Осик

**№7**

Архитектуры компьютера: Фон - Неймана и Гарвардская архитектура. Сравнительный анализ.

**Архитектура фон Неймана** (модель фон Неймана, [Принстонская](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%81%D1%82%D0%BE%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82) архитектура) — широко известный принцип совместного хранения [команд](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D1%8F_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) и [данных](https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5_(%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0)) в [памяти компьютера](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C). [Вычислительные машины](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) такого рода часто обозначают термином «машина фон Неймана», однако соответствие этих понятий не всегда однозначно. В общем случае, когда говорят об архитектуре фон Неймана, подразумевают принцип хранения данных и инструкций в одной памяти.



1. Принцип программного управления.

Процессор в определенной последовательности выполняет команды программы на машинном языке.

2. Принцип однородности памяти.

Команды и данные хранятся в одной и той же памяти (и кодируются в двоичной системе счисления).

3. Принцип адресуемости памяти.

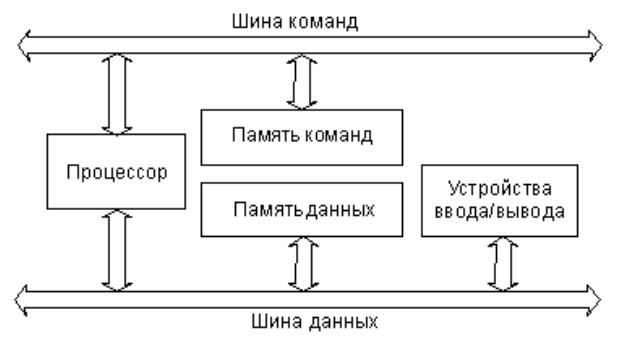
Память состоит из пронумерованных ячеек; процессору в произвольный момент времени доступна любая ячейка.

Классическая архитектура. Используется для настольных компьютеров

**Гарвардская архитектура** — [архитектура ЭВМ](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%AD%D0%92%D0%9C), отличительными признаками которой являются:

* хранилище инструкций и хранилище данных представляют собой разные физические устройства;
* канал инструкций и канал данных также физически разделены.

Архитектура была разработана [Говардом Эйкеном](https://ru.wikipedia.org/wiki/%D0%AD%D0%B9%D0%BA%D0%B5%D0%BD,_%D0%93%D0%BE%D0%B2%D0%B0%D1%80%D0%B4_%D0%A5%D0%B5%D0%B9%D0%B7%D1%8D%D0%B2%D0%B5%D0%B9" \o "Эйкен, Говард Хейзэвей) в конце [1930-х](https://ru.wikipedia.org/wiki/1930-%D0%B5) годов в [Гарвардском университете](https://ru.wikipedia.org/wiki/%D0%93%D0%B0%D1%80%D0%B2%D0%B0%D1%80%D0%B4%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82).



* Коды команд и данные хранятся на разных устройствах памяти.
* Канал чтения команд из памяти и канал чтения /записи данных физически разделены и работают параллельно.

**+** Быстродействие выше

**-** Больше аппаратных затрат (Проще реализовать в пределах одного чипа.)

***Применение :***

* *в процессорах - два кэша: данных и команд; мобильные системы, системы на кристалле.*

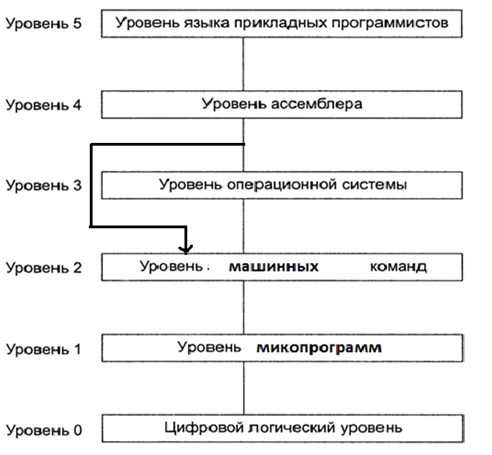
**Сравнительный анализ**

В чистой [архитектуре фон Неймана](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D1%84%D0%BE%D0%BD_%D0%9D%D0%B5%D0%B9%D0%BC%D0%B0%D0%BD%D0%B0) процессор в каждый момент времени может либо читать инструкцию, либо читать/записывать единицу данных из/в памяти. Оба действия одновременно происходить не могут, поскольку инструкции и данные используют один и тот же поток (*шину*).

