ОСНОВНЫЕ ПРИНЦИПЫ ПОСТРОЕНИЯ И СОСТАВ ПРОЦЕССОРОВ

- **Принцип дискретности – основной принцип работы микропроцессорных систем (МПС)**

- **Цифровые устройства (ЦУ)** выполняют различные операции над объектами информации в виде цифровых сигналов (ЦС).
- Для представления ЦС служат **биты, байты, кодовые слова, двойные кодовые слова,** особенность которых состоит в том, что:
- => для их построения используется **простейший алфавит**, состоящий из двух букв, которые обозначаются символами «0» и «1». Во многих случаях эти символы отождествляются с арабскими цифрами, и тогда кодовое слово (КС) представляет собой число в двоичной системе счисления.

Для представления КС в виде электрических сигналов наибольшее распространение получил потенциальный способ, при котором одному из символов, например **логическому 0, соответствует низкий уровень напряжения, а другому — высокий.**

Операция – это любое действие, связанное с обработкой информации и приводящее к изменению выходного слова по отношению к входному.

Для выполнения операции на вход(ы) устройства подается цифровой сигнал в течение некоторого фиксированного промежутка времени. На время выполнения операции состояние входов устройства остается неизменным.

- На рисунке следующего слайда графически представлены ЦСи способы ввода- вывода ЦС в ЦУ и из него:
- - параллельный ввод и параллельный вывод;
- - последовательный ввод и параллельный вывод;
- - параллельный ввод и последовательный вывод.

ройством, имеющим в общем случае n входов и m выходов.

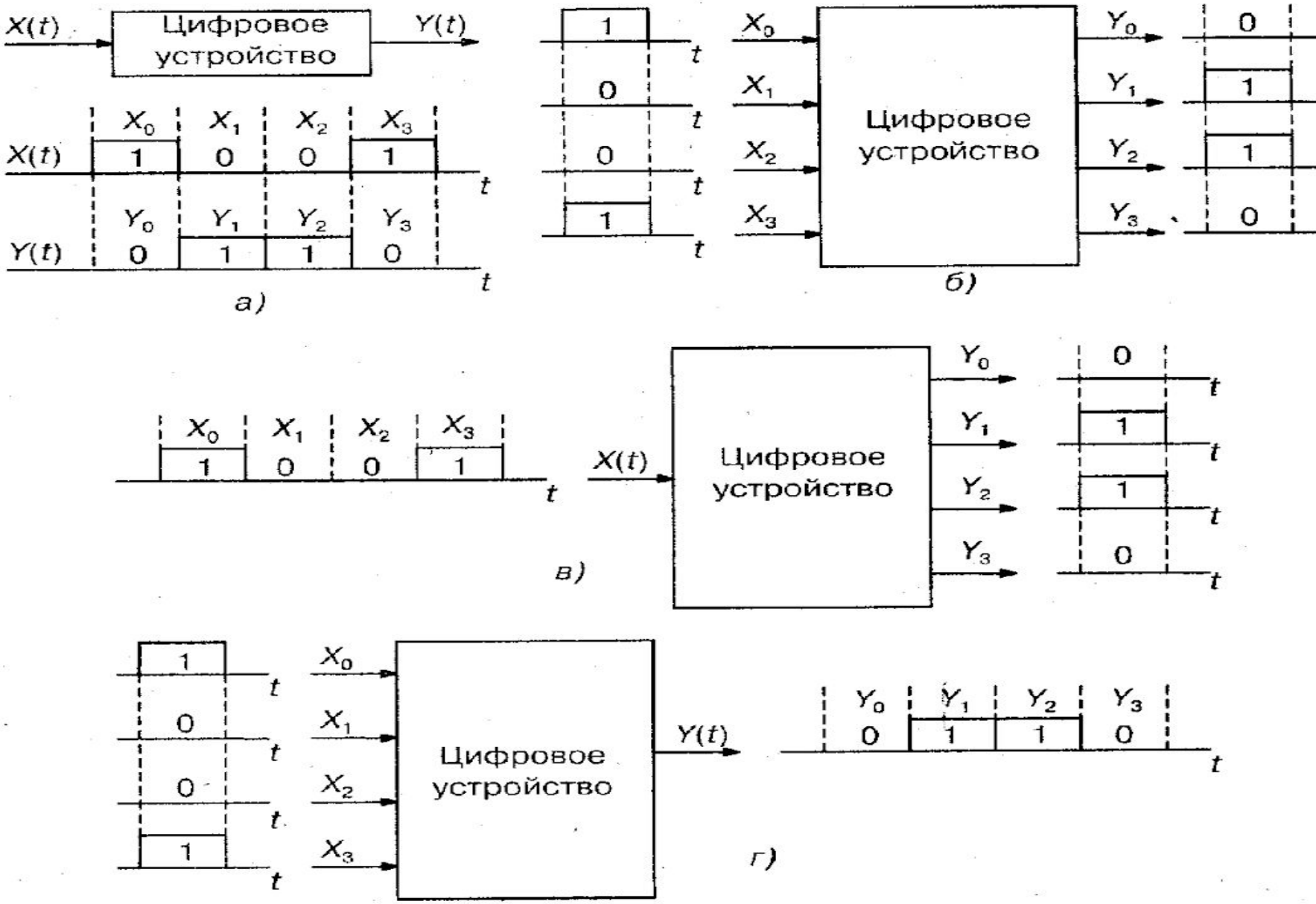

Рис. 1.4.1. Способы обработки кодовых слов:

а — с последовательным вводом/выводом; *б* — с параллельным вводом/выводом; *в* — с последовате

Принцип функционирования цифровых сметем (ЦС).

- Работа ЦС связана с выполнением некоторой последовательности микрокоманд (МКК), в совокупности составляющих команду (К), а совокупность команд , в свою очередь, составляющих программу (Пр).
- Для выполнения той или иной К (функции, операции, преобразования) с помощью **генератор тактовых импульсов (ГТИ)** отводится определенное время.
- ГТИ формирует управляющие сигналы, предназначенных для активизации **начала** выполнения отдельных К;
- После активизации начала К происходит преобразование всех входных КС (логических нулей и единиц) в требуемые выходные КС;
- Выходные КС отправляются на хранение в память ЦС, если они потребуются в дальнейшем, или во внешнее устройство для выполнения определенных действий.

- В отличие от аналоговых систем, в работу которых заложен **принцип непрерывности**, работа микроконтроллеров (МК), относящихся к цифровым устройствам (ЦУ), основана на принципе дискретности.
- **Принципе дискретности** реализуется путем выделения временных интервалов (тактирования). Для исполнения одной микрокоманды (МКК) выделяется квант времени (временной интервал).

- Рассмотрим последовательность исполнения отдельной МКК:
- => *в начальный момент времени* микропроцессорная система (МПС) находится в состоянии $A = \{a_i\}$, где a_i — состояния отдельных устройств системы, которые память хранит в виде определенной совокупности дискретных сигналов (0 и 1);
- => *поступившая на вход системы МКК* в виде совокупности сигналов $\{X_т\}$, или М-разрядного входного кода $X_„..,X_т...X_,,$ переводит систему в другое состояние *B = {bj},* где b_j, — состояния отдельных устройств системы;

- => *новое состояние* МПС *B* запоминает и хранит память;
- => *по завершении выполнения всей предписанной последовательности МКК* из памяти считывается требуемая совокупность хранящихся сигналов в виде *N*-разрядного выходного кода $Y_N \ldots Y_1$.
- Выходные сигналы, как отмечалось выше, подаются в память или на исполнительное устройство.
- Упорядоченная во времени совокупность МКК образует *команду,* или *инструкцию, а* совокупность команд и данных — *программу.* В процессе выполнения программы МПС может работать в различных режимах.

Выполнение команды (К).

- Время, затрачиваемое на выполнение К, называется *командным циклом (КЦ).* КЦ разбивается на машинные циклы (МЦ).
- **Машинным циклом называется промежуток времени между двумя последовательными обращениями процессора к ОЗУ или внешнему устройству по системной шине.**
- Длительность МЦ может составлять 3-5 и более системных *тактов* (периодов синхросигналов шины), которые требуются:
- - для установки требуемого адреса;
- - выдачи сигналов, определяющих вид цикла — чтение или запись;
- - получения сигнала готовности к обмену (от памяти или внешних устройств) и собственно передачи данных или команд.

- Код команды (КК) представляет собой многоразрядное двоичное число, в котором можно выделить две части:
- => *код операции (КО),* задающий вид операции, выполняемой данной командой;
- => *код адресации операндов (КАО),* задающий адреса источников операндов, над которыми производится заданная операция, и адреса приемников для операнда-результата операции.
- КК хранятся в оперативном запоминающем устройстве (ОЗУ).

