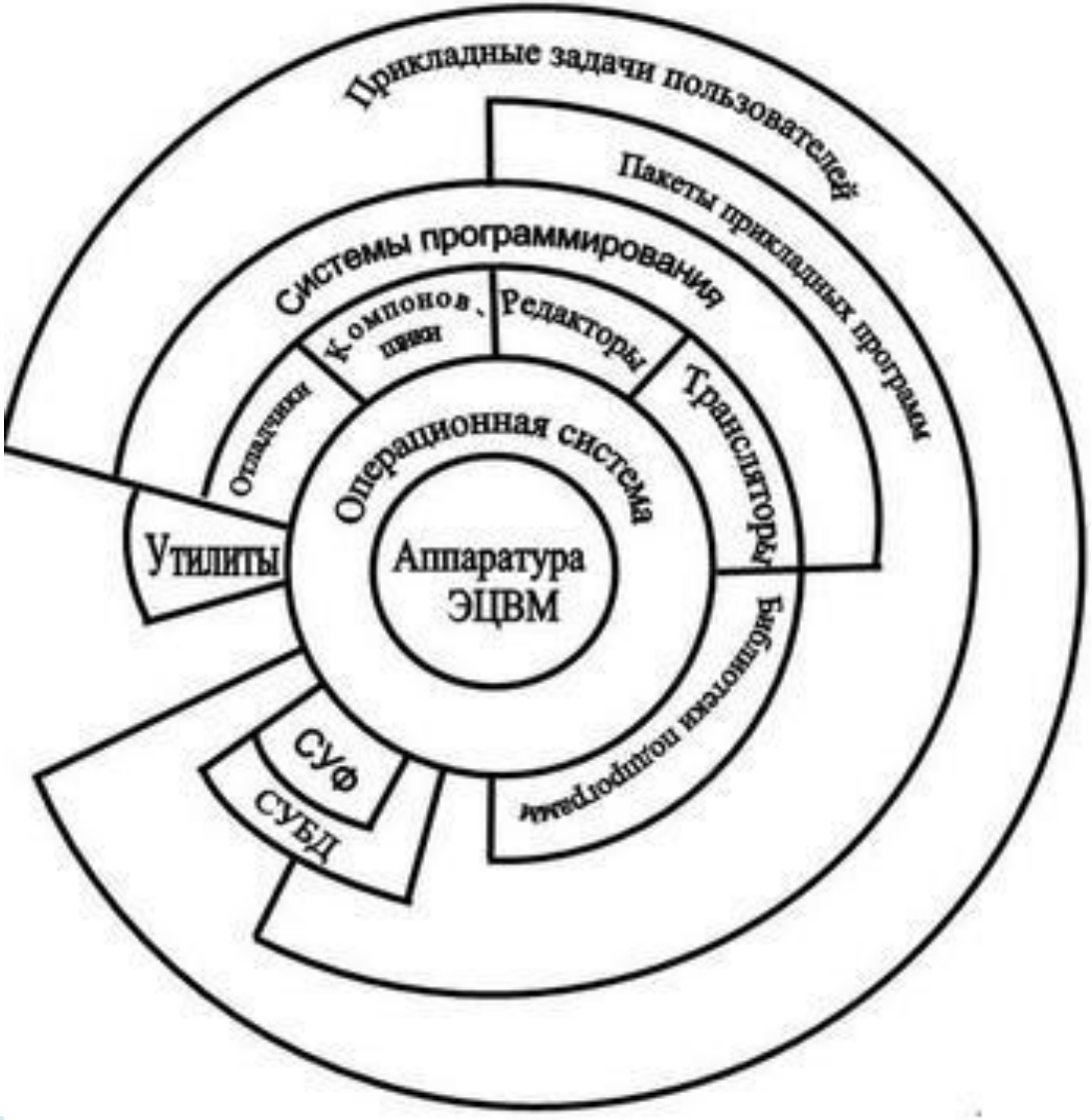
Системное программное обеспечение (System Software) - программы и комплексы программ, являющиеся общими для всех, кто совместно использует технические средства компьютера, и применяемые как для автоматизации разработки (создания) новых программ, так и для организации выполнения программ уже существующих.

1. Операционные системы (ОС).
2. Системы управления файлами (СУФ).
3. Интерфейсные оболочки.
4. Системы программирования.
5. Утилиты.

Обобщенная структура программного обеспечения вычислительной системы



Операционная система (ОС) - комплекс управляющих и обрабатывающих программ, который, с одной стороны, выступает как интерфейс между аппаратурой компьютера и пользователем с его задачами, а с другой – предназначен для наиболее эффективного использования ресурсов вычислительной системы и организации надёжных вычислений.

Основные функции ОС:

1. Управление элементами, блоками и устройствами ЭВМ;
2. Обеспечение хранения и доступа к информации;
3. Выделение ресурсов ПЭВМ (памяти, процессорного времени, внешних устройств и др.) для выполняемых процессов (управление ресурсами);
4. Организация взаимодействия между выполняемыми процессами (система прерываний);
5. Приём от пользователя (или от оператора системы) заданий или команд и их обработка.

Система управления файлами предназначена для организации более удобного доступа к данным, организованным как файлы.

Вместо низкоуровневого доступа к данным с указанием конкретных физических адресов нужной записи используется логический доступ с указанием имени файла и записи в нем.

Интерфейсная оболочка предназначена для удобства взаимодействия пользователя с ОС.
Назначение интерфейсной оболочки – расширить возможности по управлению ОС или изменить встроенные в систему возможности.

Виртуальная машина - это программное обеспечение, позволяющие смоделировать в одной операционной системе какую-либо другую операционную систему.

Система программирования – комплекс программных средств, предназначенный для разработки и отладки новых программ на определенном языке программирования.

Компоненты:

- транслятор (компилятор или интерпретатор);
- редактор;
- компоновщик;
- отладчик;
- библиотека подпрограмм и функций.

Утилита - программный продукт, предназначенный не для решения какой-либо прикладной задачи, а для решения вспомогательных задач. С помощью утилит можно обслуживать операционную систему, подготавливать для работы носители данных, выполнять перекодирование данных, осуществлять оптимизацию размещения данных на носителе и другие работы, связанные с обслуживанием вычислительной системы.

Основные принципы построения ОС:

1) модульности;
2) функциональной избирательности;
3) генерируемости ОС;
4) функциональной избыточности;
5) независимости программ от внешних устройств;
6) совместимости;
7) открытой и наращиваемой ОС;
8) переносимости;
9) обеспечения безопасности вычислений.

Вычислительный процесс (или задача) – это выполнение отдельной программы с ее данными на последовательном процессоре.

Вычислительный процесс – минимальный программный объект, обладающий собственными системными ресурсами (запущенная программа).

