Основные принципы построения компьютеров

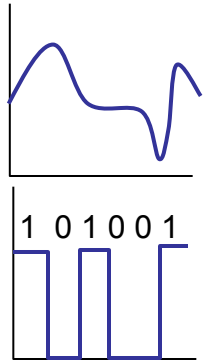
Термины и определения

- Компьютер, hardware, software. Программа, команда, типы команд.
- Архитектура компьютера, структура компьютера. Архитектура Фон Неймана, принципы Фон Неймана. Другие архитектуры. Процесс выполнения программы.
- Принцип открытой архитектуры.
- Магистрально-модульное устройство компьютера.
- Система команд ЭВМ.
- Управление памятью. Физическая организация памяти. Иерархия памяти. Виртуальная память. Методы распределения памяти. Кэширование данных.
- Стек.

Определения

Компьютер *(computer)* – это **программируемое** электронное устройство для обработки числовых и символьных данных.

- **аналоговые** компьютеры – складывают и умножают аналоговые (непрерывные) сигналы
- **цифровые** компьютеры – работают с цифровыми (дискретными) данными.



Hardware – аппаратное обеспечение, «железо».

Software – программное обеспечение, «софт».

Определения

Программа – это последовательность команд, которые должен выполнить компьютер.

Команда – это описание операции (1…4 байта):

- **код команды**
- **операнды** – исходные данные (числа) или их адреса
- **результат** (куда записать).

Типы команд:

- **безадресные** (1 байт)
- **одноадресные** (2 байта)
- **двухадресные** (3 байта)
- **трехадресные (**4 байта**)**

- Трехадресная команда

a1,a2 – адреса ячеек (регистров), где находятся числа, участвующие в операции (операнды)

a3 – адрес ячейки оперативной памяти, куда нужно поместить результат

- Двухадресная команда

Результат записывается в ячейку a2

- Одноадресная команда

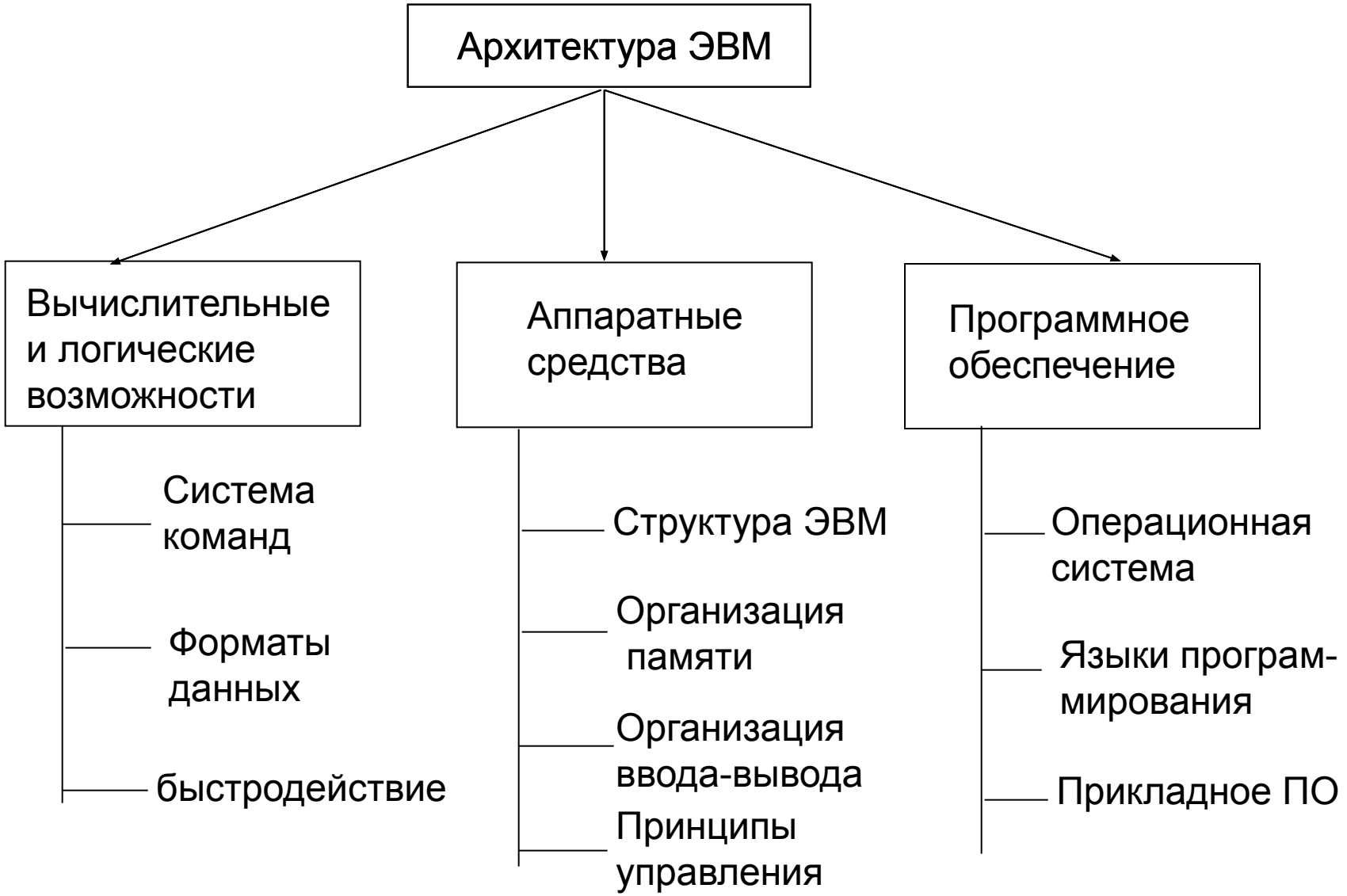
a1 – адрес ячейки, где хранится число участвующее в операции или адрес ячейки, где записывается результат

- Нуль адресная команда

Все операнды в регистре ЦП

Архитектурой компьютера называется его описание на некотором общем уровне, включающее описание пользовательских возможностей программирования, системы команд, системы адресации, организации памяти и т.д.

Архитектура определяет принципы действия, информационные связи и взаимное соединение основных логических узлов компьютера: процессора, оперативного ЗУ, внешних ЗУ и периферийных устройств. Общность архитектуры разных компьютеров обеспечивает их совместимость с точки зрения пользователя.


Архитектура ЭВМ
Вычислительные и логические возможности
Аппаратные средства
Программное обеспечение
Система команд
Форматы данных
быстродействие
Структура ЭВМ
Организация памяти
Организация ввода-вывода
Принципы управления
Операционная система
Языки програм-мирования
Прикладное ПО

- ***Структура компьютера*** — это совокупность его функциональных элементов и связей между ними. Элементами могут быть самые различные устройства — от основных логических узлов компьютера до простейших схем. Структура компьютера графически представляется в виде структурных схем, с помощью которых можно дать описание компьютера на любом уровне детализации.

При создании первых вычислительных машин в 1945 математик Джон фон Нейман описал основы конструкции компьютера. Согласно принципам фон Неймана, компьютер должен иметь следующие устройства:

- Арифметико-логическое устройство — для непосредственного осуществления вычислений и логических операций.
- Устройство управления — для организации процесса управления программ.
- Запоминающее устройство (память) — для хранения программ и информации.
- Внешние устройства — для ввода и вывода информации.

Обосновал использование двоичной системы представления чисел, продемонстрировал преимущества технической реализации, удобство и простоту выполнения в ней арифметических и логических операций.

