Лекция №5

Управление памятью

(часть 2)

Защита памяти

Защита памяти

- Доступ к общесистемным структурам данных и к пулам памяти возможен только из режима ядра.
- Преобразование виртуальных адресов процесса осуществляется только в те физические адреса, которые выделены данному процессу или входят в разделяемую часть виртуального адресного пространства.
- Аппаратная защита памяти (например, страницы кода помечаются атрибутом «только для чтения»).
- Совместно используемые объекты имеют списки контроля доступа (*ACL*). При обращении процесса к разделяемой памяти происходит проверка прав доступа.

Атрибуты защиты памяти

Атрибут	Разрешено	Запрещено
PAGE_NOACCESS	–	Все действия
PAGE_READONLY	Чтение	Запись, выполнение
PAGE_READWRITE	Чтение, запись	Выполнение
PAGE_EXECUTE	Выполнение	Чтение, запись
PAGE_EXECUTE_ READ	Чтение, выполнение	Запись
PAGE_EXECUTE_ READWRITE	Чтение, запись, выполнение	–
PAGE_WRITECOPY (копирование при записи)	Чтение. Запись в закрытую копию страницы	Выполнение
PAGE_EXECUTE_ WRITECOPY	Чтение, выполнение. Запись в закрытую копию страницы	–

Атрибуты защиты памяти

Атрибут	Разрешено	Запрещено
PAGE_GUARD	При чтении или записи страница перестаёт быть сторожевой	Нельзя комбинировать с PAGE_NOACCESS
PAGE_NOCACHE	Все действия разрешены	Кэширование страницы (имеет смысл для драйверов, например, буфер видео)
PAGE_ WRITECOMBINE	Позволяет объединять несколько операций записи на устройство в один пакет, что увеличивает скорость передачи данных	

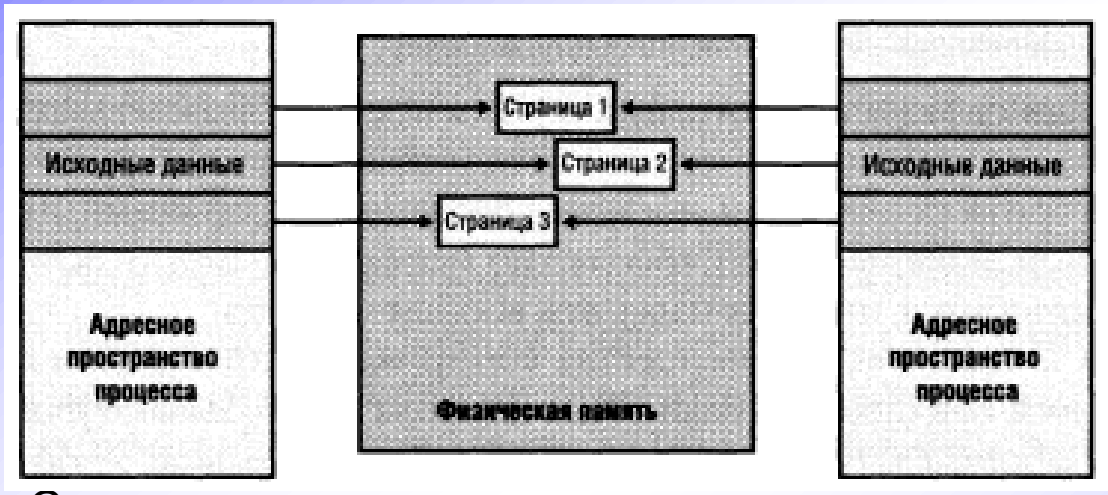
Сторожевые страницы

- Обращение к сторожевой странице приводит к исключению, после чего страница перестаёт быть сторожевой и с ней можно работать как с обычной страницей переданной памяти.
- Используется для увеличения размера стека. Нижняя страница стека потока является сторожевой. При заполнении стека происходит обращение к нижней странице. Генерируется исключение, потоку передаётся ещё одна страница, которая становится сторожевой.

