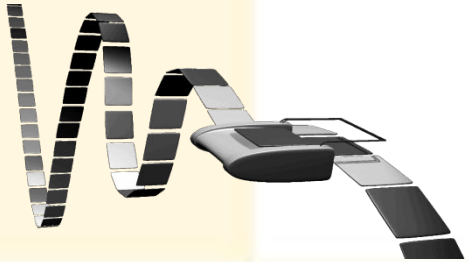

Семинар 1, 2022

# ЭВМ (компьютеры)

- предназначены для выполнения программ – различных (***любых***) алгоритмов, описанных на языках программирования
- отличие от конечных автоматов в том, что компьютер исполняет ***«любые»*** программы (не только заранее определенные функции)

Компьютер – реализация идеи «Машины Тьюринга».

# «Машина Тьюринга»



Машина Тьюринга – алгоритм – вычислимость

- была предложена для формализации понятия алгоритма
- является расширением конечного автомата;
- абстрактный исполнитель (абстрактная вычислительная машина)

Любая **задача может быть решена**, если достаточно ресурсов (памяти и времени)

# УВМ (СВТ, ПЭВМ)

- частично (с конечной памятью) моделируют машину Тьюринга, давая псевдо неограниченные возможности и толкая на экстенсивный путь развития

- не хватает памяти – добавим. Не хватает времени – увеличим тактовую частоту, количество ядер, виртуализируем ресурсы

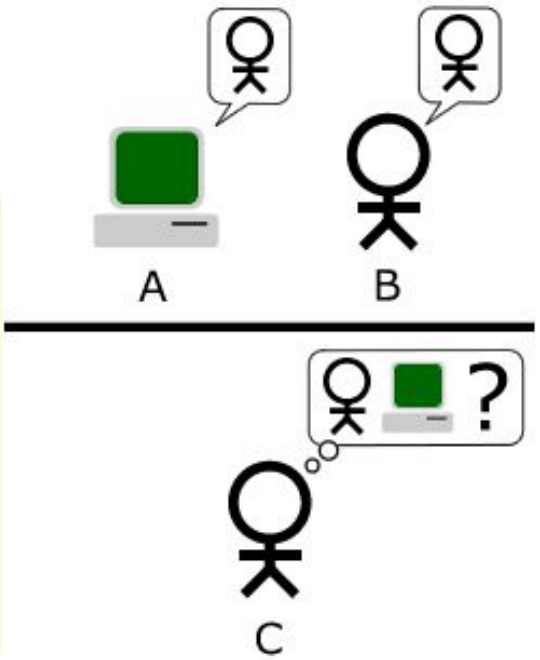
# Полнота по Тьюрингу

- характеристика исполнителя (множества вычисляющих элементов) в теории вычислимостихарактеристика исполнителя (множества вычисляющих элементов) в теории вычислимости, означающая возможность реализовать на нём любую вычислимую функцию

- с ограничениями (конечность памяти) на УМ можно моделировать МТ – УМ полные по Тьюрингу – должны выполнять элементарные операции, свойственные МТ

# Самообучение УМ

**Тест по Тьюрингу**




- УМ потенциально может **самообучаться**;
- недопустимо *бесконтрольное самообучение* контроллеров АСУ ТП АЭС, ж/д транспорта, непрерывных производств – для них применяют конечные автоматы (являются неполными по Тьюрингу);

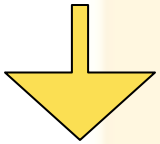
Существует много задач, которые *нужно* решать *не на универсальном*, а на **специализированном** «исполнителе».