- Процесс выполнения команды (К) можно разбить на отдельные этапы:
- => в начале первого МЦ по адресу, который задается содержимым программного счетчика PC (Program Counter), из ОЗУ считывается код команды (КК), подлежащей выполнению. КК поступает в регистр команд МК;
- => при дешифрации КК(кода операции и кода адресации операндов) определяется:

  - вид выполняемой операции и адреса необходимых операндов;

  - необходимое число МЦ для выполнения

- Если для выполнения К не требуется обращения к внешним ЗУ дл считывание операндов из памяти или запись в память результатов операции, **то такая команда выполняется за один цикл. (Это характерно для МК).**
- В противном случае требуется выполнение дополнительных циклов чтения (ввода) или записи (вывода). В зависимости от разрядности обрабатываемых операндов и разрядности используемой системной шины число циклов, необходимых для выполнения команд, может составлять от 1 до 10-15;

- После считывания кода текущей К содержимое программного счетчика (РС) автоматически увеличивается на 1 и более (при условных переходах). Тем самым обеспечивается последовательная выборка К в процессе выполнения Пр. При выборке очередной К содержимое РС поступает на шину адреса (ША), обеспечивая считывание из ОЗУ следующей команды выполняемой Пр. При реализации безусловных или условных переходов (ветвлений) или других изменений последовательности выполнения команд происходит загрузка в РС нового содержимого. В результате этого осуществляется переход к другой ветви Пр.

В соответствии с выполняемой операцией устройство управления формирует необходимые сигналы для реализации МЦ и требуемую последовательность микрокоманд в каждом

РЕГИСТРОВАЯ МОДЕЛЬ МИКРОКОНТРОЛЛЕРА

- *Функционирование* процессора (П)можно представить как процедуры изменения состояния регистров (регистровые пересылки) путем чтения-записи их содержимого. В результате таких пересылок обеспечивается адресация и выборка команд и операндов из основной памяти, хранение и пересылка результатов, изменение последовательности команд и режимов функционирования процессора в соответствии с поступлением нового содержимого в служебные регистры, а также все другие процедуры, реализующие процесс обработки информации согласно заданным условиям.

- При составлении программ весьма важно знать, какие из регистров МК являются программно-доступными регистрами, в которых можно хранить **подлежащие обработке данные - операнды** и **управляющие сигналы - команды**. Совокупность программно-доступных регистров образуют регистровую *модель* МК.
- В регистровой модели можно выделит две группы регистров:
- *=> регистры общего назначения* (РОН), предназначенные для хранения операндов (в том числе адресных кодов). Эта группа регистров образует внутреннюю память МК;
- -» *служебные регистры,* предназначенные для управления исполняемой программой, обеспечения требуемого режима работы МК, организации обращения к памяти и выполнения других функций.

- Состав и количество служебных регистров определяется архитектурой МК.
- К *основным* служебным регистрам следует отнести:
- - программный счетчик PC (Program Counter) или указатель команд IP (Instruction Pointer);
- - регистр состояния SR (Status Register), или флагов (EFLAGS);
  - регистры управления режимом работы процессора CR (Control Register);
- - регистры, реализующие сегментную и страничную организацию памяти;
- - регистры, обеспечивающие отладку программ и тестирование процессора.
  Кроме того, различные модели МК содержат ряд других
  специализированных регистров

• Основные классификационные признаки архитектур.

- По форматам используемых команд (инструкций) можно выделить:
- => *CISC-архитектуру,* которая относится к компьютерам с *набором сложных команд* (Complex Instruction Set Computer). Она реализована во многих типах микропроцессоров (например, Pentium), выполняющих большой набор разноформатных команд с использованием многочисленных способов адресации. Система команд процессоров с CISC-архитектурой может содержать более 200 команд разной степени сложности (от 1 до 15 байт) и использовать десятки различных способов адресации, что позволяет программисту реализовать наиболее эффективные алгоритмы решения различных задач. *Недостаток* CISC-архитектуры — дальнейшее ее развитие связано с существенным усложнением структуры МП, повышением его стоимости и увеличением временных затрат на исполнение программы;

- *=» RISC-архитектуру,* которая относится к компьютерам с *сокращенным набором команд* (Reduced Instruction Set Computer). Появление RISC-архитектуры продиктовано тем, что многие CISC-команды и способы адресации используются достаточно редко. Современные RISC-процессоры реализуют около 100 команд, имеющих фиксированный формат длиной 4 байта, и используют небольшое число наиболее простых способов адресации (регистровую, индексную и некоторые другие). Для сокращения количества обращений к внешней оперативной памяти RISC-процессоры содержат десятки-сотни регистров общего назначения (РОН), тогда как в CISC-процессорах всего 8-16 регистров.

- Обращение к внешней памяти в RISC-процессорах используется только в операциях загрузки данных в РОН или пересылки результатов из РОН в память. В результате существенно упрощается структура микропроцессора, сокращаются его размеры и стоимость, значительно повышается производительность. Благодаря указанным достоинствам во многих современных CISC-процессорах (последние модели Pentium и K7) используется RISC-ядро. При этом сложные CISC-команды предварительно преобразуются в последова тельность простых RISC-операций и быстро выполняются RISC-ядром;

- По способу организации выборки команд и данных различают два *вида* архитектур:
- ·=> *Принстонская архитектура,* или архитектура Фон-Неймана, особенностью которой является использование: *общей оперативной памяти* для хранения *программ, данных и организации стека,* что позволяет оперативно и эффективно перераспределять ее объем в зависимости от решаемых задач в каждом конкретном случае применения микропроцессора;

  - -» *общей системной шины,* по которой в процессор поступают *команды* и *данные,* а в оперативную память записываются *результаты,* что значительно упрощает отладку, тестирование и текущий контроль функциониро
  вания системы, повышает ее надежность. **Однако использование общей шины для передачи команд и данных ограничивает**

- => *Гарвардская архитектура,* особенностью которой является физическое разделение памяти на память команд (программ) и память данных. **Это обстоятельство вызвано постоянно возрастающими требованиями к производительности МПС систем.**
- **Память команд и память данных соединяются с процессором отдельными шинами.** Благодаря разделению потоков команд и данных, а также совмещению операций их выборки (и записи результатов обработки) **обеспечивается более высокая производительность**, чем при использовании Принстонской архитектуры.
- *Недостатки* Гарвардской архитектуры: *усложнение конструкций* из-за использования отдельных шин для команд и данных; *фиксированный объем* памяти для команд и данных; *увеличение общего объема памяти* из-за невозможности ее оптимального перераспределения между командами и данными.

- Гарвардская архитектура получила широкое применение в МК— специализированных МПС для управления различными объектами, а также во внутренней структуре современных высокопроизводительных МПС в кэш-памяти с раздельным хранением команд и данных. В то же время во внешней структуре большинства МПС систем реализуются принципы Принстонской архитектуры.
- Отметим, что архитектура МПС тесно связана с его структурой. Реализация тех или иных архитектурных особенностей требует введения в структуру МПС соответствующих устройств и обеспечения механизмов их совместного

ПРОЦЕССОР КАК СОВОКУПНОСТЬ ОПЕРАЦИОННОГО И УПРАВЛЯЮЩЕГО АВТОМАТОВ (ОА) И (УА)

- Процессор состоит из двух автоматов:
  - операционного (ОА) и
  - управляющего (УА).
- **ОА** можно представить в виде трех функциональных модулей (рис. 15.1.2):
- **- памяти;**
- **- комбинационной схемы, реализующей микрооперации и**
- **- комбинационной схемы, вычисляющей значения логических условий.**

- **УА генерирует последовательность управляющих сигналов из множества Y, предписанную МКПР и соответствующую значениям ЛУ X.**
- При выполнении пакета МКПР на его входы последовательно подаются коды МКОП, которые соответствуют той или иной МКПР. На входы П могут поступать внешние сигналы ЛУ, а с выходов сниматься сигналы для управления внешними устройствами.