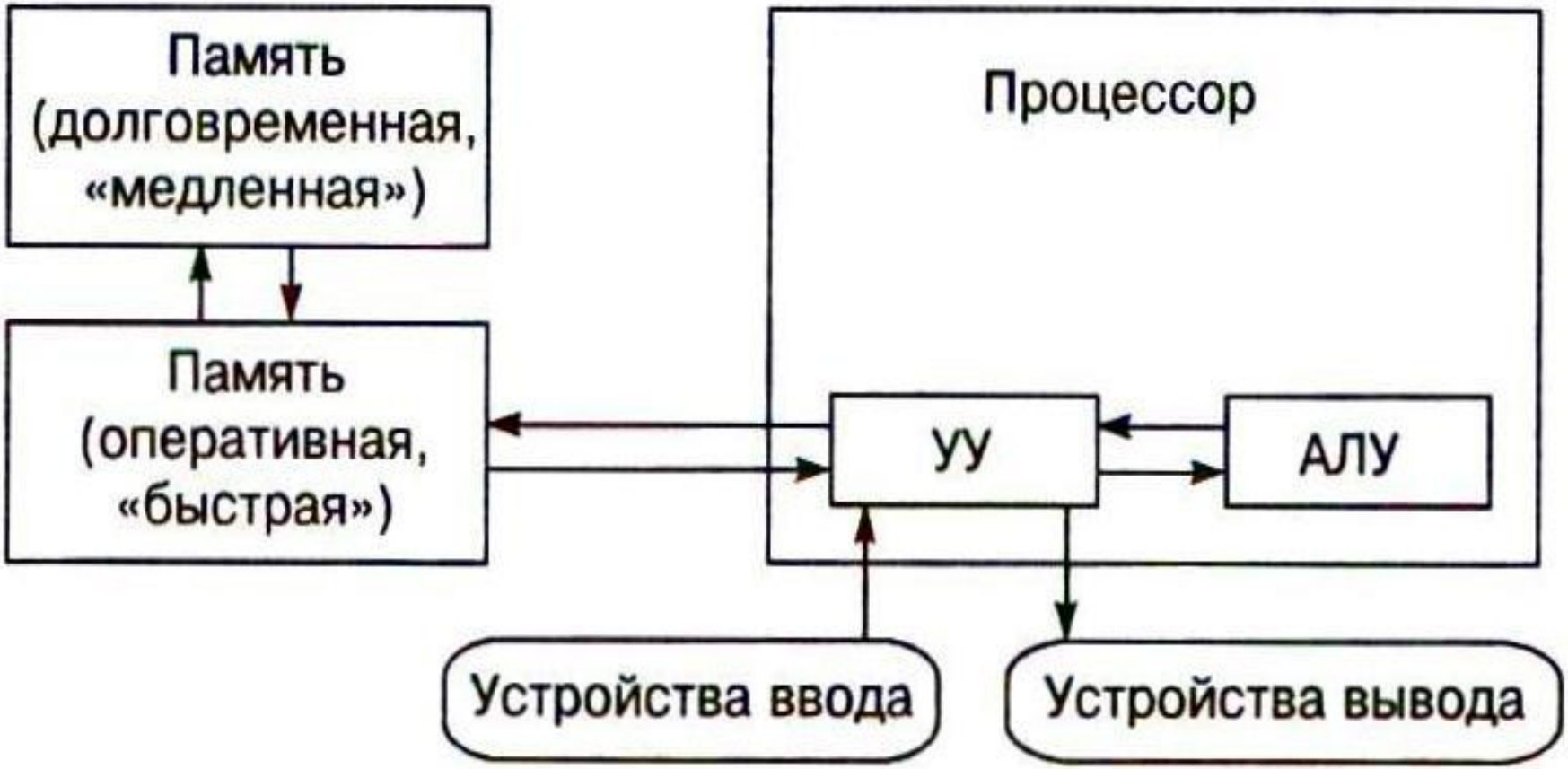


Схема взаимодействий устройств компьютера согласно архитектуре фон Неймана

Обозначения: УУ – устройство управления;
АЛУ – арифметико-логическое устройство

Принципы фон Неймана

1. **Принцип двоичного кодирования**: вся информация кодируется в двоичном виде.
2. **Принцип программного управления**: программа состоит из набора команд, которые выполняются процессором автоматически друг за другом в определённой последовательности.
3. **Принцип однородности памяти:** программы и данные хранятся в одной и той же памяти.
4. **Принцип адресности:** память состоит из пронумерованных ячеек; процессору в любой момент времени доступна любая ячейка.

Выполнение программы

Счетчик команд (*IP = Instruction Pointer*) – регистр, в котором хранится адрес следующей команды.

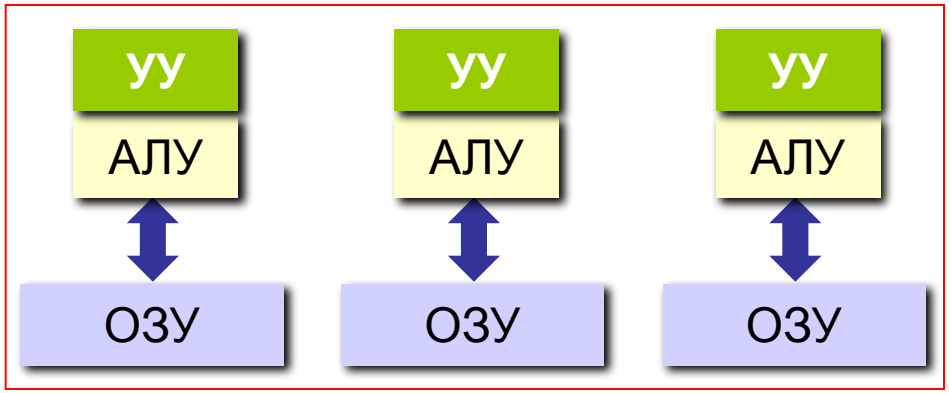
1. **Команда**, расположенная по этому адресу, передается **в УУ**. Если это не команда перехода, регистр **IP** увеличивается на длину команды.
2. УУ расшифровывает **адреса операндов**.
3. Операнды загружаются **в АЛУ**.
4. УУ дает команду АЛУ на **выполнение операции**.
5. **Результат** записывается по нужному адресу.
6. Шаги 1-5 повторяются до получения команды **«стоп»**.

Подавляющее большинство компьютеров в своих основных чертах соответствует принципам фон Неймана, но схема устройства современных компьютеров несколько отличается от классической схемы. В частности, арифметико-логическое устройство и устройство управления, как правило, объединены в центральный процессор. Многие быстродействующие компьютеры осуществляют параллельную обработку данных на нескольких процессорах.

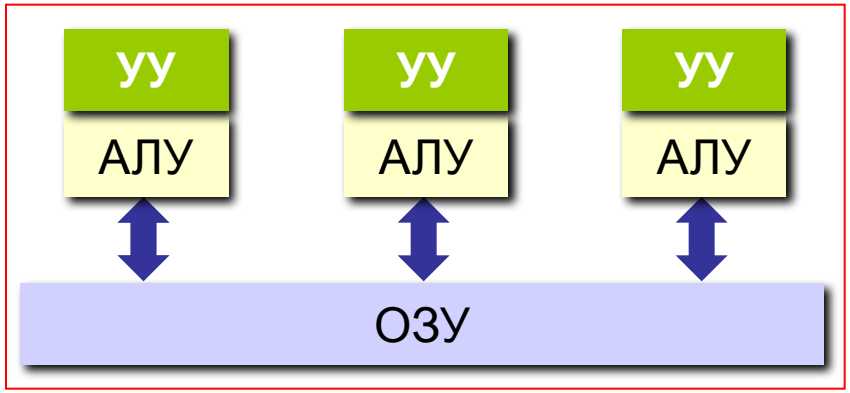
Архитектуры компьютеров

фон Неймана

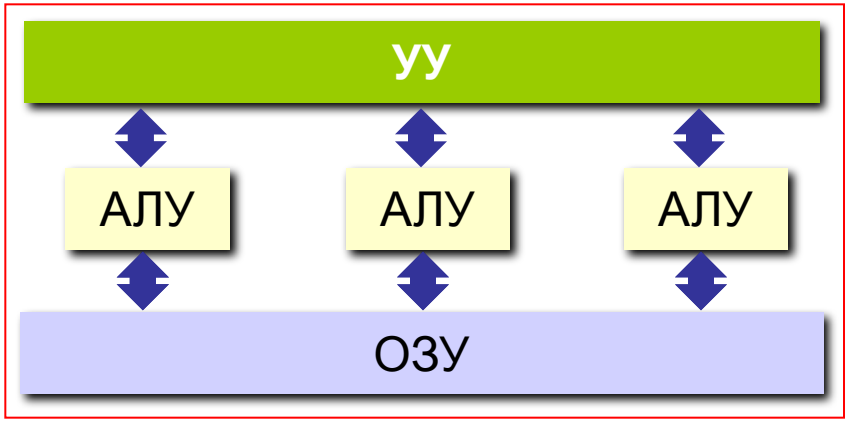
B) многомашинная
(независимые задачи)



A) многопроцессорная
(части одной задачи,
по разным программам)



C) параллельные процессоры
(части одной задачи,
по одной программе)



Многопроцессорная архитектура. Наличие в компьютере нескольких процессоров означает, что параллельно может быть организовано много потоков данных и много потоков команд. Таким образом, параллельно могут выполняться несколько фрагментов одной задачи. Структура такой машины, имеющей общую оперативную память и несколько процессоров, представлена на рисунке А.

- *Многомашинная вычислительная система*. *Рис. В.* Здесь несколько процессоров, входящих в вычислительную систему, не имеют общей оперативной памяти, а имеют каждый свою (локальную). Каждый компьютер в многомашинной системе имеет классическую архитектуру, и такая система применяется достаточно широко. Однако эффект от применения такой вычислительной системы может быть получен только при решении задач, имеющих очень специальную структуру: она должна разбиваться на столько слабо связанных подзадач, сколько компьютеров в системе.
- Преимущество в быстродействии многопроцессорных и многомашинных вычислительных систем перед однопроцессорными очевидно.

Архитектура с параллельными процессорами. Здесь несколько АЛУ работают под управлением одного УУ. Это означает, что множество данных может обрабатываться по одной программе — то есть по одному потоку команд. Высокое быстродействие такой архитектуры можно получить только на задачах, в которых одинаковые вычислительные операции выполняются одновременно на различных однотипных наборах данных. Структура таких компьютеров представлена на рисунке С.

Принцип открытой архитектуры

1. Регламентируются и стандартизируются только описание принципа действия компьютера и его конфигурация (определённая совокупность аппаратных средств и соединений между ними). Таким образом компьютер можно собирать из отдельных узлов и деталей, разработанных и изготовленных независимыми фирмами-производителями.

