Безопасность операционных систем

Управление памятью

Иерархия памяти

Вид памяти	Быстродействие	Стоимость
Кэш-память	Очень высокое	Высокая
Оперативная	Высокое	Средняя
Дисковые накопители	Низкое	Низкая
Сменные накопители / сеть	Низкое	Низкая

Менеджер (диспетчер) памяти

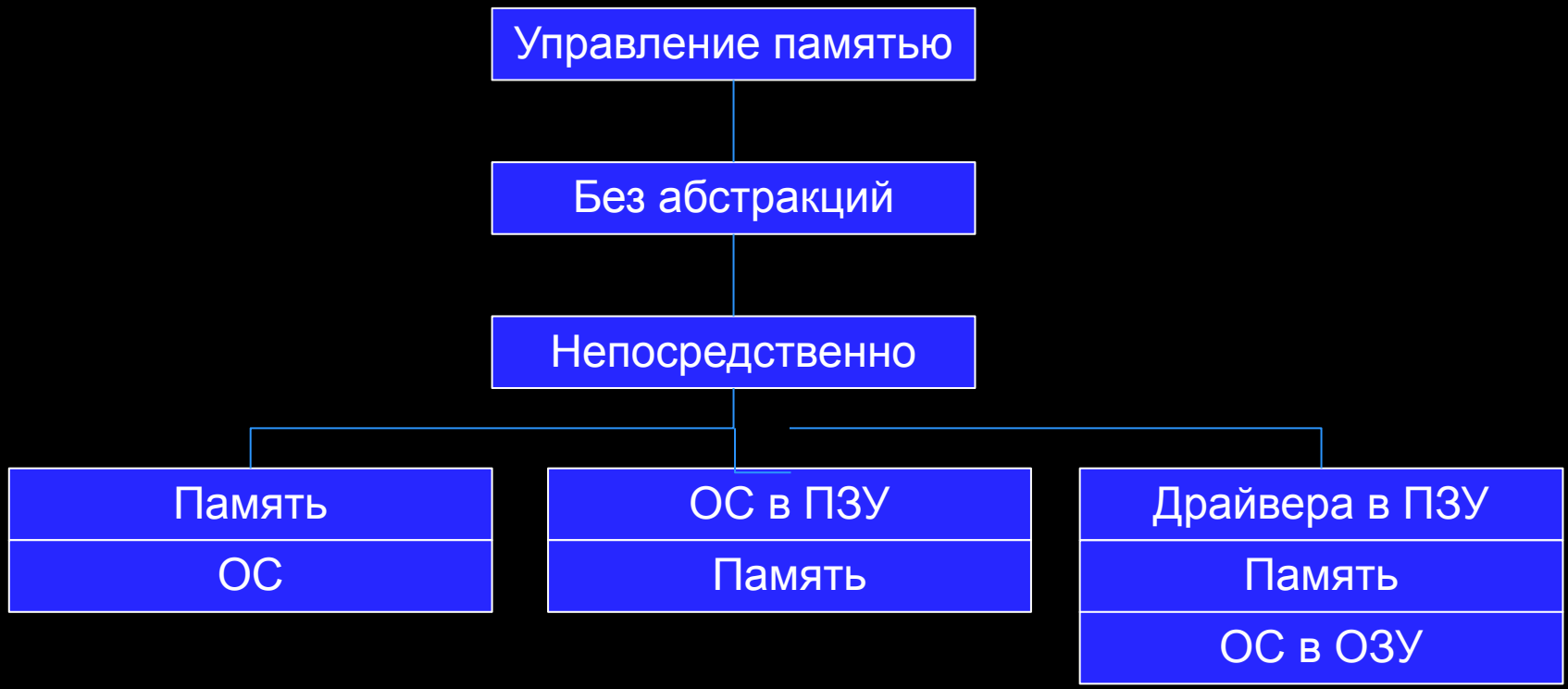
- Задача ОС – превратить иерархию памяти в модель (абстракцию) и управлять этой моделью.
- Диспетчер (менеджер) памяти – часть операционной системы, которая предназначается для управления памятью и осуществляет:
  - Контроль используемых участков оперативной памяти
  - Выделение памяти процессам
  - Освобождение ненужной более памяти
- Будем рассматривать программную модель оперативной памяти и способы ее эффективного использования

Память без использования абстракций

- Ранние компьютеры не использовали абстракцию памяти.
- Все программы работали с памятью непосредственно.
- Адрес памяти указанный в программе соответствовал физической ячейке памяти.
- Модель памяти – набор адресов от 0 до какой-то величины, где каждый адрес соответствует ячейке, содержащей какое-либо количество бит (обычно 8).
- Содержание в памяти двух работающих программ не представляется возможным.
- В современных системах применяется во встроенных устройствах, там где пользователь не может вмешаться в работу системы и перечень запущенных программ заранее определен.

Варианты использования памяти

А – использовалась на универсальных машинах и миникомпьютерах, Б – на КПК и встроенных системах, В – в ранних персональных компьютерах.


Управление памятью
Без абстракций
Непосредственно
Память
ОС
ОС в ПЗУ
Память
Драйвера в ПЗУ
Память
ОС в ОЗУ

Особенности вариантов

- Программа пользователя может затереть операционную систему.
- Процессы запускаются по одному. При запуске нового процесса старая программа в памяти затирается.
- Единственный способ обеспечения параллельности – запуск нескольких потоков, работающих с общей памятью.
- Не подходит большинству пользователей, т.к. может потребоваться запустить несколько разнородных программ

Использование защитных ключей (1/2)

- Применялось на IBM 360
- Память делилась на блоки по 2 КБ, каждому блоку соответствовал код защиты, содержащийся в специальных регистрах, отдельно от процессора.
- Слово состояния программы (PSW, несколько регистров) также содержало защитный ключ, относящийся к текущей исполняемой программе. Программа могла получить доступ только к тем блокам, код защиты которых совпадал с кодом, хранящемся в PSW.
- Распределением ключей занималась операционная система.


ЕС 1045.01

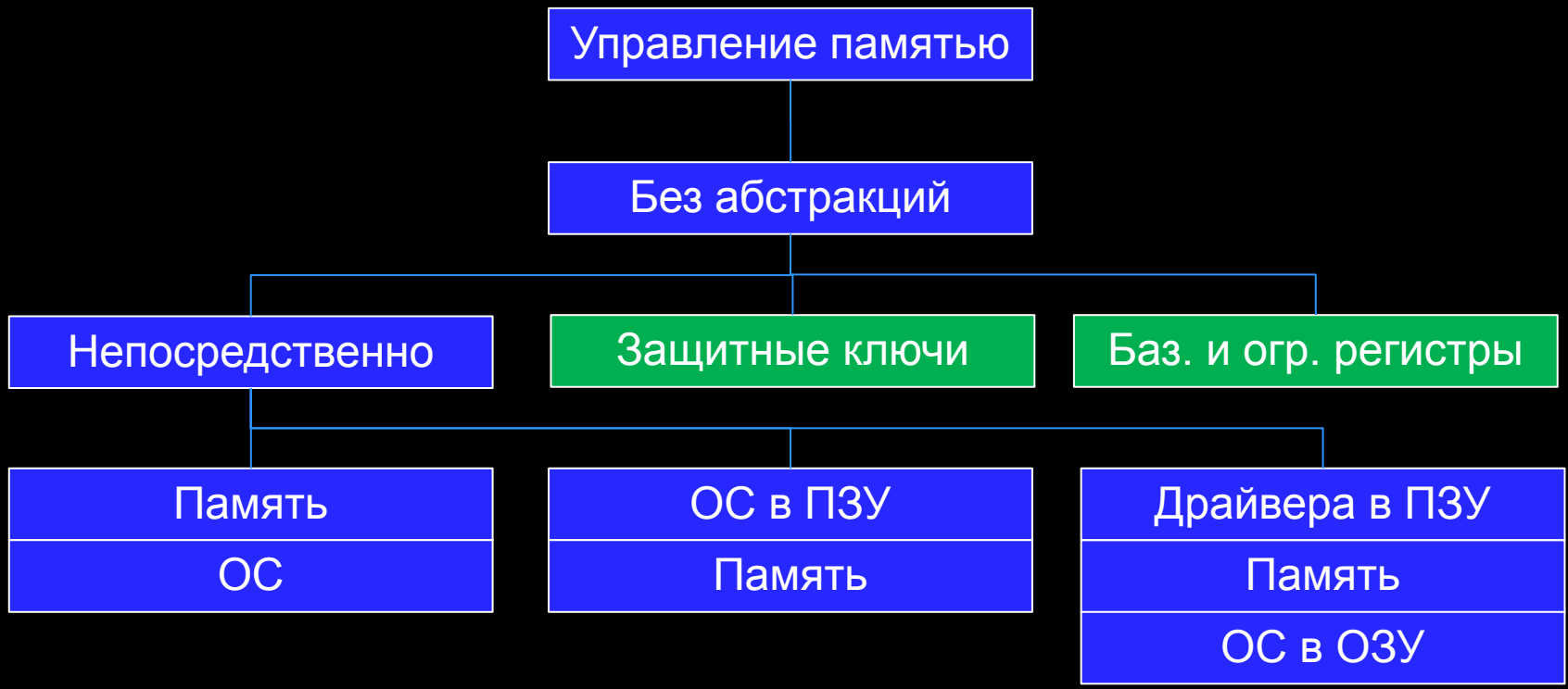
Использование защитных ключей (2/2)

- При таком подходе программы могут загружаться в различные места оперативной памяти.

