R_{i-1} = 4R_{i-1} .$$

- вычитание модуля делителя из полученной кодовой комбинации:

$$R_{i+1} = 4R_{i-1} - |Y| .$$

Таким образом, требуется перейти от текущего остатка:

$$R_i = 2(R_{i-1}) - |Y| .$$

к последующему виду $R_{i+1} = 4R_{i-1} - |Y|$, избавившись от операции восстановления.

Если первым действием является сдвиг отрицательного остатка (R_i) влево, то:

$$2R_i = 4R_{i-1} - 2|Y|.$$

Правая часть отличается от требуемого вида величиной $|Y|$.

Поэтому вторым действием будет корректировка:

$$\Delta = R_{i+1} - 2R_i = 4R_{i-1} - |Y| - 4R_{i-1} + 2|Y| = |Y| \Rightarrow R_{i+1} = 2R_i + |Y|.$$

Сформулируем общее правило.

Чтобы определить цифру частного в некотором разряде, необходимо сдвинуть логически (R_{i-1}) влево на один разряд, а затем прибавить к нему код делителя, которому приписывается знак, противоположный знаку предыдущего остатка; если полученный (R_i) положительный, то в частном проставляется (1), если же отрицательный, – то (0).

Алгоритм №2.

Деления целых двоичных чисел методом без восстановления остатка

1. 1 – 2. Аналогично п.п. 1, 2 предыдущего.
2. Аналогично п. 4.
3. Из частичного остатка вычитается делитель, если остаток положительный, или к частичному остатку прибавляется делитель, если остаток отрицательный.
4. Частное сдвигается; в младший разряд заносится очередная цифра частного (1 – при положительном остатке, 0 – при отрицательном);
5. П.п. 3-5 последовательно выполняются для получения всех цифр частного, пока (Сч.Т) не станет равен (0).
6. Аналогично последнему пункту предыдущего алгоритма.

Реализация данного алгоритма не требует никаких дополнительных аппаратурных затрат. $T = (s+1)(t_+ + t_c)$.

Например: необходимо разделить два беззнаковых числа $(21:7=3)$.

Для удобства возьмем длину разрядной сетки равную четырем битам, а именно:

$X = 21$ – делимое;

$Y = 7$ – делитель;

$Z = 3$ – частное.

Если (Z) и (Y) равняется четырем битам, то как было отмечено выше (X) должно быть восьмиразрядным значением, т.е длина разрядной сетки делимого в два раза больше делителя и частного.

Алгоритм деления приведен в табл. 2.9.

Таблица 2.9 – Алгоритм деление целых двоичных беззнаковых чисел методом без восстановлением остатка.

	Регистр (B) делимое X								Регистр (C) делитель Y				Регистр (A) частное Z				Счетчик тактов (Сч.Т)
	0	0	0	1	0	1	0	1	0	1	1	1	0	0	1	1	4
	1	0	0	1													
	1	**0**	**1**	**0**	<0												
	0	1	0	0	1	0	1										
	0	1	1	1													
	1	**0**	**1**	**1**	<0												3
	0	1	1	1	0	1											
	0	1	1	1													
	1	**1**	**1**	**0**	<0												2
	1	1	0	0	1												
	0	1	1	1													
1	**0**	**0**	**1**	**1**	>0												1
	0	1	1	1													
	1	0	0	1													
1	**0**	**0**	**0**	**0**	>0												0
СТОП																	

Например: необходимо разделить два беззнаковых числа $(21:7=3)$.

Для удобства возьмем длину разрядной сетки равную четырем битам, а именно:

$X=20$ – делимое;

$Y=7$ – делитель;

$Z=2$ – частное;

$R=6$ – остаток.

Если (Z) и (Y) равняется четырем битам, то как было отмечено выше (X) должно быть восьмиразрядным значением, т.е длина разрядной сетки делимого в два раза больше делителя и частного.

Как было отмечено выше последний частичный остаток должен быть больше нуля.

Если последний частичный остаток меньше нуля, его необходимо восстанавливать путем прибавления к нему делителя.

Алгоритм деления приведен в табл. 2.10.

Таблица 2.10 – Алгоритм деление целых двоичных беззнаковых чисел методом без восстановлением остатка.

	Регистр (B) делимое X				Регистр (C) делитель Y				Регистр (A) частное Z				Счетчик тактов (Сч.Т)
	0	0	0	1	0	1	0	0	0	1	1	1	0	0	1	0	4
	1	0	0	1													
	1	**0**	**1**	**0**	<0												
	0	1	0	0	1	0	0										
	0	1	1	1													
	1	**0**	**1**	**1**	<0												3
	0	1	1	1	0	0											
	0	1	1	1													
	1	**1**	**1**	**0**	<0												2
	1	1	0	0	0												
	0	1	1	1													
1	**0**	**0**	**1**	**1**	>0												1
	0	1	1	0													
	1	0	0	1													
	1	**1**	**1**	**1**	<0												0
СТОП																	
Счетчик тактов равняется нулю деление окончено, но последний частичный остаток получился равен меньше нуля, поэтому необходимо выполнить коррекцию, путем прибавления к последнему частичному остатку делитель.																	
	0	1	1	1	– коррекция												
1	**0**	**1**	**1**	**0**	>0 – истинный остаток.												

Для закрепления материала рекомендуется выполнить следующие упражнения:

1. Перевести в **<u>двоичную</u>** систему счисления и вычислить выражение:

$$\mathbf{((15_{(10)} - 10_{(10)}) * (18_{(10)} - 11_{(10)}))/5_{(10)}.}$$

2. Перевести в **двоичную** систему счисления и вычислить выражение:
$$((21_{(10)} - 15_{(10)}) * (18_{(10)} - 11_{(10)}))/6_{(10)}.$$
3. Перевести в **двоичную** систему счисления и вычислить выражение:
$$((19_{(10)} - 15_{(10)}) * (18_{(10)} - 11_{(10)}))/7_{(10)}.$$
4. Перевести в **двоичную** систему счисления и вычислить выражение:
$$((18_{(10)} - 11_{(10)}) * (18_{(10)} - 14_{(10)}))/4_{(10)}.$$
5. Перевести в **двоичную** систему счисления и вычислить выражение:
$$((23_{(10)} - 17_{(10)}) * (21_{(10)} - 15_{(10)}))/6_{(10)}.$$
6. Перевести в **двоичную** систему счисления и вычислить выражение:
$$((14_{(10)} - 11_{(10)}) * (18_{(10)} - 10_{(10)}))/3_{(10)}.$$

2.3.6 Умножение двоичных знаковых чисел в микропроцессорных системах

При выполнении операции умножения знаковых чисел исходные сомножители могут быть представлены в ПК, ОК или ДК:

$$(\pm Z) = (\pm X) \cdot (\pm Y).$$

При данном способе умножения знаковые и числовые разряды обрабатываются отдельно. Для определения знака произведения осуществляют суммирование (по модулю 2) цифр, записанных в знаковых разрядах операндов. Модуль произведения получают, перемножая модули сомножителей в соответствии с одним из рассмотренных ранее алгоритмов:

$$|Z| = |X| \cdot |Y|$$
$$S_Z = S_X \oplus S_Y.$$

Определение знака произведения показано в табл. 2.11. Где (0) – обозначает (+), а (1) – обозначает (–).

Таблица 2.11 – Определение знака произведения

$0 \oplus 0 = 0$
$0 \oplus 1 = 1$
$1 \oplus 0 = 1$
$1 \oplus 1 = 0$

Однако в современных микропроцессорах операнды в памяти обычно хранятся в ДК, что продиктовано, в частности, удобством выполнения операции сложения (учитывая автоматическое формирование знака суммы).

Для того, чтобы умножить два числа, представленных в ДК можно использовать следующий прием: перед выполнением операции умножения исходные числа преобразуют в ПК, затем их перемножают, а результат далее переводят в ДК.

Такой прием не является эффективным. Поэтому на практике чаще всего используют специальные алгоритмы умножения чисел в дополнительных кодах. Алгоритмы корректного умножения операндов в ДК можно разделить на две группы:

- алгоритмы первой группы это алгоритмы с обработкой знаковых разрядов отдельно от числовых;
- алгоритмы второй группы это алгоритмы с обработкой знаковых разрядов вместе с числовыми.

Т.е., на сумматоре дополнительных кодов в процессе перемножения машинных изображений операндов получают одновременно знаковую и числовую части произведения.

Теорема: произведение дополнительных кодов сомножителей равно дополнительному коду результата только в случае положительных сомножителей. Если хотя бы один сомножитель отрицателен, то произведение чисел на сумматоре дополнительных кодов получается прибавлением поправки (Δ) к произведению дополнительных кодов сомножителей. Рассмотрим арифметическое обоснование алгоритмов **первой группы** при различных вариантах комбинаций знаков операндов. С этой целью у операндов отбрасывается знаковая цифра и выполняется умножение полученных псевдомодулей по одному из известных алгоритмов. Результат такого умножения назовем псевдопроизведением (Z'). Обозначим вес знакового разряда сомножителей ($A = 2^{s-1}$). Тогда, в соответствии с рассмотренным ранее правилом формирования дополнительного кода, отрицательные сомножители будут представлены в ДК следующим образом:

$$\begin{matrix} [X]_{ДК} = C - |X| \\ [Y]_{ДК} = C - |Y| \end{matrix}.$$

При удалении знакового разряда получаем псевдомодули:

$$\begin{matrix} [X]'_{ДК} = C - |X| - A = A - |X| \\ [Y]'_{ДК} = C - |Y| - A = A - |Y| \end{matrix}.$$

Всего возможны четыре варианта сочетания знаков сомножителей:

Случай №1. $X > 0; Y > 0$.

$$Z' = [X]'_{ДК} \cdot [Y]'_{ДК} = |X| \cdot |Y| = [|X| \cdot |Y|]_{ДК} = Z.$$

В этом случае сразу получено истинное значение положительного произведения в ДК, которое не требует корректирования.

Случай №2. $X > 0; Y < 0$.

$$Z' = [X]'_{ДК} \cdot [Y]'_{ДК} = |X| \cdot (A - |Y|) = A \cdot |X| - |X| \cdot |Y|.$$

Результат значительно отличается от модуля кода истинного произведения:

$$Z = A^2 - |X| \cdot |Y|.$$

$$Z - Z' = (A^2 - |X| \cdot |Y|) - (A \cdot |X| - |X| \cdot |Y|) = \\ = (A^2 - A \cdot |X|) = A \cdot (A - |X|) = \Delta_2$$

Сомножитель $(A - |X|)$ представляет собой $(s-1)$ – разрядный псевдомодуль числа $(-X)$, представленного в ДК. Сомножитель A эквивалентен сдвигу $[X]'_{ДК}$ на $(s-1)$ разряд влево. Сравнение (Z) и (Z') показывает, что результат умножения должен быть скорректирован посредством сложения (Z') с поправкой (Δ_2).

Случай №3. $X < 0; Y > 0$.

$$Z' = [X]_{ДК} \cdot [Y]_{ДК} = (A - |X|) \cdot |Y| = A \cdot |Y| - |X| \cdot |Y|,$$

$$\Delta_3 = Z - Z' = (A^2 - |X| \cdot |Y|) - (A \cdot |Y| - |X| \cdot |Y|) = (A^2 - A \cdot |Y|) = A \cdot (A - |Y|).$$

Случай №4. $X < 0; Y < 0$.

$$Z' = [X]'_{ДК} \cdot [Y]'_{ДК} = (A - |X|) \cdot (A - |Y|) =$$
$$= A \cdot A - A \cdot |X| - A \cdot |Y| + |X| \cdot |Y|$$
$$Z = |X| \cdot |Y| \Rightarrow \Delta_4 = Z - Z' = \cancel{|X| \cdot |Y|} - A^2 +$$
$$+ A \cdot |X| + A \cdot |Y| - \cancel{|X| \cdot |Y|} = A \cdot (|X| + |Y| - A)$$

На практике используют упрощенную поправку $\Delta_4 = A \cdot (|X| + |Y|)$, поскольку нескорректированный операнд (C) проявит себя в виде (1) в знаковом разряде. Это не имеет значения, поскольку знак (Z) был сформирован отдельно на начальном этапе. Подход к формированию алгоритма умножения состоит в следующем.

1. Получим псевдомодули операндов, отбросив старшие (знаковые) цифры: (0) – в положительном числе, (1) – в отрицательном.

2. Считая разрядную сетку наращиваемой, видим, что операнд в старших разрядах содержит незначащие цифры, тождественно равные знаковой.

3. В таком случае истинные операнды можем считать «удлиненными псевдомодулями», у которых самые левые цифры играют роль знаковых разрядов.

4. Выполняя их умножение по правилам, рассмотренным выше, получаем уже не модуль (Z'), а само псевдопроизведение с некоторыми псевдознаковыми цифрами в двух самых старших разрядах. Указанные цифры относятся к знаковым по расположению, но не имеют смысла таковых.

5. Возникает задача найти истинное значение знаковой цифры числа (Z).

Рассмотрим арифметическое обоснование алгоритмов **второй группы** при различных вариантах комбинаций знаков операндов, т.е. умножение чисел в ДК с обработкой знаковых разрядов вместе с числовыми в зависимости от сочетания знаков сомножителей. При этом используем предпосылки, приведенные в предыдущем доказательстве. Считаем, что вес знакового разряда равен (A). Тогда вес разряда, следующего за знаковым: $C = 2A = 2^s$.

Случай №1. $X > 0; Y > 0$.

Перемножая микропроцессорные представления сомножителей в ДК, получим:

$$Z = [X]_{ДК} \cdot [Y]_{ДК} = |X| \cdot |Y| = Z; \ \Delta_1 = 0.$$

Случай №2. $X > 0; Y < 0$.

Учитывая, что разрядность произведения (и псевдопроизведения) составит $(2s)$ – бит, вес разряда, следующего за знаковым разрядом произведения:

$$C_Z = 2^{2s} = C^2 = 4 \cdot A^2.$$

Тогда:

$$[Y]_{ДК} = C - |Y|.$$

Перемножая ДК сомножителей, получим:

$$Z' = [X]_{ДК} \cdot [Y]_{ДК} = |X| \cdot (C - |Y|)) = C|X| - |X| \cdot |Y|.$$

Истинное значение произведения получается:

$$Z = C^2 - |X| \cdot |Y|.$$

$$\Delta_2 = Z - Z' = C^2 - |X| \cdot |Y| - C \cdot |X| + |X| \cdot |Y| = C^2 - C \cdot |X| = C \cdot (C - |X|).$$

Случай №3. $X < 0; Y > 0$.

Аналогично предыдущему можно показать, что данному варианту требуется коррекция:

$$\Delta_3 = C \cdot (C - |Y|).$$

Случай №4. $X < 0; Y < 0$.

$$Z' = [X]_{ДК} \cdot [Y]_{ДК} = (C - |X|) \cdot (C - |Y|) = C^2 - C \cdot |X| - C \cdot |Y| + |X| \cdot |Y|$$

$$Z = |X| \cdot |Y| \Rightarrow \Delta_4 = Z - Z' = \cancel{|X| \cdot |Y|} - C^2 + C \cdot |X| + C \cdot |Y| - \cancel{|X| \cdot |Y|} =$$

$$= C \cdot (|X| + |Y| - C)$$

На практике используют упрощенную поправку:

$$\Delta_4 = C \cdot (|X| + |Y|).$$

При умножении в ДК по любому из вариантов данной группы в процессе прибавления (Δ) к (Z') важно соблюдать соответствие весов разрядов псевдопроизведения и корректирующей поправки (с учетом наличия псевдознака). Старший разряд корректирующей поправки сформирует знаковую цифру. Из рассмотренных ранее вариантов умножения в соответствии с алгоритмами обеих групп, следует, что для коррекции результата умножения ДК операндов необходимо в каждом случае по комбинации знаков определять величину коррекции, а затем выполнять 1 – 2 шага суммирования. Однако этого можно избежать, если коррекцию совмещать с процессом суммирования частичных произведений. Рассмотренные алгоритмы являются базовыми и реализуют непосредственный способ введения поправок. Более широкое применение **на практике получили алгоритмы с косвенным способом введения поправок.**

Случай №1. $X > 0; Y > 0$.

По примеру первой и второй групп результат является истинным произведением и корректировка не требуется. Т.е. сформированы истинные модуль и знак.

Случай №2. $X > 0; Y < 0$.

При умножении младшими разрядами вперед получится известный для этого случая модуль псевдопроизведения. Если при умножении на знаковую цифру множителя не прибавлять последнее ЧП к их сумме, а вычесть его, тем самым, (Z') будет уменьшено на $(C|X|)$. Кроме того, если для представления последнего ЧП воспользоваться ДК, то при его прибавлении не только будет откорректирована числовая часть произведения, но и сформируется правильный его знак. При этом, последний сдвиг ЧП должен быть арифметическим с учетом отрицательного знака ЧП.

Случай №3 $X < 0; Y > 0$.

Если сдвиг суммы ЧП вправо на (i) разрядов выполнять по правилам сдвига модифицированного ДК, а затем выполнять суммирование ЧП по правилам суммирования модифицированных ДК (с потерей переноса из знакового разряда), то будет получено правильное произведение исходных чисел в ДК, т.е. $(Z' = Z)$. Т.о., коррекция не требуется.

Случай №4 $X < 0; Y < 0$.

Коррекция результата выполняется объединением двух предыдущих вариантов, т.е. при умножении на знаковый разряд множителя выполняют вычитание, а суммирование и сдвиг частичных произведений производят с использованием МДК. Таким образом при положительном множителе умножение в ДК можно выполнять по алгоритмам умножения в ПК, если только суммировать ЧП и сдвиг выполнять по правилам сложения и сдвига модифицированного ДК (перенос из ЗР будет игнорироваться). Существует еще один способ умножения знаковых чисел в ДК. Он заключается в том, что отрицательный множитель необходимо преобразовать в положительный. Это выполняется умножением на (-1) множимого и множителя. В итоге, в

соответствии с выше приведенной теоремой, умножение выполняется по обычным правилам, а результат не нуждается в корректировке. Общий алгоритм умножения целых двоичных знаковых чисел, представленных в ДК заключается в следующем:

1. Исходное значение суммы ЧП принимается равным (0), (Сч.Т) присваивается значение, равное числу разрядов множителя.

2. Анализируется младшая разрядная цифра множителя. Если она равна (1), то к сумме ЧП прибавляется множимое; если (0) – прибавление не производится. Множимое при этом совмещается с суммой ЧП по старшим разрядам и представлено в исходном коде.

3. Производится арифметический сдвиг суммы ЧП вправо на (1) разряд с учетом флагов переноса (FC) и переполнения (FO). Содержимое (Сч.Т) уменьшается на (1).

4. пп.2-3 последовательно выполняются для всех цифровых разрядов множителя.

5. Если множитель – положительное число, то полученный результат является истинным произведением (Z). Если множитель – отрицательное число, то к полученному результату (Z') прибавляется множимое с обратным знаком (дополнение), совмещенное по старшим разрядам. Полученная сумма представляет собой истинное произведение (Z). $T=(s+1)\cdot(t_c+t_+)$. Рассмотрим примеры вышеописанного алгоритма.

Пример №1. Необходимо перемножить два знаковых числа:

$((+7)(+3)=(+21))$. Для удобства возьмем длину разрядной сетки равную четырем битам, а именно: $X=(+7)$ – множимое, $Y=(+3)$ – множитель, $Z=(+21)$ – произведение. Если (X) и (Y) равняется четырем битам, то как было отмечено выше (Z) должно быть восьмиразрядным значением, т.е длина разрядной сетки произведения в два раза больше множимого и множителя. Алгоритм умножения приведен в табл. 2.12.

Таблица 2.12 – Алгоритм умножения со сдвигом вправо двоичных знаковых чисел

Регистр (B) множимое X	Регистр (C) множитель Y	Регистр (A) произведение Z	Счетчик тактов (Сч.Т)	Комментарии
0 1 1 1	0 0 1 1	00000000	4	
		0111		множимое
		01110000		$1^{я}$ СЧП
	0 0 1	00111000	3	$1^{ыи}$сдвиг СЧП
		0111		множимое
		10101000		$2^{я}$ СЧП

			0	0	01010100	2	2 ОИсдвиг СЧП
				0	00101010	1	3 ИИсдвиг СЧП
					00010101	0	4ЫИсдвиг СЧП и получили число – (+21)
СТОП							

Пример №2. Необходимо перемножить два знаковых числа:

$((+7)(-3)=(-21))$.

Для удобства возьмем длину разрядной сетки равную четырем битам, а именно:

$X=(+7)$ – множимое;

$Y=(-3)$ – множитель;

$Z=(-21)$ – произведение.

Если (X) и (Y) равняется четырем битам, то как было отмечено выше (Z) должно быть восьмиразрядным значением.

То есть длина разрядной сетки произведения в два раза больше множимого и множителя.

Алгоритм умножения приведен в табл. 2.13.

Таблица 2.13 – Алгоритм умножения со сдвигом вправо двоичных знаковых чисел

Регистр (B) множимое X				Регистр (C) множитель Y				Регистр (A) произведение Z	Счетчик тактов (Сч.Т)	Комментарии
0	1	1	1	1	1	0	1	00000000	4	
								0111		множимое
								01110000		1Я СЧП
					1	1	0	00111000	3	1ЫИсдвиг СЧП
						1	1	00011100	2	2 ОИсдвиг СЧП
								0111		множимое
								10001100		2Я СЧП
							1	01000110	1	3 ИИсдвиг СЧП
								0111		множимое
								10110110		3Я СЧП
								01011011	0	4ЫИсдвиг СЧП и (Z')

		1001		кор-ция на ($X_Д$)
		11101011		число (-21)
СТОП				

Пример №3. Необходимо перемножить два знаковых числа:

$((-7)((+3)=(-21))$.

Для удобства возьмем длину разрядной сетки равную четырем битам, а именно:

$X=(-7)$ – множимое;

$Y=(+3)$ – множитель;

$Z=(-21)$ – произведение.

Если (X) и (Y) равняется четырем битам, то как было отмечено выше (Z) должно быть восьмиразрядным значением.

То есть длина разрядной сетки произведения в два раза больше множимого и множителя.

Алгоритм умножения приведен в табл. 2.14.

Таблица 2.14 – Алгоритм умножения со сдвигом вправо двоичных знаковых чисел

Регистр (B) множимое X	Регистр (C) множитель Y	Регистр (A) произведение Z	Счетчик тактов (Сч.Т)	Комментарии
1 0 0 1	0 0 1 1	00000000	4	
		1001		множимое
		10010000		$1^{я}$ СЧП
	0 0 1	01001000	3	$1^{ыи}$сдвиг СЧП
		1001		множимое
		11011000		$2^{я}$ СЧП
	0 0	01101100	2	$2^{ои}$сдвиг СЧП
	0	00110110	1	$3^{ии}$сдвиг СЧП
		00011011	0	$4^{ыи}$сдвиг СЧП и ($Z^{'}$)
		1101		кор-ция на ($Y_Д$)
		11101011		число (-21)
СТОП				

Пример №4. Необходимо перемножить два знаковых числа:

$((-7)((-3) = (+21))$.

Для удобства возьмем длину разрядной сетки равную четырем битам, а именно:

$X = (-7)$ – множимое;

$Y = (-3)$ – множитель;

$Z = (+21)$ – произведение.

Если (X) и (Y) равняется четырем битам, то как было отмечено выше (Z) должно быть восьмиразрядным значением.

То есть длина разрядной сетки произведения в два раза больше множимого и множителя. Алгоритм умножения приведен в табл. 2.15.