# Уязвимость машины Тьюринга

- УМ выполняет «любые» программы, значит, выполнит и **вредоносную программу**;
- Универсальность обеспечивается архитектурой УМ;
- уязвимость – оборотная сторона универсальности – **МТ архитектурно уязвима**;
- все компьютеры потенциально уязвимы (плата за универсальность);

# Уязвимость машины Тьюринга

Архитектуру **нельзя** изменить программным путем

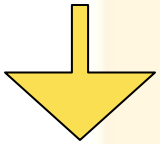


Никакие **программные средства** не помогут защититься от хакеров надежно

**Как же быть?**

# Уязвимость машины Тьюринга

Архитектуру **нельзя** изменить программным путем



Никакие **программные средства** не помогут защититься от хакеров надежно

**Если уязвимость в архитектуре – то и совершенствовать нужно архитектуру.**

# Классические архитектуры

- архитектура фон-Неймана
(настольные компьютеры)

- гарвардская архитектура
(планшетные компьютеры и телефоны)

# Принципы фон-Неймана организации вычислительного процесса (П1-П3)

1. Использование **двоичной системы счисления** в вычислительных машинах.
2. Программное управление ЭВМ. Работа ЭВМ контролируется программой, состоящей из набора команд. **Команды выполняются последовательно.**
3. Память компьютера используется **не только для хранения данных, но и программ**. При этом и команды программы и данные кодируются в двоичной системе, т.е. их способ записи одинаков. Поэтому в определенных ситуациях над командами можно выполнять те же действия, что и над данными.

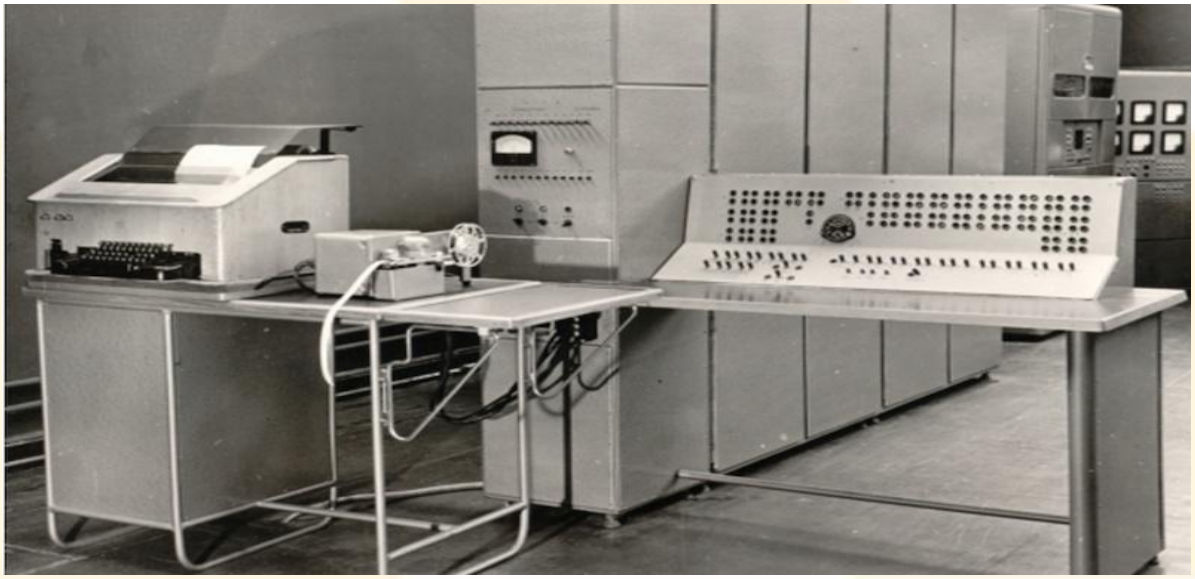
# Принципы фон-неймана организации вычислительного процесса (П4-П5)

4. Ячейки памяти ЭВМ имеют **адреса**, которые ***последовательно*** **пронумерованы**. В любой момент можно обратиться к любой ячейке памяти по ее адресу.
5. Возможность **условного перехода** в процессе выполнения программы. Несмотря на то, что команды выполняются последовательно, в программах можно реализовать возможность перехода к любому участку кода.

# ЭВМ «Сетунь» (1958-61гг.)

- нарушение П1 – троичная система счисления с коэффициентами (1, 0, -1) - возможно естественное представление натурального ряда чисел со знаком;
- преимущество в скорости операций и их энергоемкости.

