




Основы профессиональной деятельности Часть первая.

Клименков С.В.
2018-2019 уч. год
v.1.21 от 16.10.2018


1

В разработке данного конспекта приняли непосредственное участие: Хорошева Виктория Валерьевна, Барышок Евгений Сергеевич, Ткешелашвили Нино Мерабиевна, Афанасьев Дмитрий Борисович, Киреев Валерий Юрьевич, Цопа Евгений Алексеевич, Яркеев Александр Сергеевич и др. Автор выражает им свою глубокую признательность.



Литература

- Введение в микроЭВМ / С. А. Майоров, В. В. Кириллов, А. А. Приблуда. — Л. Машиностроение, 1988. — 304 с. ISBN 5-217-00180-1
- Кириллов В.В. Архитектура базовой ЭВМ — СПб: СПбГУ ИТМО, 2010. - 144с.
- Онлайн-ресурсы
<https://se.ifmo.ru/disciplines/csbasics>
 - Методические указания к лабораторным работам
 - Генераторы вариантов
 - Эмулятор "базовой ЭВМ"
- Дополнительная литература ОЦ



2

Основным источником информации по курсу является конспект лекций на основе презентации, методические указания к лабораторным работам (электронная, выложена и периодически обновляется на сайте. Две лабораторные работы этого семестра делаются в соответствии с требованиями, указанные этой методичке. Учебник (в библиотеке или на онлайн-ресурсе) используйте для получения дополнительной информации. В учебнике есть опечатки.

Информация о дополнительных заданиях, изменении в расписании и другие организационные вопросы, все транслируется на сайте и дублируется в Вконтакте. ВК меня можно найти по id serge_klimenkov.

Так же будут организованы консультационные занятия, куда вы сможете приходиться и задавать свои вопросы по учебе.

Для различных частей курса может потребоваться дополнительная литература, для части, посвященной архитектуре ЭВМ можно использовать учебник Таненбаума, которая является достаточно большой, но полезной книжкой по архитектуре ЭВМ, которая точно потребуется на старших курсах. Другая необходимая дополнительная литература будет названа на соответствующих лекциях.



БАРС

- Отличается от ЦДО (<http://de.ifmo.ru>)
- Используем журнал в Google Documents

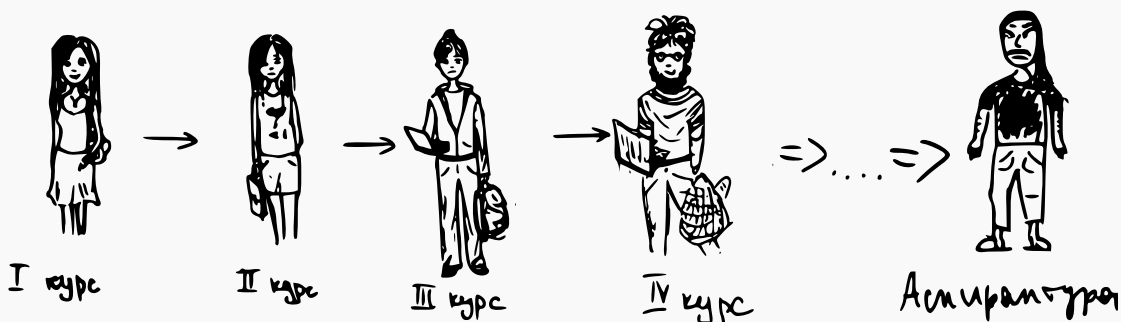
Задания	Кол-во Сем. 1	Кол-во Сем. 2	Баллы Мин.	Баллы Макс.
<i>Лабораторные работы</i>	2	4+1	?	?
<i>Текущее тестирование</i>	2	2	?	?
<i>Рубежи</i>	2	2	6	10
<i>Личностные качества</i>	2	2	0	3
<i>Зачет/Экзамен</i>	Зачет	Экзамен	12	20

3



Кем можно стать после окончания ПИИКТ?

1



Эволюция девочки с ВТ (С) Локальный мем



Немного формальностей. Что значит «работать»?

- *Вид профессиональной деятельности* - совокупность обобщенных трудовых функций, имеющих близкий характер, результаты и условия труда;
- *Обобщенная трудовая функция* - совокупность связанных между собой трудовых функций, сложившаяся в результате разделения труда в конкретном производственном или (бизнес) процессе;
- *Трудовая функция* - система трудовых действий в рамках обобщенной трудовой функции;
- *Трудовое действие* - процесс взаимодействия работника с предметом труда, при котором достигается определенная задача.



Вид профессиональной деятельности — это обобщенное описание того, чем вы будете заниматься, поступив на свою будущую работу. С точки зрения государственных органов Российской Федерации (таких, как министерство образования и науки, министерства труда и социальной защиты) все профессии достаточно строго регламентированы и описаны, что позволяет работодателям и образовательным организациям установить, что необходимо первым и кого необходимо готовить вторым. Государство постепенно подводит к тому, что определенным видом профессиональной деятельности может заниматься только те сотрудники, которые прошли соответствующую подготовку и сдали необходимые квалификационные испытания. Вид профессиональной деятельности состоит из обобщенных трудовых функций — совокупности конкретной деятельности сотрудника на его рабочем месте, позволяющих достичь основной цели работодателя. В современном мире — основной целью обычно является извлечения прибыли, без которой ни одна коммерческая компания не может существовать. Обобщенные трудовые функции детально специфицируют конкретные трудовые функции, задачи и трудовые действия, которые работник должен выполнять на своем рабочем месте.

Например, первобытный человек изобрел каменный топор для выполнения нескольких трудовых действий — рубить дерево, добыть животное как средство пропитания. Соответственно, с точки зрения его занятости, его вид профессиональной деятельности может быть описан как плотник или охотник.

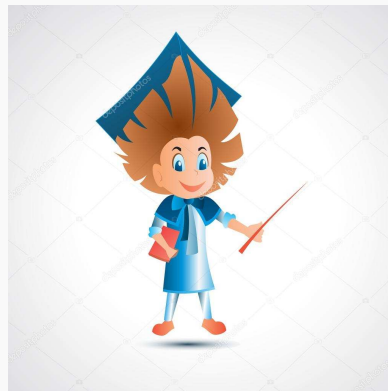
Обобщенные трудовые функции охотника в таком случае могут быть описаны как поиск и выслеживание дичи, добыча дичи, транспортировка объектов охоты к жилищу, обработка результатов охоты. Для обобщенной трудовой функции «добыча дичи» могут быть определены трудовые функции, такие как — поиск и выбор подходящих путей миграции дичи, подготовка ловушек, устройство засады, ожидание и пр. Соответственно, использование каменного топора, является лишь одним из многих трудовых действий, которые определены для профессии «охотник».

Мы поговорим с вами о профессиях, где основным средством труда является компьютер, и другие устройства, с ним связанные.



Направления подготовки бакалавриата

- 09.03.01 Информатика и вычислительная техника
 - ВТ
- 09.03.04 Программная инженерия
 - ВТ, ИПМ, ГТ, КОТ
- 44.03.04 Профессиональное обучение
 - ГТ





Профстандарты Минтруда для выпускников ИВТ (09.03.01)

- 06.001 «Программист»
- 06.004 «Специалист по тестированию в области ИТ»
- 06.011 «Администратор баз данных»
- 06.015 «Специалист по информационным системам»
- 06.016 «Руководитель проектов в области ИТ»
- 06.019 «Технический писатель»
- 06.022 «Системный аналитик»
- 06.025 «Специалист по дизайну графических и пользовательских интерфейсов»
- 06.026 «Системный администратор информационно-коммуникационных систем»
- 06.027 «Специалист по администрированию сетевых устройств информационно-коммуникационных систем»
- 06.028 «Системный программист»




Профстандарты Минтруда для выпускников ПИ (09.03.04)

- 06.001 «Программист»
- 06.004 «Специалист по тестированию в области ИТ»
- 06.022 «Системный аналитик»
- 06.028 «Системный программист»



Профстандарты Минтруда для выпускников ПО (44.03.04)

- 01.003 «Педагог дополнительного образования детей и взрослых»
- 01.004 «Педагог профессионального обучения, профессионального образования и дополнительного профессионального образования»



ИТМО BT

06.001 «Программист»

- Разработка, отладка, проверка работоспособности, модификация ПО
 - Разработка и отладка программного кода
 - Проверка работоспособности и рефакторинг кода программного обеспечения
 - Интеграция программных модулей и компонент и верификация выпусков программного продукта
 - Разработка требований и проектирование программного обеспечения

10

Основная цель вида профессиональной деятельности: Разработка, отладка, проверка работоспособности, модификация программного обеспечения. Обобщенные трудовые функции для данного вида деятельности включают в себя следующие трудовые функции:

- Разработка и отладка программного кода (формализация и алгоритмизация поставленных задач; написание программного кода с использованием языков программирования, определения и манипулирования данными; оформление программного кода в соответствии с установленными требованиями; работа с системой контроля версий; проверка и отладка программного кода)
- Проверка работоспособности и рефакторинг кода программного обеспечения (разработка процедур проверки работоспособности и измерения характеристик программного обеспечения; разработка тестовых наборов данных; проверка работоспособности программного обеспечения; рефакторинг и оптимизация программного кода; исправление дефектов, зафиксированных в базе данных дефектов)
- Интеграция программных модулей и компонент и верификация выпусков программного продукта (разработка процедур интеграции программных модулей; Осуществление интеграции программных модулей и компонент и верификации выпусков программного продукта)
- Разработка требований и проектирование программного обеспечения (анализ требований к программному обеспечению; разработка технических спецификаций на программные компоненты и их взаимодействие; проектирование программного обеспечения)



06.004 «Специалист по тестированию в области ИТ»

- Оценка качества разрабатываемого ПО путем проверки соответствия продукта заявленным требованиям, сбора и передачи информации о несоответствиях
 - Подготовка тестовых данных и выполнение тестовых процедур
 - Разработка тестовых случаев, проведение тестирования и исследование результатов
 - Разработка документов для тестирования и анализ качества покрытия
 - Разработка стратегии тестирования и управление процессом тестирования

11

Основная цель вида профессиональной деятельности: оценка качества разрабатываемого ПО путем проверки соответствия продукта заявленным требованиям, сбора и передачи информации о несоответствиях. Обобщенные трудовые функции для данного вида деятельности включают в себя следующие трудовые функции:

- Подготовка тестовых данных и выполнение тестовых процедур (подготовка выполнения рабочего задания; подготовка тестовых данных в соответствии с рабочим заданием; выполнение процесса тестирования; регистрация дефектов в системе контроля (базах данных); тестирование сопроводительной документации на соответствие требованиям заказчика).
- Разработка тестовых случаев, проведение тестирования и исследование результатов (определение и описание тестовых случаев, включая разработку автотестов; проведение тестирования по разработанным тестовым случаям; восстановление тестов после сбоев, повлекших за собой нарушение работы системы; анализ результатов тестирования; проверка исправленных дефектов в порядке их приоритета; предоставление результатов тестирования руководителю группы (отдела) тестировщиков; деятельность по обучению младших тестировщиков)
- Разработка документов для тестирования и анализ качества покрытия (оценка требований исходной документации; определение требований к тестам; разработка тестовых документов, включая план тестирования; оценка тестов; подбор персонала совместно с руководителем подразделения и специалистом соответствующей службы; проведение обучения тестировщиков)
- Разработка стратегии тестирования и управление процессом тестирования (выявление приоритетных функций для покрытия тестирования; утверждение с аналитиком (и/или руководителем проекта) требований заказчика; формирование и утверждение стратегии тестирования; организация рабочего процесса команды специалистов по тестированию (включая оценку трудозатрат); мониторинг работ и информирование о ходе работ заинтересованных лиц; проведение интервью, оценка технических знаний кандидата на замещение вакансии)




06.016 «Руководитель проектов в области ИТ»

- Менеджмент проектов в области ИТ (планирование, организация исполнения, контроль и анализ отклонений) для эффективного достижения целей проекта в рамках утвержденных заказчиком требований, бюджета и сроков
 - Управление проектами в области ИТ на основе полученных, планов проектов в условиях, когда проект не выходит за пределы утвержденных параметров
 - Управление проектами в области ИТ малого и среднего уровня сложности в условиях неопределенностей, порождаемых запросами на изменения, с применением формальных инструментов управления рисками и проблемами проекта
 - Управление проектами в области ИТ любого масштабав условиях высокой неопределенности, вызываемой запросами на изменения и рисками, и с учетом влияния организационного окружения проекта; разработка новых инструментов и методов управления проектами в области ИТ

12

Всего профстандарт включает в себя более 150 трудовых функций. К ним относятся:

- Идентификация конфигурации, ведение отчетности по статусу конфигурации, аудит конфигураций, организация репозитория проекта в области ИТ, управление выпуском и поставкой информационной системы.
- Планирование управления, проверка реализации, анализ и согласование запросов на изменение.
- Планирование управления, организация, мониторинг выполнения, заключения договоров, организация заключения дополнительных соглашений, закрытие договоров в проектах.
- Организационное и методологическое обеспечение регистрации запросов заказчика, регистрация обработка и закрытие запросов заказчика.
- Планирование управления, согласование и утверждение, управление распространением, хранением и контролем документации.
- Планирование управления и привлечение (набор) персонала. Командообразование, управление эффективностью и развитие команды.
- Подготовка предложений по новым инструментам и методам управления проектами, повышения их эффективности, развитие офиса управления проектами в организации.
- Сбор информации для инициации проекта, планирование, организация исполнения работ, мониторинг и управление работами, общее управление изменениями, завершение проекта.
- Планирование, подготовка к выбору и выбор поставщиков, исполнение и закрытие закупок.
- Планирование, обеспечение и контроль качества в проектах в области ИТ.
- Организация приемо-сдаточных испытаний.
- Планирование выявления требований, организация и управление выполнением работ по выявлению требований.
- Организация выполнения анализа, управление анализом, согласование и утверждение требований в соответствии с полученными планами.
- Принятие и реализация мер по неразглашению информации, полученной от заказчика.
- Принятие мер для своевременного финансирования проектов, планирование субподряда, подбор субподрядчиков, управление исполнением субподрядных работ.
-



06.028
«Системный программист»

- Разработка, отладка, модификация и поддержка системного программного обеспечения
 - Разработка компонентов системных программных продуктов
 - Разработка систем управления базами данных
 - Разработка операционных систем
 - Организация разработки системного программного обеспечения
 - Интеграция разработанного системного программного обеспечения

13

Основной целью вида деятельности является разработка, отладка, модификация и поддержка системного программного обеспечения. Эта деятельность включает в себя:

- Разработка компонентов системных программных продуктов (разработка драйверов устройств; разработка компиляторов, загрузчиков, сборщиков; разработка системных утилит; создание инструментальных средств программирования).
- Разработка систем управления базами данных (Разработка и отладка компонентов системы управления базами данных; документирование и сопровождение разрабатываемой системы управления базами данных и ее компонентов)
- Разработка операционных систем ((формирование требований, разработка архитектуры, написание компонентов и контроль соблюдения архитектуры в процессе написания операционной системы; отладка разрабатываемых компонентов операционной системы; документирование и сопровождение разрабатываемой операционной системы).
- Организация разработки системного программного обеспечения (планирование разработки системного программного обеспечения; формирование группы программистов, организация работы и контроль группы программистов для разработки системного программного обеспечения; предоставление заказчику результатов разработки системного программного обеспечения).
- Интеграция разработанного системного программного обеспечения (включая планирование интеграции разработанного системного программного обеспечения).



06.025 «Специалист по дизайну графических и пользовательских интерфейсов»

- Проектирование, графический дизайн и юзабилити-исследование интерактивных пользовательских интерфейсов, обеспечивающих высокие эксплуатационные (эргономические) характеристики программных продуктов и систем
 - Подготовка интерфейсной графики
 - Графический дизайн интерфейса
 - Проектирование пользовательских интерфейсов по готовому образцу или концепции интерфейса
 - Юзабилити-исследование программных продуктов и/или аппаратных средств
 - Проектирование сложных пользовательских интерфейсов
 - Экспертный анализ эргономических характеристик

14

Основная цель вида профессиональной деятельности: проектирование, графический дизайн и юзабилити-исследование интерактивных пользовательских интерфейсов, обеспечивающих высокие эксплуатационные (эргономические) характеристики программных продуктов и систем. Включает в себя следующие трудовые функции:

- Подготовка интерфейсной графики (графический дизайн по ранее определенному визуальному стилю; подготовка графических материалов для включения в интерфейс).
- Графический дизайн интерфейса (создание визуального стиля интерфейса; создание стилизованных руководств к интерфейсу; визуализация данных).
- Проектирование пользовательских интерфейсов по готовому образцу или концепции интерфейса (проектирование интерфейса по концепции или по образцу уже спроектированной части интерфейса; формальная оценка интерфейса; анализ обратной связи о пользовательском интерфейсе продукта).
- Юзабилити-исследование программных продуктов и/или аппаратных средств (формирование выборки респондентов (участников юзабилити-исследования или иного эргономического тестирования интерфейса); планирование и проведение, сбор данных и анализ юзабилити-исследования).
- Проектирование сложных пользовательских интерфейсов (разработка проектной документации по проектированию интерфейсов; создание формальных методик оценки интерфейса; концептуальное проектирование интерфейса; создание структурных руководств по проектированию интерфейса и продуктовых стандартов на пользовательский интерфейс).
- Экспертный анализ эргономических характеристик программных продуктов и/или аппаратных средств (анализ программных продуктов на предмет соответствия задачам пользователей; разработка рекомендаций по оптимизации интерфейсных решений; определение возможных вариантов интерфейсных решений, наилучшим образом соответствующих задачам пользователей).



01.004 Педагог профобучения, профобразования и ДПО

- Организация деятельности обучающихся по освоению знаний, формированию и развитию умений и компетенций, обеспечение достижения ими результатов образования; создание педагогических условий для профессионального и личностного развития обучающихся, в углублении и расширении образования; методическое обеспечение реализации образовательных программ
 - Преподавание по программам профессионального обучения, СПО и ДПП
 - Организация и проведение учебно-производственного процесса

15

Вид профессиональной деятельности: организация деятельности обучающихся по освоению знаний, формированию и развитию умений и компетенций, позволяющих осуществлять профессиональную деятельность, обеспечение достижения ими нормативно установленных результатов образования; создание педагогических условий для профессионального и личностного развития обучающихся, удовлетворения потребностей в углублении и расширении образования; методическое обеспечение реализации образовательных программ. Включает в себя следующие обобщенные трудовые функции (трудовые функции):

- Преподавание по программам профессионального обучения, среднего профессионального образования (СПО) и дополнительным профессиональным программам (ДПП), ориентированным на соответствующий уровень квалификации (организация учебной деятельности обучающихся по освоению учебных предметов, курсов, дисциплин (модулей); педагогический контроль и оценка освоения образовательной программы в процессе промежуточной и итоговой аттестации; разработка программно-методического обеспечения)
- Организация и проведение учебно-производственного процесса при реализации образовательных программ различного уровня и направленности (организация учебно-производственной деятельности обучающихся по освоению программ профессионального обучения и(или) программ подготовки квалифицированных рабочих, служащих; педагогический контроль и оценка освоения квалификации процессе учебно-производственной деятельности обучающихся; разработка программно-методического обеспечения учебно-производственного процесса)



01.004 Педагог (продолжение)

- Организационно-педагогическое сопровождение группы (курса) обучающихся по программам СПО
- Организационно-педагогическое сопровождение группы (курса) обучающихся по программам ВО
- Проведение профориентационных мероприятий
- Организационно-методическое обеспечение реализации ПО, СПО и ДПП
- Научно-методическое и учебно-методическое обеспечение ПО, СПО и ДПП

16

- Организационно-педагогическое сопровождение группы (курса) обучающихся по программам СПО (создание педагогических условий для развития группы (курса) обучающихся по программам СПО; социально-педагогическая поддержка обучающихся по программам СПО в образовательной деятельности и профессионально-личностном развитии;)
- Организационно-педагогическое сопровождение группы (курса) обучающихся по программам ВО (создание педагогических условий для развития группы обучающихся по программам высшего образования (ВО); социально-педагогическая поддержка обучающихся по программам ВО в образовательной деятельности и профессионально-личностном развитии)
- Проведение профориентационных мероприятий со школьниками и их родителями (законными представителями) (информирование и консультирование школьников и их родителей по вопросам профессионального самоопределения и профессионального выбора; проведение практикоориентированных профориентационных мероприятий со школьниками и их родителями)
- Организационно-методическое обеспечение реализации программ профессионального обучения, СПО и ДПП, ориентированных на соответствующий уровень квалификации (организация и проведение изучения требований рынка труда и обучающихся к качеству СПО и(или) (ДПО) и(или) ПО; организационно-педагогическое сопровождение методической деятельности преподавателей и мастеров производственного обучения; мониторинг и оценка качества реализации программ учебных предметов, курсов, дисциплин (модулей), практик)
- Научно-методическое и учебно-методическое обеспечение реализации программ профессионального обучения, СПО и ДПП (разработка научно-методических и учебно-методических материалов, обеспечивающих реализацию программ; рецензирование и экспертиза научно-методических и учебно-методических материалов)



01.004 Педагог (продолжение)

- Преподавание по программам бакалавриата и ДПП, ориентированным на соответствующий уровень квалификации
- Преподавание по программам бакалавриата, специалитета, магистратуры и ДПП
- Преподавание по программам аспирантуры (адъюнктуры), ординатуры, ассистентуры-стажировки и ДПП

17

- *Преподавание по программам бакалавриата и ДПП, ориентированным на соответствующий уровень квалификации (преподавание учебных курсов, дисциплин (модулей) или проведение отдельных видов учебных занятий; Организация научно-исследовательской, проектной, учебно-профессиональной и иной деятельности обучающихся под руководством специалиста более высокой квалификации; профессиональная поддержка ассистентов и преподавателей, контроль качества проводимых ими учебных занятий; разработка под руководством специалиста более высокой квалификации учебно-методического обеспечения реализации учебных курсов, дисциплин (модулей) или отдельных видов учебных занятий).*
- Преподавание по программам бакалавриата, специалитета, магистратуры и ДПП, ориентированным на соответствующий уровень квалификации (преподавание учебных курсов, дисциплин (модулей); профессиональная поддержка специалистов, участвующих в реализации курируемых учебных курсов, дисциплин (модулей), организации учебно-профессиональной, исследовательской, проектной и иной деятельности обучающихся; Руководство научно-исследовательской, проектной, учебно-профессиональной и иной деятельностью обучающихся; разработка научно-методического обеспечения реализации курируемых учебных курсов, дисциплин (модулей) программ)
- Преподавание по программам аспирантуры (адъюнктуры), ординатуры, ассистентуры-стажировки и ДПП, ориентированным на соответствующий уровень квалификации (преподавание учебных курсов, дисциплин (модулей) по программам подготовки кадров высшей квалификации и(или) ДПП; руководство группой специалистов, участвующих в реализации образовательных программ ВО и(или) ДПП; руководство подготовкой аспирантов (адъюнктов) по индивидуальному учебному плану; руководство клинической (лечебно-диагностической) подготовкой ординаторов; руководство подготовкой ассистентов-стажеров по индивидуальному учебному плану; разработка научно-методического обеспечения реализации программ подготовки кадров высшей квалификации и(или) ДПП)



Простыми словами

- Програмер
- Сисадмин
- БД-шник
- ДевОпс
- Начальник
- Педагог ВО, СПО, ДПО
- UI-щик
- Тестировщик



Типы компаний связанных с разработкой ПО в РФ

- RND центры крупных вендоров
- Стартапы и небольшие частные компании
- Software подразделения системных интеграторов
- Оборонные заводы и институты связанные с обороной
- Компании, предоставляющие Web-услуги
- Бизнес-ПО, ориентированное на управление и учет; автоматизация банков
- Промышленная автоматизация

Achtung! ИМНО

19

К категории RND относятся широко представленные офисы разработки крупных западных (и отечественных) компаний - Oracle, EMC, Intel, AMD, Veam, Yandex и другие. Они характеризуются относительно большим количеством разработчиков (100-1500), где существуют множество проектных команд, развитым типом «западной» бюрократии. Плюсами этих компаний являются возможность участвовать в разработке мейнстрим технологий, высокой заработной платой. Минусом является практически отсутствующий карьерный рост (хотя возможен рост «по диагонали»).