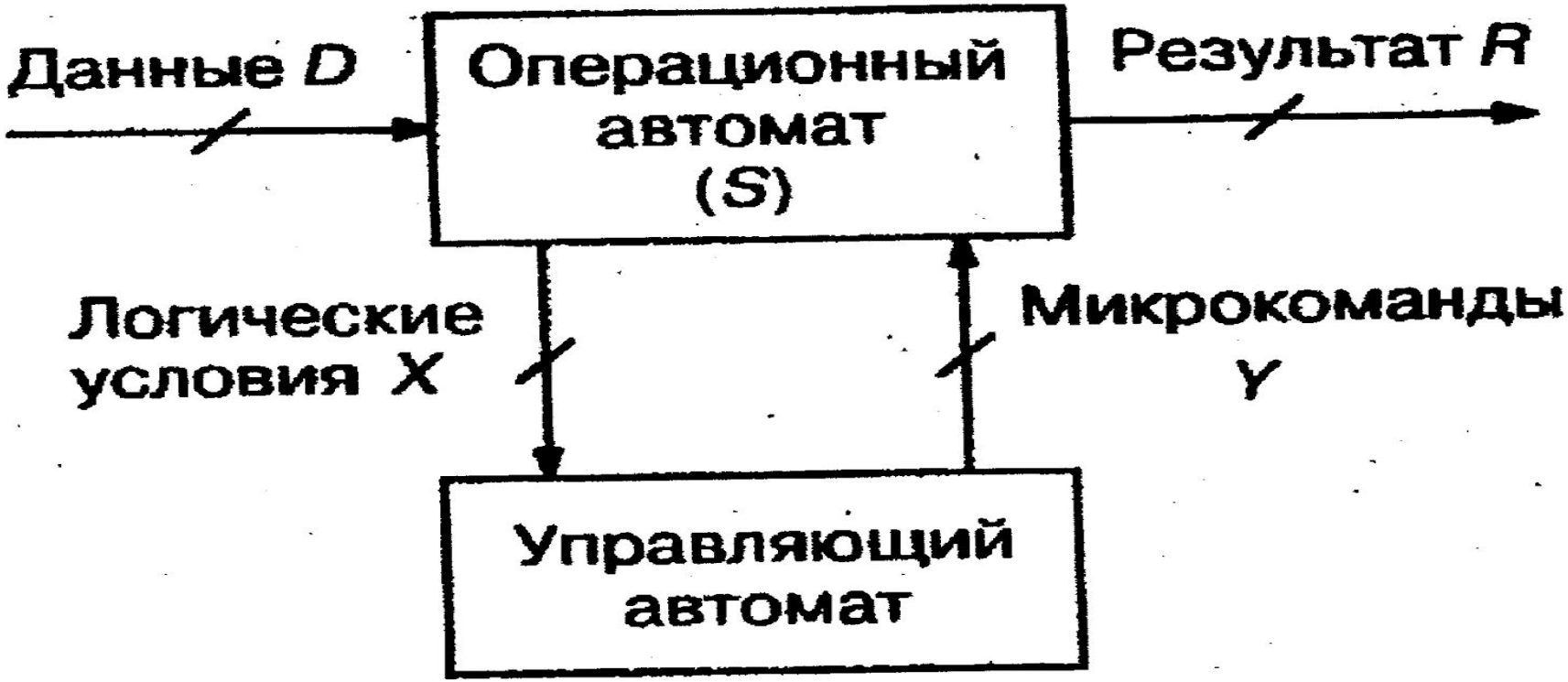

Рис. 15.1.1. Структура процессора

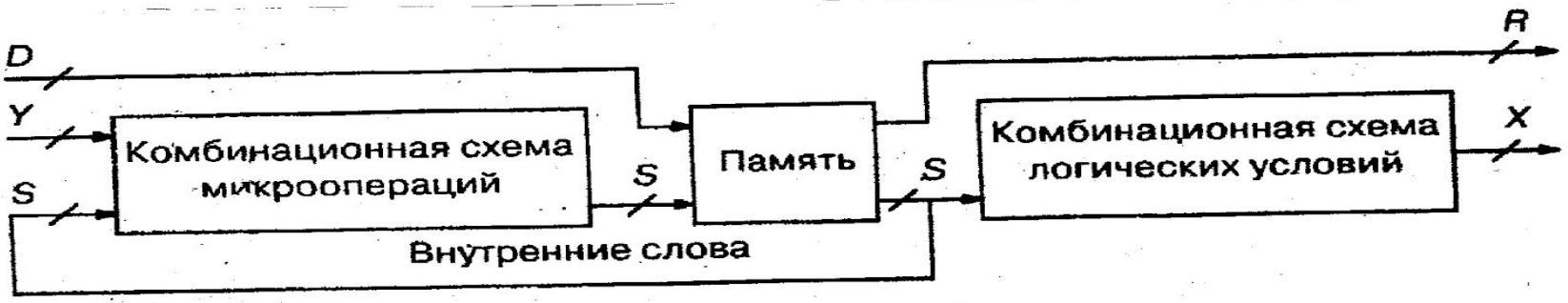

Рис. 15.1.2. Структура операционного автомата

ОПЕРАЦИОННЫЙ АВТОМАТ ДЛЯ УМНОЖЕНИЯ ДВОИЧНЫХ ЧИСЕЛ

Принцип построения операционного автомата.

- Для обоснования принципа построения операционного автомата воспользуемся правилом умножения двоичных чисел, при котором частичные произведения формируются, начиная с младших разрядов множителя Б (рис. 15.2.1).

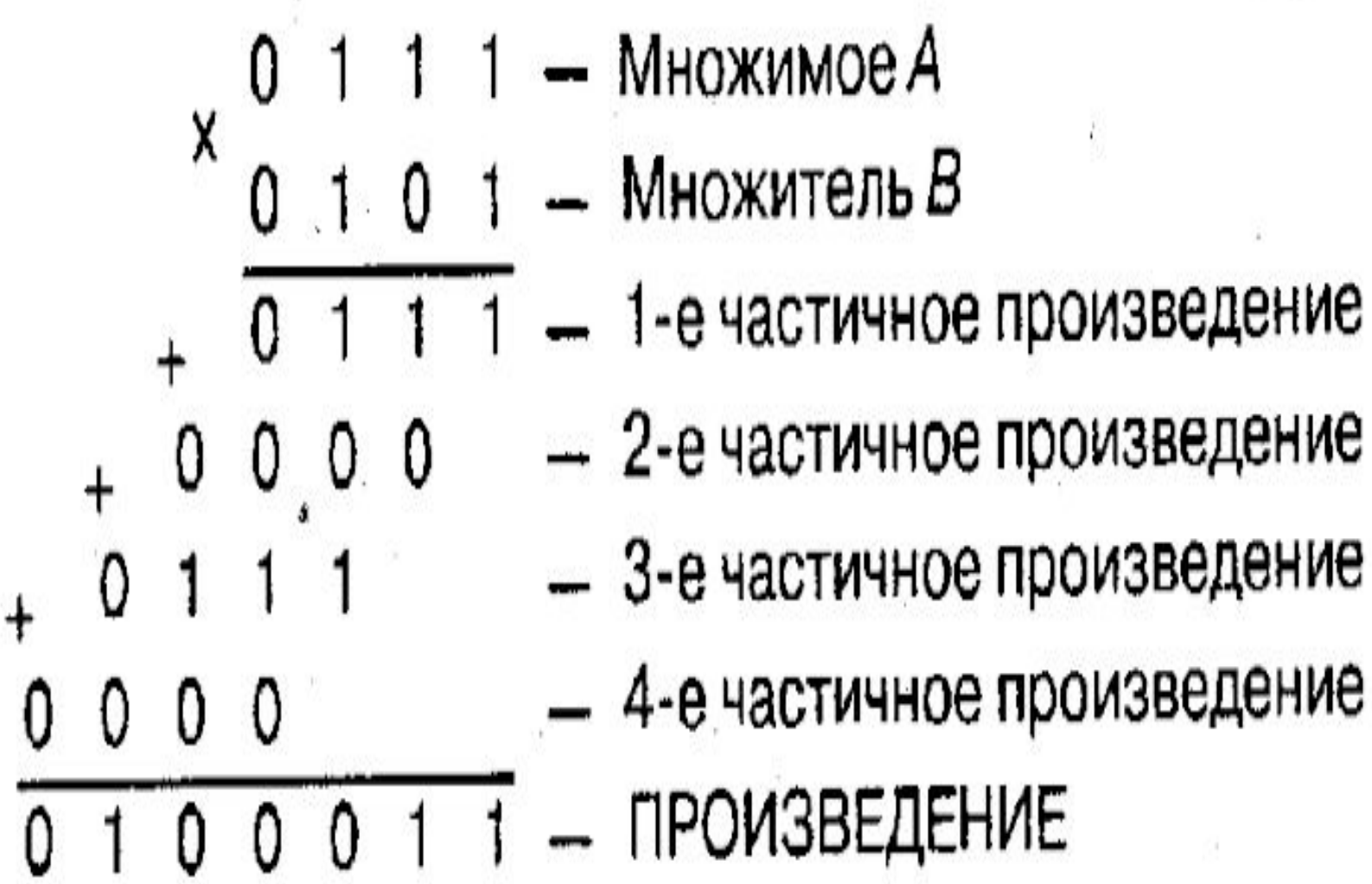

Рис. 15.2.1. Правило умножения двоичных чисел начиная с младших разрядов множителя

- Выбор аппаратных средств для реализации операции ум ножения двоичных чисел. Прежде всего необходимо иметь в виду, что суммирование частичных произведений должно осуществляться *последовательно во времени* с помощью сумматоров, предназначенных для сложения двух операндов, так как сложность схемного решения сумматоров возрастает как с ростом числа операндов, так и с увеличением их разрядности. Как видно из рис. 15.2.1, особенность умножения двоичных чисел состоит в том, что частичные произведения могут принимать лишь *два значения:* значение множимого *А* либо нуля. Значение частичного произведения определяется значением текущего разряда множителя Б. Если частичное произведение равно нулю, то микрооперацию сложения можно не выполнять. Таким образом, множимое *А* используется как частичное произведение и его будем постоянно хранить в регистре RG,. Для аппаратного определения значения текущего разряда множителя Б необходимо располагать сдвигающим регистром RG_2.

- В исходном состоянии регистр RG_2 загружен множителем Б, причем выходной сигнал должен соответствовать самому младшему разряду множителя. Чтобы выявить значение следующего разряда множителя *B,* после каждой микрооперации сложения частичного произведения необходимо производить сдвиг содержимого RG_2 в сторону самого младшего разряда. Для хранения частичных сумм частичных произведений необходимо располагать третьим регистром RG_3. В исходном состоянии RG_3 должен быть загружен нулями. В процессе умножения осуществляется сложение содержимого регистра RG_3 с частичным произведением *A.* Частичная сумма помещается в RG_3, после чего выполняется сдвиг в сторону младших разрядов, так как в 2 раза увеличивается вес каждого очередного разряда множителя Б (рис. 15.2.1).
- Алгоритм умножения двоичных чисел. На рис. 15.2.2 показан про цесс умножения с использованием *трех регистров* и *сумматора.* Множимое *A* = 0111 постоянно находится в регистре RG_t. В исходном состоянии в регистр RG_3 помещен нуль 0000, а в регистр RG_2 — множитель Б = 0101. Нуль в старшем разряде операндов *A* и Б свидетельствует о том, что перемножаются *положительные числа.* В процессе умножения в регистре RG_3 размещаются частичные суммы частичных произведений и произведение. Анализируется младший разряд регистра RG_2 ($MPRG_2$), который отождествляется с логическим условием X,.