В сумматоре перенос в следующий разряд – в 8 ситуациях из 27 ~~(в 4 из 8)~~; умножение на -1 инвертирует множимое.



# ЭВМ МИР-1 (1967г.)

- нарушение П2, П3 – язык высокого уровня («Аналитик»), непосредственно исполняемый машиной;
- разработчики реализовали все стандартные рекурсивные функции, включая их в состав языка;
- пришлось добавить в язык оператор перехода.

# Макроконвеер (1974г.)

- нарушение П2, П3 – суть принципа макроконвейерной обработки данных:

✔ ЭВМ содержит не один, а **много процессоров**,

и

✔ каждому процессору на очередном шаге вычислений дается такое задание, которое позволяет ему длительное время работать **автономно без взаимодействия с другими процессорами**.

# Аккорд-СБ

- макроконвейерные многостековые сопроцессоры безопасности;
- применялись для подписи и проверки на всех этапах обработки финансовых документов;
- решали задачу увеличения производительности без использования ресурсов основной ЭВМ;

Принцип макроконвейера решил проблему медленной шины ЭВМ – вначале загружалась задача, перестраивалась архитектура вычислителя, потом отправлялась пачка данных, и пока они обрабатывались – в стек загружались следующие данные.

# ЭВМ B5000 (1961 г.)

- нарушение П2, П3 – двухпроцессорный компьютер с виртуальной памятью;
- особенности:

✔ адресация на основе дескрипторов (каждое слово содержит не только информационную, но и управляющую часть – тег элемента, что позволяет снизить количество ошибок);

✔ использование языка высокого уровня (Алгол) как входного языка, контролируя операнды.

# Важность идей, заложенных в ЭВМ В5000 и МИР

- впервые опробован механизм динамического изменения структуры ЭВМ в соответствии с исполняемой программой и показано, что при разработке ЭВМ ***необходимо понимать, какие программы будут на ней исполняться***;

- создавая программное обеспечение, ***необходимо понимать архитектуру компьютера.***

# ЭВМ «Эльбрус» (1961 г.)

- особенности – использование Алгола в качестве управляющего языка и теговая организация памяти;
- отдельные операции могут выполняться независимо одна от другой, вычислительные ресурсы распределяются аппаратно.

# ПС-2000 (1972-1975 г.)

- многопроцессорная машина с одним потоком команд и многими потоками данных;
- управление состоянием ПС-2000 осуществлялось ЭВМ СМ-2М.
- ПС – «перестраиваемые структуры».

# МВК ПС-3000 (1979г.)

- в полной мере были реализованы идеи динамической перестраиваемости структуры;
- основные архитектурные и структурные принципы организации:

динамическая перестраиваемость его структуры по текущим требованиям параллельных вычислительных процессов.

Перераспределение ресурсов осуществлялось как программно, так и аппаратно, оптимизируя структуру комплекса под текущую задачу.

