МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Омский государственный технический университет»

В. Ф. Нестерук

И Н Ф О Р М А Т И К А

Учебное текстовое электронное издание
локального распространения

Омск
Издательство ОмГТУ
2016

Сведения об издании: 1, 2

ISBN 978-5-8149-2197-0

УДК 004.382.7(075)
ББК 32.973.26-04я73
Н56

Рецензенты:
А. П. Загородников, канд. техн. наук, директор ООО «АИСистемс»;
М. Ф. Шакиров, канд. техн. наук, доцент, БПОУ «ОМАВИАТ»

Нестерук, В. Ф.

Н56 Информатика : консп. лекций / В. Ф. Нестерук ; Минобрнауки России, ОмГТУ. – Омск : Изд-во ОмГТУ, 2016.

ISBN 978-5-8149-2197-0

Приводятся краткие теоретические сведения по терминологии алгебры логики и минимизации логических функций с помощью карт Карно. Рассматриваются примеры графического представления алгоритмов выполнения арифметических операций над двоичными кодами чисел и примеры машинной реализации этих операций в системе инструкций гипотетического операционного устройства.

Предназначен для студентов, обучающихся по направлению подготовки «Информатика и вычислительная техника».

УДК 004.382.7(075)
ББК 32.973.26-04я73

Рекомендовано редакционно-издательским советом
Омского государственного технического университета

ISBN 978-5-8149-2197-0

1 электронный оптический диск

Оригинал-макет издания выполнен в Microsoft Office Word 2007
с использованием возможностей Adobe Acrobat X.

Минимальные системные требования:

- процессор Intel Pentium 1,3 ГГц и выше;
- оперативная память 256 Мб;
- свободное место на жестком диске 260 Мб;
- операционная система Microsoft Windows XP/Vista/7;
- разрешение экрана 1024×576 и выше;
- акустическая система не требуется;
- дополнительные программные средства Adobe Acrobat Reader 5.0 и выше.

Редактор *Т. А. Москвитина*
Компьютерная верстка *О. Н. Савостеевой*
Сводный темплан 2016 г.
Подписано к использованию 10.04.16.
Объем 585 Кб.

———————

Издательство ОмГТУ.
644050, г. Омск, пр. Мира, 11; т. 23-02-12
Эл. почта: info@omgtu.ru

Общие сведения

Предметом дисциплины являются отдельные аспекты информатики в рамках направлений

09.03.01 «Информатика и вычислительная техника»
09.03.02 «Информационные системы и технологии»
09.03.03 «Прикладная информатика»
09.03.04 «Программная инженерия»
02.03.03 «Математическое обеспечение и администрирование информационных систем»

Целью изучения дисциплины является формирование у студентов знаний в области истории, современных направлений развития и фундаментальных методов и принципов информатики и вычислительной техники, способствующих готовности выпускника к решению задач, связанных с разработкой перспективных информационных систем.

Основные задачи дисциплины:

1) приобретение студентами необходимых знаний в области теоретических основ информатики и путей развития информатики и средств вычислительной техники;

2) изучение основных принципов организации и функционирования различных поколений и архитектур ЭВМ;

3) изучение общих подходов к организации машинного выполнения арифметических и логических операций;

4) привитие способности применения полученных знаний и практических навыков в последующей профессиональной деятельности.

В настоящее время под информатикой понимается наука, изучающая процессы сбора, кодирования, передачи, накопления, обработки, отображения информации и использования информации в социальных и технических системах.

Основы теории информации

Информация имеет определенные функции. Основными из них являются:

• познавательная – получение новой информации. Функция реализуется в основном через такие этапы обращения информации, как:

– ее синтез (производство),

– представление,

– хранение (передача во времени),

– восприятие (потребление);

• коммуникативная – функция общения людей, реализуемая через такие этапы обращения информации, как:

– передача (в пространстве),

– распределение;

• управленческая – формирование целесообразного поведения управляемой системы, получающей информацию. Эта функция информации неразрывно связана с познавательной и коммуникативной и реализуется через все основные этапы обращения, включая обработку.

В каждой отрасли народного хозяйства наиболее значимым является тот или иной аспект общего направления, например, в области разработки и применения измерительной аппаратуры – повышение точности съёма информации с объекта измерения; в области вычислительной техники – повышение быстродействия ЭВМ, точности вычислений, увеличение объёма средств хранения информации; в области машинной графики и дизайна – наглядность, точность и скорость отображения и пр.

Поэтому в рамках данной дисциплины будут затрагиваться наиболее общие вопросы, касающиеся перечисленных выше направлений.

Нельзя рассматривать информатику, не определив базовое понятие «информация».

На **содержательном уровне** под информацией мы понимаем представимые для нас свойства объектов или процессов, позволяющие идентифицировать эти объекты и процессы и использовать полученные сведения для каких-либо целей. Например, прежде чем пить чай, мы должны убедиться, холодный он или горячий, сколько его содержится в стакане и т. п.

Следовательно, информация для нас несёт как количественные, так и качественные характеристики об объекте или событии.

Причём эти характеристики об одном и том же объекте могут быть различными, в зависимости от того, что нас в данный момент интересует. Например, камень может быть тяжёлый, если нас интересует его вес, либо цветной, если мы планируем его использовать для инкрустации. То есть любой объект имеет некоторое множество качественных и количественных характеристик, из которых в данный момент для его идентификации нам достаточно некоторого подмножества из данного множества.

Количественные характеристики информации на содержательном уровне могут измеряться в различных единицах (литрах, граммах, километрах и пр.) в зависимости от того, какие качественные параметры нас интересуют.

Формальный подход требует определённого абстрагирования и унификации. Проблема формализации понятия «информация» появилась с развитием систем автоматизированной обработки и передачи данных.

Статистическая мера информации

В основу количественной теории информации положена формула К. Шеннона

$$H = -\sum_{i=1}^{i=n} p_i \cdot \log p_i, \text{ для } p_i = 0,...,1,$$

предложенная им в работе «Математическая теория связи», где H – энтропия системы событий; p_i – вероятность появления *i*-го события.

Энтропия трактуется как мера неопределённости системы. Для полностью определённой системы при $p_i = 1$ энтропия равна 0 и любые дополнительные сведения о системе не будут нести информацию.

Свойства энтропии:

- так как вероятности событий не более 1, энтропия всегда положительна;
- энтропия равна 0, когда какая-либо вероятность $p_i = 1$;
- энтропия максимальна при $p_i = 1/n$;
- энтропия составного объекта равна сумме энтропий всех объектов.

Чем выше энтропия, тем информация о системе, необходимая для придания ей определённости, должна быть более полной, что позволит свести p_i в пределе к 1. Поэтому К. Шеннон предложил рассматривать H как «разумную количественную меру возможности выбора, или меру количества инфор-мации».

При равной вероятности исходов событий $p_i = 1/n$ величина H принимает максимальное значение:

$$H = \log n.$$

Количество информации, полученное о системе, может быть определено разностью между начальной и конечной энтропией системы до и после опыта

$$I = H1 – H2,$$

Для полностью определённой системы I = H.

Структурная мера информации

Структурная мера определяет строение массива информации из информационных элементов, количество элементов в массиве и предполагает дискретное строение информационного сообщения.

При структурном подходе различают аддитивную, геометрическую и комбинаторную меры.

Аддитивная мера (мера Хартли) предполагает измерение информации в битах. Под битом понимается количество информации для события, имеющего два равновероятных значения. Соответственно, при n = 2

$$H = \log_2 n = 1.$$

Для системы с n равновероятными событиями формула Хартли для измерения информации в битах имеет вид

$$H = \log_2 n.$$

Геометрическая мера предусматривает отображение возможных состояний в виде вершин n-мерного куба.

Определяет количество информации по числу информационных элементов (квантов), занимающих определенную область информационного поля (длина, площадь, объём).

Информационная емкость – максимальное количество квантов, размещаемых в заданных структурных габаритах.

Например, для трехмерного информационного поля, имеющего по осям координат протяженности X, Y, Z и имеющего шаги квантования по осям DX, DY, DZ информационная емкость будет определяться формулой

$$M = mx \cdot my \cdot mz,$$

где mx , my, mz – число квантов по соответствующим осям, соответственно равное X/DX, Y/DY, Z/DZ.

Комбинаторная мера. Количество информации в комбинаторной мере вычисляется как количество комбинаций элементов.

1. Сочетания из h элементов по *l* различаются составом элементов. Их возможное число равно:

$$Q = C_h^l = h \cdot (h - 1)\ldots(h - l + 1)/l!$$

Например, число сочетаний из трехбуквенного алфавита А, Б, В по 2 будет равно 3: АБ, АВ, БВ.

2. Перестановки h элементов различаются их порядком. Число возможных перестановок h элементов:

$$Q = \Pi_h = h!$$

Например, число перестановок букв А, Б, В будет равно 6: АБВ, АВБ, БАВ, БВА, ВАБ, ВБА.

3. Размещения из h элементов по *l* различаются составом элементов и их порядком. Возможное число:

$$Q = P_h^l = h \cdot (h - 1)\ldots(h - l + 1).$$

Например, число размещений букв А, Б, В по 2 будет равно 6: АБ, БА, АВ, ВА, БВ, ВБ.

При применении комбинаторной меры возможное количество информации Q заключается не в простом подсчете квантов, как в геометрическом представлении, а в определении количества осуществляемых комбинаций.

Семантическая мера

Данная мера учитывает содержательность, целесообразность, ценность, существенность или полезность информации. Применяется при оценке эффективности логического опыта путем введения:

– степени истинности или ложности информации;

– пустой (нулевой), добротной (положительной) информации и дезинформации (отрицательной);

– динамической энтропии – положительная информация уменьшает неопределенность ситуации, отрицательная (дезинформация) – увеличивает;

– функции существенности информации для отражения степени ее важности с учетом времени и пространства;

– тезауруса («запаса знаний») для самообучения и восстановления информации в приемнике при передаче в условиях шумов.

Качественная теория информации

Несмотря на то, что слово «информация» широко используется, формального определения понятия «информация» до сих пор не существует, хотя предпринимаются попытки решить эту проблему. В основном под теориями информации понимаются теории измерения количественных характеристик объёма передаваемых сведений.

Одним из интересных подходов является качественная теория информации, предложенная М. Мазуром (Мазур, М. Качественная теория информации. – М. : Мир, 1974. – 240 с.). Рассматривая типовой замкнутый контур управления, в форме «система управления <– >объект управления» М. Мазур предлагает как управляющую цепь, так и цепь обратной связи рассматривать как цепь управления от источника воздействия к приёмнику воздействия. В одном случае источник воздействия – это система управления, а приёмник – объект управления, в другом случае источник воздействия – это объект управления, а приёмник воздействия – это система управления. Физическое состояние в цепи управления, отличающееся от других возможных физических состояний, является сообщением.

В цепи управления от источника к приёмнику могут быть дополнительные преобразования сообщений как заранее известные, например, электрический сигнал → оптический сигнал, так и непредусмотренные, например, помеха. Множество сообщений в цепи управления разделяется на поперечное множество и продольное множество.

Поперечное множество – это множество сообщений в конкретном месте цепи, а продольное множество формируется из поперечных в ходе конкретного процесса передачи от источника к приёмнику:

$$X1 \rightarrow Y1 \rightarrow Z1 \text{ (первое продольное множество)}$$
$$X2 \rightarrow Y2 \rightarrow Z2 \text{ (второе продольное множество)}$$
$$X3 \rightarrow Y3 \rightarrow Z3 \text{ (третье продольное множество),}$$

где X, Y, Z (i = 1, …, 3) – поперечные множества;

X – множество оригиналов;

Y – множество промежуточных сообщений;

Z – множество образов.

В частном случае, если этого не требуется и нет непредусмотренных воздействий, промежуточные сообщения могут отсутствовать. В общем случае, процесс передачи рассматривается как множество преобразований в продольном множестве первичных сообщений во вторичные.

Ассоциация сообщений – любая неупорядоченная пара сообщений из продольного или поперечного множества. Преобразование – трансформация одного из сообщений пары в другое.

Кодовая ассоциация – ассоциация преобразований в последовательном множестве. Соответственно, код – преобразование в кодовой ассоциации. Например, K1xy, K2xz и пр. Соответственно, Y1 = K1xy X1, Z2 = K2xz, X2 = = K2xy K2yz X2.

Информационная ассоциация – ассоциация преобразований в поперечном множестве.

Информация – преобразование в информационной ассоциации. Например, Ix12, Iy13.

Информирование – преобразование информации в цепи оригиналов в информацию в цепи образов. Для наблюдателя или приёмника изменение образа несёт информацию об изменении оригинала.

Соответственно, различают следующие виды информирования:

Трансинформирование – когда информация в множестве образов такая же, как в множестве оригиналов, является правильным информированием, т. е. Ix12 = Iz12.

Дезинформирование – когда некоторые кодовые цепи не полны. Соответственно, дезинформация – информация в множестве образов, искажающая её в результате дезинформирования.

Симуляционное дезинформирование – когда некоторые кодовые цепи не содержат оригиналов. Соответственно, симуляционная дезинформация – искажение информации через симуляционное дезинформирование.

$$X1 \rightarrow Y1 \rightarrow Z1$$
$$Iz12$$
$$Y2 \rightarrow Z\,2$$

Диссимуляционное дезинформирование – когда некоторые кодовые цепи не содержат образов. Соответственно – диссимуляционная дезинформация.

X1 → Y1 → Z1
Ix12
X2 → Y 2

Конфузионное дезинформирование – сочетание симуляцинного и диссимуляционного дезинформирования.

X1 → Y1 → Z1
Ix12
X2 → Y2
Iz13
Y3 → Z3

Параинформирование – сообщение, принадлежащее информационной цепи, но не принадлежащее ни одной из кодовых цепей.

X1 –> Y1–> Z1
Ix12 Iz12
X2 Z2

Метаинформирование – информирование в разных контурах управления, имеющих связи между информациями в ассоциациях оригиналов. Для таких контуров управления связи между информациями в ассоциациях образов отличаются от связей информацией в ассоциациях оригиналов только используемыми кодами.

Информационные системы

Информационная система (ИС) как объект информатики. ИС – это взаимосвязанная совокупность информационных, технических, программных, математических, организационных, правовых, эргономических, лингвистических, технологических и других средств, а также персонала, предназначенная для сбора, передачи, хранения, обработки и выдачи информации и принятия управленческих решений.

Свойства информационных систем:

– любая ИС должна быть доступна анализу и синтезу на основе общих принципов построения сложных систем;

– ИС является динамичной и развивающейся системой;

– выходной продукцией ИС является информация, на основе которой принимаются решения или производится автоматическое выполнение рутинных операций;

– участие человека зависит от сложности системы, типов и наборов данных, степени формализации решаемых задач.

Процессы в информационной системе:

– получение и преобразование информации из внешних и внутренних источников;

– ввод информации;

– обработка входящей информации;

– хранение информации для последующего ее использования;

– вывод информации в удобном для пользователя виде;

– обратная связь для воздействия на объект управления (для замкнутых ИС).

С учетом сферы применения выделяют: технические ИС, экономические ИС, ИС в гуманитарных областях и т. д.

В соответствии с вышеизложенным, ИС может быть определена как объект информатики.

Структура информационной системы. С технической точки зрения ИС в общем случае представляет контур управления (рис. 1).

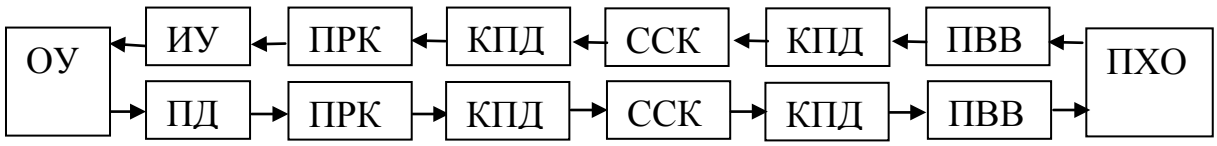

Рис. 1

На рисунке используются следующие сокращения: ОУ – объект управления; ПД – подсистема датчиков; ПРК – подсистема преобразования и кодирования; КПД – каналы передачи данных; ССК – средства сетевой коммутации; ПВВ – подсистема ввода-вывода; ПХО – подсистема хранения и обработки информации; ИУ – подсистема исполнительных устройств.

Рассмотренная структура ИС классифицируется как замкнутый контур управления с обратной связью. Канал прямой связи (ПХО → ПВВ → КПД → ССК → КПД → ПРК → ИУ → ОУ) позволяет передавать корректирующие управляющие воздействия от ПХО к ОУ. Канал обратной связи (ОУ → ПД → ПРК → КПД → ССК → КПД → ПВВ → ПХО) обеспечивает получение информации от ОУ и дальнейшую передачу информации по сети связи к ПХО, где осуществляются накопление и обработка полученной информации, принятие управленческих решений.

Корректирующие воздействия в зависимости от влияния на ОУ классифицируются как положительная или отрицательная обратная связь. Положительная обратная связь приводит к возрастанию тех или иных параметров ОУ (например, разгон двигателя, повышение температуры нагрева, повышение продуктивности в сельском хозяйстве и пр.). Отрицательная обратная связь вызы-

вает уменьшение каких-либо параметров ОУ (например, торможение двигателя, снижение температуры нагрева, уменьшение объёма выпускаемой продукции и пр.). Наличие обратной связи обеспечивает управление ОУ по адаптивным алгоритмам, учитывающим его текущее состояние.

В разомкнутых ИС отсутствует та или иная ветвь контура управления. Например, наличие лишь канала прямой связи позволяет осуществлять только получение информации в ПХО (прямой видеорепортаж со спортивных состязаний, получение изображения марсианской впадины и пр.). Таким образом, прямое управление ОУ в этом случае отсутствует.

Если в ИС имеется только канал прямой связи, то возможно управление ОУ по жёсткому неадаптивному алгоритму, не учитывающему текущее состояние ОУ (например, наполнение резервуара водой по времени при заданном расходе воды, управление школьными звонками по расписанию).

Типовые звенья информационной цепи. Объектом управления в зависимости от назначения ИС (технические ИС, экономические ИС, ИС в гуманитарных областях и т. д.) могут быть технические устройства, системы экономического назначения, социальные среды и пр.

Назначение ИС в существенной мере определяет и состав используемых для её построения средств. Так, подсистема датчиков осуществляет первичный съём информации с ОУ и преобразует её в определённые состояния вторичных физических носителей (в основном в параметры электрических сигналов, оптических носителей, воздушных и гидросред, радиосигналов и пр.). Многообразие первичных датчиков объясняется многообразием физических сред первичных источников: твердые, жидкие, газообразные, лучевые аудио- и видеосреды и пр., а также конкретными измеряемыми параметрами этих сред. В частности, в социальной среде датчиком может являться опрашиваемый житель, а первичным преобразователем – корреспондент. Таким образом, ПД реализует и первичное кодирование информации.

Подсистема преобразования и кодирования выполняет очередное кодирование, приводящее исходные коды ПД к стандартным кодам и стандартным физическим носителям, принятым в системах передачи данных. Подобная унификация позволяет резко сократить число типов средств передачи данных.

Непосредственно для обмена стандартной кодированной информацией используются КПД, включающие в себя средства управления передачей, линии связи и средства управления приёмом передаваемой информации. Назначение средств управления передачей – перекодировка передаваемой информации с учётом параметров физического носителя линии связи и отслеживания стандартного протокола обмена при передаче информации. Линии связи (проводные, оптические, радио и пр.) позволяют при заданном уровне искажений пере-

давать информацию на входы средств управления приёмом. Последние осуществляют восстановление стандартных параметров сигналов, отслеживают протоколы обмена и преобразуют принятую информацию к стандартам ССК.

ССК предназначены для приёма, контроля, буферизации, первичной обработки информации, коммутации КПД либо коммутации сообщений в КПД.

Второй уровень КПД предназначен для передачи информации к ПВВ центральной вычислительной системы. Выходные сигналы КПД этого уровня преобразуются к стандартам и протоколам интерфейсов каналов или адаптеров ввода-вывода вычислительной системы.

ПВВ передают информацию ядру вычислительной системы – ПХО, где осуществляются сохранение информации на внешних и внутренних носителях, обработка информации по заданным алгоритмам и, в зависимости от назначения ИС, формирование корректирующих воздействий для ОУ или оповестительных сообщений для разомкнутых ИС.

При передаче по цепям обратной связи от ПХО к ОУ производятся обратные преобразования и пересылки относительно рассмотренных выше в цепи прямой связи.

Задачи информатики

Исходя из вышеизложенного, основной задачей информатики является создание методологии сбора требуемой информации, её преобразования и передачи в ядро ИС, обработки и накопления информации, формирования корректирующих воздействий на ОУ или оповестительных сведений. В прикладном плане – создание аппаратных и программных средств ИС.

Информатика в технических системах. Рассмотренная ранее циркуляция информации в контуре управления называется информационным процессом. В технических системах информационный процесс ориентирован на обеспечение оптимального в каком-либо смысле процесса управления техническими объектами.

Оптимальность обеспечивается за счёт минимизации или максимизации определённой целевой функции, что достигается совершенствованием как аппаратной среды ИС (повышается быстродействие интегральной среды, используется аппаратная реализация алгоритмов, улучшается точность преобразований за счёт увеличения разрядной сетки вычислительных средств, применения новых принципов съёма информации и пр.), так и программного обеспечения (разработка эффективных алгоритмов решения задач управления).

