# Оглавление



# Вопрос 1

**Что такое компьютерная система? Отличие информационной и управляющей системы? Почему большинство современных компьютерных систем считаются системами с преобладающей программной составляющей? Примеры. Понятие информационного процессора.** Наверх

**Компьютерная система** – любая система, оснащенная внутренними алгоритмами управления. С другой стороны, это объединение каких-то компонентов и вычислительной части в единый целостный объект. Примеры – микроволновка, светодиодные лампы, умные часы, станки и т. д.

**Информационные системы** – системы, которые получают какие-то данные извне, каким-то образом их обрабатывают и возвращают результат обработки. Направлены именно на возврат результата пользователю. Приоритет – быстрая работа (реализуется через спекулятивные вычисления, параллелизм, кластерные и облачные вычисления). Пример – почти любое прикладное ПО на нашем компе, тот же поисковик google или word.

**Управляющие системы** – системы, направленные на непосредственное взаимодействие с окружающим миром, для контроля или управления. Это происходит тоже путем получения данных, преобразования или накопления и выдачи результата. Эти системы работают в реальном времени (предсказуемо и в заданное время), встраиваются в конкретное устройство со специализацией на конкретную задачу, могут использовать автономно. Пример – лампочка с контролем яркости, управление водосбросом на ГЭС.

Большинство современных компьютерных систем – с **преобладающей программной составляющей**, потому что наибольшие затраты на разработку и поддержку системы идут именно на ПО. Почти вся их сложность воплощается именно через ПО.

**Информационный процессор** – абстрактное программируемое устройство, которое получает на вход один поток информации, на выход выдает другой. Части – процессор, хранилище и части, ответственные за ввод и вывод.

## Вопрос 2

**Системная инженерия. Понятие системы. Варианты рассмотрения систем. Модульность. Жизненный цикл. Операционное окружение и обеспечивающие системы. Заинтересованные стороны (stakeholders). Проблема передачи информации при разработке компьютерных систем.** Наверх

**Система** – совокупность частей, объединенных в единое целое, позволяющая реализовать нужный нам функционал. Системная инженерия – подход к разработке систем. Фокусируется на вовлечении в процесс представителей разных дисциплин, согласовании их работы и сборки результатов их работы в единое целое.

Системы **можно рассматривать** как совокупность частей (1), как функциональное место (2) и как жизненный цикл (3).

1) Модульность: система разбивается на подсистемы, которые в свою очередь так же разбиваются. При разработке систем мы выбираем некоторый уровень абстракции, ниже которого нам не нужно для этой разработки спускаться. Смотрим на разные вещи в зависимости от того, что мы хотим от системы.
2) Этот уровень абстракции зависит от того, в каком функциональном месте находится наша система. То есть нам безразлично, что на самом деле представляет собой часть системы, играет роль только то, в каком месте и какую функцию выполняет эта часть. Функциональное место в первую очередь определяется стейкхолдерами.
   **Стейкхолдеры** – лица, которые имеют какой-либо интерес к нашей системе, формирующие ограничения на систему. Примеры – бизнес, пользователи, вы сами.
   **Операционное окружение** – окружение, в котором система развертывается и работает. В этом окружении существует проблема или возможность, для которых система и была разработана.
3) ЖЦ системы – ее эволюция от концепции до вывода из эксплуатации. Типовой ЖЦ:
   - Концептуальный этап  (проектирование, выделение задач)
   - Разработка (документации, спецификации)
   - Производство (получение программного продукта или физического устройства)
   - Утилизация (использование)
   - Поддержка
   - Вывод из эксплуатации

   **Обеспечивающая система** – система, которая помогает целевой системе проходить через стадии ЖЦ (команда разработчиков, производство и т. д.)

**Проблемы**, связанные с информацией: формулирование (люди интуитивно понимают, но не могут зафиксировать формально), передача (разночтения документации), использование (понять и применить на практике), неполнота, неоднозначность, противоречивость.

## Вопрос 3

**Цели и задачи архитектурного проектирования компьютерных систем, его эффект. Понятие архитектуры. Различные трактовки (Гради Буч, ISO 42010 и др.) и их практическая значимость.** Наверх

У **архитектуры** есть несколько трактовок:

1) Гради Буч: это логическая и физическая структура компонентов и их взаимосвязи, сформированные решениями, применяемыми во время разработки. Логический взгляд учитывает концепции, созданные в концептуальной модели и устанавливает

существование и роль ключевых механизмов и абстракций. Физическая модель описывает конкретный программный и аппаратный состав реализации. Трактовка – взгляд со стороны программной инженерии.