ОС контролирует деятельность, связанную с процессами:

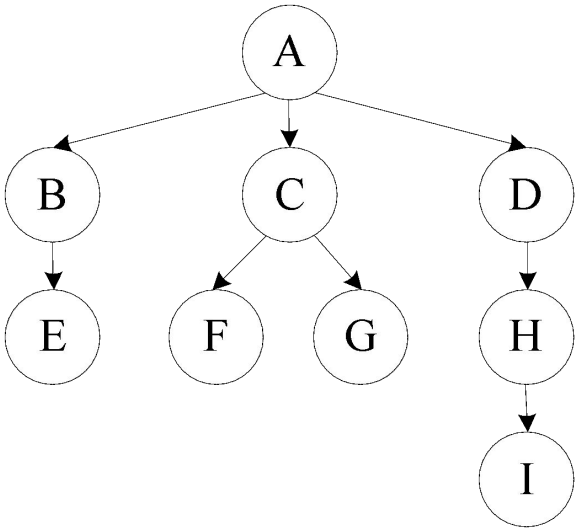
- создание и удаление процессов;
- планирование процессов;
- синхронизация процессов;
- коммуникация процессов;
- разрешение тупиковых ситуаций и др.

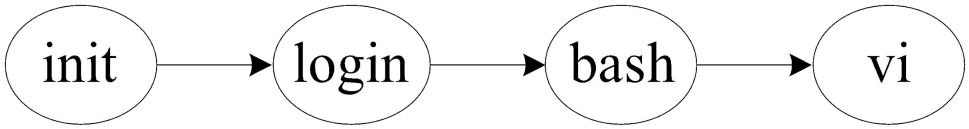
Понятие процесса включает:

- программный код;
- данные;
- содержимое стека;
- содержимое адресного [illegible] регистров процессора

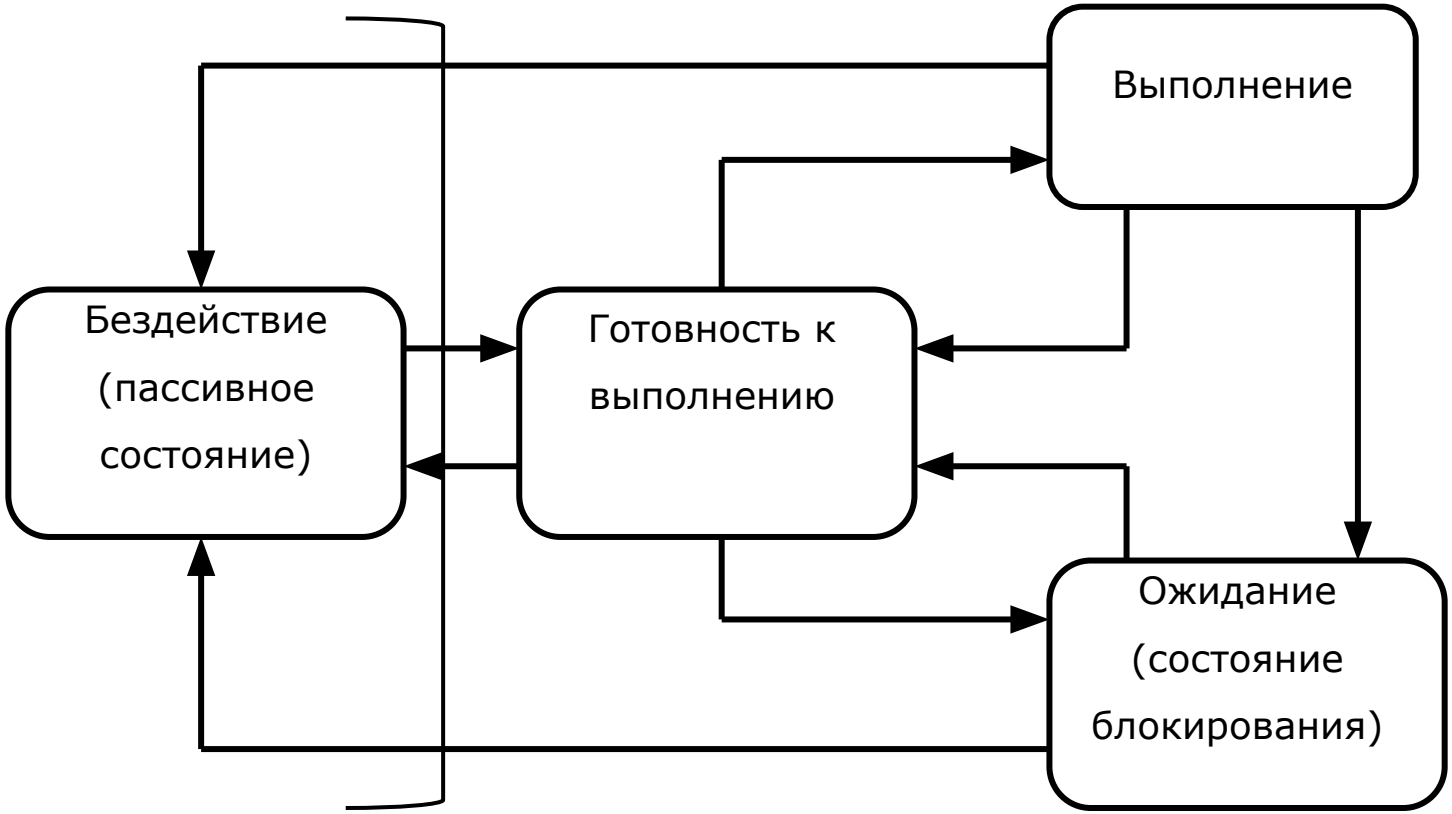
Вычислительный ресурс - повторно используемые, относительно стабильные и часто недостающие объекты, которые запрашиваются, используются и освобождаются процессами в период их активности.

Иллюстрации родительских и дочерних процессов





Динамика состояния процесса



Дескриптор процесса содержит информацию:

- идентификатор процесса (PID – process identificator);
- тип (или класс) процесса;
- приоритет процесса;
- переменную состояния;
- защищённую область памяти (или адрес такой зоны), в которой хранятся текущие значения регистров процессора, если процесс прерывается, не закончив работы (контекстом задачи);
- информацию о ресурсах, которыми процесс владеет и/или имеет право пользоваться (указатели на открытые файлы, информация о незавершенных операциях ввода/вывода, т.п.);
- место (или его адрес) для организации общения с другими процессами;
- параметры времени запуска (например, момент времени, когда процесс должен активизироваться, и периодичность этой процедуры) и др.

Многопоточность — свойство платформы (например, операционной системы или виртуальной машины) или приложения, состоящее в том, что процесс, порождённый в операционной системе, может состоять из нескольких потоков, выполняющихся «параллельно».

Особенности потоков (нити, треды):

- в основном не имеют своих собственных ресурсов;
- развиваются в одном виртуальном адресном пространстве;
- могут использовать файлы, виртуальные устройства и др. ресурсы, выделенные процессу.

Достоинства многопоточности при программировании:

- упрощение программы в некоторых случаях, за счет использования общего адресного пространства;
- меньшие относительно процесса временные затраты на создание потока;
- повышение производительности процесса за счет распараллеливания процессорных вычислений и операций ввода/вывода.