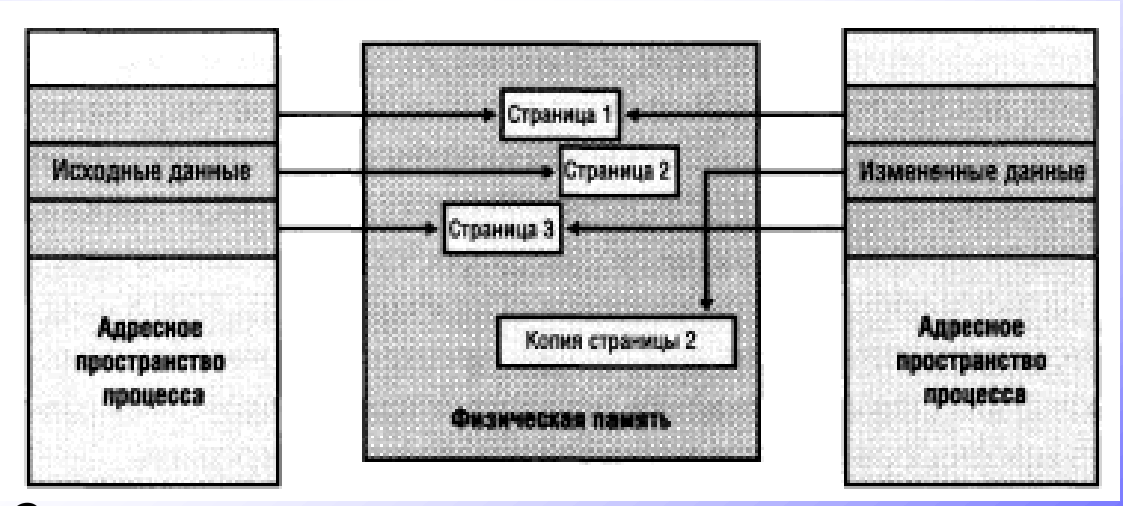
Копирование при записи

- Копирование при записи – механизм, использующийся для экономии физической памяти.
- Несколько процессов могут использовать одну и ту же страницу с атрибутом «копирование при записи», не изменяя её. Если процесс пытается модифицировать страницу, то в неразделяемой части адресного пространства процесса создаётся копия этой страницы, которая и будет изменяться.
- Используется при создании дочерних процессов, изначально работающих с разделяемыми страницами родителя. Только при попытке изменения какой-либо страницы происходит её копирование в неразделяемую память дочернего процесса.

Копирование при записи



Состояние страниц до копирования при записи



Состояние страниц после копирования при записи

Запрет на выполнение

- Data Execution Prevention (Предотвращение выполнения данных) – не позволяет приложению исполнять код из области памяти, помеченной как «данные».
- Предотвращает некоторые атаки, которые сохраняют код в области данных с помощью переполнения буфера.
- DEP работает в двух режимах: аппаратном, для процессоров с поддержкой данного атрибута и программном – для остальных процессоров.

Учёт использования памяти

Способы учёта использования памяти

- Использование битовых массивов.
- Использование списков свободных и занятых участков.

Управление памятью при помощи битовых массивов

- Память разделяется на единичные блоки размещения фиксированного размера.
- В битовом массиве каждому блоку соответствует один бит.
- Если блок свободен, то бит равен 0.
- Если блок занят, то бит равен 1.

а

б

1	1	1	1	1	1	0	0
0	0	1	1	0	0	0	0

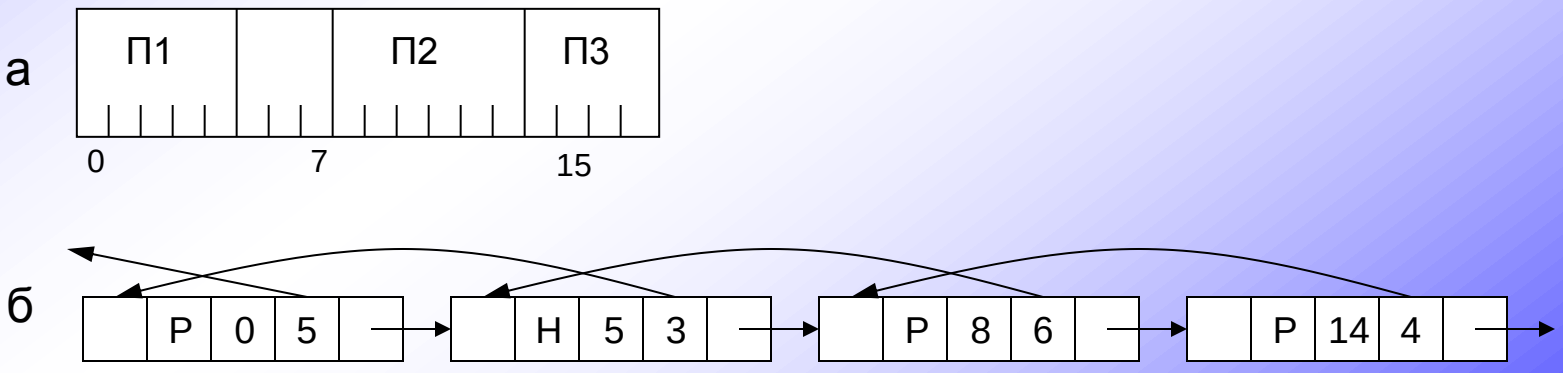
а – часть памяти;
б – битовый массив, соответствующий (а).

Особенности использования битовых массивов

- Уменьшение размера единичного блока приводит к увеличению размера битового массива.
- Увеличение размера единичного блока приводит к увеличению свободного места в последнем блоке каждого процесса.
- Поиск серии следующих друг за другом нулевых битов для размещения процесса является медленной операцией.

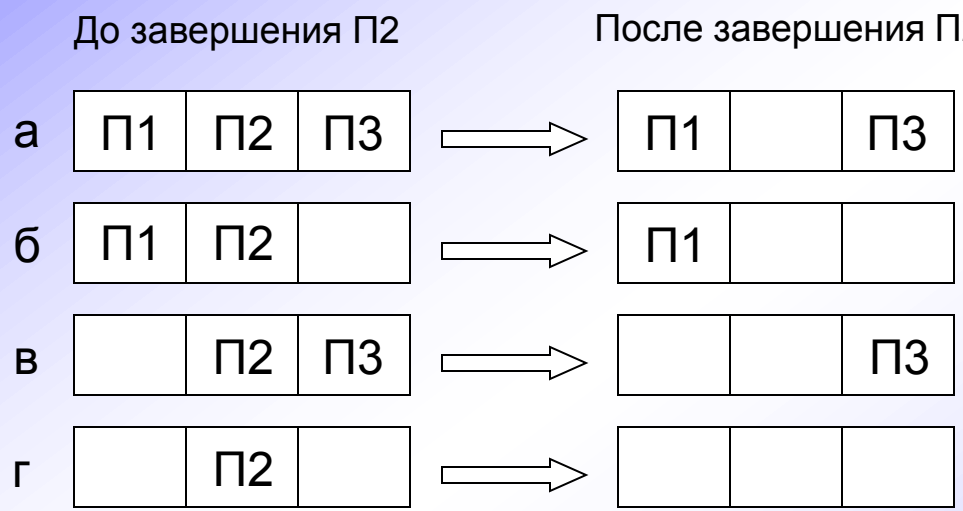
Управление памятью при помощи связных списков

Каждая запись в списке указывает является ли область памяти свободной (*H*) или занятой процессом (*P*); адрес, с которого начинается эта область; длину области; содержит указатели на предыдущую и следующую записи.