Стартапы характеризуются высокой нестабильностью на рынке (и в том числе это касается и заработной платы). Отношение внутри строятся по дружескому и семейному принципу. Плюсом является резкий карьерный рост, в случае, если идея находит реализацию на рынке.

Подразделения системных интеграторов — компаний, которые поставляют заказчикам интегрированные решения от разных вендоров, сами по себе достаточно крупные и обеспечивают работой до 1000 человек. Характерными примерами являются Jet Infosystems, Срос, Technoserv и др. С точки зрения трудовых отношений внутри таких компаний обычно наблюдаются крайности от «раздолбайства» до строго формализма.

В оборонной промышленности можно встретить по настоящему интересную работу, которая обеспечивается относительно невысокой заработной платой. Трудовые отношения обычно забюрократизированы по советскому (СССР) принципу. Однако, в таких компаниях возможен вертикальный карьерный рост.

Компании, предлагающие Web-информационные системы обычно имеют средний размер, с четко выраженной прикладной областью, которая может быть интересной для работника, а может и быть унылой и скучной. В них обычно предлагаются достойные зарплаты и используются современные технологии разработки.

Компании, автоматизирующие бизнес-ПО обычно используют 1С, OEBS и SAP в качестве технологической основы, со сложной предметной областью, в которой приходится одновременно быть и программистом и бухгалтером и психологом. Вилка зарплат очень большая.

Промышленная автоматизация работает на крупные российские компании (РЖД, Газпром и пр.) или на открытый рынок (StarLine). Интересная работа для увлеченных «гиков»..






Операционные системы (включая ОС UNIX)

2



21



Операционные системы

- Исторически предназначена для замены работы оператора компьютерной системы
- Условно делятся на пользовательские, серверные и встроенные ОС
- Windows, Linux/Unix, Android/IOS, VXWoks/EmbeddedLinux/DOS, Гипервизоры
- Обычно включают ядро с подсистемами управления памятью и процессами и драйверы устройств

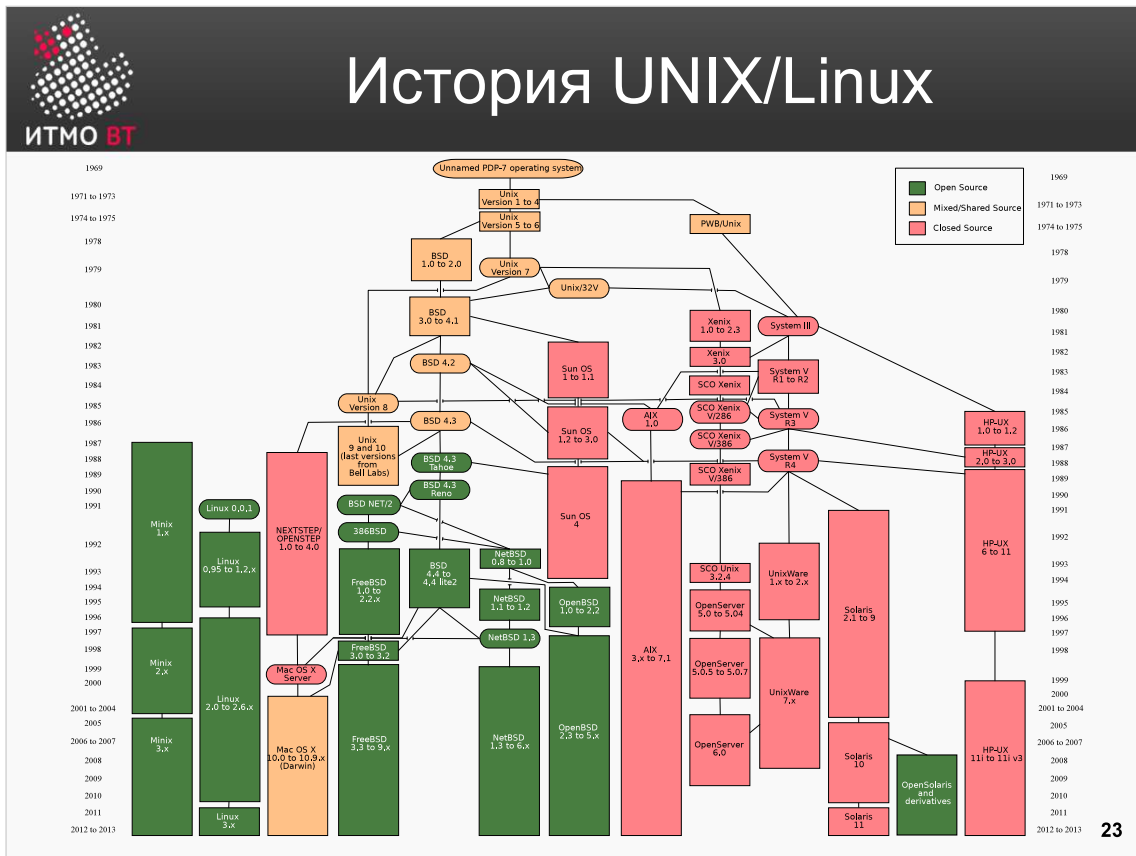
22

Оператор в компьютерных системах первых поколений занимался тем, что принимал от программистов программу и исходные данные, вводил и запускал ее на исполнение, а после ее окончания считывал результаты работы для передачи программистам. При этом дисплей и клавиатура появились у вычислительных машин не сразу, вместо них использовался так называемый пульт управления вычислительной машиной. С развитием вычислительной техники появилась естественная идея заменить оператора специальной программой, которая будет заменять его работу. Такие программы получили название операционных систем.

Операционная система загружает программы в память, контролирует их исполнение, предоставляет результаты работы в понятном для человека виде, обеспечивает возможность одновременного исполнения нескольких программ. В современном мире операционные системы можно условно разделить на пользовательские — где основной целью является предоставление конечному пользователю удобства работы с вычислительной системой, серверные — предназначенные для обработки большого количества запросов от других систем и встроенные — целью которых является поддержка функционирования аппаратного обеспечения вычислительных систем, датчиков, преобразователей сигнала и другого оборудования, например, в бортовых системах самолета, космического корабля, светофора, стиральной машины и других аппаратных средств, содержащих компьютеры.

На слайде представлены названия современных операционных систем. Отдельно следует упомянуть гипервизоры — операционные системы, которые предназначены для управления другими операционными системами. Например VMWare vSphere позволяет запускать на одной вычислительной системе несколько различных типов операционных систем одновременно.

Операционная система содержит ядро и окружение (библиотеки, файлы) пользовательских программ. В ядро обычно включаются подсистемы, позволяющие управлять и эффективно разграничивать данные и программы пользователя для исключения их неразрешенного взаимодействия между собой, драйверы аппаратного обеспечения, подсистемы обобщенного ввода и вывода информации.



Название Unix произошло от акронима UNICS (UNiplex Information and Computing System). Разработчики Кен Томпсон, Денис Риччи и Руд Канадей создали однопользовательскую (позднее двухпользовательскую), написанную на ассемблере операционную систему для популярной в то время вычислительной системы PDP-7. Позднее имя было заменено на UNIX. Позднее система была переписана на язык C и портирована на PDP-11 (популярная платформа, в том числе и в СССР, где ее представителями являлась машина ДВК и Электроника-60). Из-за ограничений американского законодательства система распространялась практически бесплатно, что и сделало ее такой популярной, в том числе и в университетских кругах.

На слайде представлено общее «генеалогическое» дерево UNIX. Можно проследить, что в развитии UNIX огромное влияние оказали несколько ветвей — BSD и System V. Особняком стоит LINUX, первые версии ядра которого, при сохранении интерфейсов для программ пользователя, были разработаны Линусом Торвалдом и MINIX - учебную операционную систему, разработанную Андрю Таненбаумом (да, тем самым автором книг по вычислительной технике). Обе, в качестве окружения пользовательских программ, используют стандарт POSIX.



Современность

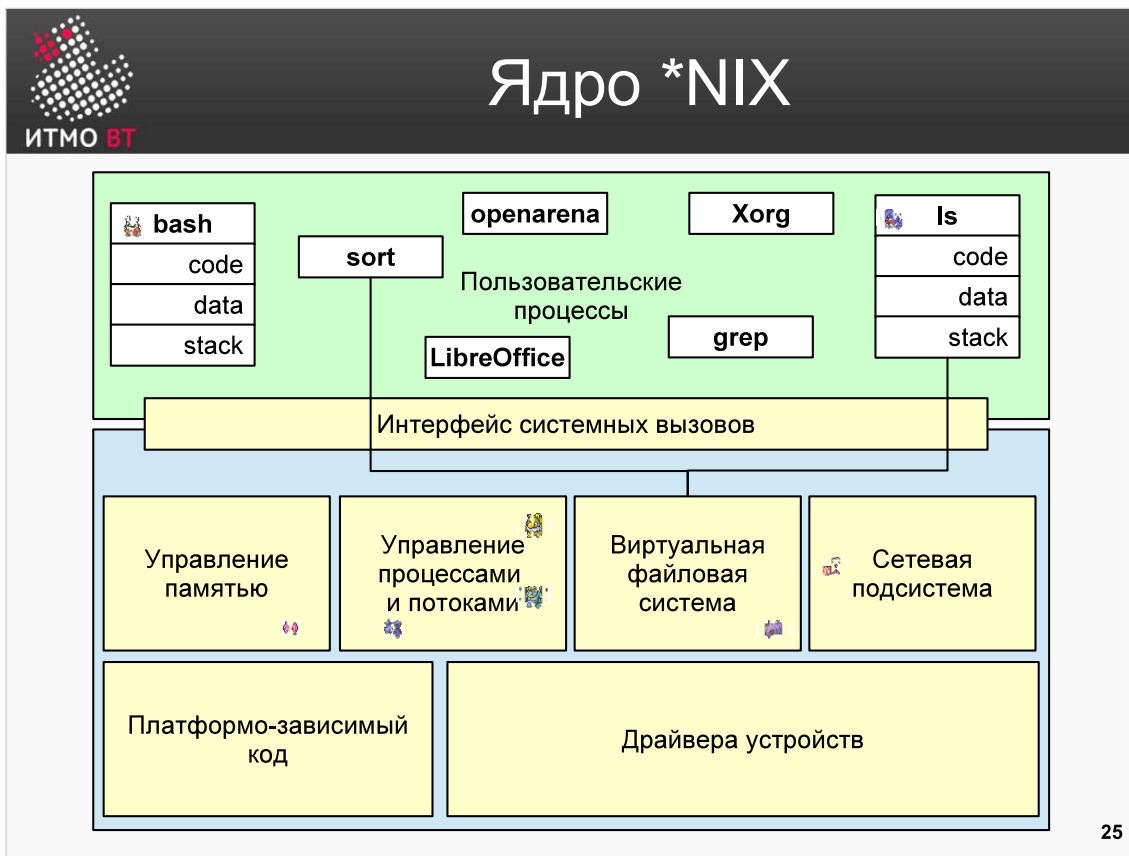
- System V
 - Solaris, AIX, HPUX
- BSD
 - Mac OS X
 - NetBSD, FreeBSD, OpenBSD
- Linux
 - RedHat, Ubuntu, SUSE
 - Fedora, Debian, OpenSUSE, ArchLinux
 - Gentoo
 - ...

24

Базовой версией коммерческих операционных систем UNIX послужило ядро ОС UNIX System V. Изначально разработанное для поддержки масштабируемых вычислительных систем оно представляет собой ядро с уникальными для всего семейства UNIX функциями программной поддержки горячей замены аппаратных компонентов (включая память и процессоры) без перезагрузки ядра, высокой масштабируемостью (одновременно могут работать от 1 до сотен процессоров, поддерживаются терабайты ОЗУ), развитой архитектурой, предоставляемой пользовательским процессам. Наиболее известными, используемыми на предприятиях, в настоящее время являются Solaris 2.x+ (Sun/Oracle), AIX (IBM) и HP/UX от компании Hewlett Packard.

BSD и ее производные часто используются для организации сетевой инфраструктуры, поскольку поддержка большого количества протоколов и средств, позволяющих управлять сетевым взаимодействием, в них широко развита. Особняком стоит MacOS — прямой наследник ОС NextStep — предоставляющая удобную оболочку для пользователя и развитый (и дада — красивый) пользовательский интерфейс.

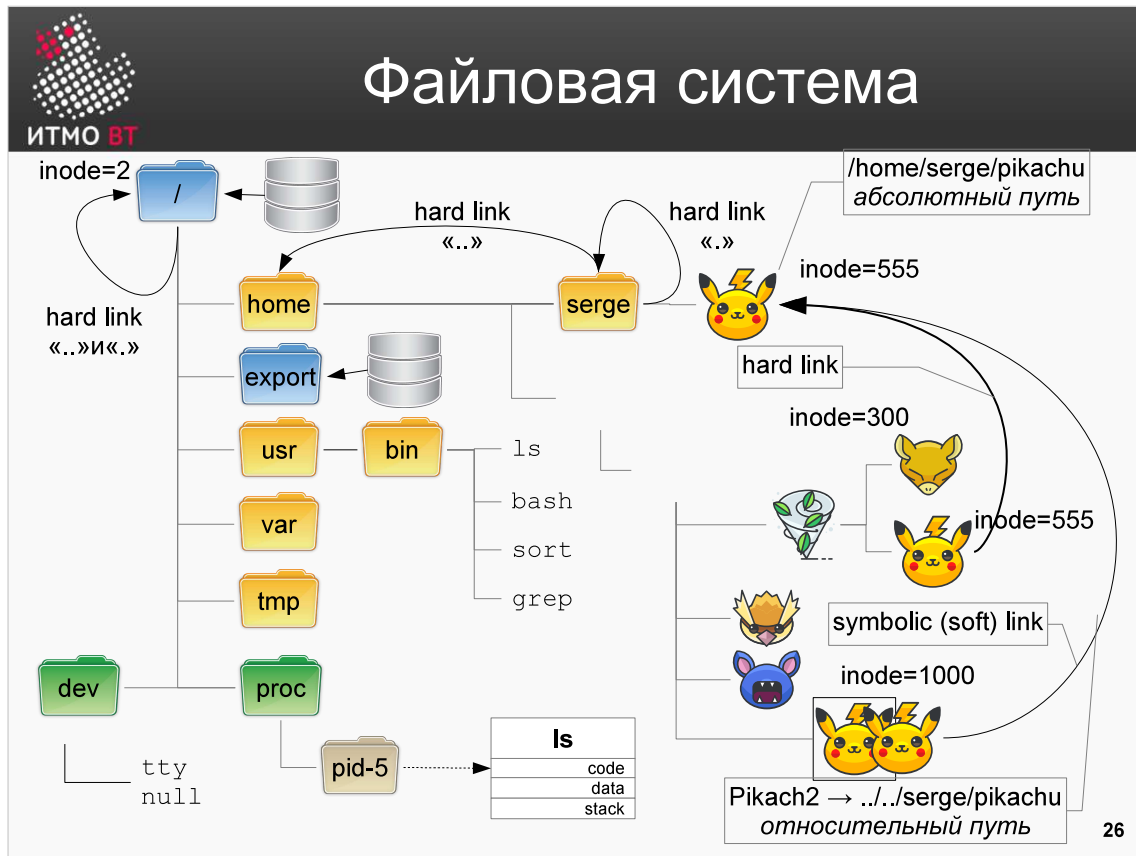
Linux представлен богатым выбором операционных систем. Отличия в версиях, которые тем не менее базируются на одном программном ядре, состоит в управлении программным обеспечением (пакетные менеджеры) и способах распространения и поддержки ОС. В них каждый пользователь может выбрать то, что ему более удобно. Отдельно распространяются десктопные (для пользователя) и серверные варианты, различающиеся набором пакетов и установках по умолчанию. Если вам хочется собирать ПО из исходников, максимально оптимизируя их для работы конкретного аппаратного обеспечения, то дистрибутив Gentoo позволяет вводить пользователю максимальное количество настроек и дополнительной оптимизации.



Каждая операционная система выполняет служебные операции в ядре. Ядро представляет собой набор подсистем, которые управляют оборудованием и программами пользователя. Основными подсистемами, которые жизненно важны для любой ОС являются управление памятью, управление процессами и потоками (отвечает за выполнение нескольких программ одновременно на одной ОС), виртуальная файловая система (обмен данными организован с помощью файлов, с точки зрения ОС в UNIX все является файлами), сетевая подсистема, драйвера устройств и платформо-зависимый код (драйвера, которые поддерживают функционирования системы на различных аппаратных архитектурах). Взаимодействие программ пользователя с ядром происходит с помощью интерфейса системных вызовов.

С точки зрения пользователя любая исполняемая программа представляет собой адресное пространство, которое разбито логически и физически на несколько сегментов. Например, сегмент code, где выполняется собственно сама программа, сегмент data, где находятся необходимые для ее работы данные, сегмент stack, для хранения адресов возврата и передачи аргументов к подпрограммам. Кроме этих основных сегментов еще используются heap или куча, сегменты разделяемых библиотек, tempory-mapped файлы, т. е. файлы непосредственно отображенные в адресное пространство процесса и другие. Все эти сегменты вместе составляют образ процесса, который работает внутри операционной системы и взаимодействует с ней.

Кроме этого, программы могут взаимодействовать между собой, используя ядро и интерфейс системных вызовов. Интерфейс системных вызовов является стандартизированным способом взаимодействия процессов с ядром, и пользовательские процессы могут получить сервисы ядра только через этот интерфейс.



На слайде изображена типичная файлов в UNIX, представляющая собой иерархию файлов и директорий (каталогов, папок), где хранится вся необходимая системная и пользовательская информация. Директории также являются файлами. Верхнем элементом иерархии файловой системы является директория root (корень иерархии). Логические разделы дисковых накопителей могут подключаться в файловую систему в *точках монтирования* (на слайде директории синего цвета) — созданных администратором директорий, в которых при подключении их файловая иерархия заменяется на расположенную на разделе диска. Каждая директория содержит два обязательных файла. Файл «.» -это ссылка на саму себя и файл «..» - это ссылка на директорию выше по файловой иерархии.

В файловой иерархии есть каталоги, которые использует сама система. К ним, например, относятся каталоги /usr/bin, где расположены основные утилиты и программы, поставляемые вместе с ОС. В каталоге /usr/lib находятся системные библиотеки. В директории /var находятся системные журналы и рабочие файлы системы, а директория /tmp предназначена для хранения временных файлов. В директории /proc находятся файлы, представляющие информацию о процессах пользователя и ОС.

В файловой системе можно адресоваться при помощи абсолютных и относительных путей. *Абсолютный путь* – это путь от корня ФС. *Относительный путь* – это путь от текущей директории, откуда была запущена программа пользователя.

В файловой системе есть специальные виды файлов, которые могут помочь организовать необходимую для пользователя или программ иерархию файлов — ссылки на другие файлы. Ссылки бывают двух типов: *жесткие* и *символические* ссылки. В ОС UNIX каждому файлу внутри файловой системы для каждого накопителя присваивается свой уникальный номер — *inode*. Директория — это просто файл, где указаны имена файлов и их *inode*. Если два имени файла имеют один и тот же *inode* (внутри одной точки монтирования) – значит это один и тот же файл и таким же содержимым. Или можно сказать, что один файл является жесткой ссылкой на другой. Файл существует в файловой системе до тех пор, пока не удалена последняя жесткая ссылка. Символическая ссылка – файл содержащий абсолютный или относительный путь до файла.

Права доступа к файлам

Кол-во жестких ссылок | Владелец | Группа-владелец | Имя

```
s207549@helios: /export/home/studs/s207549/lab0$ ls -la
total 26
drwxr-xr-x  5 s207549  studs   10 дек.  8 2015 ./
drwxr-xr-x 24 s207549  studs   37 дек.  8 2015 ../
----rw----  1 s207549  studs   21 дек.  6 2015 Conkeldurr2
lrwxrwxrwx  1 s207549  studs    5 дек.  6 2015 Copy_50 -> Xatu9/
dr-x--x-wx  5 s207549  studs    9 дек.  8 2015 Flareon0/
drwx-wxrw  4 s207549  studs    8 дек.  8 2015 Gengar7/
-rw-----  2 s207549  studs   37 дек.  6 2015 Hypno5
-rw-r--r--  1 s207549  studs  183 дек.  8 2015 Hypno5_21
-r--r-----  1 s207549  studs  285 дек.  6 2015 Psyduck4
dr-xr-xr-x  5 s207549  studs    8 дек.  8 2015 Xatu9/
```

Тип файла	Права для владельца	Права для группы	Права для остальных
-	r w -	r - -	r - -

Read, Write, eXecute

Directory, symbolic Link, file (-)

Дата:

- модификации
- последнего доступа [-u]
- изменения inode [-c]

27

На слайде показан пример вывода команды `ls -la`. Ключи `-la` показывает содержимое каталога в расширенном формате (права, размер, дату последней модификации и т.д. файла) со скрытыми файлами. Скрытыми в ОС Unix считаются файлы, начинающиеся с точки. В расширенном формате информация представлена в следующей последовательности:

- 1) Тип файла и права доступа к нему
- 2) Кол-во жестких ссылок на файл (сколько, как вы думаете, файлов и директорий в каталоге на уровень выше от показанного на слайде?)
- 3) Имя владельца
- 4) Имя группы владельца
- 5) Длина файла в байтах
- 6) Время. По умолчанию показывается время последней модификации файла, в зависимости от ключей может быть отображено время последнего доступа или изменения записи inode для файла.