Таблица 2.15 – Алгоритм умножения со сдвигом вправо двоичных знаковых чисел

Регистр (B) множимое X	Регистр (C) множитель Y	Регистр (A) произведение Z	Счетчик тактов (Сч.Т)	Комментарии
1 0 0 1	1 1 0 1	00000000	4	
		1001		множимое
		10010000		$1^{я}$ СЧП
	1 1 0	01001000	3	$1^{ыи}$сдвиг СЧП
	1 1	00100100		$2^{ои}$сдвиг СЧП
		1001		множимое
		10110100		$2^{я}$ СЧП
	1	01011010	2	$3^{ии}$сдвиг СЧП
		1001		множимое
		11101010		$3^{я}$ СЧП
		01110101	0	$4^{ыи}$сдвиг СЧП и $(Z^{'})$
		0111		кор-ция на $(X_Д)$
		11100101		
		0011		кор-ция на $(Y_Д)$
		00010101		число (+21)
СТОП				

Для закрепления материала рекомендуется выполнить следующие упражнения:

1. Перевести в **<u>двоичную</u>** систему счисления и вычислить выражение:

$((15_{(10)} - 11_{(10)}) * (20_{(10)} - 17_{(10)}))$.

2. Перевести в **двоичную** систему счисления и вычислить выражение:

$((15_{(10)} - 11_{(10)}) * (12_{(10)} - 17_{(10)}))$.

3. Перевести в **двоичную** систему счисления и вычислить выражение:

$((14_{(10)} - 21_{(10)}) * (18_{(10)} - 16_{(10)}))$.

4. Перевести в **двоичную** систему счисления и вычислить выражение:

$((14_{(10)} - 21_{(10)}) * (10_{(10)} - 16_{(10)}))$.

5. Перевести в **двоичную** систему счисления и вычислить выражение:

$((17_{(10)} - 21_{(10)}) * (11_{(10)} - 17_{(10)}))$.

2.3.7 Деление двоичных знаковых чисел в микропроцессорных системах

Так как данные в памяти микропроцессора хранятся в ДК, операцию деления целесообразно выполнять в ДК.

За основу можно принять базовый алгоритм деления (без восстановления остатка), т.к. он предполагает использование отрицательного делителя, дополнительного кода для вычитания и учет знаков остатка и делителя на каждом шаге деления.

Делимое участвует в операции только на первом шаге, а далее используются остатки, которые могут быть как положительными, так и отрицательными. Поэтому не имеет смысла рассматривать алгоритмы деления в ДК, где знаки операндов обрабатываются отдельно от их числовых частей.

Таким образом, при делении чисел в ДК трудности, связанные с коррекцией результата (как при умножении в дополнительном коде), практически отсутствуют.

Пусть:

$$X = S_X x_{2s-2} x_{2s-3} \ldots x_1 x_0 - \text{разрядность } (2s), - |X| \le 2^{2s-1}.$$

$$Y = S_Y y_{s-2} y_{s-3} \ldots y_1 y_0 - \text{разрядность } (s), - |Y| \le 2^{s-1}.$$

$$Z = S_Z z_{s-2} z_{s-3} \ldots z_1 z_0 - \text{разрядность } (s), - |Z| \le 2^{s-1}.$$

$$R = S_R r_{s-2} r_{s-3} \ldots r_1 r_0 - \text{разрядность } (s), - |R| < |Y|.$$

Для проверки на корректность операции деления используется условие: $(|X| - |Y|') < 0$. Таким образом, при пробном вычитании делитель должен быть сдвинут на $(s-1)$ разряд влево. Однако чтобы обеспечить регулярность процесса деления, делитель сдвигается влево на s разрядов, а перед пробным вычитанием делимое сдвигается на один разряд влево.

В каждом цикле сдвинутый остаток складывается с делителем, которому приписывается знак, противоположный знаку частичного остатка.

Алгоритм деления знаковых чисел отличается от алгоритма деления беззнаковых чисел способом формирования разрядных цифр частного.

Очередная цифра определяется на основе анализа знаков частичного остатка и делителя. Если знак полученного остатка совпадает со знаком делителя, то цифра частного равна (1), если нет – (0). При этом частное формируется в прямом или обратном коде (в зависимости от соотношения знаков делимого и делителя). На этапе пробного вычитания автоматически формируется старшая цифра частного, которая представляет его знак. Если знак полученного частичного остатка совпадает со знаком делителя, участвующего в операции пробного вычитания, деление является корректным. В противном случае имеет место переполнение разрядной сетки микропроцессорной системы.

Алгоритм деления целых двоичных знаковых чисел заключается в следующем:

1. Частному присваивается значение (0); (Сч.Т) – (s). Исходное значение частичного остатка равно (s) старшим разрядам делимого.

2. Частичный остаток удваивается сдвигом влево на (1) разряд, с занесением в младший разряд очередной цифры делимого.

3. Если частичный остаток и делитель разного знака, то они складываются, если же одного знака, то из частичного остатка вычитается делитель.

4. Частное сдвигается влево на (1) разряд. В освобождающийся младший разряд заносится очередная цифра частного: (1) – если знак делителя и остатка совпадают, (0) – в противном случае (Сч.Т) уменьшается на (1).

5. Пункты 2 – 4 повторяются до тех пор, пока (Сч.Т) не станет равным (0).

6. Частное и остаток сформированы в обратном коде. Если знак окончательного остатка не совпадает со знаком делимого, то выполняется его восстановление. Для получения ДК результата выполняется его коррекция в зависимости от соотношения знаков делимого и делителя.

При делении в дополнительном коде возможные комбинации знаков операндов образуют следующий перечень.

Случай №1 $X > 0; Y > 0$. Деление в этом случае ничем не отличается от деления положительных операндов в прямом коде. Коррекция не требуется;

Случай №2 $X > 0; Y < 0$. Этот случай легко сводится к обычному делению в прямом коде, если считать, что в наличии имеется $(-Y)$ и $(+Y)$.

Однако для формирования правильного знака частного и обратного кода отрицательного результата необходимо псевдознаковую цифру и цифры частного получать равными цифрам знаковых разрядов соответствующих остатков. Переполнение разрядной сетки здесь фиксируется по $(z_s = 0)$. В конце деления для образования дополнительного кода и округления частного следует прибавить единицу к младшему разряду: $(Z = Z' + 1)$;

Случай №3 $X > 0; Y > 0$. Из отрицательного делимого должен вычитаться положительный делитель, в отличие от деления без восстановления

остатка в прямом коде, когда на первом шаге $(X > 0; Y < 0)$. Поэтому псевдознаковая цифра частного ($z_S = 0$), полученная по знаку нулевого остатка в соответствии с алгоритмом деления без восстановления остатка, здесь указывает на переполнение разрядной сетки. Если же переполнения разрядной сетки нет, эта цифра является правильной знаковой цифрой частного (то есть, $z_S = 1$).

Знаки всех других остатков в этом случае определяют инверсные значения цифр частного.

В результате будет записан обратный код отрицательного частного.

Для округления и получения дополнительного кода результата необходимо добавить единицу в младший разряд, если полученный остаток не равен нулю: $(Z = Z' + 1)$.

Случай№4 $(X < 0; Y < 0)$.

На первом шаге алгоритма для формирования правильного знака частного ($z_S = 0$), необходимо из отрицательного делимого вычитать отрицательный делитель ($z_S = 1$ соответствует переполнению разрядной сетки). Далее же этот случай сводится к третьему.

Здесь все цифры частного, включая псевдознаковую, равны знакам остатков.

В случае, когда полученный остаток равен нулю, к частному следует прибавить единицу $(Z = Z' + 1)$.

При использовании методов с восстановлением остатка и без восстановления остатка сдвиг влево может вызвать передачу значащей цифры из старшего разряда остатка в знаковый, и остаток будет искажен.

Поэтому надо оперировать двумя знаковыми разрядами.

Правильный знак при этом находится в дополнительном знаковом разряде, а единица в основном относится к разряду переполнения (в случае правильных дробей - к разряду целых).

Знак остатка от деления должен совпадать со знаком делимого.

Например: случай №1 $X > 0; Y > 0; Z > 0$.

Необходимо разделить два знаковых числа:

$X = +21$ – делимое;

$Y = +7$ – делитель;

$Z = +3$ – частное.

$((+21) : (+7) = (+3))$.

Для удобства возьмем длину разрядной сетки равную четырем битам, а именно: если (Z) и (Y) равняется четырем битам, то как было отмечено выше (X) должно быть восьмиразрядным значением, т.е длина разрядной сетки делимого в два раза больше делителя и частного. На первом шаге разрядная сетка делителя увеличивается на один разряд путем арифметического сдвига вправо. Далее, когда знаки делимого и делителя совпадают, то на втором шаге производится вычитание, когда знаки делимого и делителя не совпадают, то на втором шаге производится сложение. Затем анализируются знаковые биты

частичного остатка и делителя, если они разные, то в частное записывается (0), если одинаковые, то в частное записывается (1). Далее анализируются знаки частичного остатка и делителя, если они разные, то после сдвига влево производится сложение частичного остатка с делителем, если одинаковые, то производится вычитание из частичного остатка делителя (или сложение с дополнительным кодом делителя).

Алгоритм деления приведен в табл. 2.16.

Таблица 2.16 – Алгоритм деление целых двоичных знаковых чисел методом без восстановления остатка.

	Регистр (B) делимое X								Регистр (C) делитель Y				Регистр (A) частное Z				Счетчик тактов (Сч.Т)
	0	0	0	1	0	1	0	1	0	1	1	1	0	0	1	1	4
(-)	1	1	0	0	1				0	0	1	1	1				
	1	**1**	**0**	**1**	**1**												3
	0	1	1	1	0	1											
(+)	0	1	1	1													
	1	**1**	**1**	**0**													2
	1	1	0	0	1												
(+)	0	1	1	1													
1	**0**	**0**	**1**	**1**													1
	0	1	1	1													
(-)	1	0	0	1													
1	**0**	**0**	**0**	**0**													0
СТОП																	

Случай №2 $X > 0; Y < 0; Z < 0$.

Необходимо разделить два знаковых числа:

$X = +21$ – делимое;

$Y = -7$ – делитель;

$Z = -3$ – частное.

$((+21):(-7) = (-3))$.

В этом случае знаки делимого и делителя не совпадают, поэтому на втором шаге производится сложение делимого и отрицательного делителя, а также необходима коррекция частного, т.е. после получения частного к последнему биту частного необходимо прибавить (1).

Алгоритм деления приведен в табл. 2.17.

Таблица 2.17 – Алгоритм деление целых двоичных знаковых чисел методом без восстановления остатка.

	Регистр (B)	Регистр (C)	Регистр (A)	Счетчик

	делимое X								делитель Y				частное Z				тактов (Сч.Т)
	0	0	0	1	0	1	0	1	1	0	0	1	1	1	0	0	4
(+)	1	1	0	0	1				1	1	0	0	1				
	1	**1**	**0**	**1**	**1**												3
	0	1	1	1	0	1											
(-)	0	1	1	1													
	1	**1**	**1**	**0**													2
	1	1	0	0	1												
(-)	0	1	1	1													
1	**0**	**0**	**1**	**1**													1
	0	1	1	1													
(+)	1	0	0	1													
1	**0**	**0**	**0**	**0**													0
СТОП																	
Счетчик тактов равняется (0), деление окончено, но как было сказано выше, в частном мы получили псевдо частное ($Z^{'}$ = 1100). Для получения истинного частного (Z), необходимо выполнить коррекцию, т.е. к ($Z^{'}$ +1) и тогда получаем истинное значение частного: Z = 1101.																	

Случай №3 $X < 0; Y > 0; Z < 0$.

Необходимо разделить два знаковых числа:

$X = -21$ – делимое;

$Y = +7$ – делитель;

$Z = -3$ – частное.

$((-21):(+7) = (-3))$.

В этом случае знаки делимого и делителя не совпадают, поэтому на втором шаге производится сложение отрицательного делимого и положительного делителя.

Если последний частичный остаток не равен (0), то необходима коррекция частного, т.е. после получения частного к последнему биту частного необходимо прибавить (1).

Алгоритм деления приведен в табл. 2.18.

Таблица 2.18 – Алгоритм деление целых двоичных знаковых чисел методом без восстановления остатка.

	Регистр (B) делимое	Регистр (C) делитель	Регистр (A) частное	Счетчик тактов

	X								Y				Z				(Сч.Т)
	1	1	1	0	1	0	1	1	0	1	1	1	1	1	0	1	4
(+)	0	1	1	1	1				0	0	1	1	1				
1	**0**	**1**	**1**	**0**	**0**												3
	1	0	0	0	1	1											
(-)	1	0	0	1													
1	**0**	**0**	**0**	**1**													2
	0	0	1	1	1												
(-)	1	0	0	1													
	1	**1**	**0**	**0**													1
	1	0	0	1													
(+)	0	1	1	1													
1	**0**	**0**	**0**	**0**													0
СТОП																	
Счетчик тактов равняется (0), деление окончено и как было сказано выше, т.к. последний частичный остаток равен нулю, то мы сразу получили истинное значение частного Z = 1101 и корректировка не потребовалась.																	

Случай №4 $X < 0; Y < 0; Z > 0$.

Необходимо разделить два знаковых числа:

$X = -21$ – делимое;

$Y = -7$ – делитель;

$Z = +3$ – частное.

$((-21):(-7) = (+3))$.

В этом случае знаки делимого и делителя совпадают, поэтому на втором шаге производится вычитание, из отрицательного делимого вычитается отрицательный делитель (или производится сложение с положительной записью делителя).

Если последний частичный остаток равен (0), то необходима коррекция частного, т.е. после получения частного к последнему биту частного необходимо прибавить (1).

Алгоритм деления приведен в табл. 2.19.

Таблица 2.19 – Алгоритм деление целых двоичных знаковых чисел методом без восстановления остатка.

	Регистр (B) делимое	Регистр (C) делитель	Регистр (A) частное	Счетчик тактов

	X								Y				Z				(Сч.Т)
	1	1	1	0	1	0	1	1	1	0	0	1	0	0	1	0	4
(-)	0	0	1	1	1				1	1	0	0	1				
1	**0**	**0**	**1**	**0**	**0**												3
	1	0	0	0	1	1											
(+)	1	0	0	1													
1	**0**	**0**	**0**	**1**													2
	0	0	1	1	1												
(+)	1	0	0	1													
	1	**1**	**0**	**0**													1
	1	0	0	1													
(-)	0	1	1	1													
1	**0**	**0**	**0**	**0**													0
СТОП																	
Счетчик тактов равняется (0), деление окончено в частном мы получили псевдо частное ($Z^{'}$ = 0010) и как было сказано выше, если последний частичный остаток равен (0), то необходима коррекция частного, т.е. после получения частного к последнему биту частного необходимо прибавить (1) для получения истинного частного (Z), необходимо выполнить коррекцию, т.е. к ($Z^{'}$+1) и тогда получаем истинное значение частного: Z = 0011.																

Для закрепления материала рекомендуется выполнить следующие упражнения:

1. Перевести в **двоичную** систему счисления и вычислить выражение:
$((19_{(10)} - 15_{(10)}) * (18_{(10)} - 11_{(10)}))/7_{(10)}$.
2. Перевести в **двоичную** систему счисления и вычислить выражение:
$((10_{(10)} - 15_{(10)}) * (18_{(10)} - 12_{(10)}))/6_{(10)}$.
3. Перевести в **двоичную** систему счисления и вычислить выражение:
$((17_{(10)} - 12_{(10)}) * (23_{(10)} - 17_{(10)}))/(-6_{(10)})$.
4. Перевести в **двоичную** систему счисления и вычислить выражение:
$((25_{(10)} - 21_{(10)}) * (10_{(10)} - 17_{(10)}))/(-7_{(10)})$.
5. Перевести в **двоичную** систему счисления и вычислить выражение:
$((14_{(10)} - 21_{(10)}) * (10_{(10)} - 16_{(10)}))/5_{(10)}$.
6. Перевести в **двоичную** систему счисления и вычислить выражение:
$((15_{(10)} - 11_{(10)}) * (12_{(10)} - 17_{(10)}))/5_{(10)}$.

2.4 Выполнение арифметических операций в микропроцессорных системах над числами с плавающей точкой

2.4.1 Особенности представления числовых данных с плавающей точкой

Каждой форме представления чисел (с фиксированной или плавающей точкой) свойственны свои преимущества и недостатки. Поэтому в современных микропроцессорных системах обычно имеется возможность использования любой из них. Микропроцессорные средства, в которых используется преимущественно представление чисел в форме с фиксированной точкой, представляют собой специализированные микропроцессорные системы для управления технологическими процессами, обработки статистической информации и результатов измерений, решения задач с большим числом логических операций и т.д. Представление же чисел в форме с плавающей точкой используется в микропроцессорных системах, которые служат для решения широкого круга задач научного и технического характера. Это обусловлено тем, что использование формы с плавающей точкой позволяет значительно расширить диапазон представимых чисел, по сравнению с фиксированной точкой, обеспечивая при этом высокую точность представления.

$$X = M_X \cdot 2^{P_X}.$$

где M – мантисса, P – порядок.

При представлении чисел в форме с плавающей точкой мантисса может быть любым знаковым числом: целым, правильной дробью, смешанным числом. Однако условие нормализации предполагает представление мантиссы в виде правильной дроби:

$$2^{-1} <| M_X |< 1.$$

где M – мантисса.

Знак мантиссы определяет знак всего числа.

Порядок определяет фактическое положение запятой, которое она занимает в изображении мантиссы, и может быть положительным или отрицательным, но обязательно целым числом. Понятие «ПЗ» можно рассматривать как перемещение запятой в изображении числа влево, если $P < 0$, или вправо, если $P > 0$. Для обозначения чисел с плавающей точкой используют форму вида:

$$X = \pm M_X E \pm P_X.$$

где M – мантисса числа, E – разделитель мантиссы и порядка, P – порядок числа.

Вместо E могут использоваться другие символы, например, D (double – двойная точность), когда для мантиссы отводится большее количество разрядов. Если для записи порядка используется (P) разрядов, а для записи мантиссы (M) разрядов, то в форме с плавающей точкой может быть представлено следующее максимальное (по абсолютной величине) число:

$$|X_{\max}| = (1-2^{-m})\cdot 2^{2^P-1} = 2^{2^P-1} - 2^{2^P-m-1}.$$

В свою очередь, минимальное по абсолютной величине, но не равное нулю, число в форме с плавающей точкой может быть представлено в виде:

$$|X_{\min}| = (2^{-1})\cdot 2^{-(2^P-1)} = 2^{-2^P}.$$

При получении последней оценки полагаем, что минимальное значение мантиссы равно (2^{-1}) из-за условия нормализации. Подобные рассуждения применимы и для определения максимального и минимального отрицательных чисел, поскольку они представляются в микропроцессорных системах. В микропроцессорных системах, где используется представление чисел в форме с плавающей точкой, арифметические операции выполняются как над мантиссами операндов, так и над их порядками. При этом часто необходимо выполнять операцию нормализации чисел. Поэтому процессоры с плавающей точкой являются более сложными и менее быстродействующими, чем процессоры с фиксированной точкой (естественно, при сравнимом быстродействии электронных компонентов этих процессоров). Вместе с тем, в микропроцессорных системах с фиксированной точкой возникают определенные трудности из-за необходимости масштабирования данных. В этом легко убедиться, сравнивания отношения для различных форм представления числовых данных:

$$G = \frac{|X_{\max}|}{|X_{\min}|}.$$

Для чисел с плавающей точкой:

$$G = \frac{(1-2^{-m})\cdot 2^{2^P-1}}{(2^{-1})\cdot 2^{-(2^P-1)}} = \frac{2^{2^P-1}\cdot 2^{2^P-1}\cdot(1-2^{-m})}{2^{-1}} = 2^{2(2^P-1)}\cdot\frac{1-2^{-m}}{2^{-1}} =$$

$$= 2^{2^{P+1}-1}\cdot(1-2^{-m}) = 2^{2^{P+1}-1}\cdot(1-2^{-m})$$

Из рассмотренного следует, что диапазон представления чисел в форме с плавающей точкой зависит от разрядности порядка. Увеличение разрядности порядка на (1) значительно расширяет диапазон (таблица 2.20).

Таблица 2.20 – Зависимость диапазона представимых чисел от разрядности порядка

P	$X_{\min}$	$X_{\max}$
6	$2^{-2^6} = 2^{-64} \approx 10^{-19}$	$2^{2^6-1} = 2^{63} \approx 10^{19}$
7	$2^{-} = 2^{-} \approx 10^{-}$	$2 = 2^{-} \approx 10^{-}$
8	$2^{-2^8} = 2^{-256} \approx 10^{-76}$	$2^{2^8-1} = 2^{255} \approx 10^{76}$

Хотя диапазон представления чисел в форме с плавающей точкой во много раз шире диапазона представления чисел с фиксированной точкой, тем не менее в процессе вычислений могут возникать ситуации, когда результат вычислений выходит за пределы диапазона.

2.4.2 Форматы двоичных числовых данных с плавающей точкой

Формат данных с плавающей точкой рис. 2.57, использовавшийся в микропроцессорных системах первых поколений, включал четыре поля, а именно: указанные ранее поля для мантиссы и ее знака (длиной $(m+1)$ разряд), а также два дополнительных поля для порядка и его знака (длиной $(p+1)$ разряд).

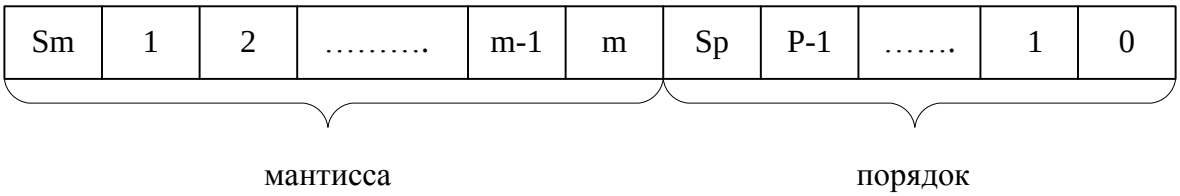


Рисунок 2.57 – Классический формат данных с плавающей точкой

В отличие от рассмотренного выше классического формата, в современных микропроцессорных системах используют смещенный порядок, т.е. порядок, представленный в смещенном коде.

Смещенный порядок (E_X) представляет собой целое беззнаковое число, равное истинному порядку, увеличенному на некоторое значение смещения.

Такой прием устраняет необходимость использования знака порядка, давая возможность манипулировать порядками как целыми беззнаковыми числами.

$$E_X = \begin{cases} D + |P_X|, & P_X \geq 0 \\ D - |P_X|, & P_X < 0 \end{cases}.$$

где D – некоторая константа (смещение).

Если $P_x > P_y$, то $E_x > E_y$. Если $(D = 2^{\cdot})$, то такой смещенный код называют кодом с положительным нулем. Для перехода от смещенного кода к дополнительному достаточно проинвертировать знаковый бит.

Широко применяются смещенные коды с отрицательным нулем, когда $(D = 2^{p-1} - 1)$. При сложении и вычитании смещенных порядков результирующий порядок также должен быть смещенным.

При использовании смещенных кодов с отрицательным нулем сложение порядков выполняется в соответствии с правилом:

$$E_Z = E_X + E'_Y + 1.$$

где E'_Y – смещенный порядок с инвертированным старшим разрядом.

В этом случае может возникать как положительное, так и отрицательное переполнение. Если во втором слагаемом старший разряд (1) был инвертирован в (0), а при сложении возник перенос $(CY = 1)$, то имеет место переполнение порядка. Результат оказывается за пределами представления. Признаком получения отрицательного переполнения (антипереполнение) является отсутствие переноса $(CY = 0)$ из старшего разряда суммы при условии, что во втором слагаемом старший разряд из (0) был инвертирован в (1).

При вычитании смещенных порядков для получения смещенного порядка разности необходимо выполнить следующее действие:

$$E_Z = E_X + \overline{E}'_Y + 1.$$

где E'_Y – инверсное значение смещенного порядка с инвертированным старшим разрядом.

2.4.3 Стандарт IEEE-754

К настоящему времени разработаны многочисленные варианты форматов чисел с плавающей точкой и практической реализации арифметических устройств с плавающей точкой. Это, в свою очередь, создает трудности при написании переносимых программ, согласовании правил обработки числовых данных и т.д.

