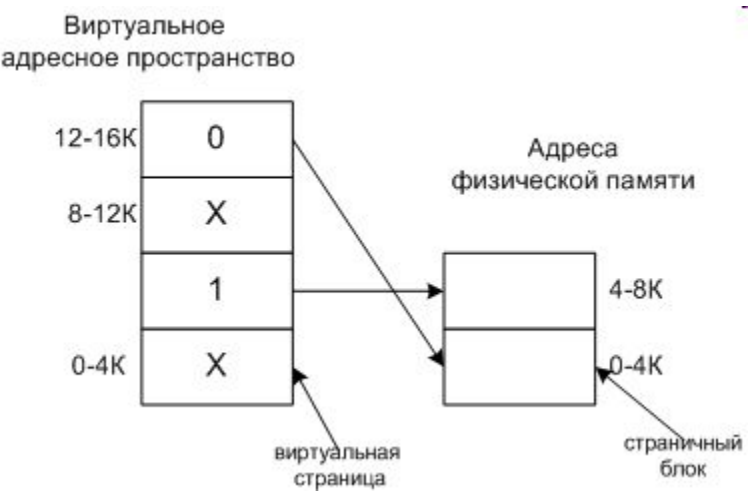
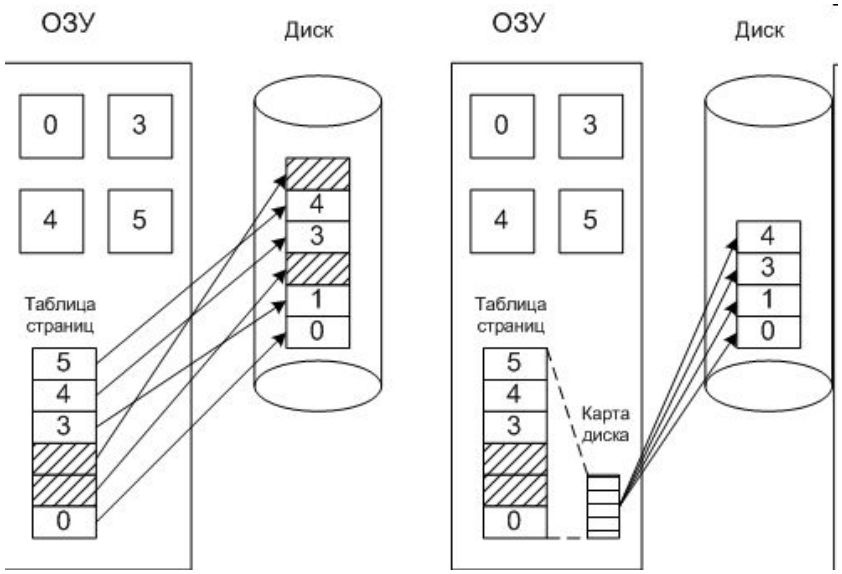
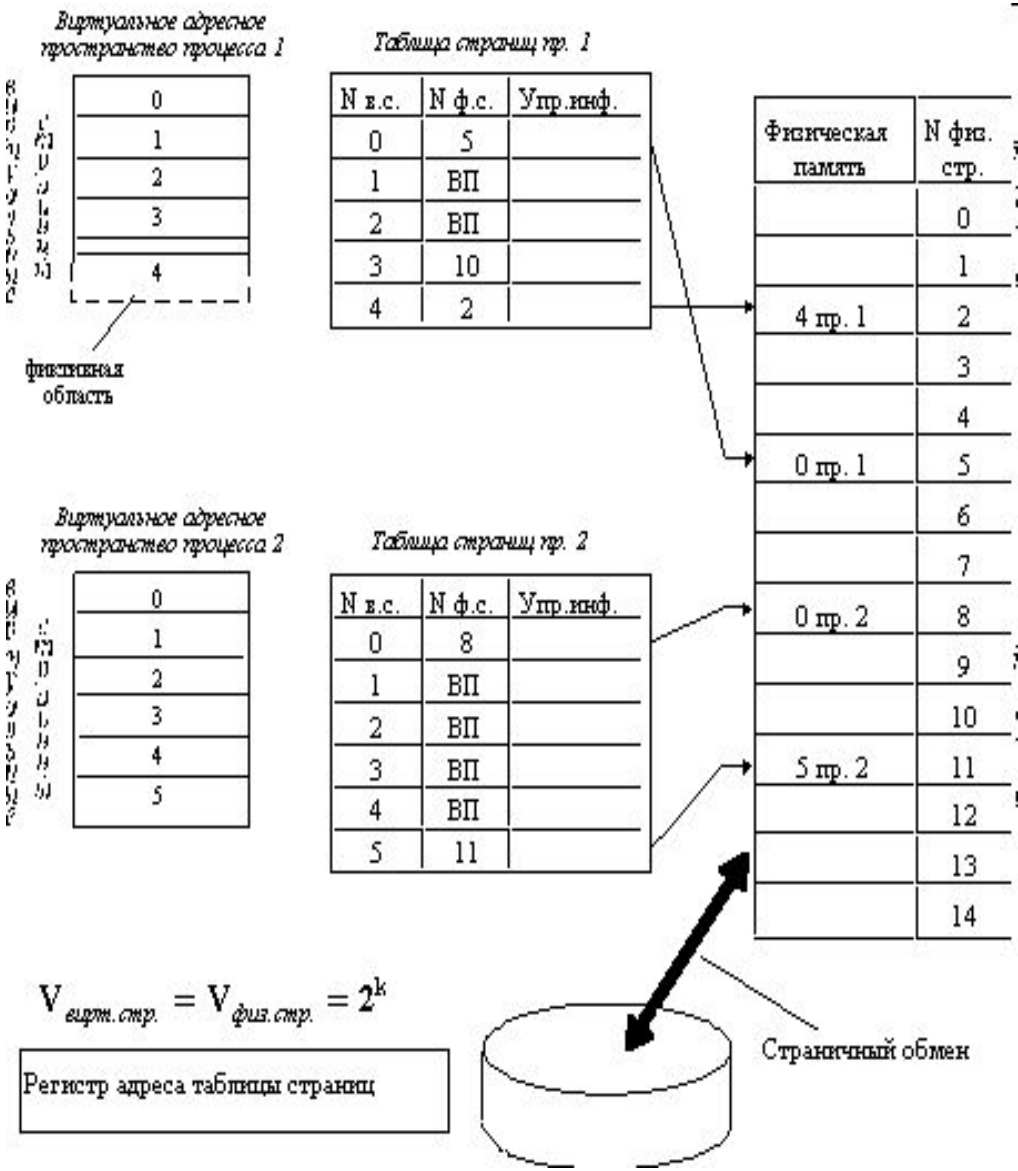
Виртуальная память.







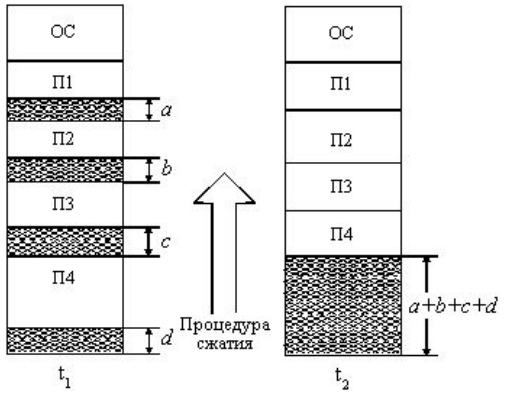
Outline

Виртуальная память:

- виртуальный адрес;
- работа транслятора.

Методы распределения памяти:

- распределение памяти фиксированными разделами;
- распределение памяти разделами переменной величины;
- перемещаемые разделы.



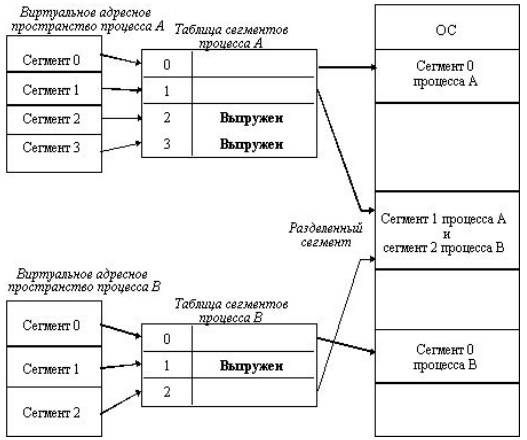
Применение механизма виртуальной памяти:

- базовые понятия:

Страничное распределение;

Сегментное распределение;

Странично-сегментное распределение.



Резюме к лекции и список используемой литературы

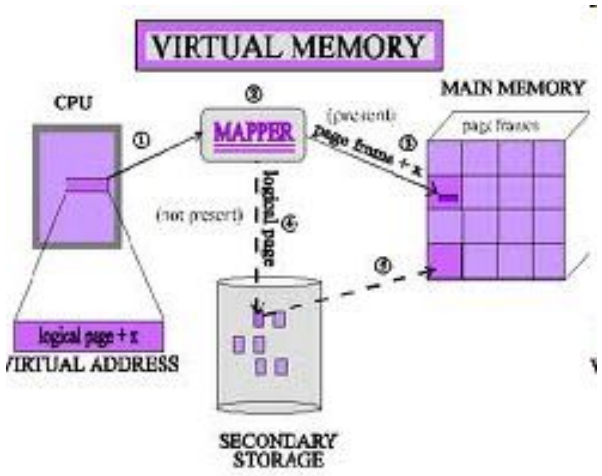
Виртуальный адрес

Виртуальная память— технология управления памятью ЭВМ, разработанная для многозадачных ОС. При использовании данной технологии для каждой программы используются независимые схемы адресации памяти, отображающиеся тем или иным способом на физические адреса в памяти ЭВМ. Позволяет увеличить эффективность использования памяти несколькими одновременно работающими программами, организовав множество независимых адресных пространств, и обеспечить защиту памяти между различными приложениями. Также позволяет программисту использовать больше памяти, чем установлено в компьютере, за счет откачки неиспользуемых страниц на вторичное хранилище

Как появляются виртуальные адреса?

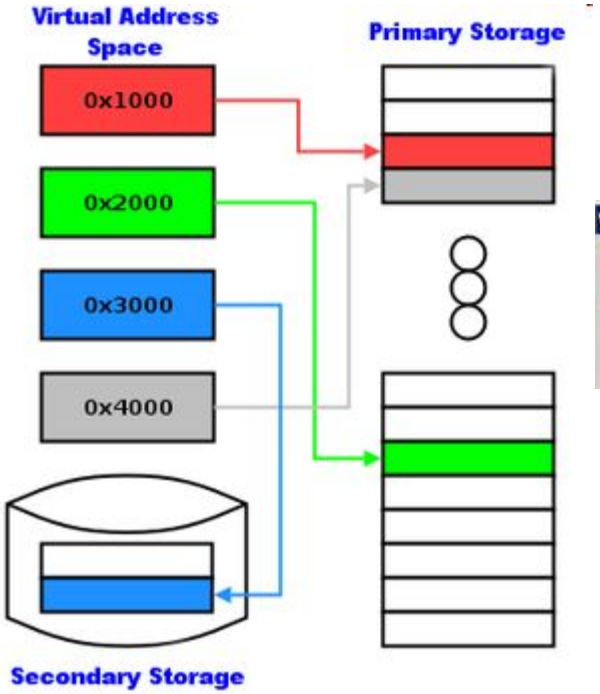
Виртуальные адреса вырабатывает транслятор, переводящий программу на машинный язык. Т.к. во время трансляции в общем случае не известно, в какое место оперативной памяти будет загружена программа, то транслятор присваивает переменным и командам виртуальные (условные) адреса, обычно считая по умолчанию, что программа будет размещена, начиная с нулевого адреса.

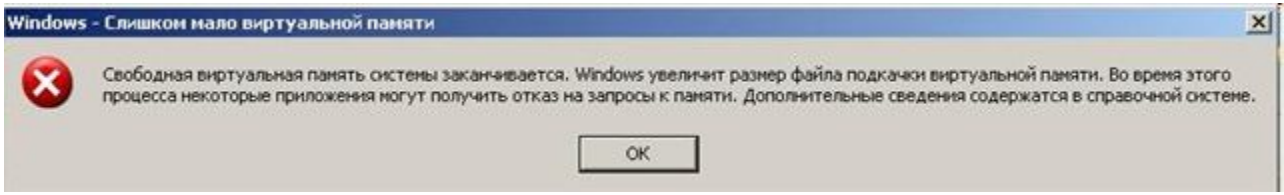
Совокупность виртуальных адресов процесса называется **виртуальным адресным пространством**.



Как появляются виртуальные адреса?

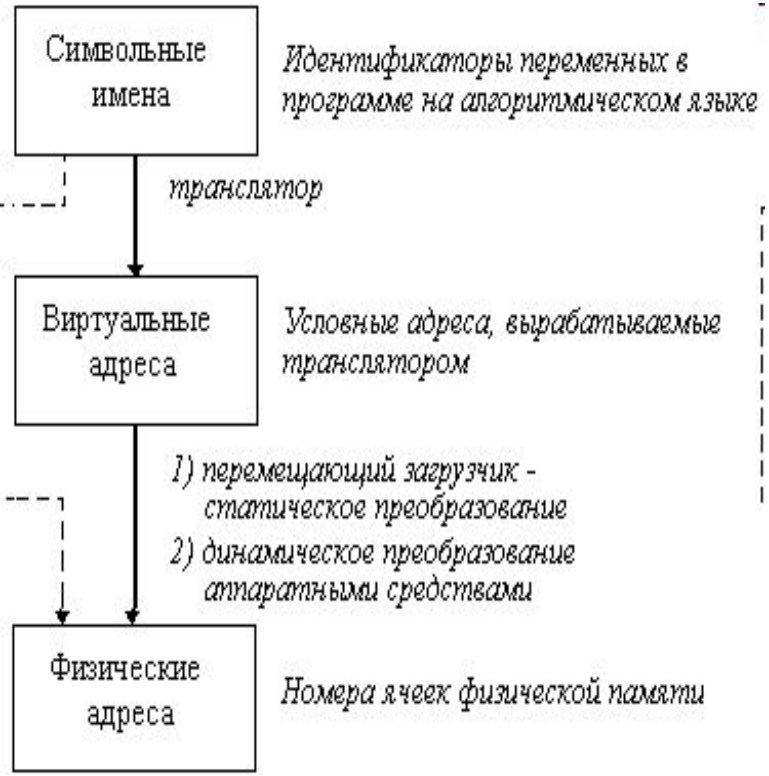
Каждый процесс имеет собственное виртуальное адресное пространство. Максимальный размер виртуального адресного пространства ограничивается разрядностью адреса, присущей данной архитектуре компьютера, и, как правило, не совпадает с объемом физической памяти, имеющимся в компьютере.