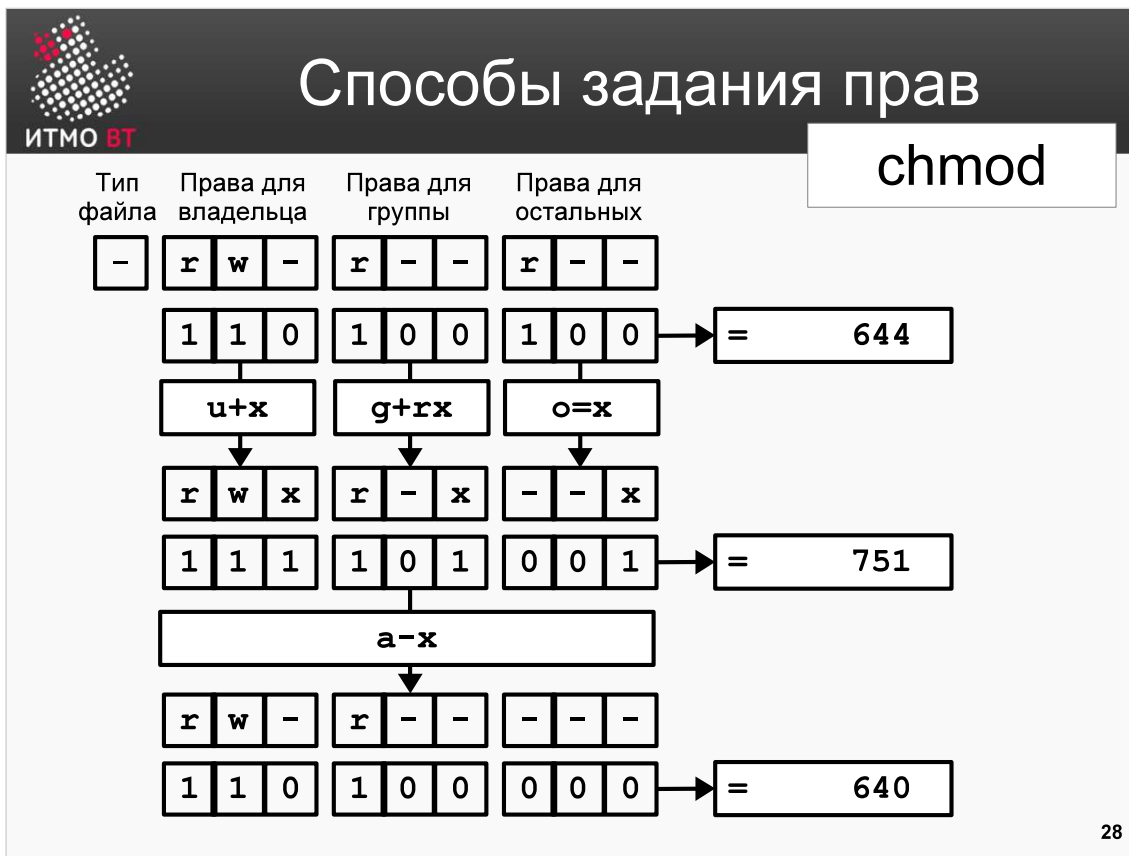
Права доступа разбиты на несколько групп и определяют права для разных категорий пользователей по отношению к текущему владельцу файла и группе владельца:

| Тип файла | Права для владельца | Права для группы | Права для остальных |

Существующие типы файлов: файл, директория или символическая ссылка. Кроме них определены дополнительные типы, как точки доступа к драйвером устройств, pipes и др.

Все типы прав содержат три значения (установленные, если право разрешено, или не установленные, тогда данное право отсутствует): право на чтение/право на запись/право на исполнение. В случае если установлено право на исполнение для директории, это значит, что в директорию можно «войти» (например, с помощью `cd`), оно проверяется в первую очередь.

Для того, чтобы задавать права существует команда `chmod` с возможностью установить права (при помощи указания их численного значения или символа «=»), убрать права («-») и добавить («+»).



При выводе команды `ls -l` права на файл отображаются в виде `gwxgwxr--`, где `r`-право на чтение, `w` – право на запись, `x` – право на исполнение, прочерк обозначает, что право было не задано. В примере `gwxgwxr--` все остальные пользователи, которые не являются владельцем и не входят в группу, не имеют права на запись и исполнение файла.

Права (для владельца, группы и остальных) принято обозначать в виде трех восьмеричных цифр. Каждая цифра отвечает за свою тройку и высчитывается так $gwx = 111_2 = 7_8$; $r-x = 101_2 = 3_8$; $--x = 001_2 = 1_8$. Таким образом, рассчитывается десятичная цифра, соответствующая определенным правам для каждой тройки типов прав. Команда `chmod 644 <имя файла>` установит права `rw - r - -`.

Еще один способ изменять права – это добавлять их или убирать с помощью буквенных обозначений. `U` – права пользователя, `G` – права группы, `O` – права для остальных. Далее можно использовать знаки `+`, `-` или `=`. Пример:

- `u+x` обозначает, что к правам пользователя следует добавить право на исполнение.
- `g+rx` – к правам группы следует добавить права на чтение и исполнение.
- `g-w` – из прав группы следует убрать право на запись.
- `o=w` – в правах для остальных пользователей установить только право на запись.

Полностью команда будет выглядеть так: `chmod u+x <имя файла>`

Потоки stdin(0), stdout(1), stderr(2)

ITMO BT

./ Flareon0
 ./ Gengar7
 Conkeldurr2 Hypno5
 Copy_50@ Hypno5_21

./Xatu9/Gothorita:
 Permission denied

students

sort

by_name

\$ sort <students >by_name 2>/dev/null
 \$

[/dev/tty]

[/dev/tty]

[/dev/tty]


[/dev/null] ✓

29

Когда запускается пользовательская программа, системный загрузчик по умолчанию открывает для процесса потоки ввода-вывода и присваивает им номера открытых файлов внутри процесса `stdin(0)` — поток стандартного ввода, `stdout(1)` — поток стандартного вывода, `stderr(2)` — поток вывода диагностических сообщений об ошибках. По умолчанию, все потоки связаны с виртуальным терминалом, откуда была запущена программа. Наличие двух потоков вывода необходимо для того, чтобы разделить вывод полезных данных и ошибок работы программы и дальнейшего перенаправления их, например, в файлы или другим командам.

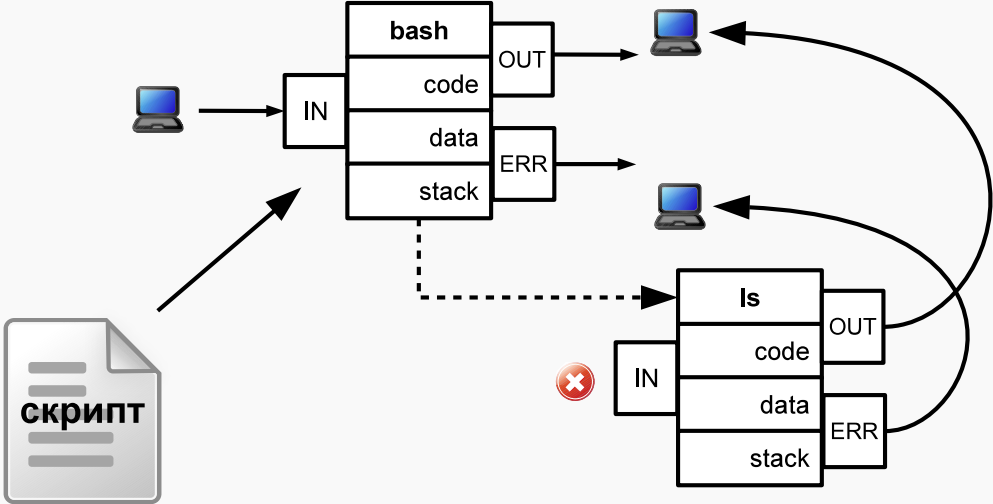
На верхнем примере запущена команда `ls -R`. Не смотря на то, что системный загрузчик открыл для нее поток ввода и по-умолчанию привязал его к терминалу, сама по себе команда не ожидает ничего на свой вход, и соответственно никак не будет его читать или анализировать.

На нижнем примере запуск команды `сорт` производится из командной оболочки (см. далее), при этом загрузчик откроет файлы по умолчанию, а командная оболочка подключит на стандартный ввод файл `students`, на стандартный выход файл `by_name`, а ошибки программы (если они возникнут) игнорирует, путем перенаправления в устройство, которое играет роль пустоты, в котором все, что на него будет записано — будет безвозвратно потеряно.



Интерпретатор команд

- sh (Bourne shell) ksh (Korn shell) csh (C shell)
bash (Bourne-again shell)




30

Всеми командами (программами и приложениями) управляет интерпретатор команд — shell. Это программная оболочка позволяет организовывать выполнение команд, задавать им переменные окружения, связывать потоки ввода вывода у различных команд между собой, а также анализировать коды возврата у команд и использовать простейший язык для написания новых исполняемых команд — скриптов.

Различных интерпретаторов существует достаточно большое количество. Исторически, shell был командной оболочкой, которая использовалась пользователями до появления первого графического интерфейса. В настоящее время в серверных операционных системах shell используется для доступа и настройки систем удаленными от сервера администраторами. Наиболее популярной оболочкой для Unix является bash, хотя не во всех операционных системах он может быть установлен по умолчанию.

При запуске пользовательской программы bash сначала связывает ее потоки с терминалом, из которого он запущен. Затем он, если указаны специальные символы *перенаправления ввода-вывода*, подключит к потокам программы необходимые файлы или свяжет команды между собой, что часто используется для организации *фильтрации потоков*.



Перенаправление потоков stdin(0), stdout(1), stderr(2)

- `> file` — перенаправить stdout в `file`
- `>> file` — добавить stdout к `file`
- `2> file` — перенаправить stderr в `file`
- `2>> file` — добавить stderr к `file`
- `< file` — взять stdin из `file`
- `<< EOF` — записать в stdin из терминала до СИМВОЛОВ «**EOF**»
- `ls | sort` — перенаправить stdout команды `ls` на stdin команды `sort`

31

Специальные символы shell для перенаправления потоков ввода-вывода представлены на слайде. Выполнение команды `ls -lR > /tmp/file` запишет весь вывод команды в файл `/tmp/file`, при этом если возникнут ошибки, они будут выведены на терминал, откуда была запущена команда. При этом содержимое файла `/tmp/file` будет удалено, и информация в файл будет записана с его начала. Если необходимо дописать информацию в конец файла, сохранив его содержимое, то следует воспользоваться командой `ls -lR >> /tmp/file`. Аналогичным образом можно управлять и стандартным вводом.

Если на вход команды необходимо подать файл то можно воспользоваться символом переопределения стандартного ввода «<». Можно при помощи символа «<<» использовать текущий стандартный ввод с терминала для подачи на стандартный ввод команды текста до появления заданного стоп-текста окончания ввода. Например:

```
cat << "хватит" > /home/serge/text
```

Думать и вспоминать.

Команды ночами писать.

Затем на защите страдать.

Лучше отчисленным быть. #спорно

Это терпеть -

хватит

Символ «|» (произносится как конвейер) необходим для связывания стандартного вывода команды слева от конвейера со стандартным вводом команды справа, таким образом осуществляется пересылка данных от одной команды в другую.

Фильтры

ИТМО BT

```
$ cat m* 2>/dev/null | grep -v "^V" | sort
```

Vasya: message1
Я не хочу mkdir pikachu!

Viktor: message2
Кризис какой-то на вашей работе?

Vasya: message3
Злобные тролли вы дяди и тети!

Veronika: message4
Может быть вам обратиться к врачу!?

Злобные тролли вы дяди и тети!
Кризис какой-то на вашей работе?
Может быть вам обратиться к врачу!?
Я не хочу mkdir pikachu!


32

Вместе с операционной системой Unix поставляются разнообразные пользовательские утилиты, которые могут осуществлять фильтрацию данных, который поступает на стандартный вход. Результат фильтрации утилита передает на стандартный выход. Такие программы получили название *фильтров*. На слайде показано использование двух утилит-фильтров — *grep*, которая осуществляет выборку (или отсечение с ключом *-v*, как на слайде) строк совпадающих с шаблоном, заданным *регулярным выражением*, и *sort* — осуществляющих сортировку строк по заданному условию (по умолчанию - в алфавитном порядке).

Строка из команд, представленная на слайде, начинается с команды *cat* (от англ. concatenate)— утилиты, которая объединяет файлы, указанные в командной строке, и выводит их содержимое на стандартный вывод. «*m**» - это подстановка (pathname expansion) интерпретатора shell, который подставит в качестве аргументов команды *cat* все файлы, начинающиеся на букву «*m*», находящиеся в текущей директории. Первая команда после подстановки будет выглядеть так: *cat message1 message2 message3 message4*. Ошибки, возникающие при чтении файлов, игнорируются (*2>/dev/null*).

Стандартный вывод команды *cat* связывается при помощи конвейера со стандартным вводом команды *grep*, которая отбрасывает все строки из полученного потока, начинающиеся на символ «*V*», фильтруя все имена пользователей из файлов сообщений, которые как мы видим из примера, все начинаются на него: *Viktor*, *Vasya*, *Veronika*. Остаются только строки с текстом «сообщений», которые поступают на стандартный вывод *grep*.

Далее стандартный вывод *grep* при помощи конвейера поступает на стандартный ввод команды *sort*, сортируется в алфавитном порядке, и поступает на стандартный вывод *sort*, который связан с терминалом. В результате выведутся строки, показанные на слайде в прямоугольнике в черно-зеленых тонах терминала.



Регулярные выражения

messages

```

Vasya:
Злобные тролли вы дяди и тети!
Viktor:
Кризис какой-то на вашей работе?
Veronika:
Может быть вам обратиться к врачу!?
Vasya:
Я не хочу mkdir pikachu!
                    
```

- Символ — соотв. сам себе
- ^ - начало строки
- \$- конец строки
- . - 1 любой символ

```

$ grep mkdir messages
Я не хочу mkdir pikachu!
$ grep "^V" messages
Vasya:
Viktor:
Veronika:
Vasya:
$ grep "т..$" messages
                    
```

33

Регулярные выражения — де-факто стандарт в выборке строк на совпадение в Unix и разработке программ. Регулярное выражение определяет шаблон, которому должна соответствовать строка в тексте, в котором проводится поиск на совпадение. На слайде приведены простейшие регулярные выражения, более глубокое их изучение предусмотрено на старших курсах.

Простейший случай регулярного выражения — это последовательность символов, которые соответствуют сами себе. В первом примере выводится все совпадения строк файлов с шаблоном (регулярным выражением) `mkdir`. Как видно из примера, этому шаблону соответствует единственная строка в файле `messages`.

Символ «`^`» определяет начало строки. Регулярное выражение «`^V`» будет соответствовать всем строкам у которых самым первым символом в строке будет латинская заглавная буква `V`. Таких строк во втором примере в файле находится четыре. Символ «`$`» означает конец строки. Его можно использовать для поиска строк, имеющих заданное регулярным выражением окончание. Например можно вывести все строки, оканчивающиеся на восклицательный знак.

Регулярное выражение «`.`» соответствует любому символу. В третьем примере будут выведены две строки в которых есть символ «`т`» и два любых символа в конце строки. Регулярное выражение «`.*`» — соответствует любой последовательности символов. Пример - «`V.*а`» - выведет три строки (какие?).

Для использования в поиске символов, которые встречаются в регулярном выражении их необходимо экранировать символом «`\`», например, если необходимо найти в тексте символ «`$`» его необходимо экранировать («`\$`»), чтобы он не был воспринят как символ конца строки. Экранирования от подстановок shell выполняется при помощи символов «`"`».

Возможности регулярных выражений позволяют использовать различные шаблоны, их применение, конечно, не ограничивается приведенными примерами. Можно, например, указывать необходимое повторение символов, диапазон символов, их группировку в определенной последовательности, и т. д.



Команды

Команда	Назначение и синтаксис
mkdir	mkdir [-m mode] [-p] dir...
echo	echo [string]...
cat	cat [-n] [file...] [-]
touch	touch [-am]... file...
ls	ls [options] [file/dir]...
pwd	pwd
cd	cd [argument]
more	more [file...]
cp	cp [options] SOURCE ... DEST
rm	rm [options] [file/dir]
rmdir	rmdir [dir]
mv	mv [-fi] SOURCE ... DEST
head	head [-num] [file...]
tail	tail [-/+num] [-bcl] [file...]
sort	sort [-unr] [-k num] [file...]
grep	grep [-v] regexp [file...]
wc	wc [-c -m] [-lw] [file...]

34

На слайде представлен список команд, которые могут потребоваться при выполнении лабораторной работы №1. Перечислим назначение этих команд:

- **mkdir** — создание директории
- **echo** — вывод символов на стандартный вывод
- **cat** — слияние и вывод файлов на стандартный вывод
- **touch** — создание (если файл отсутствует) или изменение времени модификации файла
- **ls** — вывод списка файлов в директории
- **pwd** — вывод текущей директории в стандартный вывод
- **cd** — смена текущей директории на указанную
- **more** — постраничная распечатка файла или стандартного ввода
- **cp** — копирование файлов и директорий
- **rm** — удаление файлов
- **rmdir** — удаление пустых директорий
- **mv** — перемещение файлов
- **head** — вывод указанного количества строк с начала файла или stdin
- **tail** — вывод указанного количества строк с конца файлов или stdin
- **sort** — сортировка строк файла или стандартного ввода
- **grep** — выборка строк по шаблону
- **wc** — подсчет количества символов, строк, слов в файле или стандартном вводе




Как работают ЭВМ?

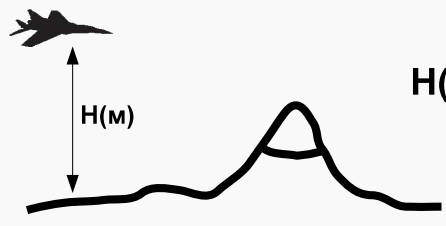
3



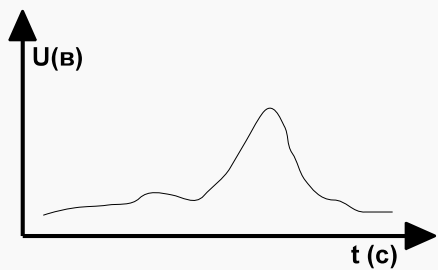
35



Аналоговые ЭВМ

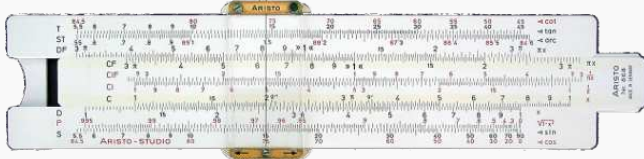


$H(m) \sim U(B)$



Для определенного класса задач

- Высокое быстродействие
- Меньшая погрешность вычислений




36

Первым вычислительным устройством в истории человечества являлся абак или счеты. В разных частях мира они имели свои небольшие особенности, но выглядели они практически одинаково. Такие устройства широко использовались для, как бы мы сейчас сказали, ведения бизнеса. Торговцы того времени подсчитывали количество продаваемого или приобретаемого товара.

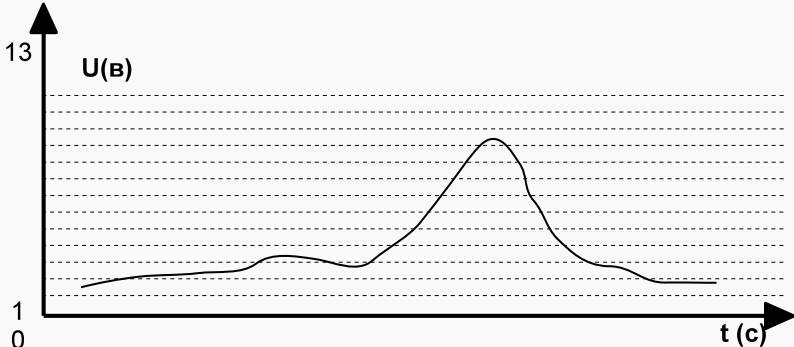
Первая счетная машина, позволявшая производить умножение и деление также легко, как сложение и вычитание, была изобретена в Германии в 1673 году Готфридом Вильгельмом Лейбницем, и называлась «Калькулятор Лейбница». По образцу двенадцати-разрядной счетной машины Лейбница в 1708 году профессор Вагнер и мастер Левин создали шестнадцати-разрядную счетную машину.

С развитием техники и технологий людьми были сконструированы приборы, которые позволяли определять различные физические величины, такие как скорость, ускорение, расстояния. Например, если мы хотим измерить высоту полета самолета, то мы можем измерить давление атмосферы или послать электромагнитный импульс к земле и посчитать, когда придет отраженный сигнал. Но сами физические значения, такие как высота, сложно использовать в дальнейших вычислениях в их естественном виде, и люди придумали заменять их аналогичными величинам или просто аналогами. В современной вычислительной технике, в качестве аналоговых величин используются значения напряжения.

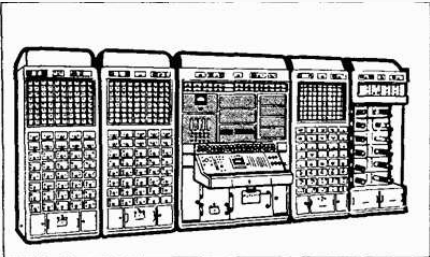
При этом высоте, например, может ставится в соответствие напряжение от датчика высоты. Аналоговая величина всегда непрерывна, и ее поведение полностью копирует поведение соответствующей физической величины. Теперь с этой аналоговой величиной можно проводить вычисления - складывать, вычитать, дифференцировать. Например, при изменении высоты, можно рассчитать вертикальную скорость относительно поверхности. Аналоговые вычисления производится аналоговыми ЭВМ. К сожалению, у таких ЭВМ есть недостатки.



Аналоговые ЭВМ




- Точность представления данных
- Достаточно большие габариты



37

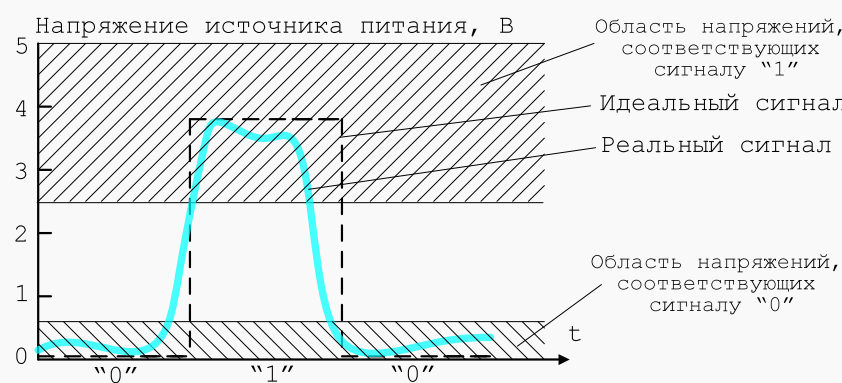
Основной из них - это конечно большие габариты таких вычислительных машин. Кроме того, не всегда просто преобразовать физическую величину в аналог и обратно. Например, человек может различать лишь ограниченное количество цветов. Соответственно, если поставить рядом два близких цвета, есть люди, которые заметят разницу, а есть те, кто не заметит. В аналоговых вычислениях происходит то же самое. Чтобы различить для летящего самолета высоту в 200м или 201м, должны быть достаточно точные приборы, которые будут реагировать на изменение сигнала.