а – часть памяти;
б – связный список, соответствующий (а).

Изменение списка при завершении процесса



а – у области 2 происходит замена *P* на *H*;
б – области 2 и 3 объединяются: устанавливается бит *H*, адрес начала 2 области, длина – сумма длин 2 и 3;
в – области 1 и 2 объединяются: устанавливается бит *H*, адрес начала 1 области, длина – сумма длин 1 и 2;
г – области 1, 2 и 3 объединяются: устанавливается бит *H*, адрес начала 1 области, длина – сумма длин 1, 2 и 3;.

Выбор вытесняемой страницы

Алгоритмы выбора вытесняемой страницы

Основной принцип – вытесняется страница, которая предположительно не будет использоваться в ближайшее время.

Используются:

- признак модификации (*M*=0, если не было модификации за заданный период времени; *M*=1, если была модификация);
- признак обращения (*R*=0, если не было обращения за заданный период времени; *R*=1, если было обращение).

Периодически (например, при каждом прерывании по таймеру) признак обращения обнуляется.

Алгоритм *NRU* (неиспользовавшаяся в последнее время страница)

При страничном прерывании происходит проверка всех страниц и деление их на 4 класса на основании текущих значений *R* и *M*:

- класс 0 – не было обращений и изменений;
- класс 1 – не было обращений, страница изменена;
- класс 2 – было обращение, страница не изменена;
- класс 3 – произошло и обращение и изменение.

Вытесняется случайная страница из непустого класса с наименьшим номером.

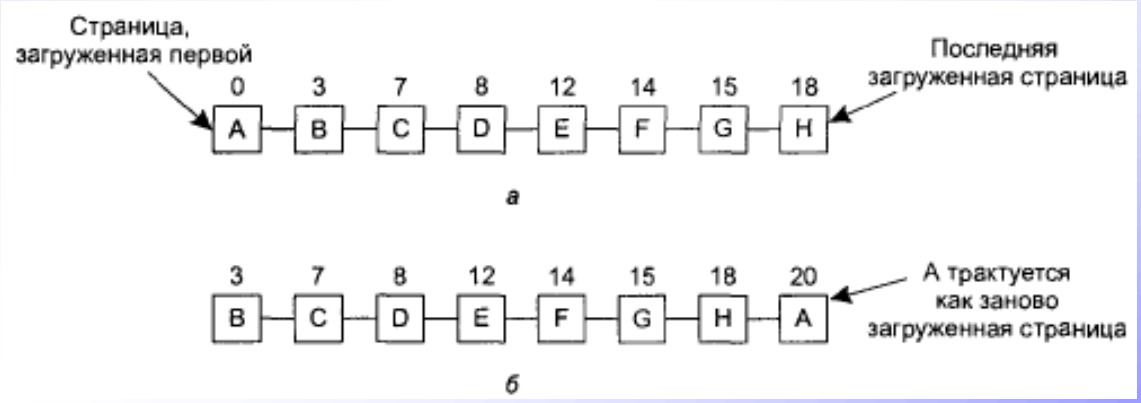
Алгоритм *FIFO* (вытесняется первая загруженная страница)

- Поддерживается список всех страниц, находящихся в памяти. Первая страница в списке – старейшая.
- При страничном прерывании вытесняется первая страница в списке, а новая добавляется в конец списка.

  Недостаток – может быть вытеснена часто используемая страница.

Алгоритм «вторая попытка»

- В дополнение к алгоритму *FIFO* у старейшей страницы проверяется признак обращения *R*.
- Если *R*=0, то страница вытесняется; если *R*=1, то ему присваивается значение 0, а страница переносится в конец списка и опять проверяется первая запись списка.
- Недостаток – медленная работа из-за передвижения по списку.



а – страницы, отсортированные в порядке очереди;

б – список страниц, если страничное прерывание произошло в момент времени 20, а у страницы *A* признак обращения *R*=1.

Алгоритм LRU (страница, не использовавшаяся дольше всего)

Из оперативной памяти вытесняется страница, не использовавшаяся дольше всего.

Пример аппаратной реализации:

- компьютер оснащается аппаратным счётчиком, который возрастает при каждой команде;
- дескриптору страниц добавляется поле для хранения счётчика;
- после обращения к странице значение счётчика сохраняется в записи таблицы;
- при страничном прерывании вытесняется страница с наименьшим значением счётчика.

Алгоритм «рабочий набор»

- Рабочий набор – множество страниц, которое процесс использует в данный момент.
- Основная задача – отслеживание рабочего набора каждого процесса и обеспечение его нахождения в памяти до запуска процесса.
- В любой момент времени существует множество страниц, использовавшихся за заданный интервал времени. Данное множество является рабочим набором.
- При страничном прерывании вытесняется страница, не входящая в рабочий набор.