Межпроцессные коммуникации

Сигналы - программные прерывания, уведомляющие процесс о наступлении определенного события.

Сигналы не позволяют процессам обмениваться друг с другом какой-либо информацией.

При поступлении сигнала ОС определяет:

1) какому процессу он предназначен,

2) как процесс должен на него отреагировать.

Реакция процессов на сигнал:

- перехват;
- игнорирование;
- маскирование.

Классификация сигналов:

- синхронные, когда сам процесс инициирует сигнал;
- асинхронные, когда извне инициируется возникновение сигнала.

Конвейер (программный канал связи, транспортер) - средство, с помощью которого можно производить обмен данными между процессами.

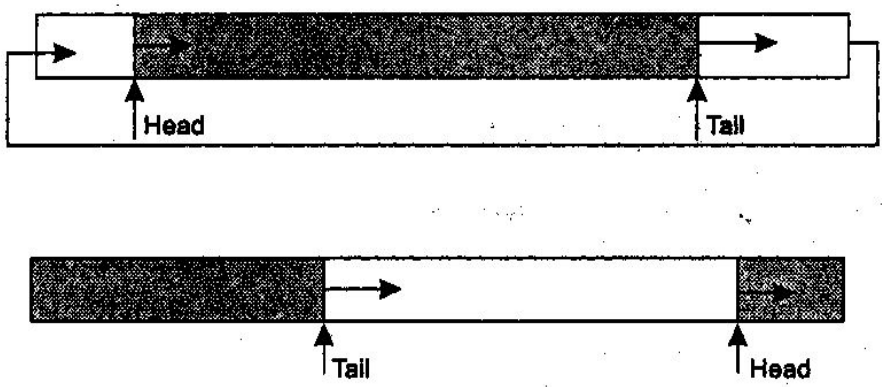
Особенности конвейеров:

- буферная память, работающая только по принципу FIFO;
- имеет определенный размер;
- данные непосредственно помещаются в конвейер;
- при чтении сообщения из конвейера оно удаляется из него;
- описывается идентификатором, размером и двумя указателями.

Основные системные запросы:

- функция создания конвейера;
- функция чтения из конвейера;
- функция записи в конвейер;
- завершение использования.

Иллюстрация конвейера



Особенности очередей сообщений :

- несколько дисциплин обработки сообщений (FIFO, LIFO, приоритетный, произвольный доступ);
- сообщение может быть прочитано несколько раз;
- присутствуют адреса сообщений в памяти и их размер.

Во время чтения из очереди задача-приёмник пользуется информацией:

- идентификатор процесса, который передал сообщение;
- адрес и длина переданного сообщения;
- ждать или нет, если очередь пуста;
- приоритет переданного сообщения.

Основные системные запросы:

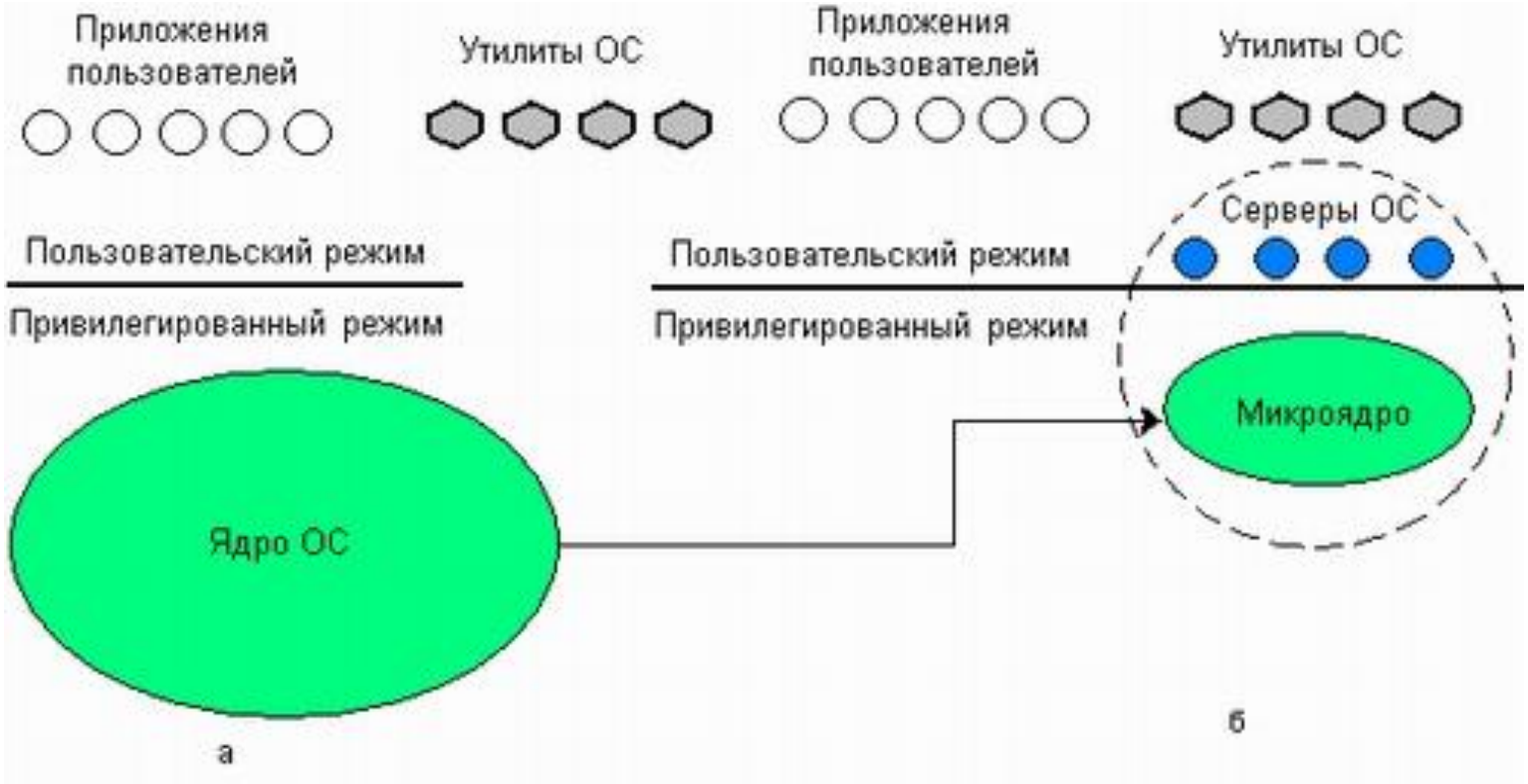
– создание новой очереди;
– открытие существующей очереди;
– чтение и удаление сообщения из очереди;
– чтение сообщения без его последующего удаления из очереди;
– добавление сообщения в очередь;
– завершение использования очереди;
– удаление из очереди всех сообщений;
– определение числа элементов в очереди.