Поэтому был разработан международный стандарт IEEE-754, нормам которого должны удовлетворять практически все вновь разрабатываемые микропроцессорные системы. Указанный стандарт решает следующие задачи:

– упорядочивает форматы чисел с плавающей точкой;

– унифицирует реакции на особые случаи в процессе обработки;
– определяет требования к точности вычислений.

В соответствии со стандартом IEEE–754 определены два базовых и два расширенных формата. Во всех форматах используется смещенный код порядка. Мантисса имеет старший разряд (m_0) с весом (2^0), относящийся к целой части. Поэтому условие нормализации изменяется:

$$1 \le |M_X| < 2.$$

Базовый одинарный формат (БОФ) –имеет длину в 32 разряда. Поле порядка в этом формате содержит 8 разрядов, а смещение равно 127.

Максимальный положительный порядок числа здесь равен $+127$, минимальный отрицательный -128.

Базовый двойной формат (БДФ) – имеет одноразрядное знаковое поле, 11-разрядное поле смещенного порядка и 52-разрядное поле мантиссы.

Смещение здесь равно 1023. Особенность двух указанных форматов заключается в том, что минимальный и максимальный смещенные порядки зарезервированы для представления специальных чисел. Поля мантисс в явном виде не содержат разряда (m_0), так как в памяти должны храниться нормализованные мантиссы чисел, а значит, их старший разряд всегда равен (1). При передаче числа из памяти в процессор этот «неявный» разряд становится «явным», и в операциях участвует 24-разрядная (*БОФ*) или 56-разрядная (*БДФ*) мантисса.

Расширенный одинарный формат(РОФ) – не имеет жестко фиксированных параметров, однако для конкретных реализаций он устанавливается, исходя из следующих требований:

– наличие одноразрядного знакового поля;
– наличие явного или скрытого разряда в мантиссе;
– длина мантиссы не менее 31 разряда;
– диапазон значений порядка -1023... +1024.

Расширенный двойной формат (РДФ) – отличается от одинарного тем, что диапазон порядка составляет -16383...+16384, а поле мантиссы - не менее 63 разрядов. В качестве примера практической реализации форматов стандарта IEEE–754 можно рассмотреть представление числовых данных с плавающей точкой в микропроцессорных системах. Здесь используются данные трех типов (таблица 2.21):

– вещественные числа одинарной точности (single real), соответствующие $(БОФ)$;
– вещественные числа двойной точности (double real), соответствующие $(БДФ)$;
– вещественные числа расширенной точности (extended real) или временные вещественные числа, соответствующие $(РДФ)$.

Временные вещественные числа имеют в своем представлении явный бит (m_0). В памяти микропроцессорной системы числа хранятся чаще всего в форматах ($БОФ$) или ($БДФ$), а их обработка выполняется в ($РДФ$).

Следует отметить, что целые числа в диапазоне ($2^{-64}...2^{+64}$) представляются абсолютно точно в ($РДФ$).

Таблица 2.21 – Зависимость диапазона представимых чисел от базовых форматов

Тип	Длина	Точность	Диапазон (дв.)	Диапазон (дес.)
БОФ	32	24	$2^{-126}\ldots2^{127}$	$1{,}18\times10^{-}\ldots3{,}40\times10^{-}$
БДФ	64	53	$2^{-}\ldots2^{-}$	$2{,}23\times10^{-}\ldots1{,}79\times10^{-}$
РДФ	80	64	$2^{-16382}\ldots2^{16383}$	$3{,}37\times10^{-}\ldots1{,}18\times10^{-}$

Если в процессе вычислений результат выходит за пределы представимых чисел, то имеет место особый случай:

- нарушение нормализации;
- если результат превышает $|X_{\max}|$, то такой случай называют переполнением или переполнением порядка, и вычисления при этом останавливают (прерывание);
- если результат меньше $|X_{\min}|$, имеет место антипереполнение или исчезновение порядка, что обычно воспринимается как нулевой результат (машинный 0);
- в результате выполнения сложения-вычитания мантисса может оказаться равной (0) при ненулевом порядке. Это ситуация потери значимости (псевдонуль). Результат интерпретируется как машинный (0).

2.4.4 Принципы выполнения арифметических операций над числами в форме с плавающей точкой

Пусть требуется выполнить некоторую арифметическую операцию над операндами X и Y в форме с плавающей точкой:

$$X = M_X \cdot 2^{P_X}.$$

$$Y = M_Y \cdot 2^{P_Y}.$$

Результатом операции будет число:

$$Z = M_Z \cdot 2^{P_Z}.$$

Порядки чисел X и Y представлены p разрядами, а мантиссы – (m-разрядные) правильные дроби, то есть:

$$2^{-1} \leq | M_X | < 1,\ 2^{-1} \leq | M_Y | < 1.$$

2.4.5 Сложение и вычитание чисел в форме с плавающей точкой

Представим операцию сложения в виде:

$$Z = X + Y.$$

Пусть $P_x > P_y$, тогда:

$$Y = M_Y \cdot 2^{P_Y} = M_Y \cdot \frac{2^{P_X - P_Y} \cdot 2^{P_Y}}{2^{P_X - P_Y}} = M_Y \cdot \frac{2^{P_X}}{2^{P_X - P_Y}} = \frac{M_Y}{2^{\Delta}} \cdot 2^{P_X}.$$

где $\Delta = P_X - P_Y$.

$$Z = M_X \cdot 2^{P_X} + \frac{M_Y}{2^{\Delta}} \cdot 2^{P_X} = (M_X + \frac{M_Y}{2^{\Delta}}) \cdot 2^{P_X} = M_Z \cdot 2^{P_Z}.$$

где $M_Z = M_X + \frac{M_Y}{2^{\Delta}};\ P_Z = P_X$.

Операция сложения и вычитания операндов в форме с плавающей точкой включает следующие этапы:

- выравнивание порядков операндов и определение порядка результата;
- сложение и вычитание мантисс;
- нормализация результата;
- округление результата.

Выравнивание порядков необходимо для того, чтобы цифры мантисс с определенными весами располагались в соответствующих им разрядах регистров. Выполняется такая операция путем сдвига вправо мантиссы того операнда, порядок которого меньше. Алгоритм выравнивания порядков сводится к следующему:

- сформировать разность порядков $\Delta = P_X - P_Y$;
- если $\Delta > 0$, то сдвинуть на один разряд вправо мантиссу (Y) и вычесть единицу из разности порядков (Δ); если $\Delta < 0$, то сдвинуть на один

разряд вправо мантиссу (X) и прибавить единицу к разности порядков (Δ);

– пункт 2 алгоритма повторять до тех пор, пока выполняется неравенство $\Delta \neq 0$; при $\Delta = 0$ выравнивание порядков считают законченным.

Так как порядки операндов могут иметь разные знаки, то для представления их разности (Δ) может понадобиться $(p+1)$ разряд. Абсолютная величина (Δ) может превышать число разрядов (m), отводимых для представления мантисс. В этом случае все разряды сдвигаемой вправо мантиссы выходят за пределы разрядной сетки и далее, при сложении и вычитании мантисс, абсолютная величина мантиссы, которая оставалась неподвижной, не меняется. Следовательно, при выполнении условия $|\Delta| > m$ можно прекратить выравнивание порядков и присвоить окончательному результату значение операнда, мантисса которого не сдвигалась.

Определять порядок результата удобно вместе с выравниванием порядков операндов. Порядок результата равен порядку большего по абсолютной величине операнда (то есть, того, мантисса которого не сдвигалась при выравнивании порядков).

Сложение и вычитание мантисс, для представления которых обычно используется прямой код, выполняется по тем же алгоритмам, что и чисел в форме с фиксированной точкой. Однако переполнение разрядной сетки здесь не приводит к остановке вычислений, что, как правило, имеет место при выполнении операций над числами в форме с фиксированной точкой. Наличие переполнения учитывается далее при нормализации результата.

Поскольку при выравнивании порядков возможна потеря части разрядов мантиссы одного из операндов, то мантисса результата может быть получена с погрешностью, по абсолютной величине почти равной единице младшего разряда. Однако среднее значение погрешности при этом равно нулю, если считать, что знаки "+" и "−" операндов и заданных операций появляются с равными вероятностями. С целью уменьшения максимальной абсолютной величины этой погрешности может быть введена операция округления мантиссы результата.

Необходимость нормализации результата обусловлена тем, что при мантиссах операндов, удовлетворяющих условиям нормализации, в зависимости от знака операции, мантисса результата может находиться в пределах $0 \le |z| < 2$.

Таким образом, при сложении и вычитании мантисс может произойти нарушение нормализации мантиссы результата на один разряд влево или на произвольное число разрядов вправо. Нормализация результата выполняется следующим образом. Если $1 \le |z| < 2$ (нормализация нарушена влево), то сдвинуть мантиссу результата на один разряд вправо и прибавить единицу к порядку результата. При этом может возникнуть переполнение порядков. При сложении мантисс близких по модулю друг к другу, но имеющих разные знаки

может возникнуть нарушение нормализации вправо на любое число разрядов (вплоть до получения нулевой мантиссы).

Если $0<|z|<1/2$, то сдвинуть мантиссу влево и вычесть единицу из порядка результата. Эти действия продолжать до тех пор, пока сохраняются условия указанного выше неравенства (если мантисса результата представлена прямым кодом, то до тех пор, пока в старшем разряде мантиссы в результате сдвига появится единица). В этом случае может возникнуть антипереполнение.

Если $|z|=0$, то результату присваивается машинный нуль (потеря значимости), а выполнять нормализацию нет необходимости.

При использовании модифицированного дополнительного кода для вычитания левое нарушение нормализации обнаруживается так же, как и в предыдущем случае, а правое нарушение – по комбинациям 00,0... и 11,11... .

2.4.6 Умножение и деление чисел в форме с плавающей точкой

Представим операцию умножения в виде:

$$Z = X \cdot Y .$$

тогда:

$$Z = M_X \cdot 2^{P_X} \cdot M_Y \cdot 2^{P_Y} = (M_X \cdot M_Y) \cdot 2^{P_X + P_Y} = M_Z \cdot 2^{P_Z} .$$

где: $M_Z = M_X \cdot M_Y$; $P_Z = P_X + P_Y$.

Знак произведения определяют путем суммирования по модулю 2 цифр в знаковых разрядах сомножителей:

$$S_Z = S_X \oplus S_Y .$$

При умножении чисел в форме с плавающей запятой порядок результата определяется путем сложения порядков сомножителей. Для определения мантиссы результата производят умножение мантисс операндов как чисел с фиксированной точкой по любому из рассмотренных алгоритмов. Мантисса произведения может оказаться ненормализованной, причем возможно только правое нарушение нормализации на один разряд. В этом случае мантиссу надо удвоить, сдвинув ее на один разряд влево, а из порядка произведения вычесть единицу.

Представим операцию деления в виде:

$$Z = X / Y .$$

Следует помнить, что при делении чисел с плавающей точкой остатка не бывает:

$$Z = \frac{M_X \cdot 2^{P_X}}{M_Y \cdot 2^{P_Y}} = \frac{M_X}{M_Y} \cdot 2^{P_X - P_Y} = M_Z \cdot 2^{P_Z}.$$

где: $M_Z = M_X / M_Y;\ P_Z = P_X - P_Y$.

Знак частного обычно определяют путем суммирования по (модулю 2) цифр в знаковых разрядах делимого и делителя. При делении чисел в форме с плавающей точкой порядок частного находят как разность порядков операндов (из порядка делимого вычитается порядок делителя). Мантиссу частного получают в результате деления абсолютных величин мантисс операндов. При делении нормализованных мантисс, как правильных дробей, может иметь место нарушение нормализации влево на один разряд.

Таким образом, при умножении и делении чисел в форме с плавающей точкой над порядками выполняются только операции сложения и вычитания.

Для этого удобно использовать дополнительные коды. В процессе выполнения операций над порядками возможно как переполнение порядков, так и антипереполнение.

Общий алгоритм выполнения операций умножения и деления:

- определение знака результата;
- сложение и вычитание порядков;
- умножение и деление мантисс;
- нормализация результата;
- округление результата.

Как отмечалось ранее, необходимость округления результата вызвана тем, что при нормализации результата может выполняться правый сдвиг мантиссы результата и младший ее разряд при этом выходит за пределы разрядной сетки. Если считать, что результат представлен прямым кодом, то отбрасывание этого разряда вносит в абсолютную величину результата либо нулевую погрешность (если отбрасывается цифра 0), либо погрешность, равную 1/2 веса младшего разряда (если отбрасывается цифра 1). Для того, чтобы погрешность по абсолютной величине результата была знакопеременной и ее среднее значение в достаточно большой последовательности операций было равно нулю, округление следует производить следующим образом. Если за пределы разрядной сетки сдвигается цифра (1), то младший разряд мантиссы результата устанавливают в единицу независимо от того, какая цифра была в этом разряде ранее. В другом случае значение младшего разряда не меняют.

Пусть (x_m) – младшая цифра мантиссы результата до округления, (x'_m) – та же цифра после округления, (x_{m+1}) – цифра мантиссы, выходящая за пределы разрядной сетки при округлении, (δ) – погрешность округления. Тогда все возможные ситуации при округлении иллюстрируются табл. 2.22, из которой видно, что при таком округлении погрешность абсолютной величины

мантиссы результата будет равна 0 или $\pm 1/2$ веса младшего разряда, а ее среднее значение равно нулю.

Таблица 2.22 – Результаты округления мантиссы результата

x_m	x_{m+1}	x'_m	δ
0	0	0	0
0	1	1	$+(1/2)2^{-m}$
1	0	1	0
1	1	1	$-(1/2)2^{-m}$

2.4.7 Алгоритмы выполнения операций над числами в форме с плавающей точкой, в представлении IEEE-754

Вещественное число может быть представлено в каком-либо формате стандарта IEEE–754 следующим образом:

$$X = (-1)^{S_X} \times m_0, M_X \times 2^{E_X - D}.$$

С учетом условия нормализации ($m_0 = 1$), минимальные и максимальные абсолютные значения мантисс:

$$M_X^{\min} = M_Y^{\min} = 1.$$

$$M_X^{\max} = M_Y^{\max} = 2 - 2^{-m}.$$

Сложение, вычитание, умножение и деление вещественных чисел в формате IEEE-754 осуществляется в соответствии с рассмотренными ранее алгоритмами. Поскольку в форматах хранятся не истинные значения порядков, а смещенные коды порядков, имеют место некоторые особенности, связанные с выполнением операций над смещенными порядками. Кроме того, должен учитываться факт наличия бита (m_0) мантиссы.

При сложение и вычитании сравнение смещенных порядков осуществляется путем вычитания смещенных кодов как целых беззнаковых чисел. Сложение мантисс выполняется по правилам сложения целых знаковых чисел. Если получен отрицательный результат, то он представлен в дополнительном коде и его необходимо преобразовать в прямой код.

При умножении смещенный порядок произведения определяется в соответствии с рассмотренными выше правилами. Мантисса произведения может иметь нарушение нормализации.

Проанализируем различные ситуации:

1. $M_X^{\min} \cdot M_Y^{\min} = 1 \cdot 1 = 1.$

Нет нарушения нормализации.

2. $M_X^{\min} \cdot M_Y^{\max} = M_X^{\max} \cdot M_Y^{\min} = 1 \cdot (2 - 2^{-m}) = 2 - 2^{-m}.$

Нет нарушения нормализации.

3. $M_X^{\max} \cdot M_Y^{\max} = (2 - 2^{-m}) \cdot (2 - 2^{-m}) = 4 - 4 \cdot 2^{-m} + 2^{-2m}.$

Здесь имеет место нарушение нормализации влево на 1 разряд.

При делении смещенный порядок частного определяется в соответствии с рассмотренными выше правилами. Мантисса частного может иметь нарушение нормализации. Проанализируем различные ситуации:

1. $M_X^{\min} / M_Y^{\min} = 1/1 = 1.$

Нет нарушения нормализации.

2. $M_X^{\min} / M_Y^{\max} = 1/(2 - 2^{-m});\ 0{,}5 < Z < 1.$

Нарушение нормализации вправо на 1 разряд.

3. $M_X^{\max} / M_Y^{\min} = (2 - 2^{-m})/1 = 2 - 2^{-m};\ 1 < Z < 2.$

Нет нарушения нормализации.

4. $M_X^{\max} / M_Y^{\max} = (2 - 2^{-m})/(2 - 2^{-m}) = 1.$

Нет нарушения нормализации.

2.5 Представление числовых данных в коде BCD

Наиболее широкое применение в микропроцессорной технике получили двоично-десятичные системы счисления, а именно BCD – коды, в которых десятичные цифры записываются как четырехразрядные двоичные числа – двоичные тетрады.

Такое представление десятичных цифр называют также безизбыточным, поскольку для него необходимо минимальное количество двоичных разрядов.

Ограниченное применение находят и избыточные представления десятичных цифр с помощью пяти, шести и семи двоичных разрядов. В

принципе, можно построить микропроцессорную систему, работающую при любом двоично-десятичном кодировании. Наличие разрешенных и запрещенных комбинаций является важным свойством BCD – кодов. В этом их отличие от обычных позиционных систем счисления.

Однако эмпирическим путем установлено, что для создания десятичных вычислительных средств наиболее целесообразно использовать двоично-десятичные коды (BCD или ДДК), обладающие свойствами взвешенности, упорядоченности, четности, дополнительности и единственности (так называемыми свойствами Рутисхаузера).

ДДК называется взвешенным, если каждому из (h) разрядов двоичного представления $(a_h a_{h-1},...,a_1)$ десятичной цифры (A) поставлены в соответствие веса – $(\gamma_h \gamma_{h-1},...,\gamma_1)$ причем:

$$A = a_h\gamma_h + a_{h-1}\gamma_{h-1} + ... + a_1\gamma_1.$$

где: A – десятичная цифра;

$(a_h a_{h-1},...,a_1)$ – разряды двоичного представления;

$(\gamma_h \gamma_{h-1},...,\gamma_1)$ – веса.

Свойство взвешенности упрощает выполнение логических и арифметических операций. ДДК, получаемые друг из друга простой перестановкой весов (γ_i), образуют кодовую группу. Для обозначения конкретных ДДК и кодовых групп часто используют последовательность их весов, записанную в порядке их убывания, например, "8,4,2,1", "4,4,2,1" и т.п.

Упорядоченность ДДК состоит в выполнении одного из условий:

$$0_{(2)} < 1_{(2)} < 2_{(2)} < ... < 9_{(2)}.$$

$$0_{(2)} < 1_{(2)} < 2_{(2)} < ... < 9_{(2)}.$$

для двоичных представлений $(0_{(2)}, 1_{(2)}, 2_{(2)},..., 9_{(2)})$ десятичных цифр. Наличие упорядоченности ДДК необходимо для реализации логических операций.

Свойство четности должно проявляться в том, чтобы всем четным десятичным цифрам соответствовали либо только четные, либо только нечетные их двоичные представления. Аналогично, всем нечетным десятичным цифрам должны соответствовать либо только нечетные, либо только четные их двоичные представления. Это свойство необходимо для реализации операций умножения, деления, округления.

Сущность свойства дополнительности («самодополняемость») ДДК заключается в следующем. Если сумма двух десятичных цифр равна (9), то переход от двоичного представления одной цифры к двоичному представлению другой цифры должен осуществляться путем инверсии двоичных разрядов.

Наличие этого свойства необходимо для упрощения алгебраических операций (операций с учетом знаков операндов) по правилам десятичной арифметики.

ДДК обладает свойством единственности, если между десятичной цифрой и комбинацией двоичных цифр установлено взаимно однозначное соответствие. Это свойство упрощает и облегчает как процедуру представления в ДДК десятичных чисел (то есть, кодирование), так и процедуру распознавания десятичных чисел в ДДК (то есть, декодирование).

Особенностью взвешенных ДДК, имеющих только положительные веса, кроме ДДК группы "8,4,2,1", является отсутствие однозначного представления десятичных цифр. Это означает, что некоторые десятичные цифры могут быть записаны несколькими комбинациями двоичных цифр. Например, в ДДК "4,4,2,1" цифру 4 можно представить как 1000 и как 0100, цифру 5 – как 1001 и как 0101, цифру 6 – как 1010 и как 0110, цифру 7 – как 1011 и как 0111. Причиной этого служит неоднозначность решения уравнения:

$$A = a_h\gamma_h + a_{h-1}\gamma_{h-1} + \ldots + a_1\gamma_1.$$

относительно переменных $a_i (i = 1, h)$.

Однозначность представления десятичных цифр, состоящая в том, что каждой такой цифре соответствует только одна из 16 двоичных тетрад, обеспечивается в ДДК группы "8,4,2,1".

Кроме того, однозначность представления десятичных цифр может быть обеспечена в ДДК с отрицательными весами. В табл. 2.23 приведены все кодовые группы однозначных ДДК. Следует отметить, что однозначность не входит в перечень свойств тех ДДК, которые наиболее целесообразно использовать для построения десятичных микропроцессорных систем.

Таблица 2.23 – Кодовые группы однозначных ДДК

«8,4,2,1»	«8,-4,2,1»	«8,-6,4,1»	«8,-4,3,2»	«8,-5,4,2»
«8,4,2,-1»	«8,4,3,-2»	«8,-6,5,3»	«7,-6,5,3»	«8,-4,-2,1»
«8,4,-2,1»	«8,6,-4,1»	«8,4,-3,2»	«8,-4,2,-1»	«8,5,-4,2»
«8,-4,3,-2»	«7,6,-5,3»	«8,4,-3,-2»	«8,5,-4,-2»	«8,6,-4,-1»
«8,7,-4,-2»				

В этом перечне ему соответствует более слабое свойство единственности. Это свойство для неоднозначных ДДК может быть обеспечено выбором одной и исключением из ДДК избыточных (хотя, в принципе, правильных) двоичных представлений десятичных цифр. В этом отношении свойство единственности ДДК не эквивалентно свойству однозначности систем счисления, так как единственность ДДК можно обеспечить искусственно, а однозначность должна быть присуща счислению «органически».

Существует 86 кодовых групп безизбыточных ДДК, обладающих свойствами взвешенности и единственности. Среди них есть ДДК, с

отрицательными весами, например, "8,4,2,1", "8,4,2,–1", "6,3,–1,–1". Свойством дополнительности обладают все взвешенные ДДК с положительными весами, у которых сумма весов равна (9).

Существуют четыре кодовые группы взвешенных ДДК, обладающих также и свойствами единственности и дополнительности, а именно: "5,2,1,1","4,3,1,1","4,2,2,1","3,3,2,1". Среди взвешенных ДДК с отрицательными весами свойствами единственности и дополнительности обладают ДДК из 19 кодовых групп, перечисленных в табл. 2.24.

Таблица 2.24 – Кодовые группы ДДК с отрицательными весами, обладающие свойствами единственности, дополнительности и взвешенности

«6,3,1,-1»	«7,3,1,-2»	«6,4,1,-2»	«6,5,1,-3»	«7,5,1,-4»
«6,2,2,-1»	«5,3,2,-1»	«6,3,2,-2»	«8,3,2,-4»	«4,4,2,-1»
«6,4,2,-3»	«8,4,2,-5»	«6,5,2,-4»	«4,4,3,-2»	«5,4,3,-3»
«8,4,3,-6»	«7,5,3,-6»	«8,4,-2,-1»	«8,6,-1,-4»	

Упорядоченность присуща лишь ДДК, не имеющим отрицательных весов. Одновременно свойствами единственности, упорядоченности, дополнительности и взвешенности обладают ДДК кодовых групп "5,2,1,1","4,3,1,1", "3,3,2,1","4,2,2,1". Всеми пятью свойствами обладают ДДК кодовой группы "4,2,2,1", называемые также кодами Эмери ("2,4,2,1" – код Айкена).

Недостатком ДДК этой кодовой группы является искусственный порядок весов, что затрудняет выполнение арифметических операций.