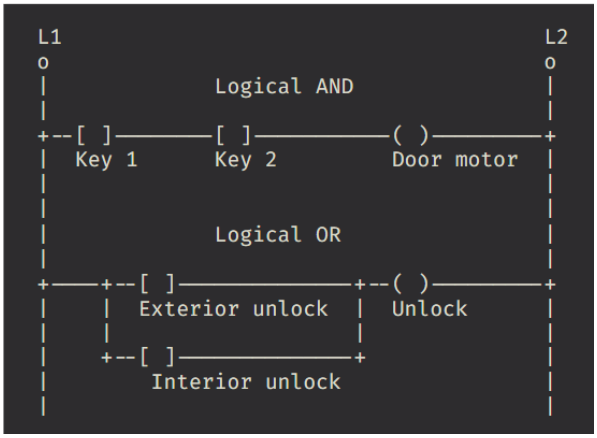
2) ISO 42010: это фундаментальная организация или свойства системы, воплощенные в ее элементах, взаимосвязях и принципах проектирования и развития. Взгляд со стороны инженерии систем в целом, не только программных.
3) Еще определение: это набор решений, ошибка в которых приведет к провалу проекта.

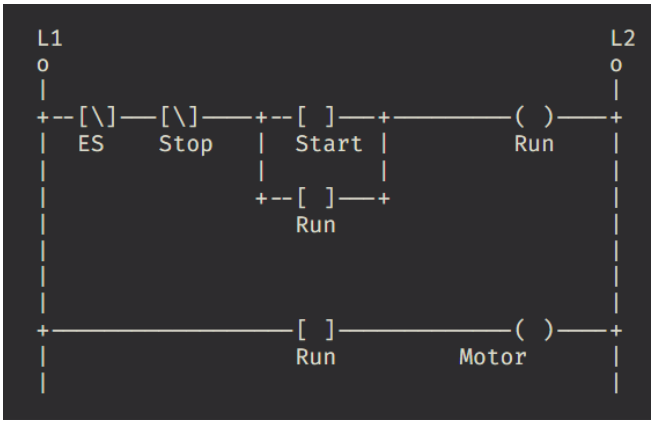
**Цели и задачи** проектирования: определить конечные требования и возможные риски как можно раньше, так как чем позже мы будем менять что-то в системе, тем дольше и дороже это выйдет. Эффект проектирования такой, что фактические затраты на проект становятся более предсказуемыми и сильно не превышают запланированные.

## Вопрос 4

**Реле как базис компьютерной системы. Область применения и принципы построения систем на базе реле. Примеры релейных схем и принцип их функционирования. Программируемые логические контроллеры (ПЛК). Область применения. Особенности аппаратуры и программирования по сравнению с компьютерами общего назначения.**
Наверх

**Реле** – база для компьютеров до появления транзисторов. Больше, чем транзистор, потребляет больше энергии. Простое реле – магнитная катушка и проводник, который катушка примагничивает при подаче управляющего сигнала. Таким образом выбирается, какую цепь замкнет проводник. Есть другие виды реле: тепловые, оптические и т. д. Область применения: когда-то делали компьютеры, сейчас в силовых шкафах управления всякими процессами типа двигателей, в неблагоприятной среде эксплуатации для полупроводников и в ПЛК. Примеры схем: базис И-ИЛИ, схема двигателя с аварийной остановкой.




**ПЛК** - цифровая электронная система, предназначенная для **применения** в производственной среде (управляющие системы на производственных линиях), которая использует программируемую память для внутреннего хранения ориентированных на потребителя инструкций по реализации специальных функций. Преимущества по сравнению с релейными схемами: меньший размер и энергопотребление, выше надежность, можно изменить логику после этапа проектирования через программирование, легче настраивать и поддерживать. **Отличия от компьютеров**:

- ориентированы на управляющие системы (в реальном времени);
- ПЛК – законченное устройство для встроенного использования;
- специализированный ввод-вывод;

- разделение между инструментальной составляющей (программированием) и железом;
- эксплуатация в тяжелых условиях

Типовое устройство: вычислительный блок (CPU, память), модули ввода и вывода, разделенные с вычислительным блоком физической защитой (например, оптической изоляцией) и инструментальный компьютер, с которого ПЛК программируют. Программируется на специальных языках, ориентированных на удобство обслуживания и замены ПЛК (например, executable документация).