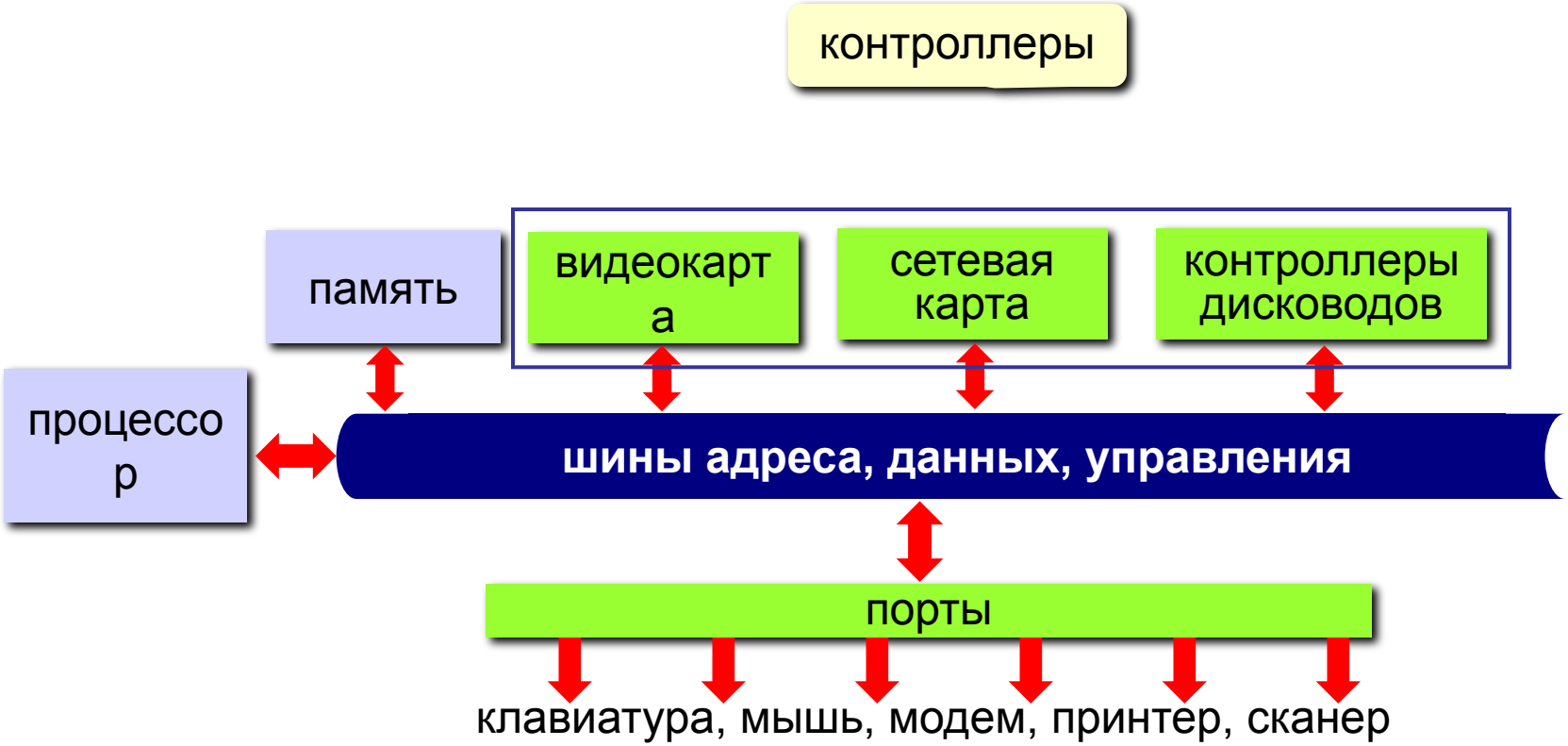
2. Компьютер легко расширяется и модернизируется за счёт наличия внутренних расширительных гнёзд, в которые пользователь может вставлять разнообразные устройства, и, тем самым устанавливать конфигурацию своей машины в соответствии со своими личными предпочтениями.

Принцип открытой архитектуры

- на **материнской плате** расположены только узлы, которые обрабатывают информацию (процессор и вспомогательные микросхемы, память)
- схемы, управляющие другими устройствами (монитором и т.д.) – это отдельные **платы**, которые вставляются в **слоты расширения**
- **схема стыковки** новых устройств с компьютером общедоступна (стандарт)

- **конкуренция**, удешевление устройств
- производители могут изготавливать **новые** совместимые устройства
- пользователь может собирать ПК «**из кубиков**»

Взаимосвязь блоков ПК



Шина – многожильная линия связи, доступ к которой имеют несколько устройств.

Контроллер – электронная схема, управляющая внешним устройством по сигналам процессора.

Модульный принцип

Модульный принцип позволяет потребителю самому комплектовать нужную ему конфигурацию компьютера и производить при необходимости её модернизацию.

Каждая отдельная функция компьютера реализуется одним или несколькими модулями – конструктивно и функционально законченных электронных блоков в стандартном исполнении. Организация структуры компьютера на модульной основе аналогична строительству блочного дома. Основными модулями компьютера являются память и процессор. Процессор – это устройство управляющее работой всех блоков компьютера. Действия процессора определяются командами программы, хранящейся в памяти.

Благодаря использованию вышеназванного принципа, появляется возможность создания большого разнообразия товаров из одного набора основных компонентов. Из набора модулей возможно создать большое разнообразие компьютеров (сложных технических систем), отличающихся друг от друга производительностью, назначением (домашний, офисный, сервер приложений и т. п.), архитектурой, платформой.

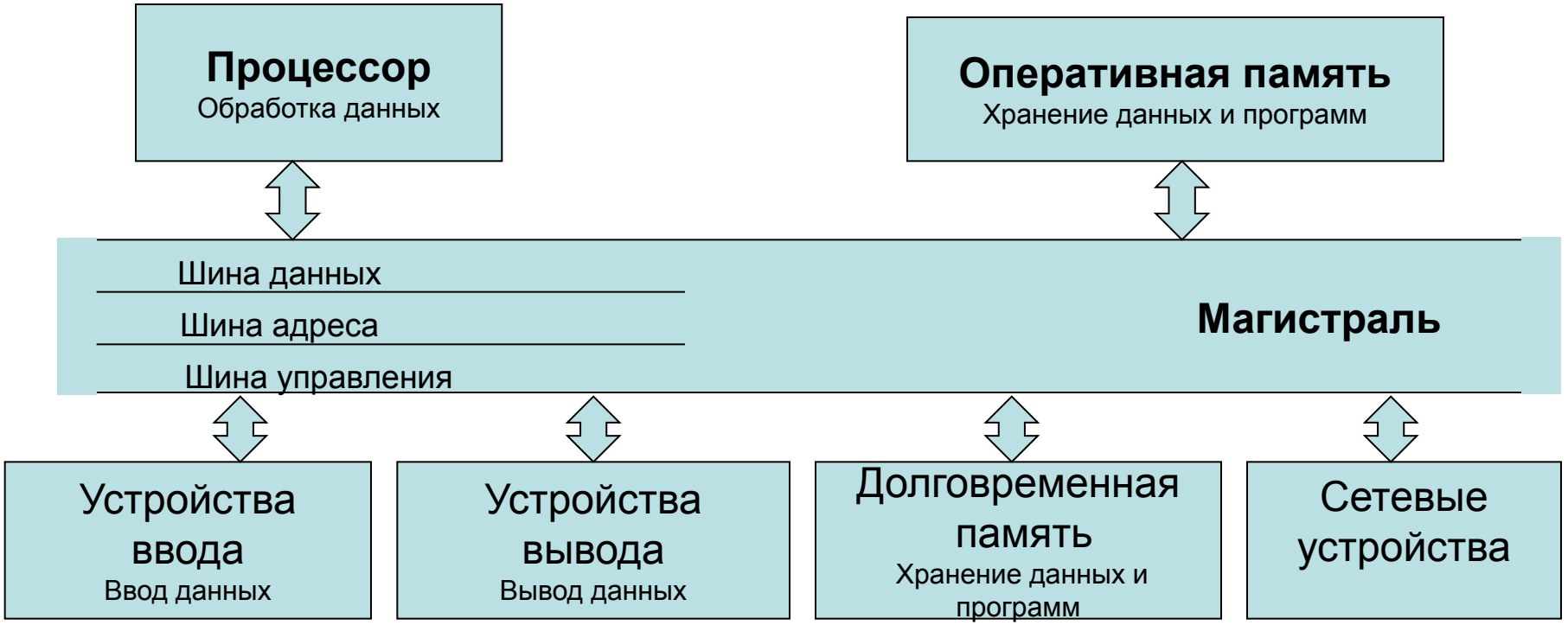
Магистрально-модульный принцип

Модульная организация опирается на магистральный (шинный) принцип обмена информацией между устройствами.

Магистрально-модульный принцип имеет ряд достоинств:

- 1. для работы с внешними устройствами используются те же команды процессора, что и для работы с памятью.
- 2. подключение к магистрали дополнительных устройств не требует изменений в уже существующих устройствах, процессоре, памяти.
- 3. меняя состав модулей можно изменять мощность и назначение компьютера в процессе его эксплуатации.

Магистрально-модульное устройство компьютера



Для обеспечения информационного обмена между различными устройствами должна быть предусмотрена какая-то магистраль для перемещения потоков информации.

Магистраль

Магистраль (системная шина) включает в себя:

1. **Шину данных;**
2. **Шину адреса;**
3. **Шину управления**.

Упрощенно системную шину можно представить как группу кабелей и электрических (токопроводящих) линий на системной плате.

К магистрали подключаются процессор и оперативная память, а также периферийные устройства ввода, вывода и хранения информации, которые обмениваются информацией на машинном языке (последовательностями нулей и единиц в форме электрических импульсов).

Шина данных

По этой шине передаются данные между различными устройствами. Например, считанные из ОЗУ данные могут быть переданы процессору для обработки, а затем могут быть отправлены обратно для хранения.

Таким образом, данные по шине данных могут передаваться от устройства к устройству в любом направлении.

Разрядность шины данных определяется процессором, т.е. количеством двоичных разрядов, которые могут обрабатываться процессором одновременно. Разрядность процессоров постоянно увеличивается по мере развития компьютерной техники.

Шина адреса

Выбор устройства или ячейки памяти, куда посылаются данные или откуда считываются данные по шине данных, производит процессор. Каждое устройство или ячейка памяти имеет свой адрес. Адрес передается по адресной шине от процессора к памяти или устройствам.

Разрядность шины адреса определяет объем адресуемой памяти.

Разрядность адресной шины определяет доступное адресное пространство, т.е. количество однобайтовых ячеек оперативной памяти, которые могут иметь уникальные адреса.

Если разрядность адресной шины равна n, то максимальный адрес, который может быть по ней передан – 2^n. Очевидно, количество байтов оперативной памяти не должно превышать 2^n, иначе байты с большими адресами не будут использоваться.

Шина управления

По шине управления передаются сигналы, определяющие характер обмена информацией по магистрали. Сигналы показывают, какую операцию – считывание или запись информации нужно производить, синхронизируют обмен данными и т.д.