В компьютере с использованием гарвардской архитектуры процессор может считывать очередную команду и оперировать памятью данных одновременно и без использования кэш-памяти. Таким образом, компьютер с гарвардской архитектурой при определенной сложности схемы быстрее, чем компьютер с архитектурой фон Неймана, поскольку потоки команд и данных расположены на раздельных физически не связанных между собой аппаратных каналах.

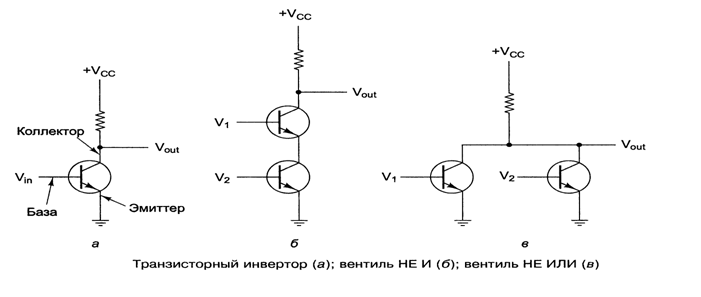
№8

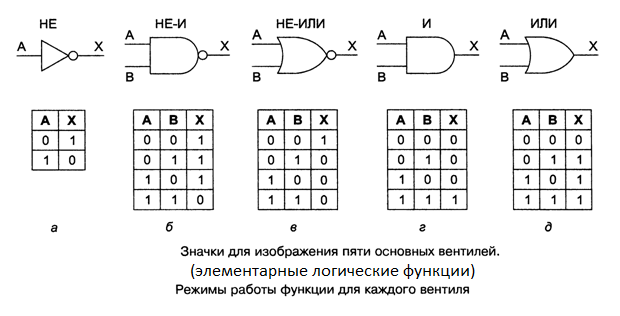
Уровневая архитектура компьютера с точки зрения программиста (цифровой логический уровень , уровень микрокоманд, уровень машинных команд, уровень ОС, уровень ассемблера)



Практически любой компьютер можно представить как систему, состоящую из нескольких уровней.

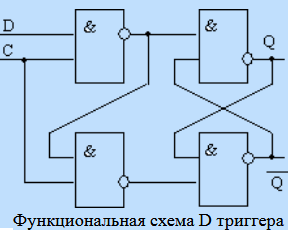
**Цифровой логический уровень**





1. Транзистор 🡪 Логический элемент 🡪 Триггер 🡪 Регистр 🡪 Цифровое устройство
2. Все цифровые устройства компьютера строятся с помощью пяти базовых цифровых логических элементов: ***И, ИЛИ, НЕ, НЕ-ИЛИ, НЕ И***.
3. Логические элементы строятся из транзисторов

* *Несколько логических элементов формируют устройство хранящее 1 бит информации (0или1) - триггер*
* *Триггеры, соединенные в группы, по 4, 8,16, 32 или 64…, образуют многобитные элементы памяти - регистры.*
* *Из логических элементов строятся сложные логические схемы из которых состоит сам компьютер (процессор, память и др.).*



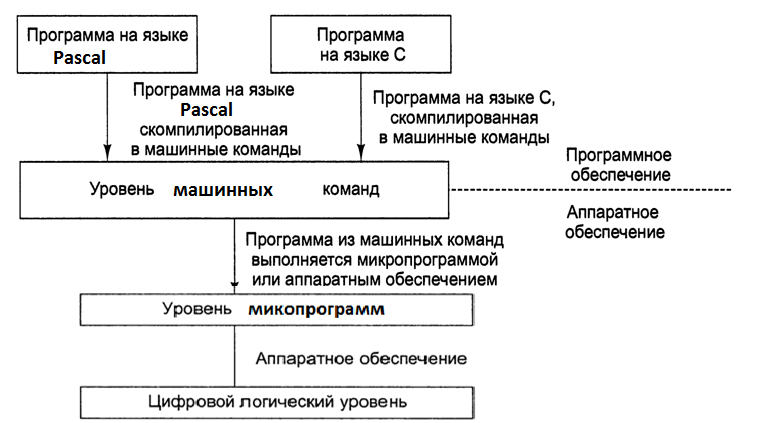
**Уровень микропрограмм**

* Каждая машинная команда *внутри процессора разбивается на последовательность элементарных операций (микрокоманд),* управляющих аппаратными устройствами процессора.
* Последовательность микрокоманд формирует микропрограмму.
* Каждой машинной команде соответствует своя микропрограмма процессора.