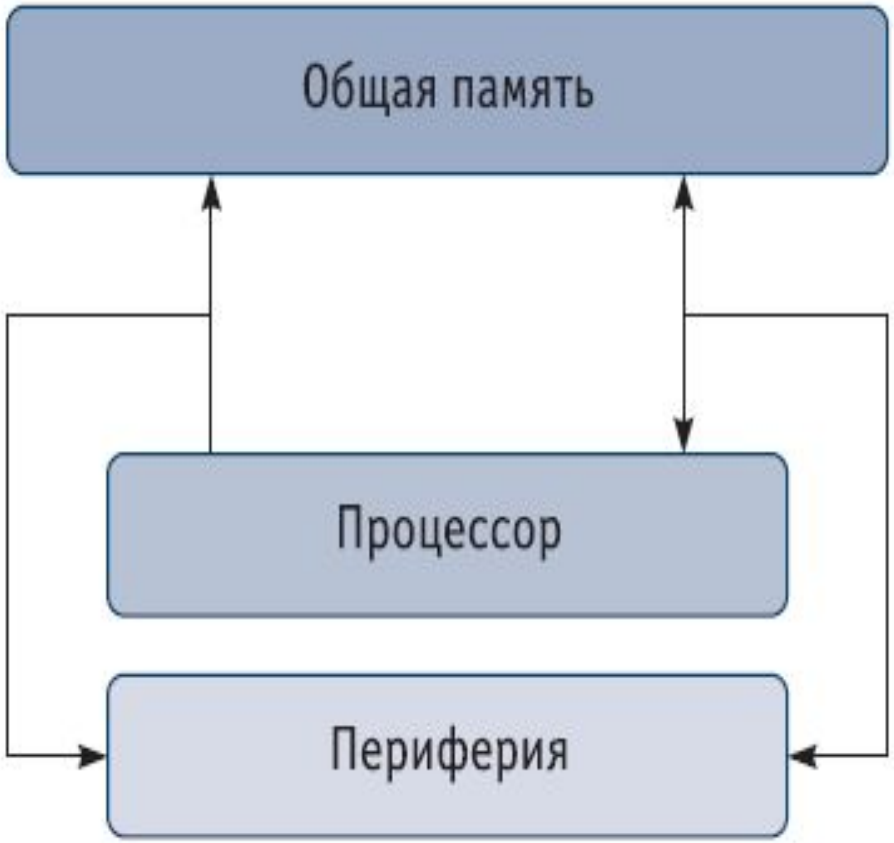
# Усовершенствование архитектуры компьютера

При разработке компьютера главное – понять, **какая часть функций должна быть реализована** *аппаратно*, а какая – *программно:*

- ✔ в аппаратную часть нужно включать то, что: снижает стоимость, редко изменяется, расширяет возможности и используется постоянно
- ✔ в процессе работы структура компьютера может динамически изменяться (структура на 1 этапе – конечный автомат, на 2 этапе – «универсальный исполнитель» по Тьюрингу)

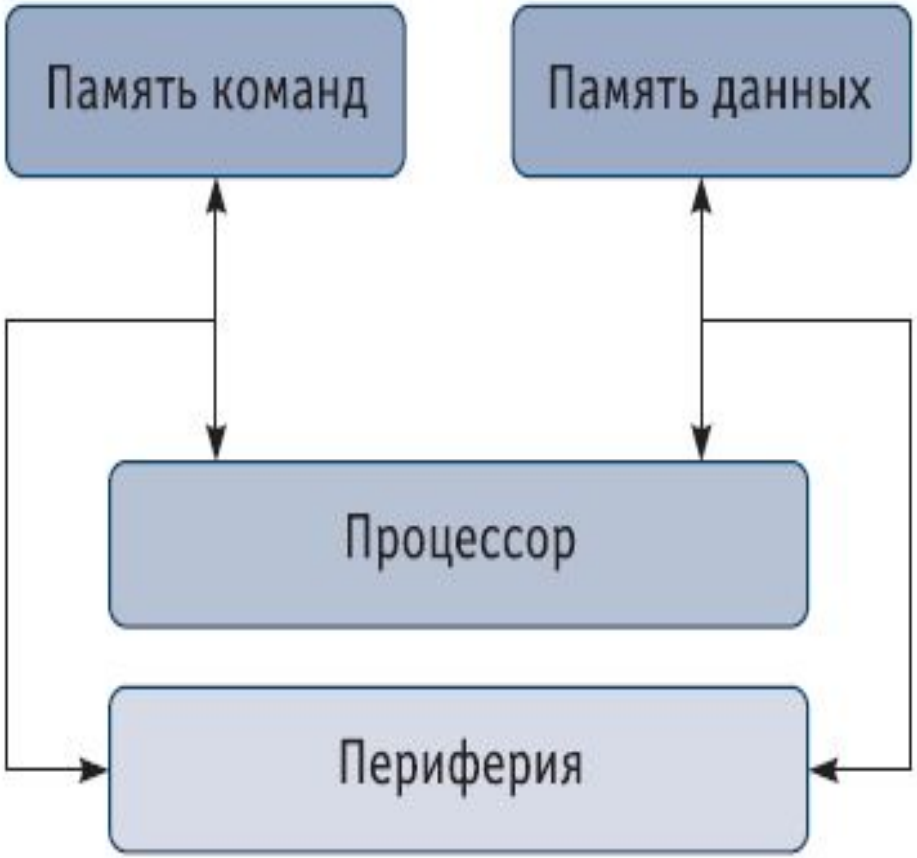
# Архитектура фон-Неймана

- команды и данные не разделяются
  (передаются по единому общему каналу)