- Нельзя использовать абсолютную адресацию или при загрузке программ необходимо изменять все встречающиеся в программе адреса (в этом случае загрузчик должен уметь различать адреса и константы).

Использование регистров «базовый» и «ограничительный»

- Метод использовался в процессорах 8088 и первых версиях процессоров 80x86 Intel.
- Адресное пространство каждого процесса проецируется на различные части физической памяти.
- Каждый процессор оснащается двумя аппаратными регистрами
  - Базовый регистр
  - Ограничительный регистр
- Программы загружаются в свободные области памяти без модификации адресов в процессе загрузки. В базовый регистр загружается физический адрес начала области, с которого загружена программа, а в ограничительный – длина программы. При каждом обращении к памяти для получения физического адреса складывается значение из базового регистра и адреса, указанного в операторе. При этом проверяется выход за границу адресов, отведенных программе.
- Обычно значения базового и ограничительного регистров может изменить только операционная система.
- Недостаток – необходимость сложения и сравнения при каждом обращении к памяти.


Управление памятью
Без абстракций
Непосредственно
Защитные ключи
Баз. и огр. регистры
Память
ОС
ОС в ПЗУ
Память
Драйвера в ПЗУ
Память
ОС в ОЗУ

Свопинг и виртуальная память

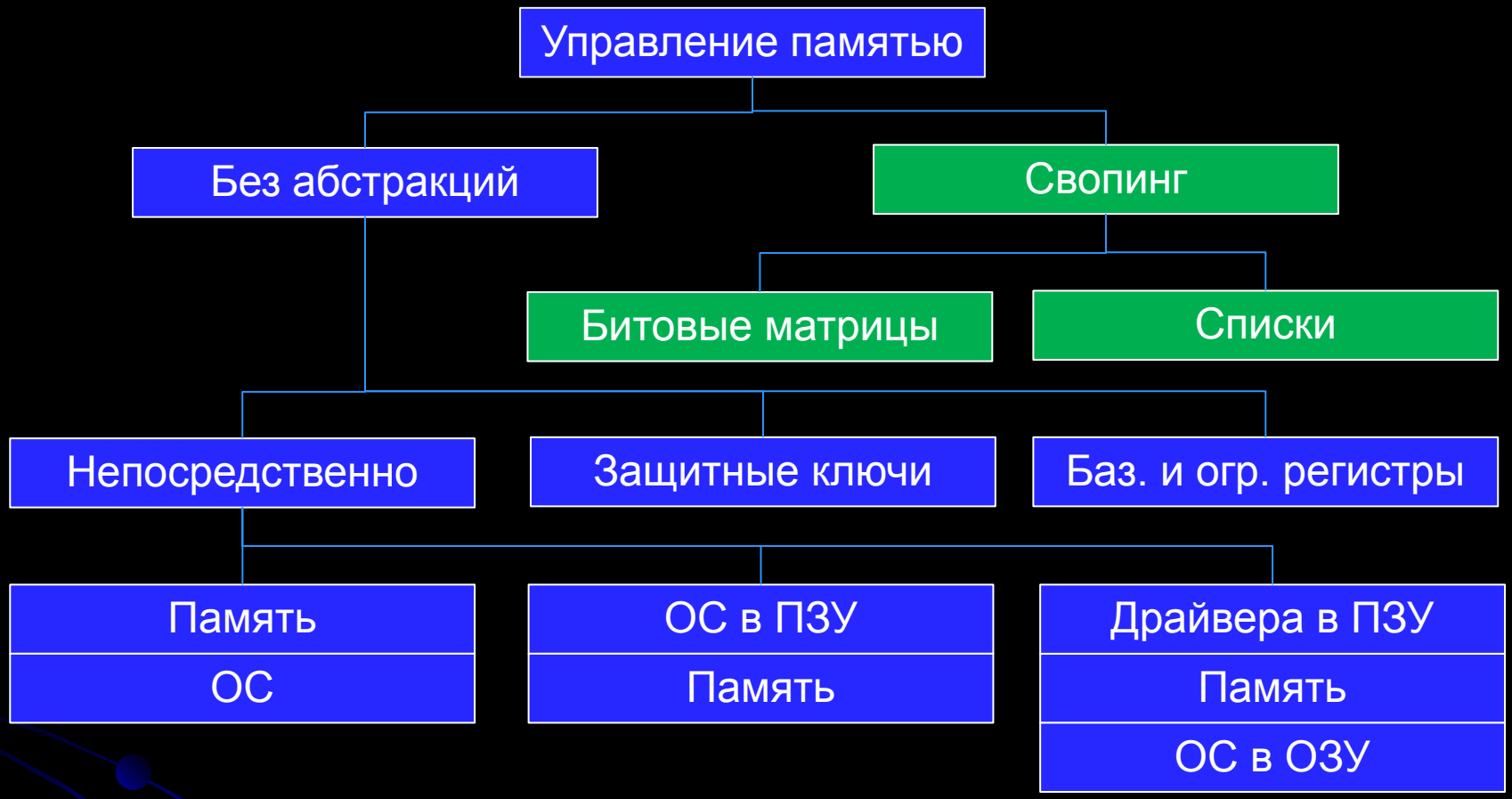
- Рассмотренные схемы будут работать только до тех пор, пока все программы могут размещаться в оперативной памяти.
- В реальной жизни память слишком дорогой ресурс и его не бывает достаточно для размещения всех работающих процессов (их количество в современных системах – несколько десятков).
- Основные подходы для преодоления перегрузки оперативной памяти – свопинг и виртуальная память.
- ***Свопинг*** – процесс размещается в памяти целиком, запускается на некоторое время, а затем сбрасывается на диск. Бездействующие процессы большую часть времени хранятся на диске в нерабочем состоянии и не занимают оперативной памяти.
- ***Виртуальная память*** – программа может работать находясь в оперативной памяти лишь частично.

Проблемы свопинга

- Проблема загрузки программы в разное время в разные адреса оперативной памяти (можно решить программно перестраивая адреса или с использованием базового и ограничительного регистров)
- Проблема фрагментации памяти. Может быть решена, но решение требует значительных ресурсов.
- Проблема разрастания сегментов данных программ. Решается перемещением сегмента в памяти, но также требует значительных ресурсов. Можно заранее выделять больше места, но нужно помнить, что копирование пустого места на диск при свопинге также потребует значительных ресурсов.

Свопинг: управление свободной памятью

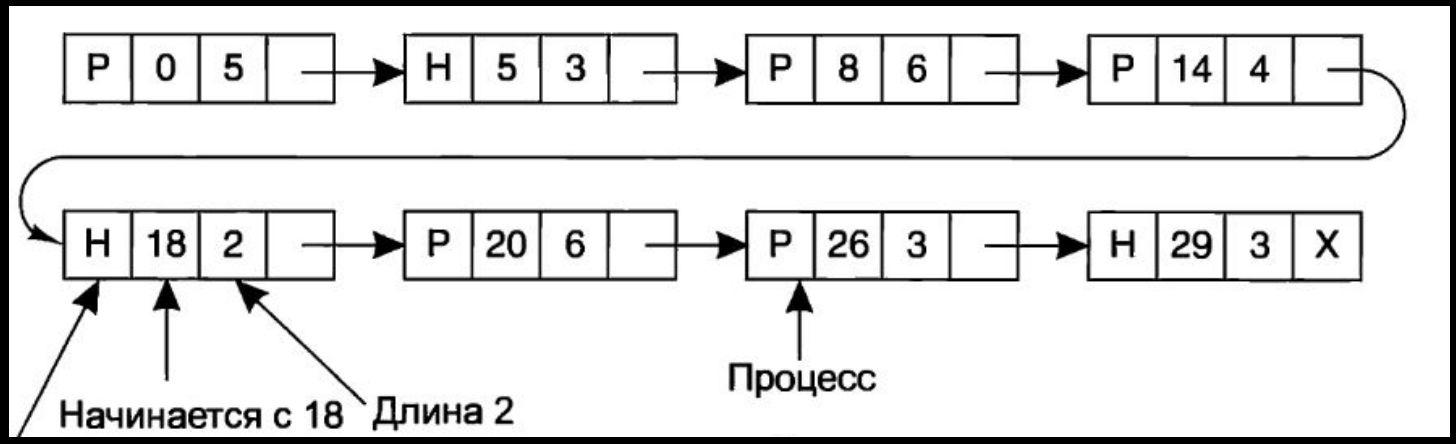
- Операционная система управляет процессом динамического распределении памяти.
- Способы отслеживания использования памяти:
  - Битовые матрицы
  - Списки свободного пространства


Управление памятью
Без абстракций
Свопинг
Битовые матрицы
Списки
Непосредственно
Защитные ключи
Баз. и огр. регистры
Память
ОС
ОС в ПЗУ
Память
Драйвера в ПЗУ
Память
ОС в ОЗУ