Аналоговые системы до сих пор широко используются в системах реального времени, в различных измерительных устройствах, в информационно-управляющих системах. Потому что у них очень высокое быстродействие для определенного класса задач и удовлетворительная погрешность вычислений. Это происходит за счет того, что аналоговая величина – непрерывна. Операции интегрирования и дифференцирования аналоговые вычислительные машины выполняются существенно быстрее, чем цифровые.



Цифровые ЭВМ

- Представления информации с помощью только двух дискретных величин — 0 и 1



Область напряжений, соответствующих сигналу "1"

Идеальный сигнал

Реальный сигнал

Область напряжений, соответствующих сигналу "0"

t

38

Вопрос о недостаточной точности вычисления аналоговых ЭВМ привел к созданию цифровых ЭВМ. Решили, что вместо сложного устройства, которое точно различает величину амплитуды напряжения в 0,001 вольта, проще использовать сигнал, у которого будет всего два состояния - либо 0, либо 1. Это существенно упростило конструкцию вычислительных машин, приведя к созданию цифровых вычислительных систем.

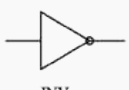
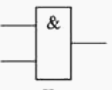
На слайде показан реальный сигнал, снятый осциллографом, который представлен двумя уровнями, 0 и 1. Следует помнить, что при передаче внутри ЭВМ форма сигнала может искажаться. Если сигнал был сильно искажен, то могут появляться ошибки. Такое происходит очень часто, но человек обычно этого не видит, так как существуют специальные программные и аппаратные решения, которые эти ошибки минимизируют. Часто ошибки появляются при работе с памятью и при передаче данных.


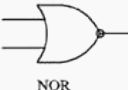
Отдельные импульсы в цифровых вычислительных машинах объединяются в группы, позволяя закодировать информацию кодовой посылкой в виде одного или нескольких чисел. Цифровое представление существенно упрощает необходимые элементы для построения вычислительной машины.

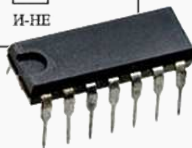


Функциональные элементы ЭВМ

• Логические элементы

ГОСТ		ANSI	
	Буфер		BUF
	Инвертор		INV
	И		AND
	И-НЕ		NAND

ГОСТ		ANSI	
	ИЛИ		OR
	ИЛИ-НЕ		NOR
	Исключающее ИЛИ		XOR
	Исключающее ИЛИ-НЕ		XNOR



И			
X1	X2	Y	Ȳ
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

ИЛИ			
X1	X2	Y	Ȳ
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Искл. ИЛИ			
X1	X2	Y	Ȳ
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

39

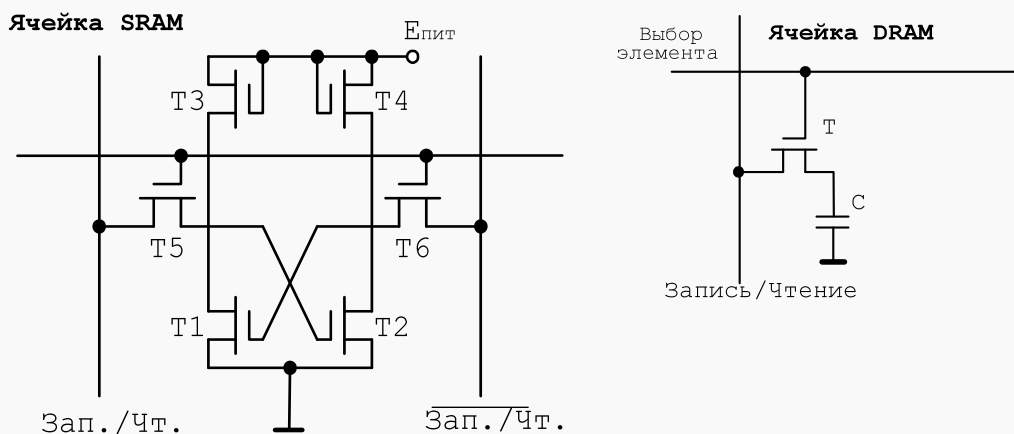
Основы функционирования ЭВМ необходимо изучать с уровня элементной базы. На самом нижнем логическом уровне компьютеры состоят из логических элементов, которые выполняют логические функции, основные из которых приведены на слайде. У каждого логического элемента есть один или несколько входов и один выход. На вход подаются значения (0 или 1), внутри элемента выполняется функция, результат которой попадает на выход. Например, рассмотрим элемент, который выполняет простейшую операцию И. У элемента есть два входа, на вход подаются значения 1(истина) или 0(ложь). Если на оба входа была подана истина – результатом должна быть тоже истина. Информацию о выполняемой функции, о соответствии значений входов и выхода обычно размещают в так называемых таблицах истинности.

На слайде представлено графическое изображение элементов на электрических схемах в стандарте ГОСТ (принят в СССР, используется для электрических схем в РФ) и стандарте ANSI (американский стандарт). Для функций И, ИЛИ и исключающее ИЛИ (обычно используется для сравнения) приведены таблицы истинности.

Простейшая микросхема объединяет несколько логических элементов и обычно содержит входы и выходы этих элементов, а также входы для подачи напряжения питания.

 **Функциональные элементы ЭВМ**

• Элементы хранения (DRAM/SRAM)



40

Для хранения информации и данных в ЭВМ используется оперативная память, информация в которой сохраняется, пока ЭВМ включена. В современных вычислительных системах существует два типа оперативной памяти: динамическая память (DRAM) и статическая память (SRAM).

В DRAM – информация хранится на емкости, которая находится в цепи зарядов транзистора. Ячейка памяти состоит из одного транзистора и одного конденсатора (на рисунке справа). Недостаток такой памяти: емкость постепенно разряжается, соответственно информация теряется. Для того, что бы поддерживать заряд на емкости предусмотрено устройство регенерации памяти: компьютер раз в определенное время (64 мс) сканирует всю имеющуюся оперативную память и считывает каждую ячейку, возобновляя заряд на конденсаторе. Достоинством DRAM является ее дешевизна на единицу хранимой информации, т. к. для одного бита используется всего один транзистор.

Статическая память SRAM не требует регенерации за счет такой конструкции ячейки, где шесть транзисторов объединены в схему, которая самоподдерживает свое состояние за счет обратных цепей связи. Однако цена такой памяти будет больше, кроме того, ячейки данного типа являются более быстродействующими. На их основе строится память, которая обычно встроена в процессор. К такой памяти относятся, например, кэш память различных уровней.

Функциональные элементы ЭВМ

- Элементы хранения (триггеры, регистры)

D-триггер

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	1	0	0	0	0	0	1	0	1	1

41


С точки зрения схемы ЭВМ, построенной на логических элементах, в качестве хранения информации используются триггеры. *Триггер* = «защелка», с помощью которой можно хранить и управлять информацией.

Например, D-триггер представляет собой ячейку информации, хранящую один бит информации. Простейшая схема D-триггера изображена слева сверху на слайде. У триггера есть 2 входа. Один из них D – информационный, на него подается значение, которое необходимо сохранить, и C – вход синхронизации, который определяет, будет ли значение на информационном входе записано в триггер.

Когда C=0, значения в триггер не сохраняются, и значение выхода Q будет неизменным, т. е. триггер хранит ранее записанную информацию. Если C=1, то триггер возьмет значение с информационного входа D, и сохранит в своей схеме хранения. Выход Q станет равным входу D. При снятии единичного сигнала со входа C триггер защелкнется, и значение на входе D не будет влиять на сигнал на выходе триггера Q.

Триггеры объединяются в регистры. Справа на слайде показан регистр, который хранит 4 бита информации. Имеется общий вход C, который для всех триггеров один. Когда на него подается 1, одновременно вся информация поместится в ячейки памяти. И будет там храниться неопределенно долг, до тех пор, пока вы не решите заменить ее любой другой информацией.

Таким образом, регистры образуют промежуточное хранение информации. Регистры мы будем обозначать прямоугольниками, как представлено внизу слева на слайде. Данный регистр 16ти-разрядный, т.е. он содержит 2 байта или 2 слова.

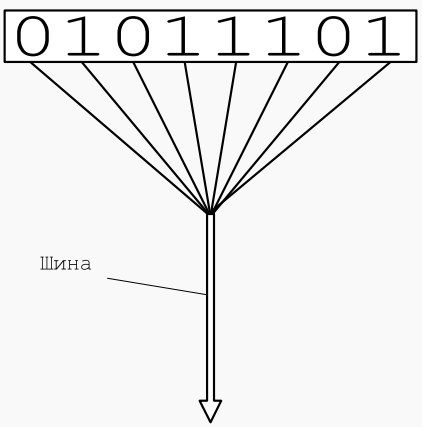


Функциональные элементы ЭВМ

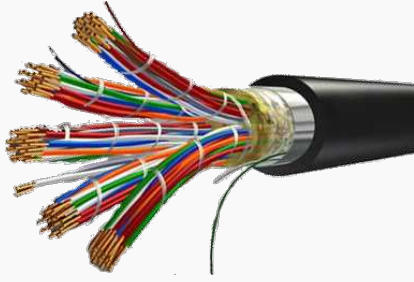
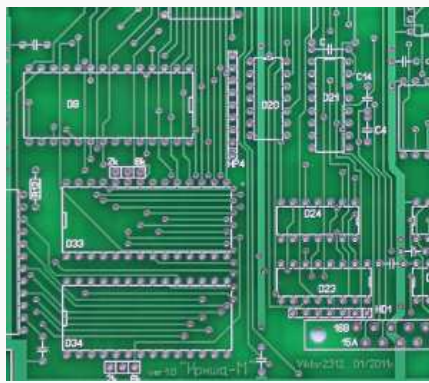
- Провода, шины

Регистр
(источник информации)

01011101



Шина

42

В ЭВМ информацию нужно не только хранить, но и передавать от одного регистра к другому. Передача информации осуществляется с помощью линий передачи (или проводов), которые объединяются в *шины*. Для четырехразрядного регистра, приведенного на предыдущем слайде потребуется 4 информационных (D0-D3), один управляющий (С) и одна общая («земля») линия передачи данных, по которой ток от всех проводов течет в противоположном информационным сигналам направлении. Таким образом, для передачи 4-х бит информации необходимо, как минимум, 6-ть проводов.

Обычно, внутри ЭВМ шины выглядят могут выглядеть как представлено на слайде справа.

При передаче информации на дальние расстояния появляются ошибки передачи. Это связано с тем, что у проводов образуется электромагнитные поля, взаимодействующие друг с другом. Кроме того, широкое использование электрических приборов наводит на линии связи большое количество внешних помех. Для минимизации помех отказываются от использования общего провода и скручивают прямой и обратный провода, по которым идет сигнал, в *витую пару*. В витой паре наведенные помехи взаимно уничтожаются, т. к. помеха наводится одновременно на прямой и обратный проводник сигнала.

На схемах шина будет обозначаться широким проводником.



Функциональные элементы ЭВМ

- **Вентили**

Регистр
(источник информации)

0 1 0 1 1 1 0 1

Управляющий сигнал



Шины

Регистр
(приемник информации)

0 1 0 1 1 1 0 1

Вентиль (И)		
Упр.	Инф.	Вых.
0	0	0
0	1	0
1	0	0
1	1	1



43

Передачу информации обычно нужно осуществлять не непрерывно, а в строго заданное (тактовым генератором) время. Для этого предусмотрено логическое устройство, называемое *вентилем*, которое функционирует по принципу хорошо известного нам крана. Вентиль может быть в открытом состоянии, и тогда информация с выхода одного регистра поступает на другой, или в закрытом, тогда передача информации блокирована. Функцию вентилей в единичной логике осуществляет функция «И».

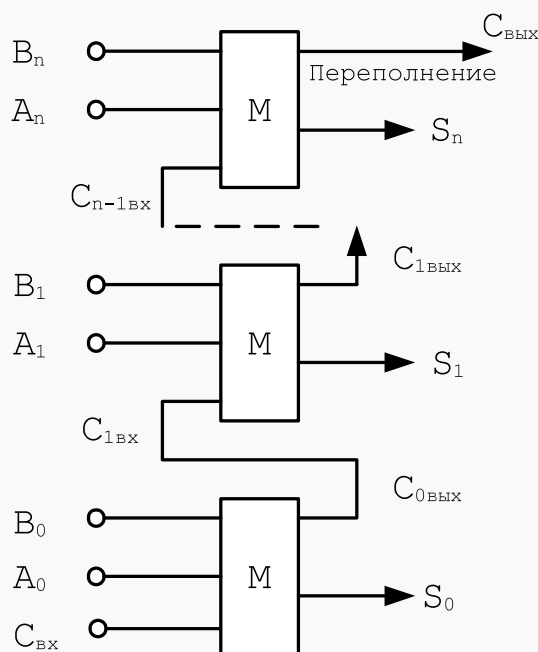
У вентилей есть управляющий и информационный сигналы. На его выходе единица будет только в том случае, если информационный и управляющий сигналы будут равны единице (кран открыт и течет вода). Если кран открыт, но вода не течет, на выходе будет 0. В домашнем кране так бывает, когда летом отключили горячую воду. Таким образом управляющий сигнал разрешает или запрещает передачу информации.

На наших функциональных схемах мы будем обозначать вентиль ромбиком.

ИТМО ВТ **Функциональные элементы ЭВМ**

- Сумматоры (входят в АЛУ)

SUM				
Свх	Ai	Bi	Свых	Si
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



44

Вычислительная машина не только передает и хранит данные. Она должна еще выполнять с ними различные арифметические или логические операции. Например, одни из простейших операций – сложение и вычитание, осуществляются с помощью сумматора (входит в АЛУ).

В общем случае, для сложения 2х разрядов многоразрядного двоичного числа у логического элемента должны присутствовать 3 входа — два для значений разрядов слагаемых и один, дополнительный, для входящего переноса из предыдущего разряда, и два выхода — для значения суммы и выходящего переноса. Перенос передается следующему разряду после суммирования предыдущих (аналогично сложению в столбик).

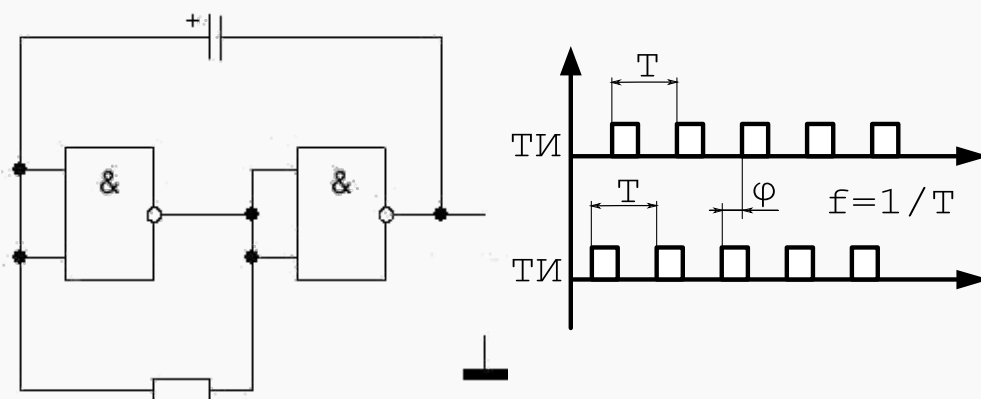
Обратите внимание на схему. На ней как раз изображено, что есть числа A и B и входящий перенос. Соответственно, результат сложения обозначен S, а поразрядный перенос C_i мы должны передавать с выхода одного элемента на вход следующему. Т.е. сколько разрядов, столько и должно быть передач поразрядных переносов. Если присутствует поразрядный перенос из старшего разряда, то фиксируется выход за границы разрядной сетки или *переполнение*.

Таким образом, одной из проблем сумматоров является то, что при сложении двух 16-ти разрядных чисел может получиться 17ти-разрядное число. Другой проблемой, является то, что при сложении значение каждого следующего разряда зависит от предыдущего, это приводит к снижению быстродействия. Если быстрота срабатывания одного элемента равна V_s , то скорость сложения последовательного поразрядного сумматора для 16-ти разрядов будет $16 \cdot V_s$.

Функциональные элементы ЭВМ

ИТМО ВТ

- Тактовые генераторы



45

Тактовые генераторы являются «сердцем» ЭВМ. Подобно человеческому сердцу они задают ритм работы вычислительной машины в целом. Все пересылки данных, арифметические или логические операции могут происходить только в строго заданное время, определяемое размером такта генератора. Одни операции могут выполняться по фронту сигнала ТК, другие по значению, некоторые по спаду сигнала. К концу такта все операции должны быть завершены. Использование разных частей импульса ТГ для обработки информации связано с тем, что в элементах ЭВМ производительность отдельных элементов имеет конечное значение, обусловленное переходными процессами в них. Поэтому, например, для суммирования может использоваться фронт импульса, а для записи значения суммы в регистр — спад.

Тактовые генераторы характеризуются частотой, длительностью периода импульса и фазовым сдвигом начала импульса. Если в ЭВМ используется несколько тактовых генераторов (а так обычно и есть) то они должны быть синхронизированы между собой, т. е. сдвиг фаз должен быть равен нулю.

Частота современных тактовых генераторов процессора = 3ГГц. Длина импульса при этом составляет около 0,3 нс.



Рассмотрим принципы функционирования простейшей ЭВМ — калькулятора. Он состоит из двух регистров — X и Y, хранящих результаты ввода пользователя и промежуточных вычислений, АЛУ, которая может выполнять простейшие арифметические и логические операции, шин и управляющих вентилях, осуществляющих передачу данных между функциональными блоками калькулятора, устройства управления (УУ), клавиатуры и дисплея.

Дисплей постоянно отображает содержимое регистра X (проследите о шине путь информации от X к дисплею, убедитесь, что вентили на этом пути отсутствуют). Клавиатура передает значение нажатой клавиши на вентиль U5, каждое нажатие на клавишу запускает УУ, которое в зависимости от текущего состояния ЭВМ формирует последовательность импульсов для выполнения требуемой операции, которая называется *циклом* импульсов. Каждая группа импульсов выдается последовательно, в моменты, совпадающие с импульсами тактового генератора.

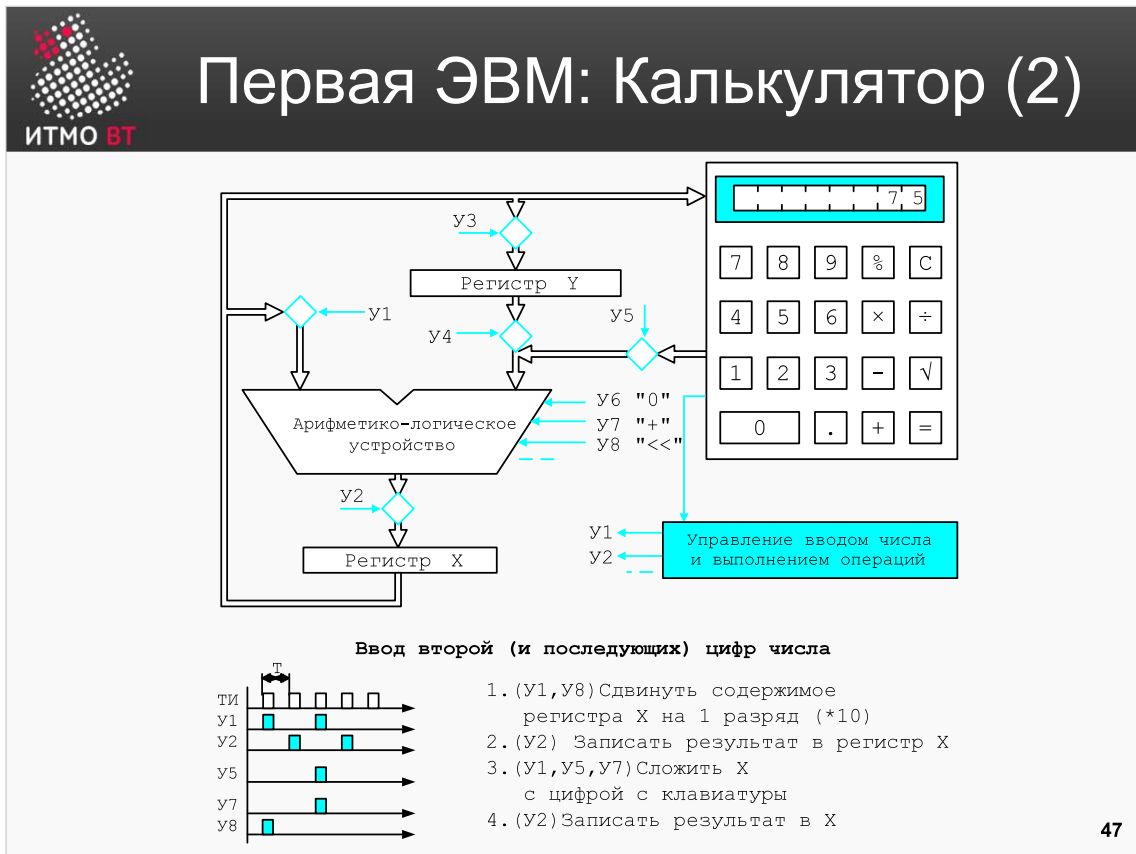
Предположим пользователь вводит первую цифру необходимого ему числа (7). Так как это новая операция, УУ, после своей активации нажатием кнопки 7, выдаст последовательность управляющих импульсов для первой цифры числа.

В первую очередь необходимо сохранить предыдущее значение регистра X в регистре Y. Для этого должен быть открыт вентиль, управляющий записью в регистр Y. Он открывается управляющим сигналом U3.

После этого необходимо обнулить регистр X, подготовив его для новой цифры числа, которая была введена с клавиатуры. Для этого должны быть закрыты все вентили, кроме U2 — который осуществляет передачу данных из АЛУ в регистр Y и U6 — который сформирует в АЛУ значение 0.