Варианты структур ядра ОС

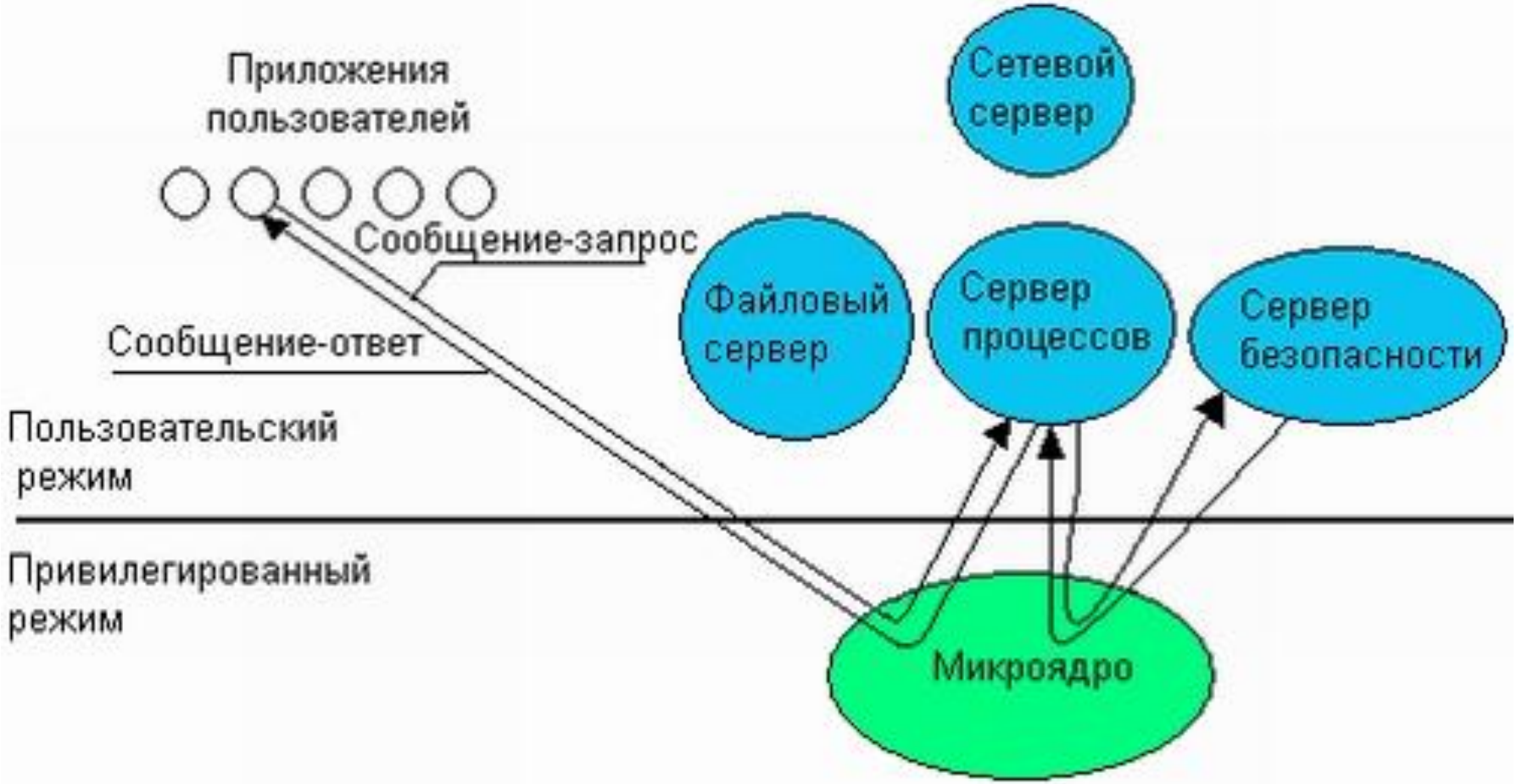
Микроядро – это минимальная стержневая часть операционной системы, служащая основой модульных и переносимых расширений.

Аппаратная поддержка как минимум двух режимов работы:

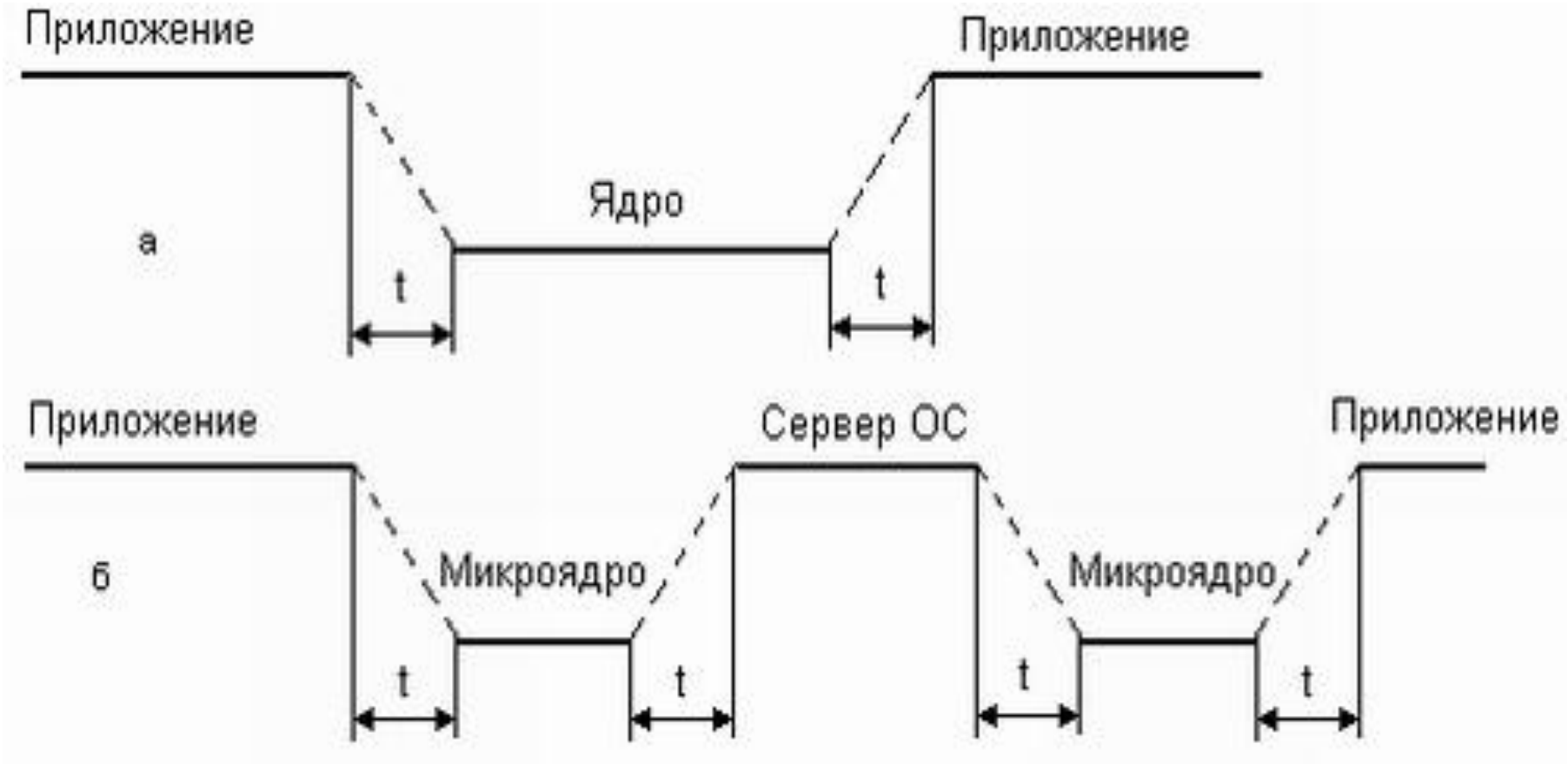
- пользовательский;
- привилегированный (режим ядра, режим супервизора).