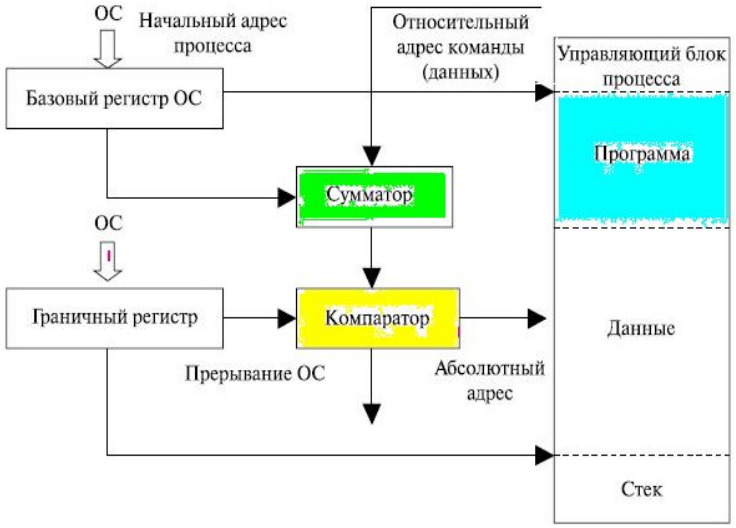


Работа транслятора

Переход от виртуальных адресов к физическим может осуществляться двумя способами.



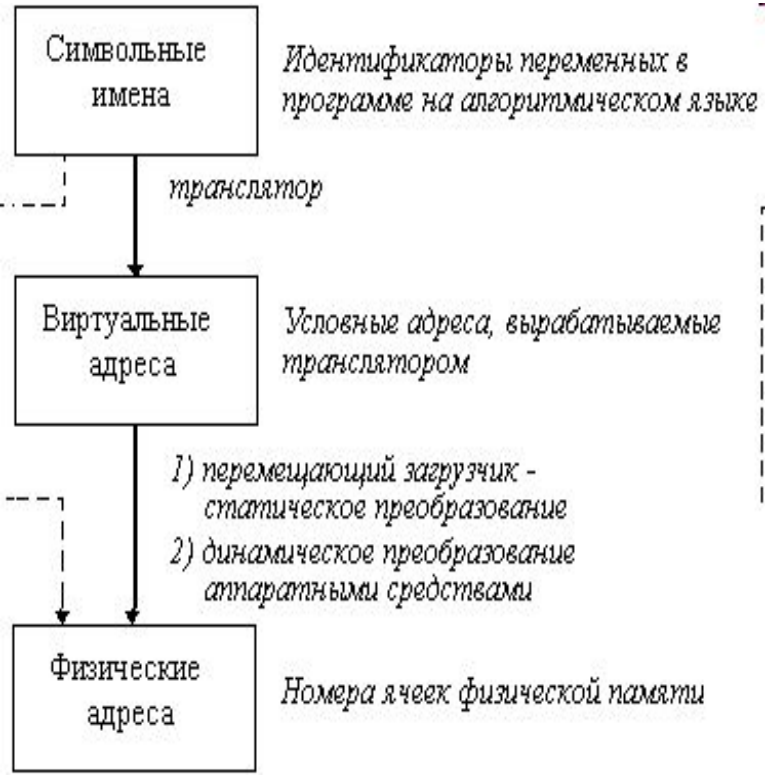
В первом случае замену виртуальных адресов на физические делает специальная системная программа - перемещающий загрузчик.



Перемещающий загрузчик на основании имеющихся у него исходных данных о начальном адресе физической памяти, в которую предстоит загружать программу, и информации, предоставленной транслятором об адресно-зависимых константах программы, выполняет загрузку программы, совмещая ее с заменой виртуальных адресов физическими.

Работа транслятора

Переход от виртуальных адресов к физическим может осуществляться двумя способами.



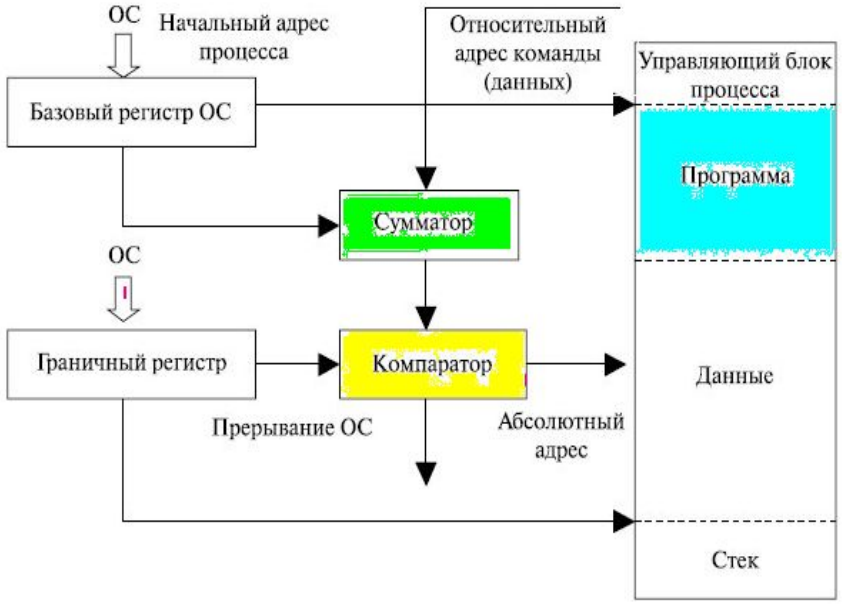
Второй способ заключается в том, что программа загружается в память в неизмененном виде в виртуальных адресах, при этом операционная система фиксирует смещение действительного расположения программного кода относительно виртуального адресного пространства. Во время выполнения программы при каждом обращении к оперативной памяти выполняется преобразование виртуального адреса в физический.

Второй способ или динамическое преобразование является более гибким, он допускает перемещение программы во время ее выполнения, в то время как перемещающий загрузчик жестко привязывает программу к первоначально выделенному ей участку памяти.

Работа транслятора

Трансляция осуществляется следующим образом:

- когда процесс переходит в состояние выполнения, в специальный регистр процесса, называемый базовым, загружается начальный адрес процесса в основной памяти

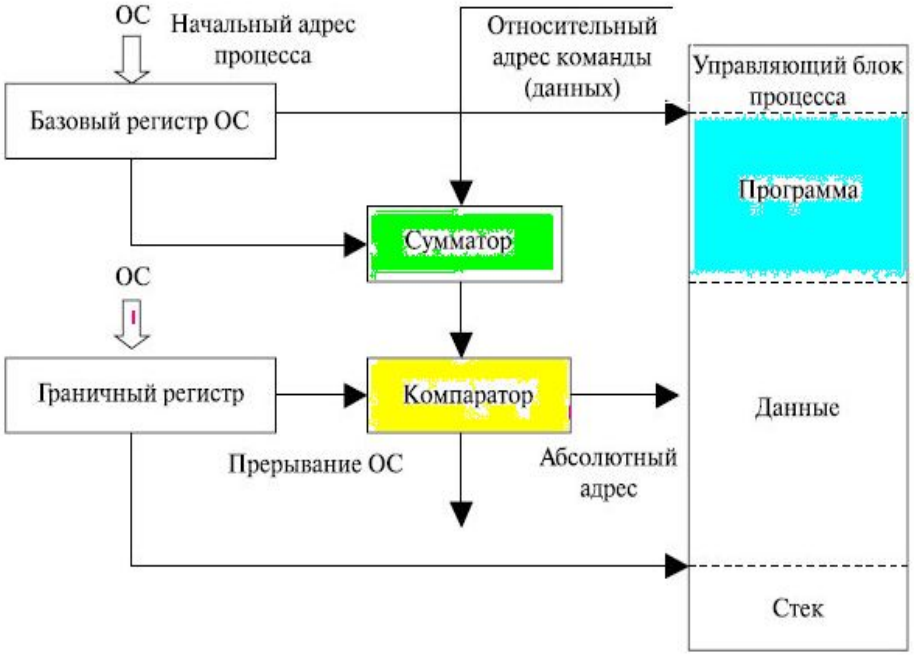


Кроме того, используется "граничный" (bounds) регистр, в котором содержится адрес последней ячейки программы. Эти значения заносятся в регистры при загрузке программы в основную память.

При выполнении процесса относительные адреса в командах обрабатываются процессором в два этапа.

Работа транслятора

- Сначала к относительному адресу прибавляется значение базового регистра для получения абсолютного адреса.



- Затем полученный абсолютный адр сравнивается со значением в граничном регистре.

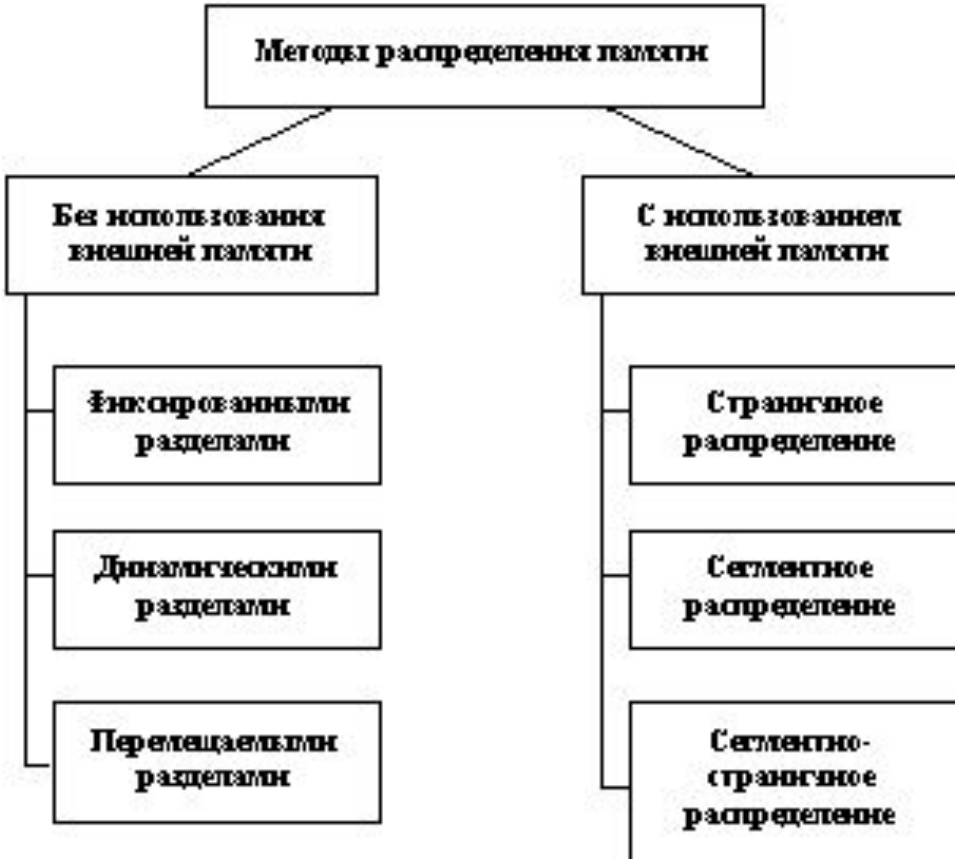
Если полученный абсолютный адрес принадлежит данному процессу, команда может быть выполнена.В противном случае генерируется соответствующее данной ошибке прерывание.

Методы распределения памяти

Все методы управления памятью могут быть разделены на два класса:

- методы, которые используют перемещение процессов между оперативной памятью и диском,
- и методы, которые не делают этого

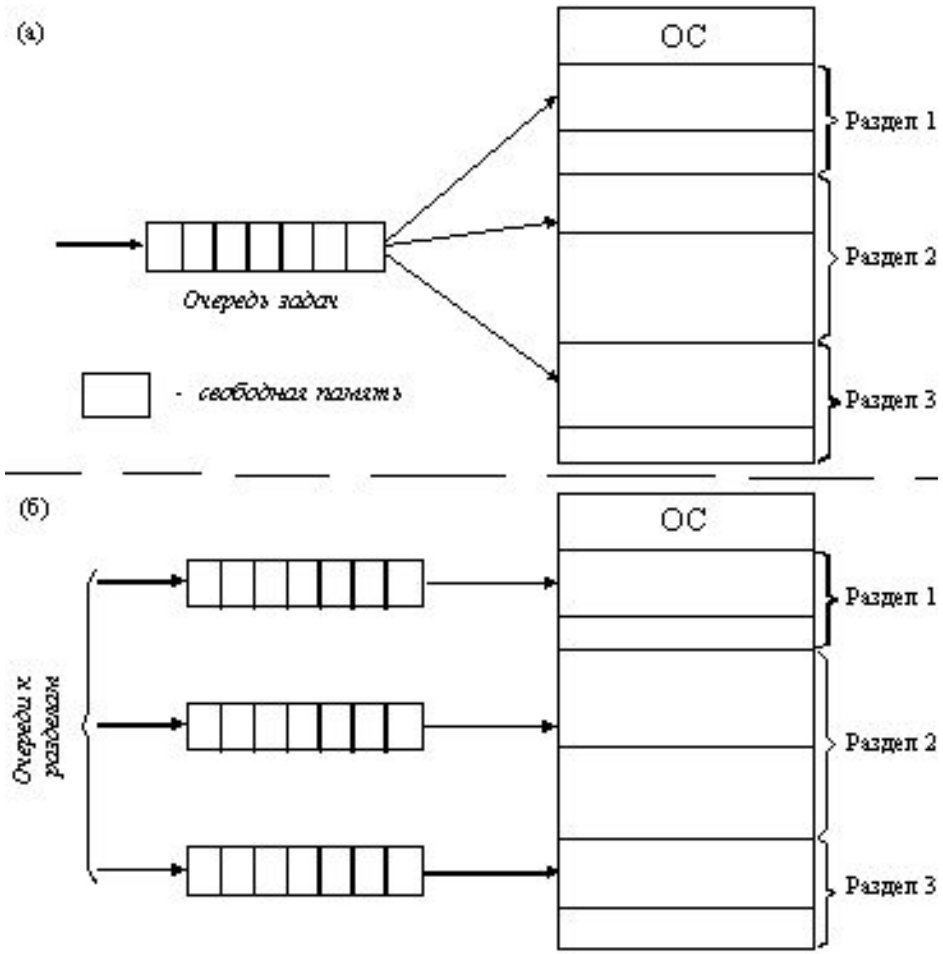
Самым простым способом управления оперативной памятью является разделение ее на несколько разделов фиксированной величины



Распределение памяти фиксированными разделами

Принцип:

Очередная задача, поступившая на выполнение, помещается либо в общую очередь(а), либо в очередь к некоторому разделу (б)



Подсистема управления памятью в этом случае выполняет следующие задачи:

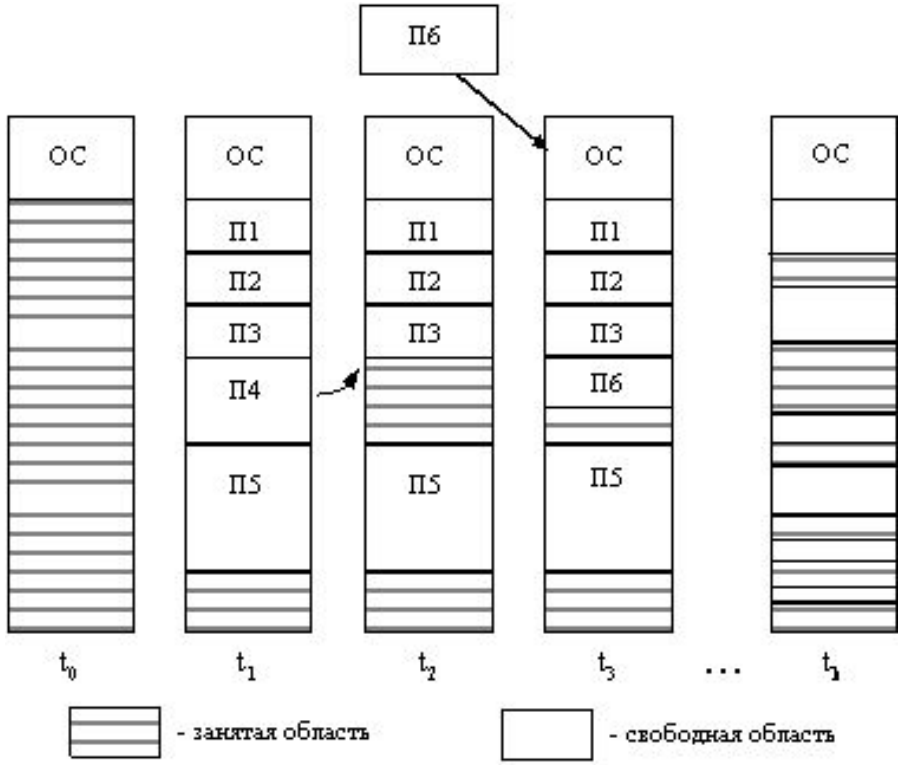
1) сравнивая размер программы, поступившей на выполнение, и свободных разделов, выбирает подходящий раздел,
2) осуществляет загрузку программы и настройку адресов.

При относительно простой реализации - данный метод имеет существенный недостаток - жесткость. Т.к. в каждом разделе может выполняться только одна программа.

Распределение памяти разделами переменной величины

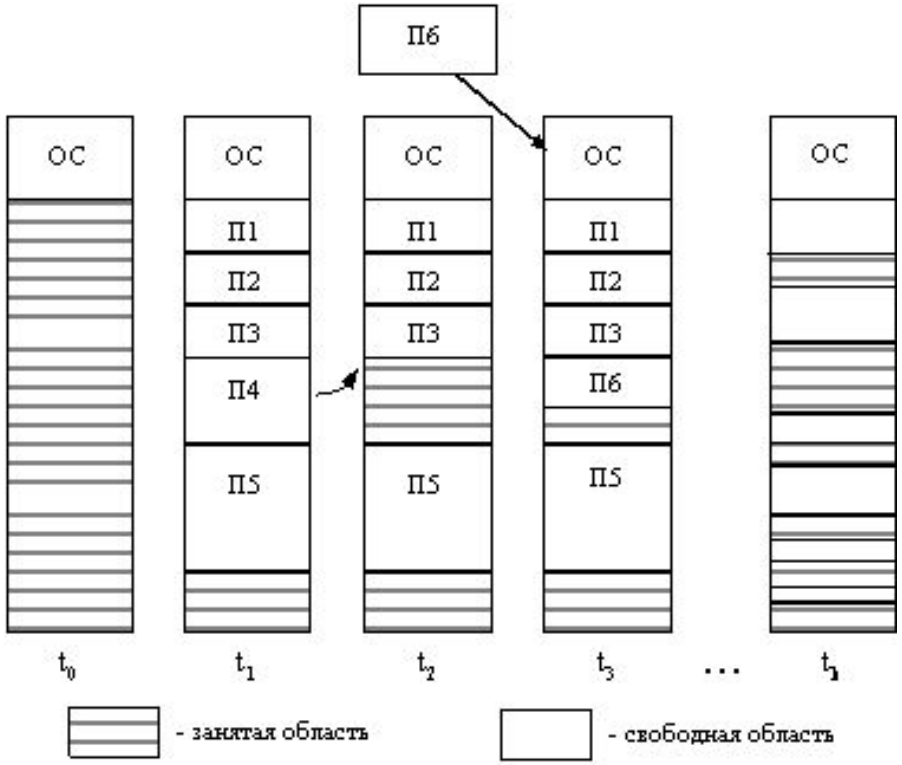
В этом случае память машины не делится заранее на разделы. Сначала вся память свободна. Каждой вновь поступающей задаче выделяется необходимая ей память. Если достаточный объем памяти отсутствует, то задача не принимается на выполнение и стоит в очереди.

После завершения задачи память освобождается, и на это место может быть загружена другая задача. Т.о., в произвольный момент времени оперативная память представляет собой случайную последовательность занятых и свободных участков (разделов) произвольного размера.



Распределение памяти разделами переменной величины

На рисунке показано состояние памяти в различные моменты времени при использовании динамического распределения. Так в момент t0 в памяти находится только ОС, а к моменту t1 память разделена между 5 задачами, причем задача П4, завершаясь, покидает память. На освободившееся после задачи П4 место загружается задача П6, поступившая в момент t3.



Распределение памяти разделами переменной величины

Задачами ОС при реализации данного метода управления памятью является:

- ведение таблиц свободных и занятых областей, в которых указываются начальные адреса и размеры участков памяти,
- при поступлении новой задачи - анализ запроса, просмотр таблицы свободных областей и выбор раздела, размер которого достаточен для размещения поступившей задачи,
- загрузка задачи в выделенный ей раздел и корректировка таблиц свободных и занятых областей,
- после завершения задачи корректировка таблиц свободных и занятых областей.

Программный код не перемещается во время выполнения, т.е. может быть проведена единовременная настройка адресов посредством использования перемещающего загрузчика.

Распределение памяти разделами переменной величины

Выбор раздела для вновь поступившей задачи может осуществляться по разному так, например, как

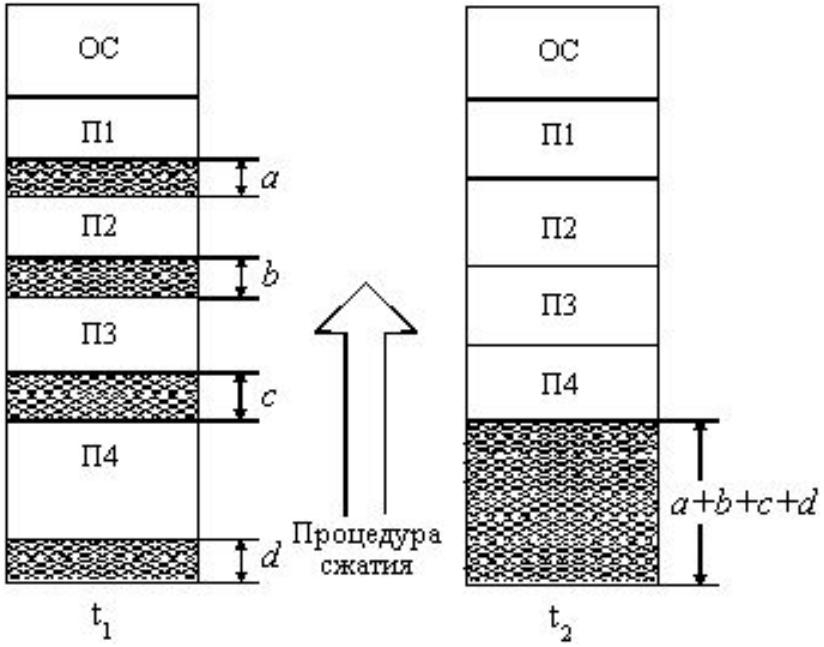
- "первый попавшийся раздел достаточного размера",
- или "раздел, имеющий наименьший достаточный размер",
- или "раздел, имеющий наибольший достаточный размер".

По сравнению с методом распределения памяти фиксированными разделами данный метод обладает гораздо большей гибкостью, но ему присущ очень серьезный недостаток - фрагментация памяти.

Фрагментация - это наличие большого числа несмежных участков свободной памяти очень маленького размера (фрагментов). Настолько маленького, что ни одна из вновь поступающих программ не может поместиться ни в одном из участков, хотя суммарный объем фрагментов может составить значительную величину, намного превышающую требуемый объем памяти.

Перемещаемые разделы

Одним из методов борьбы с фрагментацией является перемещение всех занятых участков в сторону старших либо в сторону младших адресов, так, чтобы вся свободная память образовывала единую свободную область

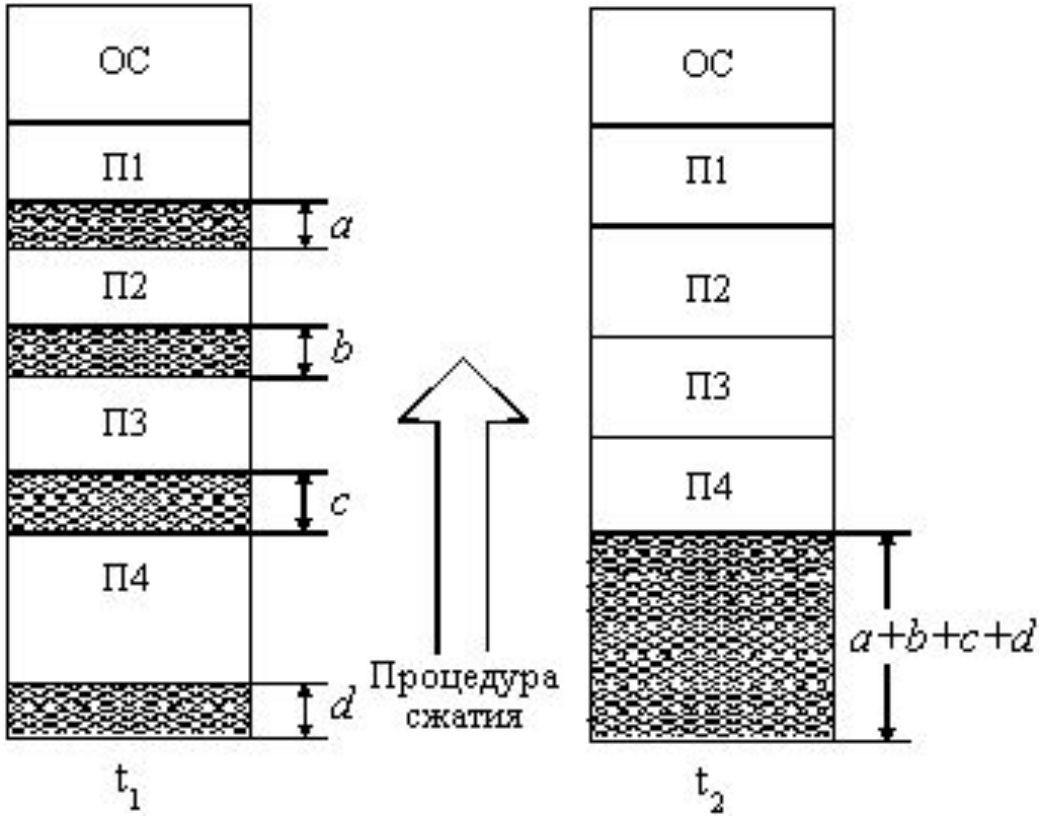


В дополнение к функциям, которые выполняет ОС при распределении памяти переменными разделами, в данном случае она должна еще время от времени копировать содержимое разделов из одного места памяти в другое, корректируя таблицы свободных и занятых областей. Эта процедура называется "сжатием".

Сжатие может выполняться либо при каждом завершении задачи, либо только тогда, когда для вновь поступившей задачи нет свободного раздела достаточного размера. В первом случае требуется меньше вычислительной работы при корректировке таблиц, а во втором - реже выполняется процедура сжатия.

Перемещаемые разделы

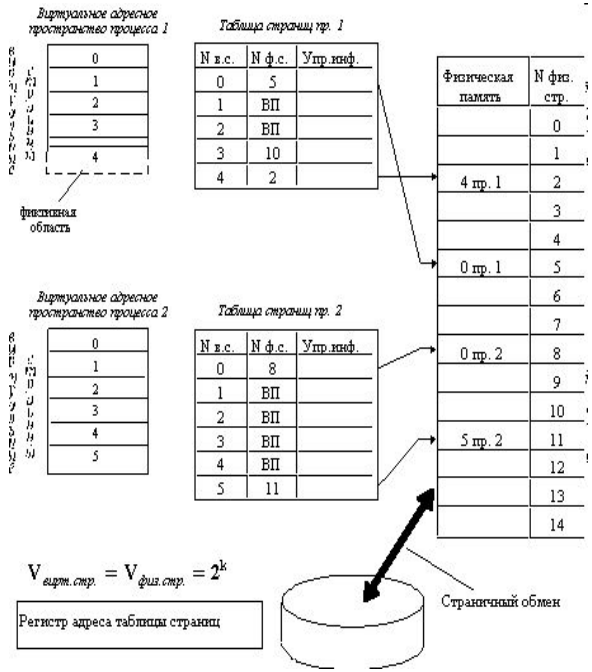
Т.к. программы перемещаются по оперативной памяти в ходе своего выполнения, то преобразование адресов из виртуальной формы в физическую должно выполняться динамическим способом.

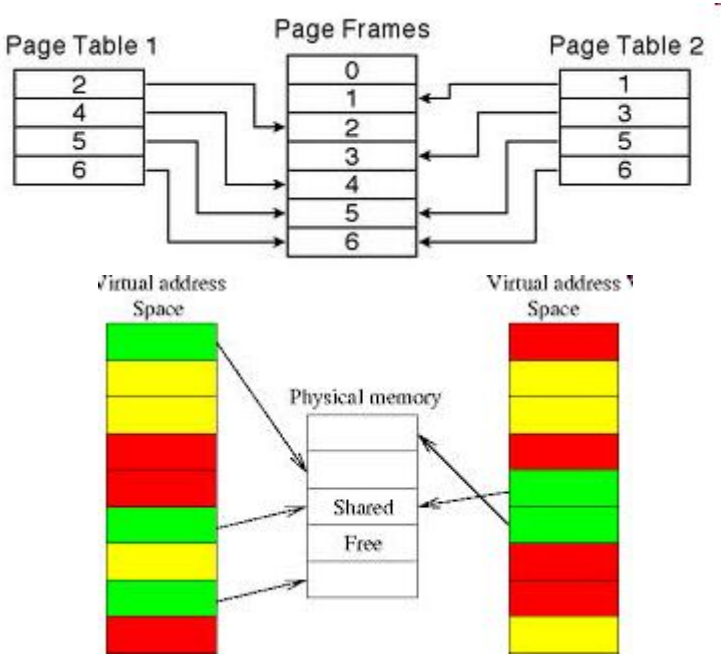


Хотя процедура сжатия и приводит к более эффективному использованию памяти, она может потребовать значительного времени, что часто перевешивает преимущества данного метода.