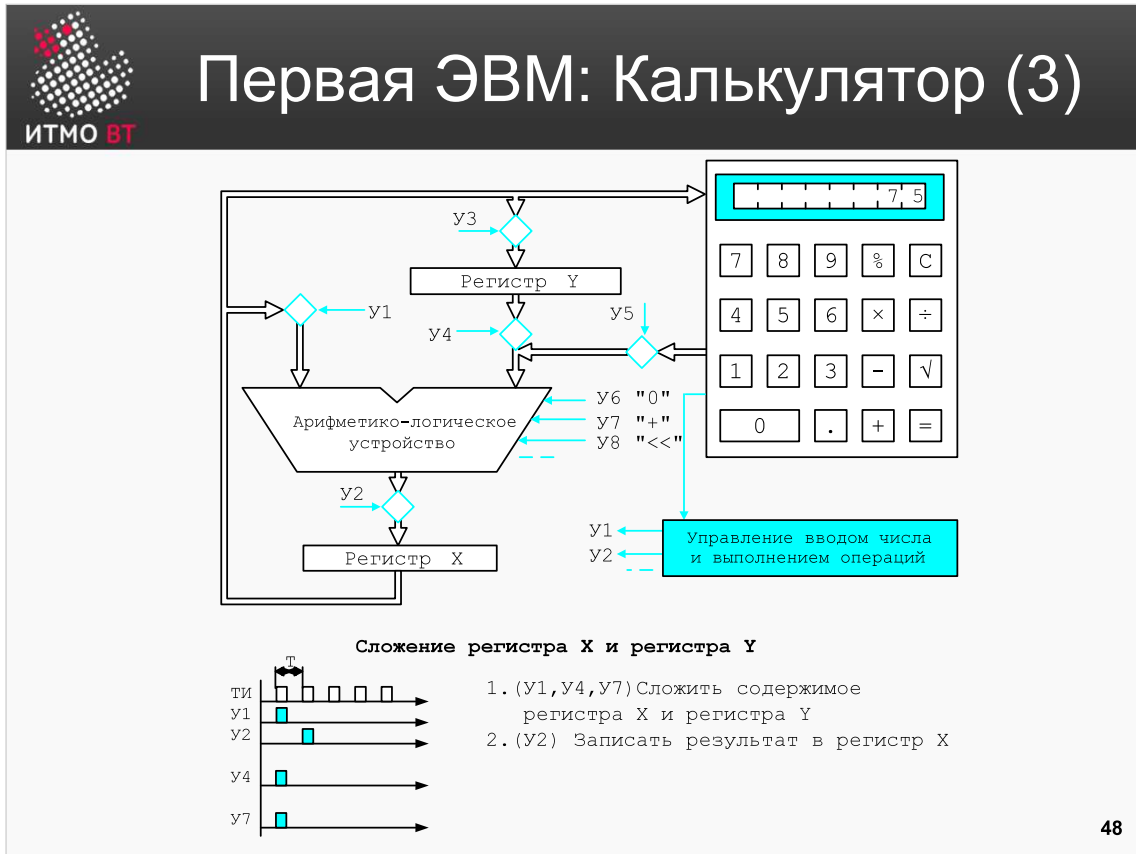
Далее необходимо сложить значение 0 с цифрой с клавиатуры. Для этого содержимое регистра X поступает на правый вход АЛУ (U1), цифра с клавиатуры на правый вход АЛУ (U5), и выбирается операция сложения (U7).

В конце цикла необходимо передать результат сложения в регистр X (U2), отобразив его, при этом, на дисплее.



Для ввода второй и последующих цифр необходимо осуществлять поразрядный сдвиг регистра X после каждой введенной цифры и складывать введенную цифру со сдвинутым содержимым регистра X. Разберем этот цикл по тактам:

1. Содержимое регистра X через вентиль, управляемый U1 подается на левый вход АЛУ, при этом вентили U4, который управляет передачей из регистра Y и U5 (ввод с клавиатуры), должны быть закрыты, при этом на правый вход АЛУ подается 0. Управляющий сигнал U8 вызовет сдвиг информации, которая поступает на левый вход АЛУ. В двоично-десятичной системе счисления (в которой обычно выполняют вычисления калькуляторы) сдвигу на один десятичный разряд соответствует умножение на 10.
2. Результат сдвига записывается (управляющий сигнал U2) в регистр X.
3. Затем необходимо сложить результат сдвига с новой цифрой с клавиатуры. Для этого содержимое регистра X поступает на правый вход АЛУ (U1), цифра с клавиатуры на правый вход АЛУ (U5), и выбирается операция сложения (U7).
4. Результат сдвига и сложения записывается (управляющий сигнал U2) в регистр X.



Когда пользователь нажимает кнопку «+» или «=», в зависимости является ли это сложение промежуточной или конечной операцией, необходимо сложить содержимое регистра X и регистра Y. Для этого:

1. Содержимое регистра X подается на левый вход АЛУ (Y1), содержимое регистра Y подается на правый вход АЛУ (Y4) и выполняется операция сложения (Y7). Все остальные вентили при этом закрыты (в первую очередь должен быть закрыт Y5).

2. Результат записывается в регистр X и показывается на экране калькулятора.

 Do you know,
how to drive that thing?


- Ты умеешь управлять вертолетом?
- Еще нет.


- Тэнк, программу управления, пожалуйста




- Пойдем!



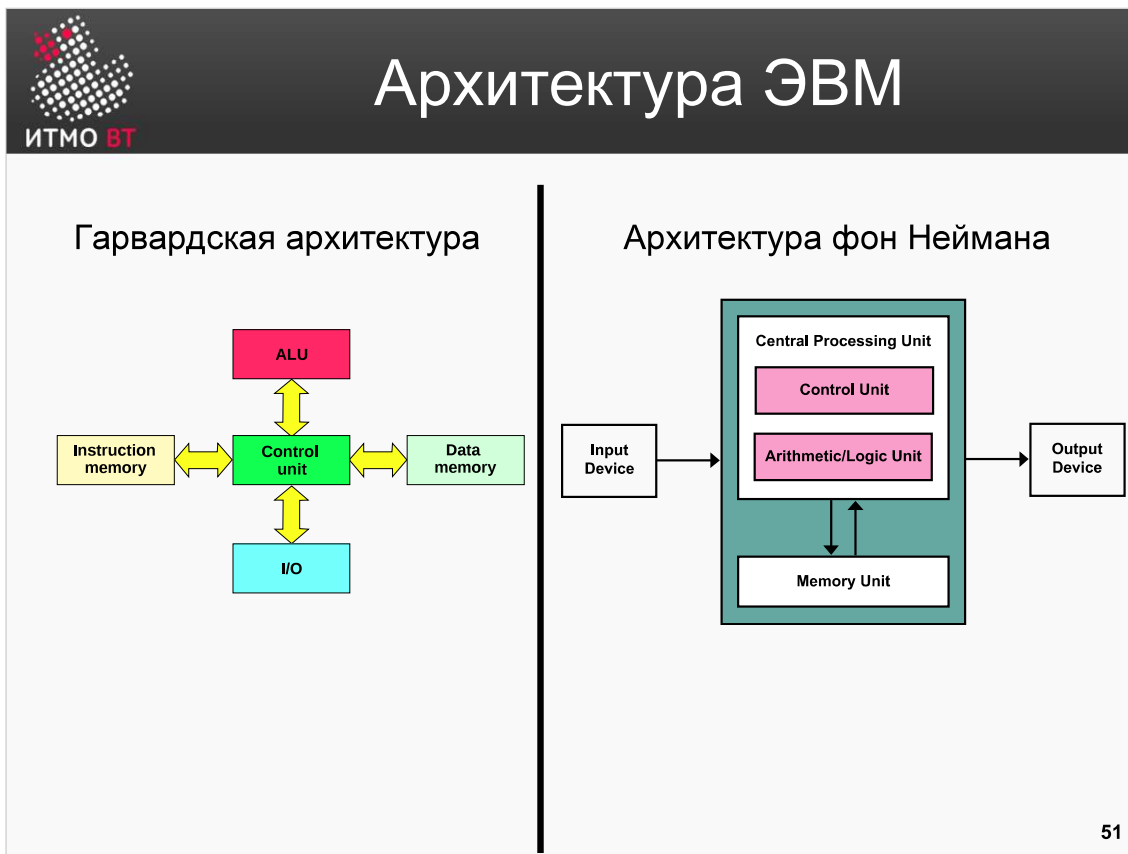
49



Общие сведения о БЭВМ

4



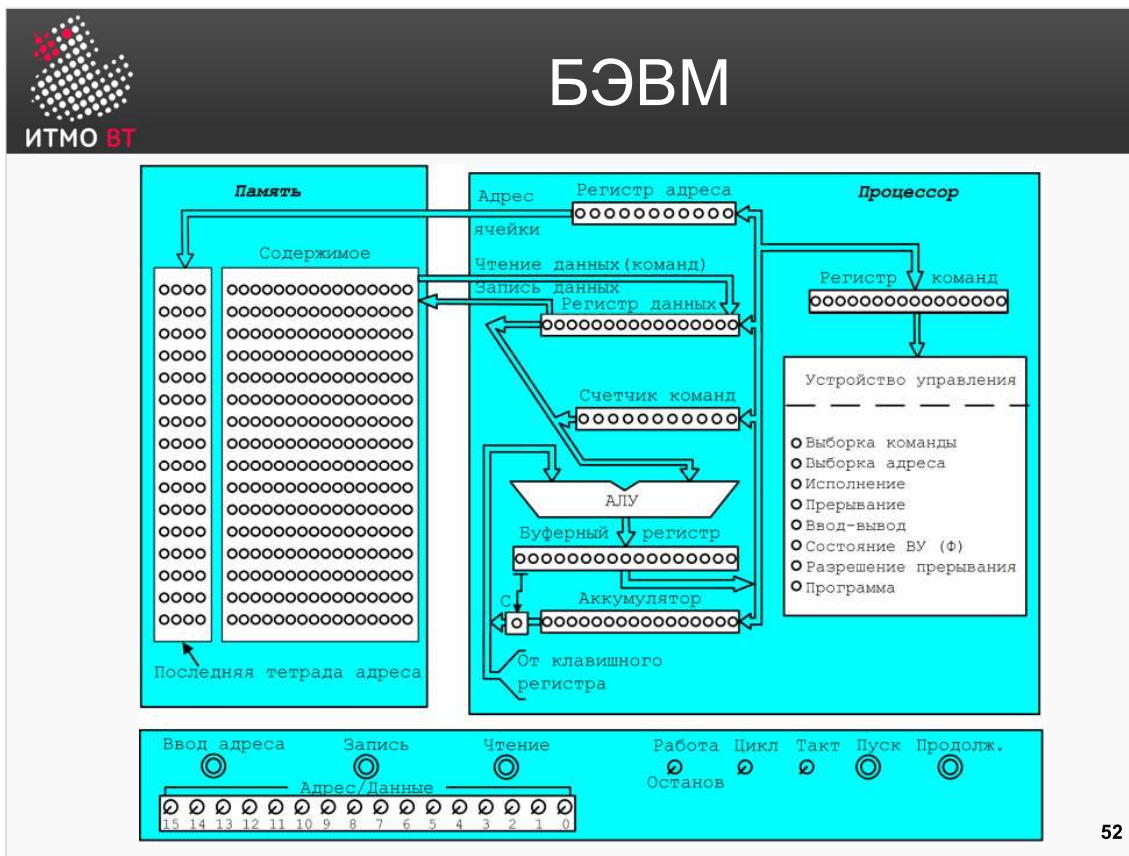


В разработке вычислительных систем исторически сложились два, отличающихся друг от друга, подхода к их построению, называемые, архитектура фон Неймана и Гарвардская архитектура. Основным отличием между архитектурами является способ работы с памятью. В настоящее время в чистом виде архитектуры встречаются редко, обычно используется их комбинация в разных функциональных блоках ЭВМ.

В Гарвардской архитектуре центральным устройством является Control Unit – управляющее устройство в ЭВМ. Все остальные устройства ЭВМ подключены к управляющему устройству и взаимодействуют через него. Память для команд (instruction memory), данных (data memory) и устройств ввода вывода физически отделена друг от друга. Как было указано выше, в чистом виде такая архитектура уже не используется, но идея физического разделения кода и данных оказалась очень полезной для развития развития функциональных подсистем ЭВМ и операционных систем. Используя такой подход нельзя нарушить логику программы, попытавшись исполнить данные или трактовать инструкции как данные программы.

В противоположность Гарвардской, архитектура фон Неймана содержит все в общей памяти. Нельзя определить, что находится в ячейке, данные или инструкции, без дополнительного анализа кода самой программы. При этом устройство обращения к памяти едино для данных и инструкций, что упрощает конструкцию ЭВМ. Отдельно выделены устройства ввода вывода, которые являются внешними по отношению к процессору — объединению памяти, АЛУ и управляющего устройства.

Базовая ЭВМ – характерный пример архитектуры фон Неймана.



52

БЭВМ включает в себя нескольких функциональных блоков и регистров:

- Память – состоит из 2048 ячеек. Каждая ячейка занимает 16 разрядов. Для обращения к памяти существует два регистра: 11-разрядный *регистр адреса* (РА), в который нужно поместить адрес прежде чем обратиться к памяти; 16-разрядный *регистр данных* (РД), который предназначен для чтения или записи данных в/из ячеек памяти. Чтение данных и запись данных реализуется по двум отдельным шинам, которые подаются на ячейку памяти.
- 11-разрядный *счетчик команд* (СК). Всегда хранит в себе адрес следующей исполняемой команды.
- АЛУ и связанный с АЛУ *буферный регистр* (БР). Буферный регистр 17-разрядный, в старшем разряде хранится бит переполнения. АЛУ может выполнять несколько операций: сложение, логическое умножение, инверсия и прибавление единицы. Одна из этих операций – сложение, в котором возможно заполнение бита переполнения. Выход из АЛУ через буферный регистр подключается к шине, по которой информация может быть передана в любой другой регистр БЭВМ.
- *Регистр команд* (РК) – используется для декодирования операций, происходящих в БЭВМ и иногда, для временного хранения данных микропрограммы.
- *Аккумулятор* (А). БЭВМ относится к ЭВМ, которые называются ЭВМ аккумуляторного типа, где все вычисления с данными производятся через него. В АЛУ всегда один из операндов всегда берется из аккумулятора, а второй из любого другого регистра.
- Содержимое регистра команд используется *устройством управления* (УУ). Оно предназначено для того, чтобы исполнять машинные команды при помощи *микроопераций* - т. е. атомарных операций по вычислению в АЛУ и пересылки данных между регистрами и памятью.
- 16-разрядный *клавишный регистр* (КР) – находится в составе *пульты оператора* ЭВМ и предназначен для ввода адреса программы, кодов программы и данных, запуска программы на выполнение и управления режимами работы БЭВМ.
- 12-ти разрядный регистр состояния (РС — на модели не показан) хранит биты управляющие работой БЭВМ (работа, прерывание и пр.) и копии признаков результата.



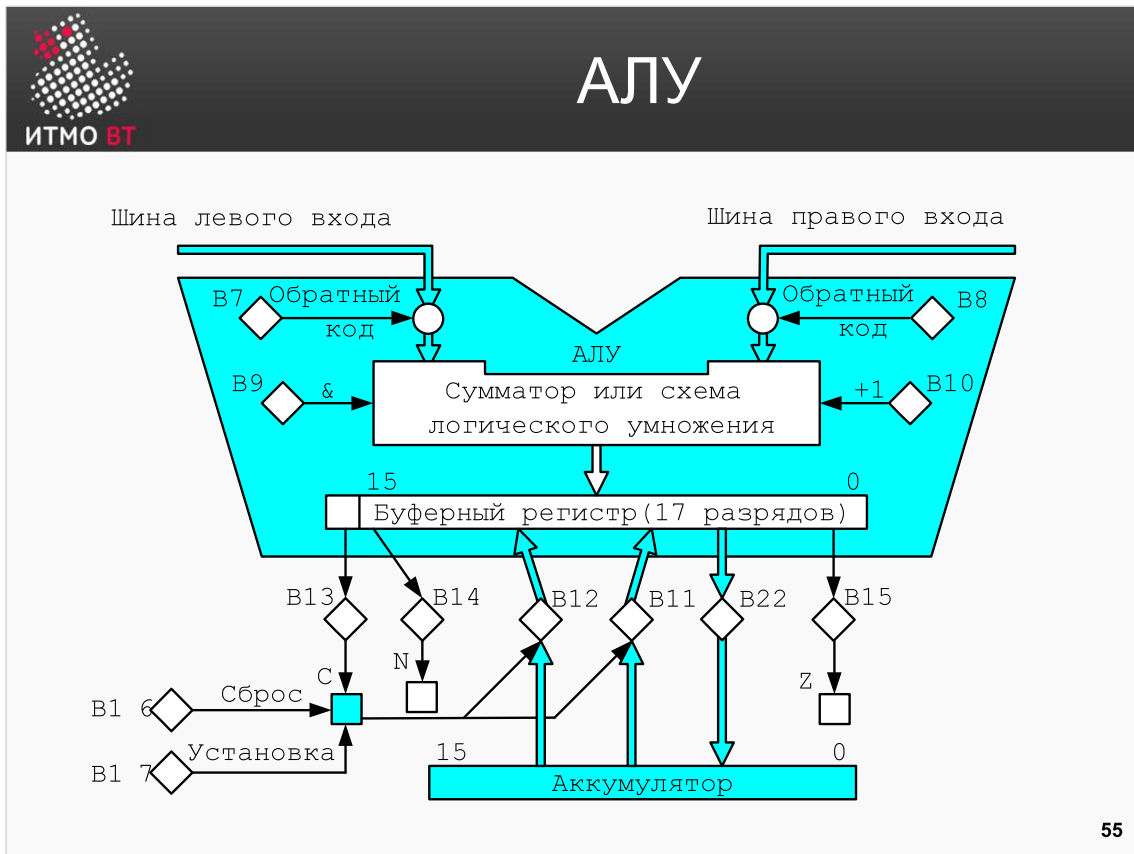
Отступление: Дешифратор

Адрес			Строка							
A ₂	A ₁	A ₀	L ₇	L ₆	L ₅	L ₄	L ₃	L ₂	L ₁	L ₀
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

53

Структурная схема памяти в простейшем случае состоит из дешифратора адреса и элементов памяти. Рассмотрим как работает дешифратор.

Дешифратор – это устройство, на вход которого подается код числа, а на выходе выбирается только одна выходная линия, номер которой соответствует коду на входе дешифратора. На слайде приведено схематическое изображение дешифратора и таблица истинности им реализуемая. В схемах памяти дешифратор используется для выбора строки памяти, соответствующую нужному адресу (коду) на входе дешифратора.



Рассмотрим структуру и состав арифметико-логического устройства (АЛУ) БЭВМ.

АЛУ имеет два входа – левый и правый. На каждом входе АЛУ расположен инвертор — схема побитной инверсии поступающих на вход сигналов. Эти схемы позволяют получить обратный код двоичного числа поступающих на заданный вход АЛУ. В обратном коде (рассмотренном далее) происходит побитовая замена единиц на нули и наоборот.

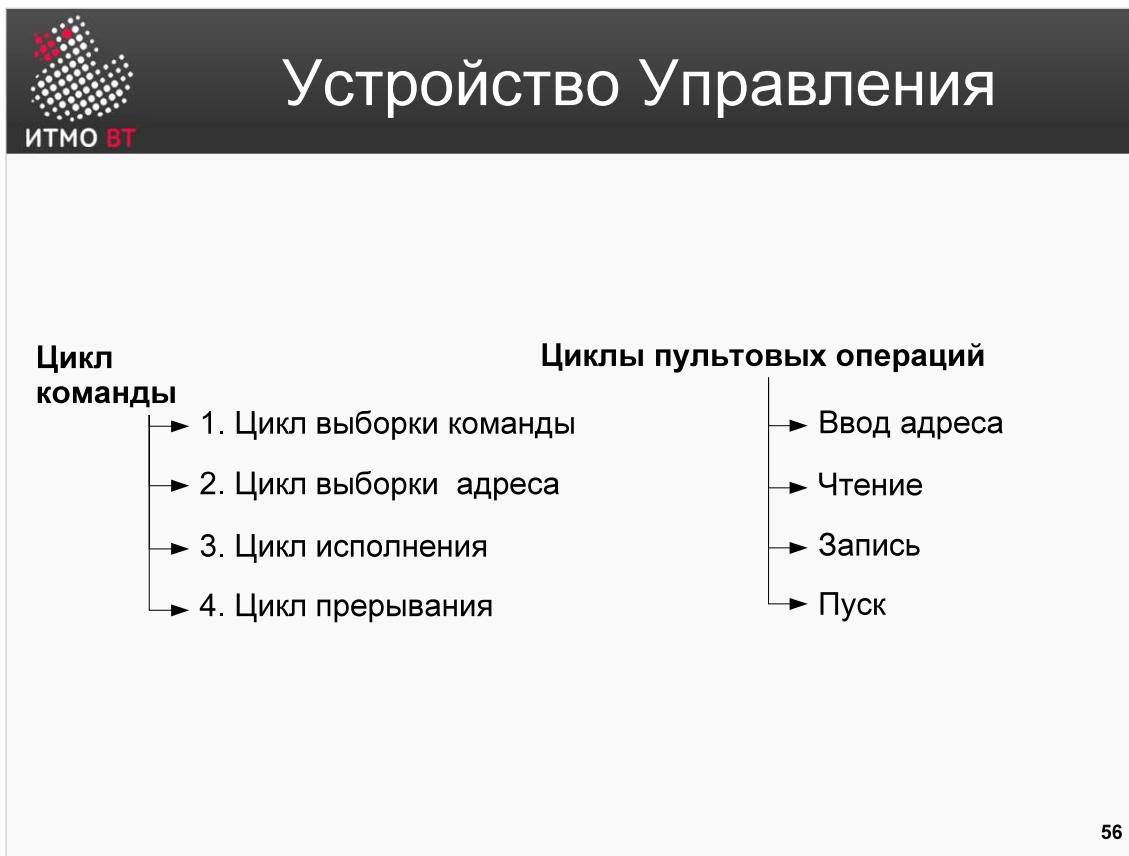
Отдельный ventиль предназначен для выполнения в АЛУ операции *инкремента* (увеличение на 1). Это сделано путем подачи единичного сигнала на вход входного переноса сумматора (+1 или V10 на слайде). Соответственно, инкремент может только осуществляться во время операции сложения в АЛУ.

Основная часть АЛУ состоит из схемы сумматора или логического умножения. Выбор операции производится при помощи соответствующего ventиля (V9). Если на его входе 1, то будет выполняться поразрядное логическое умножение левого и правого входа АЛУ, в противоположном — сложение операндов с учетом входного переноса и формирование выходного переноса сумматора в 17-ом разряде буферного регистра. Схемы сумматора или логического умножения непосредственно подключены к буферному регистру. БР содержит в себе результат операции (разряды с номера 0 по 15-й) и бит переноса, который помещается в разряд с номером 16 (17-тый по счету, т. к. нумерация начинается с нуля).

16 битов результата операции из БР можно переписать в аккумулятор (V22). Для выполнения сдвигов данных есть еще дополнительные шины, управляемые ventилями V11 и V12), которые осуществляют асимметричную передачу А обратно в БР, для обеспечения возможности выполнения сдвигов на один разряд.. В свою очередь, сдвиг влево или вправо на разряд может быть использован в БЭВМ, чтобы выполнить операции умножения или деления на 2 соответственно.