Техническая ИС может быть автоматической – без участия человека в контуре управления (АСУТП) и автоматизированной – с включением человека в контур управления в качестве оператора или менеджера.

Информатика в социальных средах. С точки зрения кибернетики, социальная среда также является ИС и представляет собой иерархическую систему контуров управления. Нижний уровень – человек с его индивидуальным циклом функционирования. Следующими уровнями являются уровни социальных групп различного ранга (семья, семейный клан, род и пр.). Высшими социальными уровнями в настоящее время являются государство и союзы государств.

Несомненно, что управление в социальных ИС не поддаётся детерминированному управлению из-за сложности формального описания соответствующих циклов управления. Частично детерминированное управление вводится путём издания свода законов. Но в целом задача формализации сложной системы изнутри самой системой не решаема.

Однако автоматизировать подобное управление на различных внутренних уровнях представляется возможным с использованием статистических и вероятностных алгоритмов управления. В простейших случаях возможно создание полностью автоматических социальных ИС. Например, ИС управления социальным циклом школьника на базе планшетного ПК с расписанием его дневного цикла, звуковым и видеооповещением школьника об очередном этапе его функционирования и получением ответных видео, звуковых, текстовых, функциональных реакций с его стороны.

Иные примеры – дистанционная диагностика и лечение больного, манипуляция общественным мнением через средства массовой коммуникации с целью обеспечения того или иного общественного поведения.

Информационное общество. В наше время все социальные группы населения развитых стран в той или иной степени задействованы в социальных контурах управления через средства массовой коммуникации. В свою очередь, эти средства связаны локальными и глобальными ИС. Это позволяет классифицировать соответствующие общества как информационные.

Информационное общество – общество, в котором большинство работающих занято производством, хранением, переработкой и реализацией информации, особенно высшей ее формы – знаний. Для этой стадии развития общества и экономики характерно:

– увеличение роли информации, знаний и информационных технологий в жизни общества;

– возрастание числа людей, занятых информационными технологиями, коммуникациями и производством информационных продуктов и услуг, рост их доли в валовом внутреннем продукте;

– нарастающая информатизация общества с использованием телефонии, радио, телевидения, сети Интернет, а также традиционных и электронных СМИ;

– развитие электронной демократии, информационной экономики, электронного государства, электронного правительства, цифровых рынков, электронных социальных и хозяйствующих сетей;

– создание глобального информационного пространства, обеспечивающего:

а) эффективное информационное взаимодействие людей;

б) их доступ к мировым информационным ресурсам;

в) удовлетворение их потребностей в информационных продуктах и услугах.

Определение и свойства алгоритма

Под алгоритмом понимается связанная последовательность правил решения задачи или множества одного класса задач.

Алгоритм должен обладать следующими свойствами.

Дискретность – алгоритм описывает процесс решения задачи как последовательное выполнение некоторых простых шагов. При этом для выполнения каждого шага алгоритма требуется конечный отрезок времени, то есть преобразование исходных данных в результат осуществляется во времени дискретно.

Детерминированность (определённость) – в каждый момент времени следующий шаг работы однозначно определяется состоянием системы. Таким образом, алгоритм выдаёт один и тот же результат (ответ) для одних и тех же исходных данных. В современной трактовке у разных реализаций одного и того же алгоритма должен быть изоморфный граф. С другой стороны, существуют вероятностные алгоритмы, в которых следующий шаг работы зависит от текущего состояния системы и генерируемого случайного числа. Однако при включении метода генерации случайных чисел в список «исходных данных» вероятностный алгоритм становится подвидом обычного.

Два графа G1 и G2 называются изоморфными, если существует взаимно однозначное соответствие между множествами их вершин, обладающее тем свойством, что число ребер, соединяющих любые две вершины в G1, равно числу ребер, соединяющих соответствующие вершины в G2.

Из определения следует, что изоморфные графы можно одинаково изображать графически и отличаться они будут только метками вершин.

Конечность – при корректно заданных исходных данных алгоритм должен завершать работу и выдавать результат за конечное число шагов.

Массовость (универсальность). Алгоритм должен быть применим к разным наборам исходных данных для определённого класса задач.

Результативность – по завершении работы алгоритма должны быть получены определённые результаты.

Выше рассмотрены пять основных свойств алгоритма. При практической реализации к алгоритму может предъявляться и следующее требование.

Понятность – алгоритм должен включать только те операции, которые входят в доступное исполнителю множество операций.

Представление алгоритма

Алгоритм может быть представлен в виде формального, содержательного и графического (структурного) описания.

Формальное описание в виде математического выражения, если оно возможно, является наиболее общим и может иметь различные прикладные отображения на том или ином алгоритмически полном (алгоритмически универсальном) множестве операций. Например, формула вычисления корней квадратного уравнения. Под алгоритмически полным множеством операций понимается такое, на котором можно отобразить любой вычислительный алгоритм.

Содержательное описание предусматривает отображение алгоритма в том или ином тезаурусе языка человеческого общения. Промежуточным формально-содержательным описанием может быть представление на алгоритмических языках.

Графическое (структурное) описание представляется в форме связного конечного графа, у которого вершина «Начало» имеет только исходящую ветвь, вершина «Конец» может иметь только входящие ветви, а тело графа состоит из операторных и условных вершин, каждая из которых может иметь несколько входящих ветвей. Операторная вершина может иметь только одну исходящую ветвь, а условная вершина – две (в случае логического условия) или более (при многопараметрическом условии) исходящих ветвей.

Алгоритм Евклида

Исторически наиболее давним известен алгоритм Евклида, который посвящён нахождению наибольшего общего делителя двух целых положительных чисел, при условии, что он имеется.

В самом простом случае алгоритм Евклида применяется к паре положительных целых чисел и формирует новую пару, которая состоит из меньшего числа и разницы между большим и меньшим числом. Процесс повторяется, пока числа не станут равными. Найденное число и есть наибольший общий делитель исходной пары.

Первое описание алгоритма находится в «Началах» Евклида (около 300 лет до н. э.), что делает его одним из старейших численных алгоритмов, используемых в наше время. Оригинальный алгоритм был предложен только для натуральных чисел и геометрических длин (вещественных чисел).

Алгоритм для поиска наибольшего общего делителя двух натуральных чисел описан также в первой книге древнекитайского трактата «Математика в девяти книгах».

Содержательное описание алгоритма Евклида

Начало.

1. Берём два числа.

2. Если первое число равно второму числу, идти к п. 6.

3. Если первое число больше второго числа, идти к п. 5.

4. Вычитаем из второго числа первое число, остаток присваиваем второму числу. Идти к п. 2.

5. Вычитаем из первого числа второе число, остаток присваиваем первому числу. Идти к п. 2.

6. Наибольший общий делитель равен первому числу. Конец.

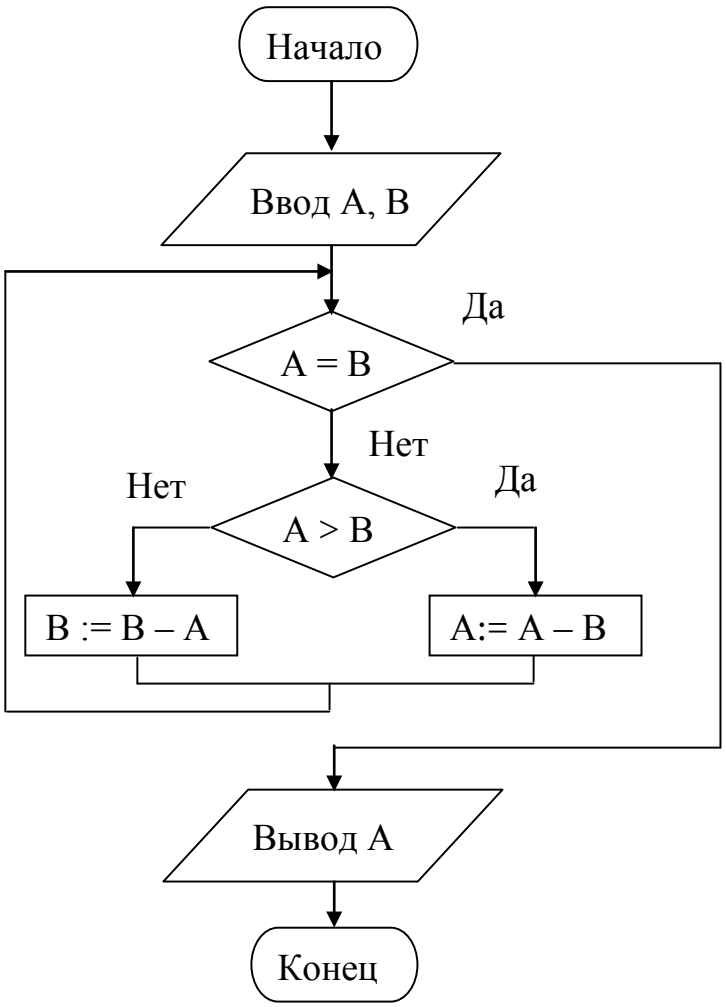

Рис. 2

Графическое (структурное) представление алгоритма Евклида

Структурная схема алгоритма приведена на рис. 2. Тело графа содержит четыре операторных и две условных вершины. Операторные вершины содержат описание ввода чисел, вывода результата и выполняемых арифметических операций. Условные вершины являются логическими, имеющими каждая по

исходам «Да» и «Нет». Дуги графа обеспечивают связность вершин и порядок их исполнения.

В общем случае в вершинах графа размещаются операции из доступного пользователю множества (логические, арифметические, трансцендентные и пр.), но в любом случае это должны быть алгоритмически полные множества операций.

Любое алгоритмически полное множество операций в современных вычислительных средствах обязательно включает логические и арифметические операции, через которые можно реализовать любые функционально сложные вычисления. В свою очередь, функциональные узлы цифровых вычислительных машин состоят из простейших логических элементов, что и определяет алгебру логики в качестве формальной базы вычислительных систем.

Терминология алгебры логики

Алгебра логики – это алгебра высказываний типа «истина» – «ложь». Допускается обозначать их символами 1 – 0, которые не рассматриваются как числа.

Логическая (булева) переменная – это простое высказывание, которое может принимать значения только из множества {0,1}.

Логические связки – объединяют простые высказывания в сложные высказывания – в логические функции. Например, знаками логических связок являются следующие символы:

\/ (допускаются «+», «or», «или»);

/\ (допускаются «*», «and», &, «и»).

Логическая функция (ЛФ) – функция f(x1, x2, ..., xn) называется логической (переключательной или булевой) функцией n аргументов xi , если она и ее аргументы могут принимать значения только из множества {0,1}.

Набор – совокупность значений аргументов логической функции. Любая логическая функция n аргументов может иметь m = 2^n наборов. Каждому набору значений аргументов приписывается номер (0, 1,…, m–1), полученный как вес двоичного позиционного числа, совпадающего по кодировке с набором.

Количество различных ЛФ n аргументов конечно и равно k = 2^m. Каждой логической функции данного набора аргументов принято приписывать номер (0, 1, …, k–1), полученный как вес двоичного позиционного числа, совпадающего по кодировке со значениями ЛФ на наборах от 0 до m–1. Так, например, множество бинарных ЛФ f(x1, x2) включает в себя 16 функций.

x2	x1	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

За каждой бинарной ЛФ закреплено определённое название, например:

f14 – дизъюнкция (логическое «или», логическое сложение, «OR»);

f8 – конъюнкция (логическое «и», логическое умножение, «AND»);

f7 – штрих Шеффера («и-не»);

f1 – стрелка Пирса («или-не»);

f6 – исключающее ИЛИ («сумма по mod2», «XOR»);

f5 – инверсия x1 («не x1», отрицание x1);

f3 – инверсия x2 («не x2», отрицание x2).

Литерал – логическая переменная или ее инверсия (отрицание). Например, переменная A и ее инверсия !A – это одна переменная с разными значениями, но два литерала.

x2	x1	f6
0	0	0
0	1	1
1	0	1
1	1	0

Таблица истинности – таблица, в которой всем возможным наборам аргументов соответствуют определённые значения ЛФ. Так, функция XOR имеет следующую таблицу истинности.

Неполностью определенная ЛФ **n** переменных, это функция, определённая на числе наборов, меньшем **m**.

Суперпозиция – подстановка в логическую функцию вместо ее аргументов других логических функций.

Функционально полная система логических функций – система, с помощью логических функций которой, применяя операции суперпозиции и подстановки, можно получить любую сколь угодно сложную логическую функцию.

Терм – группа логических переменных в прямой или инверсной форме (группа литералов), объединенных одним и тем же знаком логической связки: логического сложения \/ (дизъюнкции) или же логического умножения /\ (конъюнкции). В терме каждая переменная или ее отрицание встречается только один раз.

Ранг терма – количество переменных и их инверсий (количество литералов), входящих в данный терм. Терм, в который входят все переменные или их отрицания, имеет максимальный ранг.

Макстерм (дизъюнктивный терм) – терм, в котором литералы связаны знаком дизъюнкции.

Минтерм (конъюнктивный терм) – терм, в котором литералы связаны знаком конъюнкции.

Конституента единицы (К1) тождественна минтерму.

Конституента нуля (К0) тождественна макстерму.

Базис – функционально полная система элементарных функций. Пример классического базиса: {«И», «ИЛИ», «НЕ»}. Пример минимального базиса: {«штрих Шеффера»}.

Способы представления логических функций

Табличный, аналитический, числовой, геометрический (графический).

Нормальные формы аналитического представления ЛФ:

ДНФ (дизъюнктивная нормальная форма) – ЛФ = \/ Fi, i <= m, где Fi – минтермы любого ранга.

КНФ (конъюнктивная нормальная форма) – ЛФ = /\ Hi, i <= m, где Hi – макстермы любого ранга.

Совершенная (каноническая) форма аналитического представления ЛФ:

СДНФ – содержит минтермы только максимального ранга.

СКНФ – содержит макстермы только максимального ранга.

Минимизация логической функции – получение из исходного аналитического представления ЛФ, используя операции алгебры логики, аналитическое представление этой ЛФ с меньшим числом литералов.

МДНФ – минимальная ДНФ, имеющая по сравнению с исходной ДНФ возможно меньшее количество минтермов с возможно меньшими рангами, ни один из которых исключить нельзя, или ДНФ, содержащая наименьшее количество литералов.

МКНФ – минимальная КНФ, имеющая по сравнению с исходной КНФ возможно меньшее количество макстермов с возможно меньшими рангами, ни один из которых исключить нельзя, или КНФ, содержащая наименьшее количество литералов.

Соседние термы – термы в ДНФ или КНФ, которые отличаются только одной переменной (в одном терме переменная без отрицания, а в другом – с отрицанием).

Импликанта – некоторая логическая функция, входящая в данную ЛФ и обращаемая в ноль при наборе переменных, на котором сама ЛФ также равна нулю.

Импликанта дизъюнктивная – любой минтерм или группа минтермов исходной ДНФ.

Импликанта конъюнктивная – любой макстерм или группа макстермов исходной КНФ.

Простые (элементарные) импликанты – импликанта минтерма или мак-стерма, никакая собственная часть которого уже не является импликантой данной ЛФ.

Сокращенная форма аналитического представления ЛФ – дизъюнкция всех ее простых импликант.

Тупиковая форма аналитического представления ЛФ – дизъюнкция простых импликант, ни одну из которых исключить нельзя. Минимальные формы ЛФ выбирают из тупиковых форм с минимальным числом литералов.

Поглощение – A(A + B) = A; A + AB = A.

Склеивание – AB +AB = B (A +A) = B; (A + B)(A +B) = A.

Неполное склеивание – xy + xy = x + xy + xy.

Формула развертывания: x = (x +y)(x +y); (x + y) = (x + y + z)(x + y +z).

Теорема (законы) де Моргана – !(A + B) = !A*!B; !(A*B) = !A +!B.

Числовой способ представления ЛФ. В случае СДНФ под знаком \/ перечисляются заключенные в скобки номера наборов, на которых функция равна единице. В случае СКНФ под знаком /\ перечисляются заключенные в скобки номера наборов, на которых функция равна нулю.

Например: f1 = \/ (1, 3, 5, 6), f2 = /\(0, 4, 7).

Геометрический (графический) способ представления ЛФ. Термы функции n переменных размещаются по вершинам n-мерного куба, дискретом каждой координаты которого является соответствующая логическая переменная.

Минимизация СДНФ ЛФ с помощью карт Карно

Карты Карно (модифицированные карты Вейча) представляют собой матрицу с количеством ячеек по числу возможных наборов переменных ЛФ, каждая ячейка которой соответствует определённому минтерму и помечается символом «истина» (например, 1 или любой символ) или «ложь» (например, 0 или любой символ, отличный от «истины») в соответствии со значениями данной ЛФ, причём минтермы в соседних ячейках являются соседними.

Пример разметки карты Карно для ЛФ от четырех переменных (n = 4) представлен на рис. 3. Ячейки карты помечены соответствующими символическими записями минтермов для данной разметки.

На рис. 4 в карте Карно размещена СДНФ ЛФ вида

f = !x4*!x3*!x2*!x1 + x4*!x3*x2*!x1 + x4*x3*x2*!x1 + x4*!x3*x2*x1+

+ x4*x3*x2*x1 + !x4*x3*!x2*!x1 + x4*!x3*!x2*x1 + !x4*x3*!x2*x1,

где символам !, *, + соответствуют логические операции инверсии, конъюнкции и дизъюнкции.

Метод Карно предусматривает выполнение операций склеивания и покрытия. Склеиванию подлежат пары, четвёрки, восьмёрки и т. д. 2^i минтермов

(1, 2,…, n) в соседних недиагональных ячейках. Матрица карты свёрнута в цилиндры по левой-правой и нижней-верхней границам. Например, крайние правые ячейки всех строк являются соседними для крайних левых ячеек этих строк, а нижние ячейки столбцов – соседними для верхних ячеек этих столбцов.

<table>
<tr><td></td><td colspan="2">!x4</td><td colspan="2">x4</td><td colspan="2">!x4</td><td></td></tr>
<tr><td rowspan="2">!x1</td><td>!x4*!x3*!x2*!x1</td><td>x4*!x3*!x2*!x1</td><td>x4*x3*!x2*!x1</td><td>!x4*x3*!x2*!x1</td><td>!x2</td></tr>
<tr><td>!x4*!x3*x2*!x1</td><td>x4*!x3*x2*!x1</td><td>x4*x3*x2*!x1</td><td>!x4*x3*x2*!x1</td><td rowspan="2">x2</td></tr>
<tr><td rowspan="2">x1</td><td>!x4*!x3*x2*x1</td><td>x4*!x3*x2*x1</td><td>x4*x3*x2*x1</td><td>!x4*x3*x2*x1</td></tr>
<tr><td>!x4*!x3*!x2*x1</td><td>x4*!x3*!x2*x1</td><td>x4*x3*!x2*x1</td><td>!x4*x3*!x2*x1</td><td>!x2</td></tr>
<tr><td></td><td colspan="2">!x3</td><td colspan="2">x3</td><td></td></tr>
</table>

Рис. 3

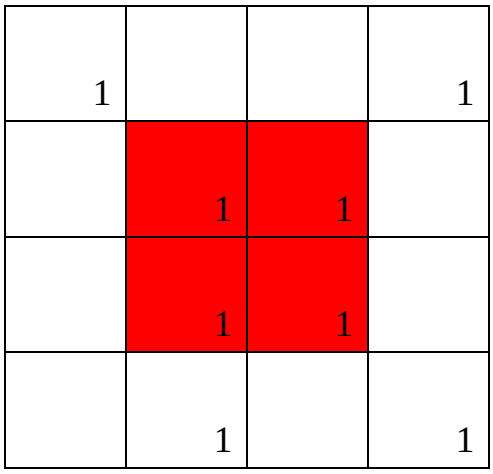

Рис. 4

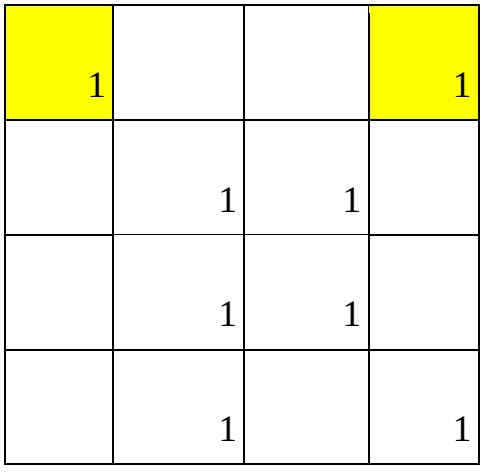

Рис. 5

Допускаются частичные покрытия одних склеек другими с целью охвата всех минтермов данной ДНФ, а также наличие минтермов, не вошедших ни в одну склейку. В первую очередь выполняются склейки максимально возможного размера. На рис. 4 – это выделенная четвёрка минтермов в центре матрицы.

Далее выполняются склейки меньшего размера. На рис. 5–7 – это выделенные пары минтермов.