Наиболее распространенным в микропроцессорных системах является ДДК "8,4,2,1", который называется также кодом прямого замещения. Этот ДДК получают путем записи десятичных цифр в двоичной позиционной однородной системе счисления с естественным порядком весов. Он обладает всеми перечисленными выше свойствами ДДК, кроме свойства дополнительности. Этим обусловлен ряд неудобств при реализации операций алгебраического сложения в таком ДДК из-за трудностей формирования переносов из младшей тетрады в старшую.

Достоинством ДДК "8,4,2,1" следует считать простоту и удобство перевода чисел из десятичной системы счисления в двоично-десятичную и обратный перевод.

В каждом конкретном случае применение какого-либо ДДК обуславливается определенными его преимуществами по сравнению с другими типами ДДК. Например, код "7,4,2,1" применяется в электромеханических цифровых устройствах, где двоичной единице соответствует замкнутое состояние некоторой контактной пары и энергопотребляющее состояние соответствующей электрической цепи, а двоичному нулю - разомкнутое состояние контактной пары и не потребляющее энергии состояние электрической цепи. В этом случае каждое двоичное представление десятичной цифры содержит не более двух единиц, что обеспечивает минимальное и

постоянное потребление энергии от источника питания. В ДДК "5,4,2,1", десятичные цифры можно рассматривать как двоично-пятеричные с кодированным представлением цифр. Три младших разряда в каждой тетраде изображают одну пятеричную цифру, старшая цифра тетрады соответствует двоичному разряду. Этот ДДК имеет ряд достоинств при выполнении арифметических операций и переводе чисел из одной системы счисления в другую.

Для представления десятичных цифр могут быть использованы и невзвешенные ДДК. Например, код "с избытком 3" обладает свойством дополнительности и его удобно использовать для выполнения операции алгебраического сложения. Для записи десятичной цифры в ДДК "с избытком 3" необходимо двоичную тетраду этой цифры в ДДК "8,4,2,1" сложить с двоичным представлением числа (3).

ДДК "8,4,2,1" и коды с избытком (полученные из кода "8,4,2,1" по аналогии с кодом "с избытком 3") обладают еще одним важным свойством, не входящим в перечень Рутисхаузера, а именно: свойством аддитивности. Это свойство состоит в том, что ДДК суммы двух десятичных цифр равен двоичной сумме ДДК этих цифр или отличается от нее на некоторую константу. Это свойство позволяет свести операции десятичной арифметики в таких ДДК к выполнению операций по правилам двоичной арифметики.

Примерами избыточных ДДК, где каждая десятичная цифра кодируется пятью двоичными разрядами, являются ДДК "$3A+2$" и "2 из 5". Первый из этих ДДК обладает свойством дополнительности, а двоичные представления десятичных цифр получают в нем путем записи в двоичной системе с естественным порядком весов числа $3A+2$, где A – заданная десятичная цифра.

В ДДК "2 из 5" каждая десятичная цифра изображается пятью двоичными разрядами, из которых только два содержат единицы. Можно считать, что ДДК "2 из 5" получается из ДДК "7,4,2,1" путем добавления справа дополнительного разряда с весом, равным нулю. В этот разряд записывают такую цифру, чтобы общее число единиц было равно двум (за исключением десятичного нуля). ДДК $3A+2$ и "2 из 5" используют обычно для передачи информации, поскольку они позволяют обнаруживать одиночные ошибки, возникающие в процессе такой передачи, сравнительно простыми средствами.

Рефлексный ДДК (код Грея, соседний код) обладает тем свойством, что двум соседним десятичным цифрам в нем соответствуют кодовые комбинации, отличающиеся только в одном двоичном разряде. Это свойство эффективно используется при построении измерительных преобразователей (датчиков) величины углового или линейного перемещения в цифровой эквивалент.

ДДК – (w,x,y,z) – обладает следующим свойством. Для десятичной цифры (A) в таком ДДК перестановка ее двоичных разрядов вида (z,w,x,y) равна младшему десятичному разряду (в двоичном представлении) произведения (A) на (3); перестановка вида (x,y,z,w) равна младшему разряду

произведения (A) на (7); перестановка (y,z,w,x) равна младшему разряду произведения (A) на (9).

Таким образом, младший разряд произведения любой заданной десятичной цифры на $(3,7,9)$ может быть получен круговой перестановкой двоичных цифр. Очевидно, что это свойство может использоваться для частичного контроля правильности выполнения десятичного умножения.

2.5.1 Форматы числовых данных BCD-кодов

В современных микропроцессорных системах, обрабатывающих не только числовую информацию, но и текстовую (алфавитно-цифровую), отдельные символы чаще всего представляются словами длиной в один байт. Так как байт равен двум двоичным тетрадам, то в микропроцессорных системах с байтовым представлением информации можно записывать две десятичные цифры в один байт. Такую запись называют также упакованной записью или упакованным форматом записи десятичных цифр. Вместе с тем, можно записывать в один байт и по одной десятичной цифре, используя остальные четыре разряда для записи знаков и вспомогательных символов. Такую запись называют распакованной записью или распакованным форматом. В микропроцессорных системах широкого назначения обычно реализуются обе возможности записи десятичных цифр. Рассмотрим упакованный десятичный формат. Младшая цифра занимает в байте правую тетраду (биты $3-0$), а старшая – левую (биты $7-4$). Обе цифры представлены в коде "8,4,2,1". Многоразрядные упакованные десятичные числа занимают несколько байт. Для представления знака числа используют запрещенные кодовые комбинации: для «+» – $C_{(16)}$), для «–» – $D_{(16)}$. Код знака числа записывают в старшей тетраде старшего байта.

Распакованный формат называют символьным или ASCII - форматом.

Каждый байт содержит код, соответствующий десятичной цифре в кодовой таблице ASCII. В ней цифры имеют коды $30_{(16)}-39_{(16)}$. Таким образом, значение десятичной цифры представлено в младшей тетраде, а старшая тетрада является кодом «зоны» (3). Поэтому такой формат также называют зонированным. Для обозначения знака используют коды символов «+» и «–» из кодовой таблицы ASCII ($2B_{(16)}2D_{(16)}$ соответственно).

2.5.2 Арифметические операции над числовыми данными BCD-кодов

Сложение и вычитание десятичных двоично-кодированных операндов может быть реализовано на основе тех же принципов, что и двоичных операндов с учетом лишь особенностей суммирования десятичных цифр в двоичном их представлении:

– правила формирования десятичного переноса отличаются от правил формирования двоичного переноса;
– сумма двоичных кодов десятичных цифр, полученная по правилам двоичной арифметики, не всегда равна их сумме, полученной по десятичным правилам.

По этим причинам при использовании ДДК, обладающих свойством аддитивности, суммирование десятичных цифр в двоичном представлении проводят в два этапа:

1. суммируют двоичные представления десятичных цифр по правилам двоичной арифметики;
2. проводят коррекцию предыдущего результата путем прибавления или вычитания некоторой поправки для образования правильного представления десятичной цифры и десятичного переноса в старший десятичный разряд.

Если же ДДК не обладает свойством аддитивности, то сначала выполняют преобразование цифр слагаемых в ДДК, обладающий таким свойством, далее суммируют цифры в аддитивном ДДК, а затем результат суммирования подвергают обратному преобразованию.

Рассмотрим, как можно определить корректирующую поправку и правильно сформировать перенос в старший десятичный разряд для ДДК "8,4,2,1". Для этого составим таблицу двоичных сумм, их представлений после первого этапа сложения, правильных их представлений в ДДК "8,4,2,1" десятичных переносов и корректирующих поправок (табл. 2.25).

Таблица 2.25 – Таблица двоичных сумм после первого этапа сложения

Десятичная сумма	Сумма после первого этапа $s_5s_4s_3s_2s_1$	Правильная сумма $pz_4z_3z_2z_1$	Коррекция $c_5c_4c_3c_2c_1$

0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0
1	0 0 0 0 1	0 0 0 0 1	0 0 0 0 0
2	0 0 0 1 0	0 0 0 1 0	0 0 0 0 0
3	0 0 0 1 1	0 0 0 1 1	0 0 0 0 0
4	0 0 1 0 0	0 0 1 0 0	0 0 0 0 0
5	0 0 1 0 1	0 0 1 0 1	0 0 0 0 0
6	0 0 1 1 0	0 0 1 1 0	0 0 0 0 0
7	0 0 1 1 1	0 0 1 1 1	0 0 0 0 0
8	0 1 0 0 0	0 1 0 0 0	0 0 0 0 0
9	0 1 0 0 1	0 1 0 0 1	0 0 0 0 0
10	0 1 0 1 0	1 0 0 0 0	0 0 1 1 0
11	0 1 0 1 1	1 0 0 0 1	0 0 1 1 0
12	0 1 1 0 0	1 0 0 1 0	0 0 1 1 0
13	0 1 1 0 1	1 0 0 1 1	0 0 1 1 0
14	0 1 1 1 0	1 0 1 0 0	0 0 1 1 0
15	0 1 1 1 1	1 0 1 0 1	0 0 1 1 0
16	1 0 0 0 0	1 0 1 1 0	0 0 1 1 0
17	1 0 0 0 1	1 0 1 1 1	0 0 1 1 0
18	1 0 0 1 0	1 1 0 0 0	0 0 1 1 0
19	1 0 0 1 1	1 1 0 0 1	0 0 1 1 0

Из таблицы видно, что если результат, полученный на первом этапе, находится в пределах от 0 до 9, то корректировать его нет необходимости, а двоичный и десятичный перенос в этом случае совпадают. Если же результат первого этапа находится в пределах от 10 до 19, то для образования правильной суммы $(pz_4z_3z_2z_1)$ к предварительной сумме $(s_5s_4s_3s_2s_1)$ необходимо прибавить корректирующую поправку (00110). Признаком нахождения суммы первого этапа в диапазоне от 10 до 19 может служить наличие единичного значения переноса p. Таким образом, если при прибавлении коррекции перенос $(p=0)$, то как правильную сумму необходимо взять результат первого этапа, если же $(p=1)$, то как правильную сумму необходимо взять откорректированный результат.

При сложении или вычитании многоразрядных десятичных операндов описанные действия должны выполняться по каждому десятичному разряду (в ДДК "8,4,2,1" – по каждой тетраде двоичных разрядов).

Умножение чисел в десятичной системе счисления с двоично-кодированным представлением цифр может быть реализовано на тех же принципах, что и в двоичной системе, то есть, имеется четыре основных алгоритма десятичного умножения, каждый из которых имеет свои преимущества и недостатки, аналогичные, в целом, преимуществам и недостаткам соответствующих алгоритмов двоичного умножения. Особенности же десятичного умножения состоят в следующем:

- результат умножения двух двоичных цифр представляется только одной цифрой произведения с весом, равным произведению весов цифр сомножителей; в отличие же от этого, результат умножения двух десятичных цифр будет представлен двумя цифрами – цифрой собственно произведения и с весом, равным произведению весов цифр сомножителей, и цифрой переноса q в разряде с весом, на единицу большим произведения весов цифр сомножителей;
- суммирование частичных произведений должно осуществляться по правилам десятичной арифметики (то есть, используются для этого десятичные сумматоры);
- сдвиг в регистрах осуществляется одновременно на один десятичный разряд;
- на каждом шаге умножения осуществляется столько суммирований множимого с суммой частичных произведений, сколько единиц содержится в очередной цифре множителя.

Для учета особенности при умножении с младших разрядов или при умножении со старших разрядов десятичный разряд регистра множителя снабжают цепями вычитания единиц и в каждом такте суммирования цифру в этом разряде уменьшают на единицу. Такты суммирования производят до тех пор, пока в десятичном разряде, управляющем умножением, не будет получен нуль. Если для представления десятичных цифр используется ДДК "8,4,2,1", то указанный десятичный разряд выполняют в виде счетчика.

При умножении по этому алгоритму на один разряд множителя в среднем приходится $4{,}5$ тактов суммирования (предполагаем, что цифры $0,1,...,9$ появляются во всех разрядах множителя с равными вероятностями).

Деление десятичных чисел в микроконтроллерных системах, как и двоичных чисел, можно выполнять по двум основным алгоритмам: с восстановлением остатка и без восстановления остатка. Реализация первого алгоритма требует несколько больших временных затрат (примерно на 12...17%) по сравнению со вторым.

В соответствии с этим алгоритмом сдвиг выполняется на один десятичный разряд, все суммирования и вычитания, а также формирование обратного и дополнительного кодов (в случае использования их для вычитания) должны выполняться по правилам десятичной арифметики. На каждый шаг деления здесь приходится несколько суммирований-вычитаний.

Для ускорения операции десятичного деления необходимо, в первую очередь, уменьшать число суммирований и вычитаний, выполняемых для получения одной цифры частного.

Вопросы для самопроверки

1. Поясните термин «каноническая система счисления».

2. Назовите методы переводов чисел из одной канонической системы счисления в другую.
3. Дайте определение форматам и типам числовых данных.
4. Поясните принцип кодирования отрицательных двоичных чисел.
5. Поясните алгоритм операции сложения знаковых двоичных чисел, представленных в дополнительном коде.
6. Что такое контроль переполнения разрядности сетки?
7. Поясните алгоритмы операций сдвига.
8. Поясните алгоритмы умножения и деления беззнаковых и знаковых двоичных чисел.
9. Дайте определение стандарту IEEE-754.
10. Поясните принципы выполнения арифметических операций над числами в формате с плавающей запятой.
11. Поясните принцип представления числовых данных в коде BCD.
12. Поясните принципы выполнения арифметических операций над числовыми данными в коде BCD.
13. Поясните принцип представления числовых данных в коде ASCII.
14. Поясните принципы выполнения арифметических операций над числовыми данными в коде ASCII.

3 ОСНОВЫ МИКРОПРОЦЕССОРНЫХ МЕТРОЛОГИЧЕСКИХ СИСТЕМ

3.1 Архитектурные особенности вычислительных систем на базе микропроцессора i80x86 в реальном режиме

Согласно рис. 3.1 микропроцессор в реальном режиме физически позволяет обращение к **1 МБт** оперативной памяти (адресное пространство ограничено **20** линиями шины адреса) и **64 КБт** портов вода/вывода (при обращении к портам ввода/вывода используется только **16** младших разрядов шины адреса). В вычислительной системе используется раздельное обращение к памяти и портам ввода/вывода с помощью специальных сигналов управления **MEM ENB** и **IO ENB**. Таким образом, адресные пространства памяти и портов ввода/вывода не перекрываются. Шина данных микропроцессора имеет **16** разрядов, поэтому передача данных осуществляется двухбайтными словами.

Рисунок 3.1 – Структурная схема вычислительной системы на базе МП i80x86

Микропроцессоры семейства **i80x86** в реальном режиме работы имеют **16** разрядную внутреннюю организацию и содержат 14 **16**–разрядных регистров (см. рис. 3.2), которые подразделяются на регистры общего назначения, регистры смещения, сегментные и регистр флагов.

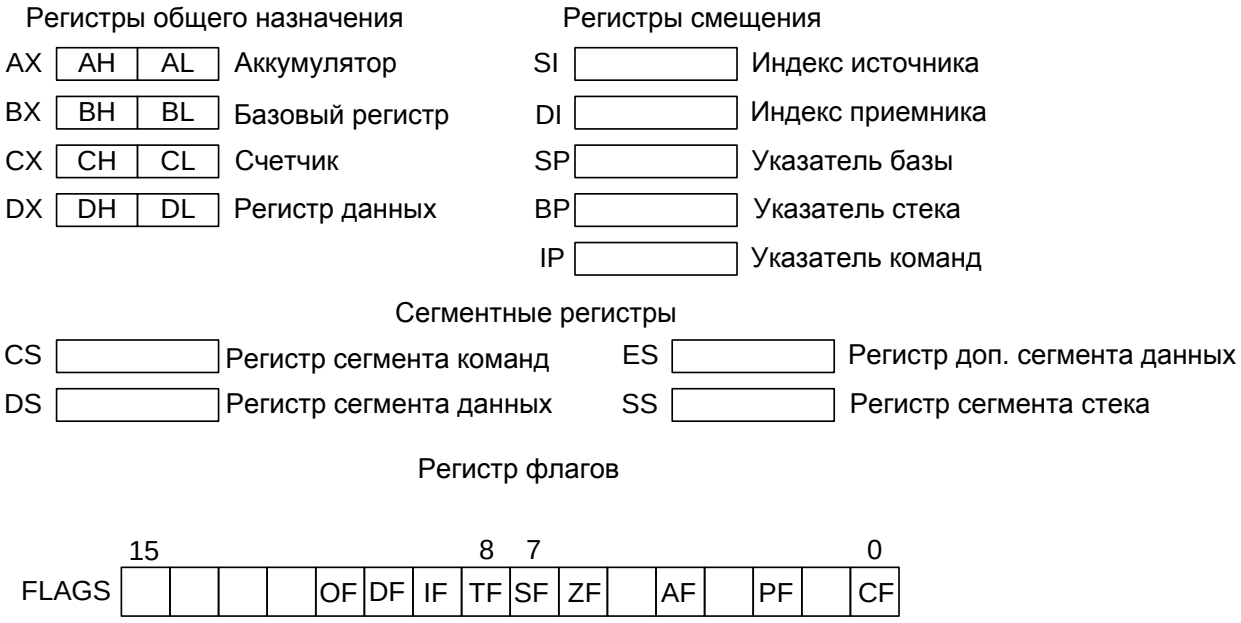


Рисунок 3.2 – Регистры микропроцессора i80x86, доступные в реальном режиме

Четыре регистра общего назначения **AX, BX,CX, DX** являются регистрами данных, поскольку они преимущественно используются для хранения данных. Регистр **AX** является аккумулятором (большинство команд выполняется быстрее и занимает в памяти меньше места при его использовании). При работе с этими регистрами допускается раздельное обращение к их младшим и старшим половинам: например, при выполнении команды пересылки данных:

mov CL, AH

в младший байт регистра **CX (CL)** записывается содержимое старшего байта регистра **AX** (**AL**), при этом вторые байты этих регистров остаются без изменения.

При обращении к остальным регистрам процессора не допускается побайтовая адресация. Регистры **CS, DS, ES, SS** предназначены для хранения начальных адресов сегментов программы, а в регистрах **SI, DI, BP, SP** находятся смещения относительно некоторой базы при выборке операндов из памяти. В регистре **IP** находится относительный адрес команды, следующей за исполняемой.

Регистр флагов **FLAGS** содержит информацию о текущем состоянии процессора. Он включает 6 флагов состояния и 3 бита управления состоянием процессора При организации контроля результатов арифметических операций, формирования условных переходов и работе с прерываниями представляют интерес следующие флаги:

ZF – флаг нуля, устанавливается в 1, если результат операции равен 0;

SF – флаг знака, устанавливается в 1 при отрицательном результате;

CF – флаг переноса, устанавливается в 1 при наличии переноса в арифметических операциях;

OF – флаг переполнения, фиксирует выход результата за пределы

допустимого диапазона;

mov DX, W – содержимое переменной **W** заносится в **DX**;

IF – управляющий флаг разрешения прерываний, разрешает (1) или запрещает (0) процессору реагировать на аппаратные прерывания.

Механизм работы с памятью на физическом уровне. Для реализации обращения к 20–и разрядному адресному пространству памяти с помощью 16–и разрядных регистров микропроцессоры семейства i80x86 используют механизм сегментной адресации – доступа к памяти с помощью сегментов (логических образований) размером от 0 до 64 КБт, накладываемых на произвольные участки физического адресного пространства. Формирование физического 20–и разрядного адреса (см. рис.1.3) осуществляется с помощью суммирования начального адреса сегмента памяти (сегментного адреса), в котором находится данная ячейка, и ее смещения (относительного адреса) относительно начала сегмента. Сегментный адрес, смещенный на 4 разряда вправо (деленный на 16), заносится в один из сегментных регистров. Согласно примеру, приведенному на рис 3.3, физический адрес **CB819h** ячейки формируется в микропроцессоре аппаратным умножением содержимого **BEC7h** одного из сегментных регистров на **10h** для получения полного шестнадцатеричного пятизначного сегментного адреса **BEC70h** и складывается со смещением **CBA9h** ячейки относительно начала сегмента. Исполнительный адрес часто обозначается в виде **SEG:OFFSET** (сегментной и относительной частей, разделенных двоеточием). Для примера, приведенного на рис. 1.3.: **BE859h = BE85:0009h**. Очевидно, что один и тот же физический адрес можно представить несколькими способами, в зависимости от выбора сегментного адреса.

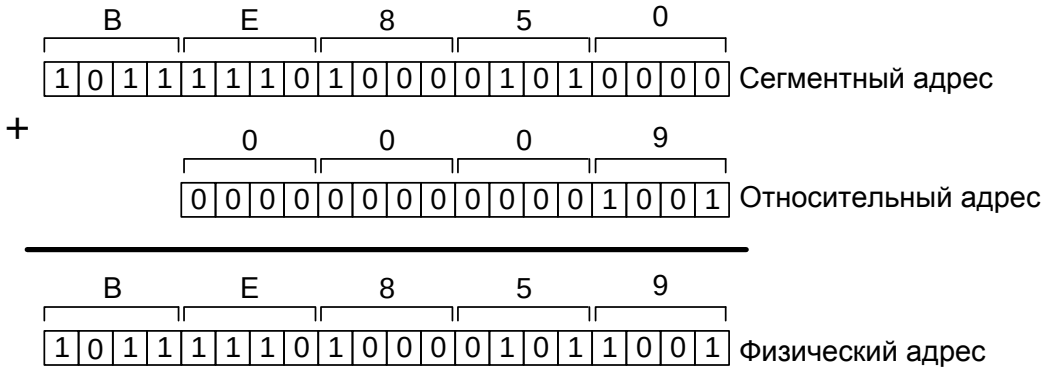


Рисунок 3.3 – Пример формирования физического адреса с помощью механизма сегментной адресации

mov AX, seg W – сегментный адрес переменной **W** заносится в **AX;**

mov ES, AX – содержимое **AX** заносится в **ES;**

mov BX, offset W – относительный адрес переменной **W** заносится в **BX.**

Минимальной адресуемой единицей информации в микропроцессорах семейства i80x86 является байт. У переменных, занимающих в памяти несколько байт, например, у переменных целых типов данных языка Borland Pascal, отображенных на рис. 3.4, младшему байту соответствует меньший адрес. При этом адресом переменной, занимающей в памяти несколько байт, является адрес ее младшего байта. На рис. 3.5. приводится пример распределения ячеек памяти четырехбайтовой переменной типа Longint, содержащей значение 1426A5E8h и адресуемой по смещению B800.