Использование битовых матриц

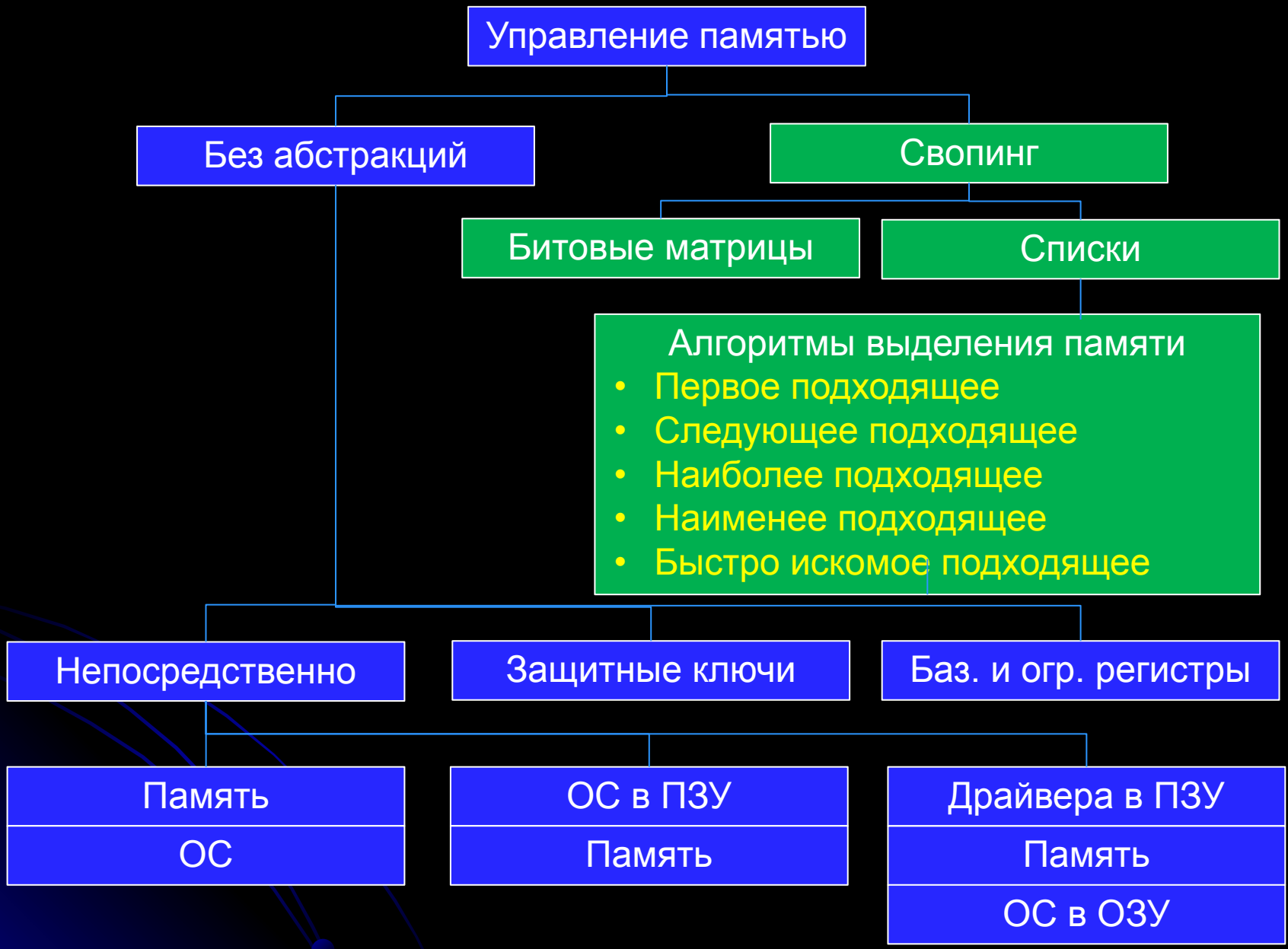
- Память делится на блоки фиксированного размера. Размер зависит от системы и может меняться от нескольких байт до нескольких килобайт.
- Создается битовая матрица, каждому биту в которой соответствует один блок памяти. 0 – блок свободен, 1 – блок занят.
- Требует достаточно мало дополнительной памяти.
- Проблема – если в памяти необходимо разместить процесс из N блоков, то в матрице необходимо найти фрагмент, состоящий из N последовательных нулей. Поиск в битовой матрицы последовательности заданной длины достаточно медленная операция из-за того, что последовательность может пересекать границы слов.

Использование связанных списков



Алгоритмы выделения памяти

- Первое подходящее – самый быстрый алгоритм
- Следующее подходящее – поиск свободного места начинается с того места в списке, где завершил работу предыдущий поиск. Производительность несколько хуже, чем «первое подходящее»
- Наиболее подходящее – выбирается наименьшее подходящее пустое пространство. Работает медленнее, расточительнее расходует память, т.к. оставляет небольшие неиспользуемые пространства.
- Наименее подходящее – выбирается самый большой свободный фрагмент. Моделирование показывает низкую эффективность.
- Все алгоритмы можно ускорить за счет ведения отдельных списков для свободных и занятых участков памяти. Усложняется процедура освобождения памяти. Можно список свободных мест сортировать по размеру. Можно список пустых пространств хранить в самих пустых пространствах.
- Быстро искомое подходящее – ведение отдельных списков для наиболее востребованных размеров.
- Проблема всех раздельных списков – требуется анализ необходимости слияния свободных фрагментов при освобождении памяти.


Управление памятью
Без абстракций
Свопинг
Битовые матрицы
Списки
Алгоритмы выделения памяти
• Первое подходящее
• Следующее подходящее
• Наиболее подходящее
• Наименее подходящее
• Быстро искомое подходящее
Непосредственно
Защитные ключи
Баз. и огр. регистры
Память
ОС
ОС в ПЗУ
Память
Драйвера в ПЗУ
Память
ОС в ОЗУ

Недостатки свопинга

- Недостаток свопинга
  - медленная работа при больших размерах программ.
  - нельзя запустить программы, размер которой превышает размер оперативной памяти. Пример – самораспаковывающиеся архивы.

- Для размещения в памяти больших программ придумана концепция оверлеев – разбиение программ на логические части, которые должны работать последовательно.
- Нагрузка по разбиению программы на части ложилась на программистов и требовала высокой квалификации.
- Виртуальная память (1961) позволяет возложить всю работу по загрузке фрагментов программ на компьютер.

Понятие адресного пространства

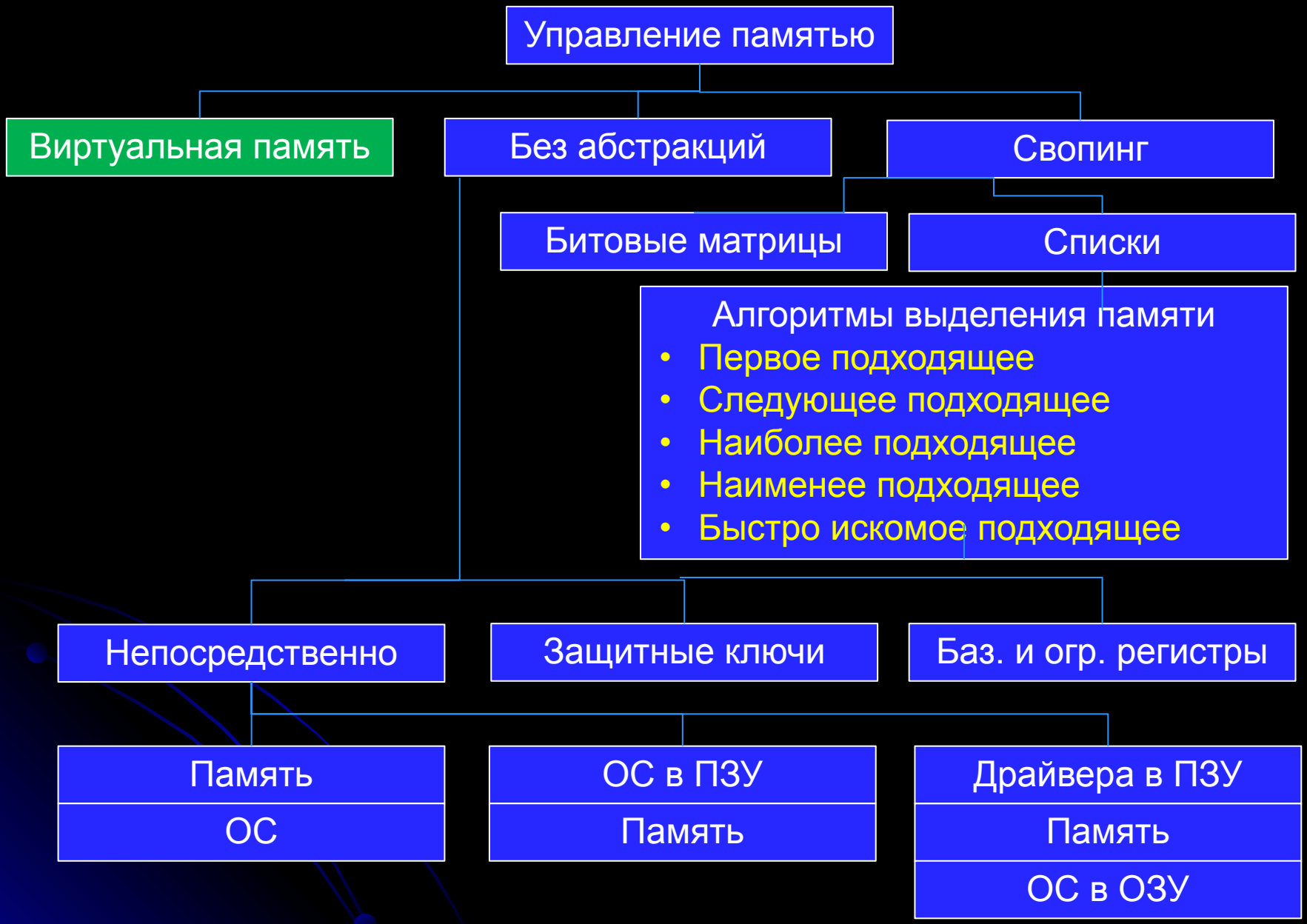
- Для одновременного размещения в памяти нескольких приложений без создания взаимных помех можно использовать абстракцию адресного пространства – модель оперативной памяти в которой программа существует изолированно.
- ***Адресное пространство*** – набор адресов, который может быть использован процессом для обращения к памяти. У каждого процесса имеется свое собственное адресное пространство, независимое от того адресного пространства, которое принадлежит другим процессам.
- Адресное пространство – универсальное понятие может использоваться в разных контекстах
  - Телефонные номера
  - Домашние адреса, квартиры
  - IP
  - …