- В исходном состоянии регистр RG_2 загружен множителем Б, причем выходной сигнал должен соответствовать самому младшему разряду множителя. Чтобы выявить значение следующего разряда множителя *B,* после каждой микрооперации сложения частичного произведения необходимо производить сдвиг содержимого RG_2 в сторону самого младшего разряда. Для хранения частичных сумм частичных произведений необходимо располагать третьим регистром RG_3. В исходном состоянии RG_3 должен быть загружен нулями. В процессе умножения осуществляется сложение содержимого регистра RG_3 с частичным произведением *A.* Частичная сумма помещается в RG_3, после чего выполняется сдвиг в сторону младших разрядов, так как в 2 раза увеличивается вес каждого очередного разряда множителя Б (рис. 15.2.1).

- Алгоритм умножения двоичных чисел. На рис. 15.2.2 показан процесс умножения с использованием *трех регистров* и *сумматора.* Множимое *А* = 0111 постоянно находится в регистре RG_t. В исходном состоянии в регистр RG_3 помещен нуль 0000, а в регистр RG_2 — множитель Б = 0101. Нуль в старшем разряде операндов *А* и Б свидетельствует о том, что перемножаются *положительные числа.* В процессе умножения в регистре RG_3 размещаются частичные суммы частичных произведений и произведение. Анализируется младший разряд регистра RG_2 ($MPRG_2$), который отождествляется с логическим условием X,.

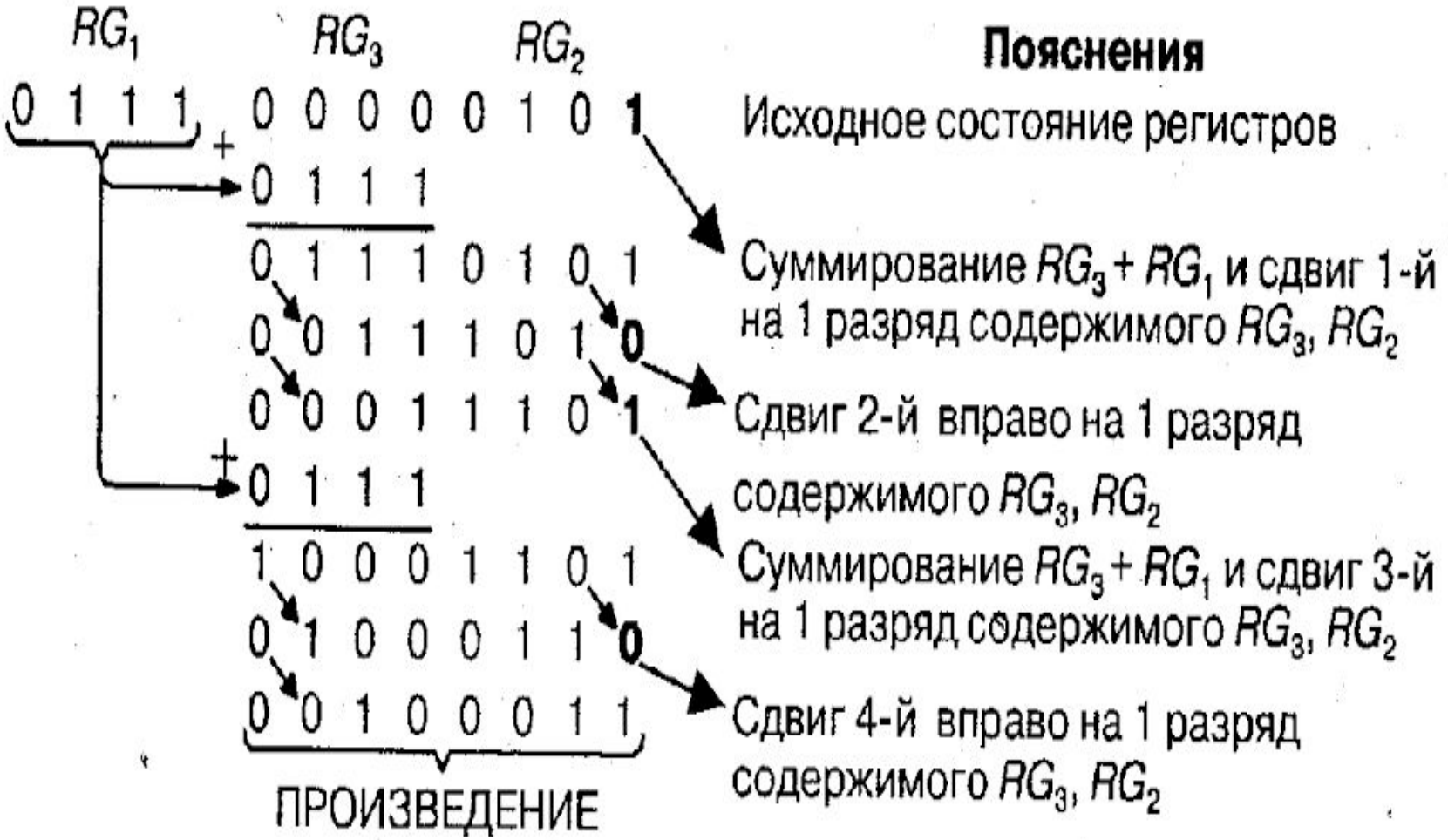

Рис. 15.2.2. Иллюстрация алгоритма умножения двоичных чисел с помощью трех регистров и сумматора

- Если МР $ЯС_2$ =X, = 1, то выполняется микрооперация сложения содержимого регистров $ЯС_3$ и ЯС,, и результат помещается в $ЯС_3$. Эта микрооперация может быть записана в следующем виде: $ЯС_3$:=$ЯС_3$ +ЯС,. Затем осуществляется микрооперация сдвига вправо на один разряд (Я?) содержимого составного регистра, образованного из регистров $ЯС_3$ и $ЯС_2$: RG_3, RG_2'-= $R1(RG_3,RG_2)$. Если же МР $ЯС_2$=X, =0, то выполнятся только сдвиг содержимого составного регистра.
- Из рис. 15.2.2 видно, что процесс носит циклический характер. Число циклов л равно числу разрядов множителя (в примере η = 4), поэтому при схемной реализации для автоматической фиксации завершения операции умножения целесообразно использовать вычитающий *счетчик СТ* числа повторений цикла. В исходном состоянии счетчик загружается числом $\pi = 4$ (100). По завершении каждого цикла содержимое счетчика уменьшается на единицу. После четвертого цикла счетчик будет пуст (000). Если к выходам счетчика подключить логический элемент **ЗИЛИ-НЕ** и его выходной сигнал принять в качестве логического условия X_2, то X_2 = 1 будет свидетельствовать о завершении четвертого цикла или об окончании операции умножения.
- **Структурная схема операционного автомата.** Схема автомата для умножения двоичных чисел приведена на рис. 15.2.3.

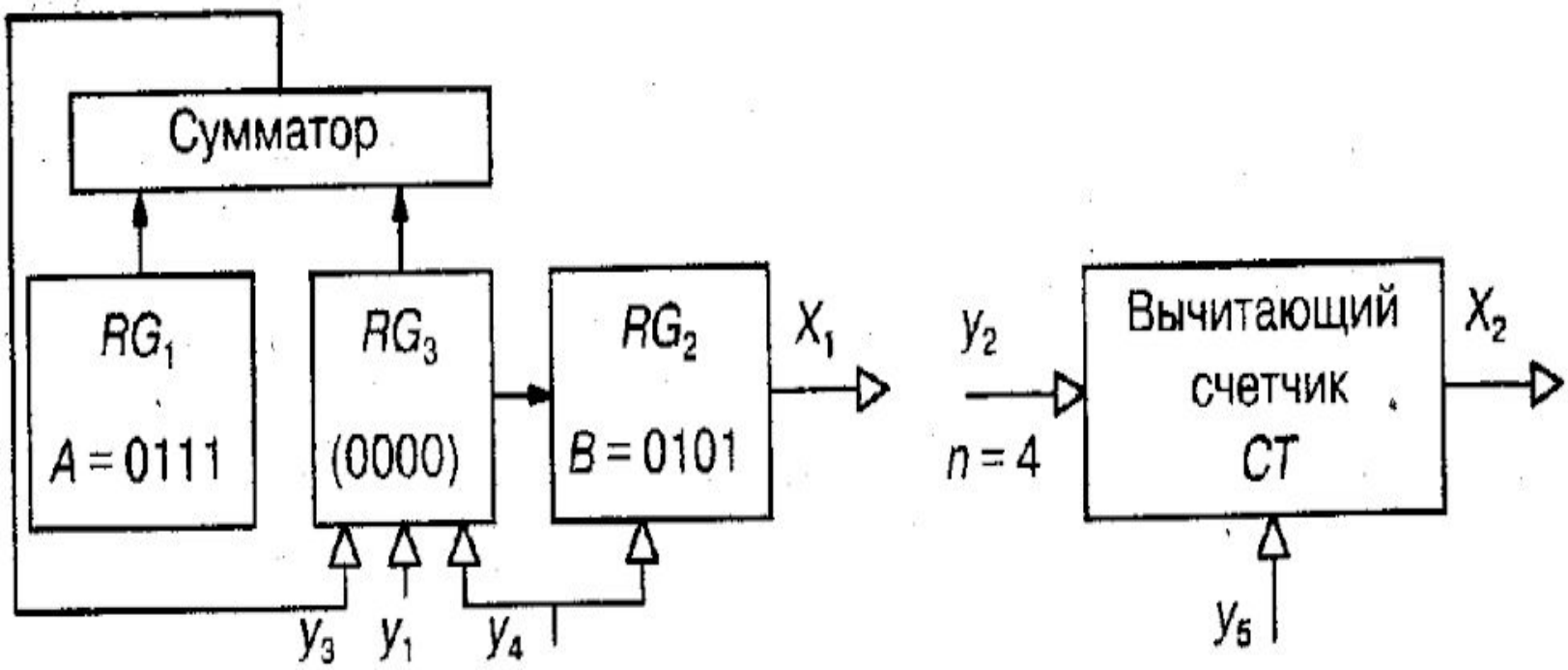

Рис. 15.2.3. Структурная схема операционного аппарата для умножения двоичных чисел:

y_1 — микрооперация загрузки регистра RG_3 нулем; y_2 — микрооперация загрузки счетчика CT числом $n = 4$ (код 0100); y_3 — микрооперация сложения содержимого регистров RG_3, RG_1 и загрузки результата в регистр RG_3; y_4 — микрооперация сдвига на один разряд вправо содержимого регистров RG_3, RG_2; y_5 — микрооперация уменьшения на единицу содержимого счетчика CT

- Составной регистр из RG_3 и RG_2 образован путем соединения выхода триггера младшего разряда RG_3 с входом триггера старшего разряда RG_2. *С* помощью управляющего сигнала у, регистр RG_3 устанавливается в нулевое состояние, а сигнала $у_2$ — в счетчик *СТ* вводится число $\eta = 4$. По команде $у_3$ результаты суммирования содержимого RG_3 и ЯС, с выхода сумматора поступают в RG_3. Для сдвига содержимого составного регистра RG_3, RG_2 используется управляющий сигнал $у_4$, а для уменьшения показаний счетчика *СТ* на единицу — сигнал $у_5$. Предполагается, что операнды *А* и *В* уже загружены в регистры. Управляющие сигналы у,-$у_5$ будем отождествлять с микрооперациями.
- В операционном автомате формируются следующие признаки: => X, — значение младшего разряда RG_2. Значение X, = 1 свидетельствует о том, что младший разряд регистра RG_2 равен 1. В этом случае выполняются описанные выше микрооперация сложения, а затем сдвига. При X, = 0 младший разряд регистра RG_2 равен 0 и выполняется только микрооперация сдвига; => X_2 — результат проверки на нуль содержимого *СТ*. Значение X_2 = 1 свидетельствует о том, что счетчик пуст (СГ = 000). В этом случае операция умножения завершается. При X_2 = 0 начинается новый цикл операции умножения. Приведем в условной записи список микроопераций, выполняемых в узлах операционного автомата, и список формируемых признаков:
- **Список микроопераций Список признаков**
- у,—$ЯС_3$: = 0; $у_2$ — СГ: = п; X, = 1 — МРЯС, := 1;
- Уз — $ЯС_3$: = $ЯС_3$ + ЯС,; X_2 = 1 — *СТ: = 0.*
- $у_4$ — RG_3,RG_2: = R1 ($ЯС_3$,$ЯС_2$); $у_5$ — *СТ:* = *СТ*- 1.

УПРАВЛЯЮЩИЙ АВТОМАТ СО СХЕМНОЙ ЛОГИКОЙ

- Рассмотрим основные этапы построения такого УА.
- **Построение граф-схемы алгоритма операции умножения.** Граф-схема алгоритма представляет собой связанный граф со следующими основными типами вершин: *начальная, конечная, операторная* и *условная.* При составлении графа руководствуются следующими правилами:
- => граф-схема алгоритма должна содержать одну начальную, одну конечную и ко нечное число операторных и условных вершин;
- => входы и выходы различных вершин соединяются дугами, направленными от вы хода к входу. При этом выход каждой вершины соединяется только с одним входом;
- => в каждой операторной вершине записывается *микрокоманда,* представляющая собой *набор микроопераций,* выполняемых на одном временном интервале (такте);
- => в каждой условной вершине записывается одно из логических условий; => между любой вершиной и конечной вершиной должен существовать, по крайней мере, один путь.
- На основании приведенного выше списка микроопераций сформируем набор микрокоманд:
- => объединим микрооперации у, , y_2 (загрузки регистра $ЯС_3$ и счетчика *СТ)* в общую микрокоманду У, : = у,,y_2, так как они могут быть выполнены одновременно (на одном тактовом периоде); => выделим в отдельную микрокоманду микрооперацию сложения содержимого
- регистров RG_3 + ЯС, с загрузкой суммы в $ЯС_3$ — $/_2$: = y_3;
- => объединим микрооперации y_4, y_5 (сдвига вправо на один разряд содержимого регистровой пары RG_3, RG_2 и уменьшения на единицу содержимого счетчика *СТ) в* микрокоманду $У_3$: = $У_4$,y_5.