Алгоритм «рабочий набор»

- τ – интервал виртуального времени, в пределах которого рассматривается рабочий набор;
- если *R*=0 и возраст страницы>τ, то данная страница не находится в рабочем наборе и вытесняется;
- если все страницы с *R*=0 имеют возраст страницы≤τ, то вытесняется страница с наибольшим возрастом.

Алгоритм WSClock

- В отличие от алгоритма «рабочий набор» используется кольцевой список страниц.
- Рассматривается страница, на которую указывает стрелка.
- Если *R*=1, то *R*=0 и стрелка передвигается на следующую страницу.

Алгоритм WSClock

- Если *R*=0, возраст страницы>т и *M*=0, то страница вытесняется.

Диспетчер рабочих наборов

- Диспетчер рабочих наборов реализует усечение рабочего набора, запись модифицированных страниц и т.д.
- Вызывается раз в секунду или при уменьшении объёма свободной памяти ниже заданного порога.

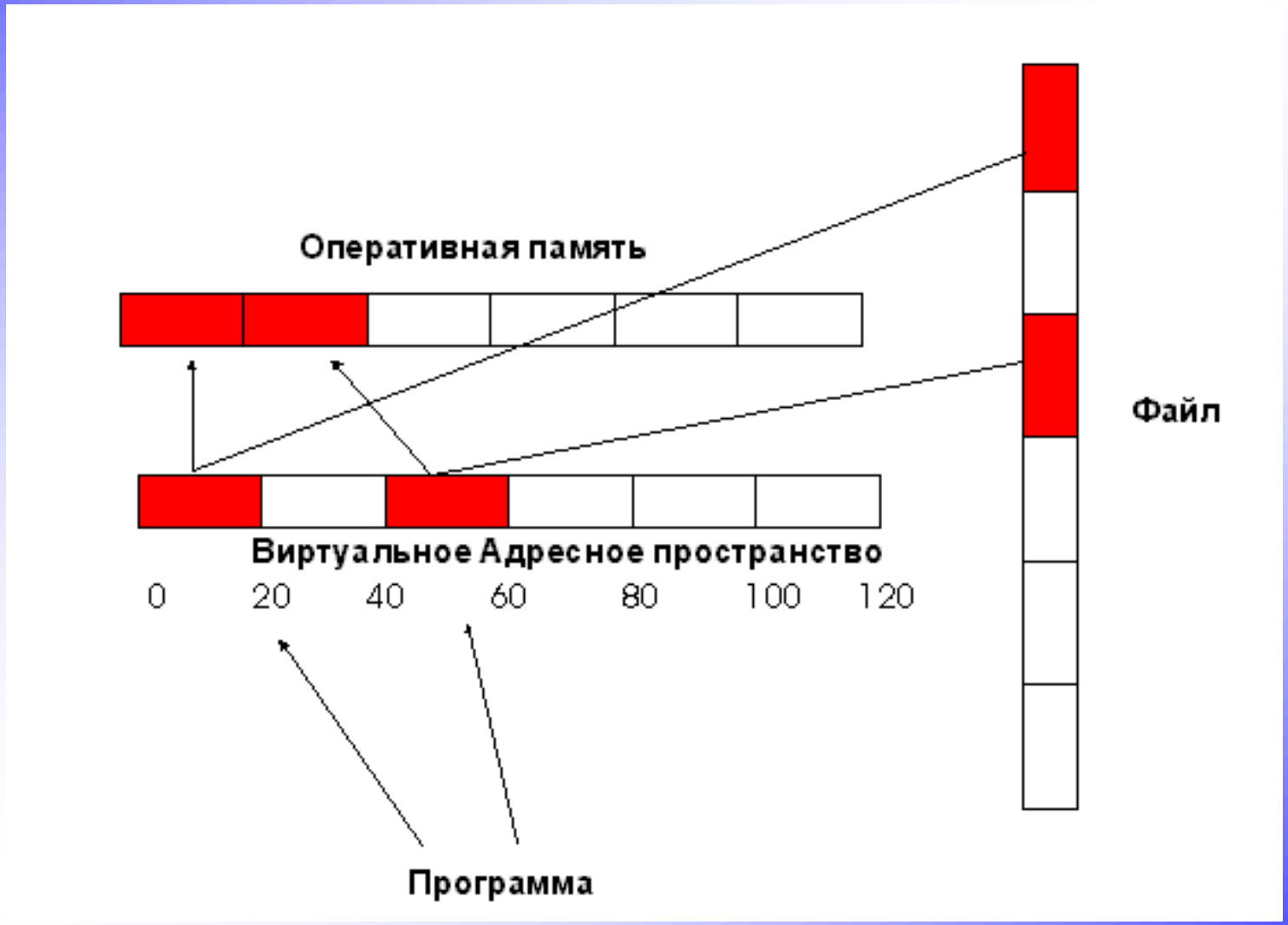
Файлы, проецируемые в память

- Проецируемые в память файлы – файлы на диске, с которыми работает процесс.
- Изменённые спроецированные страницы записываются на диск, если необходимо уменьшить список модифицированных страниц или когда страницы модифицированных файлов находятся в списке более 5 минут.