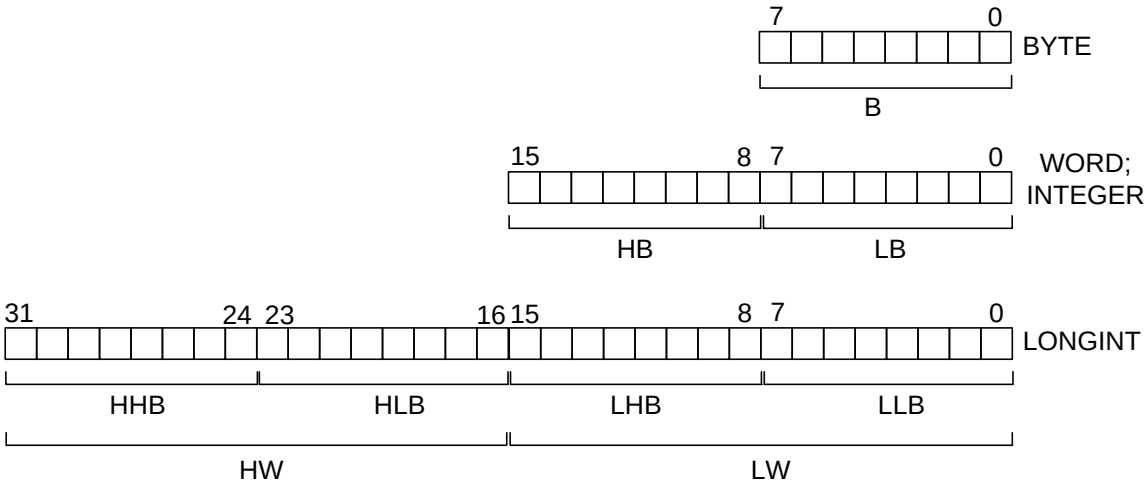


Рисунок 3.4 – Распределение ячеек памяти у переменных целых типов языка Borland Pascal

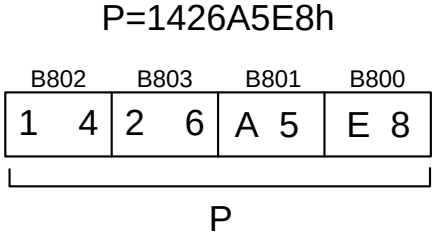


Рисунок 3.5 – Распределение ячеек памяти переменной P типа Longint

Для определения сегментного и относительного адресов переменных в ассемблере служат операторы **seg** и **offset**. Следующий фрагмент программы заносит содержимое, сегментный и относительный адреса переменной W типа Word в регистры DX, ES, BX:

Ввиду невозможности обеспечения произвольной пересылки данных в сегментные регистры для временного хранения сегментного адреса переменной W используется регистр-аккумулятор AX. Раздельный доступ к младшему и старшему байтам переменной W, можно организовать следующим образом:

mov CL, ES:[BX] – в **CL** заносится младший байт переменной **W**;
mov DH, ES:[BX+1] – в **DH** заносится старший байт переменной **W**.

При работе с адресными выражениями необходимо помнить, что с одного и того же адреса могут начинаться ячейки разных типов. В общем случае тип пересылаемых данных совпадает с типом операнда приемника (например, **Word** – для регистра **AX** и **byte** – для регистра **AH**). Для явного указания типа применяется оператор **PTR**. Так, выражения **Byte PTR A** и **WORD PTR A** позволяют обращение к ячейке памяти с адресом **A** как к байту и как к слову соответственно.

В общем случае в программе, работающей в операционной системе MSDOS, используются 3 сегмента памяти: сегмент кода, сегмент данных и сегмент стека. Причем в регистр **CS** заносится начальный адрес **сегмента кода,** в **DS** – начальный адрес **сегмента данных**, в **SS** – начальный адрес **сегмента стека**. Размеры сегментов могут быть произвольными в пределах от 0 до 64 КБт (см. рис. 3.6). В сегменте кода содержатся команды программы, в сегменте данных – глобальные переменные, сегмент стека используется для вспомогательных целей, хранения локальных переменных процедур и т.д. Регистр дополнительного сегмента данных ES может указывать на произвольную область памяти.

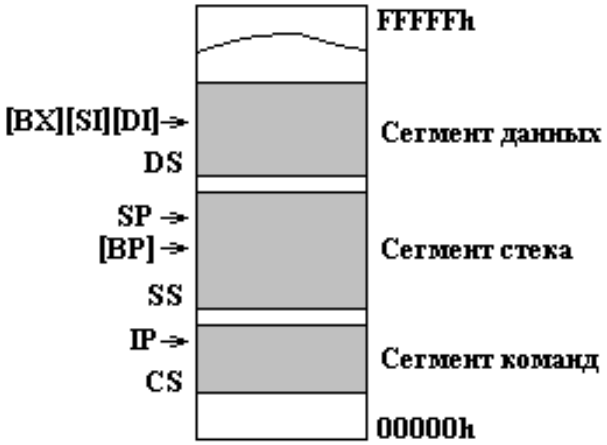


Рисунок 3.6 – Сегментная структура программы

Как видно из рисунка 3.6 при реализации механизма сегментной адресации смещения могут указываться не произвольно, а в соответствие с принятыми соглашениями: **CS:[IP], DS:[BX][SI][DI], ES:[BX][SI][DI], SS:[BP][SS].** В одном из сегментных регистров находится сегментный адрес, а через символ **:** в квадратных скобках указывается регистр, содержащий относительный адрес (смещение) ячейки памяти.

Основные принципы прямого обращения к видеопамяти в текстовом режиме. Процедуры обработки прерываний выполняют определенные узкоспециализированные функции. Как правило, это ввод/вывод данных из портов внешнего устройства, передача флага состояния в основную программу. Это связано прежде всего с тем, что **функции** базовой системы ввода/вывода

BIOS и операционной системы **DOS** являются **нереентерабельными** (нерекурсивными). Это означает, что прерывание выполняющейся функции **BIOS** или **DOS** (например, в главной программе) при выполнении аналогичной функции в обработчике прерывания приведет к нарушению последовательности управляющих команд и краху системы. Поэтому в обработчиках прерываний наиболее распространенной методикой является прямое управление аппаратурой на уровне ячеек памяти и портов ввода/вывода. Так, вывод информации на экран терминала в процедурах обработки прерываний проводится путем непосредственного обращением к видеопамяти.

. Вывод информации на экран терминала в большинстве случаев целесообразно проводить путем непосредственного обращения к видеопамяти.

В текстовом режиме изображение обычно состоит из **25** строк по **80** символов в каждой строке. Каждый символ и фон под этим символом могут отображаться любым из **16** цветов. Текстовый видеобуфер адаптера **EGA\VGA** начинается с физического адреса **B8000h** и включает 8 видеостраниц по **4 КБт.** Активной, по умолчанию, является нулевая видеостраница, начальный адрес которой совпадает с начальным адресом видеобуфера. Каждый символ занимает в видеобуфере поле из **2–х** байт (см. рис. 3.7). Младшие (четные) байты всех полей отводятся под **ASCII – коды** отображаемых символов, старшие (нечетные) байты – под атрибуты символов. Биты **R, G, B** и **r, g, b** байта атрибутов задают составляющие красного, зеленого и синего цветов символа и фона соответственно. Биты I и i – яркость цветов символа и фона соответственно. Код, записываемый в видеобуфер, отображается на экране в виде изображения цветного символа на одном из знакомест.

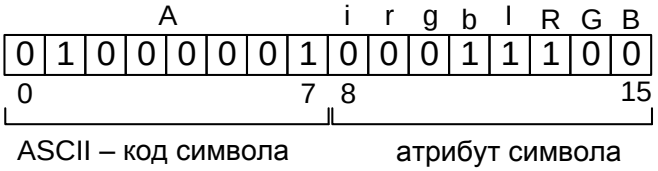


Рисунок 3.7 – Логическая организация текстового видеобуфера

При выводе символа на экран необходимо поместить в сегментный регистр (например, **ES**) сегментный адрес видеобуфера (B8000h/10h=**B800h**). Относительный адрес (четный) задает позицию размещения символа на экране. Первые **80** 2–х байтовых полей соответствуют первой строке экрана, вторые **80** 2–х байтных полей – второй и т.д. Пример вывода на экран символа ‘**A**’ на **4**–ю позицию нулевой строки экрана приводится ниже:

mov AX, 0B800h	загрузить в **AX** сегментный адрес видеобуфера;
mov ES, AX	настроить **ES** на сегментный адрес видеобуфера;
mov AL, 'A'	загрузить в **AL** код символа ‘**A**’;
mov AH, 00011100b	загрузить в **AH** атрибут символа ‘**A**’;

mov ES:[8], AX вывод на экран символ ‘**A**’ красного цвета на синем фоне.

После выполнения данного примера на экран в 4–й позиции нулевой строки будет выведен символ ‘**A**’ красного цвета на синем фоне.

Обмен информацией с внешними устройствами осуществляется с помощью портов ввода/вывода. Порт ввода–вывода – 8, 16 – разрядный регистр, имеющий определенный адрес в 64 кБт адресном пространстве ввода/вывода (см. рис.3.1). Взаимодействие прикладной программы с внешним устройством можно осуществлять с помощью «высокоуровневых» функций операционной системы или функций Базовой системы ввода/вывода **BIOS** (средний уровень), и непосредственно – прямым программированием портов, которое является единственным методом в случае применения нестандартных внешних устройств, или для достижения максимального быстродействия при обмене данными.

Для обращения к портам ввода/вывода в микропроцессоре предусмотрены специальные команды **in** и **out**:

in AL (AX), DX Ввод данных из порта с номером в DX в аккумулятор AX(AL);
out DX, AL(AX) Вывод данных из аккумулятора AX(AL) в порт с номером в DX;

Причем, регистр **AX** – используется при 16–и разрядной, а **AL** – при 8–и разрядной передаче данных. Если номер порта не превосходит 255, его можно указывать непосредственно. Примеры:

out 61h, AL Вывод в порт 61h содержимого **AL**.
mov DX, 378h Запись номера порта **378h** в **DX**;
in AL,DX ввод данных в аккумулятор **AL** из порта с номером в **DX**.

Подключаемые внешние устройства, как правило, содержат несколько портов ввода/вывода, каждый из которых предназначен для выполнения определенных функций, например, для передачи данных (регистр данных), адреса (регистр адреса), управляющих сигналов (регистр управления), получения информации о готовности к обмену данными (регистр состояния) и т.д. При этом адресация портов, используемых данным устройством, ведется относительно некоторого базового адреса.

3.1.1 Исследование сегментной структуры программы

Запустить файл **turbo.exe** из каталога **…TP7\BIN** и создать файл с текстом программы, назвав его, например, **lab11.pas**.

Ввести в разделе **var** описаний глобальных переменных программы следующие идентификаторы:

```
var
b1,b2: byte;
```

w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15: word;
l1,l2,l3,l4,l5,l6: longint;

Ввести в разделе описаний подпрограмм описания для двух процедур, выполняющих произвольные действия. Например:

```
procedure summa;
   begin
b1:=b1+b2;
   end;
procedure raznost;
   begin
b1:=b1-b2;
   end;
```

Определить с помощью операторов **seg** и **offset** сегментные и относительные адреса переменных **b1, b2, w1, w2** а также процедур **summa и raznost.** Записать значения сегментных и относительных адресов (раздельно) в двухбайтные переменные **w1 - w12**. Например:

mov AX, seg b1 загрузка сегментного адреса переменной **b1** в **AX**;
mov BX, offset b1 загрузка относительного адреса переменной **b1** в **BX**;
mov w1, AX сохранение сегментного адреса в переменной **w1**;
mov w2, BX сохранение относительного адреса в переменной **w2**.

Записать в двухбайтовые переменные **w13, w14, w15** значения регистров **CS, DS, ES** соответственно. Например:

mov w13,CS сохранение значения сегментного регистра **CS** в **w13**;
mov w14,DS сохранение значения сегментного регистра **DS** в **w14**;
mov w15,SS сохранение значения сегментного регистра **SS** в **w15**.

Значения переменных **w1 - w15 вывести** на экран монитора с помощью стандартных процедур Pascal **writeln**, например:

writeln('w1=',w1); вывод значения пременной w1 на экран монитора.

Записать в отчет текст программы, полученные результаты и их анализ. Полный текст программы приводится в Приложении **1.**

3.1.2 Исследование прямой адресации памяти на примере работы с видеобуфером

Вывести на экран монитора произвольный алфавитно-цифровой символ в позицию в соответствие с номером варианта (см. табл. 3.1).

Таблица 3.1 – Варианты заданий для работы с видеобуфером

№ варианта	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
№ строки	12	4	15	0	12	3	10	17	24	10	6	21	5	8	12	0	14	16	23	9
№ столбца	5	40	6	40	20	11	24	20	60	5	0	79	60	50	10	0	22	18	50	30

В соответствие со структурной организацией видеопамяти сегментный адрес видеобуфера в текстовом режиме равен **B800h**, а относительный адрес, однозначно определяющий позицию символа, определяется из формулы:

$$off = N*80*2 + M*2,$$

где **N** – номер строки экрана при нумерации строк от 0 до 24, **M** – номер столбца на экране при нумерации столбцов от 0 до 79.

Фрагмент программы, выводящей символ **C** в 12 строку экрана и 20 столбец (off=12*160+40=1960) приведен ниже:

mov AX, 0B800h загрузить в **AX** сегментный адрес видеобуфера;

mov ES, AX настроить **ES** на сегментный адрес видеобуфера;

mov AL, 'A' загрузить в **AL** код символа '**C**';
mov AH, 00001111b загрузить в **AH** атрибут символа '**C**';
mov BX, 1940 загрузить в **BX** смещение (номер позиции) символа '**C**';
mov ES:[BX], AX вывод на экран символа '**C**' белого цвета на черном фоне.

Полный текст программы приводится в Приложении **2.**

3.1.3 Получение практических навыков при работе с портами ввода/вывода

В ПЭВМ типа IBM PC/AT энергонезависимая память **CMOS** используется для хранения конфигурации аппаратных средств. Эта память состоит из набора ячеек (размером в байт), доступ к которым выполняется через порты ввода/вывода с адресами **70h** и **71h.**

Процедура чтения данных из ячейки **CMOS** состоит из 2–х шагов:

- запись в выходной порт с адресом **70h** номера ячейки, из которой необходимо считать информацию;
- считывание содержимого данной ячейки из входного порта с адресом **71h**.

Необходимо получить информацию об объеме установленной в ПЭВМ основной оперативной памяти из ячеек **CMOS 15h** и **16h**:

mov AL, 15h загрузить в регистр **AL** номер ячейки **CMOS 15h**;
out 70h, AL отправить в порт **70h** микросхемы **CMOS** номер ячейки;
in AL, 71h считать из порта **71h** микросхемы **CMOS** ячейку № **15h**;
mov byte ptr w, AL записать в младший байт **c** содержимое ячейки № **15h**;
mov AL, 16h загрузить в регистр **AL** номер ячейки **CMOS 16h**;
out 70h, AL отправить в порт **70h** микросхемы **CMOS** номер ячейки;
in AL, 71h считать из порта **71h** микросхемы **CMOS** ячейку № **16h**;

mov byte ptr w+1,AL записать в старший байт **с** содержимое ячейки № **16h**.

В результате выполнения данного фрагмента переменная w будет содержать объем (в килобайтах) основной оперативной памяти, установленной в ПЭВМ. Вывод значения переменной w на экран целесообразно проводить с использованием стандартных процедур Borland Pascal **writeln**. Полный текст программы определения объема установленной основной оперативной памяти в ПЭВМ приводится в Приложении **3**.

3.2 Изучение арифметических и логических команд МП i80x86

Для работы с целыми двоичными числами целочисленное АЛУ микропроцессора i80x86 поддерживает более десятка арифметических команд. Основными командами данной группы являются команды сложения, вычитания, умножения и деления.

Команды сложения и вычитания:

add op1,op2 команда **сложения** (addition), действие: **op1:=op1+op2;**

sub op2,op2 команда **вычитания** (subtraction), действие: **op1:=op1+op2**.

В качестве операндов допускается использовать регистры общего назначения или ячейки памяти размером в байт или слово. Операнды должны совпадать по разрядности и не могут одновременно быть ячейками памяти. Второй операнд может быть представлен непосредственным значением. Команды изменяют флаги переноса **CF**, переполнения **OF**, знака **SF** и нуля **ZF**.

add BH,15 **BH:=BH+15;**

sub SI, w **SI:=SI+w.**

Команды положительного и отрицательного приращения (инкремента и декремента):

inc op **инкремент (increment),** действие **op:=op+1;**

dec op **декремент (decrement),** действие **op:=op-1;**

В качестве операнда допускается использовать регистр или ячейку памяти размером в байт или слово. Команды **inc op** и **dec op** эквивалентны командам **add op,1** и **sub op,1** соответственно, за исключением того, что не изменяют флаг переноса **CF**, занимают меньший объем памяти и выполняются быстрее соответствующих универсальных команд сложения и вычитания.

Команда умножения целых чисел без знака:

mul op **умножение (multiply): AX:=AL*op$_8$, (DX,AX):=AX*op$_{16}$.**

Операнд, указываемый в команде, является одним из сомножителей. Он может быть регистром или ячейкой памяти размером в байт или слово, но не может быть непосредственным операндом. Местонахождение второго сомножителя фиксировано: при умножении байтов он берется из регистра **AL**, а при умножении слов из регистра **AX**. Местонахождение результата, имеющего разрядность **2n** при умножении **n**-значных чисел, также фиксировано и не указывается в команде: при умножении байтов результат размером в слово заносится в AX, при умножении слов результат размером в

двойное слово заносится в регистровую пару **(DX, AX)**, в **AX** – младшее слово, в **DX** – старшее слово. При значениях флагов **CF=OF=1** произведение занимает двойной формат, при **CF=OF=0** для произведения достаточно формата сомножителей.

Примеры команды умножения:

mul BL AX=AL*BL;

mul CX (DX,AX):=AX*CX.

Команда деления целых чисел без знака:

div op деление (divide): DX:=(DX,AX) mod op_{16}, AX:=(DX,AX) div op_{16}; AH:= AX mod op_8; AL:=AX div op_8.

Местонахождение первого операнда (делимого) фиксировано и явно в команде не указывается. Указывается только второй операнд, который может находится находиться в регистре или ячейке памяти, но не может быть непосредственным значением. При делении слова на байт делимое должно находиться в регистре **AX**, при делении двойного слова на слово делимое должно находиться в регистровой паре **(DX, AX)**, в **AX** – младшее слово, в **DX** – старшее слово. В области целых чисел получение точного результата от деления невозможно. Поэтому под делением подразумевается получение сразу двух величин: неполного частного **div** и остатка от деления **mod**, которые заносятся в младшую и старшую части делимого соответственно.

Примеры команды деления:

div BL AH:= AX mod BL; AL:=AX div BL;

div CX DX:=(DX,AX) mod CX, AX:=(DX,AX) div CX

Основными логическими командами микропроцессора являются:

or op1, op2 логическое **или**; действие: **op1:=op1 or op2**;

and op1, op2 логическое **и**; действие: **op1:=op1 and op2**;

xor op1, op2 **исключающее или**; действие: **op1:=op1 xor op2.**

Указанные команды используются для выполнения операций поразрядного маскирования: **and** – для установки нулей в заданных разрядах, **or** – для установки единиц, **xor** – для выяснения совпадений значений битов первого операнда с маской. В качестве первого операнда в командах **and**, **or**, **xor** можно использовать регистры общего назначения и ячейки памяти, в качестве второго – дополнительно можно использовать непосредственное значение. Оба операнда одновременно не могут быть ячейками памяти. Команды изменяют флаги **SF**, **ZF**, **PF**. Таблицы истинности, а так же примеры поразрядных логических операций приводятся в таблице 3.1.

Таблица 3.1 – Таблицы истинности, а так же примеры поразрядных логических операций

and (и)			or (или)			xor (искл. или)		
Вход		Выход	Вход		Выход	Вход		Выход
A	B	Q	A	B	Q	A	B	Q
0	0	0	0	0	0	0	0	0

0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

Пример поразрядного маскирования AND		Пример поразрядного маскирования OR		Пример поразрядного маскирования XOR	
op1	xxxxxxxx	op1	xxxxxxxx	op1	10100110
op2	10110101	op2	00010010	op2	00010010
Результат	x0xx0x0x	Результат	xxx1xx1x	Результат	10110100

Команда поразрядного инвертирования:

not op логическое **not** , действие: **op:= not (op)**,

выполняет изменение значений двоичных разрядов операнда на противоположные.

Для проведения контрольного сравнения битов целесообразно применять команду **test op1,op2**, которая полностью эквивалентна **and**, за исключением того, что результат операции логического умножения не сохраняется, а устанавливаются только флаги (**SF**, **ZF**, **PF**). Примеры команд:

or AL, 00100000 данная команда устанавливает единицу в 5–м бите **AL**;

and AH, 11011111 данная команда устанавливает нуль в 5–м бите **AH**.

Дополняют совокупность логических операций команды логического сдвига. Команда **shl op, n (CL)** осуществляет сдвиг влево всех битов операнда, а в младшие разряды добавляются нули. Старший бит операнда поступает в флаг переноса **CF**. В качестве первого операнда могут выступать регистры общего назначения и ячейки памяти, в качестве второго – непосредственное значение числа сдвигаемых битов, или занесенное в регистр–счетчик **CL.** Команда **shr op, n (CL)** выполняет сдвиг всех битов операнда вправо; механизм работы и синтаксис аналогичен **shl.** Пример:

shr BX, 5 Выполнение сдвига вправо всех разрядов в **BX** на 5 битов.

3.2.1 Изучение принципов логического анализа данных

Необходимо **разработать** программу, выводящую на экран значение каждой из 4-х тетрад машинного слова. Значения тестируемого машинного слова **w** выбираются в соответствие с номером варианта из таблицы 3.2.

Таблица 3.2 – Значения тестируемого машинного слова согласно вариантам

№ вар.	1	2	3	4	5	6	7	8	9	10

w	2582h	8243h	17F1h	6257h	8A54h	C421h	4E91h	3729h	A582h	2A8Bh
№ вар.	11	12	13	14	15	16	17	18	19	20
w	5791h	1A43h	F2E1h	6F51h	5A94h	C412h	4791h	6A29h	7582h	5D8Ch

Анализ тетрад слова основан на независимом выделении значений в младших и старших тетрадах байтов, входящих в слово, путем применения операций логического поразрядного "**и**" и логического сдвига. Пример программы анализа тетрад машинного слова **2832h** приводится ниже:

```
Program lab21;
var                          раздел описаний идентификаторов;
w:word;                      ввод в раздел описаний идентификатора
слова w; b1,b2,b3,b4:byte;   и идентификаторов его тетрад - b1,b2,b3,b4;
begin                        начало записи операторов программы;
asm                          начало записи ассемблерных операторов;
mov w, 2832h;                ввод значения 2832h в слово w;
mov AX,w                     копирование значения слова w в регистры
                             AX и DX

mov DX,AX                    для ускорения работы программы;
and AL,00001111b             получение значения младшей тетрады слова w;
mov b1,AL                    сохранение значения младшей тетрады слова w в
                             b1;
shr DL,4                     сдвиг вправо на 4 разряда значения в DL;
mov b2,DL                    сохранение значения второй тетрады слова w в
                             b2;
and AH,00001111b             получение значения третьей тетрады слова w;
mov b3,AH                    сохранение значения третьей тетрады слова w в
                             b3;
shr DH,4                     сдвиг вправо на 4 разряда значения в DH;
mov b4,DH                    сохранение значения старшей тетрады слова w в
                             b4;
end;                         завершение ассемблерного кода;
writeln ('b1=',b1,b2,b3,b4); вывод на печать значений b1,b2,b3,b4;
end.                         завершение программы.
```

Полный исходный текст программы приводится в Приложении **4.**

3.2.2 Изучение арифметических команд МП i80x86

Необходимо **разработать** программу, вычисляющую целое 16-разрядное значение арифметического выражения, согласно номеру варианта, указанному в таблице 3.3. Разрядность переменных – 8 бит. При вычислениях необходимо учесть старшинство арифметических операций.

Таблица 3.3 – Арифметические выражения согласно вариантам

№	Выражение	№	Выражение	№	Выражение	№	Выражение
1	A + B - C / D * E	6	A + B * C - D / E	11	A + B * C / D - E	16	A * B / C + D - E
2	A + B / C * D - E	7	A - B * C + D / E	12	A / B * C - D + E	17	A / B - C * D + E
3	A * B - C / D + E	8	A*B-C+D / E	13	A * B + C / D - E	18	A - B / C * D + E
4	A - B + C * D / E	9	A-B*C/D+ E	14	A / B - C + D * E	19	A * B / C - D + E
5	A - B / C + D * E	10	A - B * C / D + E	15	A - B * C + D * E	20	A - B + C - D * E

В качестве примера приводится текст программы, вычисляющей значения выражения **w:=b/c+a+d+e/f:**

program LAB22;
var раздел описаний идентификаторов;
a,b,c,d,e,f:byte; ввод в раздел описаний идентификаторов переменных;
w:word; ввод в раздел описаний результата - слова **w**;
begin начало записи операторов программы;
a:=10; b:=20; c:=5; d:=20; e:=10; f:=10; ввод исходных данных;
asm начало записи ассемблерного кода;
mov AL,b ввод младшего байта делимого **b** в регистр **AL**;
mov AH,0 обнуление старшего байта делимого в регистре **AH**;
div c команда деления **AL:=AX/c**;
add AL,a суммирование **AL:=AL+a**;
add AL,d суммирование **AL:=AL+d**;
mov CL,AL сохранение результата в регистре **CL**;

mov CH,0 обнуление старшего байта результата в регистре **CH**;
mov AL,e загрузка первого множителя **e** в регистр **AL**;
mul f умножение **AX:=AL*f**;
add AX,CX суммирование **AX:=AX+CX**;
mov w,AX сохранение результата в слове **w**;
end; завершение ассемблерного кода;
writeln ('w=',w); вывод на печать значения **w**;
end. завершение программы.