Реализация системного вызова в микроядерной архитектуре



Смена режимов при выполнении системного вызова



Управление вводом/выводом

Главный принцип ввода/вывода – любые операции по управлению вводом/выводом объявляются привилегированными и могут выполняться только самой ОС.

Для обеспечения этого принципа вводятся ***два режима***:

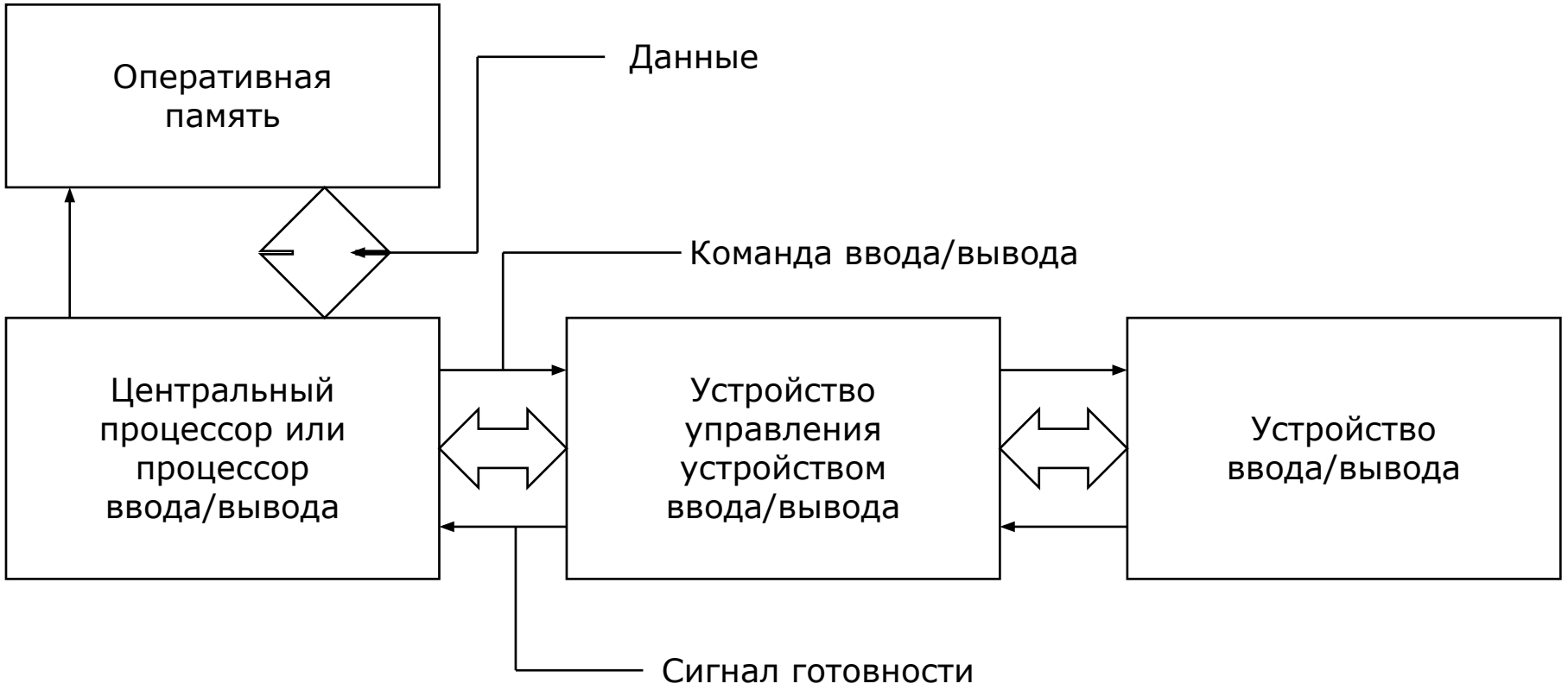
- режим пользователя, выполнение команд ввода/вывода запрещено;
- режим супервизора, выполнение команд ввода/вывода разрешено.

Основные задачи супервизора ввода/вывода :

- получение, проверка на корректность и выполнение запросов на ввод/вывод от прикладных задач и от модулей самой системы;
- планирование ввода/вывода;
- инициирование ввода/вывода;
- при получении сигналов прерывания передача управления соответствующей программе обработки прерывания;
- передача сообщений об ошибках, если они появляются;
- передача сигнала о завершении операции ввода/вывода.

Основные режимы ввода/вывода:

- режим обмена с опросом готовности;
- режим обмена с прерываниями.



Основные системные таблицы ввода/вывода:

1. Таблица оборудования. Элемент таблицы - UCB (unit control block) блок управления устройством ввода/вывода, содержащий:

- тип устройства, модель, символическое имя;
- способ подключения устройства (интерфейс, разъем, порты и др.);
- указание на драйвер, который должен управлять этим устройством, адрес секции запуска и секции продолжения драйвера;
- информация о том, используется или нет буферирование при обмене данными с этим устройством, «имя» (или просто адрес) буфера, если такой выделяется из системной области памяти;
- уставка тайм-аута и ячейки для счетчика тайм-аута;
- состояние устройства и др.