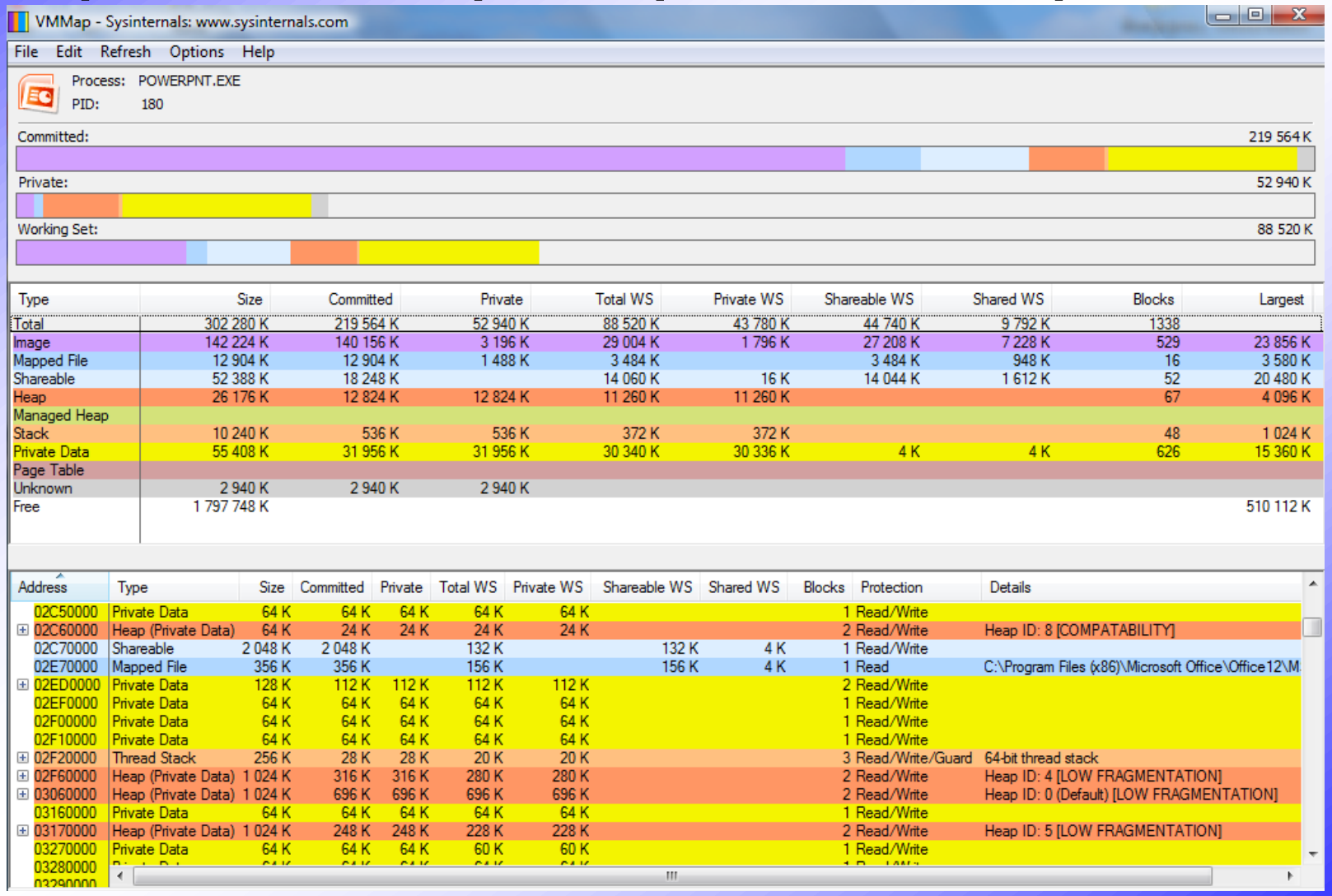
Разреженная память

- Разреженная структура памяти применяется при работе с большими файлами, содержащими данные, т.к. часто работать нужно только с частью файла.
- Всё необходимое адресное пространство резервируется, но не выделяется. Выделяется и заполняется только та часть памяти, которая необходима для работы.

Разреженная память



Пример структуры виртуального адресного пространства процесса



Type	Size	Committed	Private	Total WS	Private WS	Shareable WS	Shared WS	Blocks	Largest
Total	302 280 K	219 564 K	52 940 K	88 520 K	43 780 K	44 740 K	9 792 K	1338	
Image	142 224 K	140 156 K	3 196 K	29 004 K	1 796 K	27 208 K	7 228 K	529	23 856 K
Mapped File	12 904 K	12 904 K	1 488 K	3 484 K		3 484 K	948 K	16	3 580 K
Shareable	52 388 K	18 248 K		14 060 K	16 K	14 044 K	1 612 K	52	20 480 K
Heap	26 176 K	12 824 K	12 824 K	11 260 K	11 260 K			67	4 096 K
Managed Heap									
Stack	10 240 K	536 K	536 K	372 K	372 K			48	1 024 K
Private Data	55 408 K	31 956 K	31 956 K	30 340 K	30 336 K	4 K	4 K	626	15 360 K
Page Table									
Unknown	2 940 K	2 940 K	2 940 K						
Free	1 797 748 K								510 112 K