Основные пользовательские характеристики:

Разрядность – количество бит информации, параллельно «проходящих» через шину;

Пропускная способность – количество бит информации, передаваемых по шине за секунду.

- Для расширения возможностей ПК и повышения функциональных характеристик микропроцессора дополнительно может поставляться математический сопроцессор, служащий для расширения набора команд МП. Например, математический сопроцессор IBM-совместимых ПК расширяет возможности МП для вычислений с плавающей точкой; сопроцессор в локальных сетях (LAN-процессор) расширяет функции МП в локальных сетях.

Система команд ЭВМ (классификация по функциональному назначению)

1. **Команды передачи данных**

   - команды передачи кодов между регистрами внутри процессора,
   - из регистров процессора в память,
   - из памяти в регистры процессора,
   - из одних ячеек памяти в другие,
   - передачи данных между процессором и портами внешних устройств;

   команды работы со стеком.

2) Команды обработки данных

Данная группа команд подразделяется на:

- Арифметические (сложит, вычесть, умножить..);
- Логические (операции И, ИЛИ, НЕ…);
- Команды сдвига

Команды этого типа могут иметь один или два операнда. Операнды могут храниться в регистрах центрального процессора, в памяти или в специализированном регистре. Они формируют признаки результатов: перенос из старшего разряда, переполнение, нулевой результат,…

3. Команды передачи управления

Используются для изменения естественного порядка следования команд организации циклических участков в программах.

- команда безусловного перехода JMP <адрес> -

загружает адрес перехода, указанный в команде, в программный счетчик.

- команды условного перехода используются после команд, изменяющих состояние флагового регистра.

4. Команды для работы с подпрограммами. Стеки

Адрес возврата – адрес команды, на которую управление передается после окончания работы подпрограммы.

Для организации подпрограмм используется аппаратно поддерживаемую структуру данных стек (последним вошел – первым вышел)

Указатель стека хранит адрес ячейки памяти, содержащей последнее помещенное в стек значение.

Для сохранения адреса возврата в стек используются команды вызова подпрограмм CALL<адрес>.

Для возврата из подпрограммы в основную программу – команды возврата RETURN.

5. Прочие команды (дополнительные или специальные)

Команды остановки центрального процессора,

сброса внешних устройств,

установка или сброс отдельных признаков.

Управление памятью

- **Оперативная память** – важнейший ресурс вычислительной системы, требующий управления со стороны ОС. Причина – процессы и потоки хранятся и обрабатываются в оперативной памяти.
- Память распределяется между приложениями и модулями самой операционной системы.
- **Функции ОС по управлению оперативной памятью**:
  - Отслеживание наличия свободной и занятой памяти;
  - Контроль доступа к адресным пространствам процессов;
  - Вытеснение кодов и данных из оперативной памяти на диск, когда размеров памяти недостаточно для размещения всех процессов, и возвращение их обратно;
  - Настройка адресов программы на конкретную область физической памяти;
  - Защита выделенных областей памяти процессов от взаимного вмешательства.
- Часть ОС, которая отвечает за управление памятью, называется **менеджером памяти**.

Физическая организация памяти

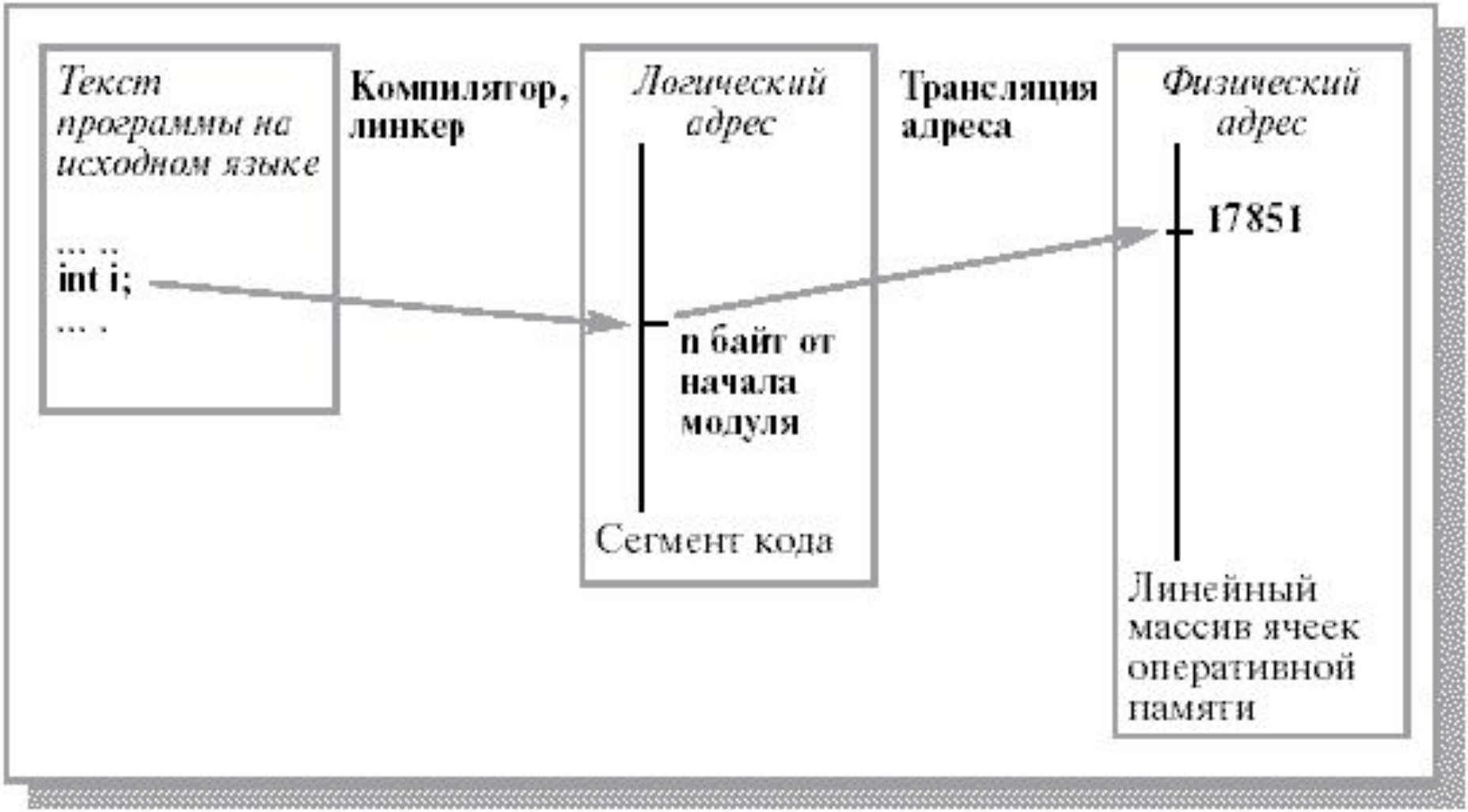
- Запоминающие устройства компьютера разделяют, как минимум, на два уровня: *основную* (главную, *оперативную*, *физическую*) и вторичную (внешнюю) память.
- **Основная память** представляет собой упорядоченный массив однобайтовых ячеек, каждая из которых имеет свой уникальный адрес (номер). Процессор извлекает команду из *основной памяти*, декодирует и выполняет ее. Для выполнения команды могут потребоваться обращения еще к нескольким ячейкам *основной памяти*.
- **Вторичную память** (это главным образом диски) также можно рассматривать как одномерное линейное *адресное пространство*, состоящее из последовательности байтов. В отличие от *оперативной памяти*, она является энергонезависимой, имеет существенно большую емкость и используется в качестве расширения *основной памяти*.

Иерархия памяти

Представление потоков в оперативной памяти

- Для идентификации переменных и команд программы используются разные типы адресов:
  - **Символьные** (имена переменных, функций и т.п.);
  - **Виртуальные** – условные числовые значения, вырабатываемые компиляторами;
  - **Физические** – адреса фактического размещения в оперативной памяти.

Связывание адресов



Виртуальное адресное пространство

- Совокупность виртуальных адресов называется *виртуальным адресным пространством*. Диапазон возможных адресов виртуального пространства у всех процессов одинаков.
- Совпадение виртуальных адресов различных процессов не должно приводить к конфликтам и операционная система отображает виртуальные адреса различных процессов на разные физические адреса.
- Разные ОС по разному организуют виртуальное адресное пространство:
  - ***Линейная организация*** – пространство представляется непрерывной линейной последовательностью адресов (по другому плоская структура адресного пространства).
  - ***Сегментная организация*** – пространство разделяется на отдельные части. В этом случае, помимо линейного адреса, может быть использован виртуальный адрес (сегмент, смещение).