Идея виртуальной памяти (1/2)

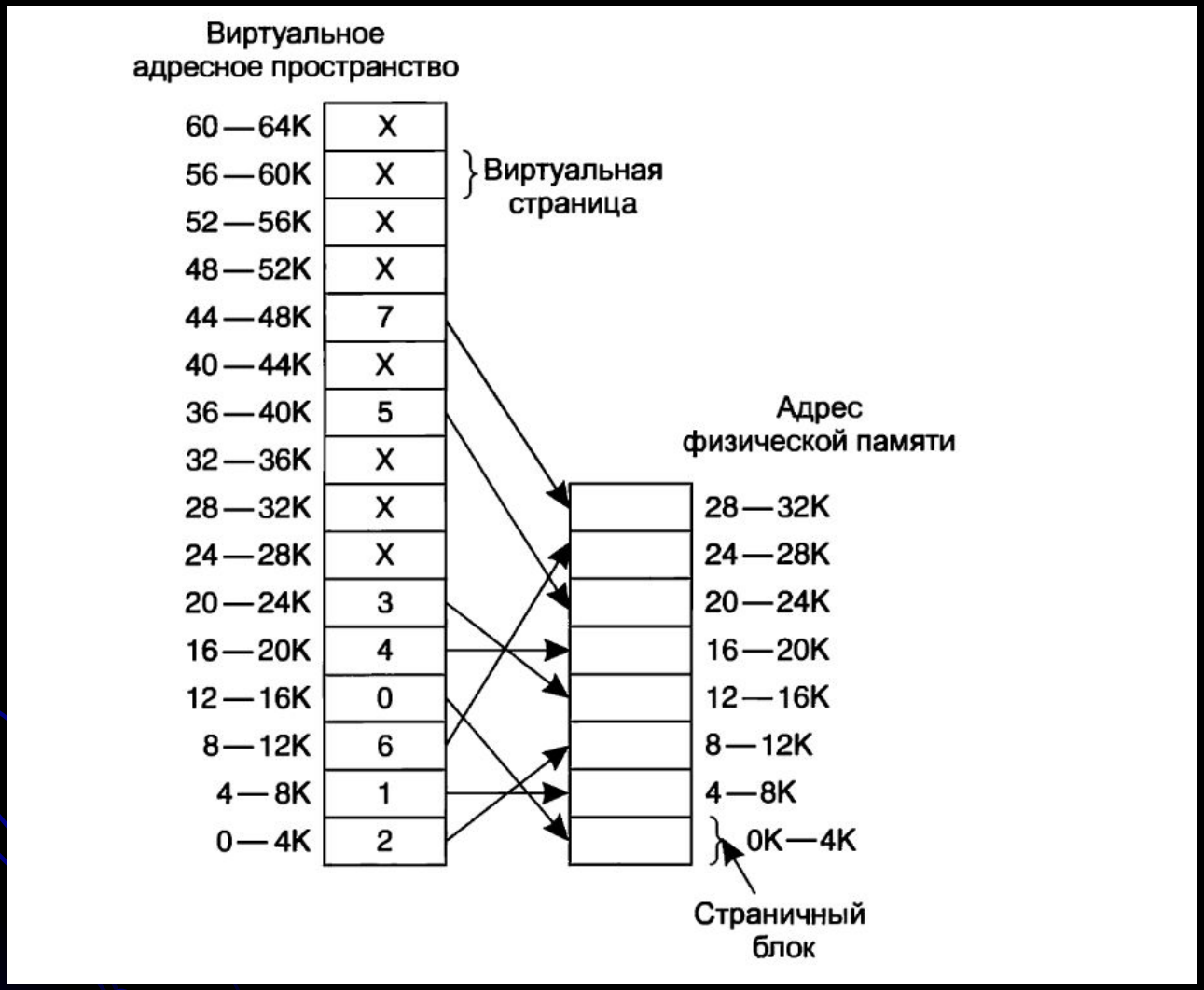
- У каждой программы есть свое собственное адресное пространство, которое разбивается на участки фиксированного размера, называемые страницами.
- Каждая страница – непрерывный диапазон адресов.
- Страницы отображаются на физическую память, но наличие всех страниц в памяти для запуска программы необязательно.
- Виртуальные адреса поступают в диспетчер (менеджер) памяти (MMU, Memory Management Unit) который отображает виртуальные адреса на адреса физической памяти.

Идея виртуальной памяти (2/2)

- Когда программа ссылается на часть своего адресного пространства, находящуюся в физической памяти, аппаратное обеспечение осуществляет отображение адреса «на лету».
- При отсутствии в памяти нужной страницы генерируется системное прерывание «page fault», ОС сбрасывает на диск редко используемый страничный блок и вместо него читает нужную страницу. После этого операция доступа повторяется.
- В многозадачных системах во время ожидания подгрузки нужной страницы ресурс центрального процессора может быть отдан другой программе.

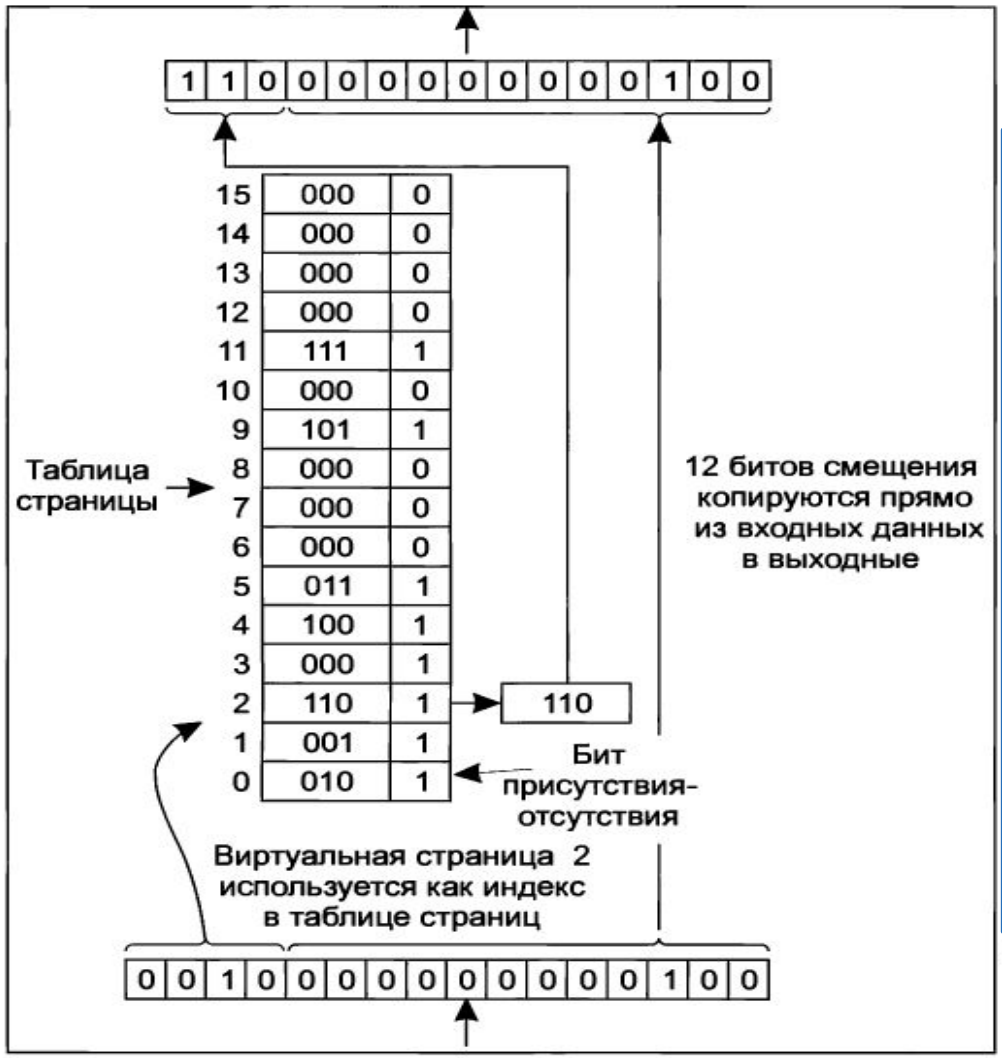

Управление памятью
Виртуальная память
Без абстракций
Свопинг
Битовые матрицы
Списки
Алгоритмы выделения памяти
• Первое подходящее
• Следующее подходящее
• Наиболее подходящее
• Наименее подходящее
• Быстро искомое подходящее
Непосредственно
Защитные ключи
Баз. и огр. регистры
Память
ОС
ОС в ПЗУ
Память
Драйвера в ПЗУ
Память
ОС в ОЗУ

Пример виртуальной памяти



Совместное использование страниц

Работа диспетчера памяти



Записи в таблице менеджера

1. Номер страничного блока
2. Бит присутствия/отсутствия
3. Бит защиты (разрешение записи, возможны более сложные значения)
4. Бит модификации – признак изменения страницы, если не было записи на страницу, то ее можно не сбрасывать на диск
5. Бит ссылки – признак наличия обращений по чтению к странице
6. Бит блокирования кэширования – предотвращает кэширование для фрагментов памяти, работающих с внешними устройствами.