Address	Type	Size	Committed	Private	Total WS	Private WS	Shareable WS	Shared WS	Blocks	Protection	Details
02C50000	Private Data	64 K	64 K	64 K	64 K	64 K			1	Read/Write	
02C60000	Heap (Private Data)	64 K	24 K	24 K	24 K	24 K			2	Read/Write	Heap ID: 8 [COMPATABILITY]
02C70000	Shareable	2 048 K	2 048 K		132 K		132 K	4 K	1	Read/Write	
02E70000	Mapped File	356 K	356 K		156 K		156 K	4 K	1	Read	C:\Program Files (x86)\Microsoft Office\Office12\M
02ED0000	Private Data	128 K	112 K	112 K	112 K	112 K			2	Read/Write	
02EF0000	Private Data	64 K	64 K	64 K	64 K	64 K			1	Read/Write	
02F00000	Private Data	64 K	64 K	64 K	64 K	64 K			1	Read/Write	
02F10000	Private Data	64 K	64 K	64 K	64 K	64 K			1	Read/Write	
02F20000	Thread Stack	256 K	28 K	28 K	20 K	20 K			3	Read/Write/Guard	64-bit thread stack
02F60000	Heap (Private Data)	1 024 K	316 K	316 K	280 K	280 K			2	Read/Write	Heap ID: 4 [LOW FRAGMENTATION]
03060000	Heap (Private Data)	1 024 K	696 K	696 K	696 K	696 K			2	Read/Write	Heap ID: 0 (Default) [LOW FRAGMENTATION]
03160000	Private Data	64 K	64 K	64 K	64 K	64 K			1	Read/Write	
03170000	Heap (Private Data)	1 024 K	248 K	248 K	228 K	228 K			2	Read/Write	Heap ID: 5 [LOW FRAGMENTATION]
03270000	Private Data	64 K	64 K	64 K	60 K	60 K			1	Read/Write	
03280000											

Кэш-память

Кэш-память

Кэш-память – способ совместного функционирования двух типов запоминающих устройств (ЗУ), отличающихся временем доступа и стоимостью хранения данных. Данный способ позволяет уменьшить среднее время доступа к данным за счёт копирования данных из более медленного ЗУ (основная память) в более быстрое. Управляется только системными средствами.

Кэш-память – память небольшого объёма с высокой скоростью работы.

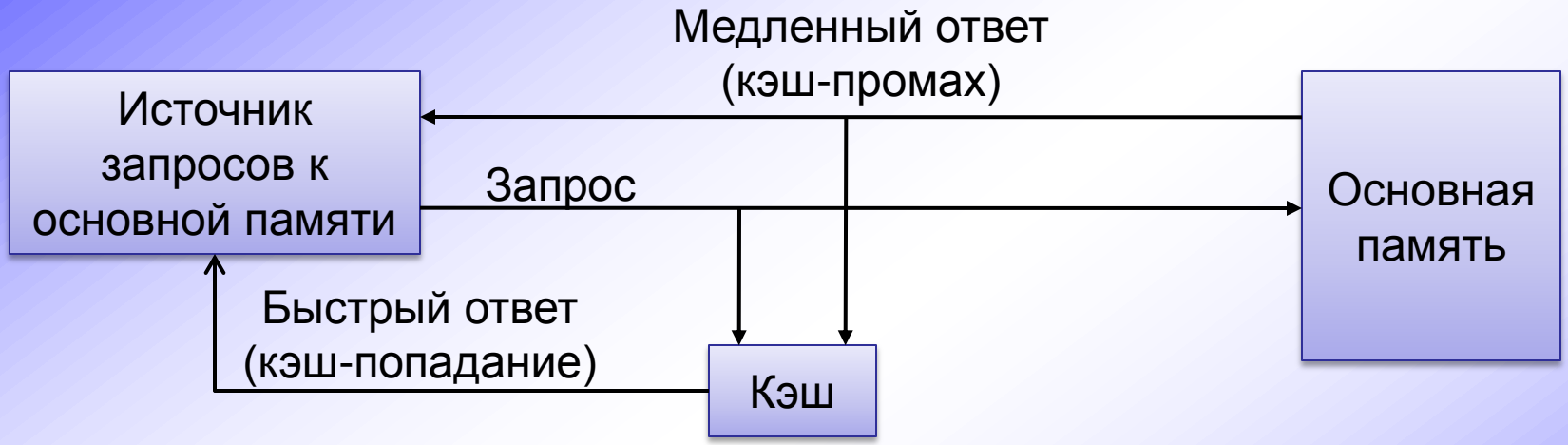
Принцип действия кэш-памяти

Кэш-память – совокупность записей обо всех загруженных из основной памяти элементах.

Записи включают:

- значение элемента данных;
- адрес, который этот элемент имеет в основной памяти;
- дополнительную информацию, используемую для реализации алгоритма замещения данных в кэше (признак модификации, признак действительности данных).

Схема функционирования кэш-памяти



- Кэш-попадание – при обращении данные оказываются в кэш-памяти и передаются источнику запроса.
- Кэш-промах – при обращении данные не оказываются в кэш-памяти, поэтому они считываются из основной памяти, передаются источнику запроса и одновременно копируются в кэш-память.

Временная локальность

- Временная локальность – после обращения к какому-либо адресу в оперативной памяти существует высокая вероятность того, что в ближайшее время произойдёт обращение к этому же адресу.
- Следствие: в кэш обязательно записываются данные, расположенные по запрашиваемому адресу.

Пространственная локальность

- Пространственная локальность – после обращения к какому-либо адресу в оперативной памяти существует высокая вероятность того, что в ближайшее время произойдёт обращение к соседним адресам.
- Следствие: в кэш записываются данные, расположенные по запрашиваемому и соседним адресам.

