**Вопрос 25** (Понятие потока. Многопоточная модель процесса. Пример многопоточного приложения. Сравнение процесса и потока)

**Потоки**

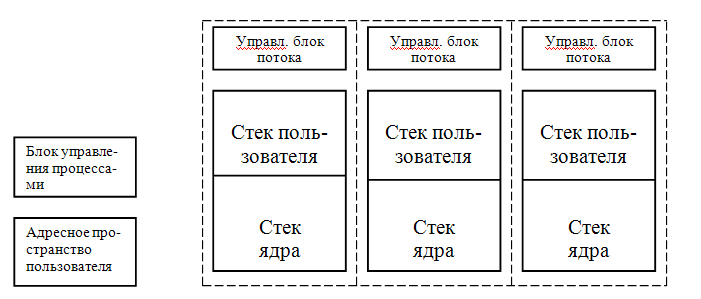
* Процесс во время своего выполнения занимает процессор и использует ресурсы (область памяти, файловую ситему,систему ввода вывода).
* Можно разбить процесс на отдельные части (мини – процессы ), называемые **ПОТОКАМИ**, которые будут по очереди выполняться процессором под управлением ОС, в рамках одного процесса, используя общие ресурсы предназначенные процессу.
* При этом у каждого потока будет иметься свой счетчик команд, свой стек, свое слово состояния.
* Потоки повышают эффективность использования процессора
* Каждый поток может иметь доступ к любому адресу памяти в пределах адресного пространства процесса
* Один поток может считывать данные из другого потока, процесса, стека другого потока, записывать туда свои данные и даже стирать оттуда данные.
* Защита между потоками отсутствует, потому что:  
  1) ее невозможно осуществить  
  2) в ней нет необходимости.
* Каждый поток может использовать одни и те же открытые файлы, дочерние процессы основного процесса.

[Необходимость создания потоков:

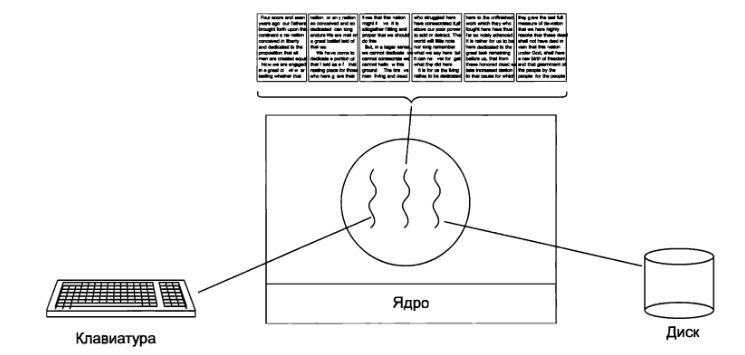
* Часто возникает необходимость распараллеливания вычислительного процесса (например при использовании блокирующих функций);
* Не все задачи удобно распараллеливать с помощью процессов (должны быть общие адресные пространства);
* Когда потоки работают в рамках одного центрального процессора, они не приносят никакого прироста производительности, но когда проводятся значительные вычисления, а также значительная часть времени тратится на ожидание ввода-вывода, наличие потоков позволяет этим действиям перекрываться по времени, ускоряя работу приложения.  
  Потоки весьма полезны для систем, имеющих несколько центральных процессоров, где есть реальная возможность параллельных вычислений.]

**!!!Многопоточная модель процесса**

Для каждого потока создаётся свои отдельные стеки и свои управляющие блоки, в котором содержится значение регистров процессора, приоритет и другая информация о состоянии потока.