Микропрограммы записывается в управляющую память

микропрограмм процессора фирмой изготовителем.

**Уровень машинных команд**



* Каждый изготовитель процессора предлагает свой набор

машинных команд

* Является связующим уровнем между программным

и аппаратным обеспечением;

Процессор Intel x8086 имел набор в 98 машинных команд

**Уровень операционной системы**

* Для программиста операционная система — это программа, предоставляющая ряд новых команд в дополнение к командам предлагаемым уровнем машинных команд.
* *Команды предлагаемые ОС в*ызывают определенные службы операционной системы и образуют интерфейс прикладного программирования (*API - Application* Program Interface)

Каждая команда операционной системы выполняются

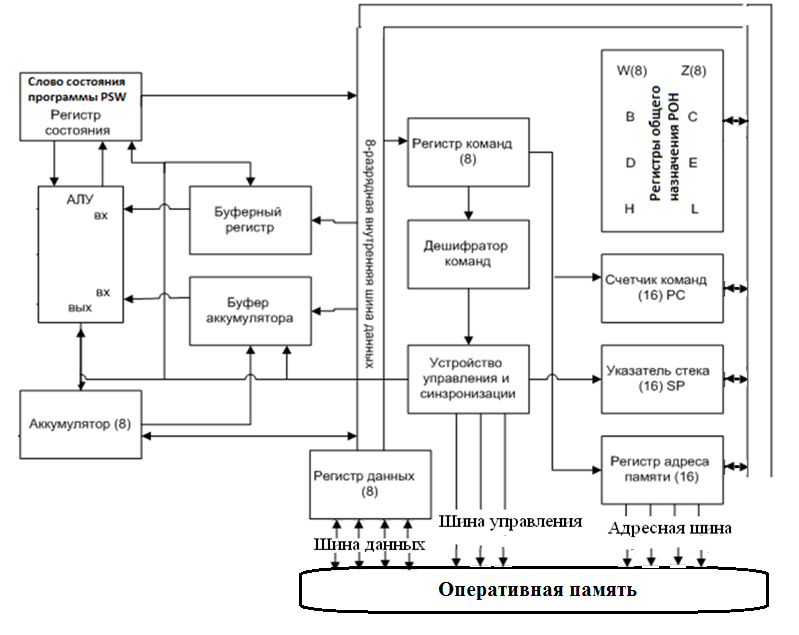
с помощью последовательности машинных команд.

**Уровень ассемблера**

* Каждая команда языка ассемблера соответствует одной команде машинного языка
* Язык ассемблера больше всех похож на машинный язык.
* Достоинство программы на языке ассемблера: минимальный объем занимаемой памяти и высокая скорость работы.
* *Примеры команд*
  + ADD - (add) сложить
  + SUB - (subtract) вычесть
  + MUL - (multiply) умножить
  + DIV - (divide) делить

**№9**

**Обобщенная архитектура 8-разрядного микропроцессора. Назначение основных узлов.**



Арифметико -логическое устройство (АЛУ)

* *АЛУ* выполняет обработку данных.
* Результат выполнения команды запоминается в аккумуляторе
* Так как в АЛУ нет запоминающих регистров, то информация в него поступает из:
  + буферных регистров (служащих для временного хранения данных);
  + аккумулятора.

Типичные операции в АЛУ:

сложение, инвертирование, сдвиг, инкремент и декремента.

**Аккумулятор**

* *Аккумулятор – главный «накапливающий» регистр МП при различных манипуляциях с данными.*
* При выполнении операций над данными они обычно помещаются в аккумулятор.
* Данные поступают в аккумулятор с:
  + АЛУ;
  + внутренней шины данных МП.