# Гарвардская архитектура

- предполагает наличие разных каналов для команд и данных




- требует более сложной организации процессора
- обеспечивает более высокое быстродействие (потоки команд и данных параллельные)

# Архитектурная уязвимость

- гибкость, универсальность обеспечивается возможностью изменения последовательности команд и данных (двунаправленные стрелки от процессора к памяти)
- это создает возможность для несанкционированного вмешательства вредоносного ПО (ВрПО)

На использовании этой уязвимости основаны хакерские атаки:

основная атака **«перехват управления»**

# Схема атаки «перехват управления»

| Шаг | Действие |
|---|---|
| s1 | Внедряется и размещается в ОЗУ вредоносное ПО (ВрПО) |
| s2 | Внедряется и размещается в ОЗУ вредоносный обработчик прерываний |
| s3 | Записывается в долговременную память ВрПО и обработчик прерываний |
| s4 | С помощью любого доступного механизма вызывается прерывание (например, с помощью DDOS-атаки) |
| s5 | Внедренный ранее обработчик прерываний срабатывает, и передает управление ВрПО; |
| s6 | ВрПО выполняет свою функцию, например, реализует разрушающее программное воздействие (РПВ) |

# Блокирование и обезвреживание атаки «перехват управления»

- обезвреживание **s1 и s2** – антивирусные программы
- блокирование последствий выполнения **s3** – при последующей загрузке с помощью механизмов контроля целостности (контролируется неизменность данных)
- блокирование генерации события на **s4** – частично с помощью средств анализа трафика (сетевых или на клиентских компьютерах)
- блокирование следствий **s5 и s6** – с помощью механизмов контроля запуска задач (процессов, потоков)

# ПАК «Аккорд» (АМДЗ с ПО разграничения доступа)

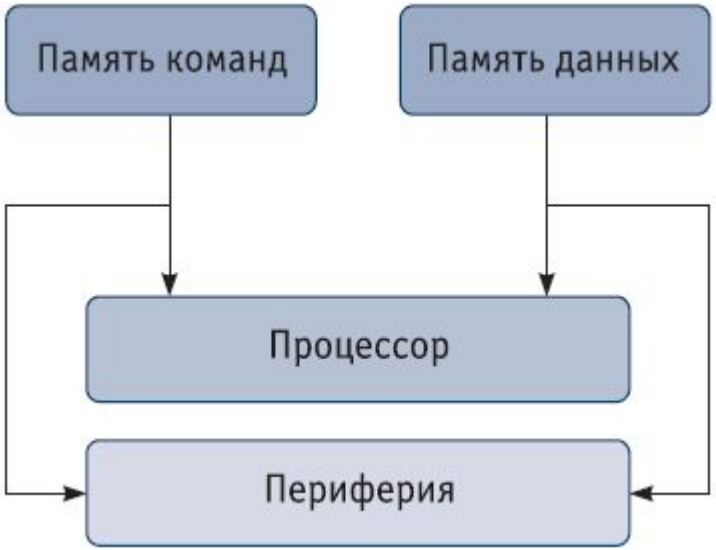
- АМДЗ выполняет контрольные функции, а ПО контролирует и запуск задач
- предназначен для работы на ПК x86 (архитектура близка к фон-Неймановской)
- блокирует уязвимости, связанные с нарушением целостности, создает доверенную среду для работы ПО, обеспечивающего защиту компьютера на s1 – s6
- цена довольно высока, настройка сложна (лучшее решение для корпоративных применений, но сложен для частного)

Сложность связана с фон-Неймановской архитектурой защищаемого ПК

нужно ***добавить неизменяемую память, разделить потоки команд и данных, исполнить контрольные процедуры в доверенной среде до запуска ОС*** и т.д.