**Пример многопоточного приложения**

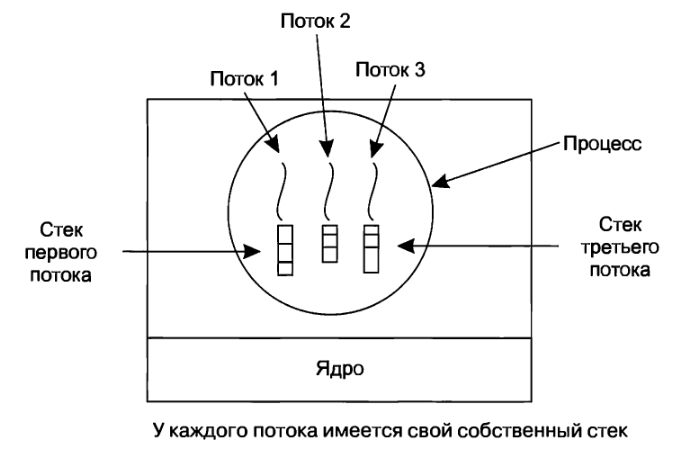
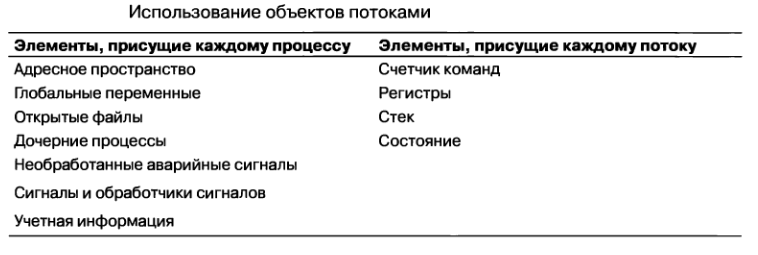


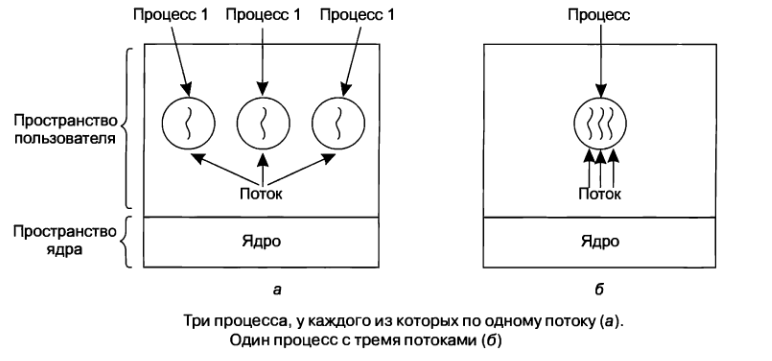
Текстовый процессор использующий три потока:

– поток чтения данных с клавиатуры;

– поток вывода на экран;

– поток сохранения на диск





**Вопрос 26 (**Классификация памяти. Порядок следования байт)

**1) Классификация по способу доступа к информации:**  
**-Память с произвольным доступом (электронная память)**  
Время доступа  не зависит от местоположения искомого участка памяти (одинаково для всех ячеек памяти).

**-Прямой циклический доступ**

При обращении к дисковой памяти используется прямой циклический доступ. Носитель информации непрерывно вращается, поэтому возможность обращения к одному и тому же участку памяти является циклической.

**-Последовательный доступ**   
Последовательный доступ к данным возможен при использовании в качестве носителя магнитной ленты, где последовательный просмотр участков носителя необходим для нахождения нужных данных.

**2) Классификация по способу доступа к информации:**

**- Адресная память.** «Классическая память»

**- Безадресная память**

При обращении к безадресной памяти в команде обращения к памяти адрес ячейки не задается .

**- Ассоциативная память -** поиск информации ведется по признаку путем сравнения тегов всех ячеек памяти с ассоциативным признаком.   
 **- Стековая память** - адрес ячейки памяти отслеживает специальный адресный регистр указатель стека. При обращении к стеку устанавливается адрес из этого регистра

**3) Классификация по способу хранения информации:**

**- Статическая память;**

**- Динамическая память;**

**- Энергонезависимая память:**

* Однократно программируемая;
* Многократно программируемая память (флэш-память);

**- Память на жестких дисках;**

**- Память на оптических дисках;**

**- Память на магнитных носителях**Информация хранится в виде намагниченных в определенном направлении участков ферромагнитной поверхности диска или магнитной ленты.