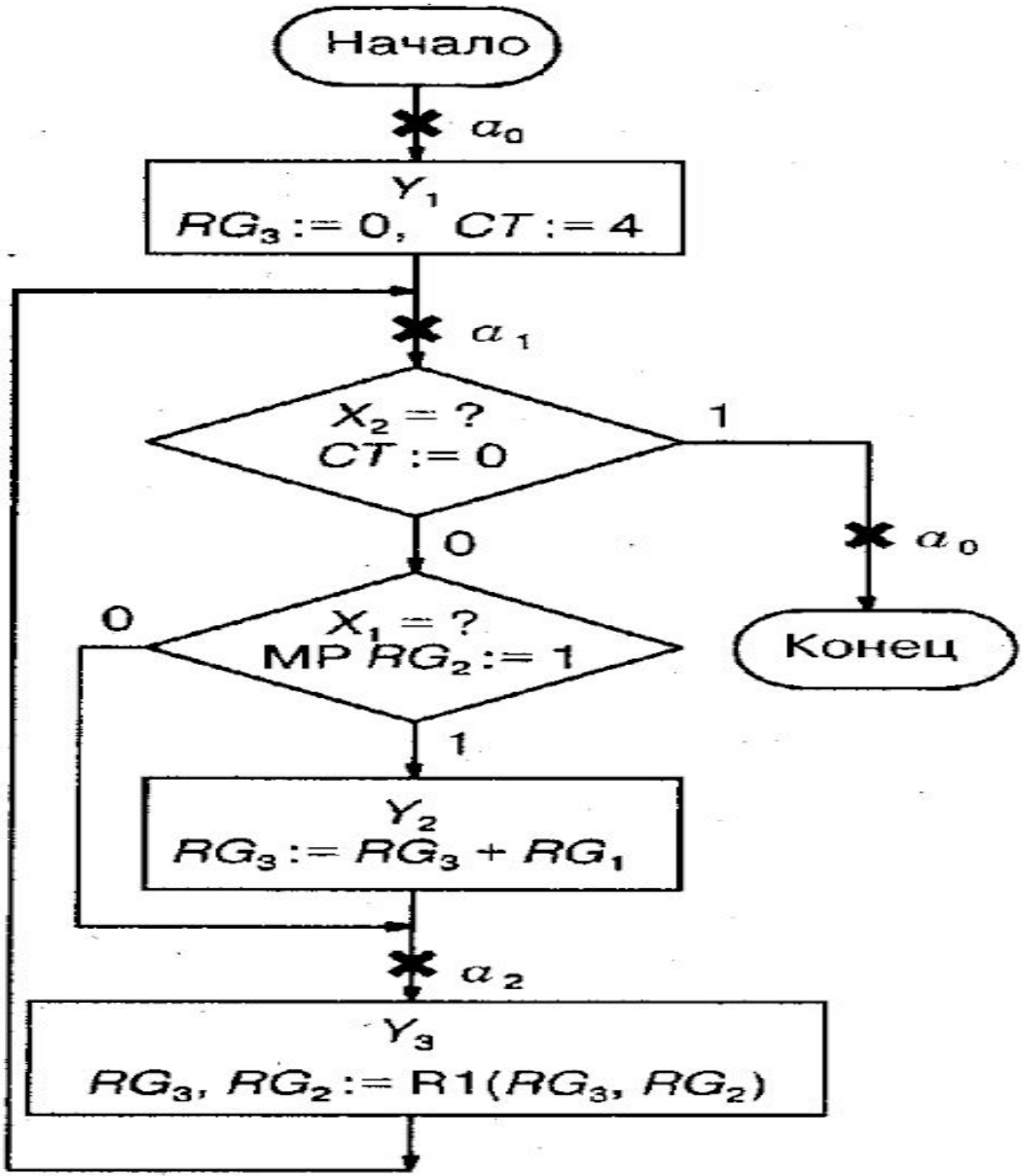

Рис. 15.3.1. Граф-схема алгоритма операции умножения двоичных чисел

Построим граф переходов для авто

- Для операции умножения можно построить несколько граф-схем, каждая из которых определяет свою структуру управляю щего автомата и последовательность его функционирования.
- На рис. 15.3.1 приведен один из возможных вариантов граф-схемы алгоритма операции умножения двоичных чисел.
- **Построение графа переходов управляющего автомата.** Управляющий автомат можно строить как аппарат *Мили* или как аппарат *Мура,* которые описываются функцией перехода и функцией выходов
- *A(t*
- *= F(A(t), X(t)); Y(t) = f(A(t), X(t))* для аппарата Мили,
- *A(t+*
- *= F\A(t),* X(f)); V(f) = *f(A(t))* -для аппарата Мура, ·
- где *A = {α_{ij}* ..... «к.,} — множество *(K)* состояний автомата; X= {x, ... *x_M]* — множество *(M)* входных сигналов; У={у, ... y_N} — множе ство *(N)* входных сигналов; f = 0, 1, 2, 3, ...; /4(0) = $α_0$ — начальное состояние автомата.
- *RG₃,RG_Z:=R-\(RG₃,RG₂*
- Рис. 15.3.1. Граф-схема алгоритма операции умножения двоичных чисел
- Функция переходов для обоих аппаратов представляет собой состояние автомата *A(t+1)* в момент времени í+1 в зависимости от состояния *A(t)* и сигнала X(f) в момент f. Функция выходов для аппарата Мили определяет зависимость выходного сигнала V(f) от состояния автомата *A(t)* и входного сигнала *X(t), а* для аппарата Мура — только от состояния автомата *A(t).*
- Построим граф переходов для автомата *Мили.* Граф строится в два этапа. На *первом этапе* производится разметка граф-схемы алгоритма по следующим правилам: => символом a_0 отмечается вход *первой* вершины, следующей за начальной,
- а также вход *конечной* вершины; => входьгвершин, следующих за операторными вершинами, отмечаются симво-
- лами а,, $a_г$, ...; => входы двух различных вершин, за исключением конечной, не могут быть от-
- мечены одинаковыми символами; => вход вершины может отмечаться только одним символом.
- Отметки графа нанесены на рис. 15.3.1 в виде крестиков. Для удобства перехода от размеченной граф-схемы алгоритма к графу переходов управляющего автомата вводится понятие *пути* от отметки $α_η$ к отметке a_s (от начального состояния к конечному):
- $α_η X(α_π, я_5)$ $У(я_п, a_s)a_s$
- (15.3.1)
- где X(a_n, a_s) — конъюнкция всех логических условий X_k *(k=* 1, 2, 3, ...), соответствующих условным вершинам на этом пути, причем X_k^s берут в прямой форме,

- если из данной вершины путь выходит по стрелке, отмеченной значением 1, и в инверсной форме, если путь выходит по стрелке, отмеченной значением 0; $Y(a_n, a_s)$ — множество микроопераций или микрокоманда, указанные в единственной операторной вершине, через которую проходит данный путь.
- Рис. 15.3.2. Граф переходов
- Допустимы пути, содержащие несколько условных вершин или не содержащие ни одной, а также пути, не содержащие операторной вершины.
- Рассматриваются все пути, кроме тех, в которых некоторое условие $X_t(j \in k)$ входит как в прямой, так и в инверсной форме. Множество путей (15.3.1) определяет множество переходов между состояниями a_n и a_s автомата. При построении графа переходов каждой отметке a_s на граф-схеме алгоритма или состоянию я; управляющего автомата ставят в соответствие вершину графа, а каждому пути (15.3.1) — дугу, направленную из вершины a_n в вершину a_s. Дуга отражает переход автомата из состояния a_n в состояние a_s и помечается конъюнкцией $X(a_n, a_s)$ и выходными сигналами $Y(a_n, a_s)$. Если в рассматриваемом пути отсутствуют логические вершины, то полагают $X(a_n, a_s) = 1$ (т. е. осуществляется безусловный переход); если же отсутствует операторная вершина, то полагают $Y(a_n, a_s) = y_0$, где y_0 — пустой оператор, означающий сохранение состояния, так как не выполняется никакая микрооперация.
- Построенный по изложенной методике граф переходов (рис. 15.3.2) определяет закон функционирования и структуру управляющего автомата.
- Таблица 15.3.1
- **Кодирование состояний управляющего автомата.** Для фиксации состояний управляющего автомата будем использовать триггеры. Каждому состоянию управляющего автомата поставим в соответствие некоторую кодовую комбинацию, отображаемую состоянием 0-выходов триггеров.

- Число разрядов η кода, или триггеров, можно выбрать на основании соотношения:
- $K \le 2^{\eta}$,
- где K — число состояний управляющего автомата. Для рассматриваемого случая $K=3$, $\eta = 2$. Выбранные коды состояний управляющего автомата приведены в табл. 15.3.1.
- Структурная схема управляющего автомата представлена на рис. 15.3.3. Она содержит:
- => *два RS-триггера* T_0, Т,, образующих регистр хранения информации для фиксации текущего состояния управляющего автомата с помощью выходных сигналов О,,O_0; => *дешифратор,* предназначенный для преобразования двухразрядного кода
- 0,0о в сигналы состояния a_0, а,, a_r;
- => *комбинационную схему,* которая вырабатывает управляющие сигналы V,, $У_2$, $/_3$ для операционного автомата и сигналы S_0, $Я_0$, S,, Я, для триггеров по входным сигналам $X_v X_2$, α_i, я,, a_e. Дальнейшей задачей является построение комбинационной схемы.

- **Составление таблицы функционирования комбинационной схемы.** Задание работы управляющего автомата с помощью графа переходов обеспечивает наглядность. Однако при записи аналитических выражений для выходных сигналов управляющего автомата удобней пользоваться таблицей (табл. 15.3.2). Каждая строка таблицы определяет один переход управляющего автомата. В ней указываются исходное состояние *a„,* его код O,O_0, состояние перехода a_s (конечное состояние) и его код O_tO_0, входные X и выходные /сигналы и сигналы S, Я, обеспечивающие изменение состояний триггеров. При составлении таблицы функционирования используются граф переходов (рис. 1,5.3.2), а также сведения о кодировании состояний (табл. 15.3.1) и об изменении состояния триггеров (табл. 15.3.3, где Ф — любое значение сигнала).
- В качестве примера покажем, как в табл. 15.3.2 заполняется графа «Сигналы управления триггерами S, *R»* первой строки, которой соответствует переход из состояния a_0 в в,.

- Из табл. 15.3.2 видно, что младшие разряды $О_0$ кодов a_a, a, изменяются (0-И), старшие О, сохраняют свое значение (0-»0). Так как триггер $Г_0$ должен изменить состояние и имеет вид перехода 0->1, на его вход следует подать сигнал $S_0 = 1$ (табл. 15.3.3). Триггер Г, не изменяет состояния, поэтому на его входы сигналы не подаются. Следовательно, в графу «Сигналы управления триггерами S, *R*» первой строки заносится только S_0.

- **Запись логических выражений для комбинационной схемы.** Для каждой строки структурной табл. 15.3.2 запишем логическое выражение в следующей форме: в левой части выражения перечислим выходные величины *Y, S, R* (содержимое двух последних столбцов), в правой части — конъюнкцию *a„X* текущего[1] (начального) состояния и условий перехода. В результате получим
- = e_0-1;
- ***Y*** **3,** oQ, ***H*** **I — &2 ' 1 **
- Y_z, R_0, S, = a, ·
- *нQ* **— Я-| ' У\2·**
- (15.3.2)
- Пользуясь (15.3.2), составим логические выражения для каждой выходной величины комбинационной схемы. Для этого в левой части запишем непосредственно выходную величину, а в правой части — дизъюнкцию правых частей тех соотношений (15.3.2), в которые входит указанная выходная величина комбинационной схемы. Полученные таким образом логические выражения для комбинационной схемы имеют следующий вид:
- Vi=o_0; $Y_2 = a, \cdot X_2 \cdot X,$; $У_3 = в, \cdot X_2 \cdot \chi, + \alpha_2$; $_0 = a_0 + a_2$; $R_0 = \alpha, \cdot X_2 \cdot X, + \beta, \cdot X_2$; $S, = \alpha, \cdot X_2 \cdot X,$; $Я, = \alpha_\zeta$.
- (15.3.3)
- **Построение комбинационной схемы.** Комбинационная схема строится по известным правилам с помощью выражений (15.3.3). Полная схема управляющего автомата со схемной логикой приведена на рис. 15.3.4. В комбинационную схему включен также дешифратор.