## Вопрос 5

**Принципы кодирования информации в компьютерных системах. Двоичный код, достоинства и недостатки. Машинное слово и адресация. Код грея, BCD, Base64, Base58.** **Наверх**

В двоичном кодировании фиксируются 2 уровня тока: высокий и низкий (один уровень может трактоваться как 0 или как 1). Допустимые интервалы на вход шире, чем на выход, для учета всяких помех. Также есть Forbidden Zone, при попадании сигнала в которую мы не можем быть точно уверены, 0 или 1 взять. Почему за низкий уровень берется не 0 – чтобы можно было определить, нулевой это сигнал или обрыв линии, также провод  с 0 больше подвержен помехам.

**Достоинства** двоичного кодирования:

1) надежное и помехоустойчивое
2) простая схемотехника для реализации арифметики
3) диапазон и точность наращиваются разрядностью
4) погрешности “by design”, то есть предсказуемая погрешность вычислений у цифровых устройств. Мы точно знаем, с какой точностью посчитаем значение, в аналоговых погрешность может плавать от экземпляра к экземпляру из-за разных факторов

**Недостатки**:

1) нечитаемое представление
2) проблема с арифметикой с плавающей запятой (неточный перевод из 2 в 10 систему)
3) дискретное кодирование (нельзя сделать абсолютно точную схему)

**Машинное слово** – единица данных, естественная для обработки вычислителем. Например, если нам надо прочитать 1 бит, мы все равно возьмем из памяти машинное слово и из него потом вытащим 1 бит. То есть объем информации, с которой мы работаем, округляется вверх до целых машинных слов. **Адресация** происходит именно с округлением по машинным словам.

**Код Грея** – соседние значения отличаются только 1 битом. Придуман для помехоустойчивости (если исказится 1 бит, число изменится не сильно, а в обычном двоичном кодировании все зависит от того, какой именно бит искажен).

**BCD** – двоично-десятичный код, используется для разных индикаторов или чтобы убрать погрешности, специфичные для двоичного кода.

**Base64** – стандарт кодирования двоичной информации только 64-мя символами ASCII. Им, например, передаются картинки по HTTP.

**Base58** – тот же Base64, но без символов, которые могут быть неоднозначно прочитаны (0 или O). В конце записи имеет хэш сумму, используется в криптовалюте.

## Вопрос 6

**Что такое комбинационная схема? Построение через таблицу истинности и алгоритмизацию. Состояние и параллелизм, переходные процессы. Состояние x и z. Особенности поведения комбинационных схем по сравнению с программами. Реализация "условного оператора". Программируемые логические интегральные схемы (ПЛИС) и их устройство.** Наверх

**Комбинационные схемы** – схемы, составленные из набора логических элементов, реализующая таблицу истинности. Простейшие элементы – AND, OR, NOT, BUF (для восстановления сигнала). Построение **через таблицу** истинности: берем все значения, в которых определена функция и соответствующие значения, которые она должна возвращать, составляем таблицу. Записываем это в КНФ или ДНФ и минимизируем. Построение **через алгоритмизацию**: на основе понимания закономерности функции, взаимосвязи входа и выхода. Может быть каскадной (например, сумматор).

Свойства комб. схем:

1) при корректном входе устанавливается стабильное **состояние** (нет циклов), и устанавливается оно через какую-то задержку после изменения входных значений (задержка зависит от условий окружающей среды, от того, как построена схема)
2) все, что может работать **параллельно** (параллельные логические пути), работает параллельно
3) может накапливаться ошибка на физическом уровне (исправляется буферами)

**Состояние** в комбинационных схемах хранится с помощью триггеров. Эти устройства могут длительно находиться в одном из двух состояний и чередовать их под воздействием сигналов. Позволяют ограничить распространение сигнала по схеме или зафиксировать состояние линии на длительное время.

Кроме 0 и 1 есть еще два состояния: x и z. **z** – отключено, то есть сигнал вообще не подается. Отдельное состояние, потому что 0 обычно не кодируют нулевым уровнем сигнала. **x** – произвольное значение, таблица истинности определена частично. При неопределенном аргументе мы получим случайное значение. Пример – деление на 0, шифратор 4->2.