# Гарвардская архитектура с памятью RO

- нужно сделать память **неизменяемой (RO)** (не нужно использовать сложные механизмы КЦ программ и данных до старта ОС)
- контрольные процедуры исполнять под управлением проверенной и неизменяемой ОС




Эти функции легко реализовать, если обеспечить движение команд и данных **только в одном направлении – из памяти в процессор**

- архитектура обеспечивает неизменность ОС, программ и данных.

# Гарвардская архитектура с памятью RO

**+** **s3** не может быть выполнен, поэтому и сама атака (шаги **s5 и s6**) тоже не исполнятся

↓

ПК приобретет **«вирусный иммунитет»**

(ВрПО не будет фиксироваться)

**–** придется дорабатывать практически все ПО

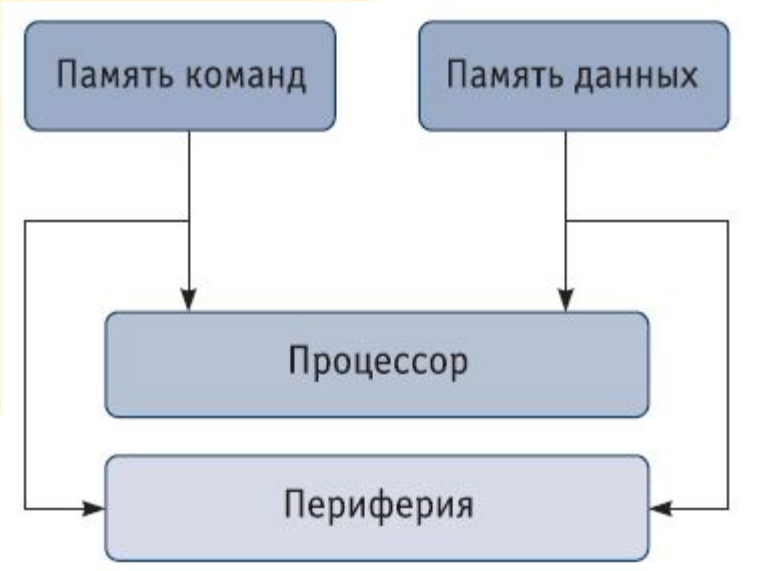
(существующее ПО использует операций записи в память)

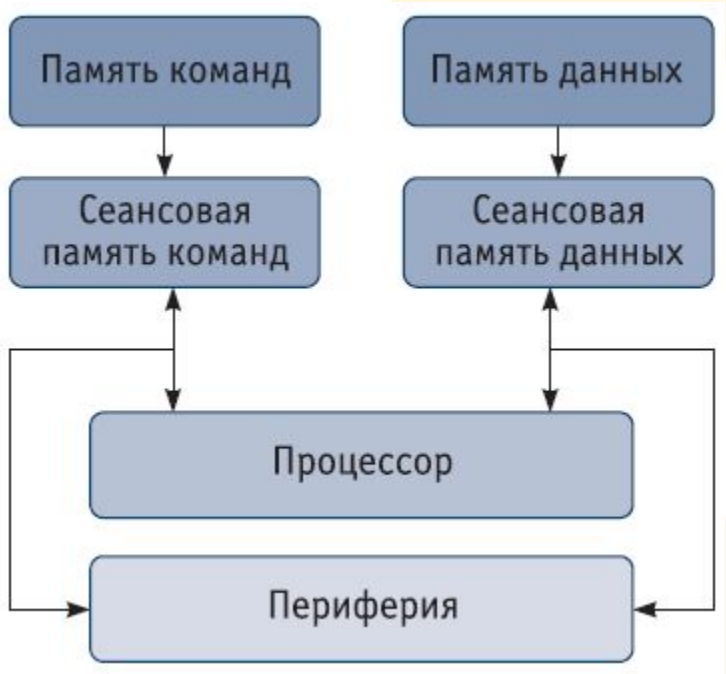
↓

!!! Предложенную архитектуру необходимо дополнить блоками ***сеансовой памяти*** – в которой будут исполняться программы