**Счетчик команд (Program Counter (РС) - программный счётчик)**

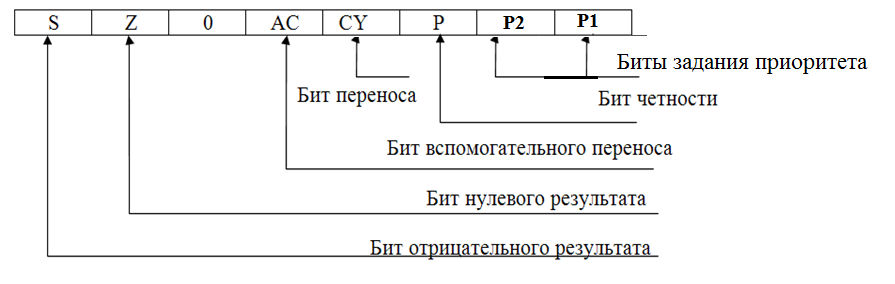
* *СК содержит адрес команды программы в памяти*
* После извлечения очередной команды из памяти процессор автоматически даёт приращение содержимому счетчика команд. С этого момента СК содержит, адрес следующей команды.
* А*дрес следующей выполняемой команды хранится на протяжении всего времени выполнения текущей команды.*

*Для начала выполнения программы в счетчик команд необходимо загрузить адрес первой команды программы.*

**Регистр команд**

* *Регистр команд - хранит текущую выполняемую машинную команду.*
* Команда хранится в регистре на протяжении всего времени её исполнения

**Регистр состояния(Слово состояния программы PSW - program status word)**



* Хранит признаки выполнения операций в АЛУ при выполнении команд программы.
* Содержимое битов регистра используется устройством управления для управления ходом выполнения программы.

В этом регистре находятся биты управления приоритетом

ЦП, задающие режим выполнения текущей команды

(пользовательский - или ядра).

**Регистр адреса памяти и регистр данных**

* Регистр адреса памяти содержит адрес ячейки памяти, к которой будет обращаться процессор. Выход этого регистра называется адресной шиной.
* Регистр данных – служит для промежуточного хранения данных считываемых или записываемых в память. Выход этого регистра называется шиной данных

**Регистры общего назначения (РОН)  
 (регистровая память)**

* РОН служат для временного хранения данных.
* РОН позволяют повысить быстродействие микропроцессора за счет сокращения пересылок кодов между МП и более медленной оперативной памятью.

РОН представляет собой сверхбыструю регистровую

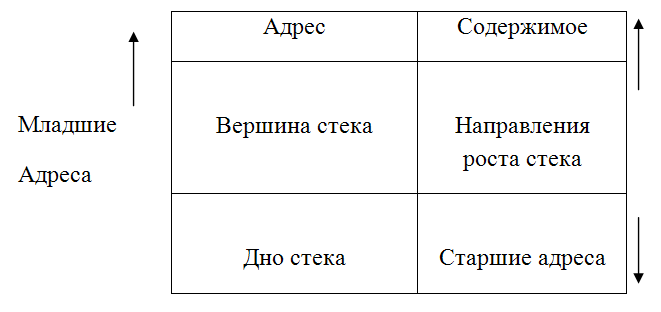
память процессора***.***

**№10**

**Назначение и принцип работы аппаратного стека.**

**Стек**

* *Стек – это область памяти, специально выделяемая для временного хранения данных.*
* Запись и чтение данных в стеке по принципу LIFO (Last In First Out) – «последним пришел, первым ушел».
* Ячейка памяти, заполненная последней, считывается первой
* *По мере записи данных стек растёт в сторону младших адресов.*



**Указатель стека (Stack Pointer - (SP))**

* Указатель стека (SP) - регистр, который содержит адрес памяти последнего, помещенного в стек слова данных.
* Обращение и адресация к стеку производится через регистр SP.
* При записи слова данных в стек значение адреса в SP уменьшается на единицу, а при считывании увеличивается на единицу.

SP – всегда хранит вершину стека (адрес его последней

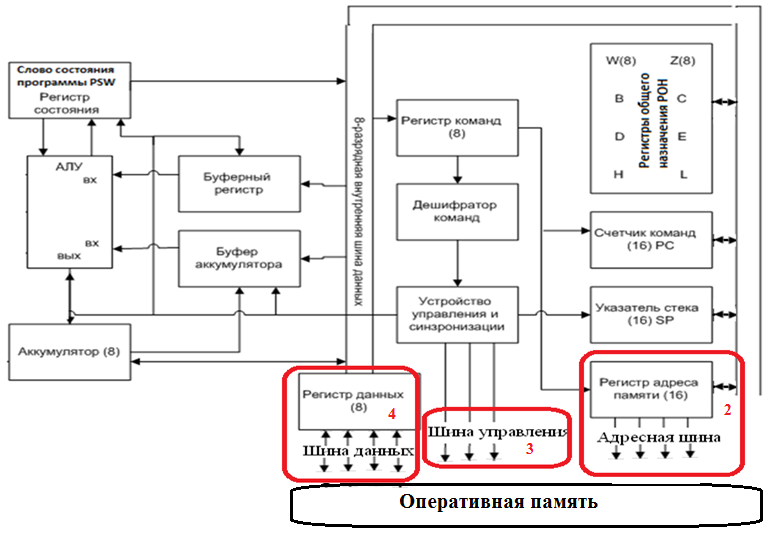
загруженной ячейки памяти.)