Проблемы страничной организации памяти (1/3)

- Противоречивые условия:
  - Отображение виртуального адреса на физический должно быть быстрым (т.к. преобразование адресов происходит при каждом обращении к памяти)
  - При большом виртуальном пространстве таблица страниц будет иметь большой размер (например, при 32-х разрядной адресации и размере страниц 4 КБ получаем 1 млн. страниц)

Проблемы страничной организации памяти (2/3)

- Простейший подход 1:
  - Наличие аппаратных регистров по количеству страниц в адресном пространстве. При выполнении процесса его таблица грузится в эти регистры.
  - Преимущество – нет необходимости обращения к памяти при вычислении физического адреса.
  - Недостатки – большие затраты, необходимость загружать в регистры всю таблицу при переключении процессов.

Проблемы страничной организации памяти (3/3)

- Простейший подход 2:
  - Вся таблица находится в оперативной памяти. Требуется лишь один аппаратный регистр, указывающий на начало таблицы в памяти.
  - Достоинство – при переключении процессов необходимо загрузить всего один регистр.
  - Недостатки – очень медленно.

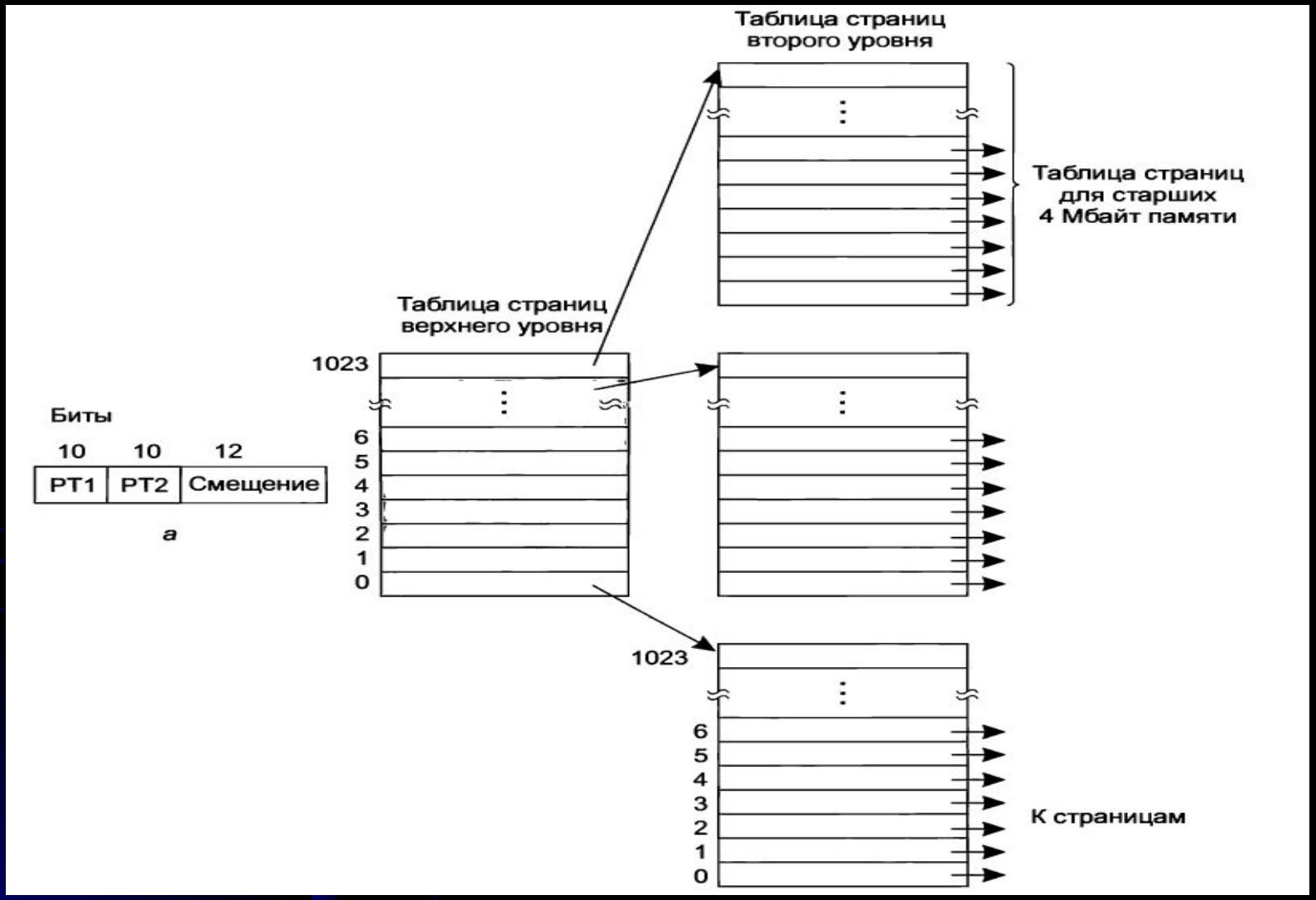
Буферы быстрого преобразования адреса

- Замечено, что программы обычно обращаются к небольшому количеству страниц. Т.е. работа производится в локальном объеме памяти.
- Интенсивному чтению подвергается лишь небольшая часть таблицы страниц. Остальные используются редко или вообще не используются.
- Для ускорения доступа компьютер оснащается небольшим устройством для отображения виртуальных адресов на физические без просмотра таблицы страниц – буфер быстрого преобразования адреса (TLB – Translation Lookaside Buffer) (или ассоциативная память).
- Реализуется аппаратно, содержит ~64 записей.
- Поля записи совпадают с полями в таблице страниц.
- Сначала проверяется наличие нужной страницы в TLB. Если нужная страница не найдена, производится поиск в таблице страниц и найденная запись переписывается в TLB.

Программное управление TLB

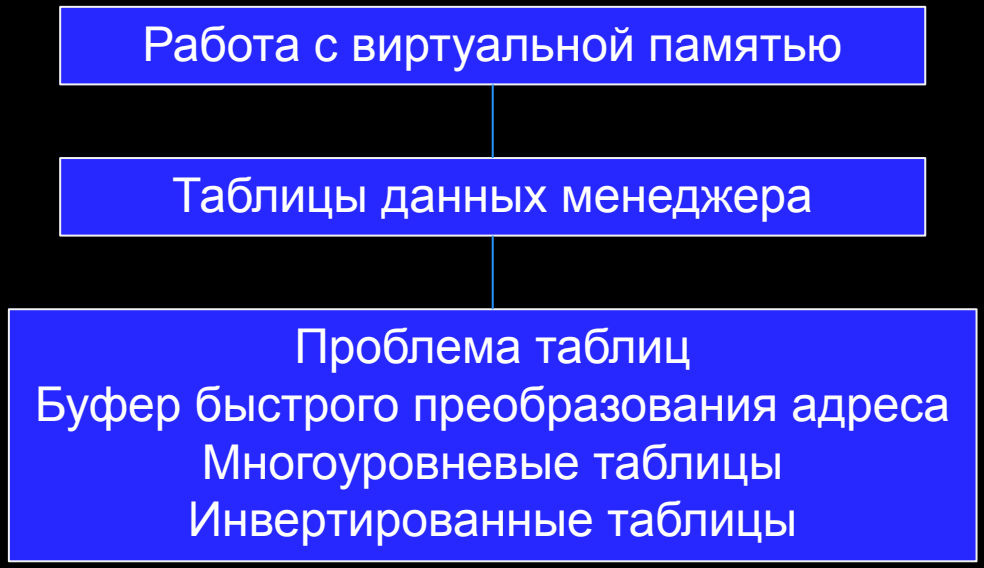
- Управление TLB может производиться как аппаратно, так и программно.
- При программном управлении, при отсутствии нужной записи в TLB генерируется прерывание и ОС должна сама решить возникшую проблему.
- Преимущества программного управления – более простое аппаратное устройство диспетчера памяти в процессоре. В чипе остается больше места для кэша и других узлов, позволяющих увеличить производительность. Возможно использование различных стратегий, позволяющих снизить количество случаев, когда нужной записи нет в TLB. Например, предсказание нужных страниц.
- Даже относительно небольшое количество записей делает программное управление достаточно эффективным, т.к. внесение новых записей в TLB происходит относительно редко.
- Следует различать различные случаи отсутствия записей
  - Запись отсутствует в TLB, но страница находится в оперативной памяти.
  - Страницы нет в оперативной памяти, требуется чтение с диска. Этот случай требует в миллион раз больше времени.

Многоуровневые таблицы - борьба с большим размером таблиц при большом виртуальном адресном пространстве



Инвертированные таблицы страниц

- Если использовать 64-х разрядное адресное пространство и страницы по 4 КБ, то потребуется таблица на 2^{52} записей. Или очень большое количество уровней в многоуровневых таблицах – снижение производительности.
- В инвертированной таблице страниц одна запись выделяется не каждой странице в виртуальном адресном пространстве, а каждому страничному блоку.
- Количество таких записей определяется размером физической памяти.
- Преобразование виртуальных адресов в физические становится намного сложнее, т.к. для поиска нужной страницы нужно просматривать весь список. Решение – использование TBL.