Полный текст программы приводится в Приложении **5**.

3.3 Обработка массивов на языке Assembler для МП i80x86

3.3.1. Принципы хранения массивов в памяти ПЭВМ, механизм базово-индексной адресации данных в МП i80x86

При описании массива стандартно указывается количество и тип элементов. При этом нумерация элементов, располагающихся в памяти последовательно, по умолчанию, начинается с нуля. Объем **S** памяти, занимаемой массивом, определяется как:

$$S = NT,$$

где **N** – количество элементов массива, T – тип (размер) элемента массива.

В общем случае при прямой адресации в команде вместе с адресом может быть указан некоторый регистр (**mov AX, w[BX]**). Данная команда будет работать не с указанным адресом w, а с исполнительным (эффективным) адресом $w_{исп}$, вычисляемым по формуле:

$$w_{исп} = (\mathrm{w} + (BX)) \bmod \cdot 2^{16},$$

где BX – обозначает значение в регистре BX.

Таким образом, прежде чем выполнить такую команду, центральный процессор прибавит к указанному адресу w текущее значение регистра BX, получит некоторый новый адрес и именно из ячейки с этим адресом возьмет второй операнд (если в результате сложения получилась слишком большая сумма, то от нее берутся только младшие 16 битов). Регистр BX носит название регистра-модификатора адреса.

В МП i80x86 разрешается в командах указывать два регистра–модификатора, причем один из них должен быть регистром **BX** или **BP**, а другой – регистром **SI** или **DI**. Такой вид адресации называется базовым-индексным (в регистры **BX** или **BP**, как правило, заносится **базовый** адрес некоторой области памяти, а в регистры **SI** или **DI** – **индексы** элементов данной области). Полный физический адрес операнда определяется как сумма содержимого следующих регистров:

DS:[BX][SI]	$\mathbf{A_{физ}}$**=DS*10h+BX+SI;**
DS:[BX][DI]	$\mathbf{A_{физ}}$**=DS*10h+BX+DI;**
ES:[BX][SI]	$\mathbf{A_{физ}}$**=ES*10h+BX+SI;**
ES:[BX][DI]	$\mathbf{A_{физ}}$**=ES*10h+BX+DI;**
SS:[BP][SI]	$\mathbf{A_{физ}}$**=SS*10h+BP+SI;**
SS:[BP][DI]	$\mathbf{A_{физ}}$**=SS*10h+BP+DI.**

В общем случае для массива с индексацией элементов, начинающейся от нуля [0..N-1], адрес i- элемента $\mathbf{A_i}$ определяется как:

$$\mathrm{A_i} = \mathrm{B} + \mathrm{i} * \mathrm{T},$$

где B – начальный (базовый) адрес массива, T – тип (размер) элемента массива.

Причем, значение базового адреса массива заносится, как правило, в регистр **BX**, а индекс элемента – в один из индексных регистров в **SI** или **DI**. Например:

var w: array[0..9] of word;	объявление массива из 10 элементов;
……	
mov ES, seg w	загрузить в **ES** сегментный адрес массива **w**;
mov BX, offset w	загрузить в **BX** относительный адрес массива **w**;
mov SI,10	загрузить в **SI** индекс 5-го элемента массива **w**;
mov AX, ES:[BX][SI]	загрузить в **AX** 5-й элемент массива **w.**

3.3.2. Команды сравнения, условного и безусловного перехода

При сравнении данных следует применять команду микропроцессора cmp op1,op2 формат которой совпадает с форматом рассмотренных в подразделе 2.1.2 логических команд. Анализ полученного результата можно проводить с использованием команд условного перехода, анализирующих флаги ZF, SF, CF, OF. Мнемокоды команд условного перехода, применяемых для сравнения беззнаковых чисел, применяемых после команды cmp, приводятся в таблице 3.4.

Таблица 3.4 – Команды условного перехода, используемые после команды сравнения CMP

Мнемокод команды	Условие для перехода по **cmp op1, op2**	Состояние флагов для перехода
JZ; JE / JNZ; JNE	**op1=op2 / op1<>op2**	**ZF=1 / ZF=0**
JB	**op1<op2**	**CF=1**
JBE	**op1<=op2**	**CF=1 or ZF=1**
JA	**op1>op2**	**CF=0 and ZF=0**
JAE	**op1>=op2**	**CF=0**

В качестве аналогов команд **JE/JNE** при логических операциях целесообразно применять команды **JZ/JNZ (ZF=1/ZF=0).** Пример: сравнить два слова A и B, если A>B то переход на метку **greater**:

mov AX, A	загрузка содержимого ячейки **A** в регистр **AX**;
cmp AX, B	команда сравнения содержимого **AX** и **B** ;
ja @greater	условный переход на метку greater, если **A>B**;
…….	пропускаемый, в случае невыполнения

условия, код;

@greater: метка указывает на выполняемые после нее команды.

Для выполнения безусловного перехода следует применять команду **jmp op**, где операнд указывает на адрес перехода, в качестве которого, как правило, указывается метка.

@lbl: метка **@lbl;**
…….. произвольный программный код;
jmp @lbl безусловный переход на метку **@lbl**.

3.3.3 Оператор цикла

С помощью команд перехода можно реализовать практически любые разветвления и циклы. Среди циклов на практике наиболее часто встречаются циклы с заранее известным числом повторений (for-циклы), поэтому в систему команд многих ЭВМ обычно включают дополнительные команды, упрощающие реализацию подобных циклов. В языке Assembler для МП i80x86 такая команда называется оператором цикла **loop op,** где операнд указывает на адрес перехода, в качестве которого, как правило, указывается метка. Оператор цикла **loop,** осуществляющий 10 итераций, эквивалентен следующему фрагменту программы.

mov CX, 10 **CX -** счетчик цикла (число повторений)
@m1: метка **@m1;**
…….. тело цикла (произвольный программный код);
dec CX уменьшение значения в **CX** на **1;**
cmp CX, 0 сравнение значения в **CX** с **0;**
jnz @m1 переход на метку **@m1**, если значение в **CX** не равно **0**.

Соответствующий цикл, организованный с помощью оператора **loop** приводится ниже:

mov CX, 10 **CX -** счетчик цикла (число повторений);
@m1: метка **@m1;**
…….. тело цикла (произвольный программный код);
loop @m1 переход на метку **@m1**, если значение в **CX** не

равно **0.**

Рассмотрим основные особенности практического использования оператора цикла **loop**. Во-первых, команда **loop** требует, чтобы в качестве счетчика цикла обязательно использовался только регистр **CX**. Во вторых, начальное значение для **CX** должно быть присвоено до цикла, причем это значение должно быть точно равно количеству повторений цикла. Например, если цикл должен выполняться **100** раз, то в регистр CX необходимо записывать именно **100**, а не 99 или 101. В третьих, если начальное значение в регистре CX равно 0, то в большинстве компиляторов Assemler операторы тела цикла не будут выполняться не одного раза.

Задание:

Необходимо разработать программу, выполняющую обработку массива данных. Варианты индивидуальных заданий представлены в таблице 3.5.

Таблица 3.5 – Варианты индивидуальных заданий

№ вар.	Задание
1	Дан массив M1 из 10 16-разрядних слов. Переписать из массива M1 в массив M2 все числа, значения которых меньше, чем 20H.
2	Дан массив M1 из 20 16-разрядних слов. Сформировать массив M2 из порядковых номеров Для чисел массива M1, значения которых больше 11h, сформировать массив M2 из их порядковых номеров.
3	Дан массив M1 из 20 16-разрядних слов. Определить количество элементов массива, значения которых больше, чем 10h
4	Дан массив M1 из 10 8-разрядних чисел. Вычислить среднее арифметическое элементов массива M1.
5	Дан массив M1 из 20 16-разрядних слов. Найти максимальный элемент массива M1.
6	Дан массив M1 из 10 16-разрядних слов. Определить количество элементов массива, значения которых больше, или равно 20h.
7	Дан массив M1 из 10 8-разрядних чисел. Сформировать массив M2 из чисел массива M1, значения которых больше 5Dh и меньше 0B5h.
8	Дан массив M1 из 10 16-разрядних знаковых чисел. Определить массив M2, содержащий все отрицательные элементы массива M1.
9	Дан массив M1 из 10 16-разрядних слов. Сформировать массив M2 из чисел массива M1, значения которых больше 205Dh и меньше 0C5h.
10	Дан массив M1 из 10 16-разрядних слов. Сформировать массив M2, выполнив взаимную перестановку старших и младших байтов числе массива M1.
11	Дан массив M1 из 10 8-разрядних чисел. Выполнить логическую сдвижку элементов массива M1 на 4 разряда влево и сформировать из полученных чисел массив M2.
12	Заданы массивы M1 и M2 из 10 16-разрядних слов каждый. Выполнить взаимную перестановку соответствующих элементов массивов M1 и

	M2.
13	Дан массив M1 из 10 8-разрядних знаковых чисел. Сформировать массив M2 из всех положительных и нулевых элементов массива M1.
14	Дан массив из 20 8-разрядних чисел, в котором находится несколько чисел 0B1H. Сформировать массив M2 из номеров этих чисел.
15	Дан массив из 10 8-разрядних чисел. Найти количество элементов массива, имеющих единицу во втором разряде.
16	Заданы массивы M1 и M2 из 10 8-разрядних чисел каждый. Сформировать массив M3, выполняя чередование элементов массивов M1 і M2.
17	Дан массив из 10 16-разрядних слов. Найти минимальный элемент массива M1.
18	Дан массив из 10 16-разрядних слов. Найти сумму элементов массива M1.
19	Дан массив из 10 16-разрядних слов, содержащий несколько нулевых элементов. Сформировать массив M2 из номеров этих элементов.
20	Дан массив из 10 8-разрядних чисел. Найти сумму чисел массива M1, значения которых больше 15h.

Пример выполнения задания приводится ниже. Дан массив m1 из 10 16-разрядних слов. Сформировать массив m2, значения которых больше 10h и меньше 20h. Блок схема алгоритма программы приводится на рисунке 3.7.

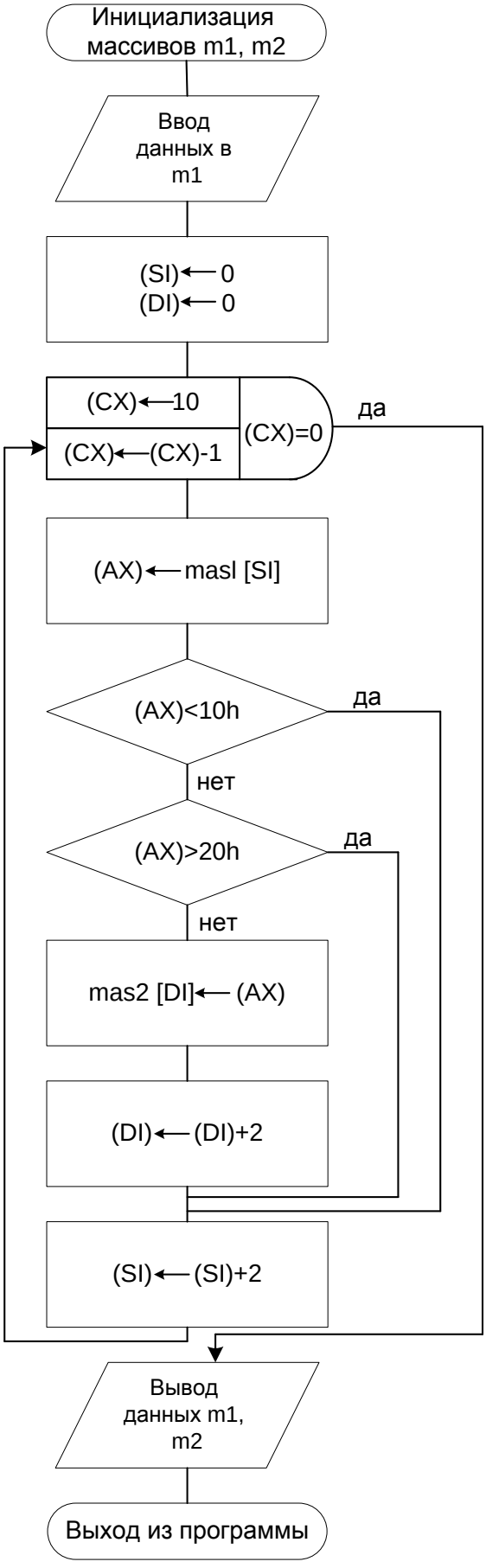


Рисунок 3.7 – Блок схема алгоритма программы обработки массива.

Основные фрагменты программы приводятся ниже:

mov SI,0	Установить начальное значение в **SI** равным **0**;
mov DI,0	установить начальное значение в **DI** равным **0**;
mov CX,10	записать в **CX** количество итераций цикла (**10**);
@M1:	метка цикла;
mov AX,word ptr m1[SI]	записать в **AX** элемент массива m1[SI];
cmp AX,10h	сравнить значение элемента массива **m1** и **10h**;
jb @M2	если **(AX) < 10h,** то преход на метку **@M2**;
cmp AX,20	сравнить значение элемента массива **m1** и **20h**;
ja @M2	если **(AX) > 20h**, то преход на метку **@M2**;
mov word ptr m2[DI],AX	присвоить элементу **m2[DI]** значение в **AX**;
add DI,2	добавить **2** к счетчику **[DI]** элементов **m2** ;
@M2:	метка условного перехода;
add SI,2	добавить **2** к счетчику **[SI]** элементов **m1**;
loop @M1	оператор цикла.

Ввод исходных данных и вывод результатов целесообразно выполнить средствами языка Turbo Pascal. Полный текст программы приводится в Приложении **6**.

3.4 Исследование принципов организации подпрограмм в языке Assembler для МП i80x86

3.4.1 Принципы функционирования стека в МП i80x86

В МП i80x86 стеком называют сегмент памяти, предназначенный для временного хранения произвольных данных и работающий по принципу LIFO (последним вошел – первым вышел.). В любой момент доступен лишь верхний элемент, т.е. элемент, загруженный в стек последним. Выгрузка из стека верхнего элемента делает доступным следующий элемент. Элементы стека располагаются в сегменте стека, начиная с максимального адреса (дна стека) по последовательно уменьшающимся адресам до вершины стека. Для работы со стеком предназначены три регистра: SS – сегментный регистр стека, SP – регистр указателя стека, в котором хранится адрес верхнего (доступного) элемента, BP – регистр указателя базы стека. Таким образом, регистровая пара SS:SP описывает полный физический адрес доступной ячейки стека.

Загрузка данных в стек осуществляется специальной командой push op. По этой команде сначала уменьшается на 2 содержимое указателя стека SP, а затем по адресу в SP помещается операнд op, который может быть регистром, или ячейкой памяти размеров в слово. Исходное состояние стека, а также стек после команд push AX и push BX приведены на рисунке 3.8 а), 3.8 б) и 3.8 в) соответственно.

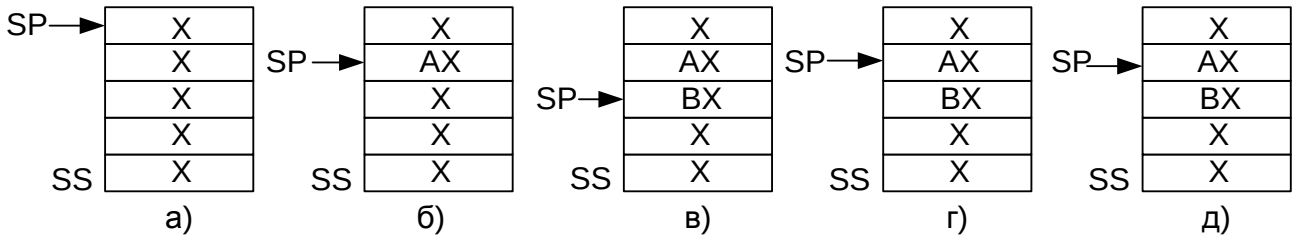


Рисунок 3.8 – Иллюстрация механизма работы стека: X – произвольное значение.

Для извлечения данных из стека используется команда pop op, по формату совпадающая с командой push op, но противоположная ей по действию. При выполнении команды pop op сначала происходит извлечение элемента из стека и запись его в op (регистр или ячейку памяти), а затем увеличение SP на 2. Состояния стека после команд pop BX и pop AX приводятся на рис. 3.8 г), и 3.8 д) соответственно. При этом данные в стеке не стираются, а изменяется лишь указатель стека SP. Для корректного восстановления содержимого регистров извлечение элементов из стека должно выполняться в порядке, строго противоположном загрузке:

Загрузка данных в стек	Извлечение данных из стека
push AX	pop CX
push BX	pop BX
push CX	pop AX

Запись и извлечение данных из стека осуществляется только словами. Произвольный доступ к ячейкам стека можно обеспечить, используя механизм сегментной базово-индексной адресации с использованием регистров: SS:[BP][SI], SS:[BP][DI].

Функции (процедуры) в языке Turbo Pascal описываются после раздела описания переменных var и имеют следующий формат:

function function_name**:** return_ value_type**;**
assembler; указываются имя функции и тип возвращаемого значения**;**

asm	начальная операторная скобка;
……	тело функции;
ret N	оператор выхода из процедуры;
end;	конечная операторная скобка.

Вызов функции осуществляется с помощью команды **call function_name**, которая записывает в стек адрес возврата в основную программу (адрес следующей за ней команды) и осуществляет переход на первую команду функции. В зависимости от дополнительно указываемых в команде **call** директив **far ptr** и **near ptr** в стек заносится полный адрес возврата, представленный относительным и сегментным адресами (дальний вызов подпрограммы), или только относительный адрес возврата (ближний вызов подпрограммы) соответственно. Например, по команде **call far ptr func1** при

переходе к выполнению кода функции func1 в стек заносится относительный и абсолютный адреса команды, следующей за **call.**

Перед вызовом функции необходимо передать соответствующие параметры. **Физически**, существуют два основных механизма передачи параметров: **через регистры** и **через стек.**

В случае передачи параметров через регистры, перед вызовом команды call параметры помещаются в регистры общего назначения (AX-DX, SI, DI) микропроцессора и затем используются в теле подпрограммы.

При передаче параметров через стек, перед вызовом команды call в стек необходимо последовательно загрузить параметры и затем в обратном порядке извлечь их в теле процедуры.

Логически, параметры могут передаваться **по значению** или **по адресу (по ссылке).** Это означает, что в регистры (или стек) могут передаваться как значения параметров, так и их адреса.

Возвращаемые значения в функциях всегда передаются через регистровую пару (**DX**, **AX**), в **DX –** старшее слово (если необходимо), в **AX –** младшее.

Для корректной работы функции необходимо обеспечить сохранение данных в регистрах, не используемых для передачи параметров, но модифицируемых в теле функции, а при выходе из процедуры **–** восстановить прежние значения.

Выход из процедуры осуществляется командой **ret N** с необязательным параметром N. Данная команда извлекает из стека адрес возврата в главную программу, помещенный командой **call**, и, в случае передачи параметров через стек, увеличивает значение указателя стека **IP** на **N байт**. В случае передачи параметров через регистры, параметр N в команде ret не используется.

Рассмотрим примеры: необходимо разработать программу, содержащую функцию, которая возвращает значение разности слов a и b. Параметры передаются: а) по значению через регистры, б) по значению через стек, в) по адресу через регистры, г) по адресу через стек.

Рассмотрим вариант а) при ближнем вызове функции:

function razn: word; assembler; описание функции **razn;**

asm	стартовая операторная скобка функции**;**
sub AX, BX	вычитание **a – b** и сохранение результата в **AX;**
ret;	извлечение из стека смещения следующей за **call** команды и возврат в основную программу**;**
end;	завершающая операторная скобка функции**;**

…….

фрагмент вызова функции основной программы:

mov AX, a	загрузка в **AX** параметра **a**;
mov BX, b	загрузка в **BX** параметра **b;**
call near ptr razn	в стек загружается смещение следующей за call команды и осуществляется переход на первый оператор функции **razn.**

Рассмотрим вариант б) при дальнем вызове функции:

function razn: word; assembler; описание функции **razn;**

asm стартовая операторная скобка функции;
код пролога функции:
push BP сохранение в стеке значения регистра **BP**;

mov BP, SP присвоение регистру **BP** значения в **SP**;
push BX сохранение в стеке значения регистра **BX**;

операционная часть функции:
mov AX, word ptr [BP+8] извлечение из стека в **AX** параметра **a**;
mov BX, word ptr [BP+6] извлечение из стека в **BX** параметра **b**;
sub AX, BX вычитание **a - b** и сохранение результата в AX;

код эпилога функции:
pop BX восстановление из стека значения регистра **BX**;
mov SP, BP восстановление прежнего значения **SP** из **BP**;
pop BP восстановление из стека значения регистра **BP**;
ret 4; извлечение из стека: полного адреса следующей за **call** команды, 2-х слов-параметров; и возврат в основную программу;
end; завершающая операторная скобка функции;
…….

фрагмент вызова функции основной программы :
push a загрузка в **AX** параметра **a**;
push b загрузка в **BX** параметра **b**;
call far ptr razn в стек загружается полный адрес следующей за **call** команды и осуществляется переход на первый оператор функции **razn**.

Рассмотрим вариант в) при дальнем вызове функции:
function razn: word; assembler; описание функции **razn**;
asm стартовая операторная скобка функции;

код пролога функции:
push SI сохранение в стеке значения регистра **SI**;
push DI сохранение в стеке значения регистра **DI**;

операционная часть функции:
mov SI, AX загрузка SI смещением переменной a из **AX**;
mov DI,BX загрузка **DI** смещением переменной b из **AX**;
mov AX, [SI] загрузка в **AX** слова (значения a) по адресу **DS**:**[SI]**;
mov BX, [DI] загрузка в BX слова (значения b) по адресу **DS:[DI**];

sub AX, BX вычитание a - b и сохранение результата в **AX**;
код эпилога функции:
pop DI восстановление из стека значения регистра **DI**;
pop SI восстановление из стека значения регистра **SI**;
ret; извлечение из стека смещения следующей за **call** команды и возврат в основную программу;
end; завершающая операторная скобка функции;

……

фрагмент вызова функции основной программы :
mov AX, offset a загрузка в **AX** относительного адреса параметра **a**;
mov BX, offset b загрузка в **BX** относительного адреса параметра **b**;
call far ptr razn в стек загружается полный адрес следующей за **call** команды и осуществляется переход на первый оператор функции razn.