Применение механизма виртуальной памяти

При загрузке процесса часть его виртуальных страниц помещается в RAM, а остальные - на HDD.

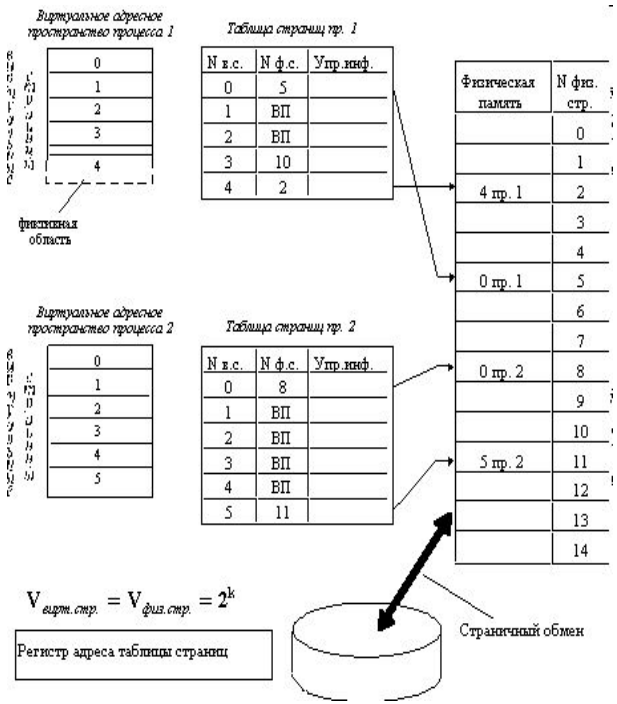


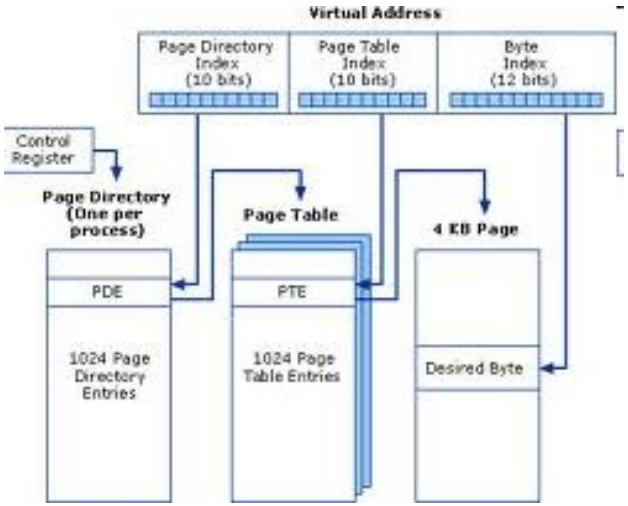


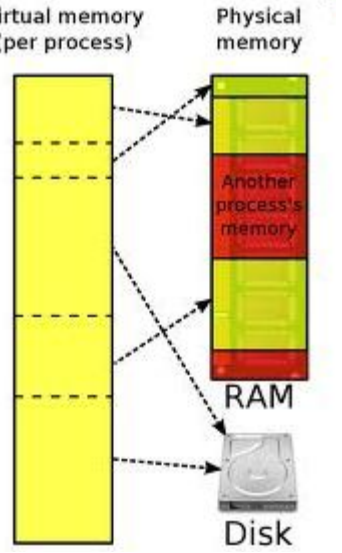
Смежные виртуальные страницы не обязательно располагаются в смежных физических страницах. При загрузке ОС создает для каждого процесса информационную структуру - таблицу страниц, в которой устанавливается соответствие между номерами виртуальных и физических страниц для страниц, загруженных в RAM, или делается отметка о том, что виртуальная страница выгружена на диск.

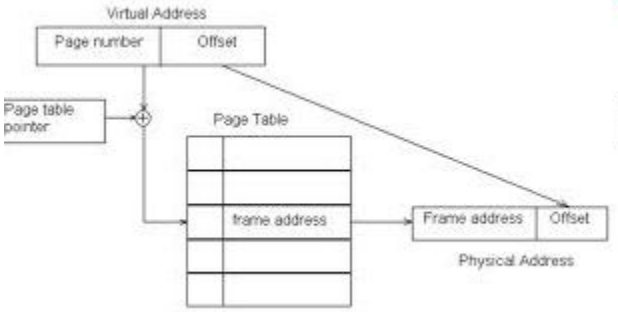
Применение механизма виртуальной памяти

Кроме того, в таблице страниц содержится управляющая информация, такая как признак модификации страницы, признак невыгружаемости (выгрузка некоторых страниц может быть запрещена), признак обращения к странице (используется для подсчета числа обращений за определенный период времени) и другие данные, формируемые и используемые механизмом виртуальной памяти.



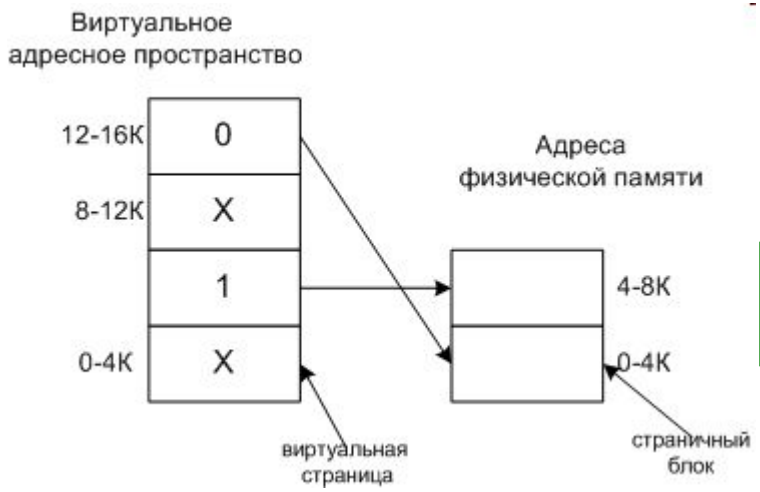






Страницы – основа виртуальной памяти

Страницы - это части, на которые разбивается пространство виртуальных адресов.
Страничные блоки - единицы физической памяти. Страницы всегда имеют фиксированный размер.
Передача данных между RAM и HDD всегда происходит в страницах.



X - обозначает не отображаемую страницу в физической памяти.

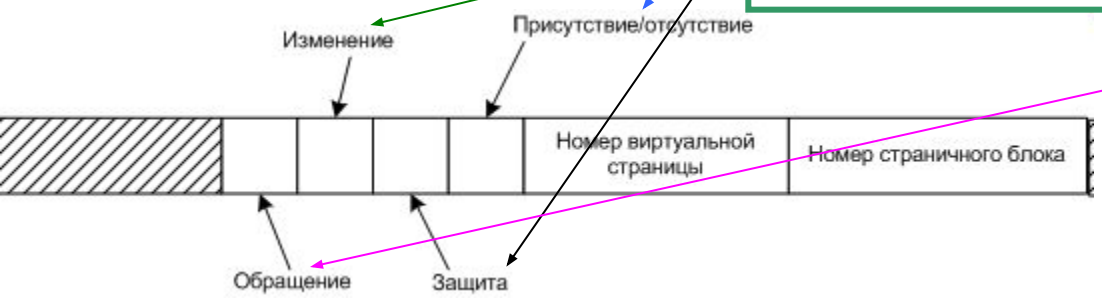
Страничное прерывание - происходит, если процесс обратился к странице, которая не загружена в RAM (т.е. X).
Процессор передается другому процессу, и параллельно страница загружается в память.

Таблица страниц - используется для хранения соответствия адресов виртуальной страницы и страничного блока.

Таблица может быть размещена:

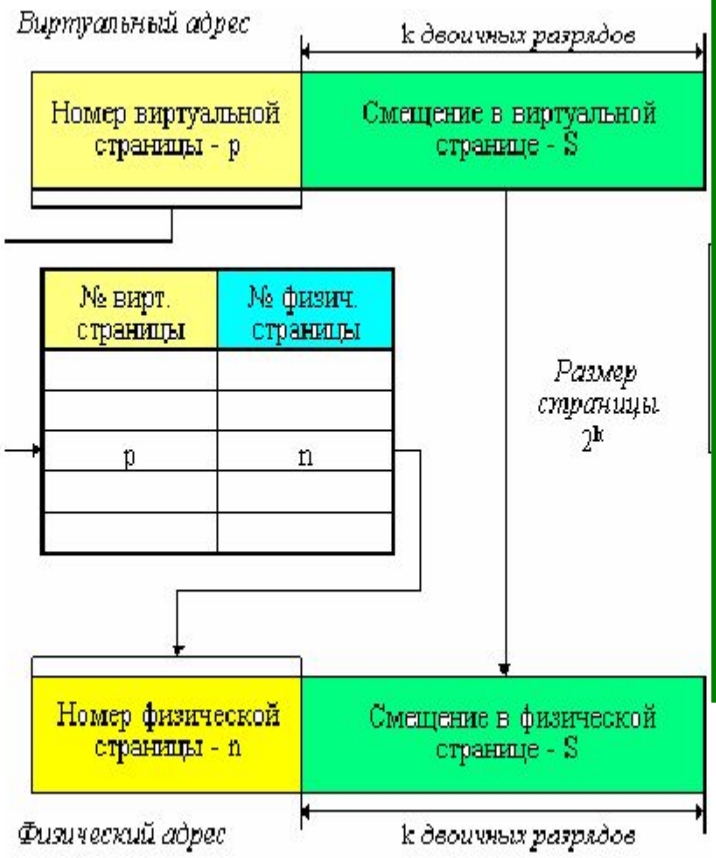
- в аппаратных регистрах (“+”: более высокое быстродействие)
- в RAM

Типичная запись в таблице страниц



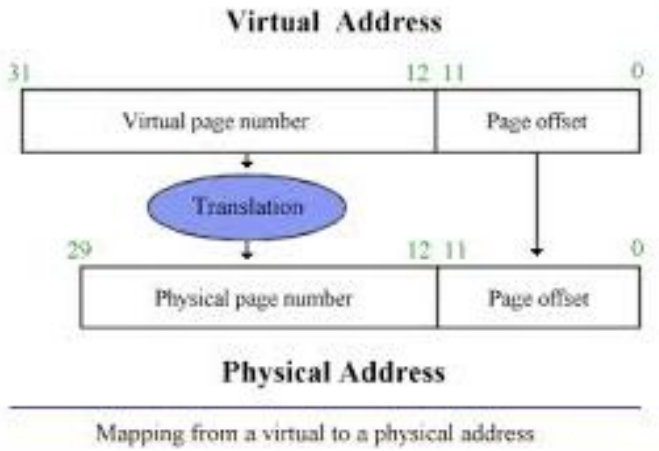
- Присутствие/отсутствие - загружена или незагружена в память
- Защита - виды доступа, например, чтение/запись.
- Изменение - изменилась ли страница, если да то при выгрузке записывается на диск, если нет, просто уничтожается.
- Обращение - было ли обращение к странице, если нет, то это лучший кандидат на освобождение памяти.
- Информация о адресе страницы когда она хранится на диске, в таблице не размещается.

Страничное распределение

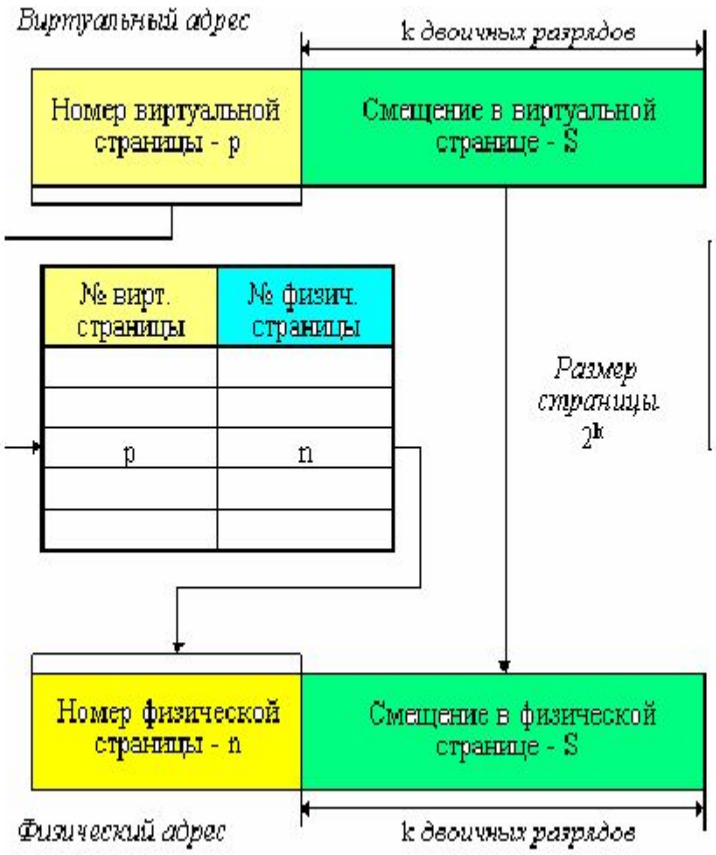


Виртуальный адрес при страничном распределении может быть представлен в виде пары (p, s),
где p - номер виртуальной страницы процесса (нумерация страниц начинается с 0),
а s - смещение в пределах виртуальной страницы.
Учитывая, что размер страницы равен 2 в степени k, смещение s может быть получено простым отделением k младших разрядов в двоичной записи виртуального адреса.

Оставшиеся старшие разряды представляют собой двоичную запись номера страницы p.



Применение механизма виртуальной памяти



При каждом обращении к оперативной памяти аппаратными средствами выполняются следующие действия:

1. на основании начального адреса таблицы страниц (содержимое регистра адреса таблицы страниц), номера виртуальной страницы (старшие разряды виртуального адреса) и длины записи в таблице страниц (системная константа) определяется адрес нужной записи в таблице,
2. из этой записи извлекается номер физической страницы,
3. к номеру физической страницы присоединяется смещение (младшие разряды виртуального адреса).

Использование в пункте (3) того факта, что размер страницы равен степени 2, позволяет применить операцию конкатенации (присоединения) вместо более длительной операции сложения, что уменьшает время получения физического адреса, а значит повышает производительность компьютера.