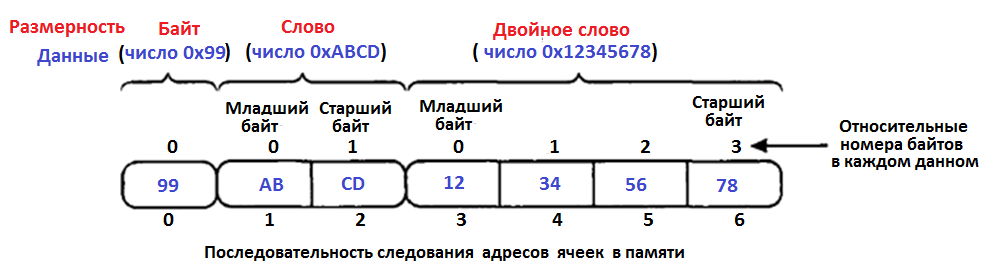
**- Оптическая  память**   В оптических внешних запоминающих устройствах информация записывается в виде участков, имеющих разные коэффициенты рассеяния света направленного луча лазера.

**Порядок следования байт:**

* Прямой
* Обратный
* Настраиваемый

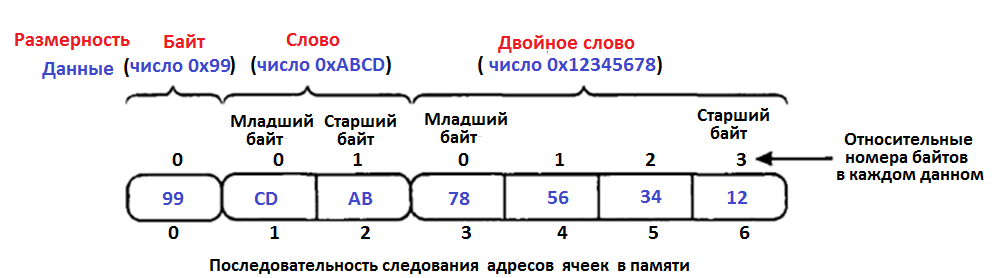
**Прямой порядок** следования байтов (big endian) (слева направо)

* Если число занимает в памяти один байт во всех системах оно читается одинаково;
* Если занимает несколько байт, то оно будет размещаться по-разному. Если в первый байт записывается старшая цифра числа, то это big endian, если в первый байт записывается младшая цифра числа, то это little endian;
* Запись начинается со старшего и заканчивается младшим разрядом.

****

**Обратный порядок**

* Размещаются в памяти от младшего к старшему (справа налево), в порядке возрастания адресов ячеек памяти.



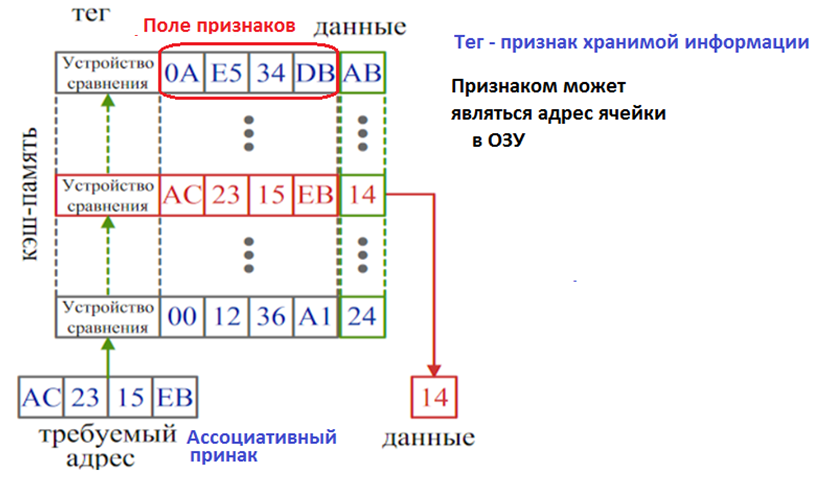
**Настраиваемый порядок следования:**

* Обычно порядок байтов выбирается программно во время инициализации операционной системы, но может быть выбран и аппаратно.

**Вопрос 27** (Ассоциативная память. Принцип работы кэш-памяти)

**Ассоциативная память**

**Ассоциативная память** - поиск информации ведется по признаку путем сравнения тегов всех ячеек памяти с ассоциативным признаком.