№11

**Последовательность операций в процессоре при выполнении программы.**



* В счетчик команд заносится адрес ячейки памяти, содержащий первую команду программы.



* Адрес команды через регистр адреса поступает на шину адреса памяти. По сигналу управления процессор считывает команду, находящуюся по этому адресу на шину данных и записывает в регистр данных процессора.



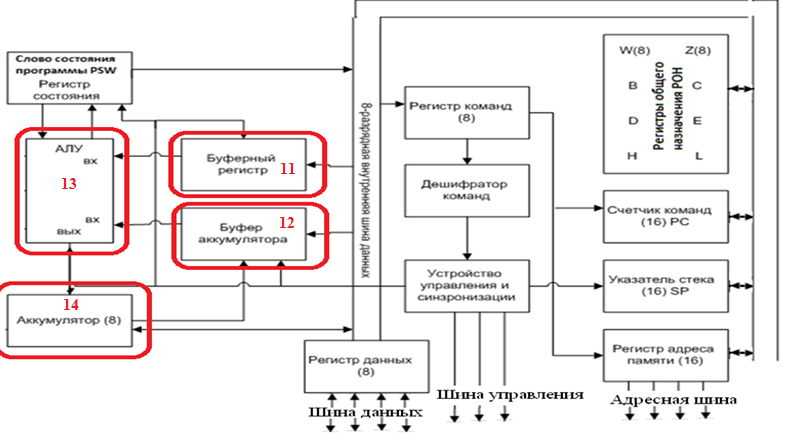
* Из регистра данных команда поступает в регистр команды. Команда дешифруется и анализируется устройством управления.
* В зависимости от кода операции УУ вырабатывает управляющие сигналы .



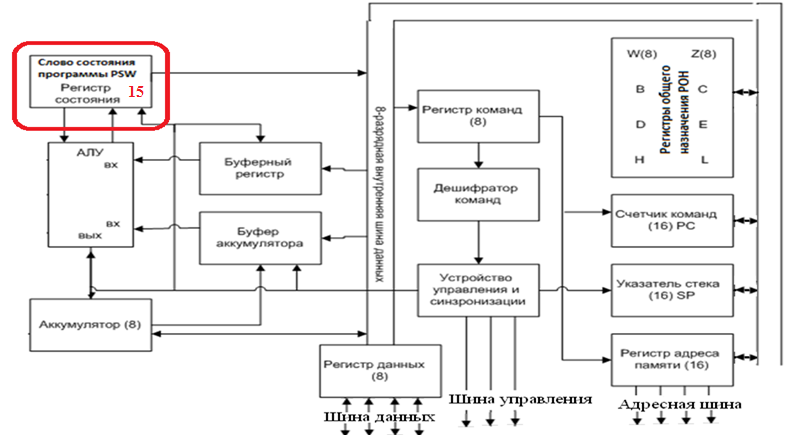
* Если машинная команда использует данные из памяти, определяется, где находятся эти данные. Процессор переносит данные в РОН.



* Содержимое счетчика команд увеличивается. В нем уже содержится адрес следующей команды.



* Данные из РОН поступают в буферные регистры
* АЛУ выполняет операцию над данными, находящимися в регистрах сохраняет результат в аккумуляторе.



* В регистре слова состояния программы устанавливаются признаки выполнения команды



* Считывается следующая команда и цикл повторяется.

№12 **Назначение и организация механизма прерываний. Типы прерываний. Множественные прерывания.**

**Организация прерываний**

* Прерывание – это временное прекращение выполнения одной программы для выполнения другой программы (запланированной или незапланированной)
* Прерывания предназначены для :
  + повышения эффективности работы процессора;
  + организации многозадачного режима
* Пример (клавиатура).