Странично-сегментное распределение

Виртуальное пространство процесса делится на сегменты,
а каждый сегмент, в свою очередь, делится на виртуальные страницы,
которые нумеруются в пределах сегмента.

VIRTUAL MEMORY

segments

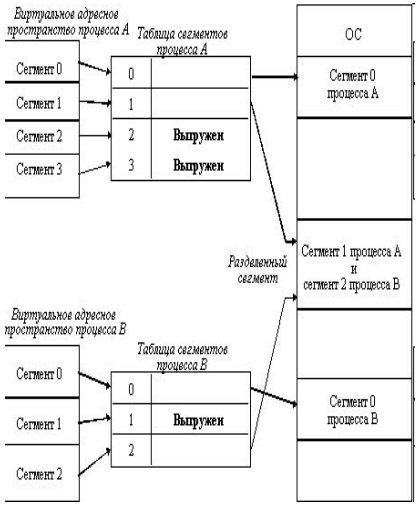
segments -> pages

RAM делится на физические страницы.
Загрузка процесса выполняется ОС постранично, при этом часть страниц размещается в RAM, а часть на HDD.

Для каждого сегмента создается своя таблица страниц, структура которой полностью совпадает со структурой таблицы страниц, используемой при страничном распределении.

Для каждого процесса создается таблица сегментов, в которой указываются адреса таблиц страниц для всех сегментов данного процесса. Адрес таблицы сегментов загружается в специальный регистр процессора, когда активизируется соответствующий процесс.

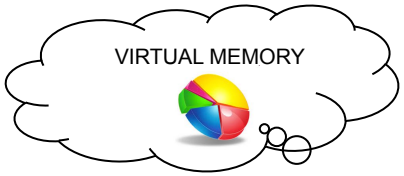
Сегментное распределение



Виртуальное адресное пространство процесса делится на сегменты, размер которых определяется программистом с учетом смыслового значения содержащейся в них информации.

Отдельный сегмент может представлять собой подпрограмму, массив данных и т.п.

Иногда сегментация программы выполняется по умолчанию компилятором.



При загрузке процесса часть сегментов помещается в RAM (при этом для каждого из этих сегментов ОС подыскивает подходящий участок свободной памяти), а часть сегментов размещается в HDD.

Сегменты одной программы могут занимать в RAM несмежные участки.

Во время загрузки система создает таблицу сегментов процесса (аналогичную таблице страниц), в которой для каждого сегмента указывается:

- начальный физический адрес сегмента в RAM,
- размер сегмента,
- правила доступа,
- признак модификации,

Если виртуальные адресные пространства нескольких процессов включают один и тот же сегмент, то в таблицах сегментов этих процессов делаются ссылки на один и тот же участок RAM, в который данный сегмент загружается в единственном экземпляре.

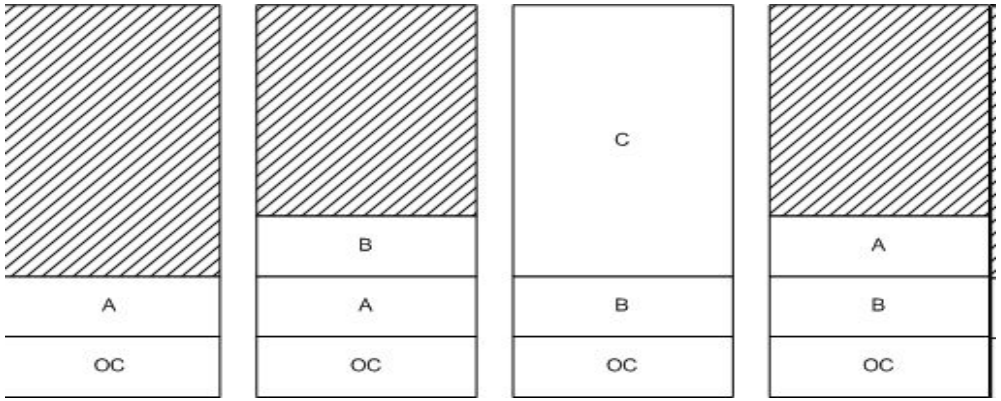
- признак обращения к данному сегменту за последний интервал времени и некоторая другая инфа.

Свопинг и виртуальная память

Так как памяти, как правило, не хватает. Для выполнения процессов часто приходится использовать диск.

Основные способы использования диска:

- Свопинг (подкачка) - процесс целиком загружается в память для работы
- Виртуальная память - процесс может быть частично загружен в память для работы



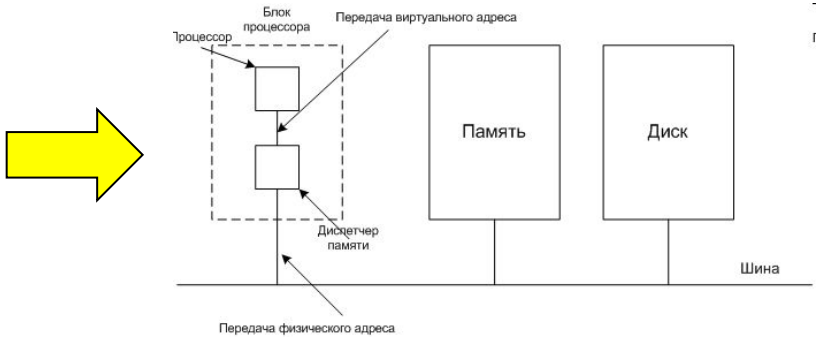
т.к. процесс С очень большой, процесс А был выгружен временно на диск, после завершения процесса С он снова был загружен в память.

Как мы видим процесс А второй раз загрузился в другое адресное пространство, должны создаваться такие условия, которые не повлияют на работу процесса.

Свопер - планировщик, управляющий перемещением данных между памятью и диском.

Основная идея виртуальной памяти

заключается в разбиении программы на части, и в память эти части загружаются по очереди. Программа при этом общается с *виртуальной памятью*, а не с физической.



Статическая и динамическая области свопинга

Статическая область свопинга

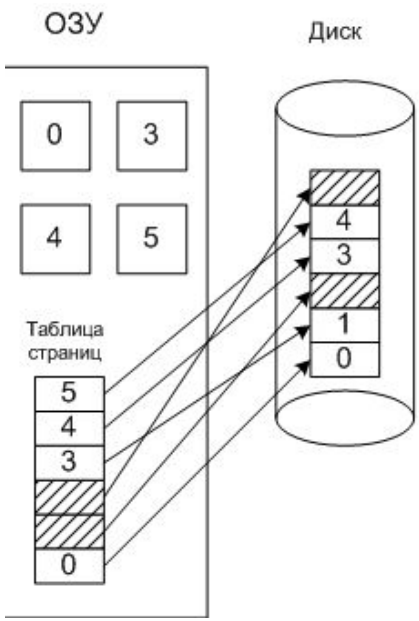
После запуска процесса он занимает определенную память, на диске сразу ему выделяется такое же пространство.
Поэтому файл подкачки должен быть *не меньше* памяти. А в случае нехватки памяти даже больше.
Как только процесс завершится, он освободит память и место на диске.

На диске всегда есть дубликат страницы, которая находится в памяти.

Динамическая область свопинга

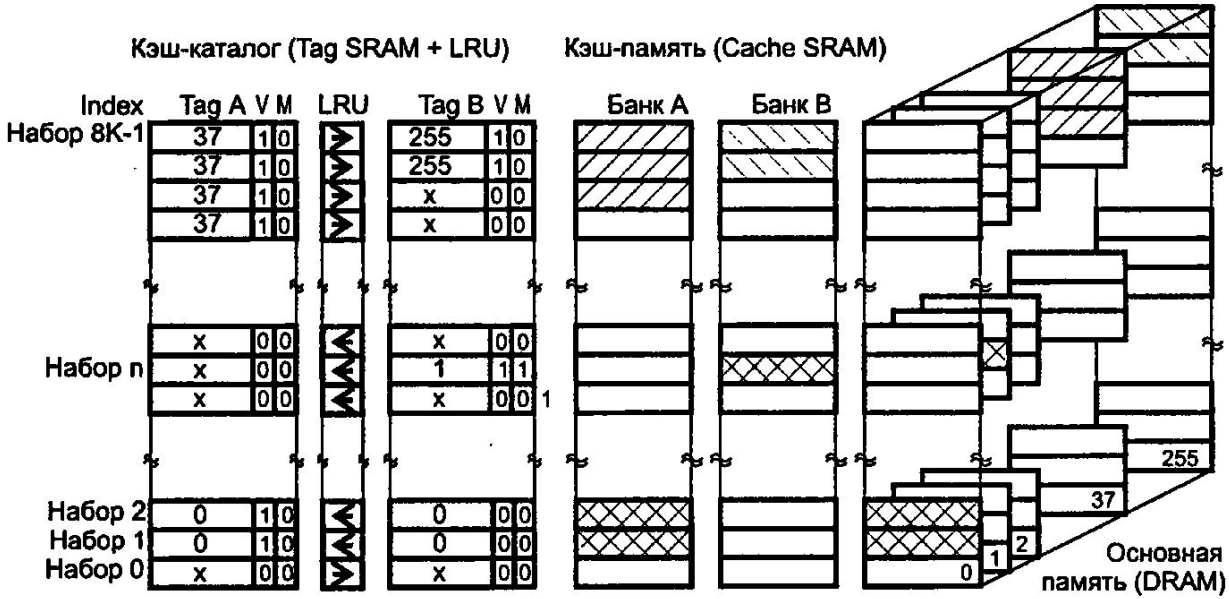
Предполагается не выделять страницам место на диске, а выделять только при выгрузке страницы, и как только страница вернется в память освобождать место на диске.

Этот механизм сложнее, так как процессы не привязаны к какому-то пространству на диске, и нужно хранить информацию (карту диска) о местоположении на диске каждой страницы.