!!!!!!!!!!!!!

Запись в АП производится без указания конкретного адреса, в первую свободную ячейку. Для отыскания свободной ячейки СУЧЗ организует операцию считывания, в которой не замаскированы только служебные разряды, показывающие, как давно производилось обращение к данной ячейке, и свободной считается либо пустая ячейка, либо та, которая дольше всего не использовалась.  
Главное преимущество ассоциативных ЗУ определяется тем, что время поиска информации зависит только от числа разрядов в признаке поиска и скорости опроса

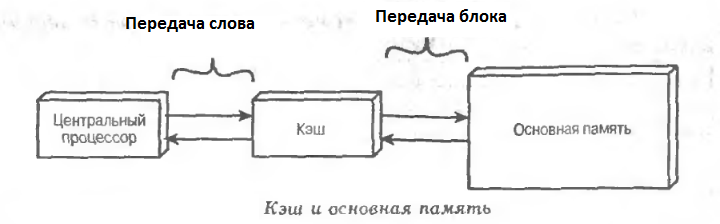
????

нным образцом и указывать на их соответствие или несоответствие друг другу. Признак, по которому производится поиск информации, будем называть *ассоциативным признаком*, а кодовую комбинацию, выступающую в роли образца для поиска, — *признаком поиска*. Ассоциативный признак может быть частью искомой информации или дополнительно придаваться ей. В последнем случае его принято называть *тегом* или *ярлыком*.

**КЭШ**

* Кэш использует временную и пространственную локальность
  + временная локальность - процессор, вероятнее еще раз обратится к тем данным, которые он недавно использовал
  + пространственная локальность – если процессор обращается к каким-либо данным, то, вероятнее, ему понадобятся расположенные рядом данные
* Команды программы, как правило, располагаются в памяти и выполняются последовательно.

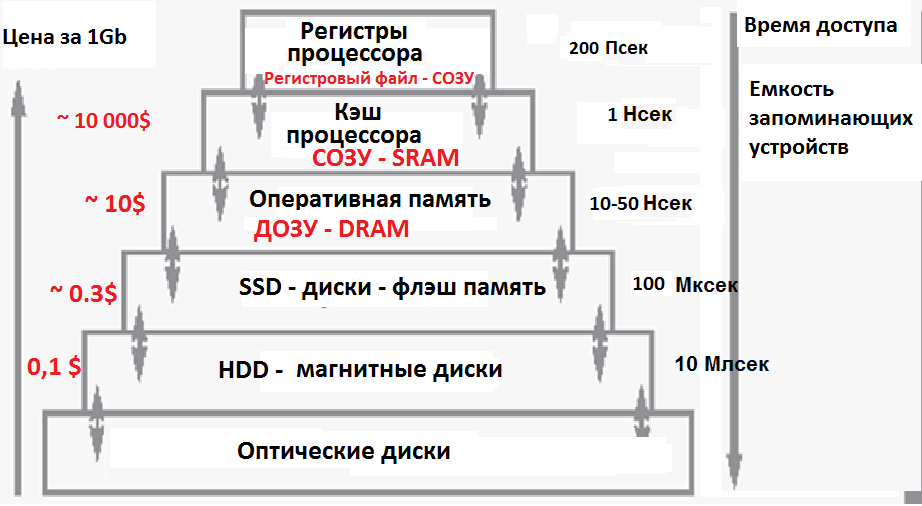
Идея кэша - когда определенная команда( байты) вызывается из памяти, она вместе с соседними командами (байтами - **целым** **блоком**) переносится в кэш-память, что позволяет при очередном запросе быстро обращаться к следующим командам (байтам) уже находящимися в кэше.



