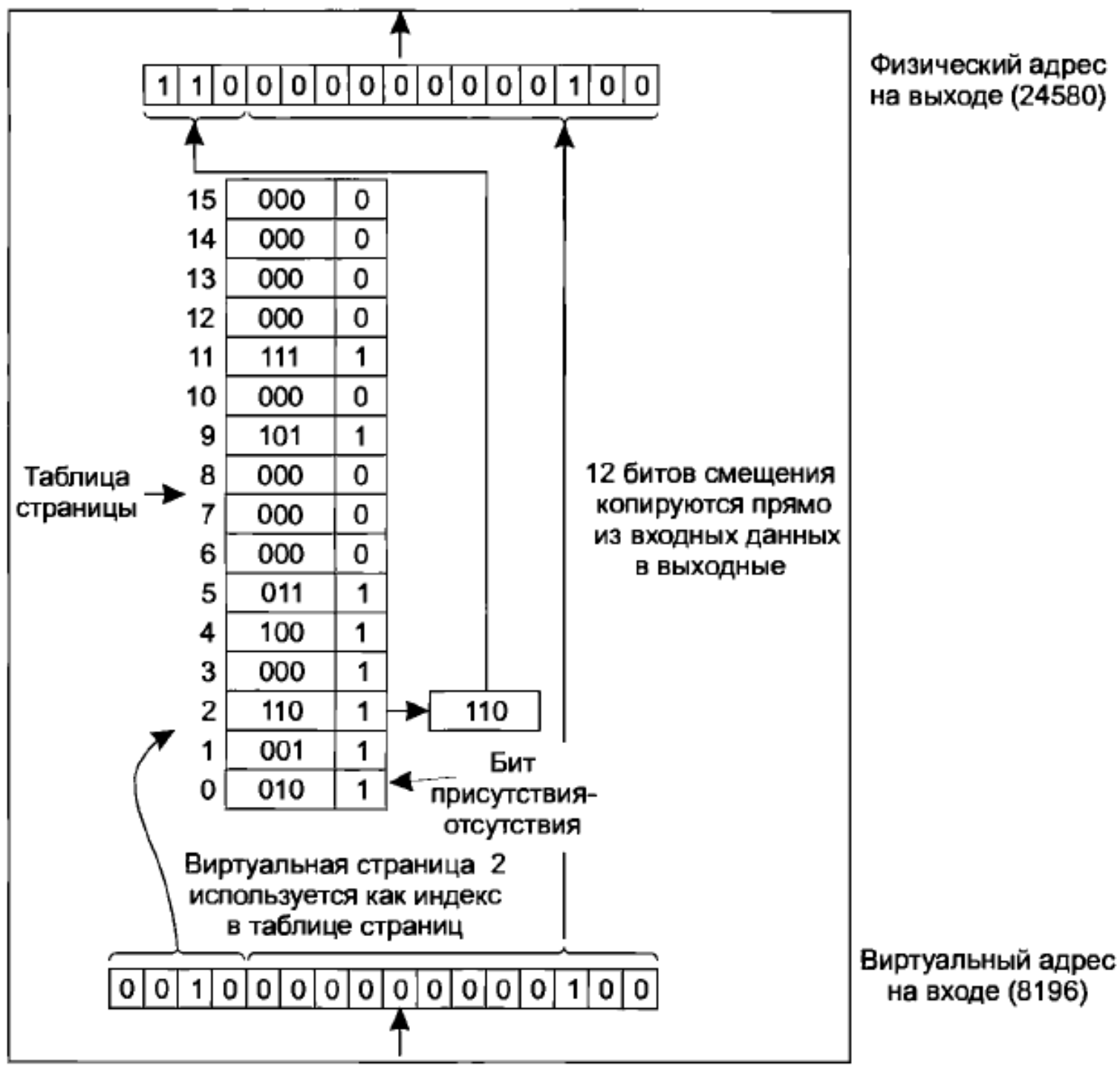
Операционные системы

Работа с памятью. Виртуальная память

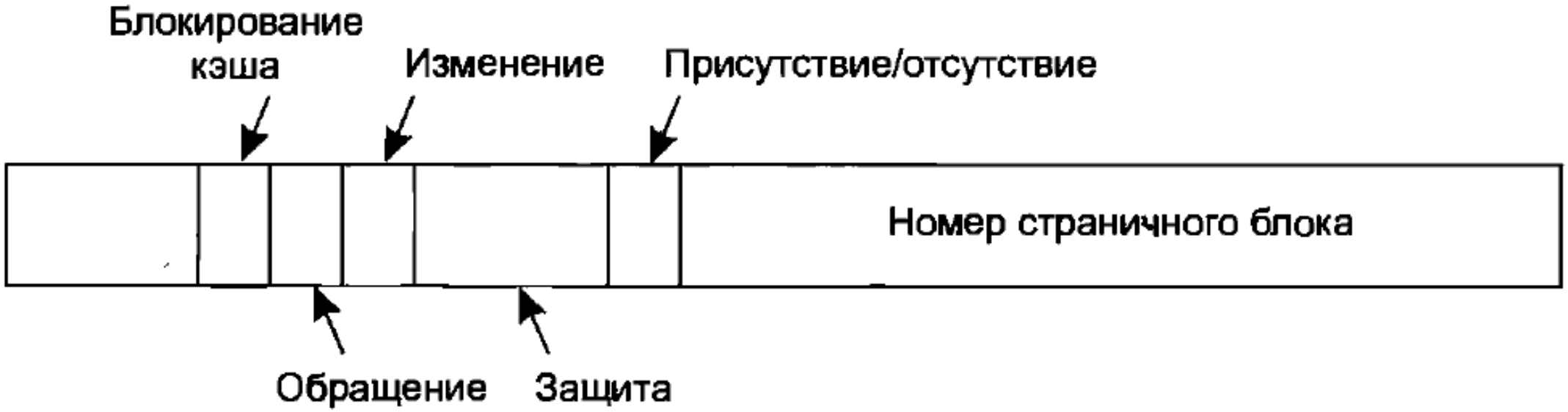
Виртуальная память

1. Требуемый объем памяти превышает физический объем памяти
2. Нужно разграничивать память между процессами
3. Виртуальная память бьется на блоки, называемые страницами
4. Страницы имеют размер кратный 2-ке
5. Виртуальные адреса попадают не на шину, а в MMU


Виртуальное адресное пространство
60—64K
56—60K
52—56K
48—52K
44—48K
40—44K
36—40K
32—36K
28—32K
24—28K
20—24K
16—20K
12—16K
8—12K
4—8K
0—4K
X
X
X
X
7
X
5
X
X
X
3
4
0
6
1
2
Виртуальная страница
Адрес физической памяти
28—32K
24—28K
20—24K
16—20K
12—16K
8—12K
4—8K
0K—4K
Страничный блок


1 1 0 0 0 0 0 0 0 0 0 0 1 0 0
Физический адрес на выходе (24580)
15 000 0
14 000 0
13 000 0
12 000 0
11 111 1
10 000 0
9 101 1
8 000 0
7 000 0
6 000 0
5 011 1
4 100 1
3 000 1
2 110 1
1 001 1
0 010 1
110
Таблица страницы
12 битов смещения копируются прямо из входных данных в выходные
Бит присутствия-отсутствия
Виртуальная страница 2 используется как индекс в таблице страниц
0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0
Виртуальный адрес на входе (8196)

Структура записи в таблице страниц



Сложности при страничной организации памяти

1. Отображение виртуального адреса на физический должно быть быстрым
2. При увеличении пространства виртуальных адресов размер таблицы страниц становится довольно большим