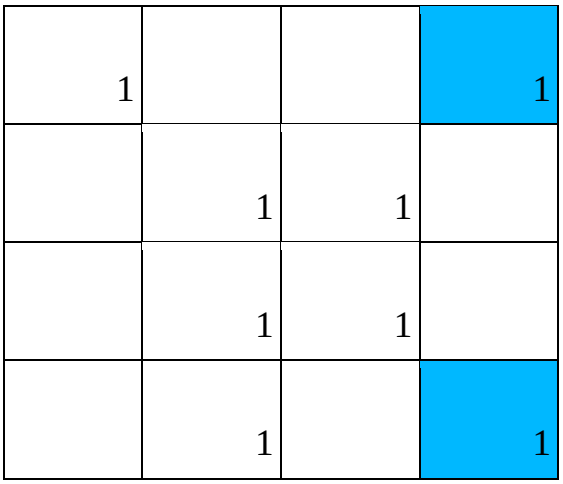

Рис. 6

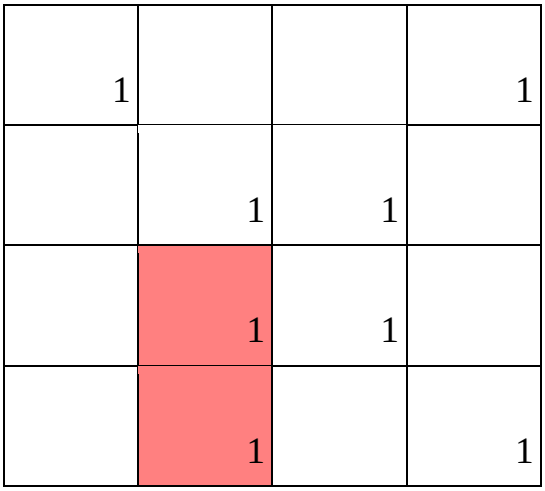

Рис. 7

Склейка на рис. 7 частично покрывает на склейку рис. 4. В результате выполненных склеек были охвачены все минтермы ЛФ. Каждая из склеек позволяет исключить из соответствующей группы минтермов те переменные, от которых эта склейка не зависит. Например, склейка на рис. 4 не зависит от переменных x1 и x3, что позволяет получить минимальную форму для данной склейки вида x4*x2, склейка на рис. 5 не зависит от x3, склейка на рис. 6 не зависит от x1, а склейка на рис. 7 не зависит от x2. Получаем соответствующие минимальные формы !x4*!x2*!x1, !x4*x3*!x2 и x4*!x3*x1.

Результирующая МДНФ ЛФ будет иметь вид

f = x4*x2 + !x4*!x2*!x1 + !x4*x3*!x2 + x4*!x3*x1.

В общем случае может быть несколько минимальных форм ЛФ, из которых выбирается та, которая в большей степени по тем либо иным критериям устраивает разработчика в плане дальнейшей аппаратной или программной реализации.

Основы машинной арифметики

Арифметические операции в ЦВМ выполняются преимущественно над позиционными двоичными кодами чисел. Любое целое число $A < 2^n$ может быть представлено в виде ряда

$$A = \sum_{i=0}^{i=n-1} a_i * p^i, \text{ где } a_i \in \{0,1\},\ p \text{ – основание системы счисления.}$$

Из каждого коэффициента a_i образуется цифра i-й позиции n-разрядного двоичного кода числа A. Разряд младшей позиции кода размещается справа (i = 0), разряд старшей позиции – слева (i = n – 1). Например, число четырнадцать разлагается по степеням числа 2 в следующий ряд:

A = 1*8 + 1*4 + 1*2 + 0*1.

Соответствующий позиционный двоичный код числа А имеет вид 1110. Допускается расширение кода нулями слева для получения требуемой разрядности. Например, этот же код в восьмиразрядной сетке имеет вид 00001110.

Полученный код является беззнаковым. Для кодирования чисел со знаками перед левой границей кода вводятся один (простой код) или два (модифицированный код) знаковых разряда. Знаки положительных чисел кодируются соответственно 0 или 00, а знаки отрицательных чисел кодируются 1 или 11.

Различают прямые, обратные и дополнительные коды. В прямых кодах арифметические операции выполняются отдельно над модулями чисел и отдельно над знаковыми битами. В обратных и дополнительных кодах арифметические операции выполняются над полноразрядными кодами, включая знаковые биты.

Прямой код числа получается из беззнакового путём добавления знакового бита (битов) слева от старшего бита. Так, для беззнакового кода числа 1110 прямой код положительного числа имеет вид 0.1110 (простой код) или 00.1110 (модифицированный код), а прямой код отрицательного числа – соответственно 1.1110 или 11.1110. Точка (или запятая) отделяет знак от модуля числа как для чисел, меньших единицы, так и для чисел, больших единицы. В первом случае для простых кодов позиции битов отсчитываются справа от точки, начиная с –1 в сторону убывания: –1, –2, …, – (n–1), а во втором случае – с (n –2), …, 1, 0 для n-разрядного кода (включая знаковый бит).

Для n-разрядных модифицированных кодов позиции битов будут соответственно в диапазонах –1, –2, … , –(n – 2) и (n –3), …, 1, 0.

Обратные коды чисел получают из прямых кодов путём инверсии битов модуля отрицательных чисел. Для рассмотренных выше примеров коды положительных чисел остаются без изменения, а коды отрицательных чисел будут иметь вид 1.0001 или 11.0001.

Дополнительные коды чисел получают из обратных кодов отрицательных чисел путём прибавления единицы младшего разряда к младшему разряду кода. Коды положительных чисел остаются без изменения. Для ранее рассмотренных примеров дополнительные коды будут иметь вид 1.1111 и 11.1111.

Системы представления двоичных чисел

Основными системами представления являются: система с фиксированной точкой (или запятой) и система с плавающей точкой (или запятой) – полулогарифмическая система.

Система с фиксированной точкой была рассмотрена в примерах, иллюстрирующих коды двоичных чисел. Ноль в любом коде должен быть представлен в виде положительного нуля.

Система с плавающей точкой предполагает определять вес числа по формуле

$$A = Am^* 10^p,$$

где Am – мантисса числа A;

10 – основание системы счисления (в данном случае двойка);

p – порядок числа A.

Мантисса всегда является числом, меньшим единицы по модулю, а порядок – целым числом.

Так, для ранее рассмотренных примеров беззнакового целого числа 1110 возможны следующие варианты представления в прямых кодах:

– положительные числа Am = 0.1110 (или 00.1110), p = 0.0100 (или 00.0100), т. е. вес порядка 4;

Am = 0.01110, p = 0.0101, т. е. вес порядка 5.

В первом случае для прямых кодов число считается нормализованным (в старшем разряде модуля мантиссы находится 1), во втором случае число денормализовано (в старшем разряде модуля мантиссы находится 0);

– отрицательные числа Am = 1.1110 (или 11.1110), p = 0.0100 (или 00.0100), т. е. вес порядка 4;

Am = 1.01110, p = 0.0101, т. е. вес порядка 5.

Если беззнаковое число 1110 рассматривается как меньшее единицы по модулю, соответствующие примеры примут вид:

– положительные числа Am = 0.1110 (или 00.1110), p = 0.0000 (или 00.0000), т. е. вес порядка 0;

Am = 0.01110, p = 0.0001, т. е. вес порядка 1;

– отрицательные числа Am = 1.1110 (или 11.1110), p = 0.0000 (или 00.0000), т. е. вес порядка 0;

Am = 1.01110, p = 0.0001, т. е. вес порядка 1.

Порядок числа может быть и отрицательным, например, когда в результате выполнения арифметической операции была получена денормализованная мантисса и при её нормализации порядок перешёл в отрицательную область. Пусть результат операции содержит Am = 0.000101101 и порядок p = 0.0001. После нормализации мантисса примет вид Am = 0.101101000, а порядок станет равным p = 1.0010, т. е. минус 2.

Отличие в представлении обратных и дополнительных кодов по сравнению с прямым кодом состоит в том, что для отрицательных мантисс признаком нормализации будет несовпадение знаковых битов и бита справа от точки (т. е. старшего значащего бита). Нулевая мантисса в любом коде должна быть представлена в виде положительного нуля.

Примеры выполнения машинных операций

Элементарные арифметические и логические преобразования в ЦВМ в основном выполняются в комбинационном (ненакапливающем) арифметико-логическом устройстве (АЛУ). Для хранения кодов двоичных чисел используются регистры (Рг) или ячейки памяти (ЯП) запоминающих устройств (ЗУ). В каждом регистре и в каждой ячейке памяти возможно хранение одного числа. В Рг и ЯП возможны операции записи (Зп), чтения (Чт) и хранения. Перед выполнением машинной операции требуется закрепить регистры АЛУ и ячейки ЗУ за операндами, промежуточными и конечными результатами. В современных процессорах число регистров колеблется от 8–10 до нескольких сотен. Поэтому в последующих примерах ЯП будут использоваться лишь для хранения больших массивов данных (например, при реализации табличных преобразований).

Основными арифметическими машинными операциями являются алгебраические суммирование, вычитание (относятся к коротким операциям), умножение и деление (относятся к длинным операциям).

Архитектура гипотетического операционного устройства

Общепринятая модель ОУ приведена на рис. 8. ОУ состоит из двух частей: операционной части (ОЧ) и управляющей части (УЧ). ОЧ предназначена для выполнения элементарных операций над хранимыми операндами из внутреннего множества {B}, внешними операндами из множества {A}, формирования результатов и сохранения их как элементов множества {B} или элементов внешнего множества {Z}.

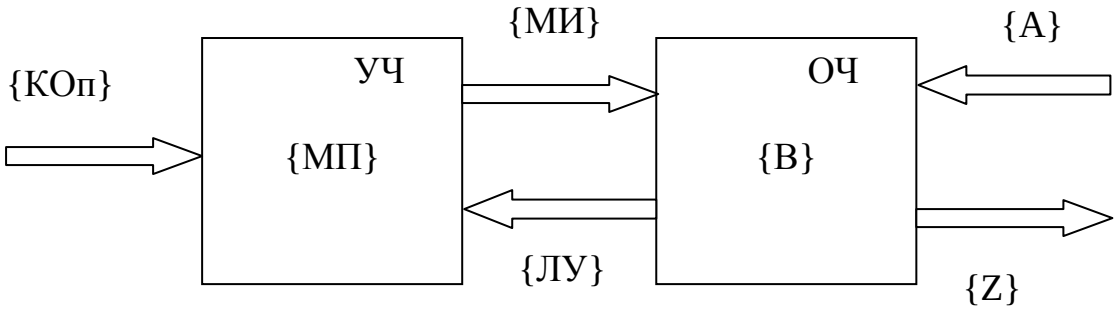

Рис. 8

УЧ хранит множество микропрограмм {МП}, соответствующих поступающим кодам машинных операций из множества {КОп}, осуществляет выборку определённой микропрограммы по конкретному коду операции, разворачивание во времени процесса выполнения микропрограммы. Каждая микропрограмма состоит из функционально связанных микрокоманд. Каждая микроко-

манда содержит микроинструкцию для ОЧ из допустимого множества микроинструкций {МИ} и информацию для выборки очередной микрокоманды выполняемой микропрограммы.

По коду микроинструкции в ОЧ выполняются соответствующие элементарные операции над операндами множеств {A} и {B} с последующим формированием элементов множеств {B}, {Z} и логических условий {ЛУ}. Логические условия являются оповестительными сигналами для УЧ в точках ветвления микропрограмм и позволяют выполнять переход по соответствующей ветви исполняемой микропрограммы.

Для машинной реализации рассматриваемых нами алгоритмов будем использовать гипотетическую ОЧ (рис. 9), содержащую комбинационное АЛУ, набор регистров – регистровый файл (РФ), шины ввода (ШВВ) и вывода (ШВ) данных, шины управления (ШУ) и выходные шины логических условий (ШЛУ).

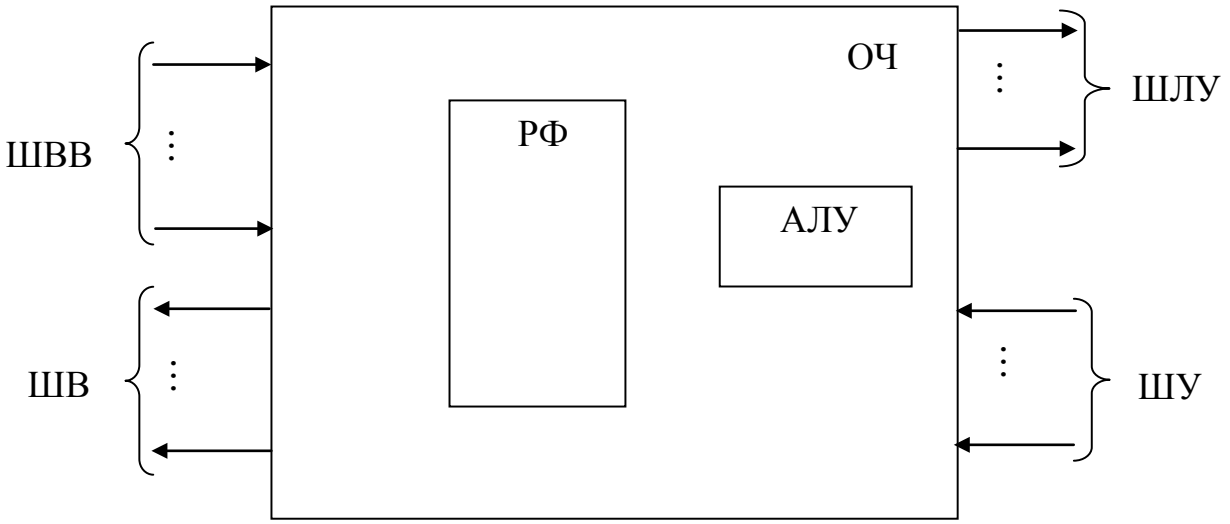

Рис. 9

В систему инструкций, выполняемых ОЧ, будем включать по мере необходимости элементарные операции, позволяющие выполнить действия над двоичными кодами, предписанные в операторных вершинах алгоритма.

Коды операндов и рабочих констант будем загружать с ШВВ, промежуточные и конечные результаты операции будем размещать в регистровом файле и выводить по завершении выполнения алгоритма на ШВ.

Дополнительно к инструкциям преобразования двоичных кодов будем использовать инструкции управления внутренними пересылками в ОУ и внешними операциями ввода и вывода данных. Будем считать, что последовательность инструкций, определяющая реализацию алгоритма, размещена во внешнем блоке управляющей памяти с линейной структурой (адреса соседних ячеек отличаются на единицу) в соответствии с принципами естественной адресации.

Суть естественной адресации заключается в размещении последовательности инструкций, соответствующей линейной ветви алгоритма, в естественном порядке, когда адрес следующей инструкции увеличивается на +1 относительно адреса предыдущей инструкции. В точках условного ветвления алгоритма один из исходов считается продолжением линейной ветви, а альтернативный исход принудительно отсылает к адресу соответствующей инструкции. Также возможны принудительные безусловные переходы на требуемый адрес.

Признаки ветвлений, соответствующие условным вершинам алгоритма, будем выводить из ОЧ на ШЛУ. Обычно логические условия снимаются с выхода переноса АЛУ и выходов правого и левого сдвигов, выполняемых в сдвигающих регистрах.

Выполнение коротких машинных операций в системе с фиксированной точкой

На рис. 10 представлен алгоритм суммирования чисел в прямых кодах. Рассмотрим реализацию данного алгоритма в гипотетическом операционном устройстве (ОУ).

Для реализации алгоритма в ЦВМ потребуются два регистра РгА и РгВ для хранения операндов А и В, регистр РгК константы 0111…1 для выделения модуля, регистры модулей операндов РгМА и РгМВ, регистр результата РгС, регистр РгН (накапливающий регистр АЛУ), регистр РгО ошибки, в котором формируется сообщение о переполнении (отсутствие ошибки кодируем нулём).

Считаем, что операнды и константа загружаются в соответствующие регистры с ШВВ, АЛУ может выполнять над двоичными кодами элементарные операции вида: сложение, левый логический сдвиг, обнуление, инверсия, конъюнкция, дизъюнкция. По результату операций АЛУ могут формироваться логические условия: выход левого сдвига ВСЛ, нулевой результат Z (Z = 1, если результат нулевой). На входы АЛУ может подключаться содержимое любого регистра или пары регистров. Результат любой элементарной операции загружается в РгН с выхода АЛУ. Возможны пересылки между любой парой регистров, которые осуществляются через РгН АЛУ. Содержимое любого регистра может быть выведено на ШВ.

Отображение алгоритма на множестве машинных инструкций уровня регистровых передач

Процесс суммирования будет состоять из следующих тактов.

1. РгА := (ШВВ); ввод операнда А
2. РгВ := (ШВВ); ввод операнда В
3. РгК := (ШВВ); ввод константы 01...1 для выделения модуля числа

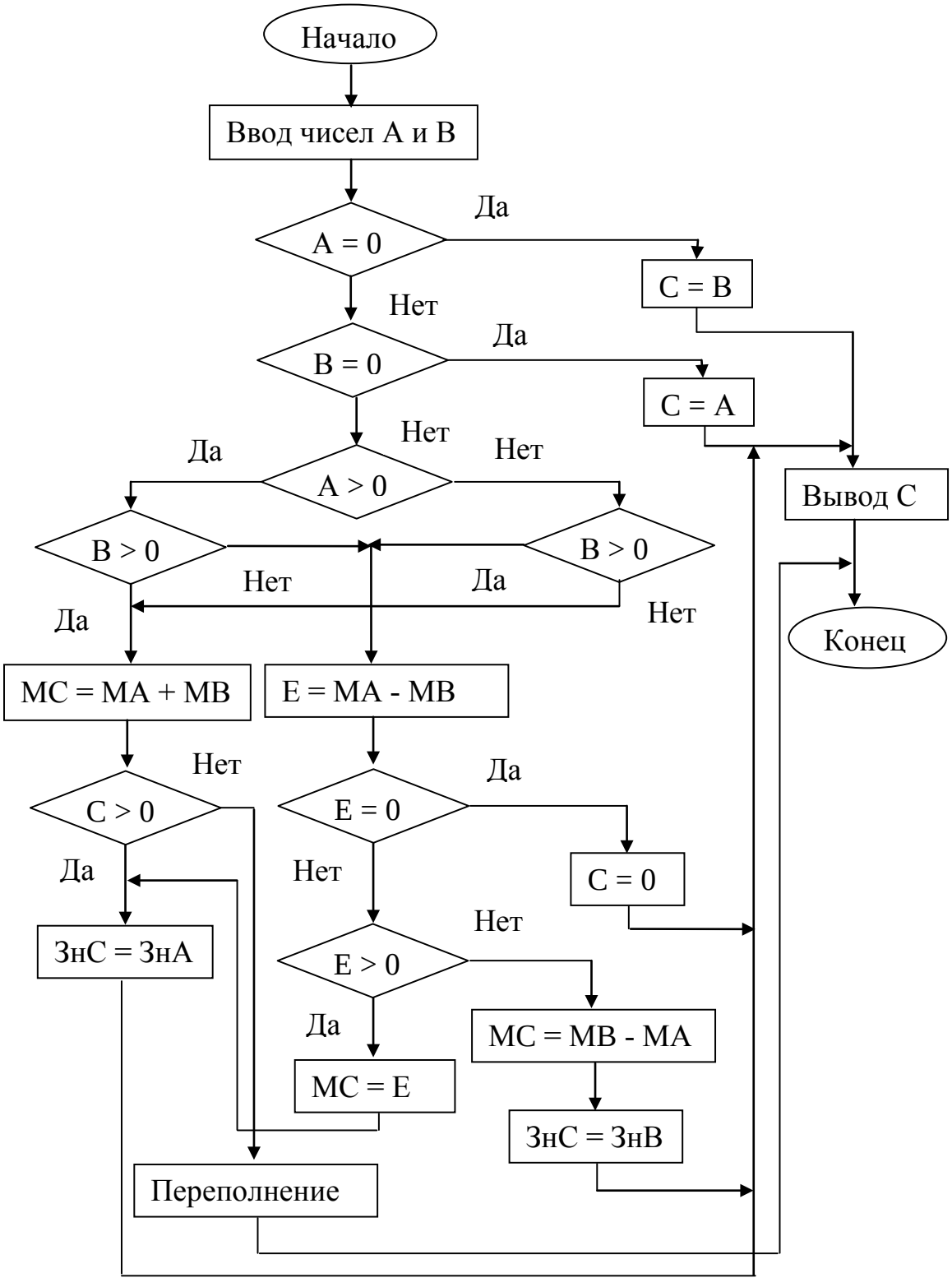

Рис. 10

4. РгН := (РгА), если Z = 1, идти к 28, иначе к 5; проверка А на 0
5. РгН := (РгВ), если Z = 1, идти к 29, иначе к 6; проверка В на 0
6. РгН := (РгА) & (РгК); выделение модуля А
7. РгМА := (РгН); сохранение модуля А

8. РгН := (РгВ) & (РгК); выделение модуля В

9. РгМВ := (РгН); сохранение модуля В

10. РгН := !(РгК); формирование инверсии константы

11. РгК := (РгН); сохранение инверсии константы (для выделения знака)

12. РгН := (РгА) & (РгК), если Z = 1, идти к 30, иначе к 13; проверка знака А

13. РгН := (РгВ) & (РгК), если Z = 1, идти к 18, иначе к 14; проверка знака В

14. РгН := (РгМА) + (РгМВ); суммирование модулей

15. РгС := (РгН); сохранение модуля суммы

16. РгН := <- (РгН), если ВСЛ = 1, идти к 31, иначе к 17; проверка на переполнение

17. РгН := (РгА) & (РгК), идти к 24: выделение знака А

18. РгН := (РгМА) – (РгМВ), если Z = 1, идти к 32, иначе к 19; прямое вычитание модулей

19. РгС := (РгН); сохранение модуля результата

20. РгН := <- (РгН), если ВСЛ = 1, идти к 21, иначе к 17; проверка знака разности

21. РгН := (РгМВ) – (РгМА); обратное вычитание модулей

22. РгС := (РгН); сохранение модуля результата

23. РгН := (РгВ) & (РгК); выделение знака В

24. РгН := (РгС) \/ (РгН); объединение модуля результата со знаком

25. РгС := (РгН); сохранение результата

26. РгН := 0; формирование нулевого сообщения (пп. 26, 27)

27. РгО := (РгН), идти к 33; сохранение сообщения

28. РгН := (РгВ), идти к 25; формирование результата С = В

29. РгН := (РгА), идти к 25; формирование результата С = А

30. РгН := (РгВ) & (РгК), если Z = 1, идти к 14, иначе к 18; проверка знака В

31. РгН := (РгК), идти к 35; формирование ненулевого сообщения об ошибке

32. РгН := 0, идти к 25; формирование нулевого результата

33. ШВ := (РгО); вывод сообщения о нормальном результате

34. ШВ := (РгС); вывод результата. Конец.