**Сравнение КС с программами**: все процессы между регистрами – параллельно; передача сигнала всегда идет, если есть контакт и питание; система всегда работает, пока есть питание; почти всегда есть неопределенные значения.

**Условный оператор**: реализуется с помощью мультиплексора. Допустим, есть 2 варианта расчетов. И в схемотехнике нет возможности посчитать только один из них. Мы считаем оба и выбираем нужный результат с помощью мультиплексора.

**ПЛИС** – настраиваемая микросхема, придуманная для того, чтобы каждый раз физически не производить комбинационную схему. Состоит из программируемого интерконнекта (мультиплексор, позволяющий связывать линии и отдельные элементы). Отдельные элементы – логические блоки (комбинационные схемы), через которые можно реализовать любой физический блок (таблицы истинности). Также есть регистры, в которые можно защелкивать или не защелкивать значения. Тем самым можно реализовать комбинационную схему с состоянием.

# Вопрос 7

**2-этапное производство. Понятия Hardware и Software, их свойства. Сравнение с понятиями программного и аппаратного обеспечения. Проблемы, специфичные для аппаратного обеспечения: производство, эксплуатация, устаревание. Принципы совместного проектирование (HW/SW CoDesign).** Наверх

**2-этапное производство** - разбиваем производственную цепочку на 2 этапа:

1) производство универсальной компьютерной системы (hardware), не вносим в нее изменения, не адаптируем под наши задачи. Переиспользуем ее, адаптируя ПО.
2) настройка прикладного поведения (software) с помощью ПО.

То есть у нас экземпляр универсальной системы и схема прикладной конфигурации соединяются в экземпляр сконфигурированной компьютерной системы. Нужно заложить в аппаратуру возможности конфигурирования и адаптации и задокументировать их. Незапланированные возможности – дыры в безопасности.

**Software** – та часть компьютерной системы, которую легко изменить и адаптировать (легко применить изменения, а не понять какие), а **Hardware** – не подлежащая каким-то значимым изменениям. **Аппаратное обеспечение** – это обязательно какие-то физические устройства или их части, которые имеют производственную линию, то есть именно сами железки. **Программное обеспечение** – виртуальная сущность, данные, которые позволяют аппаратному обеспечению выполнять вычисления или функции управления. Примеры о различиях: процессорное производство и Minix на инструментальных процессорах (ПО, но Hardware) или наоборот, виртуализация типа Docker (оперируем аппаратными терминами, но на самом деле это Software, легко изменить).

Проблемы **производства** аппаратного обеспечения: логистика, склады, специалисты, производственная цепочка, тестирование (долго и тщательно, каждый экземпляр), упаковка, дистрибуция, гарантии.

Проблемы **эксплуатации**: амортизация (аппаратное обеспечение деградирует со временем), необходимость физического доступа, особенности среды эксплуатации (обеспечение условий), долгосрочные эффекты (например, оловянные нити на проводниках). Также сложно вносить изменения из-за необходимости физического контакта или из-за длинного производственного цикла.

Проблемы **устаревания**: в пределах срока службы надежность аппаратуры деградирует, тратятся ресурсы на обслуживание. Нужно где-то хранить комплектующие (дорого, и их запас не бесконечный). Эти комплектующие тоже устаревают и больше не производятся из-за низкого спроса, отсюда проблемы с тем, чтобы их достать. Также выходит из эксплуатации и сопутствующее оборудование (дисководы например).

**HW/SW CoDesign** – подход к проектированию систем, когда мы думаем о системе в общем представлении и сначала продумываем, как система будет функционировать, как она должна быть построена, и только потом разбить ее на HW и SW. Первый этап – разработка исполняемой спецификации, которые как-то могут имитировать работу. Потом одновременно начинаем разработку HW и SW, с оглядкой на спецификацию. Таким образом получается более распараллеленная разработка и меньшие временные затраты.

## Вопрос 8

**Понятие модели вычислений (МоС). Сопоставление понятия парадигмы программирования и МоС. Примеры МоС (последовательные, параллельные, функциональные) и их роль в разработке компьютерных систем. Использование МоС в разных вычислительных платформах.** Наверх

**Модель вычислений** – специально выбранные математические абстракции для описания вычислительных процессов. Описывает, как организованы элементы вычислений, памяти и связи.