# Гарвардская архитектура с сеансовой памятью

- архитектура компьютера будет отличаться на разных этапах

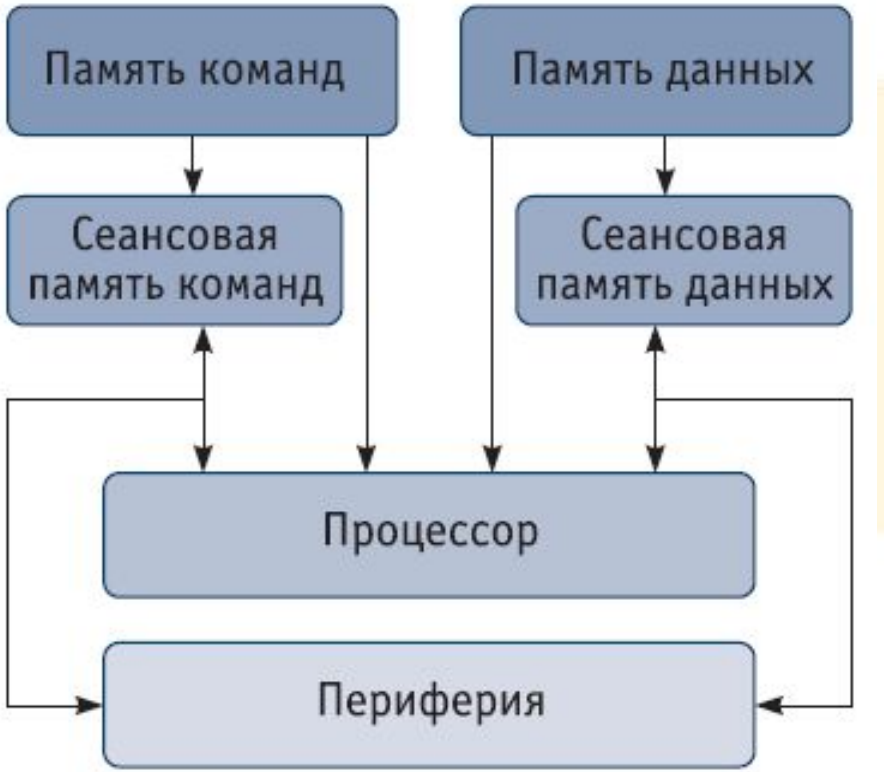



- архитектура изменяется от этапа начальной загрузки к этапу функционирования

# Новая гарвардская архитектура

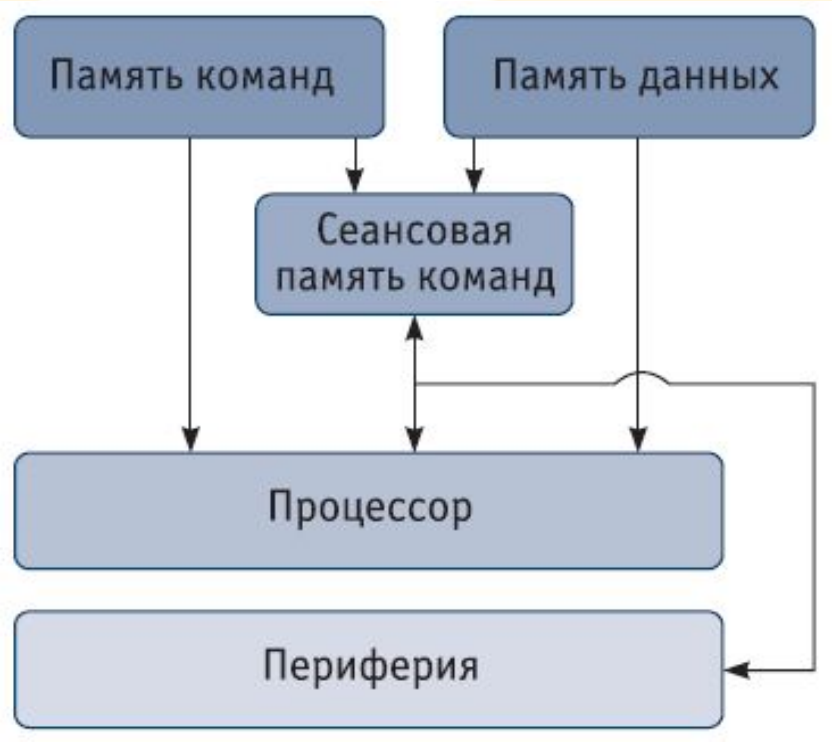
- **изменяемая архитектура** гарвардского типа