* Кэш L1 — до 128 кб внутренний кэш, интегрированный в ЦП.
  + Самый близкий к процессорному ядру
* Кэш L2 — 128 кб – 12мб — интегрируется в ЦП.
* Кэш L3 — 12мб и более. Располагается в процессоре или на материнской плате

**Вопрос 28** ( Иерархия памяти. Краткое описание физической реализации основных типов памяти)

**Иерархия памяти:**



* Регистр процессора — блок ячеек памяти, образующий сверхбыструю оперативную память внутри процессора; используется самим процессором и большей частью недоступен программисту;
* Кэш процессора — кэш, используемый микропроцессором компьютера для уменьшения среднего времени доступа к компьютерной памяти/\* Кэш— промежуточный буфер с быстрым доступом, содержащий информацию, которая может быть запрошена с наибольшей вероятностью.
* Оперативная память — энергозависимая часть системы компьютерной памяти, в которой во время работы компьютера хранится выполняемый машинный код (программы), а также входные, выходные и промежуточные данные, обрабатываемые процессором.
* Магнитные диски (Магнитные ленты) - используют явление магнитного гистерезиса для записи информации. /\*Гистерезис — свойство систем, мгновенный отклик которых на приложенные к ним воздействия зависит в том числе и от их текущего состояния, а поведение системы на интервале времени во многом определяется её предысторией.\*/

**Статическая память**

* Статическая память (SRAM – Static Random Access Memory ячейки реализуются на триггерах (6 транзисторов на триггер - дорого), занимают большую площадь на кристалле.
* Имеет высокую стоимость хранения информации.

**Динамическая память**

* **Матрица** запоминающих элементов (конденсаторов)
* Низкая стоимость. (два транзистора)
* Низкое быстродействие (конденсатор надо периодически подзаряжать специальными циклами регенерации. Во время подзарядки обращение к ячейке не возможно).
* При чтении конденсатор так же разряжается

**Запись** – ячейка выбирается сигналами выбора строки (транзистор VT1 – открывается) и столбца (конденсатор заряжается, что соответствует единице ).

**Чтение** - ячейка выбирается сигналами выборки строки (транзистор VT1 – открывается) и столбца . При этом, если конденсатор был заряжен то напряжение на шине столбца увеличивается на некоторую величину, что соответствует логической единице.

**Постоянная (Энергонезависимая) память**

* **Однократно программируемая (**ROM (Read Only Memory)**) только для чтения**   
     Запоминающим элементом является пережигаемая плавкая перемычка или полупроводниковый диод, играющий роль разрушаемой перемычки.

**Многократно программируемая память**

* EPROM*(Erasable Programmable Read Only Memory)* Многократно программируемая память со стиранием ультрафиолетовыми лучами
* EEPROM *(Electrically Erasable Programmable Read-Only Memory) электрически стираемая память* . Стирание информации производится путем подачи управляющего напряжения другого знака. В настоящее время строится на основе флеш - памяти

* В ERPROM используется транзистор с плавающим затвором

**Транзистор с плавающим затвором**

Плавающий затвор изолирован от остальных частей схемы и может хранить заряд долгое время

**FLASH - память – запись**

Если транзистор подать напряжение определенной полярности электроны могу накапливаться на плавающем затворе. Если на сток подается меньшее напряжение чем на управляющий затвор то часть электронов в канале получает энергию, достаточную чтобы преодолеть потенциальный барьер, который создается тонким слоем диэлектрика, и переходят(туннелируют) в область плавающего затвора.

**FLASH - память – стирание**

Для стирания (удаления заряда надо подать на транзистор напряжение обратной полярности). Чтобы удалить заряд с плавающего затвора (выполнить стирания ячейки памяти) на управляющий затвор подается высокое отрицательное напряжение, а на область истока подается положительное напряжение. Это приводит к тому, что электроны стекают из области плавающего затвора в область истока.