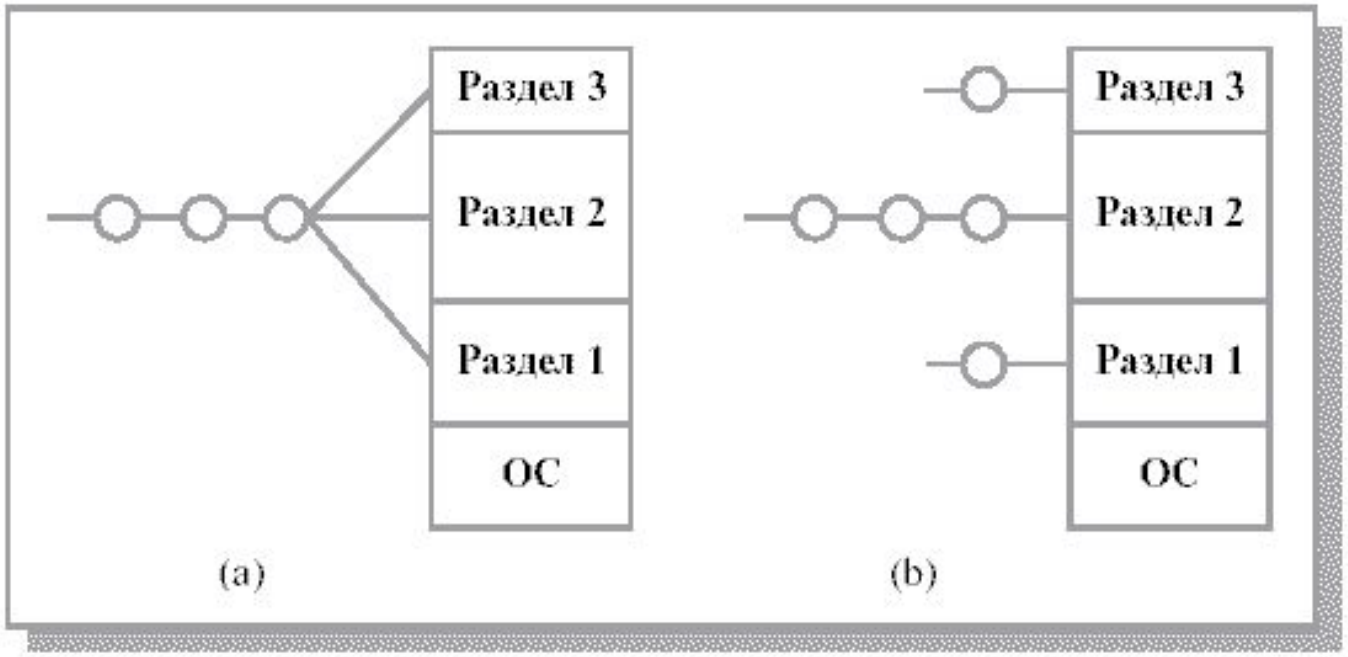
Виртуальное адресное пространство

- В виртуальном адресном пространстве выделяют две непрерывные части:
  - Системная – для размещения модулей общих для всей системы (размещаются коды и данные ядра ОС, другие служебные модули);
  - Пользовательская – для размещения кода и данных пользовательских программ.
- Системная область включает в себя область, подвергаемую страничному вытеснению, и область, на которую страничное вытеснение не распространяется. В последней располагаются системные процессы, требующие быстрой реакции или постоянного присутствия в памяти. Остальные сегменты подвергаются вытеснению, как и пользовательские приложения.

Алгоритмы распределения памяти

Схема с фиксированными разделами

- Схема основана на предварительном разбиении общего адресного пространства на несколько разделов фиксированной величины.
- Процессы помещаются в тот или иной раздел.
- Связывание физических и логических адресов процесса происходит на этапе его загрузки.

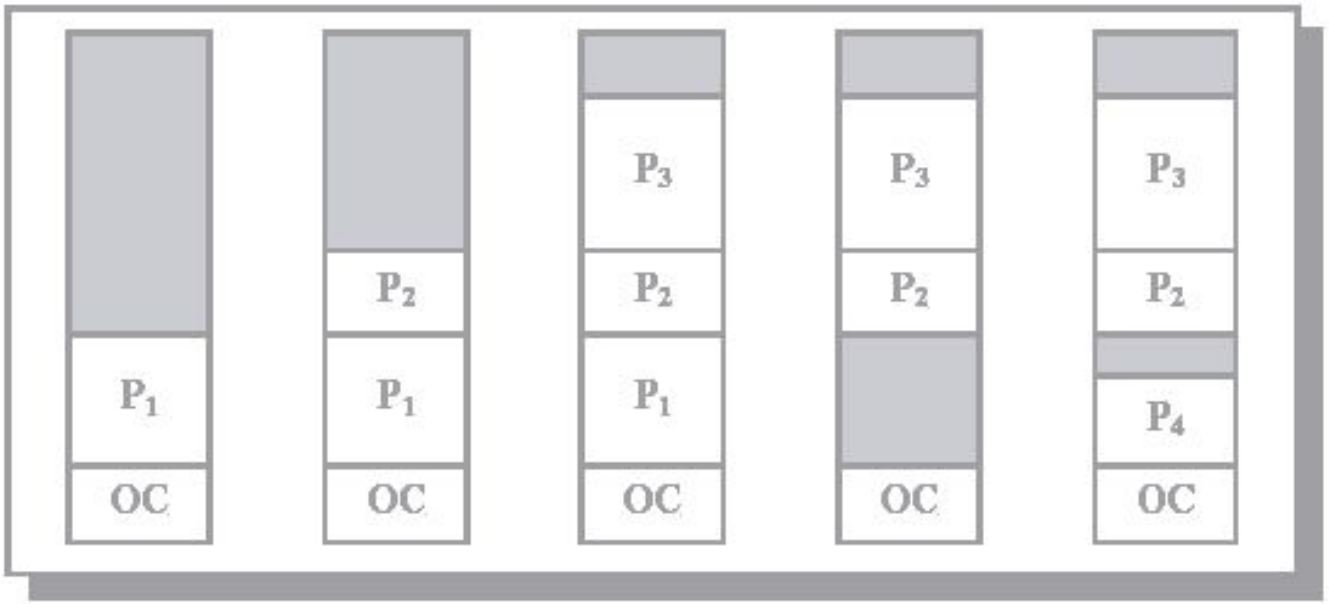


Динамическое распределение. Свопинг.

- В системах с разделением времени возможна ситуация, когда память не в состоянии содержать все пользовательские процессы.
- В таких случаях используется свопинг (swapping) – перемещению процессов из главной памяти на диск и обратно целиком. Частичная выгрузка процессов на диск осуществляется в системах со страничной организацией (paging).
- Выгруженный процесс может быть возвращен в то же самое *адресное пространство* или в другое. Это ограничение диктуется методом *связывания*. Для схемы *связывания* на этапе выполнения можно загрузить процесс в другое место памяти.

Схема с переменными разделами

- Типовой цикл работы менеджера памяти состоит в анализе запроса на выделение свободного участка (раздела), выборе его среди имеющихся в соответствии с одной из стратегий (первого подходящего, наиболее подходящего и наименее подходящего), загрузке процесса в выбранный раздел и последующих изменениях таблиц свободных и занятых областей.
- Аналогичная корректировка необходима и после завершения процесса. *Связывание адресов* может осуществляться на этапах загрузки и выполнения.



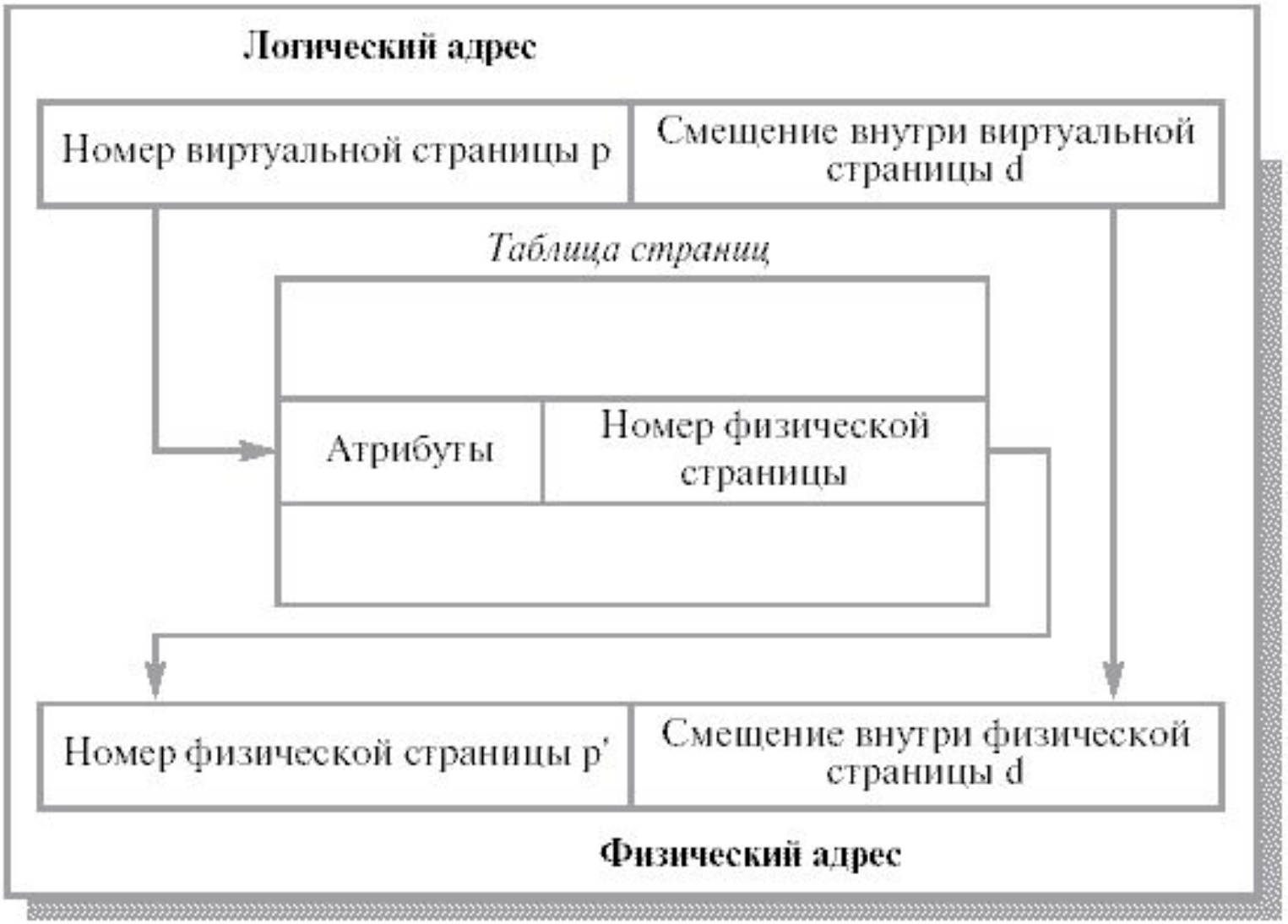
Страничная организация

- В случае страничной организации памяти (или paging) как логическое *адресное пространство*, так и физическое представляются состоящими из наборов блоков или *страниц* одинакового размера.
- При этом образуются логические *страницы* (page), а соответствующие единицы в *физической памяти* называют страничными кадрами (page frames). *Страницы* (и страничные кадры) имеют фиксированную длину, обычно являющуюся степенью числа 2, и не могут перекрываться.
- Каждый кадр содержит одну *страницу* данных. При такой организации *внешняя фрагментация* отсутствует, а потери из-за *внутренней фрагментации*, поскольку процесс занимает целое число *страниц*, ограничены частью последней *страницы* процесса.