Кроме всего прочего АЛУ содержит однобитовые признаки результата операции, которые характеризуют результат операции, проведенной в АЛУ, и ventили, которые ими управляют:

1. Бит переноса C (carry bit)
2. Бит признака отрицательного числа (N - negative)
3. Признак того, что буферный регистр содержит 0 (Z — zero). Данный бит содержит единицу, когда все биты буферного регистра = 0.



Устройство управления разработано в виде *микропрограммного устройства* — простейшего компьютера, программа которого непосредственно состоит из по-тактною изменения значений вентилях БЭВМ и простейших проверок. МПУ выполняет все машинные команды БЭВМ. Исполнение в МПУ машинной команды называется циклом команды.

Цикл команды логически разбит на четыре цикла:

- Цикл выборки команды. Осуществляет загрузку исполняемой команды в регистр команды и частичное ее декодирование. Выполняется для каждой исполняемой команды.
- Цикл выборки адреса. Исполняется только в случае косвенной адресации в команде, осуществляя дополнительное обращение к памяти, и увеличение на 1 содержимого индексной ячейки.
- Цикл исполнения. Производится исполнение команды.
- Цикл прерывания. Цикл выполняется в том случае, если разрешены прерывания и устройство ввода-вывода готово к обмену (.т.е. требует прерывания — будет обсуждено позднее).

Для обеспечения работы оператора БЭВМ в ней предусмотрена микропрограммная реализация пультовых операций:

- Ввод адреса — адрес из клавишного регистра вводится в счетчик команд.
- Чтение — информация по адресу в СК читается из памяти и помещается в РД, СК увеличивается на единицу.
- Запись — информация из КР записывается в память по адресу в СК, СК увеличивается на единицу. Используется для ввода программы и данных в режиме оператора.
- Пуск — осуществляет сброс состояния БЭВМ и переход к выполнению программы.



57

Все команды БЭВМ можно разделить на три типа, представленные на слайде. Выбор одного из типов команды осуществляется МПУ при помощи анализа старших четырех бит кода команды, которые называются *кодом операции* (сокращенно, КОП).

Адресные команды. Предназначены для осуществления операций БЭВМ с памятью. В коде команды присутствует 11-ти разрядный адрес ячейки памяти, к которой производится обращение, и бит вида адресации, который определяет, будет ли обращение к памяти прямым (0) или косвенным (1). Вместе адрес и бит косвенности составляют *адресную часть команды*.

Безадресные команды. Код операции всегда равен 1111. Остальные разряды являются расширением кода операции, позволяя реализовать дополнительные безадресные команды.

Команды ввода-вывода. Состоят их трех частей: код операции (всегда имеет значение 1110), приказ на ввод-вывод (ввод, вывод, проверка и сброс флага устройства) и 8-ми разрядный адрес устройства ввода-вывода. 8 разрядов адреса позволяет адресовать максимум 256 устройств.



Адресные команды

Наименование	Мнемон.	Код	Описание
Логическое умножение	AND M	1XXX	(M) & (A) → A
Пересылка	MOV M	3XXX	(A) → M
Сложение	ADD M	4XXX	(M) + (A) → A
Сложение с переносом	ADC M	5XXX	(M) + (A) + (C) → A
Вычитание	SUB M	6XXX	(A) – (M) → A
Переход, если перенос	BCS M	8XXX	Если (C) = 1, то M → СК
Переход, если плюс	BPL M	9XXX	Если (N)= 0, то M → СК
Переход, если минус	BMI M	AXXX	Если (N) = 1, то M → СК
Переход, если ноль	BEQ M	BXXX	Если (Z) = 1, то M → СК
Безусловный переход	BR M	CXXX	M → СК
Приращение и пропуск	ISZ M	0XXX	M + 1 → M, если (M) >= 0, то (СК) + 1 → СК
Обращение к подпрограмме	JSR M	2XXX	(СК) → M, M + 1 → СК

58

Адресные команды представлены на слайде. На этом слайде скобки обозначают «содержимое». Т.е. (M) – содержимое адресной части команды, (A) — содержимое аккумулятора, (C) содержимое одноразрядного регистра переноса. В дальнейшем не путать с обозначением косвенной адресации.

Адресные команды состоят из арифметических и логических команд, команд пересылки, команд переходов, организации подпрограмм и циклов.

Команды перехода выполняют переход на указанный адрес, если у результата имеется заданный признак. На пример, при исполнении команды BCS M (переход, если перенос) проверяется бит переноса (C) у результата, полученного предыдущей командой. Если C=1, то адресная часть команды M будет записана в счетчик команд (СК) и далее программа будет выполняться с этого адреса. Если C=0 то программа продолжит выполнение следующей по порядку команды.

Команда ISZ (IsZero) — предназначена для организации циклических вычислений. Она увеличивает на 1 содержимое ячейки, указанной в адресной части команды, и если это содержимое после увеличения меньше нуля, то выполняется следующая команда программы, а если больше либо равно нулю, то происходит «пропуск» одной команды программы. Обычно сразу после ISZ следует безусловный переход на начало цикла.

Подпрограммы — часто повторяющиеся участки, которые вынесены в отдельную программную конструкцию. В современных языках программирования они организованы в виде процедур или функций. JSR — вызов подпрограммы, возврат из нее производится с помощью команды BR с косвенной адресацией.



Безадресные команды

Наименование	Мнемон.	Код	Описание
Очистка аккумулятора	CLA	F200	$0 \rightarrow A$
Очистка рег. переноса	CLC	F300	$0 \rightarrow C$
Инверсия аккумулятора	CMA	F400	$(!A) \rightarrow A$
Инверсия рег. переноса	CMC	F500	$(!C) \rightarrow C$
Циклический сдвиг влево на 1 разряд	ROL	F600	Содержимое A и C сдвигается влево, $A(15) \rightarrow C, C \rightarrow A(0)$
Циклический сдвиг вправо на 1 разряд	ROR	F700	Содержимое A и C сдвигается вправо, $A(0) \rightarrow C, C \rightarrow A(15)$
Инкремент аккумулятора	INC	F800	$(A) + 1 \rightarrow A$
Декремент аккумулятора	DEC	F900	$(A) - 1 \rightarrow A$
Останов	HLT	F000	Отключение ТГ, переход в пультовый режим
Нет операции	NOP	F100	Место для точек отладки, «патч» программы
Разрешение прерывания	EI	FA00	
Запрещение прерывания	DI	FB00	

59

К безадресным командам относятся команды очистки и манипулирования с аккумулятором и битом C, циклических сдвиги, инкремент, декремент, команда останова и команды для работы с прерываниями.

Циклический сдвиг осуществляет побитовый сдвиг содержимого аккумулятора и бита C (17 разрядов) в заданную сторону. При сдвиге влево 0-й бит передвигается в 1-й, 1-й во второй и т. д., при этом 15-ый бит помещается в бит переноса, а бит переноса в 0-ой бит и т.д. все биты смещаются на один влево. Циклические сдвиги могут использоваться для приема побитовой информации или для умножения или деления двоичных чисел на 2.

Команды инкремент/декремент в современных процессорах реализуются быстрее, чем команды сложения и вычитания с единицей. Во всех современных процессорах реализованы отдельно.

Команда HLT переводит программу из режима работы в режим останова, отключая при этом тактовый генератор. Отключение ТГ — это особенность БЭВМ, в современных процессорах ТГ не отключается.

Команда NOP (NoOperation) — обычно используется для временной правки программы, с целью удаления записанных там инструкций. Кроме того, она может быть использована, для отладочных целей, когда необходимо временно остановить программу для анализа содержимого регистров и ячеек памяти. Для этого в программе в необходимых местах размещаются команды NOP, которые оператор может заменить на HLT для останова и анализа состояния. После останова программу можно продолжить. В современных отладчиках для языков программирования это реализовано при помощи break points или watch points.

Команды EI (Enable Interrupts) и DI (Disable Interrupts) управляют битом разрешения прерывания регистра состояния, устанавливая там 1 или 0 соответственно.



Команды ввода-вывода

Наименование	Мнемон.	Код	Описание
Очистка флага	CLF DEV	E0XX	0 → флаг устр.
Опрос флага	TSF DEV	E1XX	Если (флаг устр. DEV) = 1, то (СК) + 1 → СК
Ввод	IN DEV	E2XX	(DEV) → A
Вывод	OUT DEV	E3XX	(A) → DEV

60

Команды ввода-вывода формируют приказы на устройства ввода-вывода. Использование будет рассмотрено позднее.



ИТМО **BT**

Как выполняются эти команды?

- Ответы на ВСЕ вопросы потактового выполнения команд:
Методические указания к лабораторным работам
Приложение В, табл. В.10
- Используйте БЭВМ в режиме ТАКТ!



Рассмотрим выполнение цикла выборки команды для инструкции ADD. Обратите внимание на содержимое памяти и содержимое регистров до выполнения команды.

В БЭВМ загружена программа, выполняющая вычисление $Z=X+Y$:

020 Значение $X=53_{16}$

021 Значение $Y=106_{16}$

022 Место хранения для переменной Z (обнулено)

023 CLA — обнуление аккумулятора

024 ADD 20 — содержимое ячейки 20 добавляется к аккумулятору

025 ADD 21 — содержимое ячейки 21 будет добавлено к значению, которое находится в аккумуляторе. Т.е. сложатся значения из 20 и 21 ячеек.

026 MOV 22 — пересылка значения суммы в ячейку 22

027 HLT — останов программы

Счетчик команд (СК), как мы помним, всегда содержит адрес следующей исполняемой команды. Перед началом цикла выборки команды в СК содержится значение 025, значит в настоящий момент выполнение программы будет производиться с этого адреса.

Цикл выборки команды начинается с выбора инструкции. Для этого:

1) В первом такте содержимое счетчика команд попадает на правый вход АЛУ. На левый вход АЛУ ничего не подается (т.е. Подается значение 0), соответственно АЛУ сложит (операция по умолчанию в АЛУ, если не установлены дополнительные вентили операций АЛУ) 0 с 25 и поместит результат в буферный регистр.

2) Во втором такте содержимое 17-ти разрядного буферного регистра по шине передается в 11-ти разрядный регистр адреса. Т.к. разрядность регистров разная, то переданы будут только 11 младших разрядов. Таким образом в РА появится адрес исполняемой в настоящий момент инструкции.

Цикл выборки команды: ADD 21

Память	
020	0053
021	0106
022	0000
023	F200
024	4020
025	4021
026	3022
027	F000

РА	025
РД	4021
РК	4020
СК	025
БР	00026
А	0053

0025+1

Память	
020	0053
021	0106
022	0000
023	F200
024	4020
025	4021
026	3022
027	F000

РА	025
РД	4021
РК	4020
СК	026
БР	00026
А	0053

3) Выборка по адресу в РА (025) кода команды и запись его в РД, одновременно с этим содержимое СК увеличивается на 1 и помещается в БР

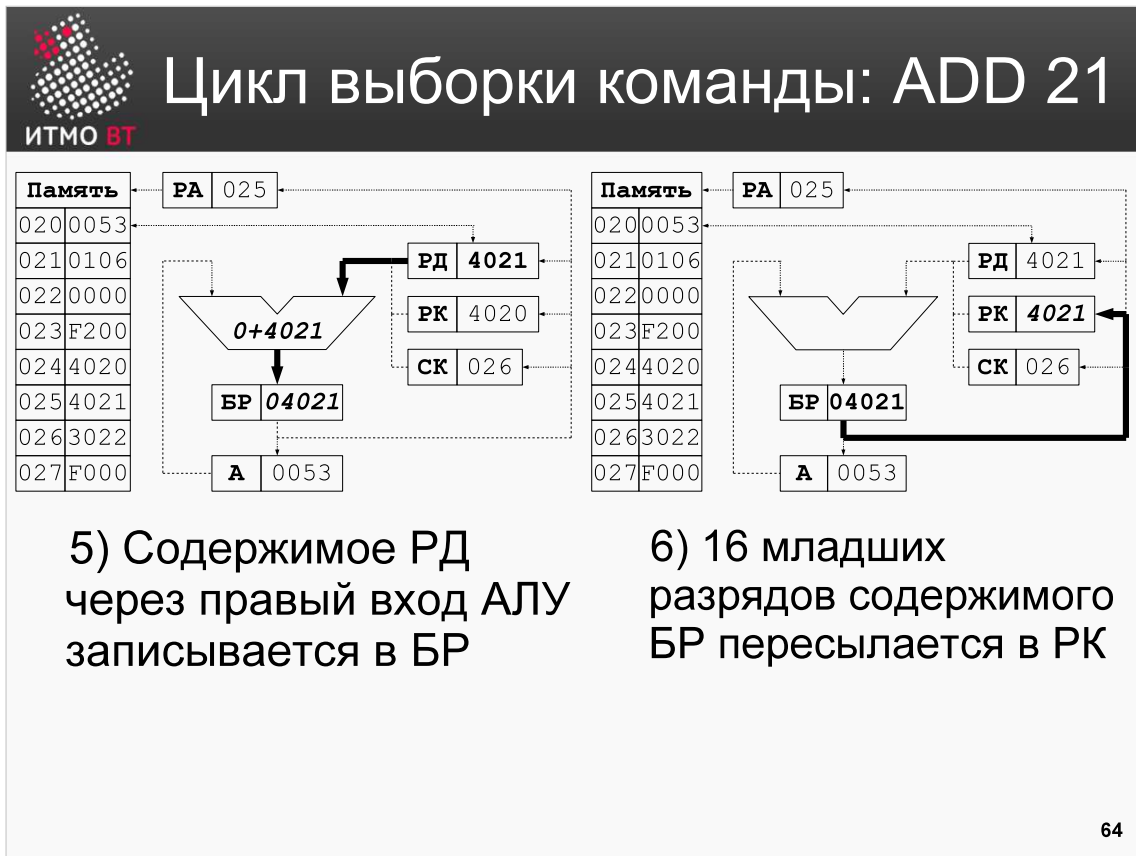
4) 11 младших разрядов содержимого БР пересылается в СК

63

Продолжение цикла выборки команды ADD 21:

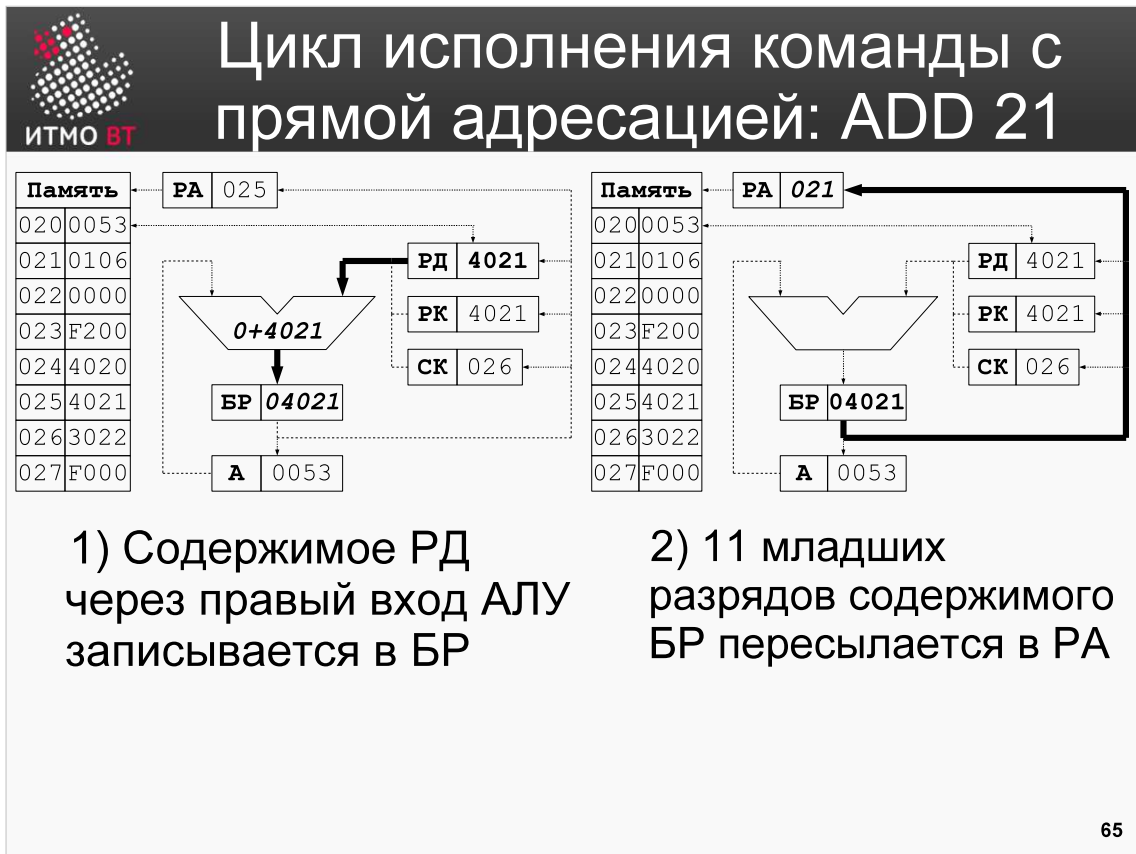
3) По содержимому регистра адреса (РА) содержимое 25 ячейки выбирается из памяти в регистр данных (РД). Т.к. при обращении к памяти правая часть схемы остается незадействованной, то возможны одновременные с обращением к памяти другие операции с АЛУ и регистрами. Соответственно содержимое СК подается на правый вход АЛУ, левый вход АЛУ при этом закрыт, и содержимое СК инкрементируется (увеличивается на 1) в АЛУ, результат этой операции попадает в БР.

4) 11 младших разрядов содержимого БР передается в обратном направлении в СК. СК теперь содержит адрес следующей исполняемой команды.



Для завершения цикла выборки команды необходимо переслать код команды, на предыдущих шагах выбранный из памяти в регистр команд. Для этого необходимо выполнить следующие микрооперации.


- 5) Содержимое регистра данных (РД) через правый вход АЛУ пересылается в буферный регистр (БР).
- 6) Содержимое БР записывается в регистр команд (РК). Теперь РК содержит код исполняемой команды для его дальнейшего декодирования и определения типа исполняемой команды.



Когда команда декодирована, адресные команды с прямой адресацией должна выбрать из памяти операнд и провести требуемую операцию. Для команды ADD 21 БЭВМ должна выбрать содержимое ячейки памяти с адресом 21 в регистр данных и сложить содержимое регистра данных с аккумулятором. Перед циклом исполнения код команды находится в регистре данных.

По тактам происходит следующее:

- 1) Содержимое РД подается на правый вход АЛУ и записывается в БР. На левый вход АЛУ в это время подается 0.
- 2) Младшие 11 разрядов содержимого БР передаются в регистр адреса.



Цикл исполнения команды с прямой адресацией: ADD 21

Память		РА
020	0053	021
021	0106	
022	0000	
023	F200	
024	4020	
025	4021	
026	3022	
027	F000	

0+0

БР 00000

А 0053

Память		РА
020	0053	021
021	0106	
022	0000	
023	F200	
024	4020	
025	4021	
026	3022	
027	F000	

53+106

БР 00159


А 0053

3) Содержимое ячейки 021 (операнд сложения) записывается в РД, **БР обнуляется!**

4) Содержимое РД на правом входе АЛУ складывается с содержимым А на левом входе АЛУ и записывается в БР

66

- 3) По адресу в РА производится обращение к памяти 021 и содержимое данной ячейки записывается в РД. В это же самое время на входы АЛУ ничего не подается, т. е. и на левом и на правом входе АЛУ будет находиться 0, АЛУ произведет операцию по умолчанию (сложение) и в буферный регистр запишется 0.
- 4) В следующем такте операнд из РД поступит на правый вход АЛУ, а содержимое аккумулятора поступит на левый вход АЛУ, произойдет операция сложения и в БР появится сумма А и выбранной ячейки памяти.



Цикл исполнения команды с прямой адресацией: ADD 21

Память	
020	0053
021	0106
022	0000
023	F200
024	4020
025	4021
026	3022
027	F000

РА 021

→

РД 0106

РК 4021

→

СК 026

БР 00159

→

А 0159

С 0

5) Младшие 16 разрядов содержимого БР записывается в А, 16 бит БР (17 по счету) записывается в бит С (carry) РС. Признаки результата N и Z обнуляются.

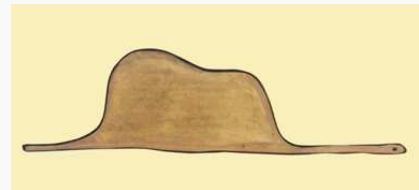
67

Последняя микрооперация переписывает содержимое БР (результат сложения) в аккумулятор, устанавливая при этом признаки выполнения операции. Т.к. при сложении переноса не было, то в С запишется 0, в признак отрицательного числа (N) запишется 0, т. к. результат сложения получился положительным, и признак Z установится так же значение 0, т. к. результат не равен нулю.



Представление информации

5



68



ИТМО ВТ

Информация и данные

3021 - что это?

69

Данные, которые хранятся в ЭВМ, это сведения, факты, показатели, выраженные как в числовой, так и любой другой форме.

Данные не могут существовать отдельно от их интерпретации человеком. Если мы напишем число 3021, что это может значить?

Правильный ответ на этот вопрос существует лишь совместно с нашим представлением о том, что имел в виду человек написавший 3021. Может его зарплата составляет 3021 рубль в месяц? Или он отмечает 3021-й день без курения? Данные, которые мы передаем или храним, должны сообщать нам сведения или, иначе, информацию о том или ином объекте или явлении.

Информация (от лат. *informātiō* — «разъяснение, представление, понятие о чём-либо», от лат. *informare* — «придавать вид, форму, обучать; мыслить, воображать») — сведения независимо от формы их представления (С) Wikipedia.

Количественные характеристики информации определены математически в теории информации Клода Шеннона. Именно он ввел понятия бит и информационная энтропия. Одна его теория не учитывает ценность информации для конкретного ее применения.

Область представления

Le Petit Prince




Le BComp

12321

0011 0000 0010 0001

MOV21

0011 0000 0010 0001

↓ код
↓ адрес
↓ операнда

бит адресации

70

“Я показал мое творение взрослым и спросил, не страшно ли им? - Разве шляпа страшная? - возразили мне. А это была совсем не шляпа. Это был удав, который проглотил слона. Тогда я нарисовал удава изнутри, чтобы взрослым было понятнее. Им ведь всегда нужно все объяснять.” (С) Антуан де Сент-Экзюпери. Маленький Принц.

В архитектуре фон Неймана память команд и данных объединена. Если мы сохранили в БЭВМ в ячейке памяти число 3021_{16} , то для нас информационная энтропия этих данных велика. Мы не можем понять, например, является ли это командой MOV21, числом 12321_{10} , набором логических переменных или чем-либо еще.