35. РгО := (РгН); сохранение сообщения

36. ШВ := (РгО); вывод сообщения об ошибке. Конец.

В прил. 1 рассматриваются числовые примеры, иллюстрирующие работу приведённой выше процедуры (нумерация тактов сохраняется). В первом примере суммировались разнознаковые числа, переполнение отсутствует, сформировано сообщение об отсутствии ошибки. Во втором примере суммировались равнознаковые числа, произошло переполнение, сформировано сообщение об ошибке.

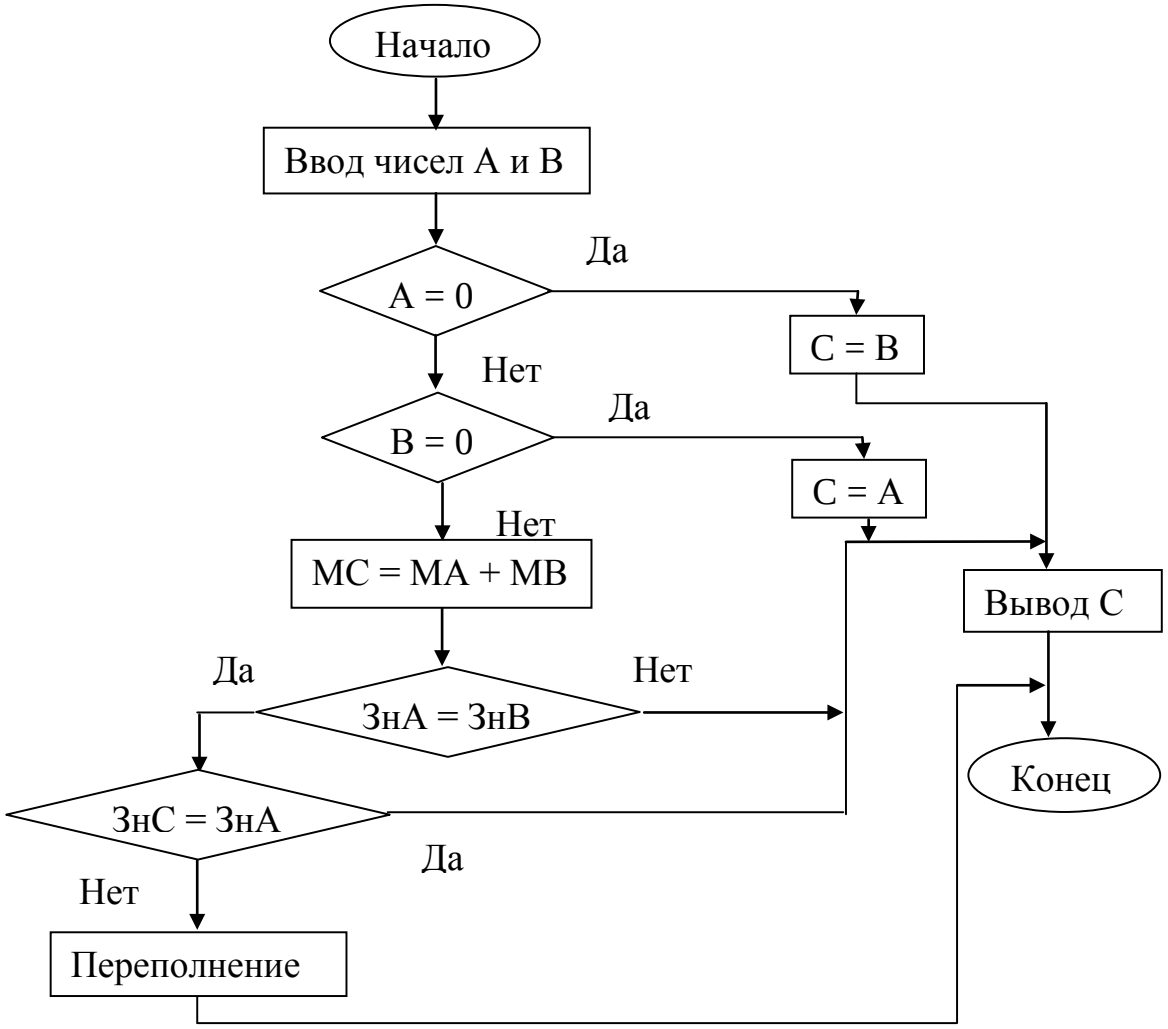

Рис. 11

На рис. 11 представлен алгоритм суммирования чисел в дополнительных кодах. Коды чисел суммируются целиком без отделения знаков, перенос из старшего разряда теряется. Признаком переполнения служит несоответствие знаков результата и одного из равнознаковых операндов. В систему операций АЛУ дополнительно введена операция суммирования по модулю два (m2).

Процесс суммирования будет состоять из следующих тактов.

1. РгА := (ШВВ); ввод операнда А
2. РгВ := (ШВВ); ввод операнда В
3. РгН := (РгА), если Z = 1, идти к 15, иначе к 4; проверка А на 0
4. РгН := (РгВ), если Z = 1, идти к 16, иначе к 5; проверка В на 0
5. РгН := (РгА) + (РгВ); суммирование кодов чисел
6. РгС := (РгН); сохранение результата
7. РгН := (РгА) m2 (РгВ); проверка равенства знаков операндов (пп. 7, 8)
8. РгН := <- (РгН), если ВСЛ = 1, идти к 11, иначе к 9
9. РгН := (РгА) m2 (РгС); проверка равенства знаков А и С (пп. 9, 10)

10. РгН := <- (РгН), если ВСЛ = 1, идти к 14, иначе к 12
11. РгС := (РгН); сохранение результата
12. РгН := 0; формирование нулевого сообщения (пп. 12, 13)
13. РгО := (РгН); идти к 17
14. РгН := (РгК), идти к 19; формирование ненулевого сообщения
15. РгН := (РгВ), идти к 11; формирование результата С = В
16. РгН := (РгА), идти к 11; формирование результата С = А
17. ШВ := (РгО); вывод сообщения о нормальном результате
18. ШВ := (РгС); вывод результата. Конец.
19. РгО := (РгН)
20. ШВ := (РгО); вывод сообщения об ошибке. Конец.

В прил. 2 рассматриваются числовые примеры, иллюстрирующие работу процедуры суммирования дополнительных кодов чисел.

Отличие в суммировании обратных кодов чисел от суммирования дополнительных кодов чисел заключается в учёте циклического переноса из старшего разряда АЛУ в младший разряд. Циклический перенос реализуется аппаратно. Коды чисел также суммируются целиком без отделения знаков, алгоритм соответствует рис. 11. Внешне числовые примеры не отличаются от двух предыдущих за исключением возможного появления отрицательного нуля в результате операции. Поэтому конечный результат всегда необходимо проверять на подобный исход и в случае появления отрицательного нуля необходимо преобразовать его в положительный код нуля.

В обратном коде отрицательный ноль имеет представление 1.111…1. Для его выявления можно выполнить инкремент результата на +1 или инверсию с дальнейшей проверкой на положительный код нуля. Ниже приводится соответствующий фрагмент операции.

…
n. РгН := !(РгС), если Z = 1, идти к n+1, иначе к m; проверка на отрицательный ноль
n+1. РгС := (РгН), идти к m; инверсия результата
…
m. Конец.

Соответствующий фрагмент числового примера при (РгС) = 1.111…1 имеет вид

…
n. РгН := 0.000…0, Z = 1, идти к n+1
n+1. РгС := 0.000…0, идти к m
…
m. Конец.

Выполнение машинных операций в системе с плавающей точкой

При осуществлении операций в системе с плавающей точкой необходимо выполнение следующих условий:

– операнды поступают в нормализованном виде (т. е. модули мантисс M должны находиться в диапазоне 1 > M > = 0,5);

– результаты операции также должны быть нормализованы.

Мантиссы и порядки обрабатываются раздельно по правилам операций над числами с фиксированной точкой, но в отличие от мантисс порядки являются целыми числами. При отрицательном переполнении порядка результат может быть приравнен к нулю, а при положительном переполнении порядка фиксируется выход за диапазон представления.

Короткие операции (суммирование и вычитание) выполняются над числами с равными порядками. При выравнивании порядков один из операндов подвергается денормализации вправо (мантисса сдвигается вправо, а порядок инкрементируется). Если число сдвигов превышает разрядность мантиссы, операнд приравнивается к машинному нулю.

При выполнении длинных операций (умножение и деление) порядки складываются или вычитаются.

В процессе нормализации результата операции также выполняются преобразования порядка, что может также привести к его переполнению.

На рис. 12 приведён алгоритм процедуры выравнивания порядков pA и pB. Предполагается, что мантиссы Am и Bm нормализованы. Можно увеличивать меньший порядок, денормализуя вправо соответствующую мантиссу.

Ниже при рассмотрении примера машинного выравнивания порядков в ранее сформированную систему операций АЛУ добавлены операции вычитания кодов чисел, правого арифметического сдвига, инкремента на +1 и декремента на –1. Числа представлены в дополнительных кодах. Введены следующие сокращения: РгПА – регистр порядка А; РгПВ – регистр порядка В; РгПС – регистр порядка С; РгМА – регистр мантиссы А; РгМВ – регистр мантиссы В; РгМС – регистр мантиссы С; РгБ – буферный регистр.

1. РгН := (РгПА) – (РгПВ), если Z = 1, идти к 9, иначе к 2; сравнение порядков

2. РгБ := (РгН); сохранение разности порядков

3. РгН := <– (РгБ), если ВСЛ = 1, идти к 11, иначе к 4; проверка знака разности

4. РгН := (РгБ) – 1; денормализация Bm (пп. 4, 5, 6, 7, 8)

5. РгБ := (РгН)

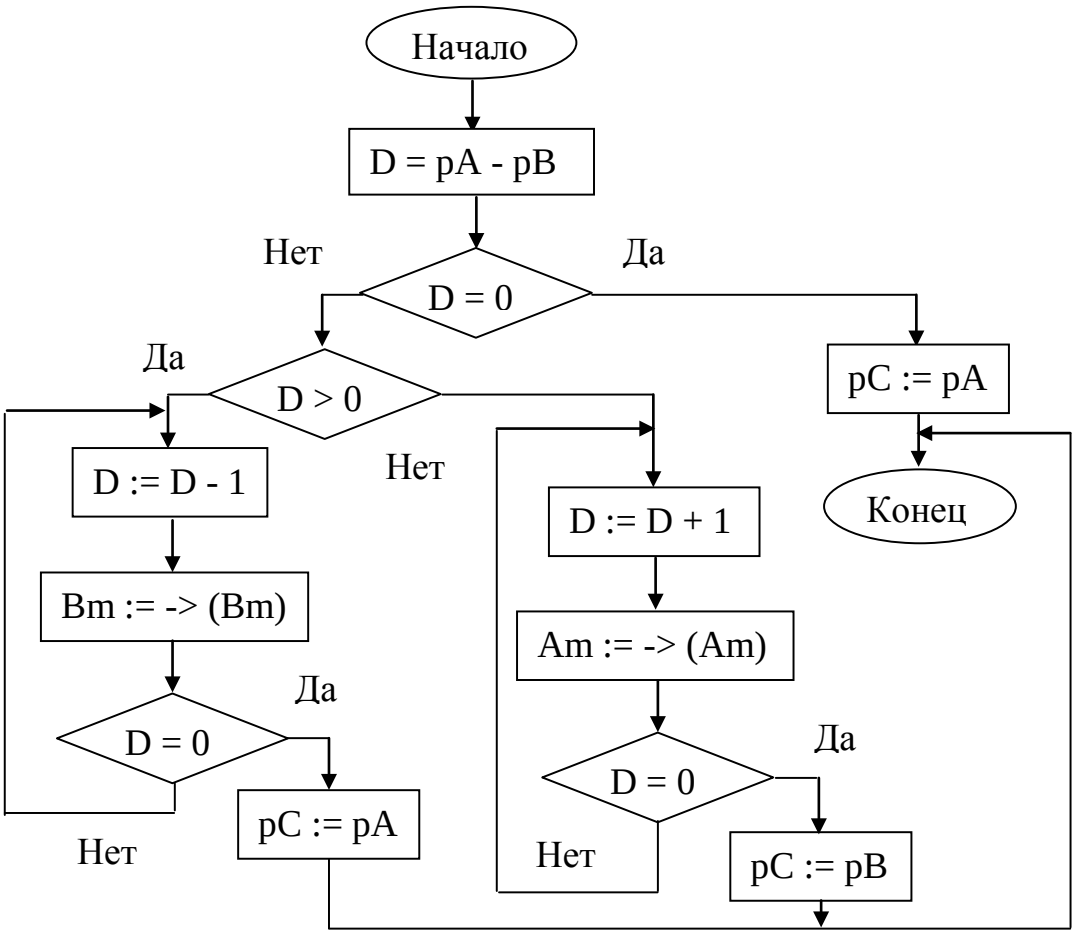

Рис. 12

6. РгН := –>(РгМВ)
7. РгМВ := (РгН)
8. РгН := (РгБ), если Z = 1, идти к 9, иначе к 4
9. РгН := (РгПА); формирование порядка С
10. РгПС := (РгН). Конец.
11. РгН := (РгБ) + 1; денормализация Am (пп. 11, 12, 13, 14, 15)
12. РгБ := (РгН)
13. РгН := –>(РгМА)
14. РгМА := (РгН)
15. РгН := (РгБ), если Z = 1, идти к 16, иначе к 11
16. РгН := (РгПВ), идти к 10; формирование порядка С

В прил. 3 приведены числовые примеры по выравниванию порядков согласно приведённой выше процедуре.

В примере 1 модуль pA был на 3 больше модуля pB, что привело к денормализации вправо Bm. Результирующий порядок pC приравнен к pA и равен 0.0000111.

В примере 2 pA = –7 был меньше pB = –5. В результате была денормализована мантисса Am на два разряда вправо. Результирующий порядок pC = –5 (т. е. большему порядку).

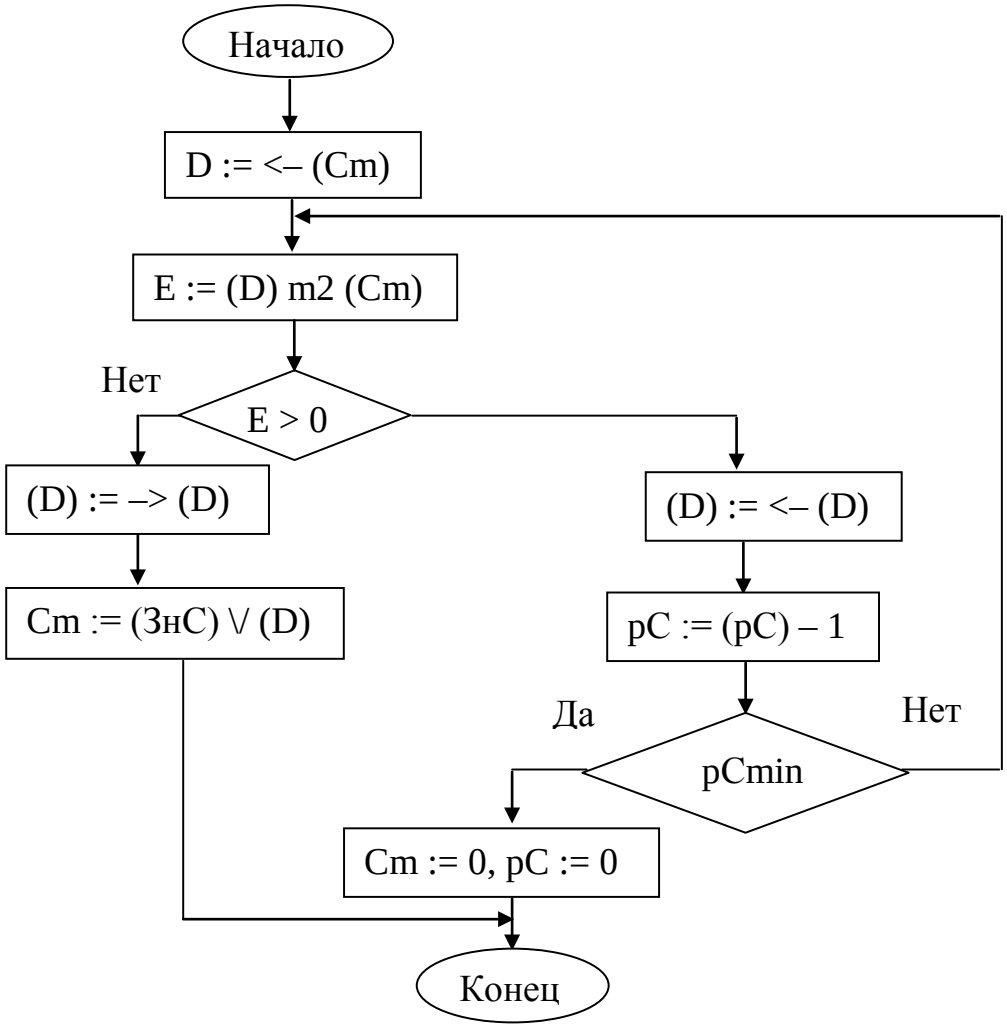

Рис. 13

На рис. 13 представлен алгоритм процедуры нормализации денормализованного вправо результата для отрицательных чисел в обратном и дополнительном кодах. При нормализации денормализованных вправо положительных чисел выполняется левый сдвиг мантиссы и инкремент порядка до появления единицы в старшем разряде модуля мантиссы независимо от кодов этих чисел. Алгоритм подобен рассмотренному выше.

Также может наблюдаться денормализация результата влево при переполнении результата суммирования равнознаковых мантисс. В этом случае нормализация осуществляется правым сдвигом мантиссы с восстановлением знака и инкрементом порядка результата. Если при этом порядок переполняется, то фиксируется переполнение результата и нормализация невозможна.

Выполнение длинной машинной операции в системе с фиксированной точкой программным неускоренным методом

На рис. 14 приведён алгоритм выполнения операции умножения чисел, меньших единицы по модулю, в прямых кодах с анализом младших разрядов множителя и сдвигом суммы частичных произведений вправо с округлением. Округление позволяет оставаться результату в пределах разрядной сетки процессора и обеспечить точность в половину веса младшего разряда. Выполнение операции умножения разворачивается во времени в соответствии со счётчиком циклов СЦ. Количество циклов на единицу меньше разрядности процессора при одном знаковом разряде (n–1 на рис. 14) либо на два при двух знаковых разрядах. В каждом цикле анализируется значение младшего бита МВ и осуществляется сдвиг МВ и МС вправо.

Введём в систему операций АЛУ операцию правого сдвига, формирующую выходной сигнал ВСП по состоянию младшего разряда. Под СЦ выделим регистр РгСЦ, а для накопления модуля произведения – регистр РгМС. Машинная реализация алгоритма умножения будет иметь следующий вид (считаем, что СЦ и константа 1.000…0 загружены в РгСЦ и РгК).