- отличие: в архитектуре используется память, для которой установлен режим **RO**

# Новая гарвардская архитектура с общей сеансовой памятью

- при загрузке команды и данные размещаются в сеансовой памяти, в которой и исполняются
- начальная загрузка и копирование кодов в сеансовую память могут выполняться последовательно и параллельно



# Особенности Новой гарвардской архитектуры

- *динамически* изменяемая архитектура:

✔ защищенность и эффективность

✔ неизменность ОС

✔ «вирусный иммунитет»

✔ возможность применения адаптированных стандартных ОС и ПО

- нарушение **П4** – память команд и память данных не доступны на запись, нумерация ячеек этой памяти и сеансовой памяти нельзя считать «последовательной»
- нарушение **П5** – возможен условный переход в пределах сеансовой памяти и невозможен – в защищенной памяти

# Возможности Новой гарвардской архитектуры

- высокий уровень «вирусного иммунитета»
- возможность создания и поддержки доверенной среды
- возможность использовать все ранее наработанное ПО (в рамках ограничений, накладываемых ОС – например, в Linux есть проблемы с видео, которых нет в Android, но это уже вопрос не к архитектуре компьютера)

# Направления устранения уязвимостей компьютерной техники

1. Усовершенствовать архитектуру уже существующих технических средств

2. Использовать новые технические средства на базе новой, более совершенной архитектуры

# Устройства с совершенной архитектурой

# Микрокомпьютеры с Новой гарвардской архитектурой

**Ответственный разработчик:**

Батраков Антон Юрьевич,
начальник отдела инновационных разработок

**МК – аппаратные платформы:**

✔m-Trust

✔*MKT-card и MKT-card long**

✔*TrusTPad**

* – история НГА

# ТРЕБОВАНИЯ К СЗИ ДЛЯ КИИ

КИИ состоит из совокупностей:

- ✔ ПКО
- ✔ каналов связи (для передачи информационных и управляющих сигналов)

=> **СЗИ для *уже функционирующих* КИИ должны обеспечивать**:

- ✔ криптографическую защиту информации о состоянии ПКО и управляющих сигналов для ИС
- ✔ информ. взаимодействие с ПКО (USB, Ethernet, и др.)
- ✔ возможность использования стандартных цифровых каналов (WiFi, BlueTooth, и др.)
- ✔ информ. взаимодействие с каналообразующей аппаратурой (RS232, RS435 и др.)

# ЗАЩИЩЕННЫЙ МИКРОКОМПЬЮТЕР «M-TRUST»

«вирусный иммунитет»

запатентованное отечественное решение

достаточная производительность при низкой цене

встроенные средства защиты информации имеют сертификаты ФСТЭК России и ФСБ России

# ЗАЩИЩЕННЫЙ МИКРОКОМПЬЮТЕР «M-TRUST»

Микрокомпьютер «m-TrusT»

Интерфейсная плата с подключенным «m-TrusT»

Интерфейсная плата («облегченный» вариант) с подключенным «m-TrusT»

Интерфейсные платы

«m-TrusT» сервер

# ИСТОРИЯ НГА*: МКТ-CARD LONG И TRUSTPAD

Микрокомпьютер
«МКТ-card long» с док-станцией

Микрокомпьютер
«МКТ-card long»

Планшет «TrusTPad»

* – МК не производятся, но их выпуск может быть возобновлен по желанию Заказчика


Семинар 1, 2022