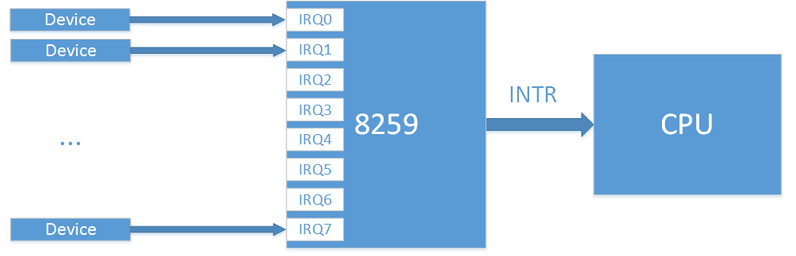
**Типы прерываний**

* В зависимости от источника возникновения, прерывания делятся на:
* Аппаратные (внешние, асинхронные) - обрабатывают запросы на обслуживание ( *Interrupt request, IRQ*), от аппаратных устройств (*например, клавиатуры, движение мышки, таймера и.т.д.)*.
* Программные (синхронные) - вызываются специальной команды в коде программы ( *команда* ***int № прерывания***).
* Исключения (внутренние) - события происходящие в самом процессоре, например:
  + деление на ноль
  + переполнение стека,
  + обращение к недопустимым
  + адресам памяти или недопустимый код операции;

Всего 256 источников прерываний

**Таблица векторов прерываний и контроллер прерываний**

* + Вектор прерываний - адрес начала программы обработки прерываний.
  + Таблица векторов прерываний - область памяти, где располагаются вектора обработки всех прерываний называется
  + Контроллер прерываний – специальное устройство для фиксации прерываний от устройств (для экономии выводов процессора).
  + Имеет несколько входов и один выход, связанный с процессором



**Обработка прерываний**

1)При возникновении прерывания процессор приостанавливает работу текущей программы.

2)Процессор **АППАРАТНО** сохраняет в стеке текущее содержимое регистра счетчика команд (*хранит адрес следующей команды прерываемой программы*) и регистра слова состояния (*хранит признаки результата выполнения последней команды прерываемой программы*).

3) По номеру прерывания процессор загружает из таблицы векторов прерывания в счетчик команд начальный адрес программы обработки прерывания и стартует её.

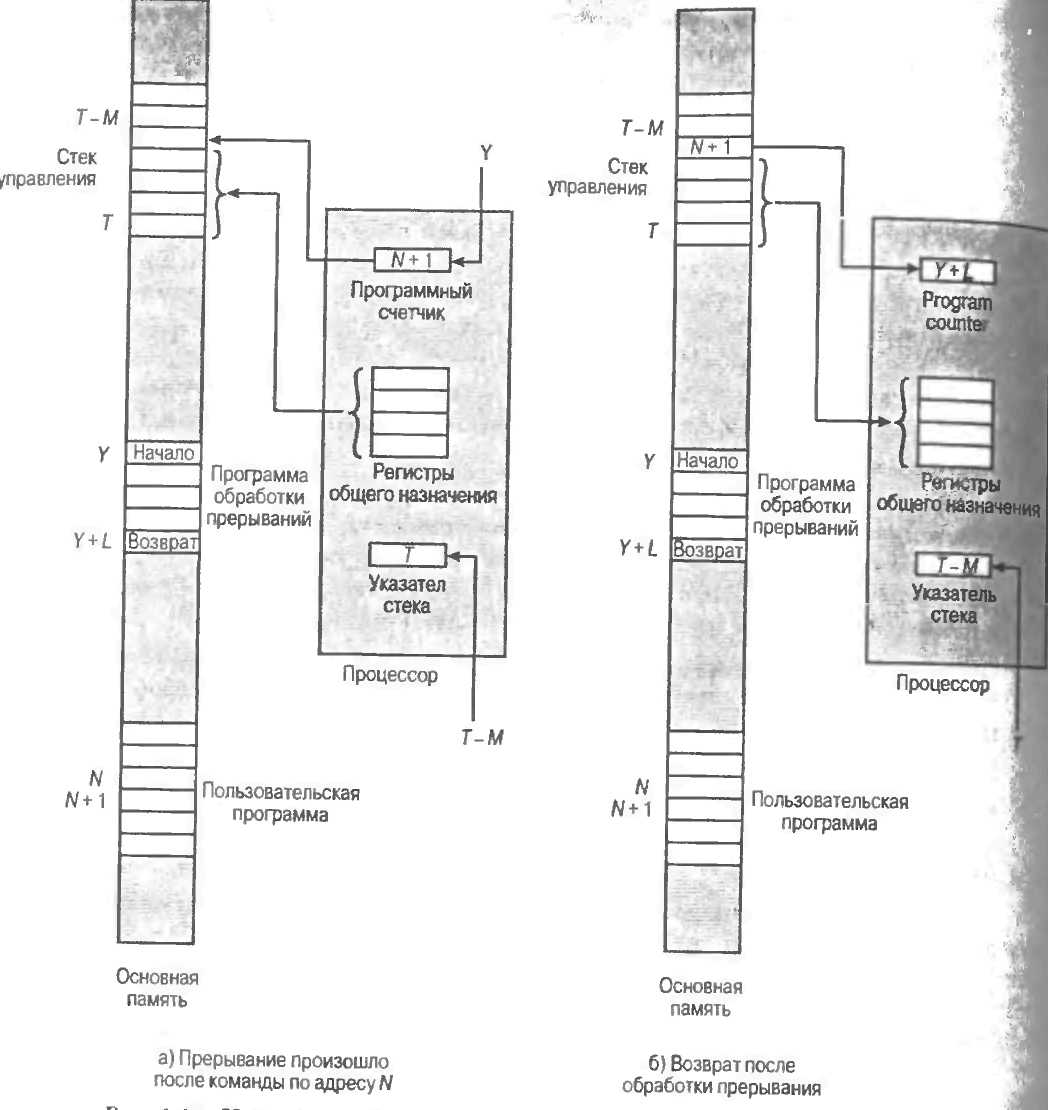
4)Программа обработки прерываний сохраняет при необходимости в стеке содержимое остальных регистров процессора (*РОН, аккумулятор*), и выполняют действия по обслуживанию прерывания.