**FLASH - память – чтение**

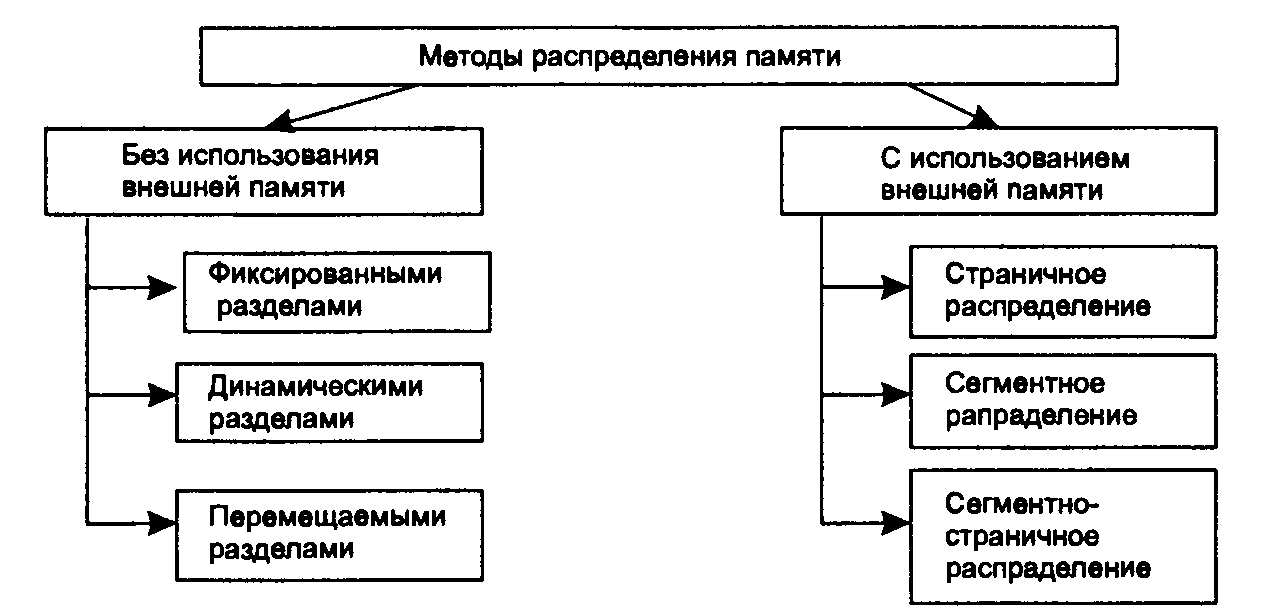
Если подать одно и тоже напряжение на управляющий затвор, то если:

* + транзистор откроется, заряда нет – это логическая ЕДИНИЦА
  + транзистор закрыт, заряд есть – это логический НОЛЬ

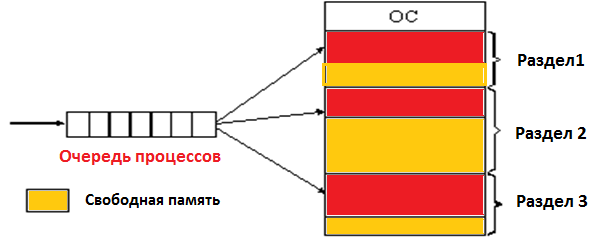
Если на плавающий затвор поместить избыточный отрицательный заряд (электроны) и подать положительное напряжение на управляющий затвор ,то этот заряд компенсирует создаваемое управляющим затвором электрическое поле и не даст образовываться каналу проводимости, транзистор будет закрыт , что соответствует логическому 0.

Если заряда на плавающем затворе нет, то при подаче управляющего напряжения на затвор транзистор сразу откроется, что соответствует логической 1.

**Вопрос 29** (Методы распределения памяти фиксированными, динамическими разделами и перемещаемыми разделами)

**Методы распределения памяти под процессы**

**Разделение фиксированными разделами**



* Вся память разбивается на фиксированное количество одинаковых или разных по величине разделов.
* Подсистема управления памятью в этом случае выполняет следующие задачи:

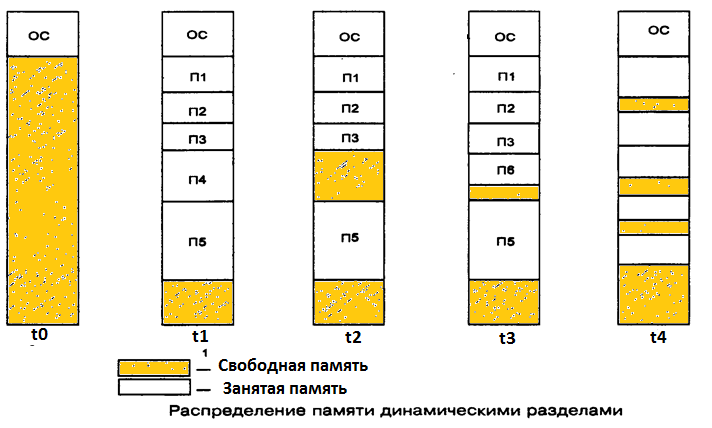
- сравнивает размер программы, поступившей на выполнение, и свободные разделы, выбирает подходящий раздел;

-осуществляет загрузку программы и настройку адресов.