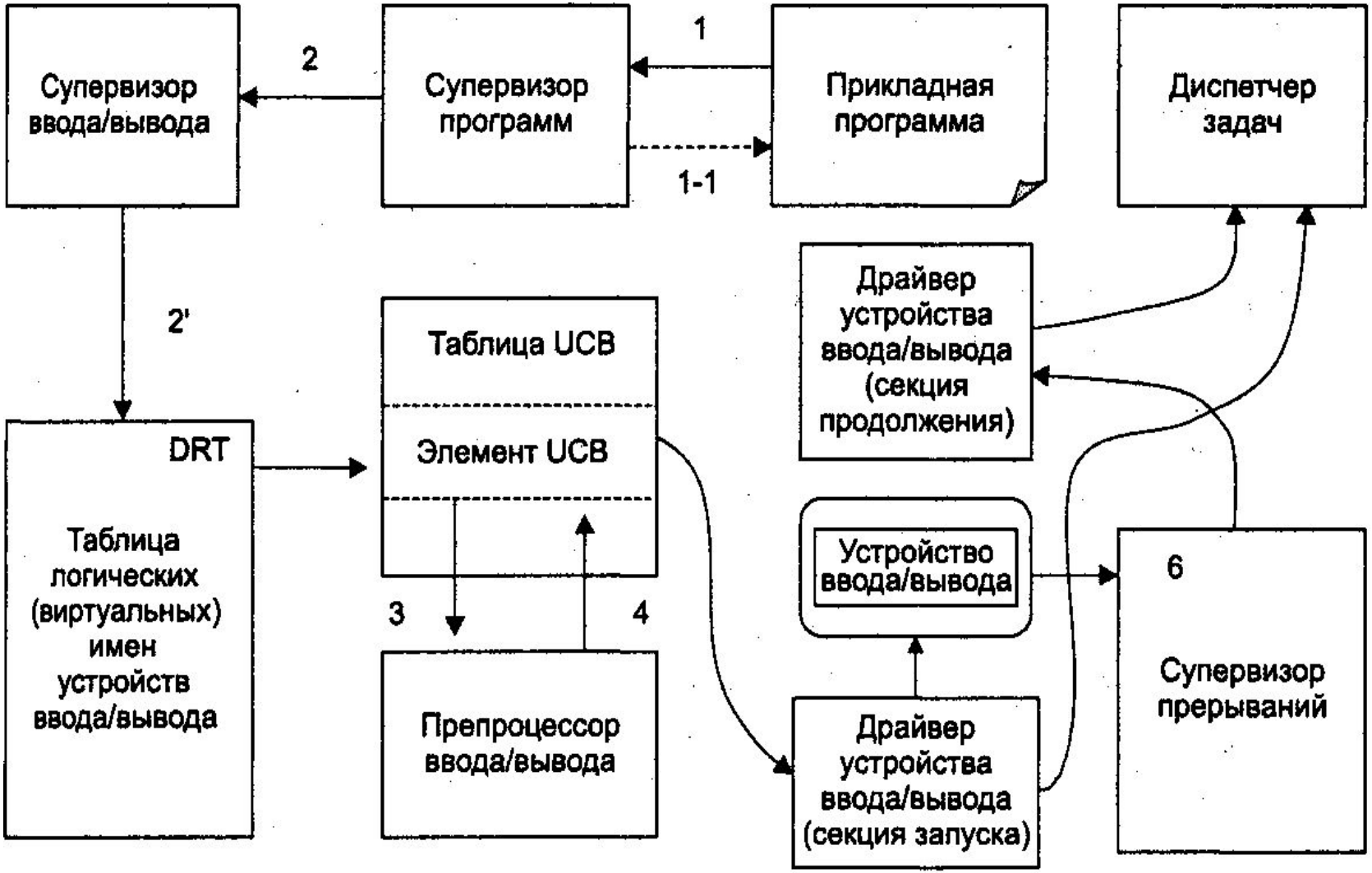
2. Таблица оборудования описания виртуальных логических устройств.

Назначение – установление связи между виртуальными (логическими) и реальными устройствами, описанными в таблице оборудования.

3. Таблица прерываний.

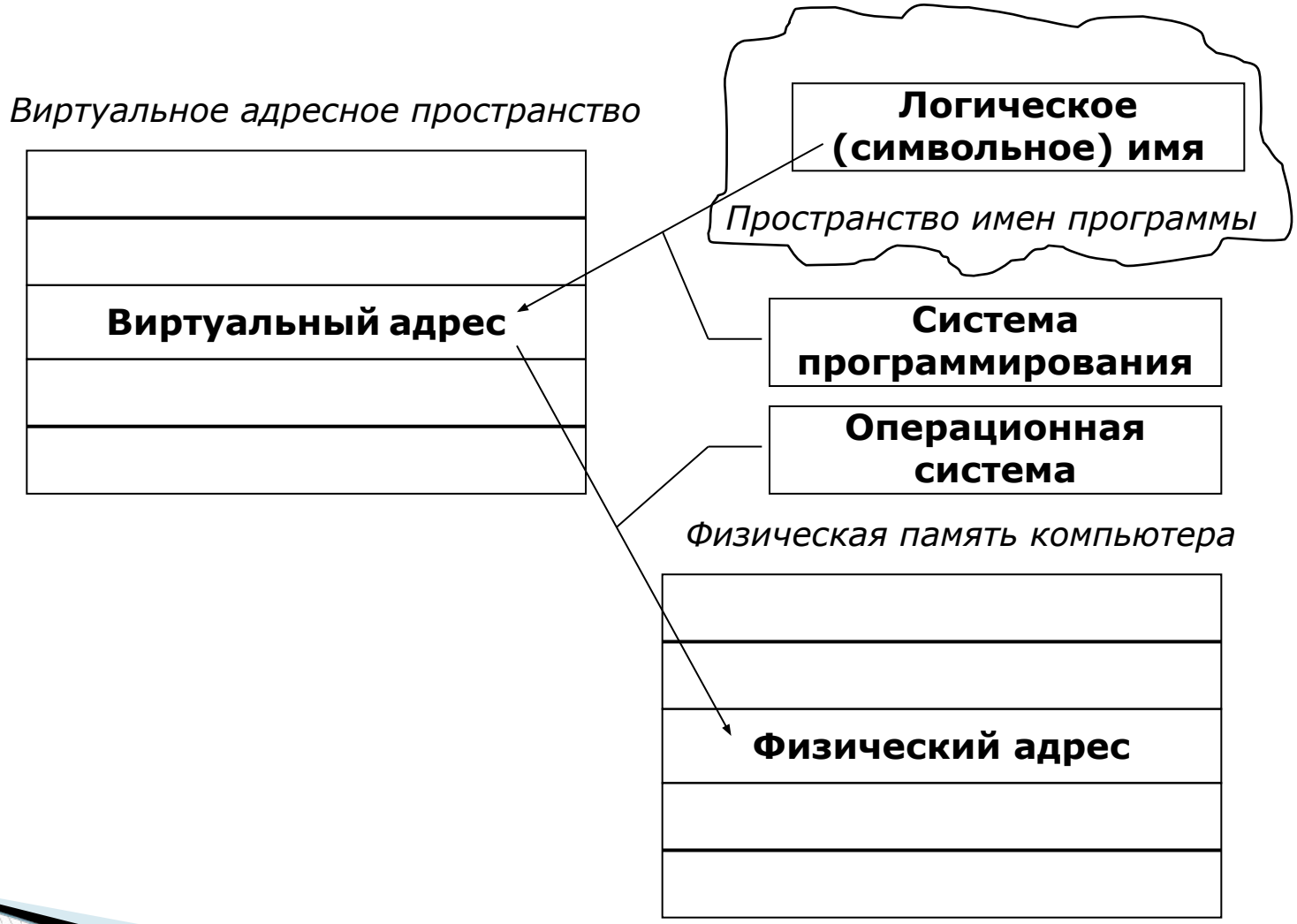
Назначение - установление связи между сигналами прерываний и таблицей оборудования.