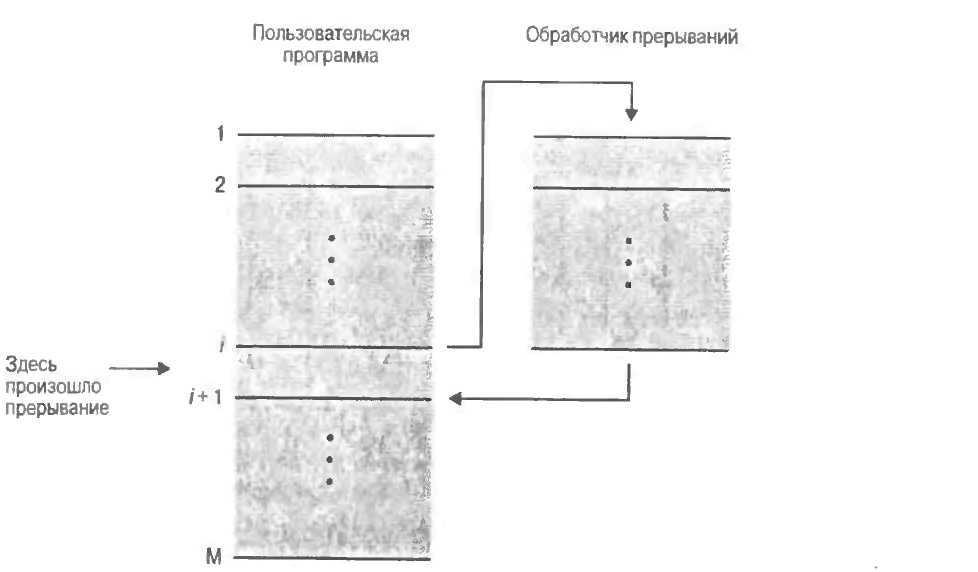
5)После обслуживания прерывания программа обработки прерывания загружает из стека содержимое регистров процессора (*РОН, аккумулятор*), прерванной программы

6)В счетчик команд процессора из стека загружается адрес следующей команды прерванной программы и слово состояния программы.

7)Прерванная программа продолжает работать.

* Пункты 1 – 3,6 выполняются аппаратурой
* Пункты 4 – 5,7 выполняются программно





**Множественные прерывания**

* Множественные прерывания – когда возникают два прерывания одновременно.
* Два способа обработки:
  + последовательная обработка прерываний - запретить новые прерывания до тех пор, пока обрабатывается предыдущее (можно пропустить важные прерывания).
  + вложенная обработка прерываний - приостановить обработку прерывания с более низким приоритетом в пользу прерывания солее высоким приоритетом.