Связь логического и физического адресов

- Логический адрес в страничной системе – упорядоченная пара (p,d), где p – номер *страницы* в виртуальной памяти, а d – смещение в рамках *страницы* p, на которой размещается адресуемый элемент.
- Разбиение *адресного пространства* на *страницы* осуществляется вычислительной системой незаметно для программиста.
- Адрес является двумерным лишь с точки зрения операционной системы, а с точки зрения программиста *адресное пространство* процесса остается линейным.

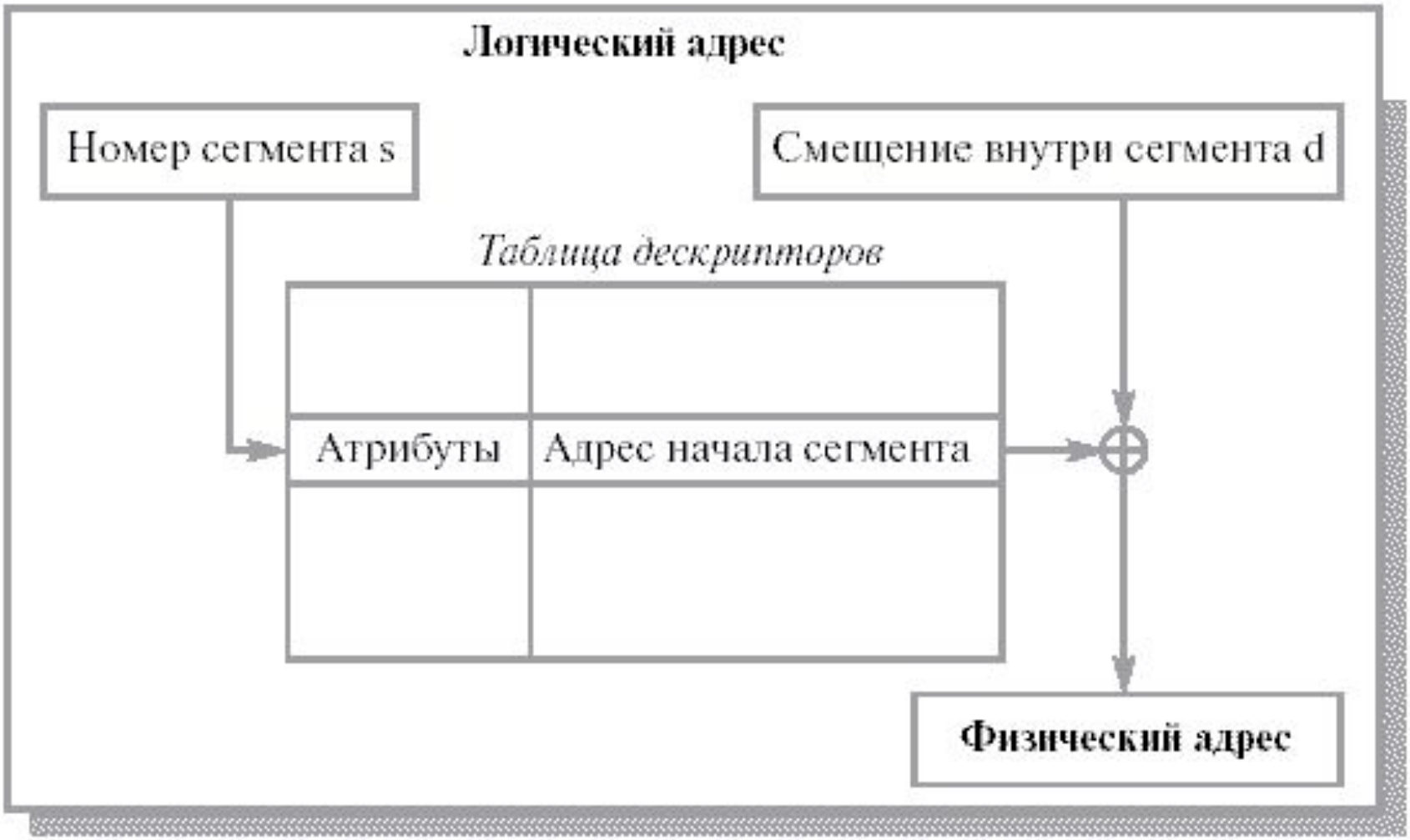
Схема адресации при страничной организации



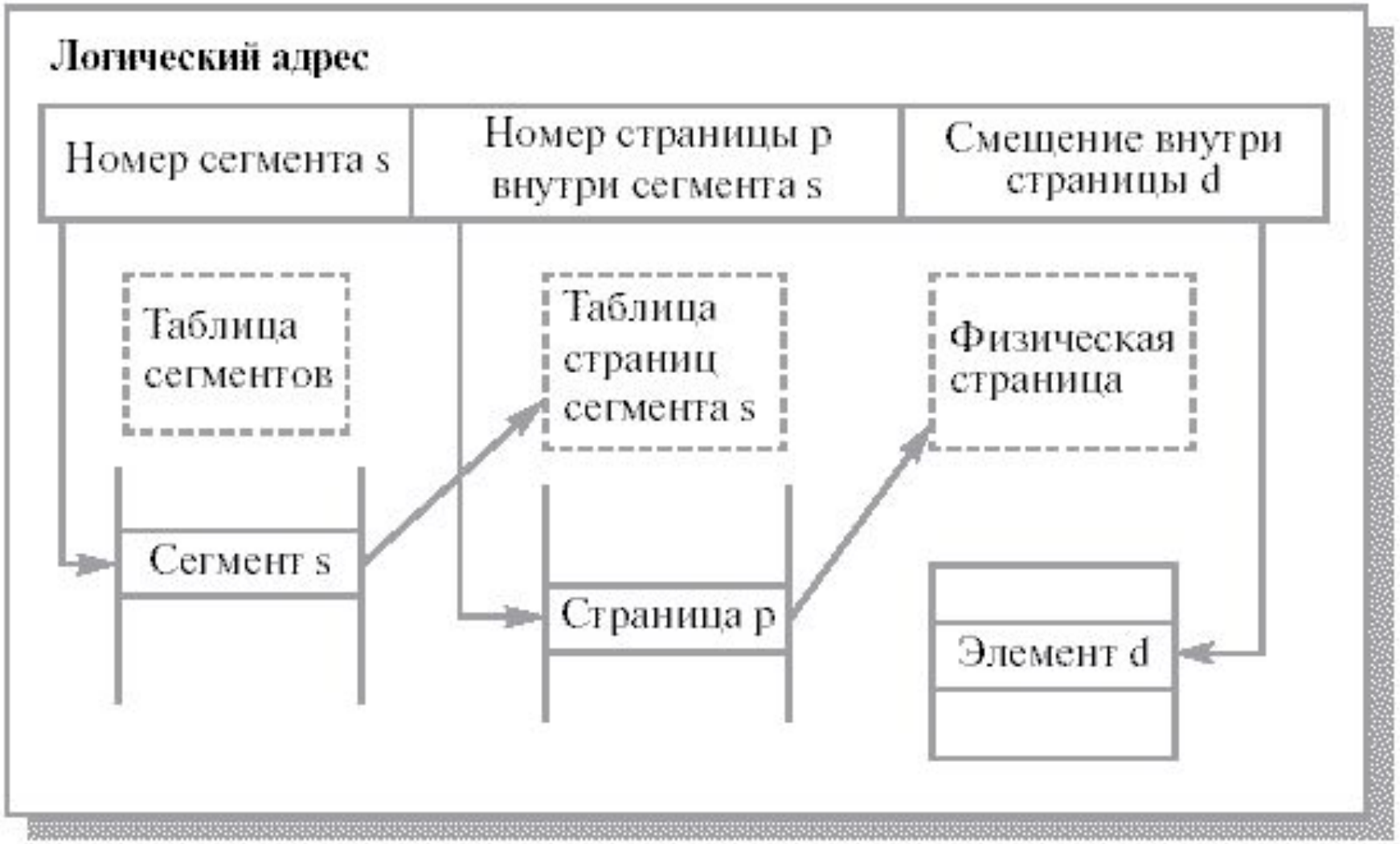
Сегментная и сегментно-страничная организация памяти

- *Сегменты*, в отличие от *страниц*, могут иметь переменный размер.
- Каждый *сегмент* – линейная последовательность адресов, начинающаяся с 0. Максимальный размер *сегмента* определяется разрядностью процессора (при 32-разрядной адресации это 2^32 байт или 4 Гбайт).
- Размер *сегмента* может меняться динамически (например, *сегмент* стека). В элементе таблицы *сегментов* помимо физического адреса начала *сегмента* обычно содержится и длина *сегмента*.
- Логический адрес – упорядоченная пара v=(s,d), номер *сегмента* и смещение внутри *сегмента*.

Преобразование логического адреса при сегментной организации



Формирование адреса при странично-сегментной организации памяти



Виртуальная память

- Разработчикам программного обеспечения часто приходится решать проблему размещения в памяти больших программ, размер которых превышает объем доступной оперативной памяти.
- Развитие архитектуры компьютеров и расширение возможностей операционной системы по управлению памятью позволило переложить решение этой задачи на компьютер. Одним из подходов стало появление *виртуальной памяти* (virtual memory).