1. РгА := (ШВВ); ввод операнда А
2. РгВ := (ШВВ); ввод операнда В
3. РгН := (РгА), если Z = 1, идти к 24, иначе к 4; проверка на 0 операнда А
4. РгН := (РгВ), если Z = 1, идти к 24, иначе к 5; проверка на 0 операнда В
5. РгН := (РгН) m2 (РгА); формирование знака С (пп. 5, 6, 7)
6. РгН := (РгН) & (РгК)
7. РгС := (РгН)
8. РгН := (РгМС) m2 (РгМС); обнуление РгМС (пп. 8, 9)
9. РгМС := (РгН)
10. РгН := !(РгК); получение МА (пп. 10, 11, 12)
11. РгН := (РгА) & (РгН)
12. РгМА := (РгН)
13. РгН := –>(РгВ), если ВСП = 1, идти к 21, иначе к 14; анализ младшего разряда В
14. РгН := –>(РгМС); сдвиг МС
15. РгМС := (РгН); сохранение МС
16. РгН := (РгСЦ) – 1, если Z = 1, идти к 17, иначе к 20; декремент СЦ
17. РгН := (РгС); формирование результата (пп. 17, 18, 19)
18. РгН := (РгМС) \/ (РгН)
19. РгС := (РгН); идти к 25
20. РгСЦ := (РгН), идти к 13; сохранение СЦ
21. РгН := (РгМС); суммирование частичных произведений (пп. 21, 22)

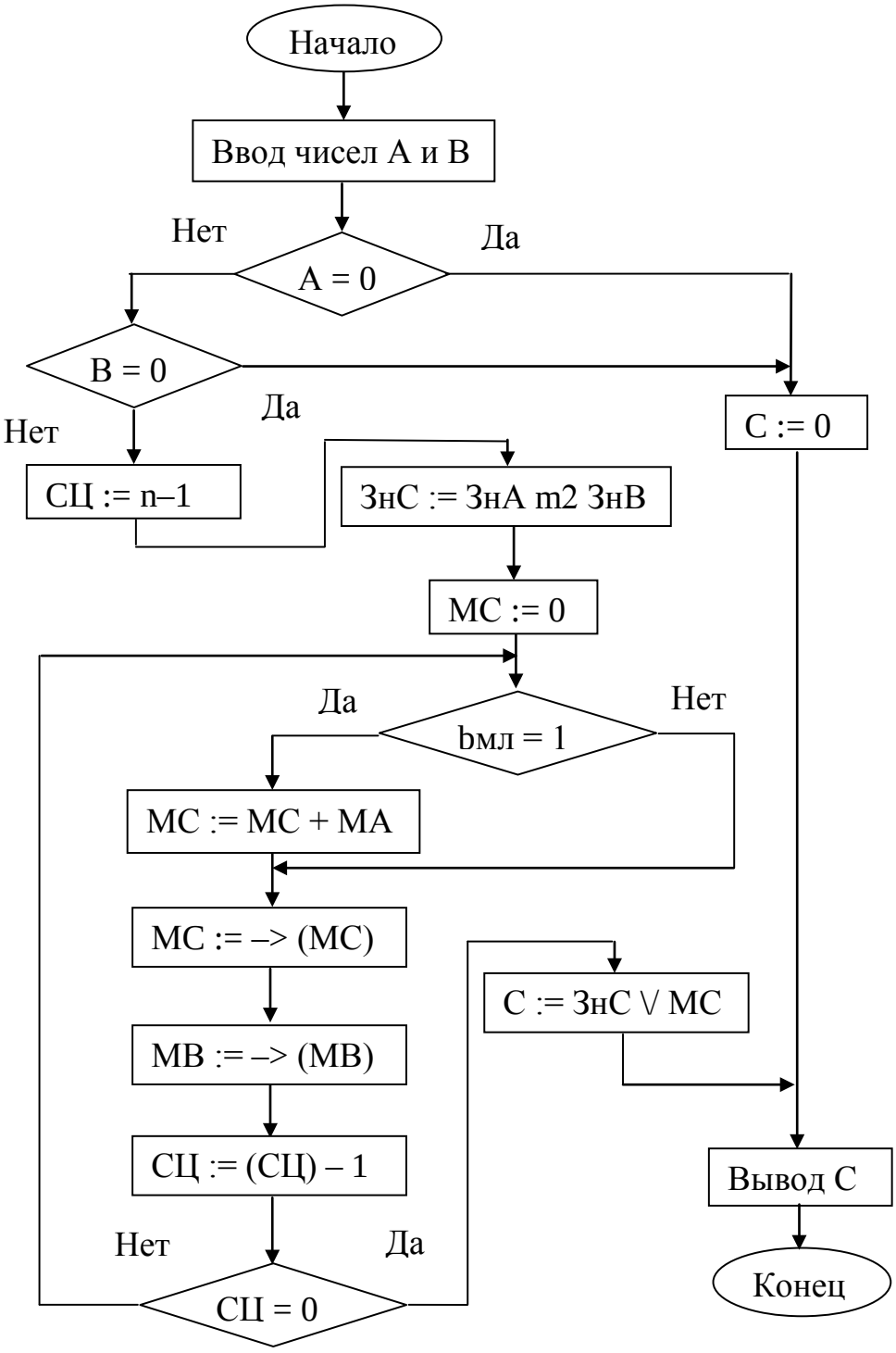

Рис. 14

22. РгН := (РгН) + (РгМА)
23. РгМС := (РгН), идти к 14
24. РгН := (РгС) m2 (РгС), идти к 19; обнуление РгС
25. ШВ:= (РгС). Конец.

Выполнение длинной машинной операции в системе с фиксированной точкой программным ускоренным методом

Программные методы ускорения предусматривают модификацию используемых программных решений, позволяющую устранить непроизводительные затраты времени в ходе вычислительного процесса. Например, через уменьшение количества вспомогательных пересылок, через сохранение промежуточных результатов, которые могут быть использованы в дальнейшем ходе вычислений, через уменьшение количества циклов и пр.

В качестве примера рассмотрим выполнение операции умножения чисел меньших единицы по модулю в прямых кодах младшими разрядами вперёд с анализом двух разрядов, что позволяет снизить количество циклов при вычислении суммы частичных произведений. Действия по реализации операции над модулями операндов Am и Bm в текущем цикле умножения в зависимости от сочетания битов в анализируемой паре разрядов множителя b1b0 и переноса pi–1 из предыдущей пары представлены в табл.1. Символами Cm, pi, pi–1 и САП2 обозначены соответственно текущие значения суммы частичных произведений, переноса из текущей пары разрядов, переноса из предыдущей пары разрядов и операция арифметического правого сдвига на два разряда.

Таблица 1

b1 b0	P(i–1)	Действие над кодами	P(i)
0 0	0	Cm := САП2 (Cm), Bm := САП2 (Bm)	0
0 1	0	Cm := САП2 (Cm + Am), Bm := САП2 (Bm)	0
1 0	0	Cm := САП2 (Cm + 2Am) , Bm := САП2 (Bm)	0
1 1	0	Cm := САП2 (Cm – Am) , Bm := САП2 (Bm)	1
0 0	1	Cm := САП2 (Cm + Am) , Bm := САП2 (Bm)	0
0 1	1	Cm := САП2 (Cm + 2Am) , Bm := САП2 (Bm)	0
1 0	1	Cm := САП2 (Cm – Am) , Bm := САП2 (Bm)	1
1 1	1	Cm := САП2 (Cm) , Bm := САП2 (Bm)	1

Алгоритм выполнения операции умножения приведён на рис. 15. В алгоритме использованы следующие сокращения: Зн X – знак операнда X ; Xm – модуль операнда X (где операнд A – множимое, операнд B – множитель, операнд C – произведение); СЦ – счётчик циклов (для n-разрядного множителя число циклов равно n/2); S – текущая двухразрядная группа множителя B; s1, s0 – текущее значение битов группы S; P – входной перенос p(i–1) i-й группы S и выходной перенос p(i) этой группы в текущем цикле (i = 1…n/2); m2 – символ операции «сумма по модулю 2»; САП2 – символ операции «сдвиг арифметический правый на два разряда».

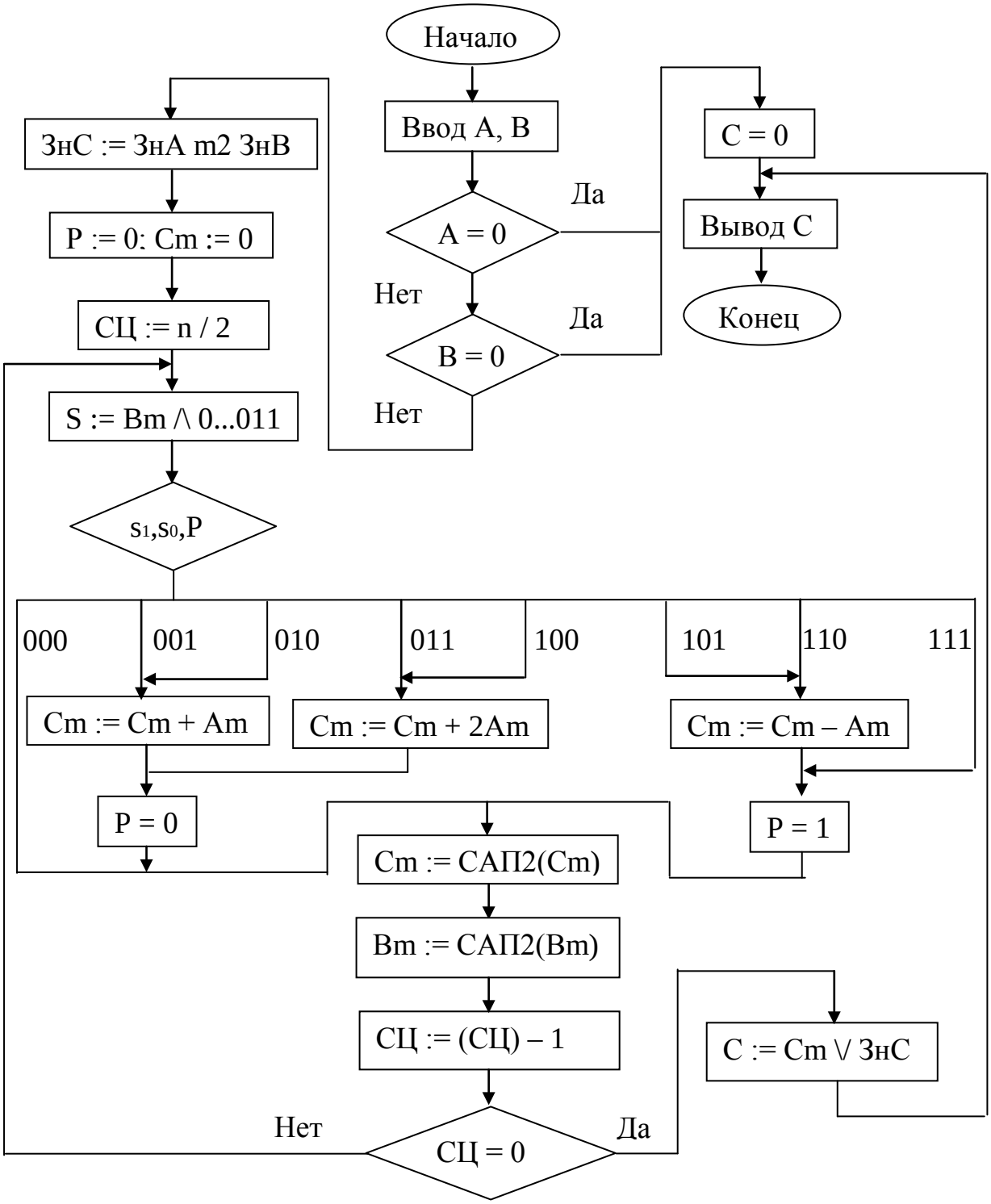

Рис. 15

В алгоритме применяется операция арифметического правого сдвига на два разряда, что позволяет расширять при сдвиге знаковый бит операнда и осуществлять в каждом проходе цикла умножения округление суммы частичных произведений. В результате будет получено округлённое произведение в одинарном формате.

Для отображения алгоритма на уровне машинных операций введём дополнительно в ранее рассмотренную систему инструкций ОУ операцию правого арифметического сдвига на два разряда (данная операция позволит ускорить сдвиги множителя и суммы частичных произведений), а в аппаратные средства добавим регистр переноса РгП, регистр удвоенного модуля Рг2МА и регистр РгДМА – регистр дополнительного кода (–Am).

Машинная реализация данного алгоритма примет следующий вид.

1. РгА := (ШВВ); ввод операнда А
2. РгВ := (ШВВ); ввод операнда В
3. РгК := (ШВВ); ввод константы 10…0
4. РгН := (РгА), если Z = 1, идти к 45, иначе к 5; проверка А на 0
5. РгН := (РгВ), если Z = 1, идти к 45, иначе к 6; проверка В на 0
6. РгН := (РгН) m2 (РгА); формирование знака С (пп. 6, 7, 8)
7. РгН := (РгН) & (РгК)
8. РгС := (РгН)
9. РгП := (ШВВ); обнуление РгП
10. РгСЦ := (ШВВ); установка СЦ
11. РгН := !(РгК); формирование Am (пп. 11, 12, 13)
12. РгН := (РгН) & (РгА)
13. РгМА := (РгН)
14. РгН := СЛЛ(РгН); формирование 2Am
15. Рг2МА := (РгН)
16. РгН := !(РгМА) + 1; формирование (–Am)
17. РгДМА := (РгН)
18. РгН := !(РгК); формирование Bm (пп. 18, 19, 20)
19. РгН := (РгН) & (РгВ)
20. РгМВ := (РгН)
21. РгМС := (ШВВ); обнуление Cm
22. РгН := (РП), если Z = 1, идти к 23, иначе к 24; анализ группы разрядов Bm и P
23. РгН := (РгМВ), если 00, идти к 25,
    если 01, идти к 35,
    если 10, идти к 39,
    иначе к 41
24. РгН := (РгМВ), если 00, идти к 35,
    если 01, идти к 39,
    если 10, идти к 41,
    иначе к 43

25. РгН := САП2(РгМС); сдвиг Cm
26. РгМС := (РгН)
27. РгН := САП2(РгМВ); Сдвиг Bm
28. РгМВ := (РгН)
29. РгН := (РгСЦ) – 1, если Z = 1, идти к 31, иначе к 30; декремент СЦ
30. РгСЦ := (РгН), идти к 22
31. РгН := (РгС); формирование С (объединение Cm и ЗнС)
32. РгН := (РгН) \/ (РгМС)
33. РгС := (РгН)
34. ШВ := (РгС). Конец.
35. РгН := (РгМС); формирование суммы частичных произведений
36. РгН := (РгН) + (РгМА)
37. РгП := (ШВВ); обнуление РгП
38. РгМС := (РгН), идти к 25
39. РгН := (РгМС); формирование суммы частичных произведений
40. РгН := (РгН) + (Рг2МА), идти к 37
41. РгН := (РгМС); формирование суммы частичных произведений
42. РгН := (РгН) + (РгДМА)
43. РгП := (ШВВ); 0..01 установка РгП
44. РгМС := (РгН), идти к 25
45. РгС := (ШВВ); идти к 34; обнуление результата

В рассмотренной машинной реализации алгоритма загрузка рабочих констант осуществляется по мере надобности с входной шины, что позволяет ускорить процесс программных вычислений. В свою очередь умножение на два разряда в каждом цикле формирования суммы частичных произведений сокращает в два раза количество циклов, что при большой разрядности операндов даст существенный выигрыш в быстродействии выполнения операции умножения.

Числовой пример для перемножения разнознаковых чисел представлен в прил. 4. В данном примере использована расширенная разрядная сетка – слева от знакового бита введён дополнительный разряд, который позволяет отличить удвоенное значение модуля множимого (см. п. 15) от отрицательного числа в дополнительном коде (см. пп. 16, 17).

Особенности выполнения операции умножения над двоичными числами с фиксированной запятой в инверсных кодах

Обратный и дополнительный коды относятся к инверсным кодам. Значащая n-разрядная часть отрицательного числа в дополнительном коде у правильных дробей находится по формуле Вдк- = 1– |B|, а в обратном коде – по формуле Вок- = 1– |B| – 2↑(–n). Соответственно, умножение на отрицательное

число в этих кодах отличается от умножения на отрицательное число в прямых кодах. Коды положительных чисел Вдк+ и Вок+ равны |B|, следовательно, умножение на положительное число в этих кодах не отличается от подобной операции в прямых кодах.

Формула умножения на Вдк- определяется выражением

$$C = A * Вдк- = A * (1– |B|) = A – A*|B|,$$

из которого находим искомое значение произведения:

$$–A * |B| = A * Вдк- – A.$$

Таким образом, после получения суммы частичных произведений при умножении на Вдк- необходимо осуществить её коррекцию на (–A). Знак произведения получается автоматически после коррекции.

Формула умножения на Вок- определяется выражением

$$C = A * Вок- = A * (1– |B| – 2\uparrow(–n)),$$

откуда находим

$$–A * |B| = A * Вок- – A + A*2\uparrow(–n).$$

Рассмотрим соответствующие числовые примеры.

В качестве примера перемножим числа A и B при A = 0,0110 и B = 0,1010 с различными сочетаниями знаков в дополнительных кодах.

A>0, B>0

```
    0,0110
  * 0,1010
    00000
+   00110
  00000
 00110
-------
0.00111100
```

A>0, B<0

```
    0,0110
  * 1,0110
    00000
+   00110
  00110
 00000
-------
0.00100100
-
  0,0110
-------
1,11000100
```

A<0, B<0

```
    1,1010
  * 1,0110
    00000
+   1.1111010
  1.111010
 0.00000
---------
1.11011100
+
  0,0110
---------
0,00111100
```

Случай A>0, B>0 используется как тестовый пример для получения модуля произведения. В случае A>0, B<0 числа разнознаковые при отрицательном B, что даёт в результате отрицательное произведение в дополнительном коде. Третий пример при A<0, B<0 иллюстрирует перемножение равнознаковых отрицательных чисел в дополнительных кодах. В результате получаем положительное произведение, совпадающее с результатом тестового примера. В этом примере расширение разрядной сетки для отрицательных частичных произведений осуществляется состоянием знакового бита. Расширение со стороны младших разрядов для отрицательных чисел в дополнительном коде осуществляется нулями по умолчанию и в примере наглядно не представлено.

Выполним те же операции в обратных кодах. Тестовый пример остаётся без изменений, поэтому рассмотрим две оставшиеся ситуации. В разнознаковом примере при B<0 получено отрицательное произведение в обратном коде. При перемножении отрицательных чисел в обратном коде формируется положительное произведение, совпадающее с результатом тестового примера.

A>0, B<0

```
    0,0110
  *
    1,0101
     00110
+    00110
    00000
   00110
  0,00100100
–
  0,0110
  1,11000011
```

A<0, B<0

```
    1,1001
  *
    1,0101
   111111001
+  111111001
  000000000
 111100111
  1,11011011
–
  1,10011111
  0,00111100
```

В последнем случае учитывались циклические переносы из старших разрядов и инверсное представление нулей в обратных кодах при расширении разрядной сетки для отрицательных частичных произведений в отличие от предыдущего примера, где расширения частичных положительных произведений заполняются нулями по умолчанию.

Табличные методы вычисления

Табличные методы вычисления функций предусматривают наличие массива (таблицы) значений функции на возможных наборах аргументов. Для обращения к массиву в качестве адреса используется конкретный набор аргументов, от которого требуется вычислить значение функции. Между каждым набором аргументов и выбираемым из таблицы значением функции существует взаимно однозначное соответствие (рис. 16, 17).

На рис. 16 приведена двумерная таблица, в которой по одной координате изменяется операнд A, по другой координате – операнд B. На пересечении строк и столбцов таблицы размещаются значения функции A + B. Эта же задача решается с помощью линейной таблицы (см. рис. 17) с числом строк, равным количеству сочетаний значений A и B. Каждое сочетание является номером строки таблицы (левый столбец), а в правом столбце каждой строки содержится значение функции, соответствующее значениям операндов в данном сочетании.

A \ B	0	1	2	3
0	0	1	2	3
1	1	2	3	4
2	2	3	4	5
3	3	4	5	6

Рис. 16

A B	F(A B)
0 0	0
0 1	1
0 2	2
0 3	3
1 0	1
1 1	2
1 2	3
1 3	4
2 0	2
2 1	3
2 2	4
2 3	5
3 0	3
3 1	4
3 2	5
3 3	6

Рис. 17

Линейная таблица может интерпретироваться как массив ячеек памяти с числом ячеек, равным числу строк таблицы, в каждой ячейке которого содержится значение функции, а соответствующее сочетание операндов является адресом ячейки. Подобная таблица позволяет получить значение функции за один цикл обращения к памяти.

Табличные вычисления относятся к аппаратным методам ускорения. Для размещения таблицы используется либо специальный блок табличного преобразователя, либо раздел основной памяти системы. Метод обеспечивает максимальное быстродействие при вычислении значений сложных функций. Однако при этом требуются и существенные аппаратные затраты. Так, например, одноместная функция от n-разрядного двоичного операнда может иметь 2↑n значений, каждое из которых кодируется k-разрядным двоичным кодом. Пусть n = 16, k = 16, тогда требуемый объём таблицы составит 65536 *16 бит. Подобная таблица позволит, например, получить обратное число от n-разрядного операнда за один-два командных цикла процессора.

Таблица для двуместной функции умножения n-разрядных операндов с получением 2n-разрядных произведений при n = 16 будет иметь 2↑16*2 32-разрядных результатов, т. е. Её объём составит 128 Гбит. Время вычисления составит два-четыре командных цикла.

На практике для снижения аппаратных затрат при табличных вычислениях прибегают к компромиссу – таблично-алгоритмическим вычислениям. При этом часть преобразований при вычислении значения функции осуществляется программными средствами, а часть вычислений возлагается на табличные преобразования. В свою очередь, табличные преобразования в этом случае могут быть как прямыми, так и косвенными – через ряд вспомогательных функций.

Ниже рассматривается ряд методов организации таблично-алгоритмических вычислений.

Выполнение длинной машинной операции таблично-алгоритмическим методом

Таблично-алгоритмические вычисления предполагают использование таблиц косвенных функций, обращение к которым позволяет получить частичные результаты, из которых алгоритмическим путём формируют конечный результат.

Рассмотрим в качестве примера получение произведения двух чисел с фиксированной точкой с использованием таблицы квадратов чисел. Считаем, что обращение к таблице осуществляется через регистр адреса таблицы РгАТ, а частичный результат с выхода таблицы, имеющий удвоенный формат, загружается в выходной регистр таблицы РгТ и в регистр-расширитель РгТР. Для работы с удвоенным форматом суммы частичных произведений необходим и регистр-расширитель РгСР.