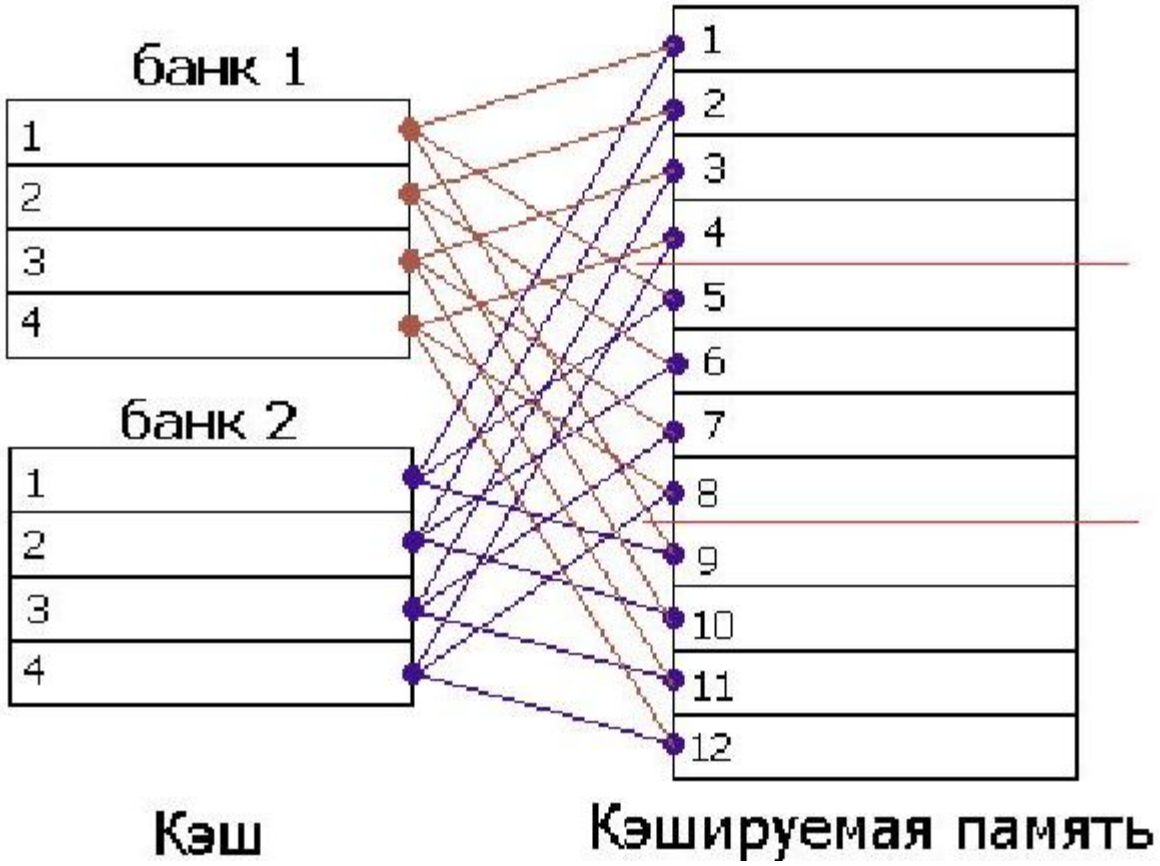


В следующей серии

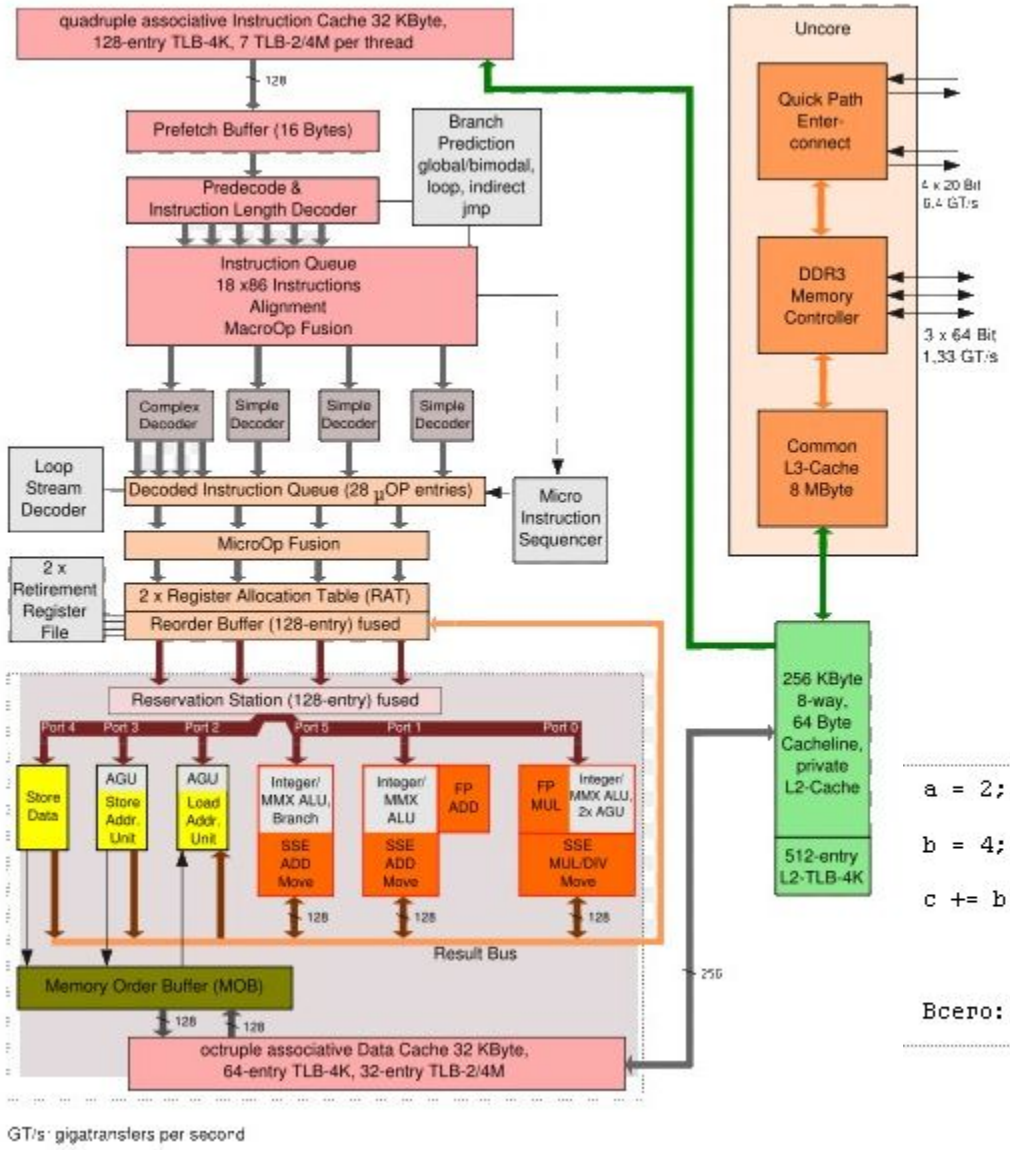
Cache



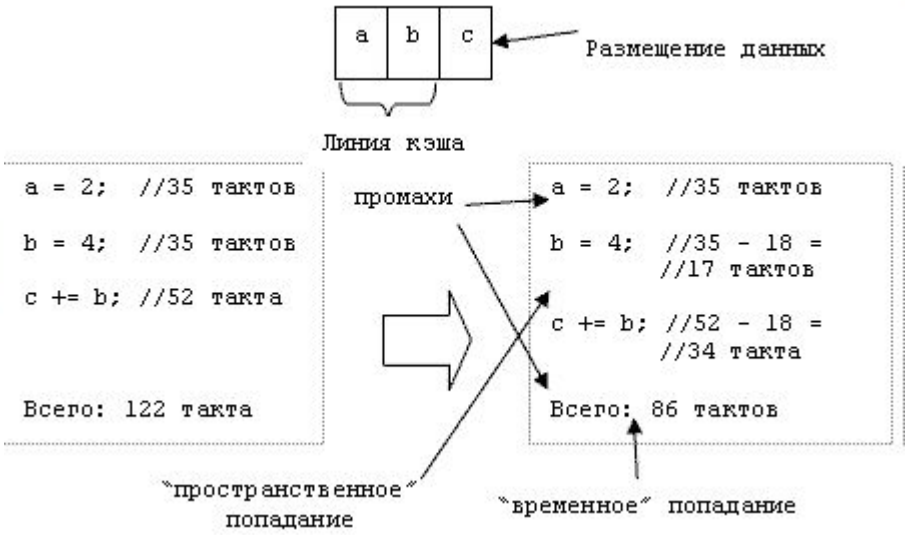
http://www.ixbt.com/cpu/intel-ci7-theory.shtml
http://citforum.ru/programming/digest/gaisarian/
http://docstore.mik.ua/svk/glava_9.htm
http://www.ixbt.com/news/all/index.shtml?06/21/44
http://alasir.com/articles/cache_principles/cache_associativity_rus.html
http://education.aspu.ru/view.php?olif=gl5
http://www.x-drivers.ru/articles/technologies/20/full.html
http://www.intuit.ru/department/hardware/microarch/4/3.html
http://www.intuit.ru/department/hardware/microarch/4/2.html
http://www.yudenisov.narod.ru/EIS/Vol07/b4.htm
http://iproc.ru/parallel-programming/lection-7/
http://www.3dnews.ru/cpu/pentium4-vs-g4e/index5.htm
http://www.intuit.ru/department/hardware/mcoreproc/5/
http://www.kit-e.ru/articles/dsp/2008_3_32.php
http://www.intuit.ru/department/os/modernos/6/4.html