Недостатки:

* Так как в каждом разделе может выполняться только одна программа, то уровень мультипрограммирования заранее ограничен числом разделов не зависимо от того, какой размер имеют программы.
* Если программа, имеющая небольшой объем, будет загружена в большой раздел она будет занимать только часть раздела, что приводит к неэффективному использованию памяти.
* Как загрузить программу размер которой превышает размер раздела?

**Распределение динамическими разделами**

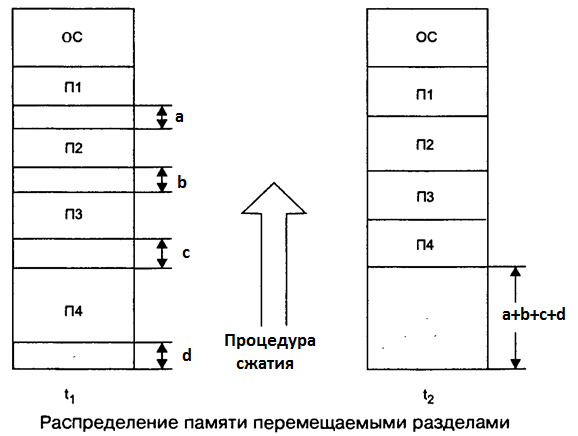


* Изначально память на разделы не разделена. Процессы целиком помещаются в память образуют условные разделы разделы.
* Количество разделов жестко не задано.

Недостатки:

* При удалении процесса и записи на его место меньшего процесса получается неиспользуемая память **между разделами (внешняя фрагментация** она меньше чем у схемы с фиксированными разделами)

**Распределение перемещаемыми разделами**



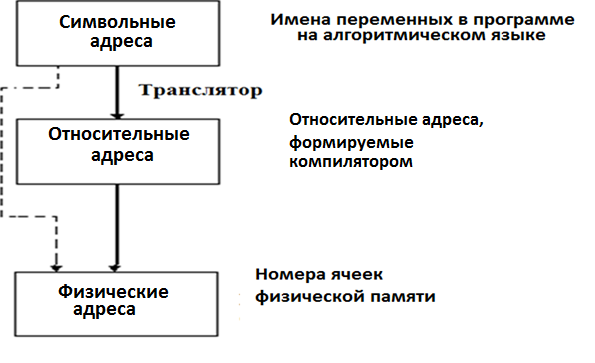
* ОС перемещает содержимое разделов из одного места памяти в другое
* Сжатие может выполняться или при каждом завершении процесса, или

когда для нового процесса нет свободного раздела нужного размера.

Недостаток – перемещение разделов занимает много времени.

**Вопрос 30** ( Типы адресов памяти (Символьные, относительные, физические))

**Типы адресов используемые процессами**



* Физические адреса соответствуют номерам ячеек оперативной памяти, где в действительности будет расположен процесс.
* Процессор, при обращении к памяти использует только физические адреса.
* Транслятор присваивает командам и данным относительные адреса, начиная с некоторого постоянного адреса одинакового для всех процессов.

Компилятор назначает **относительные адреса** командам и переменным (формирует таблицу соответствия символов и их адресов ) и сохраняет все это в объектном файле.

**Компоновка**

Работа компоновщика заключается в том, чтобы объединить все объектные файлы в один-единственный файл с машинным кодом, который называется исполняемым файлом. Компоновщик перемещает данные и команды в объектных файлах так, чтобы они не наслаивались друг на друга. Он использует информацию из таблицы символов для коррекции адресов перемещаемых глобальных переменных и меток.