Работа с виртуальной памятью
Таблицы данных менеджера
Проблема таблиц
Буфер быстрого преобразования адреса
Многоуровневые таблицы
Инвертированные таблицы

Алгоритмы замещения страниц

- Если страницы нет в оперативной памяти, то ее нужно загрузить с жесткого диска.
- Обычно для этого нужно освободить место в памяти. Если удаляемая страница изменялась, то ее нужно записать на диск. Если не изменялась, то можно не тратить время на запись.
- Хотелось бы удалять из памяти самую «ненужную» страницу.
- Такая же проблема возникает при работе кэша.
- Для определения «ненужной» страницы разработано несколько алгоритмов

1- «Оптимальный» алгоритм замещения страниц

- Оптимальный алгоритм несложно описать, но совершенно невозможно реализовать:
  - Удалять ту страницу, обращение к которой состоится через самое большое время.
- Узнать, какая страница понадобится, невозможно.

2 - Алгоритм исключения недавно использовавшейся страницы

- Основывается на битах чтения и записи на страницу. Они обновляются при каждом обращении к памяти.
- Биты чтения сбрасываются регулярно по системному таймеру. Биты записи не сбрасываются, чтобы была возможность определить факт модификации страницы.
- Классы страниц:
  - 0 – в последнее время не было обращений и модификаций
  - 1 – обращений не было, но страница модифицирована
  - 2 – были обращения, но не было модификации
  - 3 – были обращения и модификации
- Алгоритм удаляет страницу, относящуюся к самому низкому непустому классу.
- Лучше удалить модифицированную, но редко используемую страницу, чем часто используемую.
- Достоинство – простота, приемлемые результаты.

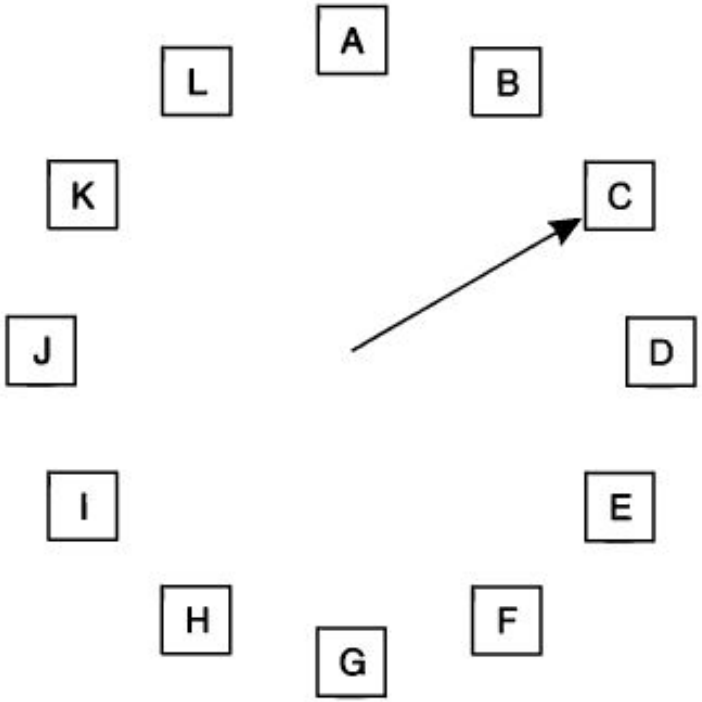
3 - Алгоритм «FIFO»

- Ведется список страниц поступивших в систему в порядке загрузки.
- Удаляется страница, находящаяся в голове списка.
- Может привести к удалению нужной страницы. В чистом виде используется очень редко.

4 - Алгоритм «Второй шанс»

- Алгоритм FIFO, дополненный проверкой бита чтения самой старой страницы.
- Если к самой старой странице было обращение, она перемещается в хвост очереди и бит чтения сбрасывается.
- Ищется самая старая страница, к которой не было обращений.
- Недостаток – низкая эффективность, связанная с необходимостью перемещения данных в списке.

5 - Алгоритм «Часы» - используется циклический список вместо линейного



Когда происходит страничное прерывание, поверяется страница, на которую указывает стрелка. Предпринимаемые действия зависят от бита R:

R=0: страница выгружается

R=1: бит R сбрасывается, стрелка движется вперед

6 - Алгоритм «замещения наименее востребованной страницы» – LRU (Least Recently Used)

- Страницы, наиболее активно используемые последними командами скорее всего будут использованы и следующими командами.
- Нужно избавиться от страницы, которая не требовалась самое длительное время.
- Стратегия эффективная, но сложна в реализации.
- Решение – аппаратная реализация.
- Вариант 1:
  - Используется аппаратный счетчик команд
  - При обращении к странице в записи таблицы страниц сохраняется значение счетчика.
  - При удалении ищется страница с наименьшим значением.
- Вариант 2:
  - Если имеется N страничных блоков, то заводится битовая матрица NxN.
  - При обращении к странице k в таблице строка k заполняется 1, а столбец k – обнуляется.
  - Самой старой будет страница с наименьшим значением в строке.
- Проблема – сложная реализация, требует существенной аппаратной поддержки.

7 - Алгоритм «Нечастого востребования» - NFU (Not Frequently Used)

- Алгоритм LRU требует значительных аппаратных ресурсов. Существует вариант его программного решения.
- Требуется набор программных счетчиков, связанных со страницами.
- При каждом прерывании от таймера система к каждому счетчику прибавляет значение бита контроля доступа.
- При возникновении ошибки страницы для замещения выбирается страница, чей счетчик имеет минимальное значение.
- Проблема – алгоритм ничего не забывает. Если когда-то станица активно использовалась, то информация об этом не сбрасывается.
- Из-за этого могут вытесняться страницы только что загруженные в память, т.к. старые страницы будут иметь больше обращений.

8 - Алгоритм «старения»

- Вариант программной реализации алгоритма замещения наименее востребованной страницы.
- Каждой странице ставится в соответствие битовое поле.
- При каждом срабатывании системного таймера битовое поле сдвигается на один разряд вправо.
- Если к странице были обращения, то в левую позицию записывается 1, если нет – 0.
- Удаляется страница с меньшим значением в поле.
- Горизонт анализа зависит от разрядности битового поля.

Сравнение алгоритмов замещения страниц

Алгоритм	Особенности
Оптимальный	Не может быть реализован, но полезен в качестве оценочного критерия
NRU (Not Recently Used) — алгоритм исключения недавно использовавшейся страницы	Является довольно грубым приближением к алгоритму LRU
FIFO (First-In, First-Out) — алгоритм «первой пришла, первой и ушла»	Может выгрузить важные страницы
Алгоритм «второй шанс»	Является существенным усовершенствованием алгоритма FIFO
Алгоритм «часы»	Вполне реализуемый алгоритм
LRU (Least Recently Used) — алгоритм замещения наименее востребованной страницы	Очень хороший, но труднореализуемый во всех тонкостях алгоритм
NFU (Not Frequently Used) — алгоритм нечастого востребования	Является довольно грубым приближением к алгоритму LRU
Алгоритм старения	Вполне эффективный алгоритм, являющийся неплохим приближением к алгоритму LRU

Работа с виртуальной памятью

Таблицы данных менеджера

Проблема таблиц
Буфер быстрого преобразования адреса
Многоуровневые таблицы
Инвертированные таблицы