Рассмотрим вариант г) при ближнем вызове функции:
function razn: word; assembler; описание функции **razn**;
asm стартовая операторная скобка функции;
код пролога функции:
push BP сохранение в стеке значения регистра **BP**;
mov BP, SP присвоение регистру **BP** значения в **SP**;
push DI сохранение в стеке значения регистра **SI**;
push SI сохранение в стеке значения регистра **DI**;
push BX сохранение в стеке значения регистра **BX**;

операционная часть функции:
mov SI, word ptr [BP+6] извлечение из стека в **SI** и **DI** относительных
mov DI, word ptr [BP+4] адресов параметров **a** и **b**;
mov AX, [SI] загрузка в **AX** слова (значения **a**) по адресу **DS:[SI]**;
mov BX, [DI] загрузка в **BX** слова (значения **b**) по адресу **DS:[DI]**;
sub AX, BX вычитание **a** - **b** и сохранение результата в **AX**;

код эпилога функции:
pop BX восстановление из стека значения регистра **BX**;
pop SI восстановление из стека значения регистра **SI**;
pop DI восстановление из стека значения регистра **DI**;
mov SP, BP восстановление прежнего значения **SP** из **BP**;
pop BP восстановление из стека значения регистра **BP**;
ret 4; извлечение из стека: относительного адреса следующей за call команды, 2-х слов-параметров; и возврат в основную программу;

end; завершающая операторная скобка функции;

……

фрагмент вызова функции основной программы :

mov AX, offset a загрузка в **AX** относительного адреса параметра **a**;

mov BX, offset b загрузка в **BX** относительного адреса параметра **b**;

push AX загрузка в стек относительного адреса параметра **a**;

push BX загрузка в стек относительного адреса параметра **b**;

call near ptr razn в стек загружается относительный адрес следующей за **call** команды и осуществляется переход на первый оператор функции **razn**.

Задание:

Необходимо разработать программу, выполняющую комплекс вычислений с помощью функции. Варианты индивидуальных заданий представлены в таблице 3.6.

Таблица 3.6 – Варианты индивидуальных заданий

№ вар.	Задания
1	Определить функцию для вычисления объема параллелепипеда. (V=a*b*c) и рассчитать суммарный объем параллелепипедов с длинами ребер (10, 15, 5), (15, 10, 20), (20, 10, 5). Передача параметров (по значению) осуществляется через стек.
2	Определить функцию для вычисления объема параллелепипеда (V=a*b*c) и и рассчитать суммарный объем параллелепипедов с длинами ребер (10, 15, 5), (15, 10, 20), (20, 10, 5). Передача параметров (по адресу) осуществляется через стек.
3	Определить функцию, которая возвращает максимальное из двух целых чисел и выполнить вычисление формулы: y=max(2, 5) + max(15, 8). Передача параметров (по значению) осуществляется через стек.
4	Определить функцию, которая возвращает максимальное из двух целых чисел и выполнить вычисление формулы: y=max(3, 8) + max(5, 7). Передача параметров (по значению) осуществляется через регистры.
5	Определить функцию, которая возвращает минимальное из двух целых чисел и выполнить вычисление формулы: y=min(4,6) + min(8, 2). Передача параметров (по адресу) осуществляется через стек.
6	Определить функцию, которая возвращает максимальное из двух целых чисел и выполнить вычисление формулы: y = min (4, 8)+ min(7, 4). Передача параметров (по значению) осуществляется через регистры.
7	Определить функцию, которая выполняет сравнение двух целых чисел, и возвращает номер (1 или 2) максимального из них. Проверить работу функции для наборов данных (10, 25), (80, 50). Передача параметров (по значению) осуществляется через стек.

8	Определить функцию, которая выполняет сравнение двух целых чисел, и возвращает номер (1 или 2) максимального из них. Проверить работу функции для наборов данных (30, 20), (10, 40). Передача параметров (по адресу) осуществляется через регистры.
9	Определить функцию, которая выполняет сравнение двух целых чисел, и возвращает номер (1 или 2) минимального из них. Проверить работу функции для наборов данных (40, 25), (15, 50). Передача параметров (по значению) осуществляется через стек.
10	Определить функцию, которая выполняет сравнение двух целых чисел, и возвращает номер (1 или 2) минимального из них. Проверить работу функции для наборов данных (10, 50), (5, 4). Передача параметров (по значению) осуществляется через регистры.
11	Определить функцию, которая выполняет сравнение трех целых чисел, и возвращает номер (1, 2, или 3) максимального из них. Проверить работу функции для наборов данных (30, 20, 10), (10, 50, 40). Передача параметров (по адресу) осуществляется через регистры.
12	Определить функцию, которая выполняет сравнение трех целых чисел, и возвращает номер (1, 2, или 3) максимального из них. Проверить работу функции для наборов данных (45, 14, 15), (2, 7, 8). Передача параметров (по адресу) осуществляется через стек.
13	Определить функцию, которая возвращает максимальное из трех целых чисел. Проверить работу функции для наборов данных (40,10,20), (5, 3, 10). Передача параметров (по адресу) осуществляется через регистры.
14	Определить функцию, которая возвращает максимальное из трех целых чисел. Проверить работу функции для наборов данных (5,15,10), (25, 30, 50). Передача параметров (по значению) осуществляется через стек.
15	Определить функцию, которая возвращает минимальное из трех целых чисел. Проверить работу функции для наборов данных (15,25,5), (20, 40, 80). Передача параметров (по адресу) осуществляется через регистры.
16	Определить функцию, которая возвращает минимальное из трех целых чисел. Проверить работу функции для наборов данных (30,20,40), (8, 70, 15). Передача параметров (по значению) осуществляется через регистры.
17	Определить функцию для вычисления площади круга (S~3*(r*r) и рассчитать площади для кругов с радиусами r=5, r=10, r=15. Передача параметров (по адресу) осуществляется через стек.
18	Определить функцию для вычисления длины окружности (P~6*r) и рассчитать сумму длин окружностей при r=5, r=10, r=15. Передача параметра (по адресу) осуществляется через регистр.
19	Определить функцию, которая возвращает 1 в случае, если сумма двух параметров размером в байт не превышает 255, и 0 – иначе. Проверить работу функции для наборов данных (100, 200), (120, 100). Передача параметров (по адресу) осуществляется через стек.
20	Определить функцию, которая возвращает 1 в случае, если сумма двух параметров размером в байт не превышает 100, и 0 – иначе. Проверить

	работу функции для наборов данных (100, 200), (200, 50). Передача параметров (по значению) осуществляется через регистры.

Пример выполнения задания: разработать программу, содержащую функцию, которая возвращает значение суммы слов a и b и помещает его в ячейку памяти c. Параметры передаются по адресу через стек.

Блок – схема алгоритма программы приведена на рисунке 3.9.

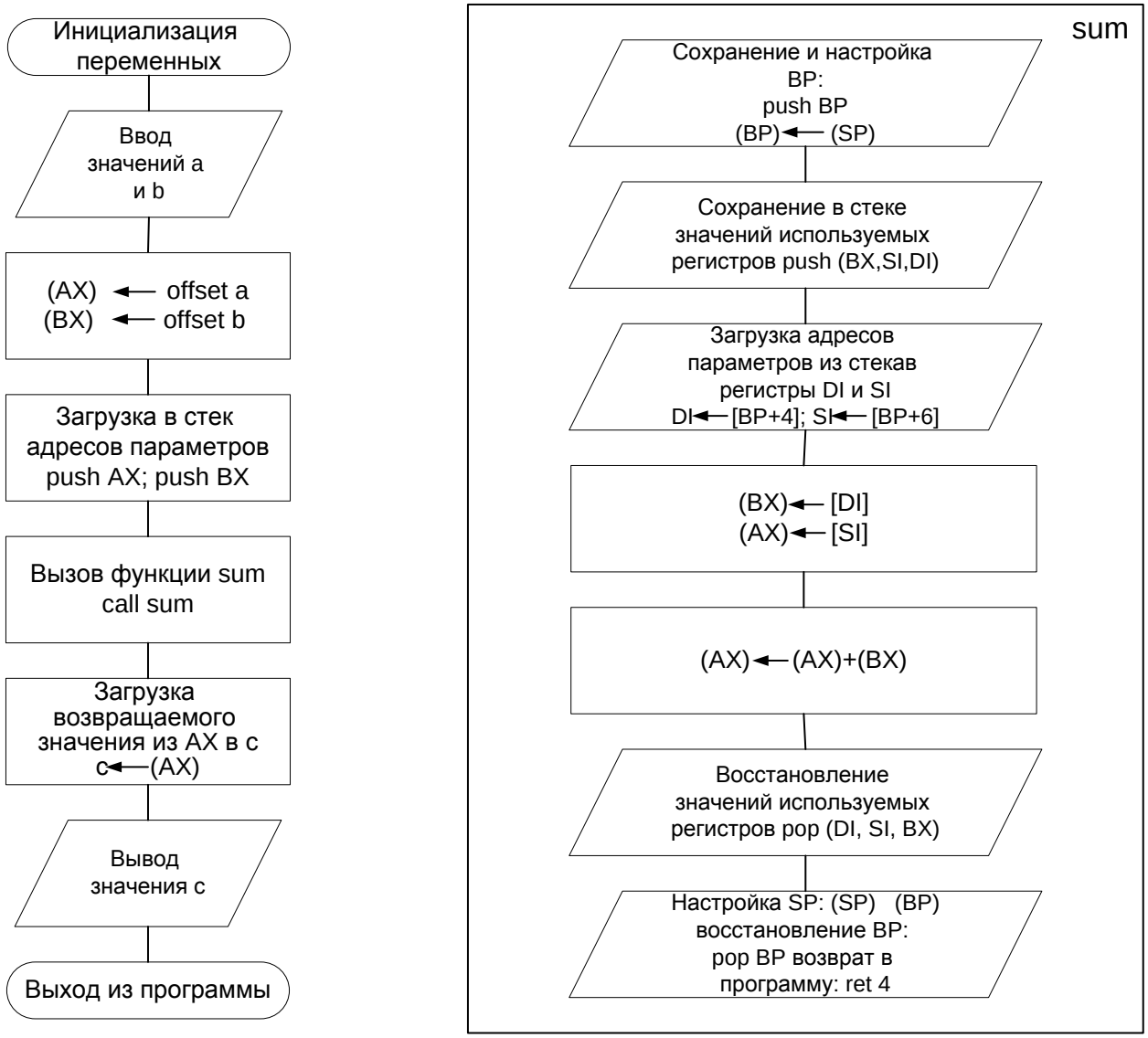


Рисунок 3.9 – Блок - схема алгоритма программы

Текст программы с комментариями приводится ниже:

Program LAB4;
var раздел описаний глобальных переменных;
a,b,c:word; описание переменных **a**, **b** и **c**;
function summa:integer;assembler; описание функции **summa;**
asm стартовая операторная скобка функции **summa**;
push BP сохранение в стеке значения регистра **BP**;
mov BP,SP присвоение регистру **BP** значения в **SP**;

push BX	сохранение в стеке значения регистра **BX;**
push SI	сохранение в стеке значения регистра **SI;**
push DI	сохранение в стеке значения регистра **DI;**
mov DI,[BP+4]	извлечение из стека в **SI** и **DI** относительных
mov SI,[BP+6]	адресов параметров **a** и **b;**
mov AX,[SI]	загрузка в **AX** слова (значения **a**) по адресу **DS:[SI];**
mov BX,[DI]	загрузка в **BX** слова (значения **b**) по адресу **DS:[DI];**
add AX,BX	сложение **a + b и** сохранение результата в **AX;**
pop DI	восстановление из стека значения регистра **DI;**
pop SI	восстановление из стека значения регистра **SI;**
pop BX	восстановление из стека значения регистра **BX;**
mov SP,BP	восстановление прежнего значения **SP** из **BP;**
pop BP	восстановление из стека значения регистра **BP;**
ret 4	извлечение из стека: относительного адреса следующей за **call** команды, **2-х** слов-параметров; и возврат в основную программу;
end;	завершающая операторная скобка функции **summa**;
begin	стартовый оператор основной программы;
asm	**стартовая операторная скобка ассемблерного кода;**
mov a, 100	загрузка численного значения по адресу **a**.
mov b, 200	загрузка численного значения по адресу **b**.
mov ax,offset a	загрузка в **AX относительного адреса параметра a;**
mov bx,offset b	загрузка в **BX относительного адреса параметра b;**
push AX	загрузка в стек относительного адреса параметра **a;**
push BX	загрузка в стек относительного адреса параметра **b**
call near ptr summa	в стек загружается относительный адрес следующей за **call** команды (**mov c,AX**) и осуществляется переход на первый оператор функции **summa.**
mov c,AX	**присвоение переменной с суммы (**a + b**) из** AX**;**
end;	**завершающая операторная скобка ассемблерного кода;**
end.	**завершающий оператор основной программы.**

Полный текст программы приводится в Приложении 7.

3.5 Изучение принципов функционирования микропроцессоров со стековой архитектурой на примере математического сопроцессора i80x87

3.5.1 Представление действительных чисел в формате с плавающей запятой

Числа в экспоненциальном формате (научная нотация) представляются в виде:

$$x = f \cdot 2^e,$$

где f – мантисса, e – порядок числа.

Поскольку в представлении числа участвуют два параметра f и e, такая форма записи является неоднозначной. Для устранения этой неоднозначности принято ограничивать диапазон допустимых значений мантиссы. При использовании двоичного представления целая часть мантиссы всегда будет содержать единицу, которую, следовательно, можно не хранить. Этот механизм называется использованием неявного старшего бита. Для представления знака мантиссы используется прямой код – модуль мантиссы и знаковый бит хранятся независимо. При представлении порядка используется excess - формат числа:

$$h = e + 2^{N-1} - 1,$$

где h – характеристика числа, N – разрядность характеристики.

В математическом сопроцессоре **i80x87** используются **3** формата представления чисел в формате с плавающей запятой (см. рис. 3.10). В **формате расширенной точности** для хранения данных отводится **80** битов (см. рис.3.10,а), из которых **64** бита отводится на мантиссу, **15** бит – на порядок и **1** - на знак. При этом число x представляется, как:

$$x = (-1)^s \cdot (m63 - m0) \cdot 2^{e-16383},$$

где s – знаковый разряд, m- разряды мантиссы. Причем в расширенном формате целая часть мантиссы, равная 1, не отбрасывается, а хранится в 63 бите.

В **формате с двойной точностью** для хранения данных отводится **64** бита (см. рис.3.10,б), из которых **52** бита отводятся на мантиссу, **11** битов – на порядок и **1** – на знак. При этом число x представляется, как:

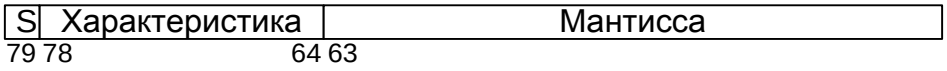
$$x = (-1)^s \cdot (1.m52 - m0) \cdot 2^{e-1023}.$$

Целая часть мантиссы в формате с двойной точностью отбрасывается.

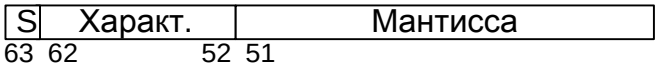
В **формате с одинарной точностью** для хранения данных отводится **32** бита (см. рис.3.10,в), из которых **23** бита отводятся на мантиссу, **8** битов – на порядок и **1** - на знак. При этом число x представляется, как:

$$x = (-1)^s \cdot (1.m22 - m0) \cdot 2^{e-127}.$$

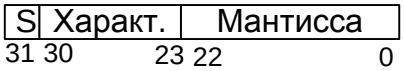
Целая часть мантиссы в формате с одинарной точностью отбрасывается.



а)



б)



в)

Рисунок 3.10 – Форматы представления действительных чисел в математическом сопроцессоре i80x87

Рассмотрим пример представления числа **7.375** при использовании формата одинарной точности: **7.375 =111.011b**. Хранимая мантисса будет равна **11011b**, а порядок **2+127=129=10000001b**. Так, как число положительное, знаковый бит равен **0**. Полное двоичное представление числа 7.375 в формате одинарной точности приведено на рисунке 3.11.

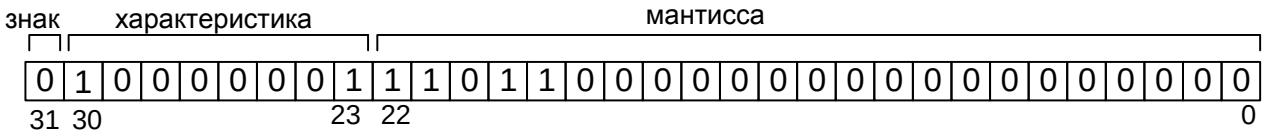


Рисунок 3.11 **–** Представление числа 7.375 в формате одинарной точности

3.5.2 Архитектура математического сопроцессора i80x87

Для обработки чисел с плавающей запятой в архитектуре вычислительных систем на базе микропроцессора i80x86 предусмотрен специализированный вычислительный блок – математический сопроцессор. Начиная с модели микропроцессора i486, математический сопроцессор выполняется в одном корпусе с центральным процессором и позволяет существенно (в 10 и более раз) увеличить скорость вычислений при выполнении операций над числами с плавающей запятой.

Сопроцессор **i80x87** аппаратно представлен следующим набором регистров (см. рис.3.12):

1. Восемь регистров данных **R0**-**R7** – составляющих основу стека сопроцессора;
2. Три служебных регистра: регистр управления **CWR**, регистр состояния SWR, регистр слова тегов **TWR**.

3. Регистры указатели данных **DPR, IPR** – используются для запоминания адреса команды, вызвавшей исключение.

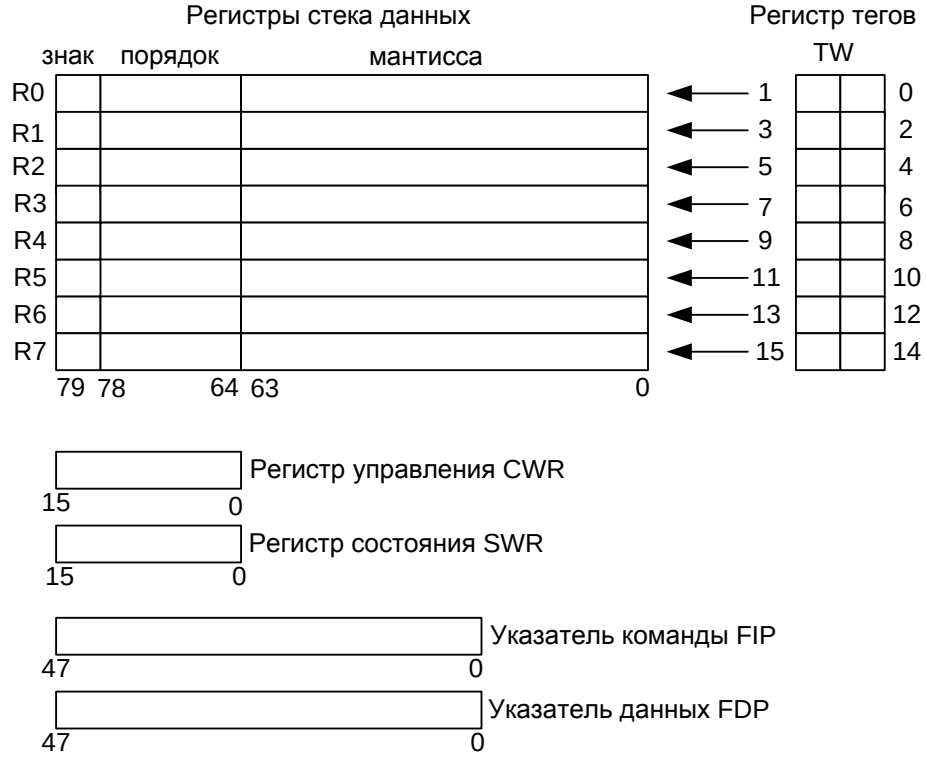


Рисунок 3.12 **–** Архитектура математического сопроцессора i80x87.

Регистровый стек сопроцессора организован по принципу кольца - нет регистра, аппаратно указывающего на вершину стека. Все регистры **R0-R7** функционально равноправны. Вершина стека является плавающей. Данный механизм реализуется с помощью системы команд сопроцессора, оперирующих не с физическими регистрами, а с логическими номерами этих регистров (**ST0-ST7**). В специально отведенном поле **top** регистра состояния **SWR,** фиксируется номер аппаратного регистра стека (**R0-R7**), являющегося в данный момент текущей вершиной стека. На рисунке 3.13,а показан пример, когда текущей вершиной стека является физический регистр **R3** (**ST0**), затем данные записываются в стек и текущей вершиной (см. рис. 3.13,б) становится физический регистр **R2** (**ST0**). В **16** – разрядном регистре тегов **TWR** хранятся **2**-разрядные значения признаков, по содержимому которых можно судить о наличии данных в физических регистрах **R0-R7.**

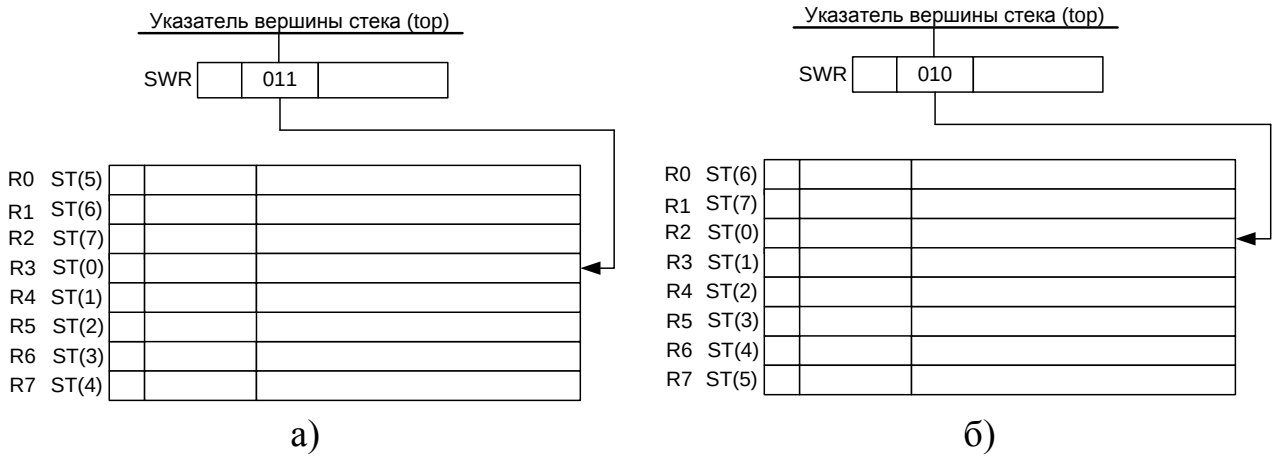


Рисунок 3.13 – Физическая и логическая нумерации регистров сопроцессора i80x87

Формат регистра статуса SWR приведен на рис. 3.14. Регистр статуса содержит 6 флагов исключительных ситуаций (exceptions flags), битов sf и es, отвечающих за корректность работы стека и общего функционирования сопроцессора соответственно, 4-х битов-флагов **c0-c3**, в которых отражаются признаки результата последней команды, поля **top**, в котором фиксируется номер аппаратного регистра стека (**R0-R7**), являющегося текущей вершиной стека.

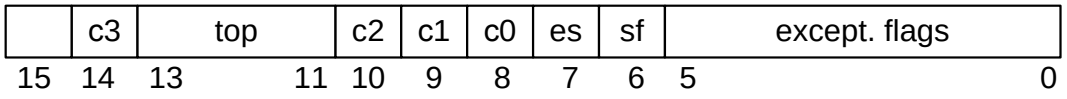


Рисунок 3.14 – Формат регистра статуса SWR сопроцессора i80x87

Формат регистра управления CWR сопроцессора i80x87 представлен на рис. 3.15. В регистре CWR содержатся 6 масок исключений для разрешения/запрета исключительных ситуаций, фиксируемых в регистре SWR, поле управления точностью вычислений pc (выбор форматов данных расширенной, двойной, одинарной точности), поле управления округлением результата операции.

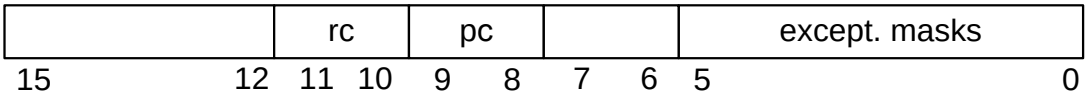


Рисунок 3.15 – ормат регистра управления CWR сопроцессора i80x87

Необходимо отметить, что внутренним форматом данных в сопроцессоре i80x87 является формат расширенной точности, а остальные форматы образуются путем округления.