Концепция работы с виртуальной памятью

- Информация, с которой работает активный процесс, должна располагаться в оперативной памяти.
- В схемах *виртуальной памяти* у процесса создается иллюзия того, что вся необходимая ему информация имеется в основной памяти.
  - во-первых, занимаемая процессом память разбивается на несколько частей, например страниц;
  - во-вторых, логический адрес (логическая страница), к которому обращается процесс, динамически транслируется в физический адрес (физическую страницу);
  - и наконец, в тех случаях, когда страница, к которой обращается процесс, не находится в физической памяти, нужно организовать ее подкачку с диска.
- Для контроля наличия страницы в памяти вводится специальный *бит присутствия*, входящий в состав атрибутов страницы в *таблице страниц*.

Кэширование данных

- Для ускорения доступа к данным используется принцип кэширования. В вычислительных системах существует иерархия запоминающих устройств:
  - нижний уровень занимает емкая, но относительно медленная дисковая память;
  - оперативная память;
  - верхний уровень – сверхоперативная память процессорного кэша.
- Каждый уровень играет роль кэша по отношению к нижележащему.

Кэширование данных

- Каждая запись в кэш-памяти об элементе данных включает в себя:
  - Значение элемента данных;
  - Адрес, который этот элемент данных имеет в основной памяти;
  - Дополнительную информацию, которая используется для реализации алгоритма замещения данных в кэше и включает признак модификации и актуальности данных.

Списки, стеки, очереди

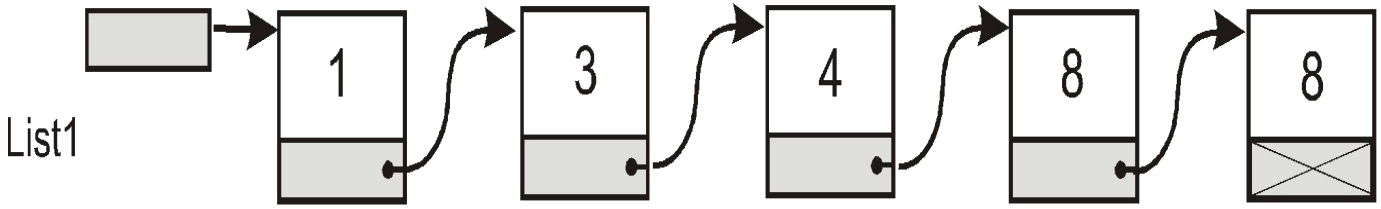
Списки

Самая простая динамическая структура данных — это линейный список.

Линейные списки находят широкое применение в приложениях, где непредсказуемы требования на:

- размер памяти, необходимой для хранения данных;
- большое число сложных операций над данными, особенно включений и исключений.

Вставка элементов в список



Вставка

- в начало списка (как в стек);
- в конец списка (как в очередь);
- в упорядоченный список с сохранением упорядоченности.

Упорядоченный список

Для получения упорядоченного списка вовсе необязательно сортировать его после построения, достаточно добавлять новый элемент таким образом, чтобы список все время оставался упорядоченным.

Такой метод позволяет иметь упорядоченный список на каждом шаге, не заботясь о том, закончились ли элементы, которые следует добавить в список.

Будем считать ключом, по которому производится упорядочивание, значение информационного поля.

Упорядоченный список

При добавлении элемента в список необходимо сначала найти место, куда его следует поместить. При этом возможны следующие варианты:

- список пуст;
- новый элемент меньше первого элемента существующего списка, и его следует поместить в начало;
- новый элемент требуется поместить в середину существующего списка.

В последнем варианте требуется сначала определить место, куда следует вставить новый элемент. Для этого нужно двигаться по списку до тех пор, пока либо не найдется элемент, больший или равный вставляемому, либо не будет достигнут конец списка.

Для удобства вставки используется дополнительный указатель — на предыдущий элемент.

Очередь

Очередь — это структура, работа с которой происходит по принципу *FIFO*:

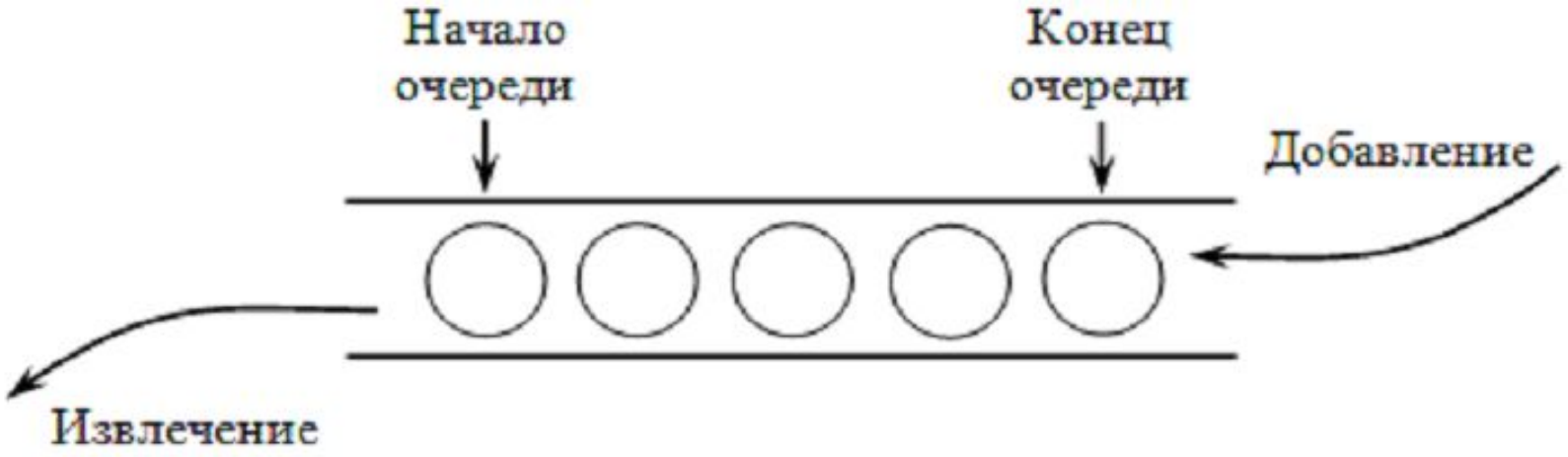
первым пришел — первым ушел

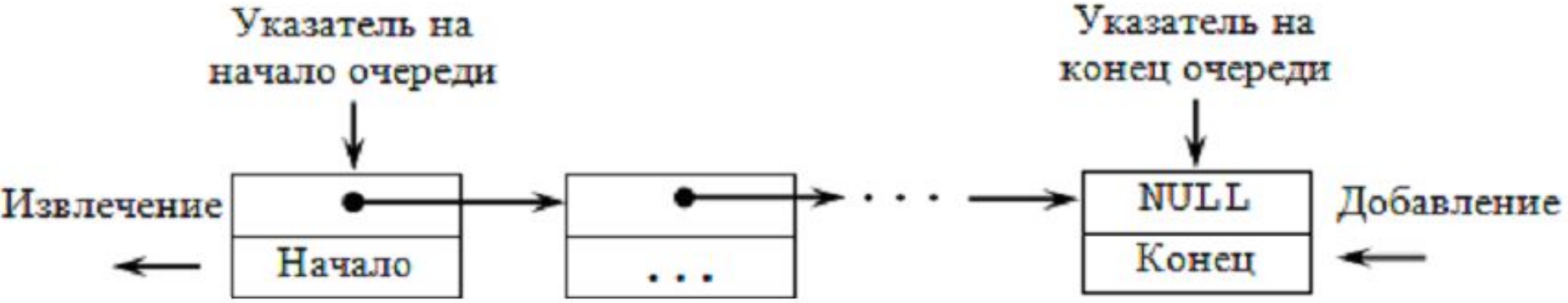
(от англ. *First — In — First — Out*).

Принцип работы очереди

Очередь — это структура, в которую новой элемент добавляется только с одной стороны. Эта сторона называется *концом* или *хвостом* (*tail*) очереди. Говорят, что *элемент добавляется в конец очереди*. Взятие элемента из очереди происходит с другой стороны — из *начала* (или из *головы (head))* очереди.

В качестве примера очереди в программировании можно назвать очередь процессов к разделяемому ресурсу под управлением операционной системы


Начало
очереди
Конец
очереди
Добавление
Извлечение


Указатель на
начало очереди
Указатель на
конец очереди
Извлечение
Начало
NULL
Конец
Добавление

Основные операции с очередью

- инициализация очереди;
- добавление элемента в очередь;
- проверка очереди на пустоту;
- взятие элемента из очереди.

В зависимости от характера решаемой задачи очередь можно организовать статически или динамически.

Если в процессе работы очередь то очень длинная (несколько десятков или сотен элементов), то короткая (один-два элемента), имеет смысл реализовать очередь с использованием динамической структуры (списка).

Если заранее известна максимальная длина очереди, то можно использовать статический массив.

Рассмотрим оба этих способа.

Динамическая организация очереди

При динамической реализации основой очереди является линейный односвязный список.

Для работы с очередью необходимы два указателя: на **голову** очереди и на ее **хвост**.

Очередь на массиве

В этом случае для хранения значений элементов очереди используется массив размерностью SizeQueue. Необходимо еще два указателя:

- *Голова*, указывающая на первый элемент в очереди, и
- *Хвост* — на элемент массива, в который можно поместить очередное значение.

Если очередь пуста, то *Голова* и *Хвост* указывают на один и тот же элемент массива.