- . *УПРАВЛЯЮЩИЙ АВТОМАТ С ПРОГРАММИРУЕМОЙ ЛОГИКОЙ*
- **Принцип построения управляющего автомата.** В рассмотренном выше управляющем автомате со схемной логикой необходимая для работы операционного автомата последовательность управляющих сигналов формируется с помощью аппаратных средств. Рассмотрим другой принцип *построения управляющего ав томата, при котором генерирование управляющих сигналов задается микропрограммой,* хранимой в ячейках управляющей памяти.
- Совокупность управляющих сигналов *Y*= {y,, y_2, ...} на каждом тактовом периоде образует *микрокоманду.* Последовательность микрокоманд, предназначенную для выполнения некоторой операции, называют *Микропрограммой.* При этом выполнение операции сводится к выборке из управляющей памяти последовательно микрокоманд микропрограммы и выдаче с их помощью управляющих сигналов *Y* в операционный автомат. В управляющей памяти можно хранить много микро программ, предназначенных для выполнения различных операций. Выбор той или иной микропрограммы осуществляется с помощью команды, поступающей из оперативной памяти. Выбранная микропрограмма реализуется путем последовательного считывания *микрокоманд микропрограммы* из ячеек управляющей па мяти. При таком принципе управления в каждом такте определяется адрес ячейки в управляющей памяти, откуда должна считываться следующая микрокоманда микропрограммы. Микрокоманда микропрограммы содержит ряд полей. Для каждого поля отведено определенное количество разрядов. Совокупность полей на зывают *форматом* микрокоманды. Как правило, в формате микрокоманды, микропрограммы предусматриваются:
- => поле *управляющих сигналов,* представляющее собой микрокоманды /для управления операционным автоматом;
- => поле *условий перехода,* в котором указывается вид перехода: *условный или безусловный.* При условном переходе указывается логическое условие X_i, по которому осуществляется переход;
- => поле *адреса,* в котором указывается ориентировочный адрес следующей микрокоманды микропрограммы. В общем случае адрес зависит от логических условий. В зависимости от вида перехода и выполнения (невыполнения) логического условия указанный адрес сохраняется или модифицируется (изменяется).
- Обобщенная структура управляющего автомата изображена на рис. 15.4.1 и включает в себя помимо управляющей памяти блок микропрограммного управления, основная функция которого состоит в формировании адреса следующей микрокоманды.
- . *УПРАВЛЯЮЩИЙ АВТОМАТ С ПРОГРАММИРУЕМОЙ ЛОГИКОЙ*
- **Принцип построения управляющего автомата.** В рассмотренном выше управляющем автомате со схемной логикой необходимая для работы операционного автомата последовательность управляющих сигналов формируется с помощью аппаратных средств. Рассмотрим другой принцип *построения управляющего ав томата, при котором генерирование управляющих сигналов задается микропрограммой,* хранимой в ячейках управляющей памяти.
- Совокупность управляющих сигналов *Y*= {y,, y_2, ...} на каждом тактовом периоде образует *микрокоманду.* Последовательность микрокоманд, предназначенную для выполнения некоторой операции, называют *Микропрограммой.* При этом выполнение операции сводится к выборке из управляющей памяти последовательно микрокоманд микропрограммы и выдаче с их помощью управляющих сигналов *Y* в операционный автомат. В управляющей памяти можно хранить много микро программ, предназначенных для выполнения различных операций. Выбор той или иной микропрограммы осуществляется с помощью команды, поступающей из оперативной памяти. Выбранная микропрограмма реализуется путем последовательного считывания *микрокоманд микропрограммы* из ячеек управляющей па мяти. При таком принципе управления в каждом такте определяется адрес ячейки в управляющей памяти, откуда должна считываться следующая микрокоманда микропрограммы. Микрокоманда микропрограммы содержит ряд полей. Для каждого поля отведено определенное количество разрядов. Совокупность полей на зывают *форматом* микрокоманды. Как правило, в формате микрокоманды, микропрограммы предусматриваются:
- => поле *управляющих сигналов,* представляющее собой микрокоманды /для управления операционным автоматом;
- => поле *условий перехода,* в котором указывается вид перехода: *условный или безусловный.* При условном переходе указывается логическое условие X_i, по которому осуществляется переход;
- => поле *адреса,* в котором указывается ориентировочный адрес следующей микрокоманды микропрограммы. В общем случае адрес зависит от логических условий. В зависимости от вида перехода и выполнения (невыполнения) логического условия указанный адрес сохраняется или модифицируется (изменяется).
- Обобщенная структура управляющего автомата изображена на рис. 15.4.1 и включает в себя помимо управляющей памяти блок микропрограммного управления, основная функция которого состоит в формировании адреса следующей микрокоманды.

- По состоянию полей адреса и условий перехода текущей микрокоманды, *а* также по значению сигналов логических условий, выдаваемых операционным автоматом, в блоке микропрограммного управления формируется адрес ячейки памяти, в которой хранится следующая микрокоманда исполняемой микропрог раммы. В следующем тактовом периоде микрокоманда считывается из управляющей памяти. Разряды поля *управляющих сигналов* поступают в операционный автомат, который выполняет данную микрокоманду V_k, а разряды поля *адреса* и поля *условий перехода* — в блок микропрограммного управления, который формирует адрес очередной микрокоманды. Процесс продолжается до тех пор, пока не будет выполнена вся микропрограмма. Так как структура управляющего автомата стандартна, основные усилия разработчика направлены на составление микропрограммы, которая записывается в ячейки постоянного запоминающего устройства.
- Составим микропрограмму для выполнения операции умножения двоичных чисел по рассмотренному выше алгоритму, представленному на рис. 15.3.1.
- **Выбор формата и числа разрядов микрокоманды.** Как отмечено выше, в формате микрокоманды микропрограммы должно быть предусмотрено *поле адреса,* которое содержит код адреса следующей микрокоманды. Выберем число разрядов кода адреса равным трем, что позволит хранить в управляющей памяти адреса восьми микрокоманд. Если выполнение микрокоманд не связано с логическими условиями, то адресный код/4_2/4,/4_0 передается через блок микропрограммного управления в управляющую память без изменения. При наличии условного перехода адрес в блоке микропрограммного управления модифицируется. *По этой причине в формате микрокоманды предусмотрено поле условий переходов,* содержащее три разряда *П, ПХ*$_2$, /7Х,. Значение /7 = 0 соответствует безусловному переходу. В этом случае разряды /7$Х_2$, *Пх^* микрокоманды микропрограммы могут принимать любые значения (0 или 1). Значение *П*= 1 инициирует проверку логических условий $Х_2$ или х,. При проверке логического условия $Х_2$ необходимо установить $ПХ_2$ = 1, ЛХ, = 0; при проверке логического условия х, — $ЛХ_2$ = О, ЛХ, = 1. Следует отметить, что сигналы $Х_2$, Х, логических условий формирует операционный автомат, а разряды *П,* $ЛХ_2$, ЛХ, проверки логических условий содержатся в микрокомандах микропрограммы, которая хранится в управляющей памяти (рис. 15.4.1). В *поле управляющих сигналов* заносятся значения микрокоманд У,, $У_2$, $У_3$ или сигналов у,, $у_2$, $у_3$, $у_4$, $у_5$, активизирующих выполнение микроопераций. В табл. 15.4.1 приведен выбранный формат микрокоманды микропрограммы.
- Таблица 15.4.1
- Формат микрокоманды микропрограммы
-
- Поле адреса
- Поле условий перехода
- Поле управляющих сигналов
- \
- Λ
- *0
- *П*
- *пх*$_2$
- ЛХ,
- $У_5$
- $У_4$
- У
- У,
- **Разметка граф-схемы алгоритма.** Каждой вершине граф-схемы алгоритма (рис. 15.3.1) операции умножения двоичных чисел за исключением начальной поставим в соответствие микрокоманду микропрограммы. Микрокоманды МК1, МК2, МКЗ, ... микропрограммы будем хранить в ячейках управляющей памяти с адресами 000, 001, 010, ... соответственно. Для обращения к ячейкам'воспользуемся естественной адресацией, при которой различают микрокоманды двух типов: *операционные* и *управляющие.* Тип микрокоманды зависит от значения одноразрядного поля признака П: значению П = 0 соответствует операционная

- микрокоманда, значению П = 1 — управляющая микрокоманда. *Операционная* микрокоманда выполняется операционным автоматом под управлением сигналов $\{y_k\}$ или микрокоманд У,, $У_2$, $У_3$. *Управляющая* микрокоманда предназначена для реализации условных переходов в соответствии со значениями проверяемых условий X,, X_2. При разметке граф-схемы алгоритма в каждую вершину вносится следующая информация:
- => *записывается одна из микрокоманд микропрограммы и ее адресный код.* Номер микрокоманды и соответствующий ему адрес ячейки памяти проставляются согласно алгоритмической последовательности выполнения операции умножения. Если *А* — адрес текущей микрокоманды, то адрес следующей микрокоманды при безусловном переходе равен *А* + 1 , при условном переходе - *А* + 1 + X_k(/f=1,2);
- => *фиксируется состояние операционного автомата* ОА. В операторных вершинах проставляется операционная микрокоманда (У,, $У_2$ или $У_3$), которую выполняет операционный автомат. В условных вершинах операционный автомат не выполняет микрокоманд, поэтому ставится прочерк «-»;
- => *фиксируется состояние* блока микропрограммного управления БМУ. В опера торной вершине выполняется безусловный переход, в условной вершине — условный переход, поэтому делаются соответствующие записи и указывается по какому условию X, или X_2 выполняется управляющая команда. Размеченная по указанным правилам граф-схема алгоритма приведена на рис. 15.4.2. В операторную вершину с микрокомандой У, занесена первая микрокоманда МК1 и адрес ее ячейки памяти 000. В условной вершине X_2 = ? записана микрокоманда МК2 с адресом 001. Адреса следующих двух микрокоманд определяется по приведенной выше формуле
-

- Следовательно, микрокоманда МКЗ с адресом 010 заносится в вершину проверки логического условия X, = ?, а МК4 с адресом 011— в вершину «Конец» (возможно для выполнения другой микропрограммы).
- При вычислении адресов следующих двух микрокоманд необходимо иметь в виду, что последняя занятая ячейка памяти имеет адрес 011. Поэтому
- *A+1* +x, =
- 011+1 + 0 = 100— адресМКб приX,=0, 011+1 + 1 = 101 ~ адресМК6 приX,=1,
- поэтому микрокоманда МК5 с адресом 100 заносится в вершину с микрокомандой $У_3$, а МК6 с адресом 101— в вершину с микрокомандой $У_2$.
- **Составление микропрограммы.** Микропрограмма составляется в соответствии с размеченной граф-схемой алгоритма (рис. 14.4.2) и представляется в виде таблицы (табл. 15.4.2).