Вытеснение данных из кэш-памяти

- Если данные не изменялись, то запись объявляется свободной путём сброса признака действительности.
- Если данные изменялись, то происходит их копирование в основную память и запись объявляется свободной.
- Сброшенное значение признака действительности позволяет заносить в эту запись новые данные.

Согласование данных в кэше и основной памяти

- При записи данных в основную память просматривается кэш, если в кэше эти данные отсутствуют, то запись идёт только в основную память.

Сквозная запись:

- если данные в кэше есть, то запись проводится и в кэш и в основную память.

Обратная запись:

- если данные в кэше есть, то запись проводится только в кэш и устанавливается признак модификации.

  Выгрузка модифицированных данных может осуществляться в первую очередь во время замещения или в фоновом режиме.

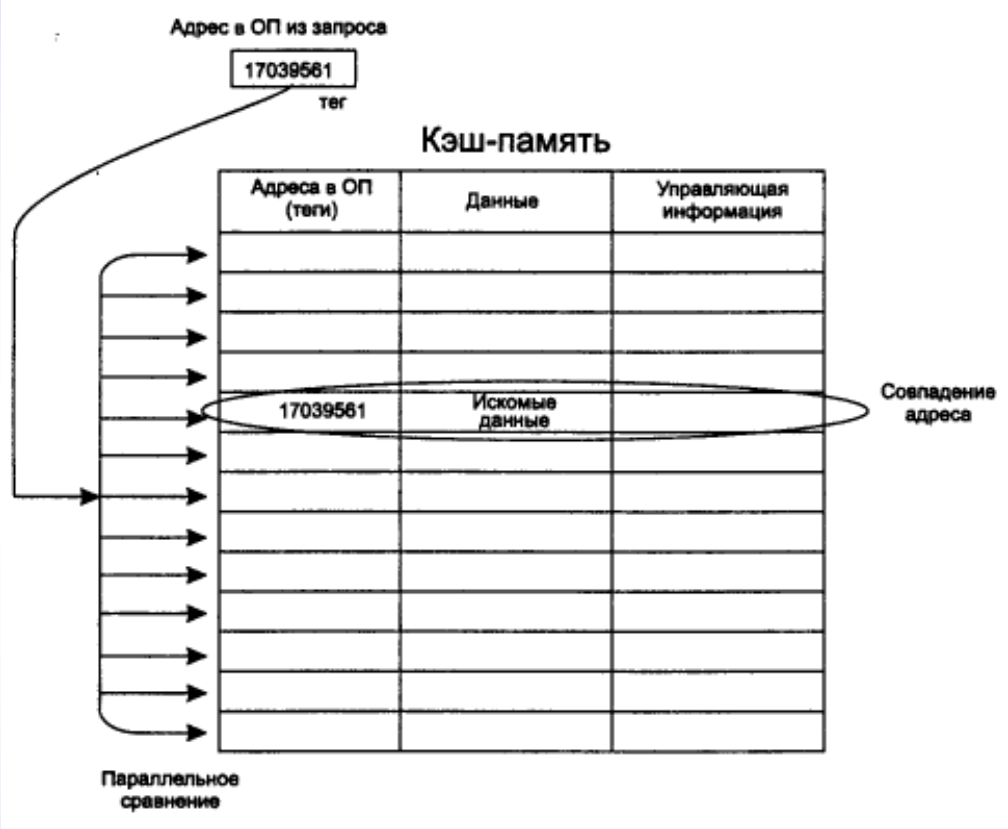
Способы отображения основной памяти на кэш

- Случайное отображение.
- Детерминированное отображение.
- Комбинированный способ.

Случайное отображение

Элемент основной памяти может быть размещён в произвольном месте кэш-памяти.

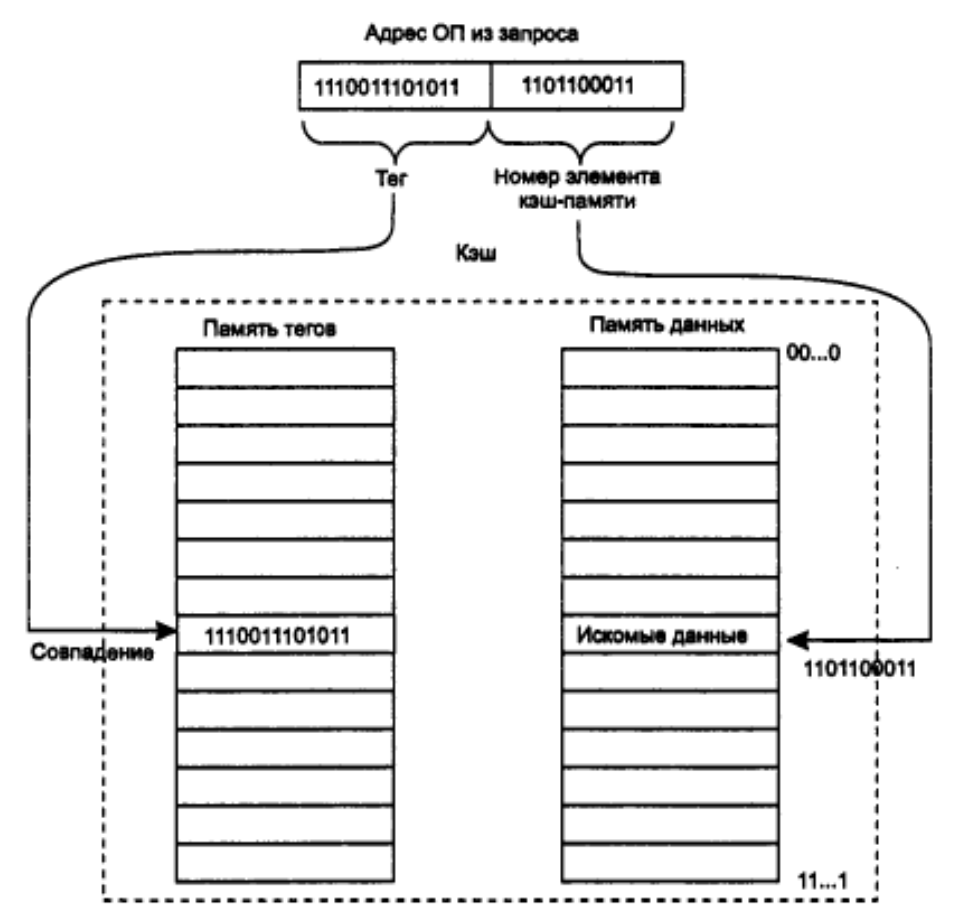
Ассоциативный поиск – сравнение производится параллельно со всеми записями кэша по тэгу.