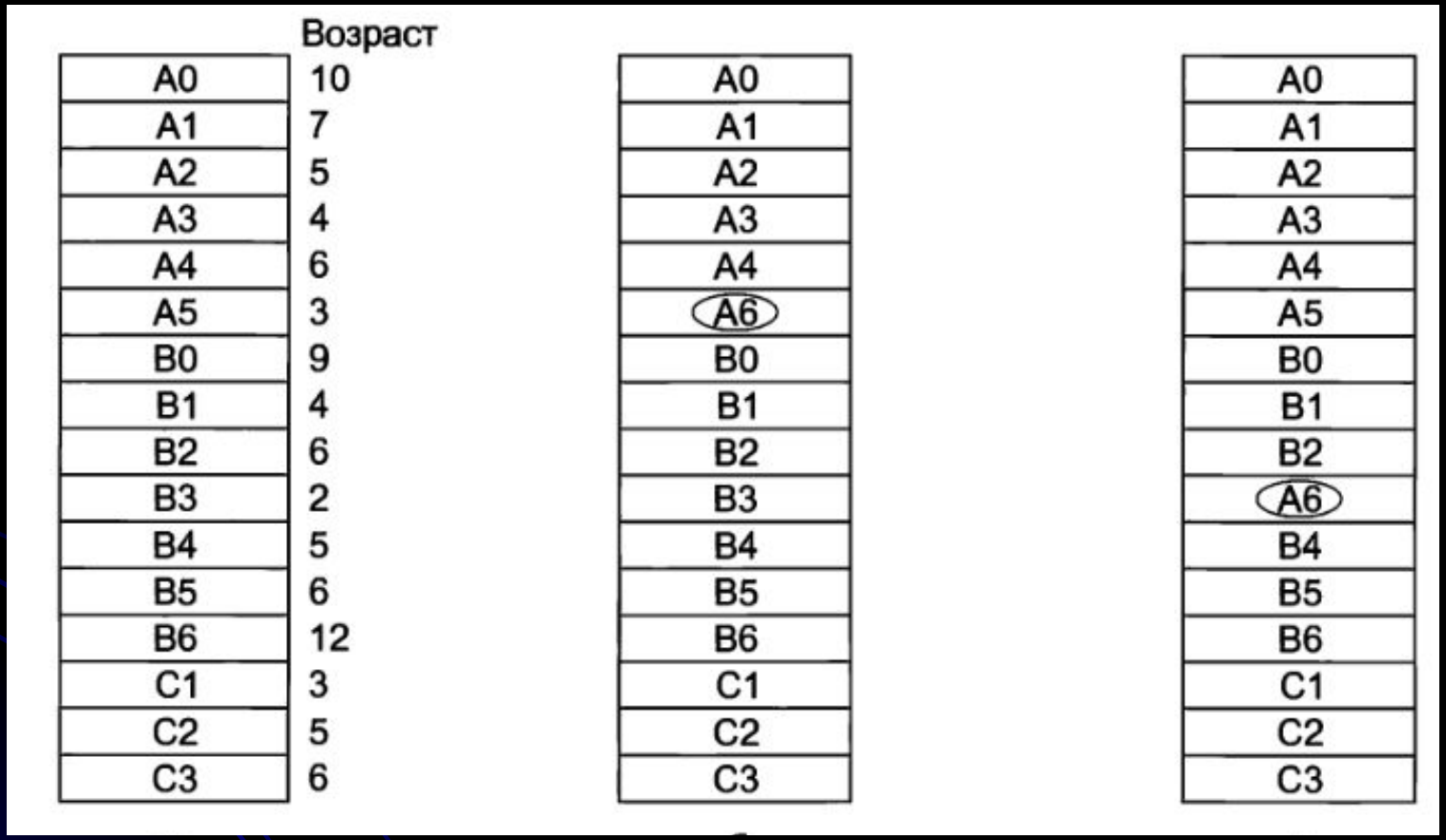
Освобождение страниц в памяти

1. «Оптимальный» алгоритм замещения страниц
2. Алгоритм исключения недавно использовавшейся страницы
3. Алгоритм «FIFO»
4. Алгоритм «Второй шанс»
5. Алгоритм «Часы»
6. Алгоритм «LRU»
7. Алгоритм «NFU»
8. Алгоритм «старения»

Алгоритм «Рабочий набор»

- В многозадачной среде операционная система отслеживает набор страниц, используемых процессом в данное время (рабочий набор).
- Перед возобновлением работы процесса обеспечивается наличие в памяти рабочего набора.
- Такой подход называется моделью рабочего набора.
- Загрузка страниц до того, как процессу будет предоставлено процессорное время называется опережающей подкачкой страниц
- Необходимо точное определение того, какие страницы относятся к рабочему набору.

Локальный и глобальный алгоритмы замещения страниц (1/2)



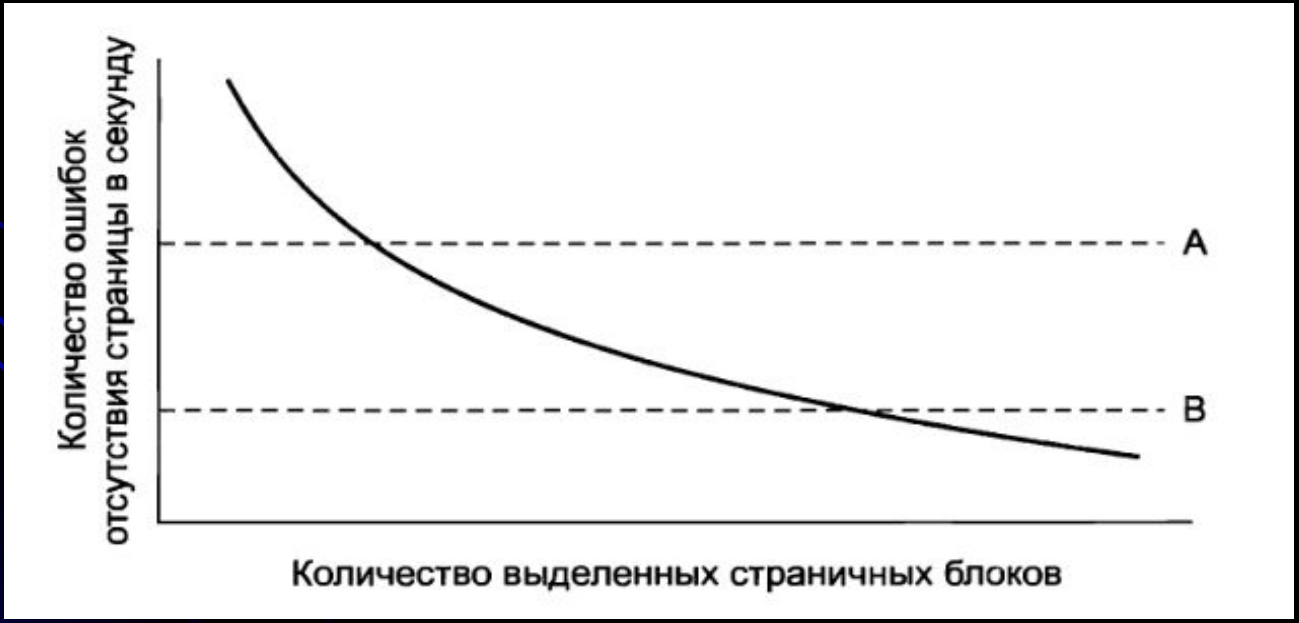
	Возраст		
A0	10	A0	A0
A1	7	A1	A1
A2	5	A2	A2
A3	4	A3	A3
A4	6	A4	A4
A5	3	A6	A5
B0	9	B0	B0
B1	4	B1	B1
B2	6	B2	B2
B3	2	B3	A6
B4	5	B4	B4
B5	6	B5	B5
B6	12	B6	B6
C1	3	C1	C1
C2	5	C2	C2
C3	6	C3	C3

Локальный и глобальный алгоритмы замещения страниц (2/2)

- Локальный алгоритм подходит для выделения процессам фиксированной доли памяти.
- В целом глобальные алгоритмы работают лучше.

Алгоритм PFF

- Алгоритм PFF (Page Fault Count) частоты возникновения ошибки отсутствия страницы.
- Подсказывает, когда нужно увеличивать или уменьшать количество выделенных процессу страниц, но не говорит о том какую страницу нужно удалять.
- Чем больше процессу выделено страниц, тем реже возникают ошибки.
- Количество ошибок измеряется раз в секунду или методом скользящего среднего.



Управление загрузкой

- Если объем всех рабочих наборов активных процессов больше объема физической памяти возможна пробуксовка процессов – процессор свободен, ожидает подгрузки страниц.
- Один из симптомов – показания алгоритма PFF, что некоторые процессы нуждаются в оперативной памяти, но нет процессов готовых освободить память.
- Решение проблемы – выгрузка какого-нибудь процесса целиком.
- Т.е. при страничной организации свопинг не утрачивает актуальности.
- Еще один фактор – степень многозадачности. Нужно при небольшом наборе задач процессор может быть недогружен, ожидая подгрузку страниц.

Работа с виртуальной памятью

Таблицы данных менеджера

Проблема таблиц
Буфер быстрого преобразования адреса
Многоуровневые таблицы
Инвертированные таблицы

Освобождение страниц в памяти

1. «Оптимальный» алгоритм замещения страниц
2. Алгоритм исключения недавно использовавшейся страницы
3. Алгоритм «FIFO»
4. Алгоритм «Второй шанс»
5. Алгоритм «Часы»
6. Алгоритм «LRU»
7. Алгоритм «NFU»
8. Алгоритм «старения»

Работа с несколькими процессами

Алгоритм «Рабочий набор»
Локальный и глобальный алгоритмы
Алгоритм PFF
Управление загрузкой

Размер страниц

- Абсолютного решения не существует
- Требуется сохранение баланса между несколькими конкурирующими факторами
  - Проблема внутренней фрагментации (соображение в пользу небольших страниц)
  - Проблема маленьких программ или программ не требующих одномоментно больших объемов данных (соображение в пользу небольших страниц)
  - При маленьких страницах требуются большие таблицы страниц. (соображение в пользу больших страниц)

Политика очистки страниц

- Замещение страниц лучше всего работает при достаточном количестве свободных страничных блоков, которые могут потребоваться при возникновении ошибки отсутствия страницы.
- Для своевременной поставки страниц используется «страничный демон» - фоновый процесс, который контролирует количество свободных страничных блоков.
- Если свободных страничных блоков слишком мало, страничный демон подбирает страницы для выгрузки и при необходимости записывает их на диск.
- Предыдущее состояние страницы запоминается. Если потребовалась сброшенная страница. То она восстанавливается.
- Такой подход способствует равномерности распределения нагрузки на процессор и контроллер DMA.

Работа с виртуальной памятью

Таблицы данных менеджера

Проблема таблиц
Буфер быстрого преобразования адреса
Многоуровневые таблицы
Инвертированные таблицы

Выбор размера страницы

Страничный демон

Освобождение страниц в памяти

1. «Оптимальный» алгоритм замещения страниц
2. Алгоритм исключения недавно использовавшейся страницы
3. Алгоритм «FIFO»
4. Алгоритм «Второй шанс»
5. Алгоритм «Часы»
6. Алгоритм «LRU»
7. Алгоритм «NFU»
8. Алгоритм «старения»

Работа с несколькими процессами

Понятие «Рабочий набор»
Локальный и глобальный алгоритмы
Алгоритм PFF
Управление загрузкой

Кэширование (1/3)

- Кэш-память – кэш (cache) – способ совместного функционирования двух типов запоминающих устройств, отличающихся временем доступа и стоимостью хранения данных, за счет динамического копирование в быстрое ЗУ (запоминающее устройство, собственно кэш-память) наиболее часто используемой информации из медленного ЗУ (основная память).
- Идея кэширования данных базируется на присущих данным свойствам временной и пространственной локальности.