В качестве базы рассматриваемого метода используется формула

$$(A + B)\uparrow 2 = A\uparrow 2 + 2AB + B\uparrow 2,$$

из которой следует

$$AB = ((A + B)\uparrow 2 – A\uparrow 2 – B\uparrow 2)/ 2.$$

По сравнению с прямым табличным преобразованием, когда адрес обращения к таблице имеет разрядность 2n (см. ранее), в данном методе адрес обращения будет n-разрядным, объём таблицы сократится до $2\uparrow n * 2n$ и при n = 16 составит $65536 * 32 = 2\uparrow 16 * 2\uparrow 5 = 2$ Мбит, что существенно меньше 128 Гбит. Однако время вычисления увеличится в 5–6 раз.

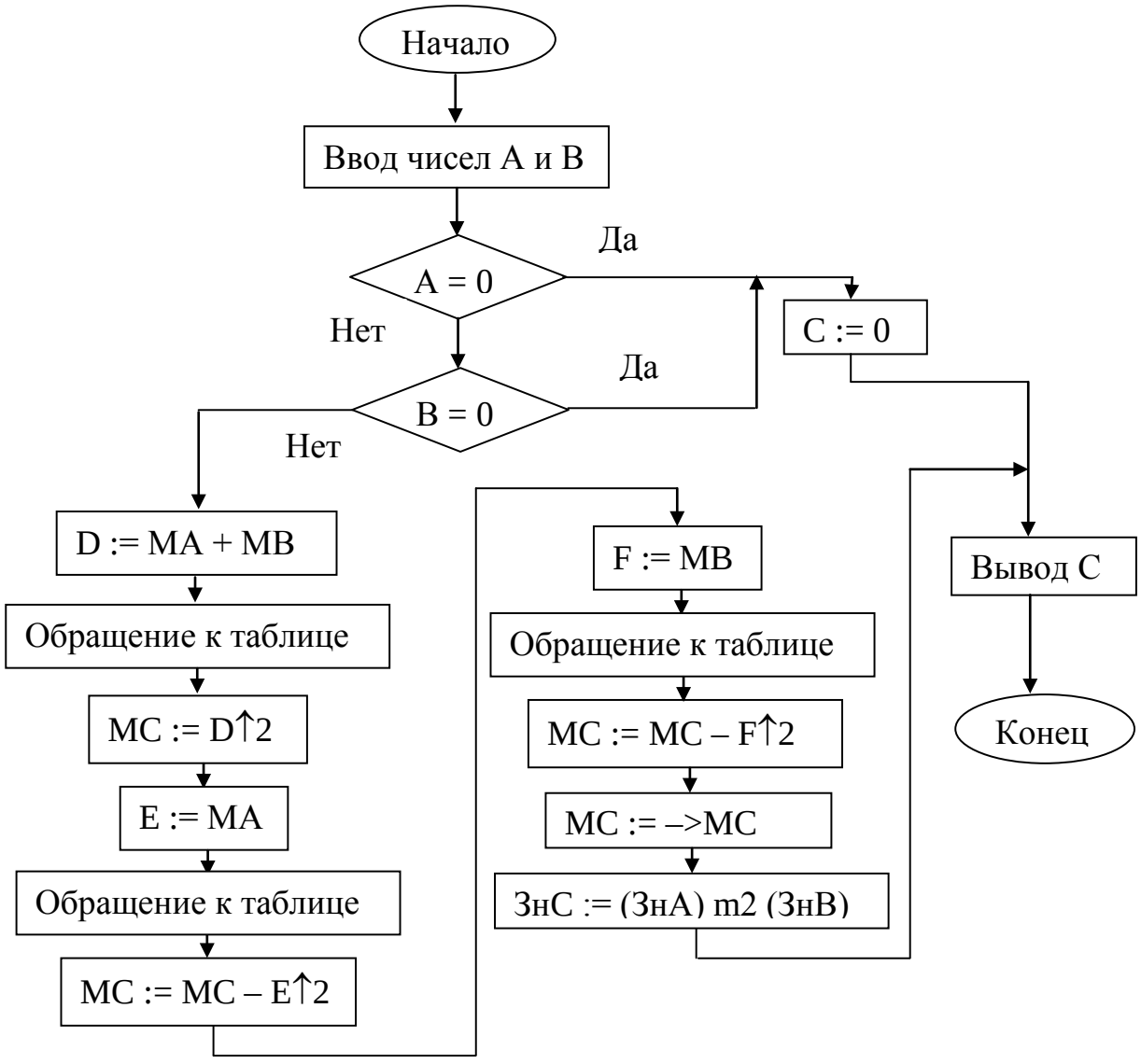

Рис. 18

Алгоритм вычисления произведения представлен на рис. 18. Машинная операция по данному алгоритму выполняется следующим образом. Считаем, что операнды и константа 10…0 загружены в РгА, РгВ и РгК соответственно.

1. РгН := (РгА), если Z = 1, идти к 50, иначе к 2; проверка на 0 операнда А
2. РгН := (РгВ), если Z = 1, идти к 50, иначе к 3; проверка на 0 операнда В
3. РгН := !(РгК); выделение модуля А (пп. 3, 4, 5)
4. РгН := (РгА) & (РгН)
5. РгМА := (РгН)
6. РгН := !(РгК); выделение модуля В (пп. 6, 7, 8)
7. РгН := (РгВ) & (РгН)
8. РгМВ := (РгН)
9. РгН := (РгМА) + (РгН); суммирование модулей
10. РгАТ := (РгН); обращение к таблице (пп. 10, 11)
11. РгТР, РгТ := [(РгАТ)]
12. РгН := (РгТ); сохранение квадрата суммы модулей (пп. 12, 13, 14, 15)
13. РгС : = (РгН)
14. РгН := (РгТР)
15. РгСР := (РгН)
16. РгН := (РгМА); получение квадрата МА (пп. 16, 17, 18)
17. РгАТ := (РгН)
18. РгТР, РгТ := [(РгАТ)]
19. РгН := (РгТ); вычитание квадрата МА (пп. 19–29)
20. РгН := (РгС) – (РгН), если заём = 1, идти к 25, иначе к 21
21. РгС := (РгН)
22. РгН := (РгТР)
23. РгН := (РгСР) – (РгН)
24. РгСР := (РгН), идти к 30
25. РгС := (РгН)
26. РгН := (РгТР)
27. РгН := (РгСР) – (РгН)
28. РгН := (РгН) – 1
29. РгСР := (РгН)
30. РгН := (РгМВ); получение квадрата МВ (пп. 30, 31, 32)
31. РгАТ := (РгН)
32. РгТР, РгТ := [(РгАТ)]
33. РгН := (РгТ); вычитание квадрата МВ (пп. 33–43)
34. РгН := (РгС) – (РгН), если заём = 1, идти к 39, иначе к 35
35. РгС := (РгН)
36. РгН := (РгТР)
37. РгН := (РгСР) – (РгН)
38. РгСР := (РгН), идти к 44
39. РгС := (РгН)

40. РгН := (РгТР)
41. РгН := (РгСР) – (РгН)
42. РгН := (РгН) – 1
43. РгСР := (РгН)
44. РгН := СЛП (РгС); сдвиг вправо модуля результата (пп. 44, 45)
45. РгС := (РгН)
46. РгН := (РгD) m2 (РгА); формирование знака С (пп. 46, 47)
47. РгН := (РгН) & (РгК)
48. РгН := (РгС) \/ (РгН); формирование результата
49. РгС := (РгН), идти к 52
50. РгН := (РгК) m2 (РгК); обнуление результата
51. РгС := (РгН)
52. РгН := (РгК) m2 (РгК); формирование сообщения
53. РгО := (РгН). Конец.

Выполнение длинных арифметических операций с применением таблиц логарифмирования и потенцирования

В основу метода положены следующие свойства логарифмов:

для C = A * B ln C = ln (A * B) = ln A+ ln B;

и

для C = A / B ln C = ln (A / B) = ln A – ln B.

Для получения значения С необходимо выполнить потенцирование

C = e↑ln C.

Соответственно, в набор аппаратных средств процессора должны входить таблица функции логарифмирования от n-разрядного аргумента и таблица функции потенцирования от m-разрядного аргумента, где m – разрядность представления значения логарифма, определяющая точность вычисления. В частном случае при m = n объём каждой таблицы составит 2↑n * n и суммарный объём будет равен 2↑(n + 1) * n. В общем случае суммарный объём таблиц определяется выражением (2↑n * m) + (2↑m * 2n) для 2n-разрядного результата или (2↑n * m) + (2↑m * n) для n-разрядного результата. При выполнении операций над знаковыми числами необходимо оперировать с их модулями, так как функция логарифма определена для положительных значений аргумента.

Алгоритм вычисления произведения двух целых знаковых чисел представлен на рис. 19.

Машинная реализация алгоритма должна учитывать наличие двух таблиц. С этой целью будем использовать два регистра адреса таблиц: РГАТЛ (для таблицы логарифмов) и РГТАП (для таблицы потенцирования). Результаты опроса

любой таблицы будем размещать в прежних регистрах РГТ и РГТР. Причём результат логарифмирования ограничим n разрядами с размещением в РГТ, а результат потенцирования будем размещать в (2n – 1)-разрядном формате в РГТР, РГТ, оставляя старший разряд РГТР под знак произведения. Промежуточные значения логарифмирования будем размещать в регистре РгЛ. Считаем, что в РгК хранится константа 10…0, а операнды A и B – в регистрах РгА и РгВ соответственно.

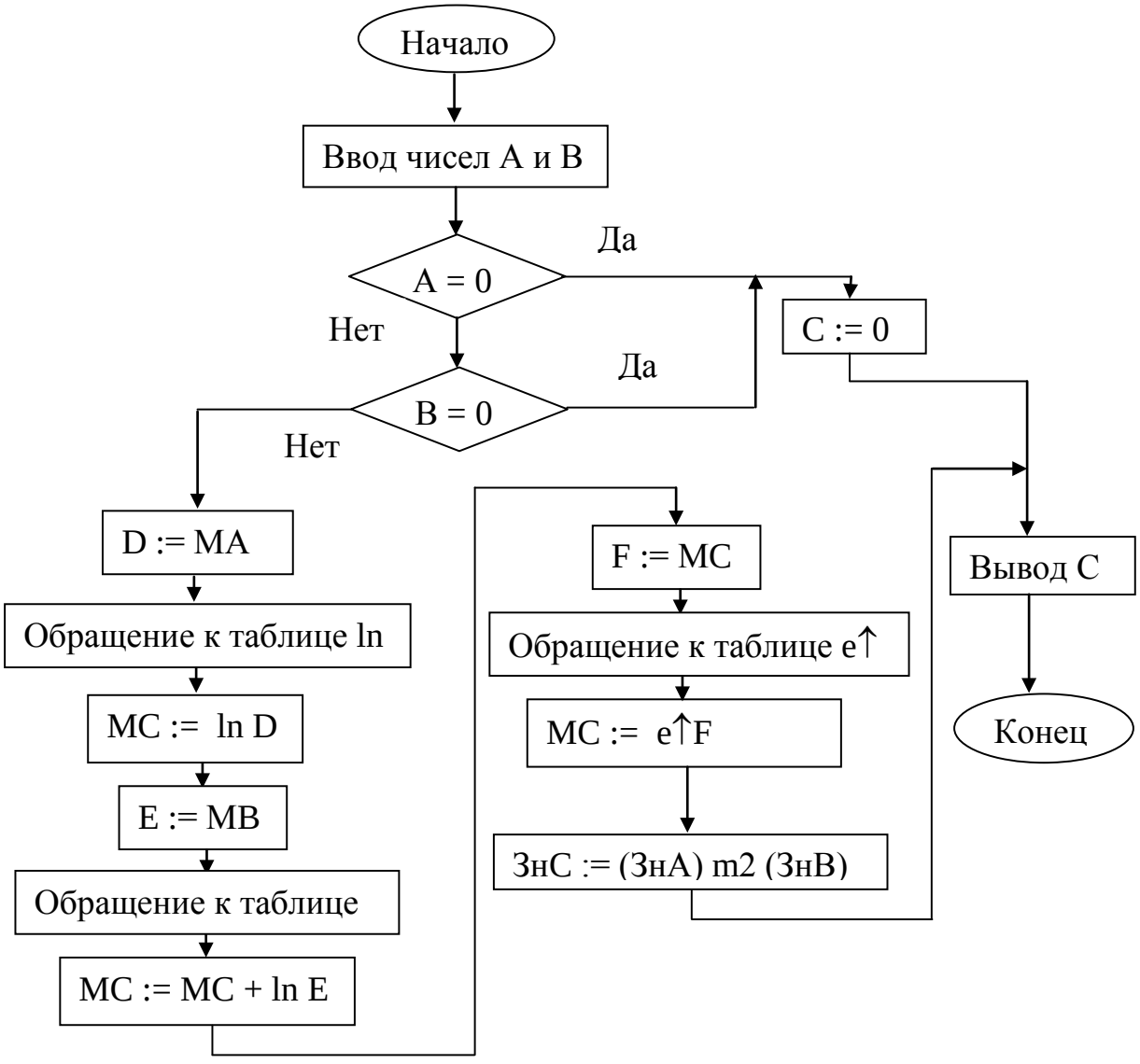

Рис. 19

1. РгН := (РгА), если Z = 1, идти к 28, иначе к 2; проверка на 0 операнда А
2. РгН := (РгВ), если Z = 1, идти к 28, иначе к 3; проверка на 0 операнда В
3. РгН := !(РгК); выделение модуля А (пп. 3, 4, 5)
4. РгН := (РгА) & (РгН)
5. РгМА := (РгН)

6. РгН := !(РгК); выделение модуля В (пп. 6, 7, 8)
7. РгН := (РгВ) & (РгН)
8. РгМВ := (РгН)
9. РгАТЛ := (РгН); формирование ln Bm (пп. 9 – 12)
10. РгТ := [(РгАТЛ)]; опрос таблицы
11. РгН := (РгТ)
12. РгЛ := (РгН)
13. РгН := (РгМА); формирование ln Am (пп. 13 – 15)
14. РгАТЛ := (РгН)
15. РгТ := [(РгАТЛ)]
16. РгН := (РгТ) + (РгЛ); получение ln Bm + ln Am
17. РгАТП := (РгН); обращение к таблице потенцирования
18. РгТР, РгТ := [(РгАТП)]
19. РгН := (РгТ); размещение модуля произведения (пп. 19 – 22)
20. РгС := (РгН)
21. РгН := (РгТР)
22. РгСР := (РгН)
23. РгН := (РгА); формирование знака произведения (пп. 23 – 25)
24. РгН := (РгН) m2 (РгВ)
25. РгН := (РгН) & (РгК)
26. РгН := (РгСР) \/ (РгН); формирование результата
27. РгСР := (РгН). Конец.
28. РгН := (РгК) m2 (РгК); обнуление результата
29. РгС := (РгН), идти к 27

Алгоритм деления двух целых разнознаковых чисел представлен на рис. 20. Отличие от предшествующего алгоритма заключается в дополнительных проверках делителя на 0 и в превышении или равенстве модуля делимого над модулем делителя (для целочисленных операций). Как и ранее, будем использовать n-разрядное кодирование значений логарифмов, а также n-разрядное представление кода частного с одним знаковым битом.

Машинная реализация алгоритма имеет следующий вид.

1. РгН := (РгВА), если Z = 1, идти к 30, иначе к 2; проверка В на 0
2. РгН := (РгА), если Z = 1, идти к 29, иначе к 3; проверка А на 0
3. РгН := !(РгК); выделение модуля А (пп. 3, 4, 5)
4. РгН := (РгА) & (РгН)
5. РгМА := (РгН)
6. РгН := !(РгК); выделение модуля В (пп. 6, 7, 8)
7. РгН := (РгВ) & (РгН)
8. РгМВ := (РгН)

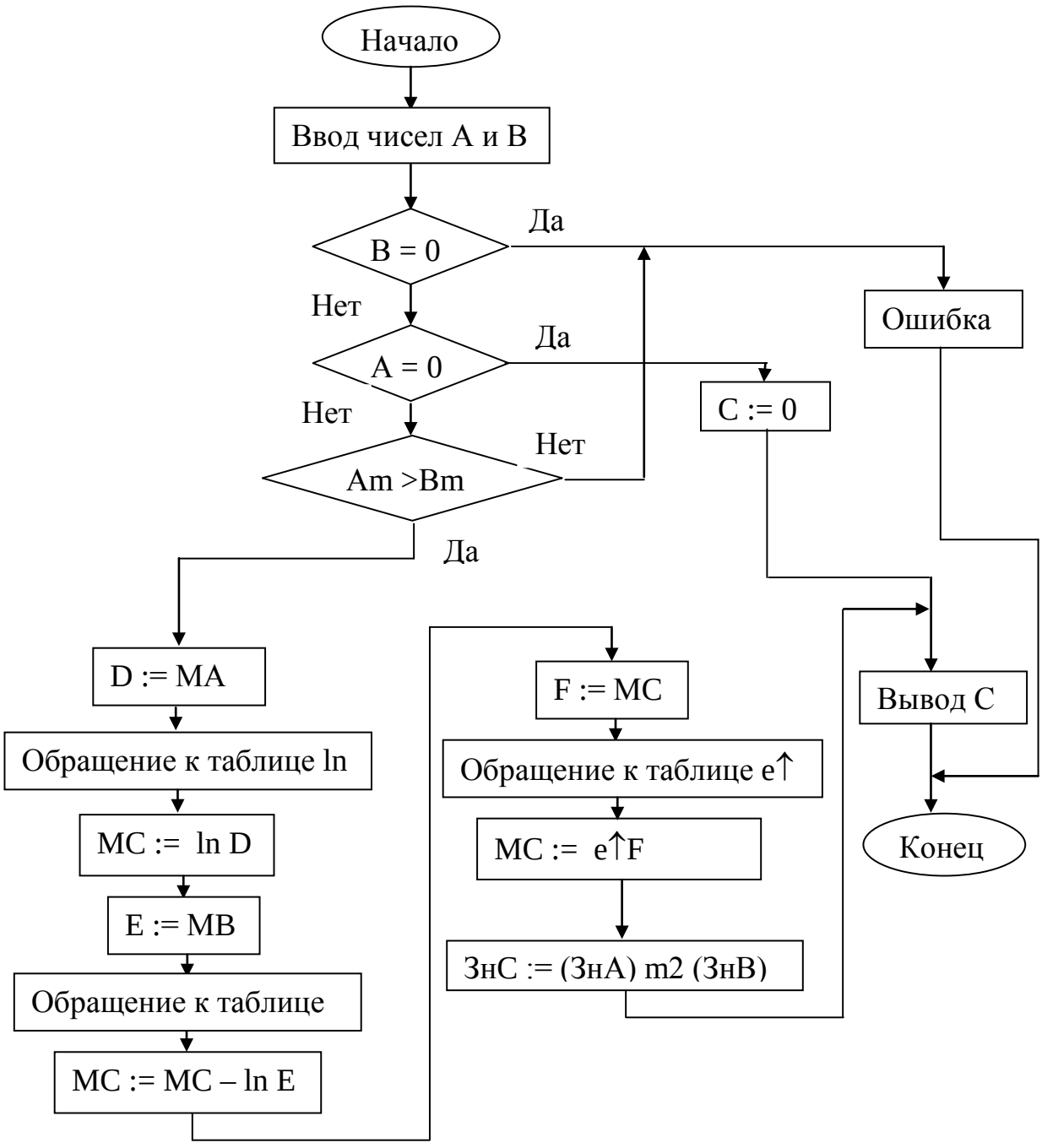

Рис. 20

9. РгН := (РгМА) – (РгН); сравнение Bm и Am

10. РгН := СЛЛ (РгН), если ВСЛ = 1, идти к 11, иначе к 30; анализ знака остатка

11. РгН := (РгМВ)

12. РгАТЛ := (РгН); формирование ln Bm (пп. 12–15)

13. РгТ := [(РгАТЛ)]; опрос таблицы

14. РгН := (РгТ)

15. РгЛ := (РгН)

16. РгН := (РгМА); формирование ln Am (пп. 16–18)
17. РгАТЛ := (РгН)
18. РгТ := [(РгАТЛ)]
19. РгН := (РгТ) – (РгЛ); получение ln Am – ln Bm
20. РгАТП := (РгН); обращение к таблице потенцирования
21. РгТ := [(РгАТП)]
22. РгН := (РгТ); размещение модуля частного (пп. 22, 23)
23. РгС := (РгН)
24. РгН := (РгА); формирование знака частного (пп. 24–26)
25. РгН := (РгН) m2 (РгВ)
26. РгН := (РгН) & (РгК)
27. РгН := (РгС) \/ (РгН); формирование результата
28. РгС := (РгН). Конец.
29. РгН := (РгК) m2 (РгК), идти к 28; обнуление результата
30. РгН := (РгК); формирование сообщения об ошибке
31. РгО := (РгН). Конец.

Применение данного метода наиболее эффективно для целых чисел, так как для малых дробных чисел, меньших единицы, значение логарифмической функции стремится к минус бесконечности и требует значительной разрядной сетки для его отображения. Соответственно, при ограниченной разрядной сетке в этом случае точность вычислений будет снижена.

Достаточная точность вычислений обеспечивается и для нормализованных чисел в системе с плавающей запятой, так как значения мантисс $\geq 0{,}5$ и соответствующие значения логарифмической функции для своего отображения не требуют существенной разрядной сетки.