**Последовательные** модели: описывают вычисления как последовательность переходов из состояния в состояние. Пример – конечные автоматы, машины Тьюринга, фон Неймана (процессор, в который приходят инструкции, каждая из которых переводит его из одного состояния в другое).

**Функциональные** модели: описывают вычисления математически, как некоторое символьное выражение, которое фиксирует текущее состояние и все возможные будущие. Пример – лямбда-исчисление. Используется, когда нужно доказать математические свойства системы.

**Параллельные** модели: применяются для систем, которые включают несколько асинхронно взаимодействующих процессов.

**Модель вычислений** – инструментарий, который позволяет оперировать вычислениями как математическим объектом. Направлена на понимание вычислительного процесса. А **парадигма программирования** – инструменты дизайна для программиста, для реализации компьютерной системы. Направлена на облегчение жизни программиста.

МоС **используются** в: computer science и формальных моделях, дизайне ЯП, управлении сложностью, ограничении процесса разработки (например статическая или динамическая типизация).

## Вопрос 9

**Универсальный процессор и его свойства. Машина Тьюринга и полнота по Тьюрингу. Виды процессоров (СБИС, FPGA, CGRA, GPU, DSP, CPU) и их сопоставление с точки зрения универсальности и эффективности.** Наверх

**Универсальный процессор** – процессор с возможностью донастройки для решения широкого круга задач. Не обязательно тьюринг-полный, не обязательно универсальный на практике.

**Свойства**: 2-этапное производство, полнота по Тьюрингу, отсутствие серьезных ограничений на объем программы, изменяемость ПО.

**Машина Тьюринга** - абстрактная математическая модель вычислительного устройства. Используется для формализации понятия алгоритма и вычислимости. Состоит из ленты (памяти), головки чтения-записи (процессора), таблицы инструкций (программа, язык) и конечного множества состояний. Работа машины Тьюринга заключается в последовательном выполнении инструкций из таблицы до тех пор, пока она не достигнет конечного состояния.

Если система **полна по Тьюрингу**, она может выполнять любые вычисления, которые может выполнить машина Тьюринга, при наличии достаточного времени и памяти.

- СБИС (ASIC) – для очень эффективного выполнения конкретных задач, узко специализированы.
- CGRA, FPGA – состоят из множества функциональных блоков с коммуникацией между ними.
- GPU – графический процессор, много параллельных однотипных вычислений
- DSP – для обработки цифровых сигналов в реальном времени
- CPU – процессор общего назначения

Расположены в порядке возрастания гибкости и убывания энергоэффективности и производительности.

## Вопрос 10

**Архитектура фон Неймана. Принципы. Свойства. Особенности и ограничения. Применение на практике. Машинное слово. Понятие системы команд и её роль в построении процессоров. Control Unit и DataPath.** Наверх

**Машина фон Неймана** – развитие машины Тьюринга, где ленту заменили на RAM, а инструкции и данные объединили в единую память. CPU читает/пишет в память, выполняет вычисления. Также остаются ввод и вывод. Простая система, оставляет всю сложность на ПО.

**Особенности**:

1) использование двоичного кодирования
2) последовательное выполнение команд
3) память однородно хранит данные и программы
4) ячейки памяти имеют адреса
5) возможность условного перехода

Сегодня чистая архитектура фон Неймана не используется на **практике**. Но ее влияние на индустрию очень трудно переоценить. Ее элементы присутствуют в подавляющем большинстве процессоров и средств разработки.

**Машинное слово** – фрагмент данных фиксированного размера, обрабатываемый процессором как единое целое.

**Система команд** процессора – описание действий, которыми мы можем менять состояние процессора и получать нужные результаты. Формирует интерфейс между ПО и процессором. Затрагивает типы данных, регистры, адресацию, модели памяти, инструкции, обработку прерываний и исключений, ввод и вывод. Не касается вопросов производительности, энергопотребления и задержек.

Виды инструкций: работа с памятью (запись, чтение регистров и памяти), арифметические и логические операции, управляющие операции (переходы), инструкции для сопроцессоров.

**Control Unit** – блок управления CPU, задача которого – управление всем процессором.
**DataPath** – часть CPU, по которой идут обрабатываемые данные. Может считывать данные, преобразовать их и записать, но само действие определяется Control Unit'ом. Unit отправляет управляющие сигналы DataPath, получает в ответ статусные сигналы, чтобы понять, какую инструкцию выполнять следующей.