Детерминированное отображение

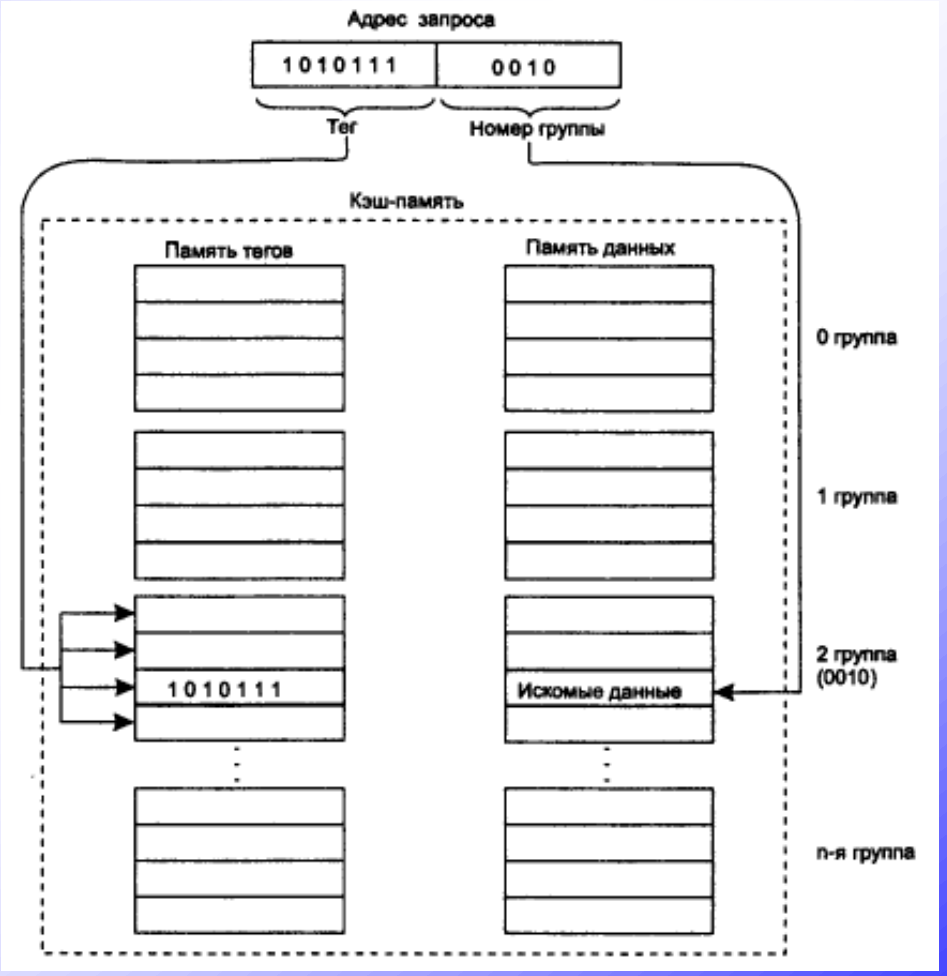
Любой элемент основной памяти всегда отображается в одно и то же место кэш-памяти.

Прямое отображение – номер строки кэш-памяти равен нескольким разрядам адреса отображаемого элемента в оперативной памяти.



Комбинированный способ

Адрес основной памяти отображается на группу записей, номер которой зависит от адреса элемента в основной памяти. В рамках группы копирование данных может производиться в любую запись.



Возможные особенности работы кэш-памяти

- В некоторых реализациях при отсутствии данных в кэше они копируются в него из основной памяти вне зависимости от типа запроса (чтение или запись).
- Поиск в кэше и основной памяти может начинаться одновременно; если в кэше обнаруживаются данные, то поиск в основной памяти прерывается.
- Возможна работа с двухуровневым кэшем. Сначала поиск проводится в кэше первого уровня, затем второго. При этом способы согласования на каждом уровне могут быть различны.

Рассмотренные вопросы

- Защита памяти.
- Способы учёта свободной и занятой памяти (битовый массив и связные списки).
- Алгоритмы выбора страницы, вытесняемой во внешнюю память.
- Работа кэш-памяти.

Всем спасибо – все свободны,

если нет вопросов