Таблично-алгоритмический аддитивный метод умножения

Метод предусматривает формирование полноформатного произведения через сумму частичных произведений, формируемых через табличное умножение групп битов сомножителей, выделяемых из полноформатных кодов сомножителей.

Формальное описание метода при разделении полноформатных n-разрядных кодов сомножителей пополам имеет вид

$$A * B = Ам * Вм + Ам * Вс * 2\uparrow(n/2) + Ас * Вм * 2\uparrow(n/2) + Ас * Вс * 2\uparrow n,$$

где Ам, Вм – младшие группы разрядов сомножителей А и В;

Ас, Вс – старшие группы разрядов сомножителей А и В;

$A = Ас * 2\uparrow(n/2) + Ам;$

$B = Вс * 2\uparrow(n/2) + Вм.$

На структурном уровне процесс умножения младшими разрядами вперёд может быть представлен в форме, приведённой на рис. 21.

Машинная реализация метода требует наличия таблицы перемножения n/2-разрядных кодов и операции правого сдвига на n/2 разрядов. В общем случае формируется 2n-разрядное произведение, при формировании которого требуется наличие 2n-разрядных регистров. При перемножении целых чисел это требование обязательно. При перемножении дробных чисел, меньших единицы, по модулю, возможно округление суммы частичных произведений на n/2 разрядов при каждом сдвиге, что позволит использовать только n-разрядные регистры.

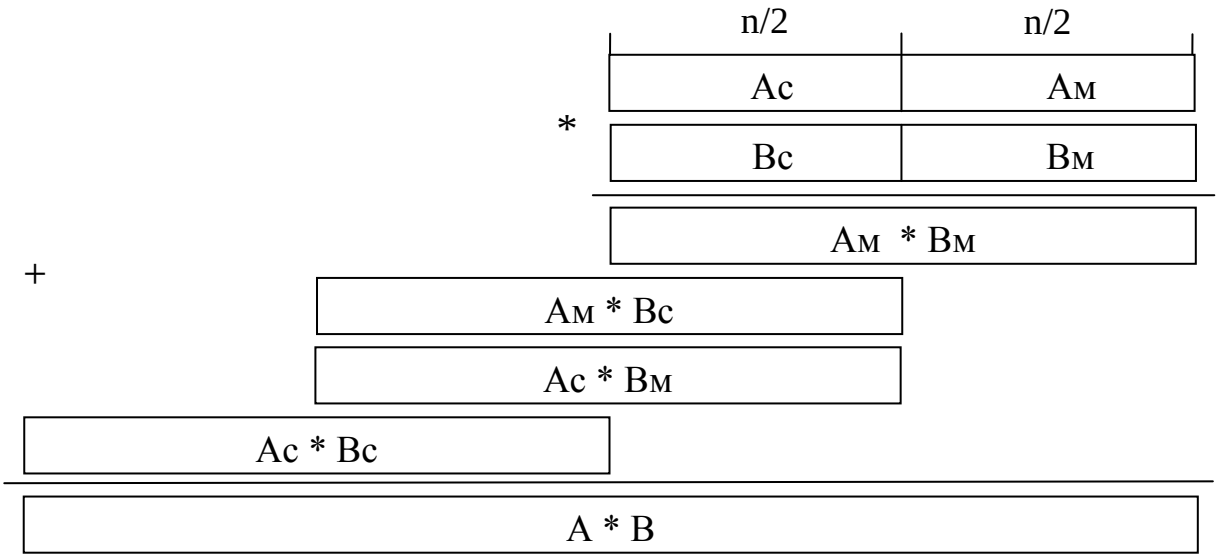

Рис. 21

Схема алгоритма умножения 16-разрядных дробных чисел в прямом коде, меньших единицы по модулю, по данному методу представлена на рис. 22. В схеме алгоритма использованы обозначения: МХ – модуль операнда Х; СЛП8 – сдвиг логический правый на 8 разрядов; СЛЛ8 – сдвиг логический левый на 8 разрядов.

Машинная реализация алгоритма представлена ниже. Для выделения групп разрядов используется специальная константа 0…01…1, вводится второй регистр константы РгК1 и вспомогательный рабочий регистр РгР.

1. РгН := (РгА), если Z = 1, идти к 50, иначе к 2; проверка на 0 операнда А
2. РгН := (РгВ), если Z = 1, идти к 50, иначе к 3; проверка на 0 операнда В
3. РгН := !(РгК); выделение модуля А (пп. 3, 4, 5)
4. РгН := (РгА) & (РгН)
5. РгМА := (РгН)

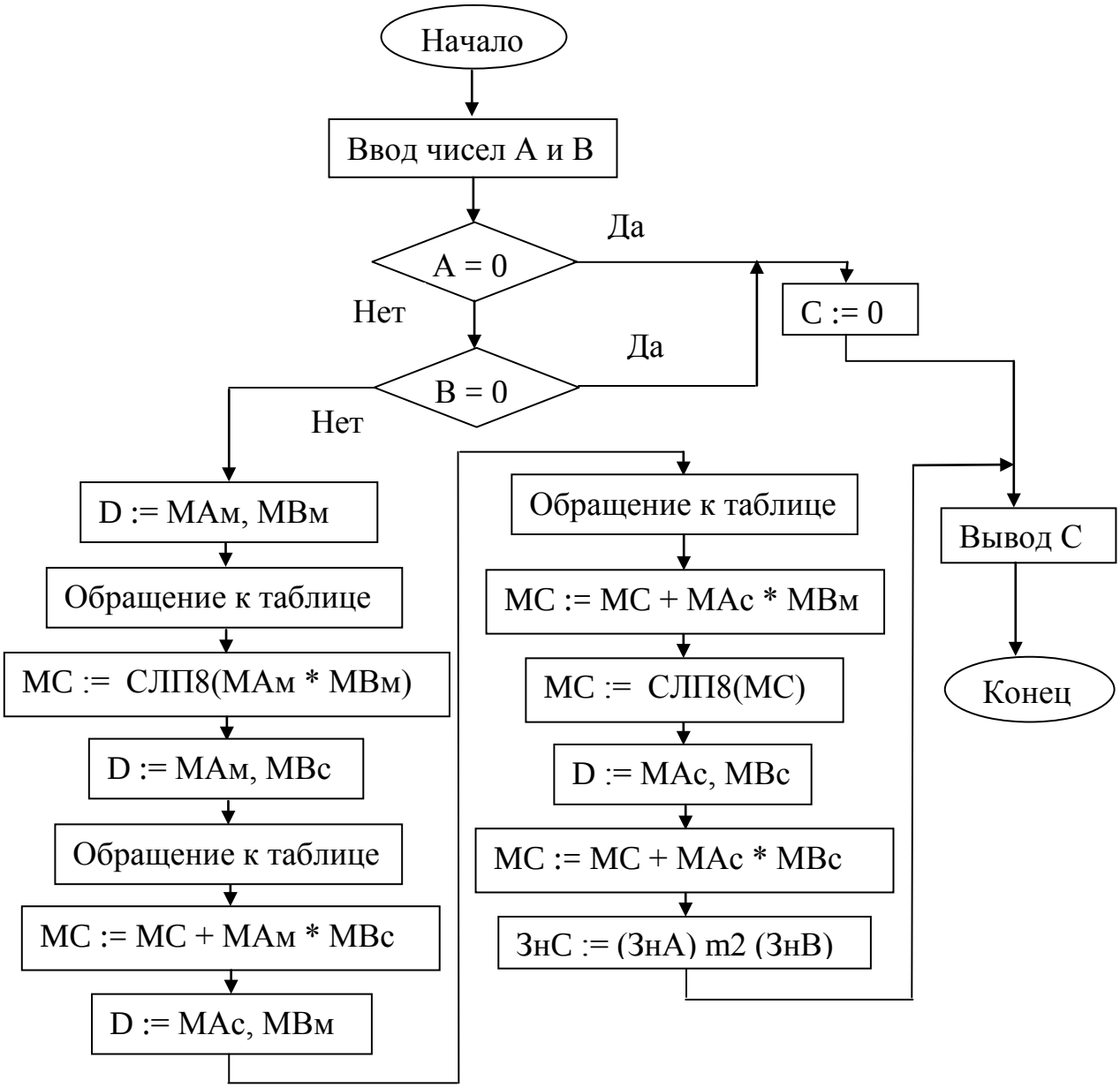

Рис. 22

6. РгН := !(РгК); выделение модуля B (пп. 6, 7, 8)
7. РгН := (РгВ) & (РгН)
8. РгМВ := (РгН)
9. РгН := (РгК1); формирование МВм (пп. 9, 10, 11)
10. РгН := (РгН) & (РгМВ)
11. РгР := (РгН)
12. РгН := (РгК1); формирование МАм
13. РгН := (РгН) & (РгМА)
14. РгН := СЛЛ8 (РгН); смещение МАм
15. РгН := (РгН) \/ (РгР); объединение МАм и МВм
16. РгН := (РгТ); обращение к таблице
17. РгН := СЛП8 (РгН); округление

18. РгС := (РгН); первое частичное произведение
19. РгН := (!РгК1); формирование МАс
20. РгН := (РгН) & (РгМА)
21. РгН := (РгН) \/ (РгР); объединение МАс и МВм
22. РгН := (РгТ); обращение к таблице
23. РгН := (РгН) + (РгС); второе частичное произведение
24. РгС := (РгН)
25. РгН := (РгК1); формирование МАм
26. РгН := (РгН) & (РгМА)
27. РгР := (РгН)
28. РгН := (!РгК1); формирование МВс
29. РгН := (РгН) & (РгМВ)
30. РгН := (РгН) \/ (РгР); объединение МВс и МАм
31. РгН := (РгТ); обращение к таблице
32. РгН := (РгН) + (РгС); третье частичное произведение
33. РгН := СЛП8 (РгН); округление
34. РгС := (РгН)
35. РгН := (!РгК1); формирование МАс
36. РгН := (РгН) & (РгМА)
37. РгР := (РгН)
38. РгН := (!РгК1); формирование МВс
39. РгН := (РгН) & (РгМВ)
40. РгН := СЛП8 (РгН)
41. РгН := (РгН) \/ (РгР); объединение МАс и МВс
42. РгН := (РгТ); обращение к таблице
43. РгН := (РгН) + (РгС); модуль полного произведения
44. РгС := (РгН)
45. РгН := (РгА)
46. РгН := (РгН) m2 (РгВ); формирование знака произведения
47. РгН := (РгН) & (РгК)
48. РгН := (РгН) \/ (РгС); формирование произведения
49. РгС := (РгН). Конец.
50. РгН := (РгК) m2 (РгК), идти к 49; обнуление результата

Выполнение операции деления дробных чисел с фиксированной запятой через умножение на обратное значение делителя

В основу метода положена следующая формула:

$$C = A / B = A * (1 / B),$$

где 1 / В – обратное значение делителя.

Получение обратного значения делителя для ускорения выполнения операции предполагается получать с помощью прямого табличного преобразования.

Особенностью данного метода является предварительное масштабирование обратного значения делителя, так как это число будет целым (в общем случае – смешанным), а операционное устройство реализует умножение дробных чисел, меньших единицы. Соответственно, исходная формула преобразуется к виду

$$C = A * ((1 / B) * 2\uparrow(-n))\ 2\uparrow n,$$

где n – разрядность операндов A и B.

Схематично данное преобразование над форматами поясняется на рис. 23, где n-разрядный операнд B после табличного преобразования перемещается относительно формата операционного устройства на n разрядов влево и выходит за пределы разрядной сетки. Для его возврата в разрядную сетку осуществляется умножение на коэффициент $2\uparrow(-n)$ и выполняется умножение, схематично отображённое на рис. 24.

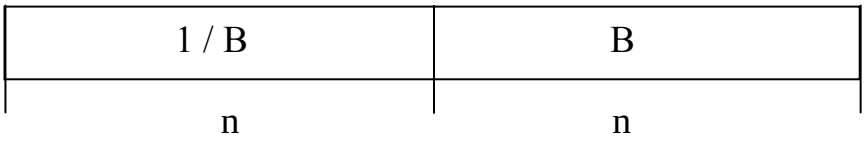

Рис. 23

Полученная величина D является двухформатной и требует для своего размещения в операционном устройстве двух регистров Dст и Dмл. Искомая величина C получается обратным масштабированием D через умножение на коэффициент $2\uparrow n$, что вызовет сдвиг Dмл влево и переход его в позицию Dст. Физически этот сдвиг можно не выполнять, а осуществлять выборку значения C из регистра Dмл.

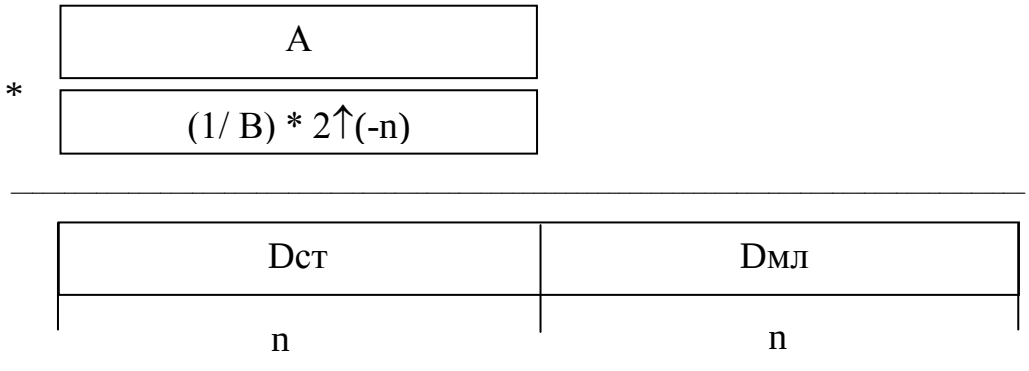

Рис. 24

К недостатку рассмотренного метода относится погрешность, возникающая из-за отбрасывания дробной части или округления до целого числа при получении 1/B. Минимальная погрешность будет получена при целочисленном результате преобразования B в 1/B либо при использовании операнда 1/B в виде смешанного числа. Однако в последнем случае процесс умножения существенно усложнится и ожидаемое ускорение операции деления не будет достигнуто.

Процедура умножения на 1/B также может быть ускорена одним из ранее рассмотренных методов, например методом с применением таблицы квадратов в соответствии с формулой

$$(A + F)\uparrow 2 = A\uparrow 2 + 2A*F + F\uparrow 2,$$

где $F = (1/B)*2\uparrow(-n)$, откуда получаем

$$C = A*(1/B) = ((A + F)\uparrow 2 - A\uparrow 2 - F\uparrow 2) * 2\uparrow(n - 1).$$

Причём преобразование B в $F = (1/B)*2\uparrow(-n)$ может быть выполнено в таблице масштабирования.

В целом потребуются две таблицы: масштабирования ТМ и возведения в квадрат ТК.

Машинная реализация процедуры деления будет иметь следующий вид.

1. РгН := (РгВА), если Z = 1, идти к 56, иначе к 2; проверка на 0 операнда B
2. РгН := (РгА), если Z = 1, идти к 55, иначе к 3; проверка на 0 операнда A
3. РгН := !(РгК); выделение модуля A (пп. 3, 4, 5)
3. РгН := (РгА) & (РгН)
4. РгМА := (РгН)
5. РгН := !(РгК); выделение модуля B (пп. 6, 7, 8)
6. РгН := (РгВ) & (РгН)
7. РгМВ := (РгН)
8. РгН := (РгМА) – (РгН); сравнение Am и Bm
10. РгН := СЛЛ (РгН), если ВСЛ = 1, идти к 11, иначе к 52; анализ знака
11. РгН := (РгМВ)
12. РгАТМ := (РгН); обращение к таблице ТМ (пп. 12,13)
13. РгН := (РгТМ)
14. РгМВ := (РгН)
15. РгН := (РгМА)
16. РгН := (РгМВ) + (РгН)
17. РгАТК := (РгН); обращение к таблице ТК (пп. 17, 18)
18. РгТР, РгТ := [(РгАТК)]
19. РгН := (РгТ); сохранение квадрата суммы (пп. 19 – 22)
20. РгС := (РгН)
21. РгН := (РгТР)
22. РгСР := (РгН)

23. РгН := (РгМА)

24. РгАТК := (РгН); обращение к таблице ТК (пп. 24, 25)

25. РгТР, РгТ := [(РгАТК)]

26. РгН := (РгТ)

27. РгН := (РгС) – (РгН), если заём равен 1, идти к 28, иначе к 34; разность квадрата суммы и $МА^2$

28. РгС := (РгН)

29. РгН := (РгСР) – 1

30. РгСР := (РгН)

31. РгН := (РгТР)

32. РгН := (РгСР) – (РгН)

33. РгСР := (РгН), идти к 35

34. РгС := (РгН), идти к 31

35. РгН := (РгМВ)

36. РгАТК := (РгН); обращение к таблице ТК (пп. 36, 37)

37. РгТР, РгТ := [(РгАТК)]

38. РгН := (РгТ)

39. РгН := (РгС) – (РгН), если заём равен 1, идти к 40, иначе к 46; формирование удвоенного модуля частного (пп. 39–46)

40. РгС := (РгН)

41. РгН := (РгСР) – 1

42. РгСР := (РгН)

43. РгН := (РгТР)

44. РгН := (РгСР) – (РгН)

45. РгСР := (РгН), идти к 47

46. РгС := (РгН), идти к 43

47. РгН := (РгС); правый сдвиг модуля частного

48. РгН := СЛП(РгН)

49. РгС := (РгН); сохранение модуля частного

50. РгН := (РгА); формирование знака частного (пп. 50–52)

51. РгН := (РгН) m2 (РгВ)

52. РгН := (РгН) & (РгК)

53. РгН := (РгС) \/ (РгН); формирование результата

54. РгС := (РгН), идти к 58

55. РгН := (РгК) m2 (РгК), идти к 54; обнуление результата

56. РгН := (РгК); формирование сообщения об ошибке

57. РгО := (РгН). Конец.

58. РгН := (РгК) m2 (РгК), идти к 57; формирование сообщения о правильном результате.

Информационные потоки в ЭВМ

Рассмотрим структуры ЭВМ на базе двух классических архитектур: Принстонской и Гарвардской. Исторически Принстонская архитектура является более ранней и предусматривает обобщённую основную память для хранения данных и программ. На рис. 25 представлена схема распределения информационных потоков в ЭВМ Принстонской архитектуры.

Наличие однопортовой обобщённой основной памяти предусматривает разделение времени её работы между циклами выборки данных и команд из-за общих физических шин данных-команд, шин адреса и физической среды накопителя. Схема (рис. 25) соответствует простейшей ЭВМ с одним операционным устройством (ОУ), основной памятью (ОП), подсистемой ввода-вывода (ПВВ), устройством командного управления (УУ) и подсистемой периферийных устройств (ПУ).

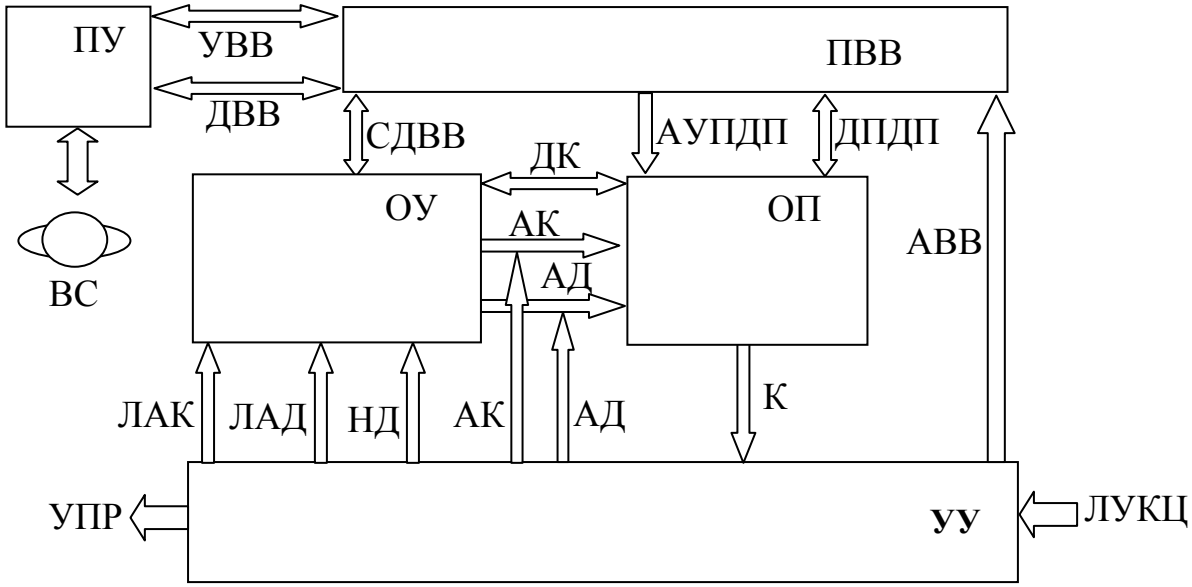

Рис. 25

Адресное пространство ОП разделяется на зоны адресации к командам и зоны адресации к данным. Под данными понимаются любые информационные массивы, не являющиеся командами выполняемых программ. Разделение адресного пространства приводит к физическому разделению ячеек памяти накопителя для хранения соответствующей информации. Несмотря на разделение адресных зон накопителя, однопортовая память не позволяет осуществлять одновременное чтение данных и команд, что приводит к снижению производительности ЭВМ.

Программа работы ЭВМ состоит из функционально связанных команд и выполнение программы заключается в выполнении соответствующей последо-

вательности команд. Выборку очередной команды из ОП, запуск её исполнения и формирование адреса выборки следующей команды осуществляет УУ.