Поэтому, для всех данных, сохраненных в памяти и используемых процессором, чтобы отражать ту или иную информацию, должна быть *определена область представления информации* — набор связей, которые сопоставляет хранимые значения с их интерпретацией программистом. Таким образом, в БЭВМ область представления – это то, каким образом биты слова БЭВМ интерпретируются программистом для решения необходимой задачи.



Допустимые значения

- Определяются областью представления
- Примеры:
 - Безадресные команды БЭВМ:

F000, F100, F200,
 F200, F300, F400,
 F500, F600, F700,
 F800, F900, FA00,
 FB00
 - Логические значения:

Истина, Ложь
 - Целые беззнаковые положительные числа ≤ 65535


0,1,2,3,4,5,6,7,...65531,65532,65533,65534,65535



С областью представления информации тесно связана область допустимых значений. *Область допустимых значений* определяет весь набор символов или значений, допустимых в контексте информационной обработки или обмена для заданной области представления.

Основным ограничением для ОДЗ обычно служит разрядность слова, используемого в процессоре. В БЭВМ ограничение слова — 16 разрядов, и, поэтому, вся информация, которую мы хотим закодировать, должна учитывать это ограничение. Кроме того, мы должны учитывать, чтобы результаты операций (вспомним перенос) также помещались в эту разрядность.

Для БЭВМ на слайде приведены примеры различных ОДЗ в зависимости от области представления. Например, ОДЗ для безадресных команд БЭВМ представлена множеством всех безадресных команд, а натуральные числа могут быть представлены целым беззнаковым числом, при этом максимальное значение числа не может превышать 65535.



Представление чисел: фиксированная точка

- **Целые: двоичная точка фиксирована за разрядом с номером 0, веса положительные**
 - 1) номер бита и степень веса разряда соответствуют

№	3	2	1	0	
Вес	8	4	2	1	
	1	0	1	1	1

$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 0 + 2 + 1 = 11$$
 - 2) степень веса разряда ~ номеру бита + 1

№	3	2	1	0	
Вес	16	8	4	2	1
	1	0	1	1	1

$$1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 = 16 + 0 + 4 + 2 = 22$$

72

Внутри разрядной сетки в БЭВМ можно закодировать различные данные, вместе со смысловым наполнением БЭВМ может использоваться для хранения информации. Одна из наиболее часто применяемых видов данных - это представление чисел.

В ЭВМ в основном используется три вида представления чисел — числа с фиксированной точкой, плавающей точкой и представление в виде строки символов. Особенностью представления чисел с фиксированной и плавающей точкой является то, что они должны размещаться в разрядной сетке ЭВМ (т. е. фактически ограничены определенным количеством двоичных разрядов). Рассмотрим представление чисел с фиксированной точкой в 4-х разрядной сетке.

Первый пример на слайде хорошо известен со школы, когда двоичная точка фиксирована после нулевого разряда двоичного представления числа. Точка разделяет целую и дробную часть числа, которая в данном примере отсутствует. У каждого из разрядов определен номер разряда и вес разряда в двоичном представлении, который соответствует основанию системы счисления (2) в степени номера разряда. Само число формируется следующим образом: значение ячейки из каждого разряда умножается на вес, и для получения представления числа суммируются все полученные значения для каждого из разрядов.

Рассмотрим ситуацию, когда вес разряда соответствует номеру разряда с положительным смещением. Это позволяет нам кодировать числа большего размера в ту же самую разрядную сетку, но при этом шаг между допустимыми числами будет больше. Во втором примере на слайде двоичная фиксированная точка находится в позиции, соответствующей номеру разряда + 1. В данном случае не возможно представить нечетные числа. Двоичную точку можно переместить еще дальше, в зависимости от задачи, при этом будут получаться числа, которые кратны 4-м, 8-ми и т.д. Алгоритм перевода двоичного числа в десятичное такой же, что и в первом примере.

Обратите внимание, что двоичное представление в разрядной сетке в обоих примерах совпадает, однако результат перевода числа в десятичную в первом случае равен 11, а во втором - 22. Как можно видеть именно человек решает, какую интерпретацию данных выбрать для решения конкретной практической задачи.



Представление чисел: фиксированная точка

- Вещественные: **двоичная** точка фиксирована за разрядом с номером 2 с весом 0, существуют «отрицательные» веса:

№	3	2	1	0
Вес	2	1	1/2	1/4

1	0	1	1
---	---	---	---

$1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = 2 + 0 + 0.5 + 0.25 = 2.75$
- Вещественные: фиксирование **десятичной** точки - изменение *масштаба*:

3	2	1	0
1	1	0	0

$(1 \times 2^3 + 1 \times 2^2) \times 1 \times 10^{-1} = (8 + 4) \times 0.1 = 1.2$



73


При помощи фиксирования двоичной точки возможно представление вещественных чисел. В этом случае мы можем фиксировать нашу двоичную точку после разряда с заданным номером, разделяя разрядную сетку на целую и дробную части числа.

На верхнем примере слайда, двоичная точка фиксирована после разряда с номером 2. Таким образом для целой части отведены веса с порядками 1 и 0 (слева от точки) и веса с порядками -1, -2 (справа от точки, образуя дробную часть). Принцип формирования дробной части такой же, как и у целой части. Отметим, что в дробной части невозможны произвольные значения дробной части, ее значения складываются из отрицательных степеней двойки.

Обратите внимание, что в предыдущих примерах мы фиксировали двоичную точку, то есть брали двоичное представление числа, выбирали разряд в этом двоичном представлении и устанавливали за ним необходимый порядок веса разряда. В представлении чисел в разрядной сетке можно фиксировать не только двоичную, но и десятичную точку.

На примере, приведенном на нижнем рисунке видим, что для двоичного числа упакованного в разрядную сетку можно применить дополнительный множитель порядка после его перевода в десятичную систему, таким образом изменяя масштаб полученного числа. Соответственно можно дробные числа представлять с любой точностью, нужно только подобрать правильный масштаб. В данном случае после перевода числа из двоичной системы мы переместили точку на одну позицию влево, тем самым получили из числа 12 число 1,2.

В приведенных выше примерах, процессор не знает, с какими числами мы работаем и где была установлена фиксированная точка. Для него все числа – целые. Представление чисел – это наши представления о том, что находится в ячейке памяти, регистре и пр. Для компьютера эти данные — это просто набор двоичных чисел, и только пользователь уже вкладывает в это свой смысл.



Представление беззнаковых целых чисел

- Количество разрядов в разрядной сетке определяет область допустимых значений
 - Минимальное 4-х разрядное беззнаковое число:

3	2	1	0	
0	0	0	0	$0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 0$
 - Максимальное 4-х разрядное беззнаковое число:


3	2	1	0	
1	1	1	1	$1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 15 = 2^4 - 1$
 - Диапазон:

$$0 \leq X \leq 2^4 - 1$$
 - Диапазон для 16-ти разрядного беззнакового числа:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	$0 \leq X \leq 2^{16} - 1 = 65535_{74}$

Внутри БЭВМ, внутри той же самой разрядной сетки можно представить знаковые и беззнаковые числа. На слайде показано представление беззнаковых чисел.

На примере четырехразрядной сетки видно, что минимальным беззнаковым числом может быть 0, а максимальным – 15. В БЭВМ максимальное значение для 16-ти разрядного беззнакового числа определено на слайде. Минимальное значение числа составляет 0, а максимальное – 65535.



Представление знаковых целых чисел

- Нужно хранить признак знака числа достаточно 1-го бита, «0» значит «+», «1»=«-»
 - Прямое кодирование (прямой код числа)

	3	2	1	0
	+	3		
0	0	1	1	

	3	2	1	0
	-	3		
1	0	1	1	

$$-7 = -(2^3 - 1) \leq X \leq 2^3 - 1 = 7$$


Двойной нуль!
 - Дополнительный код

$$M = b^n - K$$

M — дополнение к числу K ($10-3=7$) !

b — основание системы счисления

n — количество разрядов



Представление чисел со знаком более сложно. Существует несколько способов кодирования знаковых чисел.

Для начала рассмотрим простое прямое кодирование. Знак можно хранить, записав его в отдельный, обычно самый старший, разряд слова. Общепринято, что знак кодируется 0 если число положительное и 1 если отрицательное. В примере на слайде закодированы положительное число +3 и отрицательное -3 в четырехразрядной сетке. Знак определяет старший бит с номером 3.

Областью допустимых значений для такого 4-х разрядного числа будет промежуток от -7 до 7 включительно. Особо отметим, что для такого способа кодирования значение нуля представлено двумя кодами — положительным и отрицательным (двойной нуль), что является нонсенсом с точки зрения математики.

Прямой код в вычислительных устройствах напрямую не используется, т.к. необходима дополнительная логика при определении знаков и больших по модулю чисел при выполнении операций. Конструкция АЛУ при использовании прямого кода существенно усложняется.

Для выполнения вычислений обычно используется *дополнительный* код числа, который представляет отрицательные числа в виде дополнения до максимально возможного положительного числа + 1 в заданной разрядной сетке. Математически дополнительный код числа представлен в виде формулы, показанной на слайде. Рассмотрим данный способ более подробно

Дополнительный код можно использовать для любой позиционной системы исчисления. Систему счисления в формуле представляет основание системы счисления b . Количество разрядов в разрядной сетке числа задается параметром n .

Например, мы хотим представить -1 в десятичной системе счисления, ограниченной 5-ю разрядами. Тогда b^n в десятичной системе исчисления = $10^5 = 100\,000$. Применяя формулу, получим M (число в дополнительном коде, соответствующее -1) = $b^n - K = 100\,000 - 1 = 99\,999$. Теперь наше АЛУ может выполнять корректные арифметические действия, не прибегая к сравнению знаков и модулей чисел. Пусть нам нужно к -1 прибавить 1, должен получиться 0. Соответственно, $99999 + 1 = 10000$, старший разряд выходит за заданные пять разрядов и отбрасывается $99999 + 1 = 0$.



Представление знаковых чисел: дополнительный код

$$M = b^n - K = ((b^n - 1) - K) + 1$$

Прямой код 5-ти разр. дес. чисел	Дополнительный код		
	5-ти разр. дес. чисел	4-х разр. шестн. чисел	16-ти разрядных двоичных чисел
-50000	50000		
-49999	50001		
-32768	67232	8000	1000 0000 0000 0000
-32767	67233	8001	1000 0000 0000 0001
-2	99998	FFFE	1111 1111 1111 1110
-1	99999	FFFF	1111 1111 1111 1111
0	00000	0000	0000 0000 0000 0000
1	00001	0001	0000 0000 0000 0001
32767	32767	7FFF	0111 1111 1111 1111
49999	49999		

K=+3

3 2 1 0

0 0 1 1

$M = b^n - K$

$2^4 - 3 = 13$

1 0 0 0 0

- 0 0 1 1

M= 1 1 0 1

ИНВ.

0 0 1 1

+ 1 1 0 0

1

M= 1 1 0 1

76

На слайде показаны примеры представления чисел в дополнительном коде. Для пятиразрядной десятичной системы счисления до 49999 включительно числа будут положительными, а начиная 50 000 отрицательными. При этом в размерной ячейке они будут выглядеть, как положительные. -1 это 99999, -2 это 99998 и т. д. Обратите внимание, что самое минимальное число, которое может быть представлено -50 000.


Для упрощения вычисления дополнительного кода в двоичном представлении используют модифицированную формулу, которая получается прибавлением и вычитанием единицы. Приведем еще один пример вычислений в допкоде для -10.

$$M = b_n - K = ((b_n - 1) - K) + 1 = 99999 - 10 + 1 = 99990 \text{ — допкод для } -10.$$

Проведем операцию сложения -10 с 9. Должно получиться -1. $99990 + 9 = 99999$, что соответствует числу -1 в прямом коде (см. таблицу на слайде). Как видно, менять конструкцию АЛУ для выполнения такой операции не нужно, достаточно просто представить число в дополнительном коде.

Смысл использования модифицированной формулы состоит в упрощении перевода числа в отрицательное. $b_n - 1$ представляет собой максимальное положительное беззнаковое число в выбранной разрядной сетке. При этом для получения дополнительного кода для числа $(K-1)$, можно использовать поразрядное дополнение для каждой цифры числа до максимально возможной в выбранной системе счисления. Например поразрядное дополнение для числа 12345 в десятичной системе будет число 87654 ($9-1=8$, $9-2=7$, и т. д.). Останется только прибавить 1 и мы получим представление числа в дополнительном коде, т. е. $-12345=87654+1=87655$. Леко заметить что сумма 12345 и 87655 дает 100000 или b_n .

В двоичной системе получение поразрядного дополнения совпадает с простой инверсией битов числа. То есть, часть формулы $= ((b_n - 1) - K)$ просто заменяется инверсией разрядов числа K. В результате отрицательное число получается в 2 действия: первое – инверсия числа, второе – прибавление 1 к инверсии.



Получение дополнительного кода БЭВМ

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
010	F200	CLA	
011	4016	ADD 16	X в аккумуляторе (2)
012	F400	CMA	Вычисление дополнения (инверсия битов - FFFD)
013	F800	INC	Инкремент (FFFE)
014	3017	MOV 17	Сохранение результата
015	F000	HLT	
016	0002	X	X
017	FFFE	R	-X

-32768	8000	1000 0000 0000 0000
-32767	8001	1000 0000 0000 0001
-2	FFFE	1111 1111 1111 1110
-1	FFFF	1111 1111 1111 1111
0	0000	0000 0000 0000 0000
1	0001	0000 0000 0000 0001
32767	7FFF	0111 1111 1111 1111

Пример программы, которая меняет знак числа на противоположный в дополнительном коде для шестнадцатиразрядного слова БЭВМ представлена на слайде.

Обратите внимание, что не важно, было ли число уже в дополнительном коде (т.е. отрицательное) или же положительное. Последовательность операций CMA и INC в любом случае меняет знак числа.

Обратите внимание на строчку в представлении отрицательных чисел, выделенную красным. Число -32768 – это минимальное отрицательное число, которое может существовать в 16-ти разрядной сетке. Для такого нет соответствующего ему положительного числа. То есть за счет особенностей дополнительного кода и отсутствия двойного нуля, отрицательных чисел на одно больше, чем положительных.

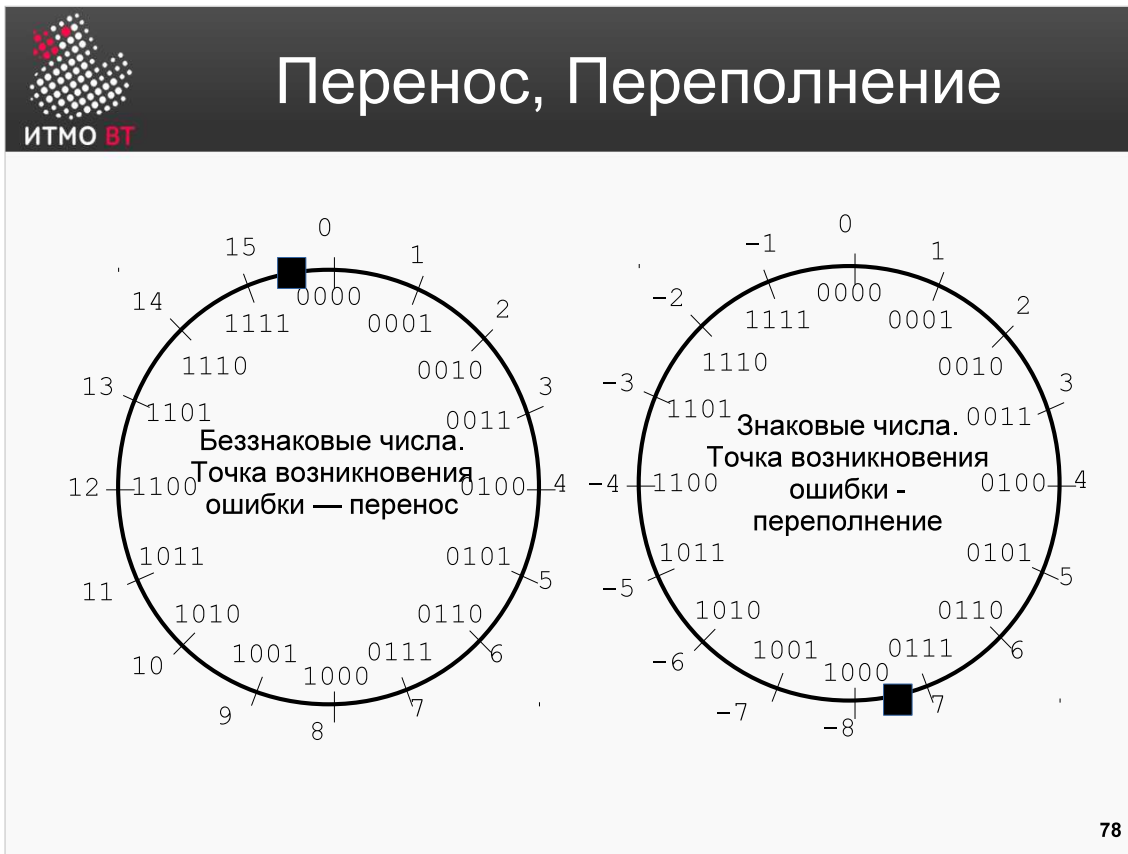
Рассмотрим, что получится, если попытаться изменить знак этого отрицательного числа.

Исходное число: 1000 000 0000 0000

1. Инвертируем. Получаем 0111 1111 1111 1111

2. Прибавляем единицу. Получаем 1000 000 0000 0000. То же самое число.

Такое поведение называется переполнением, которое аналогично возникновению перенос в знаковых числах.



Подчеркнем еще раз, что в БЭВМ числа могут быть знаковые и беззнаковые. Т.е в одной разрядной сетке, под «шляпой» могут скрываться различные представления значений числа. Механика их обработки в АЛУ одинаковая. Если в нашей области представления, мы думаем о том, что нам нужно работать с отрицательными числами, то тогда числа превращаются в знаковые, если нам нужны только положительные числа, тогда они «магически» становятся беззнаковыми. Однако любом случае мы должны контролировать выход за пределы разрядной сетки и потери значения числа, вызванных переполнением или переносом.

На слайде в виде круговой диаграммы показаны знаковое и беззнаковое представление чисел в четырехразрядной сетке в виде круговой диаграммы. Для каждого из представлений в виде черной точки показана точка возникновения ошибки при выходе за границы разрядной сетки.

В беззнаковых числах, если прибавить 1 к 15 или вычесть из 0 единицу произойдет потеря значения числа, которая сопровождается возникновением переноса. Этот перенос в конструкции ЭВМ учитывается в бите C (carry), который служит сигналом программисту, что произошла ситуация потеря значения, и ее необходимо обработать отдельно.

В знаковых числах точка возникновения ошибки расположена между представлением чисел -8 и 7. При вычитании из -8 единицы или прибавления единицы к 7 возникает переполнение (Overflow), которое в ЭВМ контролируется битом переполнения, обозначаемым в разных архитектурах O, V или OV. АЛУ определяет переполнение по следующему правилу: если поразрядные переносы в знаковом и старшем разряде одновременно отсутствуют или присутствуют - значит переполнения нет, если присутствует только в одном – значит переполнение знаковой разрядной сетки есть.

<pre> 1101 ← биты совпадают +1101 -3 0101 +5 ----- 10010 2 C=1 OV=0 </pre>	<pre> 1000 ← биты различны +1000 -8 1111 -1 ----- 10111 7 (≠-9) ←ошибка C=1 OV=1 </pre>	<pre> 0111 ← биты различны +0111 7 0001 1 ----- 01000 -8 (≠8) ←ошибка C=0 OV=1 </pre>
---	--	--

В БЭВМ бит переполнения в конструкции не предусмотрен.



БЭВМ: представление чисел

Представление в разрядной сетке	Беззнаковые числа	Знаковые числа
0000 0000 0000 0000	0	0
0000 0000 0000 0001	1	1
...		
0111 1111 1111 1110	32766	32766
0111 1111 1111 1111	32767	32767
1000 0000 0000 0000	32768	-32768
1000 0000 0000 0001	32769	-32767
...		
1111 1111 1111 1110	65534	-2
1111 1111 1111 1111	65535	-1

ОДЗ: $0 \leq X \leq 2^{16} - 1$ $-2^{15} \leq X \leq 2^{15} - 1$

79

Область допустимых значений для представления в БЭВМ знаковых и беззнаковых чисел представлена на слайде. Разрядная сетка слова, с которой БЭВМ выполняет операции включает 16 разрядов. Если необходимо более высокая разрядность числа, то при помощи команды учета переноса при сложении (ADC) возможны операции с 32-х разрядными и более числами. В зависимости от необходимого представления числа программист должен учитывать максимальное и минимальное значение для используемых чисел.



Представление чисел с плавающей точкой

В БЭВМ — НЕТ!



IEEE 754 Single Precision Format

$$X = (-1)^{(\text{sign})} \times (1 + \text{Mantissa}) \times 2^{(\text{exponent} - 127)}$$

80

Кроме представления чисел с фиксированной точкой существует формат для математических вычислений, который использует *плавающую* точку. В плавающей точке положение десятичной точки не фиксируется за определенным разрядом, т.е. оно может меняться в зависимости от порядка числа.

На слайде видим, представление числа состоит из мантиссы и порядка. *Мантисса* – это дробная часть числа. Она всегда внутри представлена *нормализованной*, т.е. подразумевается, что она всегда начинается с единицы.


Пример ненормализованной мантиссы: 0.000100101.

Так как мантисса представлена в виде двоичного числа, то перед тем, как число упаковать, его сдвигают так, чтобы всегда число было нормализовано. Для нашего примера это будет 0.100101. Так как незначащий ноль всегда присутствует в представлении, то его можно убрать. Поэтому после нормализации, упакованная в разрядную сетку согласно формату IEEE 754 Single Precision Format, мантисса числа будет выглядеть так: .100 1010 0000 0000 0000 0000

Для хранения порядка числа, который указывает, на сколько разрядов была сдвинута запятая, используются отдельные разряды. Старший разряд используется для хранения знака числа. Знак порядка образуется путем вычитания константы из значения порядка.

Зная формат хранения числа можно вычислить, какое число может быть максимальным и минимальным. Предлагаю вам сделать это самостоятельно.

Форматов с плавающей точкой разработано достаточно много. Они все зависят от модели процессора и менялись со временем. В современных процессорах вычисления с плавающей точкой производятся специальным устройством — FPU (Floating point unit).



Представление логической информации

- 1-true, 0-false
- 16-ти разрядное число содержит 16 логических значений

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

ОДЗ: $X_i \in \{0,1\}$ где $0 \leq i \leq 15$

81

Логическая информация представлена в БЭВМ как показано на слайде.

В 16-ти разрядной сетке БЭВМ каждому биту из логической информации соответствует значению: либо true, либо false. При этом общепринято, что 1 это истина (true), а 0 = ложь или false. И таких операндов, соответственно, 16.