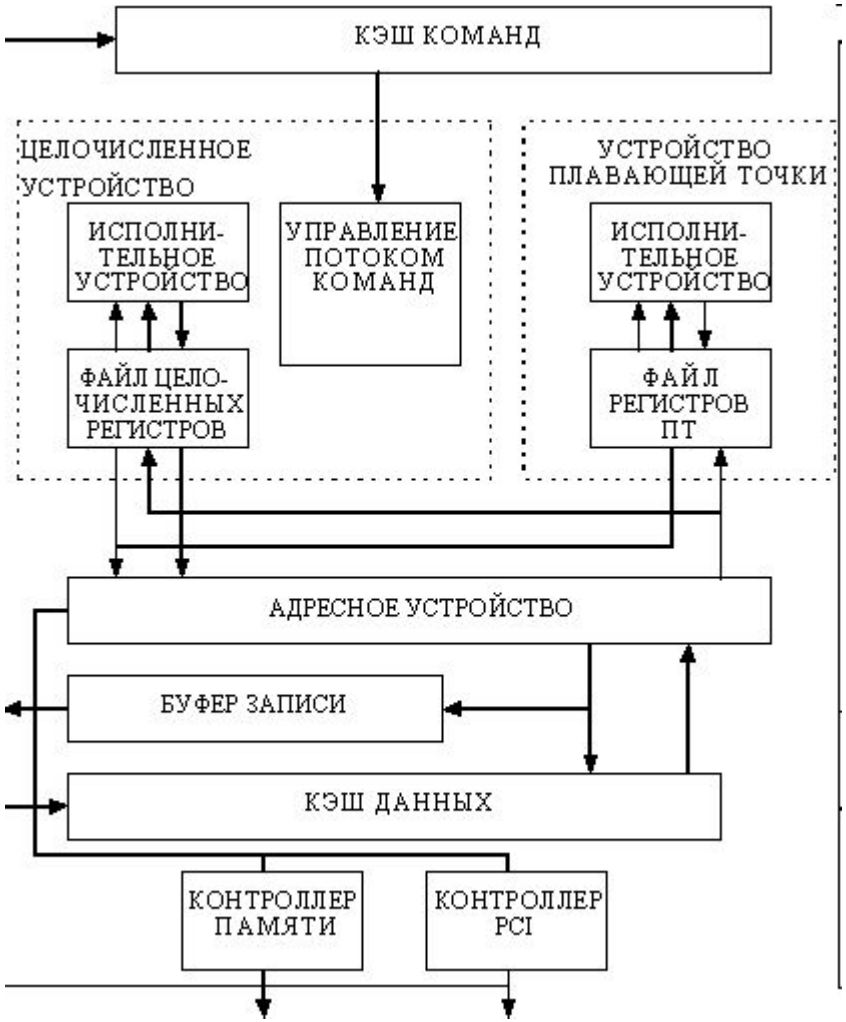

банк 1
1
2
3
4
банк 2
1
2
3
4
1
2
3
4
5
6
7
8
9
10
11
12
Кэш
Кэшируемая память

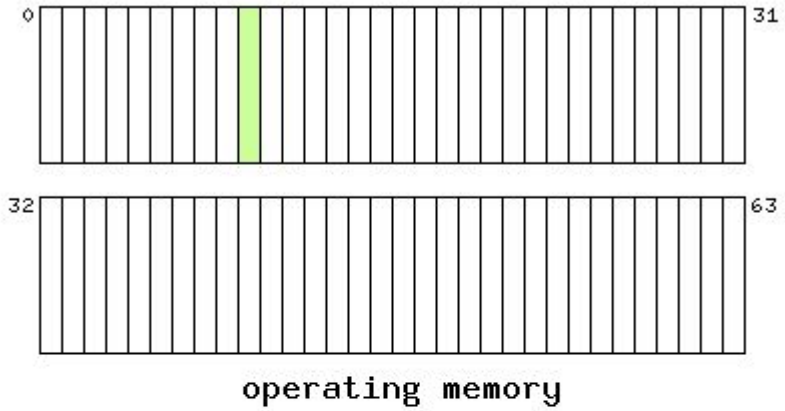
Intel Nehalem microarchitecture

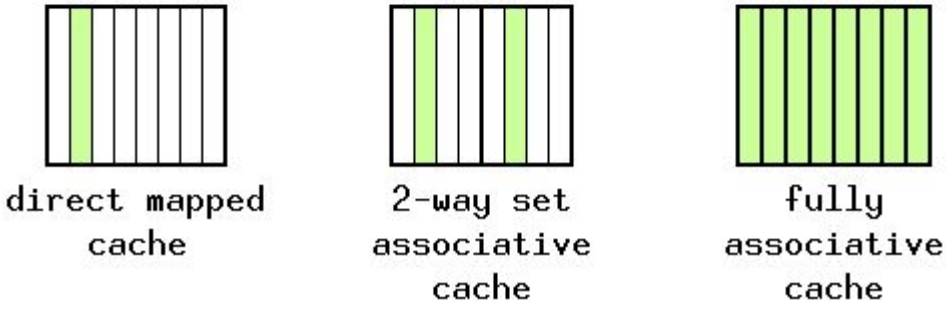


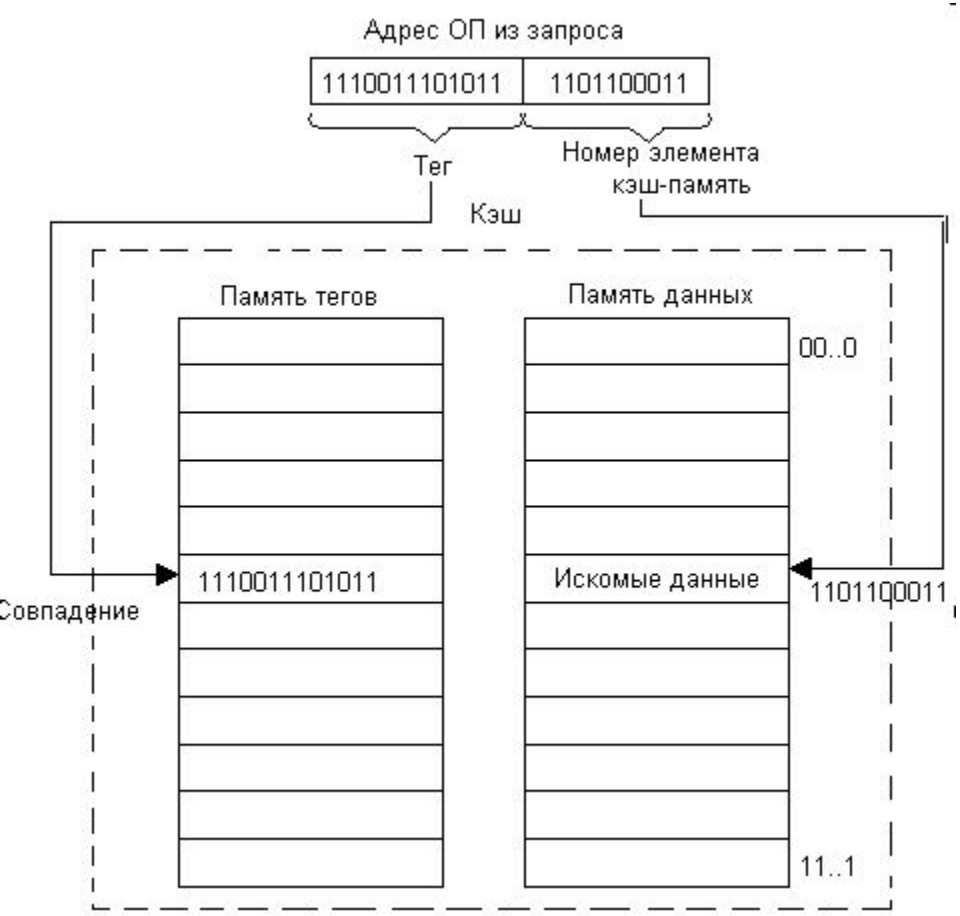
GT/s: gigatransfers per second

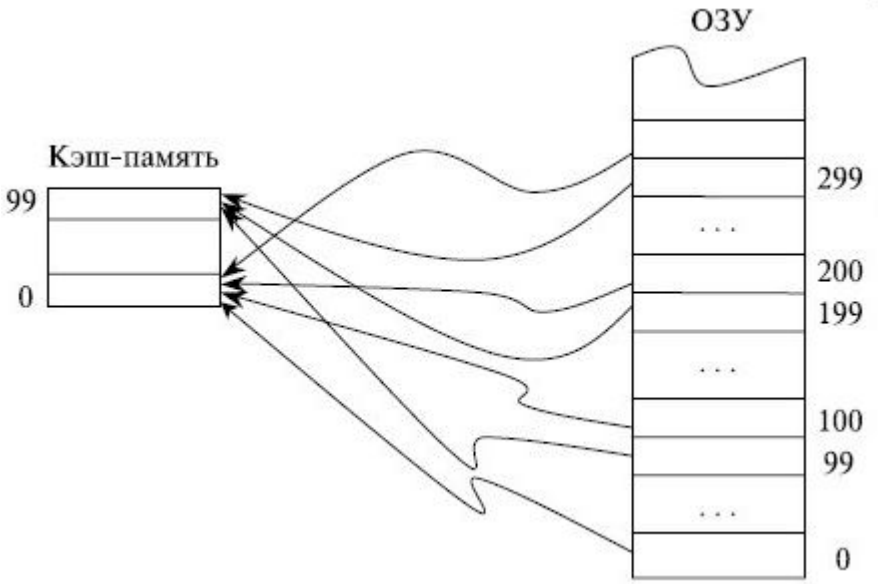


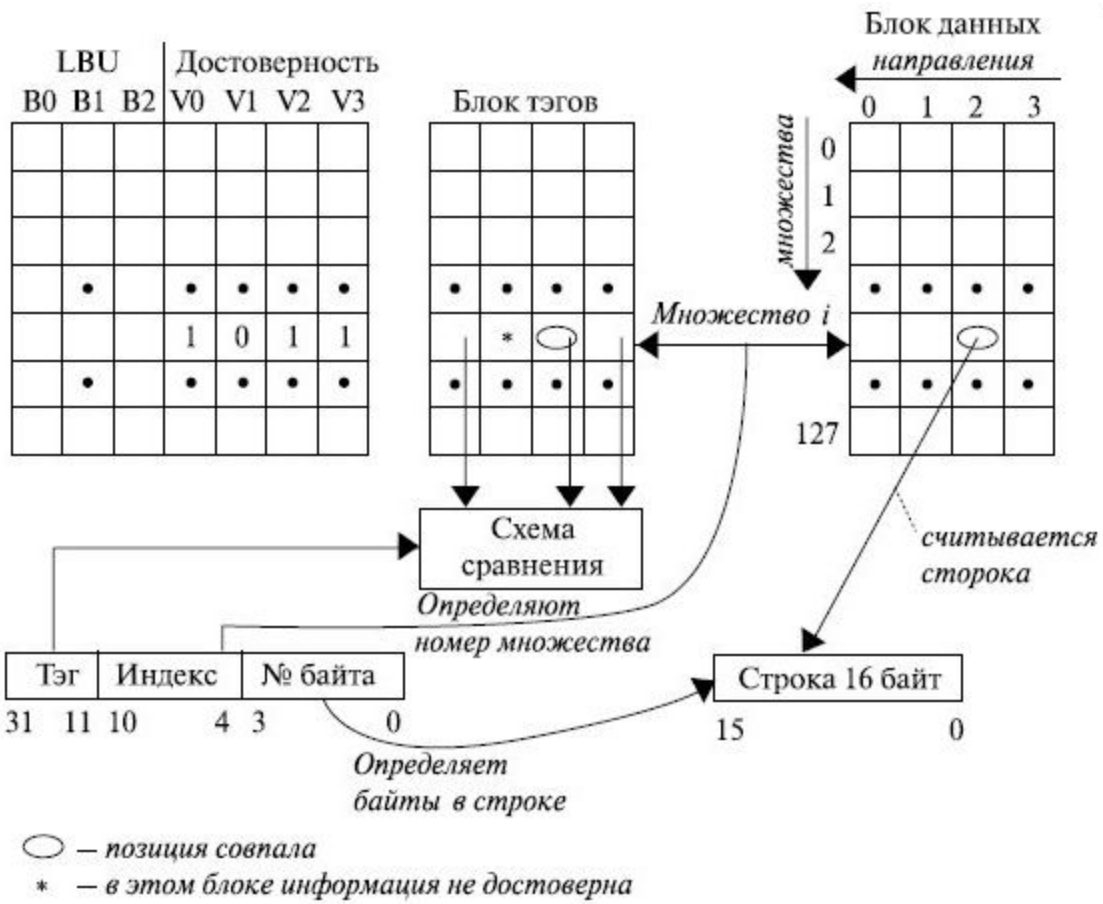

КЭШ КОМАНД
ЦЕЛОЧИСЛЕННОЕ УСТРОЙСТВО
ИСПОЛНИ-ТЕЛЬНОЕ УСТРОЙСТВО
ФАЙЛ ЦЕЛО-ЧИСЛЕННЫХ РЕГИСТРОВ
УПРАВЛЕНИЕ ПОТОКОМ КОМАНД
УСТРОЙСТВО ПЛАВАЮЩЕЙ ТОЧКИ
ИСПОЛНИ-ТЕЛЬНОЕ УСТРОЙСТВО
ФАЙЛ РЕГИСТРОВ ПТ
АДРЕСНОЕ УСТРОЙСТВО
БУФЕР ЗАПИСИ
КЭШ ДАННЫХ
КОНТРОЛЛЕР ПАМЯТИ
КОНТРОЛЛЕР PCI

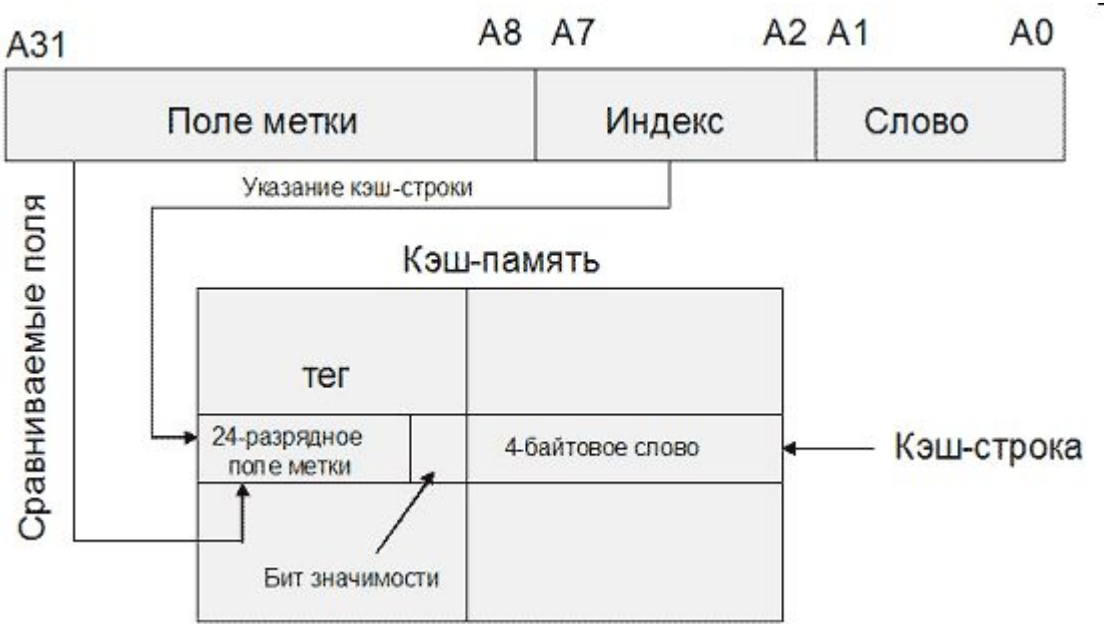

0
31
32
63
operating memory

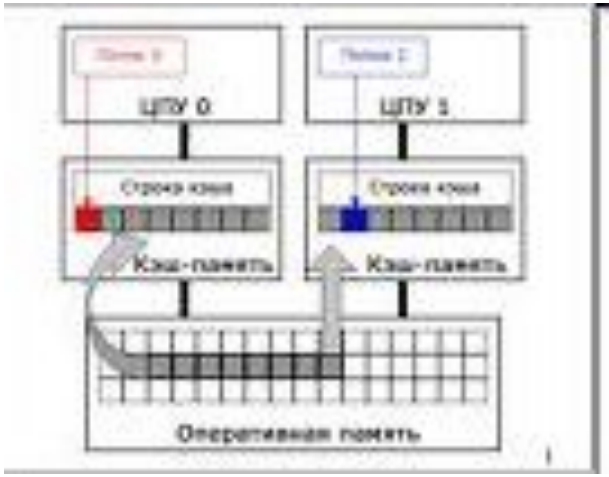

direct mapped cache
2-way set associative cache
fully associative cache

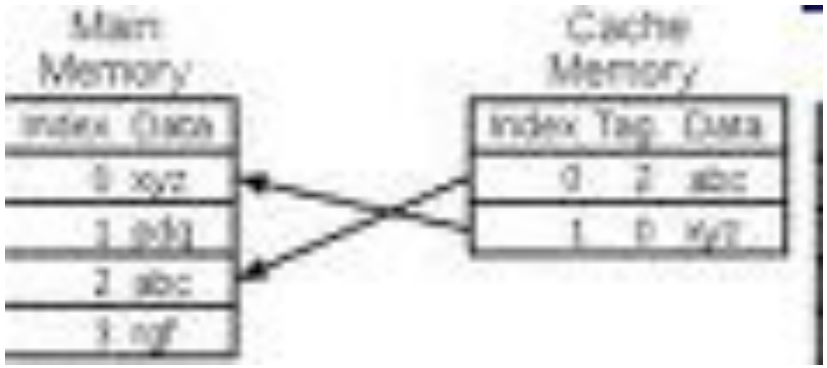

Адрес ОП из запроса
1110011101011
1101100011
Тег
Номер элемента кэш-память
Кэш
Память тегов
Память данных
00..0
1110011101011
Искомые данные
1101100011
Совпадение
11..1

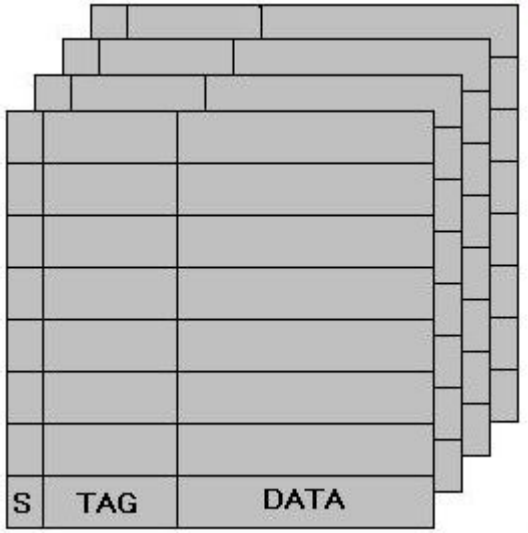

ОЗУ
Кэш-память
99
0
299
200
199
100
99
0

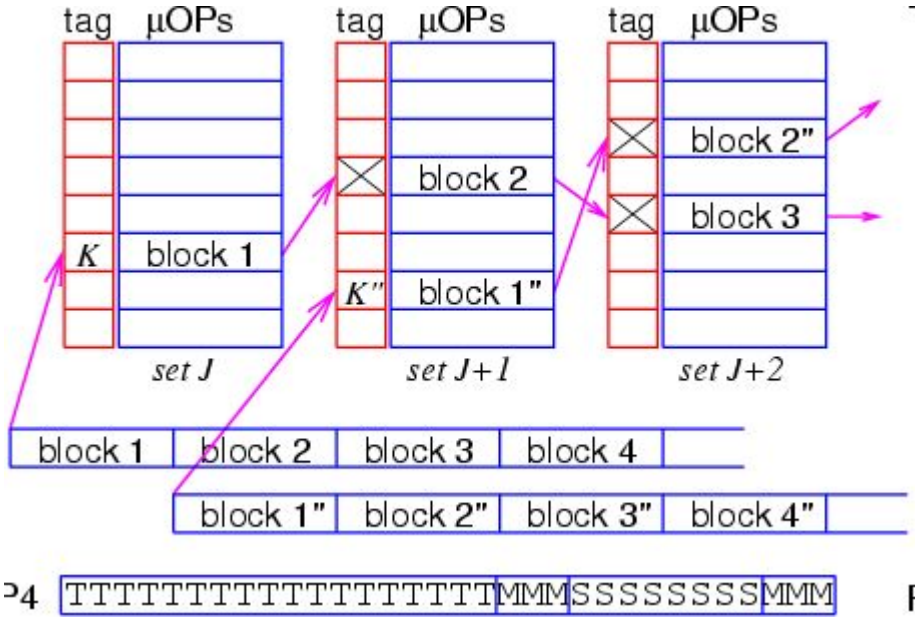

LBU
B0 B1 B2
Достоверность
V0 V1 V2 V3
Блок тэгов
Блок данных
направления
0 1 2 3
множества
0
1
2
127
Множество i
Схема сравнения
считывается сторока
Определяют номер множества
Тэг
Индекс
№ байта
31 11 10 4 3 0
Строка 16 байт
15 0
Определяет байты в строке
— позиция совпала
* — в этом блоке информация не достоверна

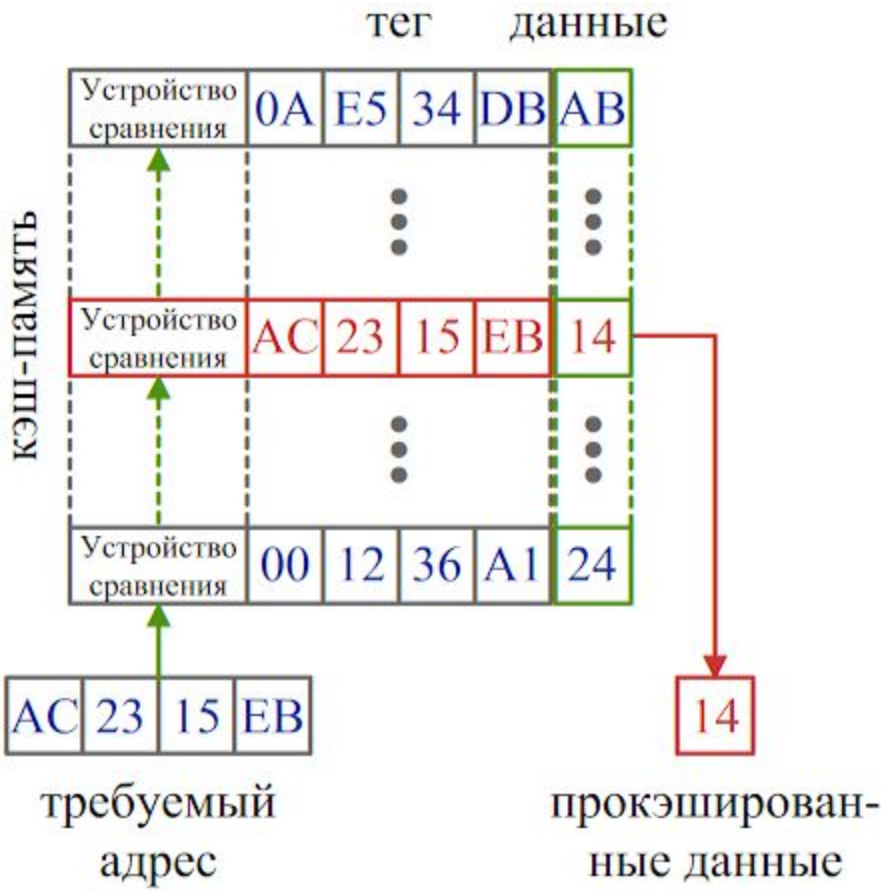

A31
A8
A7
A2
A1
A0
Поле метки
Индекс
Слово
Указание кэш-строки
Сравниваемые поля
Кэш-память
тег
24-разрядное поле метки
4-байтовое слово
Кэш-строка
Бит значимости