В начале командного цикла УУ посылает к ОП адрес команды (АК) текущего цикла. Выбранная из ОП команда (К) поступает в УУ, где формируются соответствующие сигналы управления ЭВМ (УПР), поступающие на все устройства и определяющие их действия в данном командном цикле. Кроме этого, УУ в зависимости от типа команды может выделить из неё непосредственные данные (НД), участвующие в исполнении команды, исполнительный адрес выборки следующей команды (АК) или логический адрес (ЛАК), на основе которого в ОУ формируется исполнительный адрес АК, исполнительный адрес данных (АД) или логический адрес данных (ЛАД).

Преобразование логических адресов в исполнительные осуществляется в ОУ, откуда исполнительные адреса поступают к ОП для адресации к командам или данным. Причём если команды только считываются из ОП, то данные можно как считывать по адресу обращения, так и записывать.

После завершения операции по данной команде в устройствах ЭВМ кроме результатов могут формироваться оповестительные сигналы – логические условия командного цикла (ЛУКЦ), влияющие на формирование в УУ адреса следующей команды.

Для общения с внешней средой (ВС) в ЭВМ имеется подсистема ПУ. Взаимодействие с ПУ реализуется через подсистему ввода-вывода, обеспечивающую обмен между УУ, ОУ, ОП и ПУ. От УУ к ПВВ поступают адреса ввода-вывода (АВВ), определяющие порты ПУ, через которые производится обмен информацией с внешней средой. Данные ввода-вывода (ДВВ) пересылаются между адресуемым портом ПУ и ПВВ в соответствии с сигналами управления вводом-выводом (УВВ).

ПВВ выполняет преобразование разнотипных данных от ПУ к стандартным данным ввода-вывода (СДВВ) ЭВМ при загрузке их в ОУ и обратное преобразование при выводе информации на ПУ. В общем случае вводимая информация (данные и команды – ДК) через ОУ заносится в ОП в область размещения программ или данных. В современных системах для ускорения процессов ввода-вывода ПВВ имеют каналы прямого доступа к памяти (ПДП), позволяющие осуществлять обмен информацией ОП – ПВВ, минуя ОУ. В этом случае адресная и управляющая информация (АУПДП) поступает от ПВВ к ОП, а данные прямого доступа (ДПДП) между ПВВ и ОП могут передаваться как в том, так и в другом направлении. Применение ПДП освобождает ОУ от вспомогательных операций при вводе-выводе и существенно повышает производительность ЭВМ.

Однако в целом разделение времени работы ОП в Принстонской архитектуре усложняет реализацию современных способов дальнейшего повышения

производительности ЭВМ, например внедрение конвейерных и параллельных вычислений.

Гарвардская архитектура устраняет основной недостаток Принстонской архитектуры за счёт физически разделённых памяти данных (ПД) и памяти команд (ПК), а также физически разделённых внутренних интерфейсов данных и команд (рис. 26).

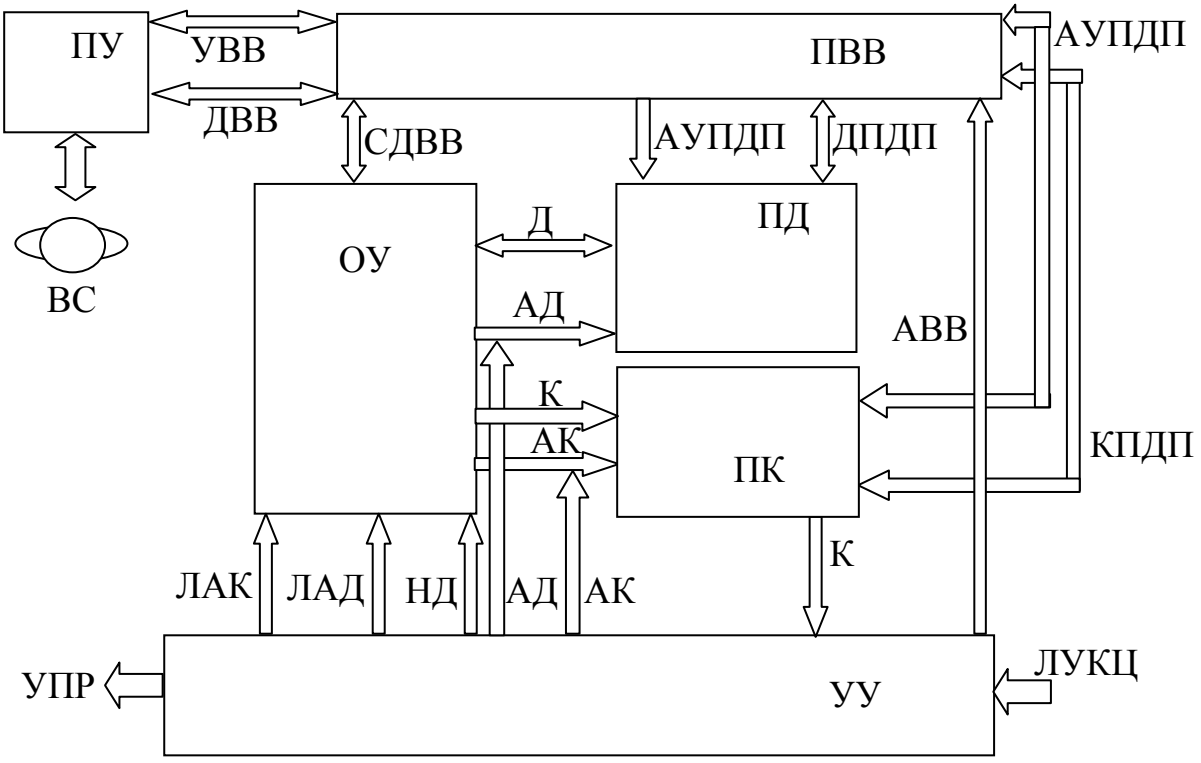

Рис. 26

Соответственно в ЭВМ появляются выделенные информационные потоки данных (Д), команд (К), адресов данных (АД) и адресов команд (АК), данных ПДП (ДПДП), загрузки команд при ПДП (КПДП) и загрузки команд (К) через ОУ. Независимая работа ПД и ПК позволяет осуществлять их параллельное функционирование и способствует конвейерным вычислениям.

Организация конвейерных вычислений

Конвейер позволяет выполнять квазипараллельные вычисления при больших информационных потоках данных и однотипным преобразованием над данными потока на базе единственного ОУ. Классическим примером конвейера является конвейер на автомобильных заводах Генри Форда.

В основу конвейера положено предложение разбиения крупной операции на более мелкие составные операции, каждая из которых выполняется в пределах определённого времени, называемого шагом конвейера.

Каждый очередной элемент потока объектов обработки загружается на первое операционное место конвейерной ленты с задержкой на шаг конвейера, а предыдущие объекты обработки соответственно перемещаются по ленте на очередное операционное место. Когда конвейерная лента будет полностью заполнена, то через шаг конвейера с неё начнёт сходить готовая продукция. Причём каждый очередной объект готовой продукции будет сходить с ленты на каждом шаге конвейера.

На рис. 27 представлена схема конвейера, состоящая из последовательно соединённых функциональных устройств ФУi (i = 1 … n), каждое из которых выполняет элементарную операцию ОПi в течение шага конвейера ТК, где Xi(t +ТК*(i–1)) – i-й элемент входного потока, Yi (t+ТК*(n+i–1)) – i-й элемент выходного потока. Первый элемент Yi выходного потока появится на выходе конвейера через ТК*n – время полной загрузки конвейера. Это время равно технологической задержке конвейера после запуска. Далее каждый элемент выходного потока будет выходить через время ТК. Среднее время обработки одного элемента входного потока составляет Tср = (ТК*(n+N–1))/N и при N>>n стремится к ТК, создавая эффект обработки каждого элемента входного потока за время ТК.

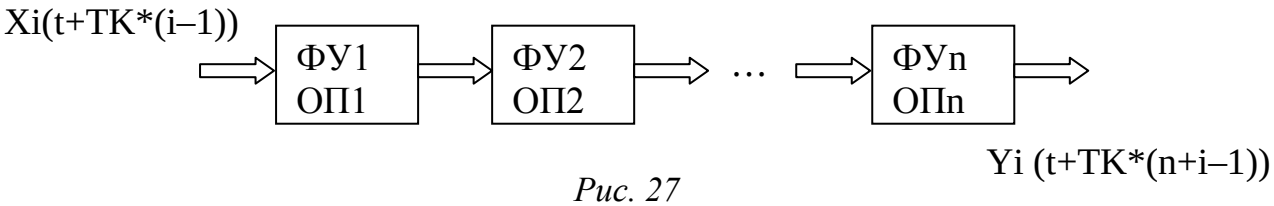

Рис. 27

В качестве примера на рис. 28 приведена временная диаграмма работы конвейерной ЭВМ, в которой выполнение машинной операции разбито на пять этапов:

IF – выборка команды из ПК;

ID – дешифрация команды в УУ и выборка операндов из регистрового файла;

EX – выполнение операции или вычисление адреса обращения к ПД;

MEM – обращение к ПД для записи или чтения операндов;

WB – запись результата в регистровый файл.

По оси ординат откладываются номера К потока команд, по оси абсцисс – время. Очередная команда поступает на вход конвейера с шагом ТК. Конвейер пятиуровневый, поэтому для его загрузки требуется время 5*ТК. Каждый очередной результат R появляется с задержкой ТК относительно предыдущего после полной загрузки конвейера. Для пяти команд среднее время выполнения текущей команды составит Tср = ((5*ТК) + (5*ТК))/ 5 = 2*ТК. Для ста команд в потоке Tср = ((5*ТК) + (100*ТК))/100 = 1,05*ТК. Таким образом, при увеличении числа команд в потоке среднее время выполнения команды стремится к ТК.

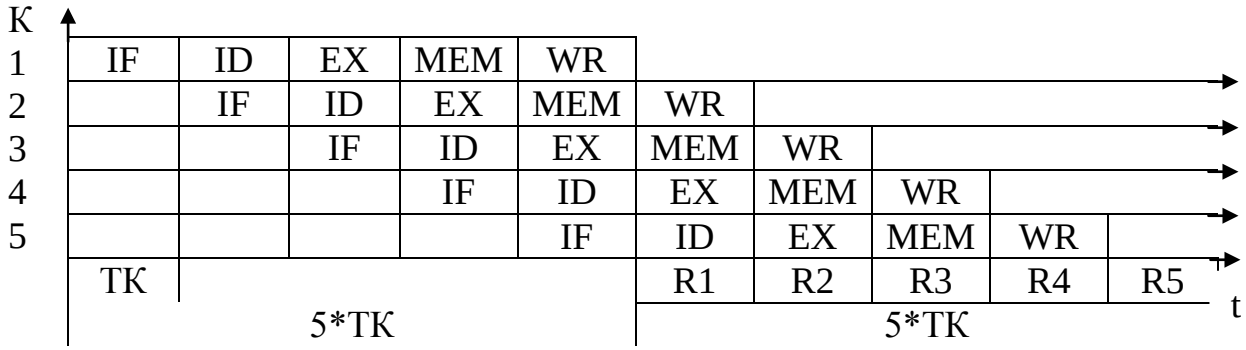

Рис. 28

По диаграмме видно, что в каждом такте работы конвейера из ПК считывается очередная команда, а после загрузки конвейера в каждом такте одновременно осуществляется обращение к ПК, ПД и к регистровому файлу ОУ. Реализовать подобную временную диаграмму в ЭВМ с Гарвардской архитектурой достаточно просто из-за физически разделённых ПК и ПД, в то время как в ЭВМ с Принстонской архитектурой без дополнительных модификаций – весьма затруднительно.

Библиографический список

1. Гуменюк, А. С. Прикладная теория информации [Электронный ресурс] : учеб. пособие / А. С. Гуменюк, Н. Н. Поздниченко. – Электрон. текст. дан. (1,25 Мб). – Омск : Изд-во ОмГТУ, 2015. – 1 электрон. опт. диск.

2. Гуменюк, А. С. Теория информации и кодирования [Электронный ресурс] : учеб. пособие / А. С. Гуменюк, Н. Н. Поздниченко. Электрон. текст. дан. (1,83 Мб). – Омск : Изд-во ОмГТУ, 2015. – 1 электрон. опт. диск.

3. Гуменюк, А. С. Информатика / А. С. Гуменюк, И. В. Потапов. – Омск : Изд-во ОмГМА, 2012.– 175 с.

4. Потапов, В. И. Компьютерная арифметика и алгоритмическое моделирование арифметических операций : учеб. пособие / В. И. Потапов, О. П. Шафеева. – Омск : Изд-во ОмГТУ, 2005. – 96 с.

Приложения

Приложение 1

Пример 1

Пусть A = 1.0010110, B = 0.0110111. Числа имеют разные знаки.

1. РгА := 1.0010110

2. РгВ := 0.0110111

3. РгК := 0.1111111

4. РгН := 1. 0010110

5. РгН := 0.0110111

6. РгН := 1. 0010110 & 0.1111111 = 0.0010110

7. РгМА := 0.0010110

8. РгН := 0.0110111 & 0.1111111 = 0.0110111

9. РгМВ := 0.0110111

10. РгН := 1.0000000

11. РгК := 1.0000000

12. РгН := 1.0010110 & 1.0000000 = 1.0000000, Z = 0, идти к 13

13. РгН := 0.0110111 & 1.0000000 = 0.0000000, Z = 1, идти к 18

18. РгН := 0.0010110 – 0.0110111 = 1.1011111, Z = 0, идти к 19.

19. РгС := 1.1011111

20. РгН := 10111110, ВСЛ = 1, идти к 21

21. РгН := 0.0110111 – 0.0010110 = 0.0100001

22. РгС := 0.0100001

23. РгН := 0.0110111 & 1.0000000 = 0.0000000

24. РгН := 0.0100001 ∨ 0.0000000 = 0.0100001

25. РгС := 0.0100001

26. РгН := 0.0000000

27. РгО := 0.0000000

33. ШВ := 0.0000000

34. ШВ := 0.0100001. Конец.

Пример 2

Пусть A = 0.1011110, B = 0.0110010. Числа имеют равные знаки.

1. РгА := 0.1011110

2. РгВ := 0.0110010

3. РгК := 0.1111111

4. РгН := 0.1011110, Z := 0, идти к 5

5. РгН := 0.0110010, Z := 0, идти к 6

6. РгН := 0.1011110 & 0.1111111 = 0.1011110

7. РгМА := 0.1011110

8. РгН := 0.0110010 & 0.1111111 = 0.0110010

9. РгМВ := 0.0110010

10. РгН := 1.0000000

11. РгК := 1.0000000

12. РгН := 0.1011110 & 1.0000000 = 0.0000000, Z = 1, идти к 30

30. РгН := 0.0110010 & 1.0000000= 0.0000000, Z = 1, идти к 14

14. РгН := 0.1011110 + 0.0110010 = 1.0010000

15. РгС := 1.0010000

16. РгН := 0.0100000, ВСЛ = 1, идти к 31

31. РгН := 1.0000000, идти к 35

35. РгО := 1.0000000

36. ШВ := 1.0000000. Переполнение. Конец.

Приложение 2

Пример 1

Пусть A = 1.1101010, B = 0.0110111.

1. РгА := 1.1101010
2. РгВ := 0.0110111
3. РгН := 1.1101010, Z = 0, идти к 4
4. РгН := 0.0110111, Z = 0, идти к 5
5. РгН := 1.1101010 + 0.0110111 = 0.0100001
6. РгС := 0.0100001
7. РгН := 1.1101010 m2 0.0110111 = 1.1011101
8. РгН := 1.0111010, ВСЛ = 1, идти к 11
11. РгС := 0.0100001
12. РгН := 0.0000000
13. РгО := 0.0000000
17. ШВ := 0.0000000
18. ШВ := 0.0100001. Конец.

Пример 2

Пусть A = 0.1011110, B = 0.0110010.

1. РгА := 1.1101010
2. РгВ := 0.0110111
3. РгН := 0.1011110, Z = 0, идти к 4
4. РгН := 0.0110010, Z = 0, идти к 5
5. РгН := 0.1011110 + 0.0110010 = 1.0010000
6. РгС := 1.0010000
7. РгН := 0.1011110 m2 0.0110010 = 0.1101100
8. РгН := 1.1011000, ВЛС = 0, идти к 9
9. РгН := 0.1011110 m2 1.0010000 = 1.1001110
10. РгН := 1.0011100, ВЛС = 1, идти к 14
14. РгН := 0.1111111, идти к 19
19. РгО := 0.1111111
20. ШВ := 0.1111111. Переполнение. Конец.

Приложение 3

Пример 1

Пусть имеются два нормализованных числа с различными положительными порядками: Am = 0.1011101 pA = 0.0000111 и Bm = 0.1101100 pB = 0.0000100

1. РгН := 0.0000111 – 0.0000100 = 0.0000011, Z = 0, идти к 2
2. РгБ := 0.0000011
3. РгН := 0.0000110, ВСЛ = 0, идти к 4
4. РгН := 0.0000010
5. РгБ := 0.0000010
6. РгН := 0.0110110
7. РгМВ := 0.0110110
8. РгН := 0.0000010, Z = 0, идти к 4

4. РгН := 0.0000001
5. РгБ := 0.0000001
6. РгН := 0.0011011
7. РгМВ := 0.0011011
8. РгН := 0.0000001, Z = 0, идти к 4

4. РгН := 0.0000000
5. РгБ := 0.0000000
6. РгН := 0.0001101
7. РгМВ := 0.0001101
8. РгН := 0.0000000, Z = 1, идти к 9
9. РгН := 0.0000111
10. РгПС := 0.0000111. Конец.

Пример 2

В данном примере сохранены прежние мантиссы, порядки отрицательные: pA = 1.1111001, pB = 1.1111011 (коды дополнительные).

1. РгН := 1.1111001 – 1.1111011 = 1.1111110, Z = 0, идти к 2
2. РгБ := 1.1111110

3. РгН := 1.1111100, ВСЛ = 1, идти к 11

11. РгН := 1.1111111

12. РгБ := 1.1111111

13. РгН := 0.0101110

14. РгМА := 0.0101110

15. РгН := 1.1111111, Z = 0, идти к 11

11. РгН := 0.0000000

12. РгБ := 0.0000000

13. РгН := 0.0010111

14. РгМА := 0.0010111

15. РгН := 0.0000000, Z = 1, идти к 16

16. РгН := 1.1111011, идти к 10

10. РгПС := 1.1111011. Конец.

Пусть A = 1,10110101 и B = 0,01110010.

1. РгА := 01,10110101
2. РгВ := 00,01110010
3. РгК := 0100000000
4. РгН := 01,10110101, Z = 0, идти к 5
5. РгН := 00,01110011, Z = 0, идти к 6
6. РгН := 00,01110011 m2 01,10110101 = 01,11000110
7. РгН := 01,11000110 & 0100000000 = 0100000000
8. РгС := 01,00000000
9. РгП := 0000000000
10. РгСЦ := 00,0000100
11. РгН := 0011111111
12. РгН := 00,10110101
13. РгМА := 00,10110101
14. РгН := 01,01101010
15. Рг2МА := 01,01101010
16. РгН := 11,01001011
17. РгДМА := 11,01001011
18. РгН := 0011111111
19. РгН := 00,01110010
20. РгМВ := 00,01110010
21. РгМС := 00,00000000
22. РгН := 0000000000, идти к 23
23. РгН := 00,01110010, идти к 39
39. РгН := 00,00000000
40. РгН := 00,00000000 + 01,01101010 = 01,01101010, идти к 37
37. РгП := 0000000000
38. РгМС := 01,01101010, идти к 25
25. РгН := 00,01011010

26. РгМС := 00,01011010

27. РгН := 00,00011100

28. РгМВ :=00,00011100

29. РгН := 00,00000011

30. РгСЦ := 00,00000011, идти к 22

22. РгН := 00,00000000, идти к 23

23. РгН := 00,00011100, идти к 25

25. РгН := 00,00010110

26. РгМС := 00,00010110

27. РгН := 00,00000111

28. РгМВ := 00,00000111

29. РгН := 00,00000010, идти к 30

30. РгСЦ := 00,00000010, идти к 22

22. РгН := 00,00000000, идти к 23

23. РгН := 00,00000111, идти к 41

41. РгН := 00,00010110

42. РгН := 00,00010110 + 11,01001011= 11, 01100001

43. РгП := 0000000001

44. РгМС := 11, 01100001, идти к 25

25. РгН := 11,11011000

26. РгМС := 11,11011000

27. РгН := 00,00000001

28. РгМВ := 00,00000001

29. РгН := 00,00000001, идти к 30

30. РгСЦ := 00,00000001, идти к 22

22. РгН := 00,00000001, идти к 24

24. РгН := 00,00000001, идти к 39

39. РгН := 11,11011000

40. РгН := 11,11011000 + 01,01101010 = 01,01000010, идти к 37

37. РгП := 0000000000

38. РгМС := 01,01000010, идти к 25

25. РгН := 00,01010000

26. РгМС := 00,01010000

27. РгН := 00,00000000

28. РгМВ := 00,00000000

29. РгН := 00,00000000, идти к 31

31. РгН := 01,00000000

32. РгН := 01,00000000 \/ 00,01010000 = 01,01010000

33. РгС := 01,01010000

34. ШВ := 1,01010000. Конец.