Когда применяется команда AND (поразрядное И) в БЭВМ, то происходит логическое умножение между одной ячейкой памяти и аккумулятором. Соответственно, будет произведено 16 логических операций одновременно. Полученный результат является результатом логической операции, и в большинстве случаев его нельзя трактовать, как математический (т. е. знаковый или беззнаковый, это просто 16 отдельных логических значений).

Область допустимых значений для логических переменных представлена на слайде.




БЭВМ Лаб№2: ОПИ и ОДЗ

- $R=(X\&Y)+Z$

Область Представления:

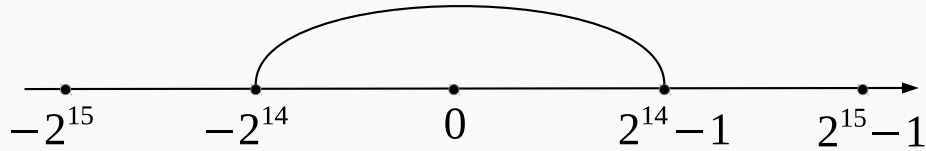
- R — знаковое, 16-ти разрядное число
- X,Y — набор из 16 логических однобитовых значений
- Z — знаковое, 16-ти разрядное число
- Результат логической операции X&Y трактуется как арифметический операнд:
 - (X&Y) — знаковое, 16-ти разрядное число

 **БЭВМ Лаб№2: ОПИ и ОДЗ**

- $R=(X\&Y)+Z$ Допустимые значения:


с R все просто: $-2^{15} \leq R \leq 2^{15} - 1$

Случай 1. Если ограничить разрядность слагаемых, то переполнения не возникнет

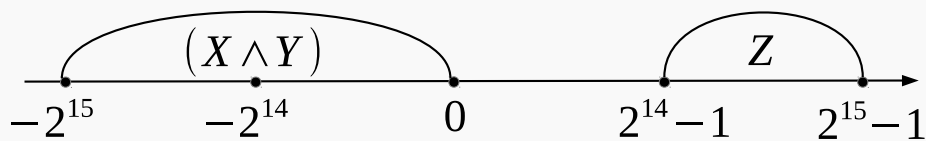


$$\begin{cases} -2^{14} \leq (X \wedge Y), Z \leq 2^{14} - 1 \\ X_i, Y_i \in \{0,1\}, \text{ где } 0 \leq i \leq 14 \end{cases}$$


- «Де-факто» мы потеряли половину возможных значений $(X\&Y)$ и Z . Плохо!


БЭВМ Лаб№2: ОДЗ $R=(X\&Y)+Z$

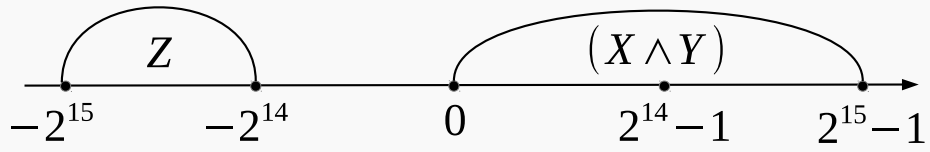
Случай 2. Пусть $2^{14} \leq Z \leq 2^{15} - 1$



$$\left\{ \begin{array}{l} 2^{14} \leq Z \leq 2^{15} - 1 \\ X_{15} = 1, Y_{15} = 1 \\ X_i, Y_i \in \{0, 1\}, \text{ где } 0 \leq i \leq 14 \end{array} \right.$$


БЭВМ Лаб№2: ОДЗ $R=(X\&Y)+Z$

Случай 3. Пусть $-2^{15} \leq Z \leq -2^{14} - 1$



$$\left\{ \begin{array}{l} -2^{15} \leq Z \leq -2^{14} - 1 \\ X_{15} = 1, Y_{15} = 0 \\ X_i, Y_i \in \{0, 1\}, \text{ где } 0 \leq i \leq 14 \end{array} \right.$$

$$\left\{ \begin{array}{l} -2^{15} \leq Z \leq -2^{14} - 1 \\ X_{15} = 0, Y_{15} = 1 \\ X_i, Y_i \in \{0, 1\}, \text{ где } 0 \leq i \leq 14 \end{array} \right.$$



Представление символьной и текстовой информации

7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	1

space	!	"	#	\$	%	&	'	()	*	+	.	-	/	
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
0	1	2	3	4	5	6	7	8	9	:	.	<	=	>	?
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111

A

86

Представление текстовой информации в вычислительных машинах основано на кодировании букв алфавитов для существующих на земле языков, которые традиционно используются человечеством. *Символ* - это графическое изображение, которое используется человеком для создания слов, текстов и другой значимой информации. Как известно, к символам относятся буквы, знаки препинания, символы валют, цифры и т.д.

В ЭВМ представление текстовой информации основывается на кодировании букв и символов при помощи кодовой таблицы. *Кодовая таблица* (или *кодировка символов*) является соглашением между разработчиками о соответствии каждому символу определенного порядкового номера или кода, чтобы его можно было сохранять в памяти ЭВМ или передавать по каналам связи. Пример такой кодовой таблицы представлен на слайде. Например, если нам необходимо закодировать латинскую букву «А», то по таблице мы можем определить, что этой букве соответствует код 65₁₀. Если нам необходимо закодировать буквами слово, то как и в написании этого слова, коды букв в памяти будут располагаться последовательно, например слово DEDA будет представлено последовательностью кодов 69, 68, 69, 65, которые можно представить в ЭВМ, например, в виде целых беззнаковых чисел размером в 1 байт.

Для работы пользователя с символьной информацией недостаточно сохранять в памяти коды символов. Необходимо еще и отображать текстовую информацию на экране (или других устройствах вывода) компьютера. Для хранения графических начертаний символов в ЭВМ существуют *шрифты* (fonts). Они хранят все графические изображения символов в заданном алфавите, и обычно устанавливаются вместе с операционной системой, драйвером принтера или микропрограммой BIOS. В шрифтах каждому коду символа соответствует свое начертание. Отдельно хранятся изображения строчных и заглавных букв, цифр, специальных символов и т. д.

Шрифты бывают векторные и растровые. В *растровых* шрифтах каждому коду символа соответствует изображение, определенного (в точках) размера. В *векторных* хранится принцип начертания символа в виде последовательности линий.



Символы: ASCII

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

7 БИТ! Старший — для контроля четности

87


Как отмечено на предыдущем слайде, символы кодируются с использованием кодовой таблицы. Первая кодировка содержала 32 символа, и была предназначена для использования в телеграфе. Легко заметить, что символы этой кодировки можно было упаковать в пять бит.

Следующей важной кодировкой символов, коды символов которой можно увидеть во всех современных кодировках стала кодовая таблица ASCII. ASCII расшифровывается как American Standard Code for Information Interchange. Таблица представлена на слайде. Как можно видеть, в ней присутствуют заглавные и прописные буквы, цифры, знаки препинания, денежные знаки, спецсимволы. В первых 32 позициях кодировки расположены управляющие символы, которые используются в качестве служебных, например, для перемещения курсора на терминале и организации текстов.

Организация текста внутри памяти вычислительной системы обычно совпадает с организацией файла формата «.txt», который, в свою очередь, был придуман исходя от принципа работы пишущей машинки (первые консоли ЭВМ и были пишущими машинками). Поэтому, в кодовой таблице есть символы перевода строки (LF), возврата на один символ (BS), возврата каретки (CR) и другие. Символ NUL в современных системах часто используется для признака окончания строки. С помощью символа BS на принтере можно печатать один символ поверх другого. В ASCII таким способом можно добавить к буквам диакритические знаки. Пример: a BS ' → á.

Отметим, что разные ОС по-разному кодируют конец строки. Это связано с особенностью работы печатных машинок и телетайпов. UNIX системы используют один символ — CR (при этом виртуальная печатная машинка как бы переводит печатную консоль в начало и автоматически на следующую строку), а Windows — последовательностью из двух: перенос строк и возврат каретки. Из-за этого текстовые файлы между этими ОС не полностью совместимы.

В ASCII символы кодировались семью разрядами, 8-й (старший) разряд не использовался для кодировки символов, а использовался для *контроля четности* битов передаваемого по каналам связи символа. При этом на передающей стороне считалось количество единичных битов в коде символа и если оно было четным, то старший разряд устанавливался в единицу. На приемной стороне также подсчитывалась четность количества единичных битов кода символа и проверялось на совпадение с старшим битом символа. Если эти разряды не совпадали, это служило признаком ошибки передачи.



Символы: ASCII (КОИ-7Н0) КОИ-7Н1 (РУС), КОИ-7Н2 (Mix)

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO		0
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS		0
2		!	"	#	\$	%	&	'	()	*	+	,	-	.		
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>		
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N		
5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^		
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n		
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~		

0

**КОИ-7
Н1**

127

0

**КОИ-7
Н2**

127

20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	

88

Для представления русских символов в СССР была предложен набор кодовых таблиц КОИ-7 (Код Обмена Информацией). КОИ-7 включает в себя 3 «набора» — Н0, Н1, Н2. Н0 — это просто US-ASCII (однако символ доллара \$ заменён на символ валюты ₭), в Н1 все латинские буквы заменены на русские; в Н2 заглавные латинские буквы оставлены, а строчные заменены на заглавные русские.

Кодировка является семибитной, как и ASCII оставляя старший бит для контроля четности. Другой ее особенностью было то, что русские символы, идентичные по начертанию латинским находились в тех-же местах, что и английские. В случае отсутствия в системе русских шрифтов текст, пусть и с неудобствами в восприятии всегда можно было прочитать с помощью английских шрифтов.

Минусом такого подхода была усложненная сортировка данных в алфавитном порядке. Для сортировки была необходима дополнительная таблица, которая содержала коды символов в алфавитном порядке.

Символы: КОИ-8

Старшая часть таблицы
Extended ASCII (КОИ-8)

80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
—		Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
⋮	⋮	⋮	Г	■	●	√	≈	≤	≥		J	°	2	.	÷
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
=		Г	ё	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
Г	Г	Г	ё	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	©
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
ю	а	б	ц	д	е	ф	г	х	и	й	к	л	м	н	о
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
п	я	р	с	т	у	ж	в	ь	ы	з	ш	э	щ	ч	ъ
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
Ю	А	Б	Ц	Д	Е	Ф	Г	Х	И	Й	К	Л	М	Н	О
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF
П	Я	Р	С	Т	У	Ж	В	Ь	Ы	З	Ш	Э	Щ	Ч	Ъ

Когда появилась Extended ASCII, где для символов использовались все 8 разрядов байта, было решено для русских заглавных и строчных символов использовать позиции верхней части таблицы. Данная кодовая таблица получила название КОИ-8. В ней кроме символов русского языка были еще и символы псевдографики для рисования таблиц.

Следует заметить, что если по каким-то причинам ПО не поддерживало 8-ми битные кодировки и обрезало старший бит (при контроле четности) то русские символы перемещались в младшую часть таблицы, и их можно было прочитать по сходным с русскими по начертанию английским символам.

КОИ-8 до сих пор используется в качестве стандарта для обмена электронной почтой.



Символы: ISO8859-5 (ГОСТ-основная)

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-		0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F
1-	0010	0011	0012	0013	0014	0015	0016	0017	0018	0019	001A	001B	001C	001D	001E	001F
2-		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3-	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	002A	002B	002C	002D	002E	002F
4-	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
5-	0030	0031	0032	0033	0034	0035	0036	0037	0038	0039	003A	003B	003C	003D	003E	003F
6-	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
7-	0040	0041	0042	0043	0044	0045	0046	0047	0048	0049	004A	004B	004C	004D	004E	004F
8-	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
9-	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	005A	005B	005C	005D	005E	005F
A-	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
B-	0060	0061	0062	0063	0064	0065	0066	0067	0068	0069	006A	006B	006C	006D	006E	006F
C-	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
D-	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079	007A	007B	007C	007D	007E	007F
E-	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	008A	008B	008C	008D	008E	008F
F-	0090	0091	0092	0093	0094	0095	0096	0097	0098	0099	009A	009B	009C	009D	009E	009F
A-	Ё	Ђ	Ѓ	Є	Ѕ	І	Ї	Ј	Љ	Њ	Ћ	Ќ	-	Ў	Ц	
B-	00A0	00A1	00A2	00A3	00A4	00A5	00A6	00A7	00A8	00A9	00AA	00AB	00AC	00AD	00AE	00AF
C-	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
D-	00B0	00B1	00B2	00B3	00B4	00B5	00B6	00B7	00B8	00B9	00BA	00BB	00BC	00BD	00BE	00BF
E-	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
F-	00C0	00C1	00C2	00C3	00C4	00C5	00C6	00C7	00C8	00C9	00CA	00CB	00CC	00CD	00CE	00CF
A-	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
B-	00D0	00D1	00D2	00D3	00D4	00D5	00D6	00D7	00D8	00D9	00DA	00DB	00DC	00DD	00DE	00DF
C-	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
D-	00E0	00E1	00E2	00E3	00E4	00E5	00E6	00E7	00E8	00E9	00EA	00EB	00EC	00ED	00EE	00EF
E-	№	ё	ђ	ѓ	є	ѕ	і	ї	ј	љ	њ	ќ	ќ	ѕ	ў	ц
F-	2116	0461	0462	0463	0464	0465	0466	0467	0468	0469	046A	046B	046C	00A7	046E	046F

ISO 8859-5 — 8-битная кодовая страница из семейства кодовых страниц стандарта ISO-8859 для представления кириллицы.

ISO 8859-5 была создана в 1988 году на базе «основной кодировки» (все русские буквы сохранили своё расположение, за исключением заглавной Ё). Российской адаптацией стандарта является ГОСТ Р 34.303-92, в котором кодировка названа КОИ-8 В1, однако в ней не установлены буквы нерусских алфавитов и коды управляющих символов.

ISO 8859-5 широко применяется в Сербии и иногда в Болгарии на юникоподобных системах. В России эта кодировка в настоящее время почти не употребляется (взамен на свободно распространяемых юникоподобных системах широкое применение нашла КОИ-8); тем не менее на некоторых коммерческих юникс-системах (Solaris, AIX, HP-UX) для русского языка по умолчанию до недавнего времени ставилась ISO 8859-5.

Надо отметить, что до широкого использования Unicode (см. далее) в российском сегменте вычислительных систем велись «войны кодировок». «Де факто» используется пять разных таблиц кодировок. Совместно с КОИ-8 использовались две кодировки ISO8859-5 (ГОСТ-основная) и cp866 (ГОСТ-альтернативная, в DOS), Win-1251, кириллическая кодировка для компьютеров Macintosh. Преимущества по сравнению с КОИ-7: русские буквы по алфавиту. Символ занимает 1 байт.



Символы: WIN1251

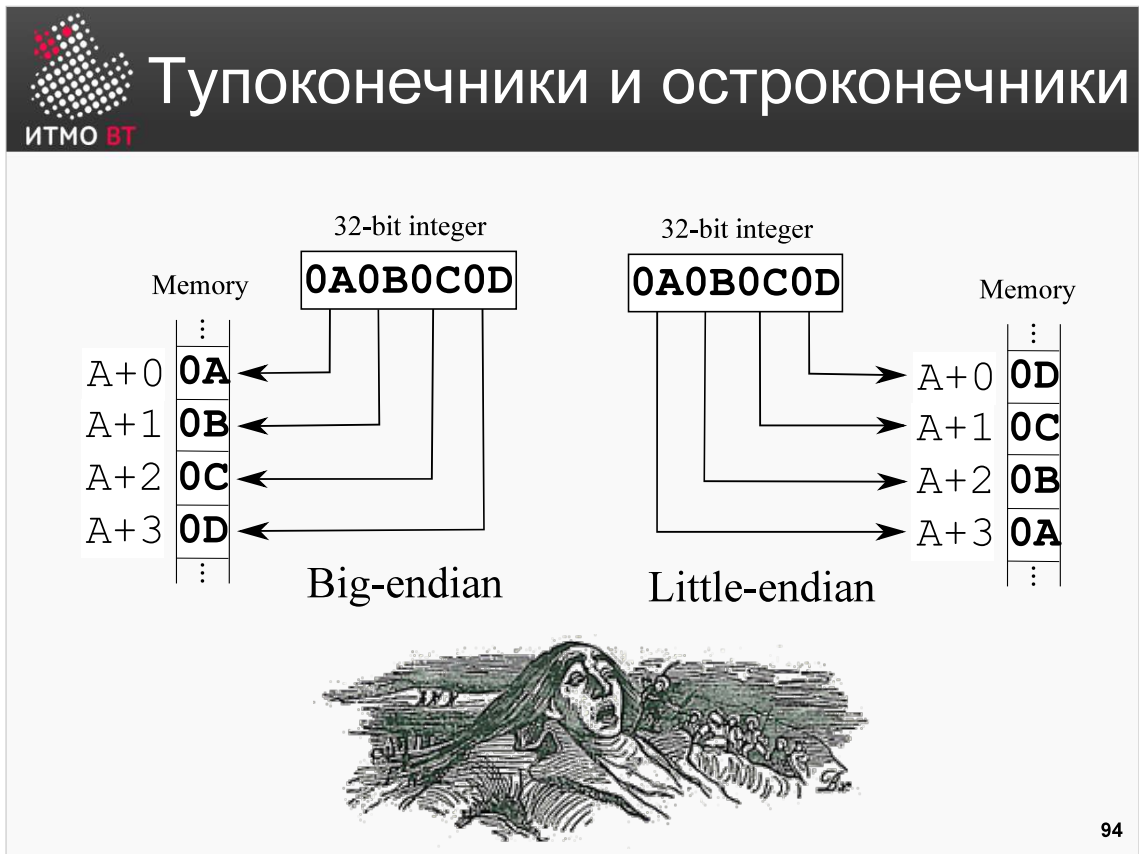
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
80	402 Ъ	403 Ѓ	201A ,	453 ґ	201E „	2026 …	2020 †	2021 ‡	20AC €	2030 ‰	409 Љ	2039 ‹	40A Њ	40C Ќ	40B Ѝ	40F Ў
90	452 ђ	2018 ‘	2019 ’	201C “	201D ”	2022 •	2013 –	2014 —	□	2122 ™	459 ъ	203A ›	45A љ	45C ѝ	45B џ	45F ѡ
A0	A0	40E Ў	45E ў	408 Ј	A4 #	490 Ґ	A6 ґ	A7 §	401 Ѓ	A9 ©	404 €	AB «	AC ¬	AD -	AE ®	407 Ґ
B0	B0 °	B1 ±	406 І	456 і	491 ґ	B5 μ	B6 ¶	B7 ·	451 ё	2116 №	454 €	BB »	458 ј	405 S	455 s	457 ї
C0	410 А	411 Б	412 В	413 Г	414 Д	415 Е	416 Ж	417 З	418 И	419 Й	41A К	41B Л	41C М	41D Н	41E О	41F П
D0	420 Р	421 С	422 Т	423 У	424 Ф	425 Х	426 Ц	427 Ч	428 Ш	429 Щ	42A Ъ	42B Ў	42C Ў	42D Э	42E Ю	42F Я
E0	430 а	431 б	432 в	433 г	434 д	435 е	436 ж	437 з	438 и	439 й	43A к	43B л	43C м	43D н	43E о	43F п
F0	440 р	441 с	442 т	443 у	444 ф	445 х	446 ц	447 ч	448 ш	449 щ	44A ъ	44B ъ	44C ъ	44D э	44E ю	44F я

Несмотря на многообразие и стандартизацию кириллических символов, компания Microsoft в операционной системе Windows использовала не кодовую страницу CP86 из своей предыдущей операционной системы DOS, а новую, которая была создана на базе кодировок, использовавшихся в ранних «самопальных» русификаторах Windows в 1990—1991 гг. совместно представителями «Параграфа», «Диалога» и российского отделения Microsoft.



Естественно войны кодировок привели к тому, что тексты в сообщениях, передаваемых между разными системами, или полученные с веб-сайтов становились нечитаемыми. На слайде представлен небольшая юмористическая блок-схема, которая помогает понять, как последовательно применить перекодировщик из разных систем, чтобы получить читаемый текст. На самостоятельное изучение предлагается дешифровка и история сообщения в заголовке слайда.

Для перекодировки символов из разных таблиц используется утилиты unix: iconv, recode и dos2unix.



Следующее важное замечание относится к преобразованию последовательности байтов в машинные слова большей разрядности. На слайде изображен Гулливер. В «Путешествиях Гулливера» описаны две великих империи — Лилипутия и Блефуску, которые на протяжении «тридцати шести лун» вели ожесточенные войны из-за спора с какого конца есть вареное яйцо — с острого или тупого. Соответственно их звали остроконечники (Little-endian) и тупоконечники (Big-endian).

В мире ВТ произошла подобная ситуация. Представьте, что у вас в регистре лежит слово, а для пересылки слова в память, вы должны расположить байты этого слова. Как удобней для ЭВМ выполнить это размещение — с младшего или со старшего байта? Этот вопрос породил большое количество споров. Приверженцев разных способов размещения по аналогии с историей из Гулливере назвали Big-endian и Little-endian.

В современных ЭВМ в основном используется Little-endian — младшие разряды слова размещаются в начальных байтах памяти. В течении эволюции ЭВМ было выяснено, что никаких преимуществ один тип кодирования над другим не имеет. До сих пор между различными архитектурами, например Intel и SPARC, возникают проблемы совместимости. Однако еще раз уточним, что данная проблема яйца выеденного не стоит =)

Представление строк (и в БЭВМ)


1) NUL terminated String

«Little-endian»

53 ₁₆	74 ₁₆	72 ₁₆	69 ₁₆	6E ₁₆	67 ₁₆	00	??
слово 1		слово 2		слово 3		слово 4	

«Big-endian»

74 ₁₆	53 ₁₆	69 ₁₆	72 ₁₆	67 ₁₆	6E ₁₆	00	??
слово 1		слово 2		слово 3		слово 4	



2) Упаковка с длиной (как в Паскале)

0006	74 ₁₆	53 ₁₆	69 ₁₆	72 ₁₆	67 ₁₆	6E ₁₆
слово 1	слово 2		слово 3		слово 4	

95

Строка в ЭВМ (String) – конечная последовательность символов заданной длины. В БЭВМ не существует поддержки строк на уровне ЭВМ или библиотек, программист должен самостоятельно организовывать их в памяти. Так как длина строки является переменной величиной, то мы должны выделять для хранения строки разное количество байт, учитывая длину строки. В современных вычислительных системах и языках программирования размещение строк обычно организовано двумя разными способами.

В первом признаком конца строки является специальный символ. Обычно используется NUL, а строки называются NULL terminated String. Такой способ применяется в языке Си и подобных ему языках программирования.

Второй способ кодировки используется в Паскаль-подобных языках, где первое слово строки используется для хранения длины строки, и мы знаем что следующие символы относятся к данной строке. Кодировка UTF-8 может доставлять дополнительную «радость» программисту и библиотекам для определения количества байт, хранимых в длине строки, и определения количества символов в ней.



Все дело в шляпе!