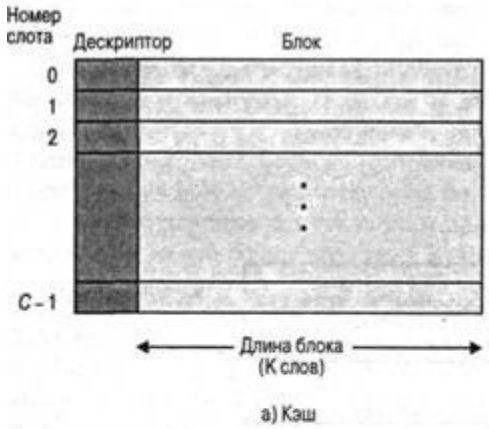

ЦПУ 0
ЦПУ 1
Кэш-память
Кэш-память
Оперативная память

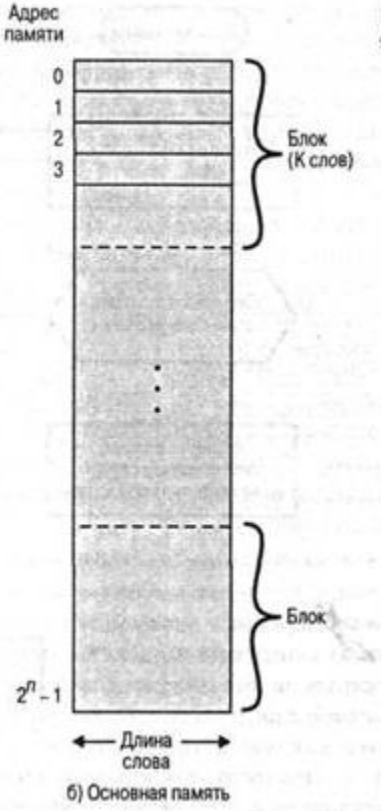

Main Memory
Cache Memory

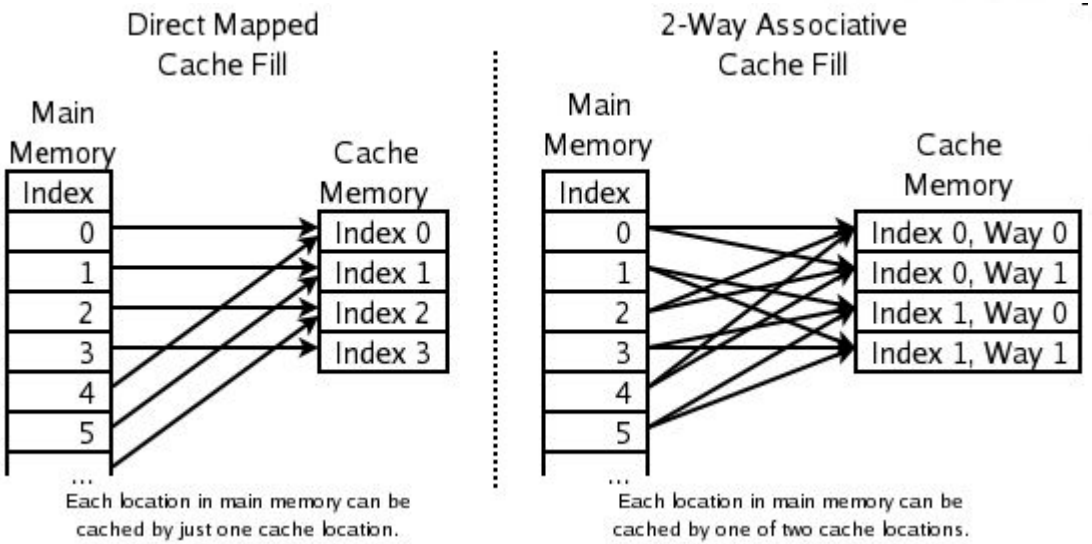

S
TAG
DATA

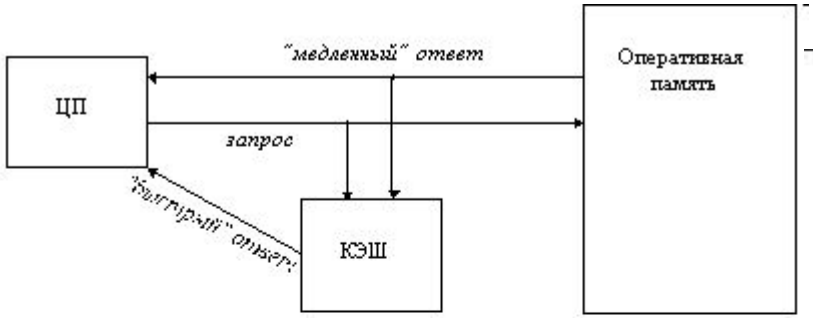

tag
µOPs
tag
µOPs
tag
µOPs
K
block 1
K''
block 1''
block 2
block 2''
block 3
set J
set J+1
set J+2
block 1
block 2
block 3
block 4
block 1''
block 2''
block 3''
block 4''
TTTTTTTTTTTTTTTTTTMMMSSSSSSSSMMM


тег
данные
кэш-память
Устройство сравнения
0A E5 34 DB AB
Устройство сравнения
AC 23 15 EB 14
Устройство сравнения
00 12 36 A1 24
AC 23 15 EB
14
требуемый адрес
прокэширован-ные данные


Номер слота
Дескриптор
Блок
0
1
2
C - 1
Длина блока (К слов)
а) Кэш


Адрес памяти
0
1
2
3
Блок (К слов)
Блок
2^n - 1
Длина слова
б) Основная память


Direct Mapped Cache Fill
Main Memory
Index
0
1
2
3
4
5
Cache Memory
Index 0
Index 1
Index 2
Index 3
Each location in main memory can be cached by just one cache location.
2-Way Associative Cache Fill
Main Memory
Index
0
1
2
3
4
5
Cache Memory
Index 0, Way 0
Index 0, Way 1
Index 1, Way 0
Index 1, Way 1
Each location in main memory can be cached by one of two cache locations.



Структура кэш-памяти

Адрес данных в ОП	Данные	Управл. информация	
		Бит модиф.	Бит обращ.

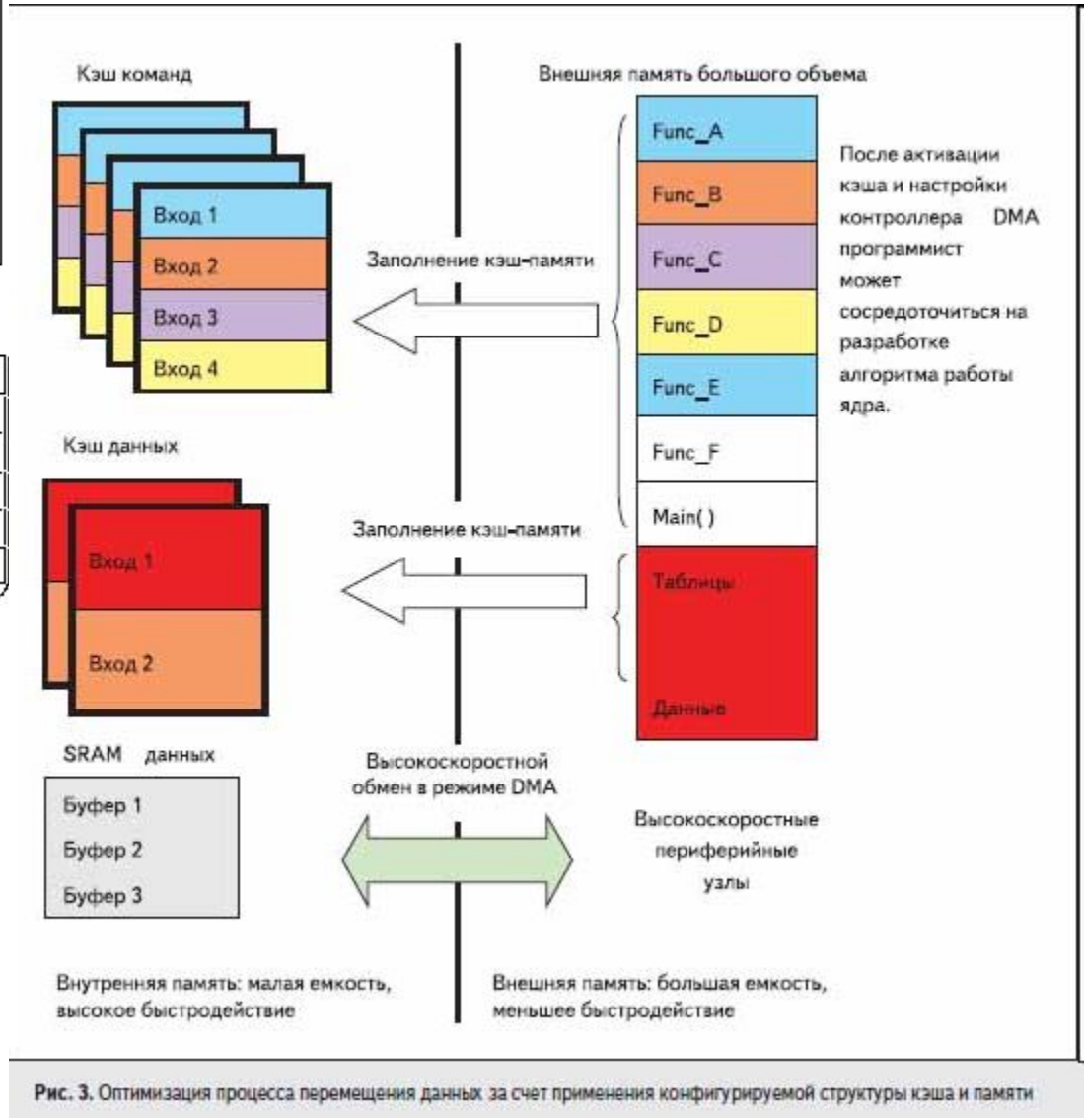
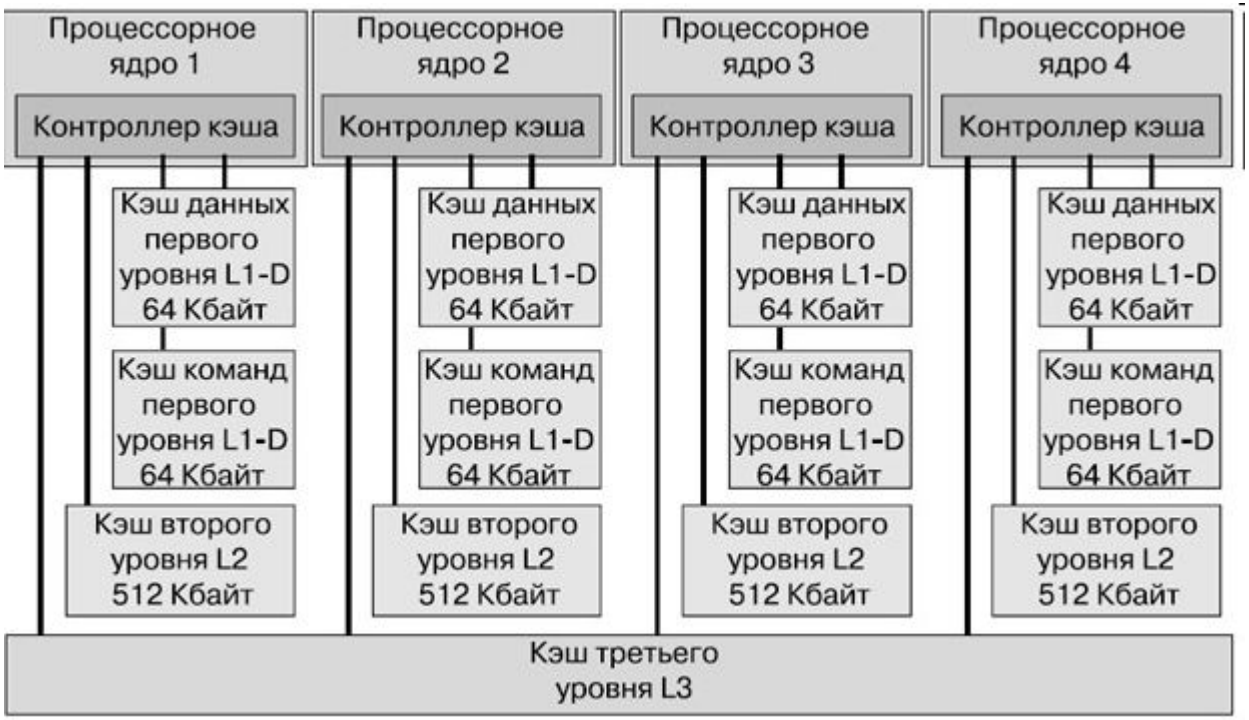
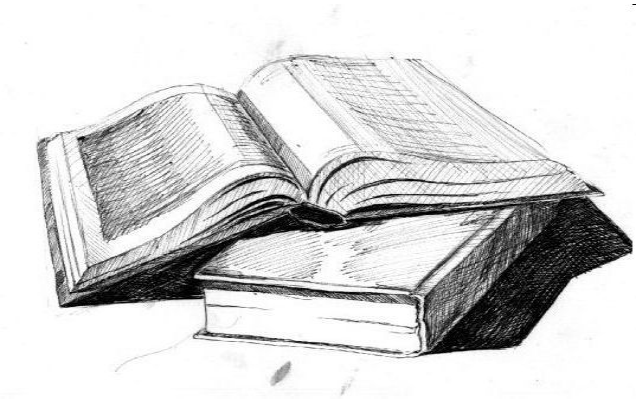


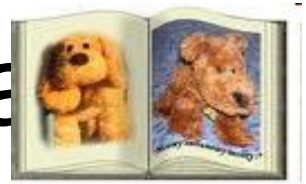
Рис. 3. Оптимизация процесса перемещения данных за счет применения конфигурируемой структуры кэша и памяти


Процессорное ядро 1
Процессорное ядро 2
Процессорное ядро 3
Процессорное ядро 4
Контроллер кэша
Контроллер кэша
Контроллер кэша
Контроллер кэша
Кэш данных первого уровня L1-D 64 Кбайт
Кэш данных первого уровня L1-D 64 Кбайт
Кэш данных первого уровня L1-D 64 Кбайт
Кэш данных первого уровня L1-D 64 Кбайт
Кэш команд первого уровня L1-D 64 Кбайт
Кэш команд первого уровня L1-D 64 Кбайт
Кэш команд первого уровня L1-D 64 Кбайт
Кэш команд первого уровня L1-D 64 Кбайт
Кэш второго уровня L2 512 Кбайт
Кэш второго уровня L2 512 Кбайт
Кэш второго уровня L2 512 Кбайт
Кэш второго уровня L2 512 Кбайт
Кэш третьего уровня L3

Используемые Интернет-ресурсы:

- http://citforum.ru/operating_systems/sos/glava_7.shtml
- http://x86.migera.ru/text/glava_2.html
- http://www.intuit.ru/department/os/osintro/10/
- http://www.moodle.ipm.kstu.ru/mod/resource/view.php?id=57





Используемая литература

- Книга «Ассемблер. Учебник для ВУЗов», авторы Михаил Гук, Виктор Юров
- Книга «Архитектура ЭВМ»,автор Мюллер
- Книга «Процессоры *Pentium4, Athlon* и *Duron*», авторы Михаил Гук, Виктор Юров
- Книга «Архитектура ЭВМ», автор Танненбаум