3.5.3 Основные команды сопроцессора i80x87. Постфиксная форма записи математических выражений

В связи с тем, что в сопроцессоре **i80x87** используется стековая архитектура, формат его команд и принцип программирования несколько отличаются от классических представлений. Для оптимального использования стековой архитектуры сопроцессора применяется постфиксная запись арифметических выражений (сначала записываются операнды, а затем знак арифметического действия). Например, с учетом старшинства операций выражение:

A+B*C= BC*A+,

где слева – стандартная (инфиксная) форма записи, справа – соответствующая постфиксная.

Для упрощения работы с постфиксной нотацией математические выражения удобно представлять в виде деревьев, в узлах которых расположены операции, а в ветвях - операнды. При использовании сопроцессора необходимо поместить сначала операнды в стек, а затем выполнить действие.

Рассмотрим основные команды сопроцессора **i80x87.**

Команды для выполнения основных арифметических операций:

fadd - команда сложения : **(ST0)=(ST1+ST0);**

fsub - команда вычитания: **(ST0)=(ST1-ST0);**

fmul - команда умножения: **(ST0)=(ST1*ST0);**

fdiv - команда деления: **(ST0)=(ST1/ST0)**

fsqrt - команда извлечения квадратного корня: **(ST0)=sqrt(ST0).**

Команды пересылки данных:

fld m - команда загрузки в **ST0** значения из ячейки памяти **m: (ST0)=m;**

fst m - команда извлечения данных из **ST0** в ячейку памяти **m: m=(ST0);**

fstp m - команда извлечения данных из **ST0** в ячейку памяти **m: m=(ST0)**

с последующим удалением значения **m** из стека

В качестве **m** могут использоваться **32, 64, 80** разрядные ячейки памяти, описанные в Pascal как **single, double** и **extended**.

Рассмотрим пример выполнения арифметического действия **C=A+B** с использованием сопроцессора:

fld A загрузка в **ST0** значения в ячейке **A**;

fld B загрузка в **ST0** значения в ячейке **B**;

fadd выполнение сложения;

fstp C запись значения из **ST0** в **C** и удаление значения **C** из стека.

Иллюстрация изменения значений в регистрах сопроцессора во время проводимых действий приводится на рис. 3.16.

Рисунок.3.16 – Иллюстрация изменения значений в регистрах сопроцессора во время выполнения операции C=A+B

Задание:

Необходимо разработать программу, выполняющую расчет математического выражения с помощью математического сопроцессора в соответствие с вариантом, указанным в таблице 2.2.

Рассмотрим пример выполнения задания: необходимо вычислить значение выражения **L6:=(L1+L2)*L3+L4*L5.**

Для облегчения составления постфиксной записи выражения, составляем дерево обратной польской записи (см. рис. 3.17). Выполним обход дерева слева направо, при этом каждый узел рассматриваются только после обхода всех исходящих из него ветвей. Получаем выражение в постфиксной нотации:

L6:=(L1+L2)*L3+L4*L5 = L1 L2 + L3* L4 L5 * +

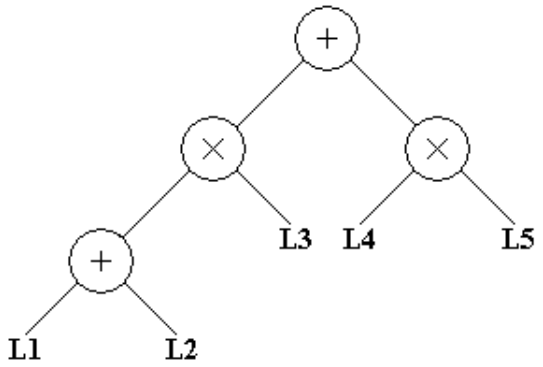


Рисунок.3.17 – Представление выражения l6:=(l1+l2)*l3+l4*l5; в виде дерева.

Вычислительная часть программы:

fld L1 загрузка в **ST0** значения из **L1**;

fld L2 загрузка в **ST0** значения из **L2**;
fadd суммирование;
fld L3 загрузка в **ST0** значения из **L1**;
fmul умножение;
fld L4 загрузка в **ST0** значения из **L1**;
fld L5 загрузка в **ST0** значения из **L1**;
fmul умножение;
fadd суммирование;
fstp L6 запись значения из **ST0** в **L6** и удаление значения **L6** из стека.

Иллюстрация работы регистров стека сопроцессора приводится на рис. 3.18.

Полный текст программы приводится в Приложении **8**.

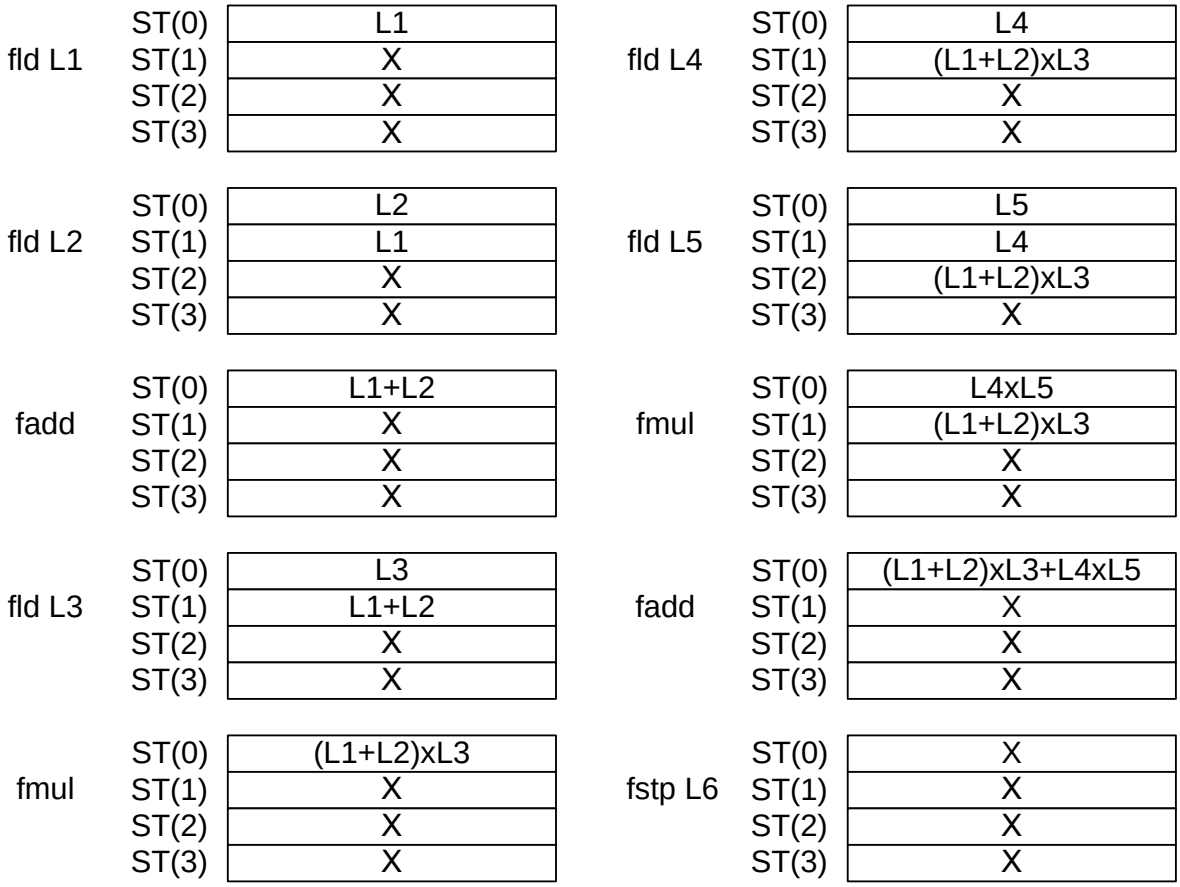


Рисунок 3.18 – Иллюстрация работы регистров стека сопроцессора при выполнении вычисления выражения L6:=(L1+L2)*L3+L4*L5

Вопросы для самопроверки

1. Каковы основные особенности архитектуры вычислительных систем на базе МП i80x86 в реальном режиме?
2. Перечислите регистры микропроцессора i80x86 и указать их функциональное назначение.
3. Поясните распределение адресных пространств памяти и устройств

ввода/вывода в МП i80x86.

4. Объясните сущность механизма сегментации адресов при обращении к памяти.

5. Приведите и поясните сегментную структуру программы.

6. Перечислите основные команды МП i80x86 при обращении к памяти и портам ввода/вывода.

7. Приведите и поясните форматы данных чисел со знаком и без знака.

8. Поясните назначение отдельных битов регистра флагов?

9. Приведите и поясните форматы и принципы использования команд сложения и вычитания беззнаковых чисел.

10. Приведите и поясните форматы и принципы использования команд умножения и деления беззнаковых чисел.

11. Приведите и поясните форматы и принципы использования логических команд для поразрядной обработки двоичных данных.

12. Приведите и поясните основные приемы и логического анализа двоичных данных.

13. Поясните операторы цикла с условиями досрочного выхода из цикла.

14. Поясните операторы условного и безусловного перехода.

15. Поясните принципы базово-индексной адресации.

16. Поясните каким образом осуществляется сравнение операндов.

17. Поясните принципы организация прямых и косвенных переходов.

18. Объясните принципы организации и работы стека в МП i80x86.

19. Объясните механизм передачи параметров подпрограмм через регистры.

20. Объясните механизм передачи параметров подпрограмм через стек.

21. Объясните различия между передачей параметров по адресу и по значению.

22. Для чего необходимы коды пролога и эпилога в подпрограммах?

23. В чем различие между дальним и ближним вызовами подпрограмм?

24. Перечислите и поясните форматы представления вещественных чисел в сопроцессоре i80x87.

25. Объясните особенности архитектуры математического сопроцессора i80x87.

26. Каким образом указываются операнды в командах математического сопроцессора i80x87?

27. Поясните принципы перехода от инфиксной к постфиксной нотации арифметических выражений.

28. Нужна ли синхронизация команд основного процессора и сопроцессора?

29. Перечислите и поясните основные команды сопроцессора математического сопроцессора i80x87.

СПИСОК ЛИТЕРАТУРЫ

1. Основы метрологии и электрические измерения / Б.Я. Авдеев, Е.М. Антонюк, Е.М. Душин и др.; Под ред. Е.М. Душина. – Л.: Энергоатомиздат, 1987. – 480 с.
2. Рубичев Н.А. Измерительные информационные системы: учебное пособие / Н.А. Рубичев. – М.: Дрофа, 2010. – 334 с.
3. Мирский Г.Я. Микропроцессоры в измерительных приборах. – М.: Радио и связь, 1984. – 160 с.
4. Измерительные приборы со встроенными микропроцессорами/ А.М. Мелик-Шахназаров, М.Г. Маркатун, В.А. Дмитриев. – М.: Энергоатомиздат, 1985. – 240 с. 1. Лысиков Б.Г. Арифметические и логические основы цифровых автоматов: Учебник.. – 2-е изд., перераб. и доп. – Мн.: Высш. школа, 1980. – 336 с.
5. Савельев А.Я. Арифметические и логические основы цифровых автоматов: Учебник. – М.: Высш. школа, 1980. – 272 с.
6. Корнейчук В.И., Тарасенко В.П. Основы компьютерной арифметики. – К.: «Корнійчук», 2003. – 176 с.
7. Самофалов К.Г., Корнейчук В.И., Тарасенко В.П., Жабин В.И. Цифровые ЭВМ. Практикум. – К.: «Вища школа», 1990, – 216с.
8. Самофалов К.Г., Корнейчук В.И., Тарасенко В.П. Цифровые ЭВМ. Теория и проектирование. – К.: «Вища школа», 1989, – 424с.
9. Водяхо А.И., СмоловВ.Б., Плюснин В.У., Пузанков Д.В. Функционально – ориентированные процессоры. –Л.: «Машиностроение», 1988, - 224с.
10. Самофалов К.Г., Романкевич А.М., Валуйский В.Р. и др. Прикладная теория цифровых автоматов.- К.: «Вища школа», 1987, – 375с.
11. Злобин В.К., Григорьев В.Л. Программирование арифметических операций в микропроцессорах: Учеб. пособие для технических вузов. – М.: Высш. шк., 1991 – 303с.
12. Рабинович З.Л., Раманаускас В.А. Типовые операции в вычислительных машинах.- К.: «Техніка», 1980, – 264с.
13. Оранский А.М. Аппаратные методы в ЦВТ. – Минск: Изд-во БГУ, 1977, – 208с.
14. Поспелов Д.А. Арифметические основы вычислительных машин дискретного действия. – М.: «Высшая школа», 1970, – 308с.
15. Синьков М.В., Губарени Н.М. Непозиционные представления в многомерных числовых системах.- К.: «Наукова думка», 1970, – 138с.
16. Карцев М.А. Арифметика цифровых машин. – М.: «Наука», 1969, – 576с.
17. Ричардс Ф.К. Арифметические операции на ЦВМ.- М.:Изд-во ИЛ, 1957, – 424с.
18. Брукшир Дж. Информатика и вычислительная техника. 7-е изд. – СПб.: Питер, 2004. – 620с.

19. Нортон П. Программно-аппаратная организация IBM PC: Пер. с англ. – М.: Радио и связь, 1991. – 328 с.

20. Шагурин И.И., Бердышев. Е.М. Процессоры семейства INTEL P6. Архитектура, программирование, интерфейс.- М.: Горячая линия – Телеком, 2000.- 248 с.

21. Финогенов К. Г. Самоучитель по системным функциям MS-DOS. – М.: Радио и связь, Энтроп, 1995. – , 382 с.

22. Пильщиков В.Н. Программирование на языке ассемблера IMB PC.– М.: ДИАЛОГ – МИФИ. – 2000.– 288с.

23. Рудаков П.И., Финогенов К. Г. Программируем на языке ассемблера IMB PC. Обнинск: Принтер, 1997.– 584с.

24. Юров В. Assembler. Спб; Питер, 2002. – 624с.

25. Зуев Е.А. Язык программирования Turbo Pascal 6.0, 7.0. – М.: Веста, Радио и связь, 1993. – 384с.

26. Фролов А.В., Фролов Г.В. Аппаратное обеспечение ПК.– М.:ДИАЛОГ-МИФИ, 1997. - 304с.

27. Андреева Е., Фалина И. Системы счисления и компьютерная арифметика. М.: Лаборатория базовых знаний, 1999. – 236с.

28. Гарднер М. Математические головоломки и развлечения. М.:АСТ, 2008. – 214с.

29. Гамаюн В.П., Шалаш Л.А. Алгоритм сжатия данных при обмене в измерительных комплексах // Нові комп’ютерні засоби, обчислювальні машини та мережі: Зб. наук. пр. / НАН України Ін-т кібернетики ім. В.М. Глушкова. – Т,1. – К., 2001. – С. 108– 116.

30. Гамаюн В.П. Квазиграфический метод вычисления остатка по модулю // Проблеми інформатизації тауправління. – Вип. 3(14). – К.: НАУ, 2005. – С.43 – 48.

31. Андреева Е.В., Босова Л.Л., Фалина И.Н. Математические основы информатики. М.: Бином, 2007.-220с.

ПРИЛОЖЕНИЯ

Приложение 1. Программа для исследования сегментной адресации

```
Program LAB11;
Var b1,b2:byte; w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15:word;
procedure summa; begin b1:=b1+b2; end;
procedure raznost; begin b1:=b1-b2; end;
begin
asm
mov AX,seg b1
mov BX,offset b1
mov w1,AX
mov w2,BX
mov AX,seg b2
mov BX,offset b2
mov w3,AX
mov w4,BX
mov AX,seg w1
mov BX,offset w1
mov w5,AX
mov w6,BX
mov AX,seg w2
mov BX,offset w2
mov w7,AX
mov w8,BX
mov AX,seg summa
mov BX,offset summa
mov w9,AX
mov w10,BX
mov AX,seg raznost
mov BX,offset raznost
mov w11,AX
mov w12,BX
mov w13,CS
mov w14,DS
mov w15,SS
end;
writeln('w1=',w1,'  w2=',w2);
writeln('w3=',w3,'  w4=',w4);
writeln('w5=',w5,'  w6=',w6);
writeln('w7=',w7,'  w8=',w8);
writeln('w9=',w9,'  w10=',w10);
writeln('w11=',w11,'  w12=',w12);
writeln('w13=',w13,'  w14=',w14,'  w15=',w15);
end.
```

Приложение 2. Программа для исследования прямой адресации памяти на примере работы с видеобуфером

```
Program LAB12;
begin
asm
mov AX,0b800h
mov ES,AX
mov Al,'C'
mov AH,10001100b
mov BX,0024h
mov ES:[BX],AX
end;
end.
```

Приложение 3. Программа для исследования доступа к портам ввода/вывода

```
Program LAB13;
var w:word;
begin
asm
mov AL,15h
out 70h,al
in AL,71h
mov byte ptr w,AL
mov AL,16h
out 70h,AL
in AL,71h
mov byte ptr [w+1],AL
end;
writeln('w=',w);
end.
```

Приложение 4.Программа для изучения принципов логического анализа данных

```
Program LAB21;
var
w:word;
b1,b2,b3,b4:byte;
begin
asm
mov w,2832h;
mov AX,w
mov DX,AX

and AL,00001111b
mov b1,AL
shr DL,4
mov b2,DL

and AH,00001111b
mov b3,AH
shr DH,4
mov b4,DH
end;
writeln ('b1=',b1,b2,b3,b4);
end.
```

Приложение 5. Программа для изучения арифметических команд МП i80x86

```
Program LAB22;
var
a,b,c,d,e,f:byte;
w:word;
begin
a:=10; b:=20; c:=5; d:=20; e:=10; f:=10;
asm
mov AL,b
mov AH,0
div c
add AL,a
add AL,d
mov CL,al
mov CH,0
mov AL,e
mul f
add AX,CX
mov w,AX
end;
writeln ('w=',w);
end.
```

Приложение 6. Программа для изучения принципов обработки массивов на языке Assembler для МП i80x86

```
Program LAB3;
var
i:byte;
mas1,mas2:array[0..9] of word;
begin
mas1[0]:=3;
mas1[1]:=10;
mas1[2]:=20;
mas1[3]:=15;
mas1[6]:=30;
mas1[5]:=14;
mas1[4]:=5;
mas1[7]:=8;
mas1[8]:=9;
mas1[9]:=19;
asm
mov cx,10
mov si,0
mov di,0
@M1:
mov ax,word ptr mas1[si]
cmp ax,10h
jb @M2
cmp ax,20
ja @M2
mov word ptr mas2[di],ax
inc di
inc di
@M2:
inc si
inc si
loop @M1
end;

for i:=0 to 9 do begin
writeln('mas1[',i,']=',mas1[i],'   mas2[',i,']=',mas2[i]);
end;
end.
```

Приложение 7. Программа для изучения организации процедур и функций программ на языке Assembler

```
Program LAB4;
var
a,b:integer;
y1,y2,y3:integer{shortint};
function sign:integer;assembler;
asm
push BP
mov BP,SP
push BX
push SI
push DI
mov DI,[BP+4]
mov SI,[BP+6]
mov AX,[SI]
mov BX,[DI]
add AX,BX
cmp AX,0
jge @M1
mov AX,-1
jmp @M3
@M1:
cmp AX,0
je @M2
mov AX,1
jmp @M3
@M2: mov AX,0
@M3:
pop DI
pop SI
pop BX
mov sp,bp
pop BP
ret 4
end;

begin
asm
mov a, 100
mov b, 100
mov ax,offset a
mov bx,offset b
```

```
push ax
push bx
call near ptr sign
mov y1,ax

mov a, -200
mov b, 100
mov ax,offset a
mov bx,offset b
push ax
push bx
call near ptr sign
mov y2,ax

mov a, -100
mov b, 100
mov ax,offset a
mov bx,offset b
push ax
push bx
call near ptr sign
mov y3,ax
end;
writeln('y1=',y1,'   y2=',y2,'   y3=',y3);
end.
```

Приложение 8. Программа для исследования системы команд математического сопроцессора

```
Program LAB5;
var
l1,l2,l3,l4,l5,l6:extended;
begin
l1:=2;
l2:=3;
l3:=4;
l4:=5;
l5:=6;
{l6:=(l1+l2)*l3+l4*l5;}
asm
fld l1
fld l2
fadd
fld l3
fmul
fld l4
fld l5
fmul
fadd
fstp l6
end;
end.
```

АЛФАВИТНЫЙ УКАЗАТЕЛЬ ТЕРМИНОВ

C

T

ВІДОМОСТІ ПРО АВТОРІВ (УПОРЯДНИКІВ)

1. Автор: Аврунін Олег Григорович
   1.1. Місце роботи: ХНУРЕ, каф. Біомедичної інженерії, доктор технічних наук, професор.
   1.2. Домашня адреса: 61024, Харків, вул. Пушкінська 79, кв.52.
   1.3. Номери телефонів: робочий 702-13-64 , моб.(093)151-55-10.

2. Автор: Носова Тетяна Віталіївна
   2.1 Місце роботи: ХНУРЕ, каф. Біомедичної інженерії, кандидат технічних наук, доцент.
   2.2 Домашня адреса: 61204, Харків, пр. Перемоги 70, кв.17.
   2.3 Номери телефонів: робочий 702-13-64, домашній 336-41-59.

3. Автор Запорожець Олег Васильович
   3.1. Місце роботи: ХНУРЕ, каф. Метрології та вимірювальної техніки, кандидат технічних наук, доцент.
   3.2. Домашня адреса: 61013, м.Харків, пров. Ново-Олександрівський, 5
   3.3. Номери телефонів: моб. 068-606-94-81

4. Автор Руденко Олег Григорович
   4.1 Місце роботи: ХНУРЕ, зав. каф. Електронних обчислювальних машин д.т.н., профессор.
   4.2 Домашня адреса:
   4.3 Номери телефонів:

5. Автор Руженцев Ігор Вікторович
   5.1 Місце роботи: ХНУРЕ, зав. каф. Метрології та вимірювальної техніки, доктор технічних наук, професор.
   5.2 Домашня адреса:
   5.3 Номери телефонів:

6. Автор Семенець Валерій Васильович
   6.1 Місце роботи: ХНУРЕ, каф. Метрології та вимірювальної техніки, доктор технічних наук, професор.
   6.2 Домашня адреса:
   6.3 Номери телефонів:

7. Автор Токарєв Володимир Володимирович
   2.1 Місце роботи: ХНУРЕ, каф. Електронних обчислювальних машин, кандидат технічних наук, доцент
   2.2 Домашня адреса: 61204, Харків, пр. Перемоги 70, кв.17.
   2.3 Номери телефонів: моб. 067-26-41-788.

8. Назва рукопису: Навчальній посібник з курсу “Мікропроцесори в інформаційно-вимірювальних систмах” для студентів спеціальностей «Метрологія та інформаційно-вимірювальні технології», «Метрологія, стандартизація та сертифікація», «Комп'ютерна інженерія», «Біомедична інженерія».

Учебное издание

АВРУНИН Олег Григорьевич, ЗАПОРОЖЕЦ Олег Васильевич, НОСОВА Татьяна Витальевна, РУДЕНКО Олег Григорьевич, РУЖЕНЦЕВ Игорь Викторович, СЕМЕНЕЦ Валерий Васильевич,ТОКАРЕВ Владимир Владимирович

МИКРОПРОЦЕССОРЫ В ИНФОРМАЦИОННО-ИЗМЕРИТЕЛЬНЫХ СИСТЕМАХ

Учебное пособие

Ответственный за выпуск И.В. Руженцев

Редактор

План 2015, поз.
Підп. до друку 00.00.00. Формат 60×84 1/16. Спосіб друку – ризографія.
Умов.друк.арк. Облік. вид.арк. . Тираж прим.
Зам. № Ціна договірна.

ХНУРЕ. Україна. 61166, Харків, просп. Леніна, 14

Надруковано в учбово-виробничому
видавничо-поліграфічному центрі ХНУРЕ
61166, Харків, просп. Леніна, 14