Процесс управления вводом/выводом

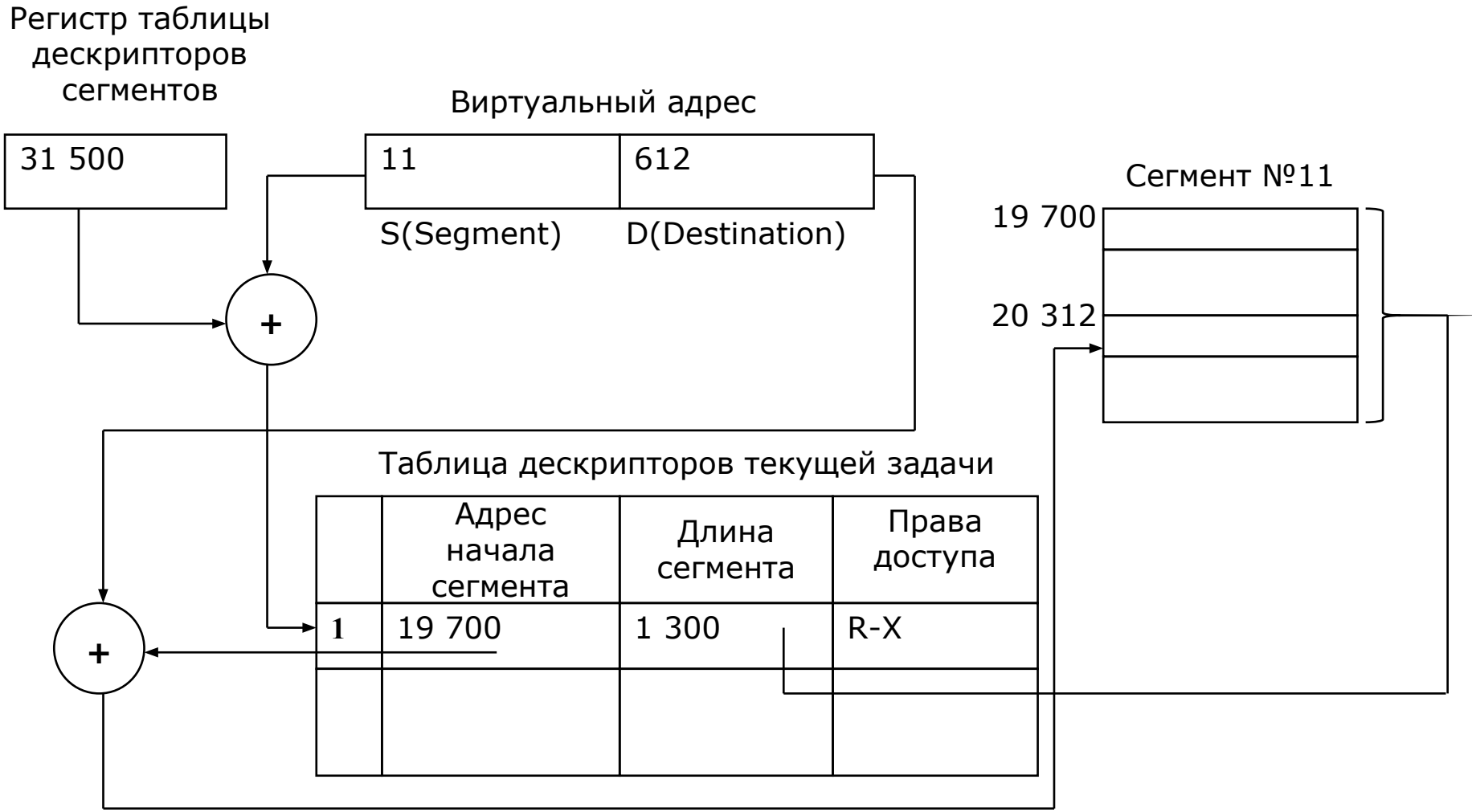


Организация и управление памятью

Отображение пространства имен на физическую память компьютера



Сегментный способ организации виртуальной памяти



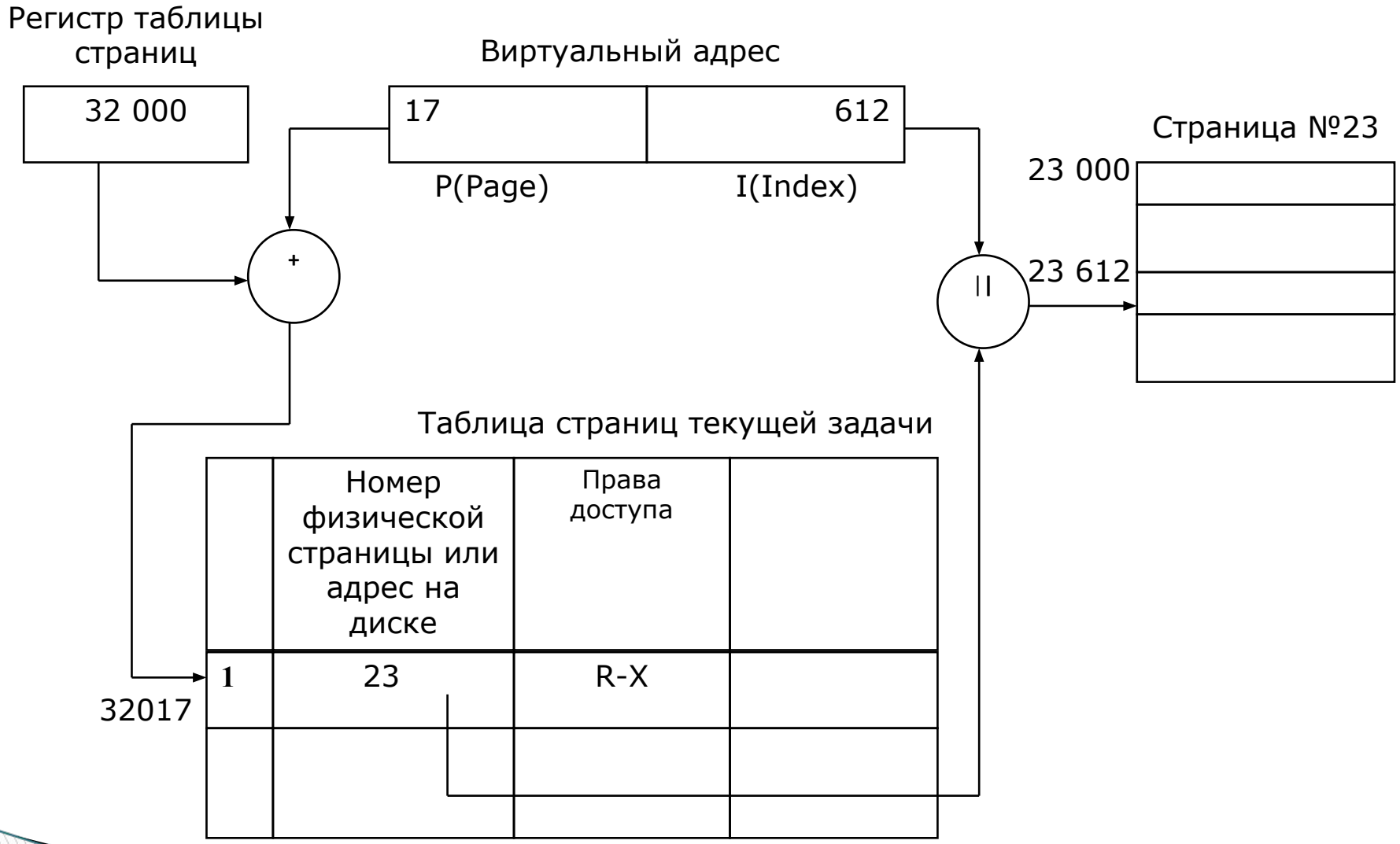
Основные моменты:

1. Программа разбивается на фрагменты разного размера.
2. Логически обращение к элементам программы производится как указание имени сегмента и смещения относительно его начала. Физически имя сегмента соответствует некоторому адресу, с которого этот сегмент начинается при его размещении в памяти.
3. Каждый сегмент имеет дескриптор сегмента, содержащий, в частности бит присутствия.
4. В оперативной памяти размещаются не все сегменты задачи, а только те, с которыми в настоящий момент происходит работа.

При загрузке необходимого сегмента в ОП поиск свободного места осуществляется на основе правил (первый подходящий, самый подходящий, самый неподходящий фрагмент).

Если свободного места нет, то происходит замещение. Выбор замещаемого сегмента осуществляется на основе дисциплин: FIFO, LRU (дольше всего неиспользуемый), LFU (используемый реже всех остальных), random.

Страничный способ организации виртуальной памяти



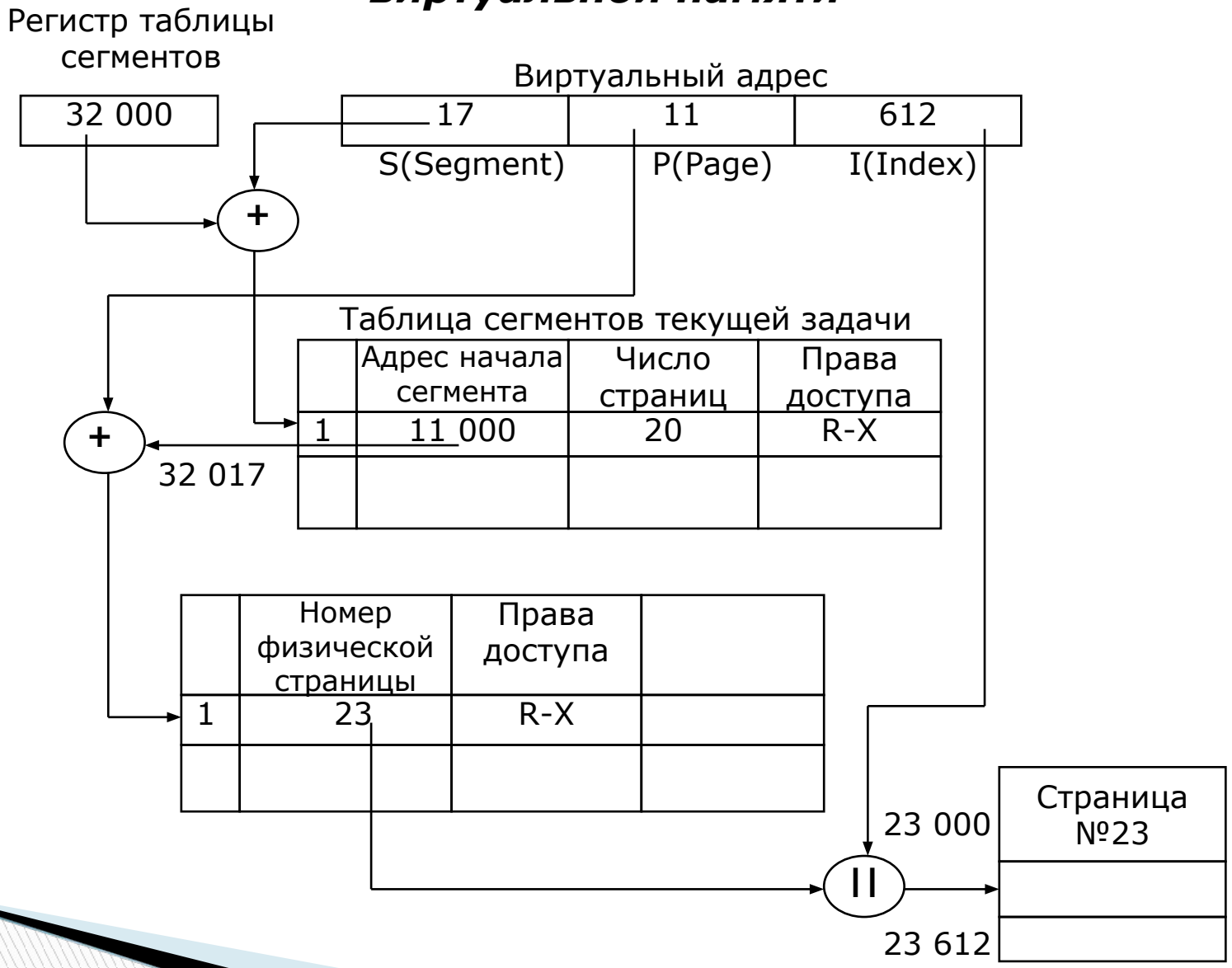
Основные моменты:

1. Программа разбивается на фрагменты одинакового размера, кратного степени двойки (без учета логических взаимосвязей).
2. Минимально возможная фрагментация.
3. Адрес (физический и виртуальный) определяется номером страницы и смещением внутри страницы.
4. Каждая страница имеет дескриптор.
5. В оперативной памяти размещаются не все страницы задачи, а только те, с которыми в настоящий момент происходит работа.

При загрузке необходимой страницы в ОП поиск свободного места осуществляется на основе правил (первый подходящий, самый подходящий, самый неподходящий фрагмент).

Если свободного места нет, то происходит замещение. Выбор замещаемой страницы осуществляется на основе дисциплин: FIFO, LRU (дольше всего неиспользуемый), LFU (используемый реже всех остальных), random.

Сегментно-страничный способ организации виртуальной памяти



Управление памятью включает в себя следующий набор основных функций:

- запрос на выделение блока памяти;
- освобождение памяти;
- изменение параметров блока памяти (например, память может быть заблокирована процессом либо предоставлена в общий доступ).