Очередь на массиве

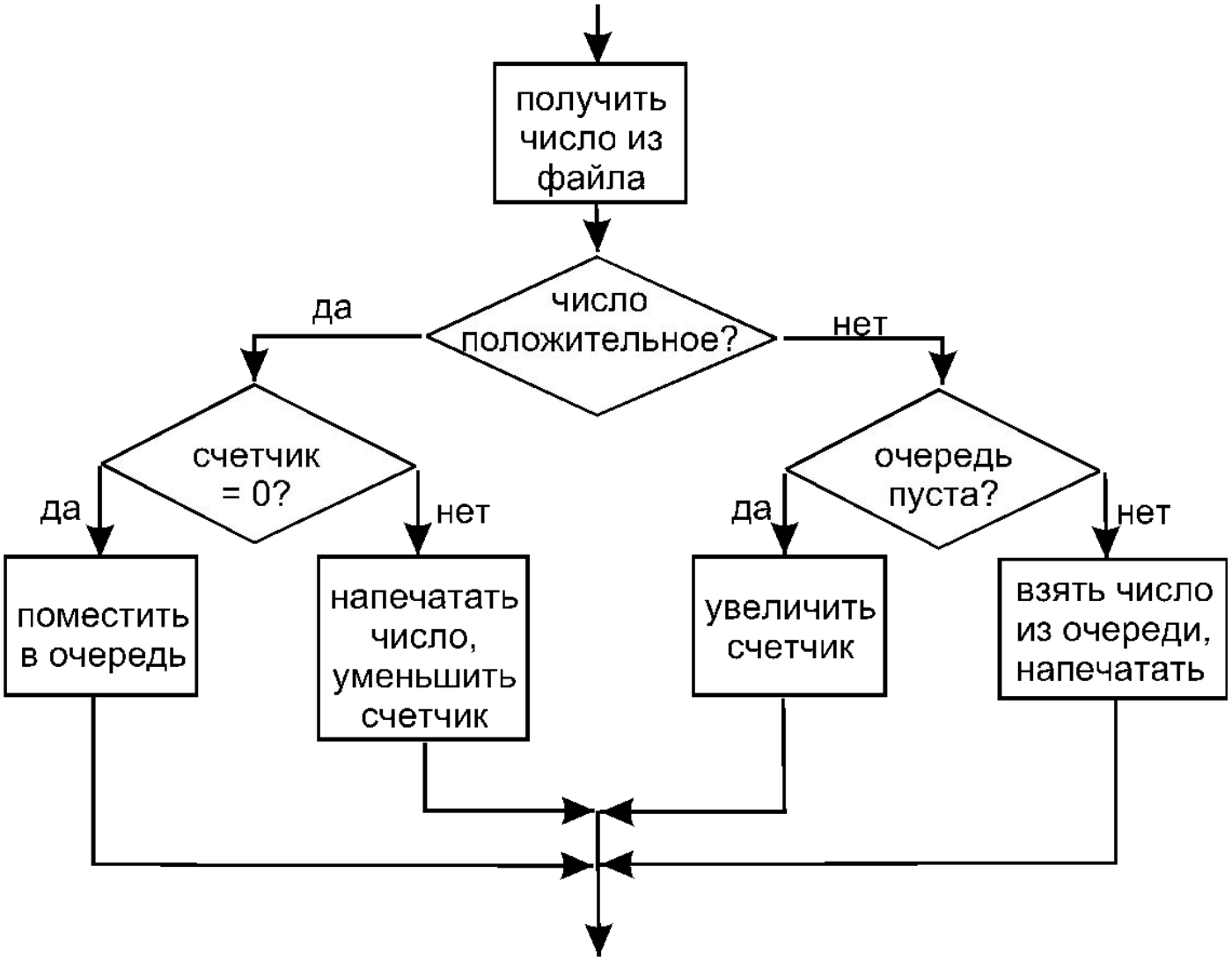
Поскольку в процессе работы при помещении значений в очередь массив заполняется с правой стороны, а при извлечении из очереди — освобождается слева, то имеет смысл сделать массив кольцевым. Это означает, что когда значение помещается в последний элемент массива, указатель *Хвост* перемещается к первому элементу массива (аналогично для *Головы*). Таким образом, *Хвост* может быть левее *Головы*. см. *QueueArray*

Очередь пуста — это значит, что в массиве не содержится ни одного значения. Очередь может стать пустой в процессе работы, при этом *Голова* и *Хвост* будут указывать на один и тот же элемент, но не обязательно на первый элемент массива. Поэтому для проверки очереди на пустоту необходим еще счетчик элементов очереди, который равен нулю, если очередь пуста, и имеет значение *SizeQueue,* если очередь забита полностью.

Задача

Завод скоропортящейся продукции, например, глазированных сырков, имеет склад, куда поступает готовая продукция: коробки с сырками. В магазин по заявке следует отправить первой ту коробку, которая раньше всех поступила на склад. Но если заявок поступило много и склад опустел, то вывозить сырки можно сразу из цеха, минуя склад. Незаявленная продукция остается на складе. В отчете требуется представить характеристики коробок, отправленных на продажу.

Блок-схема задачи



Использование стека в программировании

Любая операционная система содержит так называемый ***системный стек***, куда помещаются адреса возврата и ряд других значений при вложенном вызове процедур и функций.

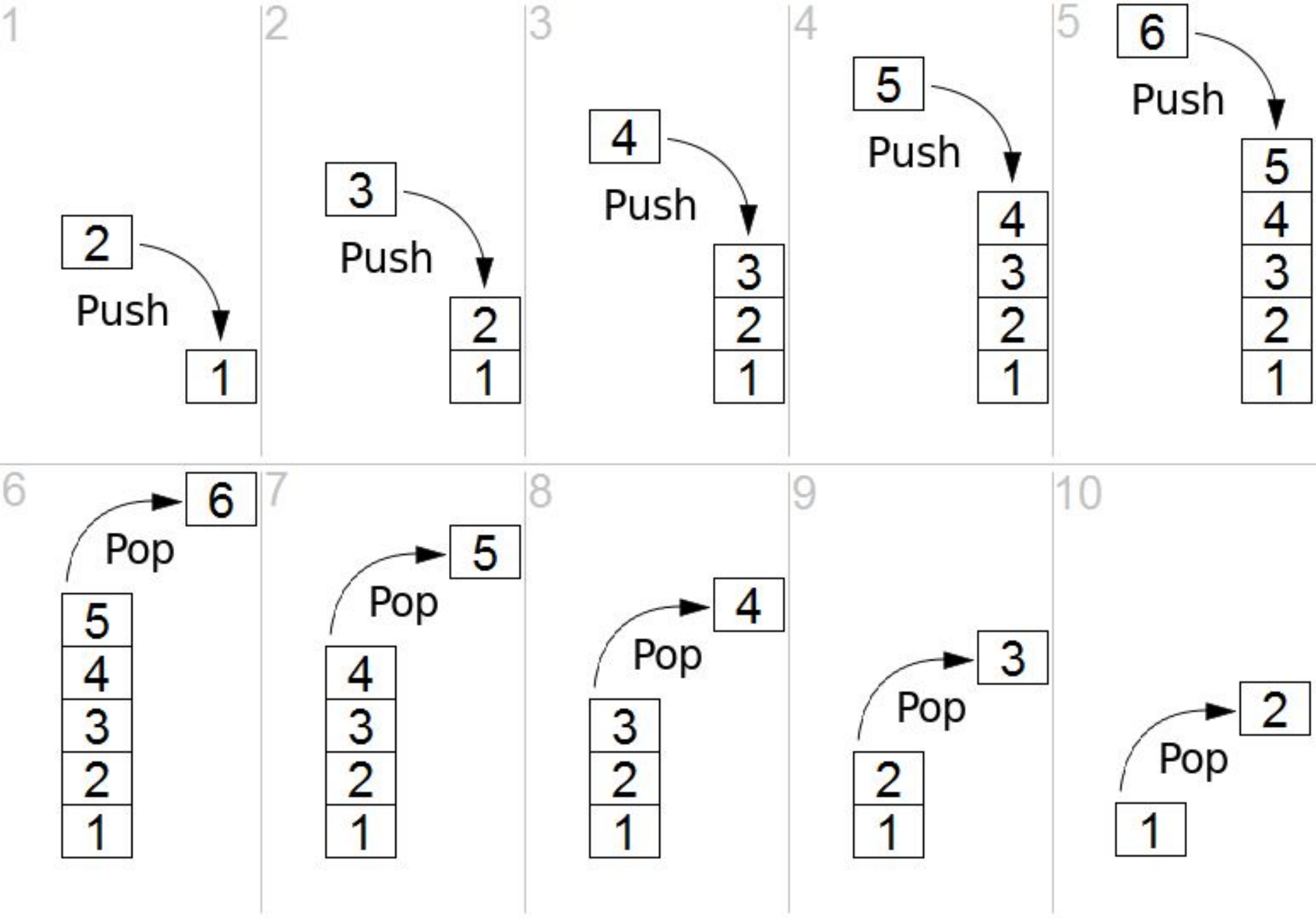
- системный стек: вызов процедур и функций
- компилятор на этапе синтаксического разбора текста программы
- интерпретатору стек необходим для вычисления выражений
- отмена операций (CTRL + Z)

Принцип работы стека

Стек — это структура, работа с которой происходит по принципу LIFO:

последним пришел — первым ушел

(от англ. Last — In — First — Out). Включение элемента в стек и исключение элемента из стека выполняются только с одной стороны, которая называется *вершиной* стека.


1
2
Push
1
2
3
Push
2
1
3
4
Push
3
2
1
4
5
Push
4
3
2
1
5
6
Push
5
4
3
2
1
6
6
Pop
5
4
3
2
1
7
5
Pop
4
3
2
1
8
4
Pop
3
2
1
9
3
Pop
2
1
10
2
Pop
1

Для работы со стеком необходимы следующие операции:

- инициализация стека, то есть подготовка структуры;
- включение нового элемента в стек (англ. *push* – заталкивать);
- проверка стека на пустоту;
- исключение элемента из стека (англ. *pop* – выталкивать).