- В ячейку с адресом 000 помещаем МК1, которая в операционном автомате выполняет загрузку, предусмотренную управляющими сигналами у,, $у_2$ (У,), а микрокоманда блока микропрограммного управления определяет безусловный переход (БП) к ячейке с адресом 001.
- В ячейке 001 располагаем МК2, которая не предусматривает никаких действий в операционном автомате, управляющая микрокоманда блока микропрограммного управления определяет условный переход по условию $Х_2$: при $Х_2 = 0$ происходит переход в следующую ячейку с адресом 010, при $Х_2 = 1$ — к ячейке 011, где хранится МК4 для продолжения программы после выполнения операции умножения.
- Микрокоманда МКЗ в ячейке 010 также не предусматривает действий в операционном автомате и предназначена для осуществления перехода по условию Х,: при Х, = 0 — переход к ячейке 100, при Х, = 1 — к ячейке с адресом 0101, где соответственно хранятся МК5 и МК6.
- Микрокоманда МК5 выполняет в операционном автомате действия, предусмотренные управляющими сигналами $у_5$, $у_4$, а в БМУ — безусловный переход к ячейке с адресом 001.
- Микрокоманда М Кб предусматривает в операционном автомате действия $У_2$, а в блоке микропрограммного управления — безусловный переход к ячейке 100 (МК5).
- При отсутствии проверки логических условий П = О значения $Х_2$ и Х, могут быть любыми.

8, 16 и 32 РАЗРЯДНЫЕ МИКРОКОНТРОЛЛЕРЫ

- Микроконтроллеры (МК) - это класс специализированных микропроцессоров (МКП) для реализации разнообразных устройств управления (в том числе и бытовой аппаратуры). Номенклатура выпускаемых МК исчисляется несколькими тысячами типов, а общий годовой объем их выпуска составляет миллиарды экземпляров.
- Особенностью МК является размещение на одном кристалле помимо центрального процессора (ЦП) внутренней памяти и большого набора периферийных устройств.
- В состав периферийных устройств обычно входят от одного до восьми 8-разрядных параллельных портов ввода-вывода данных, один или два последовательных порта,

- Кроме этого в МК входят такие специализированные устройства, как блок формирования сигналов с широтно-импульсной модуляцией, контроллер жидкокристаллического дисплея и ряд других. Благодаря использованию внутренней памяти и периферийных устройств реализуемые на базе МК системы управления содержат минимальное количество дополнительных компонентов.
- Для удовлетворения запросов потребителей выпускается большая номенклатура МК, которые принято подразделять на 8-, 16- и 32-разрядные.

- *8-разрядные МК* являются наиболее простыми и дешевыми изделиями этого класса, ориентированными на использование в относительно несложных устройствах массового выпуска.
- МК этой группы обычно выполняют с относительно небольшим набор команд (50-100).
- В этих МК используются наиболее простые способы адресации.
- Основными областями применения 8-разрядных МК являются промышленная автоматика, автомобильная электроника, измерительная техника, теле-, видео- и аудиотехника, средства связи, бытовая аппаратура.

- Для 8-разрядных МК характерна Гарвардская архитектура.
- => с отдельной внутренней памятью *для хранения программ,* в качестве которой используются масочно-программируемые ПЗУ (ROM), однократно программируемое ПЗУ (PROM) или электрически репрограммируемое ПЗУ (EPROM, EEPROM или Flash) с объемом от нескольких единиц до десятков килобайт;
- => с отдельной внутренней памятью *для хранения данных, в* качестве которой используется регистровый блок, организованный в виде нескольких регистровых банков, или ОЗУ. Ее объем составляет от нескольких десятков байт до нескольких килобайт.

- В случае необходимости имеется возможность дополнительно подключать внешнюю память команд и данных объемом до 64-256 Кбайт.

- Для повышения производительности во многих моделях 8-разрядных микроконтроллеров реализованы принципы RISC-архитектуры, обеспечивающие выполнение большинства команд за один такт машинного времени.

- *16-разрядные микроконтроллеры* помимо повышенной разрядности обрабатываемых данных характеризуются:
- => повышенной производительностью;
- => расширенной системой команд и способов адресации;
- => увеличенным набором регистров и объемом адресуемой памяти;
- => возможностью расширения объема памяти программ и данных до нескольких мегабайт путем подключения внешних микросхем памяти;
- => программной совместимостью с 8-разрядными микроконтроллерами и другими возможностями.
- Основные области применения — сложная промышленная автоматика, телекоммуникационная аппаратура, медицинская и измерительная техника

- *32-разрядные МК* ориентированы на применение в системах управления *сложными* объектами промышленной автоматики (средствами комплексной автоматизации производства, робототехнические устройствами, двигателями и др.), в контрольно-измерительной аппаратуре, телекоммуникационном оборудовании и других сложных

- 32-разрядные микроконтроллеры содержат:
- => высокопроизводительный *CISC-* или *RISC-процессор,* соответствующий по своим возможностям младшим моделям микропроцессоров общего назначения. Например, в микроконтроллерах компании Intel используется процессор I386, а в микроконтроллерах компании Motorola — процессор 680x0. Введение этих процессоров в состав микроконтроллеров позволяет использовать в соответствующих системах управления огромный объем прикладного и системного программного обеспечения, созданный ранее для соответствующих персональных компьютеров. Некоторые типы МК содержат несколько исполнительных конвейеров, образующих суперскалярную структуру;

- => *внутреннюю память команд* емкостью до десятков килобайт и *память данных* емкостью до нескольких килобайт;
- => *средства для подключения внешней памяти* объемом до 16 Мбайт и выше;
- => набор сложных периферийных устройств — таймерный процессор, коммуникационный процессор, модуль последовательного обмена и ряд других.
- Во внутренней структуре этих МК реализуется Принстонская или Гарвардская архитектура.

УСТРОЙСТВА ПАМЯТИ И ХРАНЕНИЯ ДАННЫХ

- **Назначение и виды устройств.** Устройства памяти, или память, МК предназначены для кратковременного и долговременного хранения информации. Память можно рассматривать как сложную иерархическую систему. Для ее классификации используется множество различных признаков: функциональное назначение, принцип действия и способ хранения, организация использования, способ доступа, энергозависимость и др. В МК и МП обычно выделяют два основных вида памяти: *внутреннюю* и *внешнюю* память .
- Внутренняя память. Для нее характерны следующие признаки:
- => память *предназначена* для работы в условиях, когда необходимо производить выборку (и обновление) информации в высоком темпе работы процессора. Она является *непосредственно доступной* для процессора;
- =» память *выполняется* в интегральном исполнении;
- => информация *хранится* в массиве ячеек. Минимальной адресуемой единицей информации является байт. Каждый байт памяти имеет свой уникальный адрес. Адрес является одномерным и представляет собой двоичное число определенной разрядности;
- => *процессор имеет непосредственный доступ* к внутренней памяти, который осуществляется по адресу, заданному программой.

- **Внутренняя память** подразделяется:
- => на *оперативную память,* информация в которой может изменяться процессором в любой момент времени. Обращение к ячейкам оперативной памяти, как по чтению, так и по записи может происходить в любом порядке, поэтому оперативную память называют памятью с произвольным доступом Random Access Memory (RAM);
- => на *постоянную память,* информация в которой остается неизменной в течение длительного времени. Процессор может только считывать информацию из памяти- ROM — память только для чтения.
- **Внутреннюю память** обычно называют *основной* памятью МК.
- **Внешняя память.** Для нее характерны следующие признаки:
- => память *предназначена* для достаточно длительного хранения информации;

- **Основные параметры.** К наиболее важным параметрам устройств памяти
- следует отнести:
- *=> объем* (емкость) хранимой *информации.* Основообразующей единицей для оценки объема памяти является байт. Для оценки больших объемов памяти используются следующие единицы: килобайты (1 Кбайт = 2^{10}байт), мегабайты (1 Мбайт = 2^{20}байт), гигабайты (1 Гбайт = 2^{30}байт). Размер файлов также оценивается в байтах.
- => *время доступа,* которое оценивается усредненной задержкой начала обмена полезной информацией относительно появления запроса на данные.
- *-> скорость обмена* при передаче потока данных (после задержки на время доступа).
- => удельная *стоимость хранения* единицы данных, определяемая отношением цены накопителя (с носителями) к единице хранения (байту или мегабайту).
- *-> другие параметры.* К ним следует отнести *энергонезависимость* (способность сохранения информации при отключении внешнего питания), *устойчивость* к внешним воздействиям, *время* и *надежность* хранения, *конструктивные особенности* (размер, масса) и др.