- **Принцип временной локальности** состоит в том, что во время считывания любых данных из памяти существует высокая вероятность обращения программы на протяжении некоторого небольшого промежутка времени снова к ним.
- **Принцип пространственной локальности** основывается на высокой вероятности того, что программа через некоторый небольшой промежуток времени обратится в ячейку памяти, следующей за той, к которой она обращалась перед этим.

Кэширование (2/3)

- Свойства
  - прозрачность для программ и пользователей;
  - отсутствие внешней информации об интенсивности использования данных;
  - перемещение данных в быстрое ЗУ из медленного ЗУ осуществляется только средствами поддержки кэша.
- Кэшированию может подвергаться обращение в ОП и во внешнюю память.
- Каждая запись об элементе данных в кэше включает:
  - значение элемента данных;
  - адрес, занимаемый элементом в основной памяти;
  - дополнительную информацию: признак модификации / признак действительности данных.

Кэширование (3/3)

- Кэш-память не является адресуемой, поэтому поиск данных осуществляется по запрашиваемому адресу. Возможны варианты:
  - если данные в кэш-памяти – кэш-попадание (cache-hit) – чтение данных и возврат;
  - если данные отсутствуют в кэш-памяти – кэш-промах (cache-miss) – чтение из основной памяти и возврат с одновременным копированием в кэш.
- Содержимое кэш-памяти постоянно обновляется – данные в кэше вытесняются. Вытесняемые данные
  - либо могут быть измененными относительно их оригинала в основной памяти,
  - либо неизмененными.
- Наличие в компьютере двух копий данных – в основной памяти и в кэше – порождает проблему согласования этих копий данных при модификации. Два решения:
  - сквозная запись (write through). При каждой записи в основную память – модификация только этой памяти (если нет кэш-копии) или модификация еще и кэш-копии;
  - обратная запись (write back). При каждой записи в основную память – модификация только этой памяти (если нет кэш-копии) или модификация только кэш-копии. При удалении кэш-копии из кэша, изменения записываются в основную память (например, в фоновом режиме).

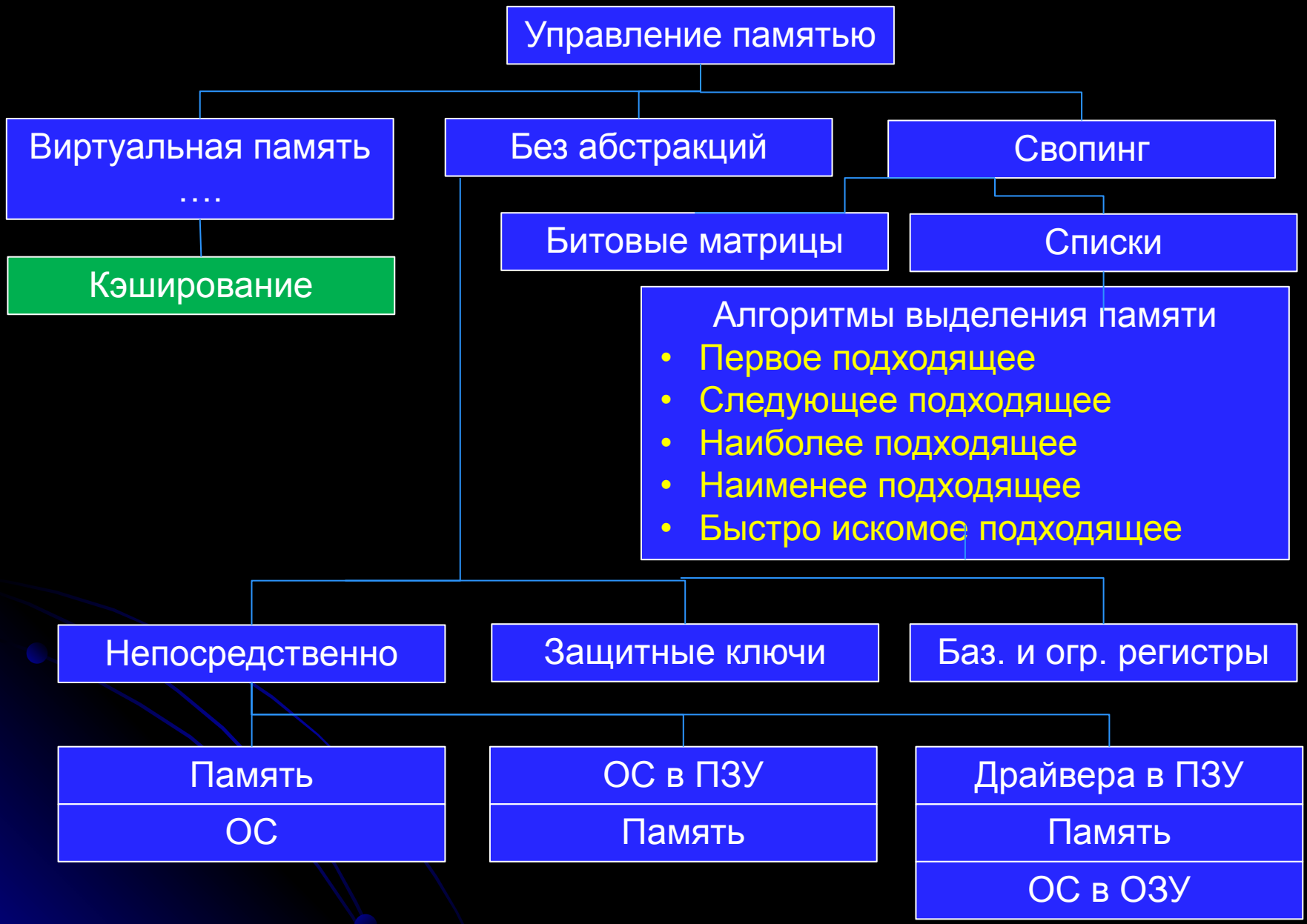
Двухуровневое кэширование

Двухуровневое кэширование

- Кэш первого уровня – кэширует кэш второго уровня. Если промах – поиск в кэше второго уровня.
- Кэш второго уровня – кэширует ОП. Если промах – чтение данных из ОП.
- Для согласования содержимого кэшей и ОП используются различные схемы: 1-й кэш – сквозная запись, 2-й кэш – обратная запись

Кэширование в современных системах

Вид	Предмет кэширования	Где осуществляется кэширование	Задержка(циклов)	Управляется
Регистры CPU	4-байтовое слово	Внутренние регистры CPU	0	Компилятор
TLB	Преобразования	Внутренний TLB	0	Аппаратный адресов MMU
Кэш уровня L1	32-байтовый блок	Внутренний кэш уровня L1	1	Аппаратные средства
Кэш уровня L2	32-байтовый блок	Внешний кэш уровня L2	10	Аппаратные средства
Виртуальная память	Страница 4-Кбайт	Основная память	100	Аппаратные средства + операционная система
Буферная кэш-память	Части файлов	Основная память	100	Операционная система
Сетевая буферная кэш-память	Части файлов	Локальный диск	10 000 000	AFS/NFS-клиент
Кэш-память браузера	Web-страницы	Локальный диск	10 000 000	Web-браузер
Web-кэш	Web-страницы	Диски удаленного сервера	1 000 000 000	Web-сервер прокси


Управление памятью
Виртуальная память
....
Кэширование
Без абстракций
Свопинг
Битовые матрицы
Списки
Алгоритмы выделения памяти
• Первое подходящее
• Следующее подходящее
• Наиболее подходящее
• Наименее подходящее
• Быстро искомое подходящее
Непосредственно
Защитные ключи
Баз. и огр. регистры
Память
ОС
ОС в ПЗУ
Память
Драйвера в ПЗУ
Память
ОС в ОЗУ

Вопросы к зачету (1/2)

- Иерархия памяти. Менеджер памяти.
- Память без использования абстракций.
  - Варианты использования памяти.
  - Использование защитных ключей.
  - Использование базового и ограничительного регистров.
- Свопинг и виртуальная память.
- Свопинг.
  - Проблемы свопинга.
  - Управление свободной памятью.
  - Использование битовых матриц.
  - Использование связанных списков.
  - Алгоритмы выделения памяти.
- Понятие адресного пространства.
- Виртуальная память.
  - Работа диспетчера памяти. Совместное использование страниц.
  - Структура записи в таблице страниц. Проблемы страничной организации памяти.
  - Буферы быстрого преобразования адреса. Программное управление TLB.

Задания к зачету (2/2)

- Проблемы таблиц менеджера.
  - Многоуровневые таблицы.
  - Инвертированные таблицы страниц.
  - Алгоритмы замещения страниц.
- Освобождение страниц в памяти
- Оптимальный алгоритм замещения страниц.
  - Алгоритм исключения недавно использовавшейся страницы.
  - Алгоритм FIFO.
  - Алгоритм «Второй шанс».
  - Алгоритм «Часы».
  - Алгоритм LRU (Least Recently Used).
  - Алгоритм NFU (Not Frequently Used).
  - Алгоритм старения.
  - Сравнение алгоритмов
- Работа с несколькими процессами
  - Понятие «Рабочий набор».
  - Локальный и глобальный алгоритмы замещения страниц.
  - Алгоритм PFF.
  - Управление загрузкой.
- Размер страниц.
- Политика очистки страниц.
- Кэширование.
  - Двухуровневое кэширование.
  - Кэширование в современных системах.