TLB(Transaction Lookaside Buffer) – буфер быстрого преобразования адреса

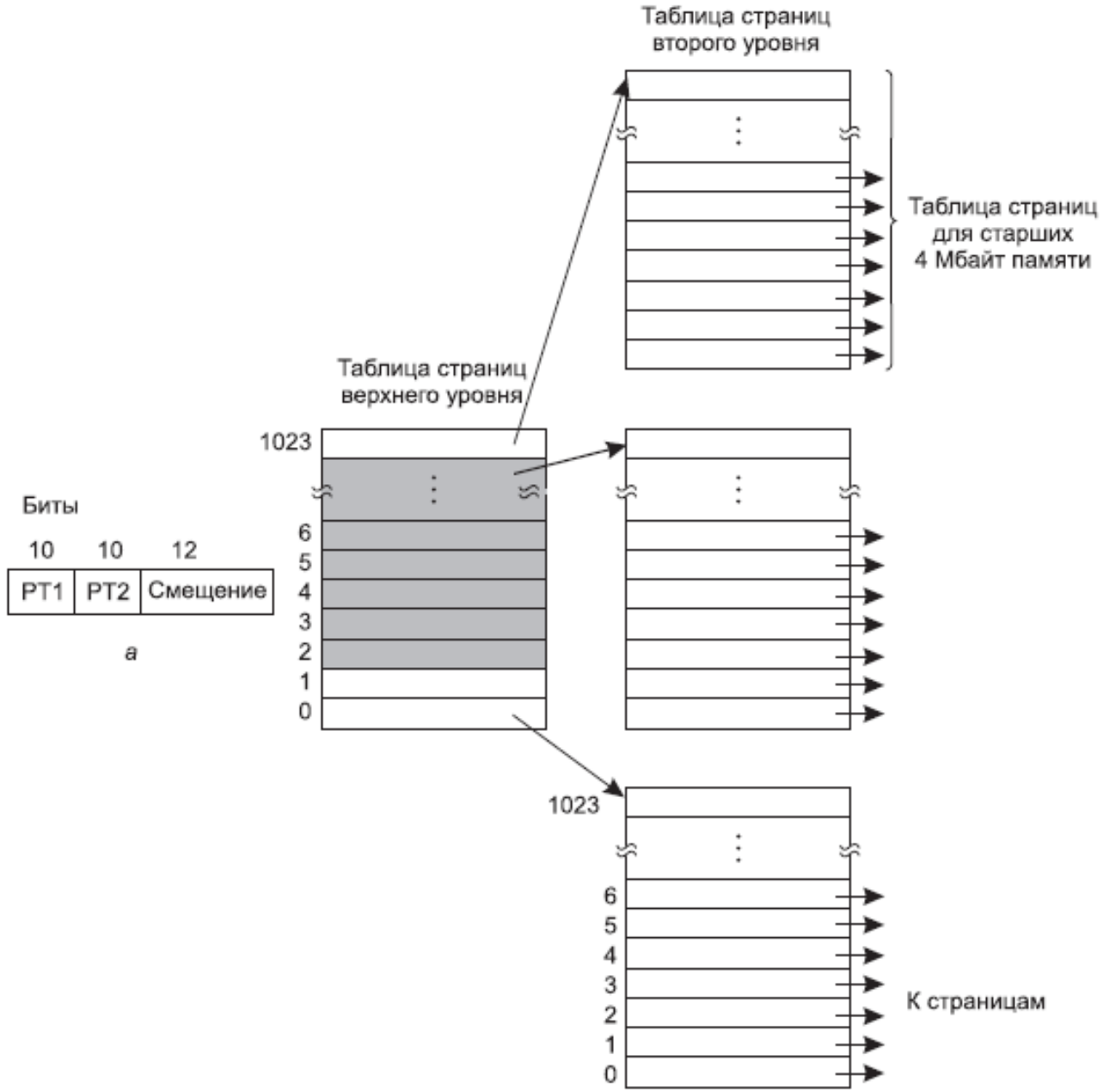
1. Быстрое преобразование виртуального адреса
2. Программная и аппаратная ошибки отсутствия страниц

Задействована	Виртуальная страница	Изменена	Защищена	Страничный блок
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

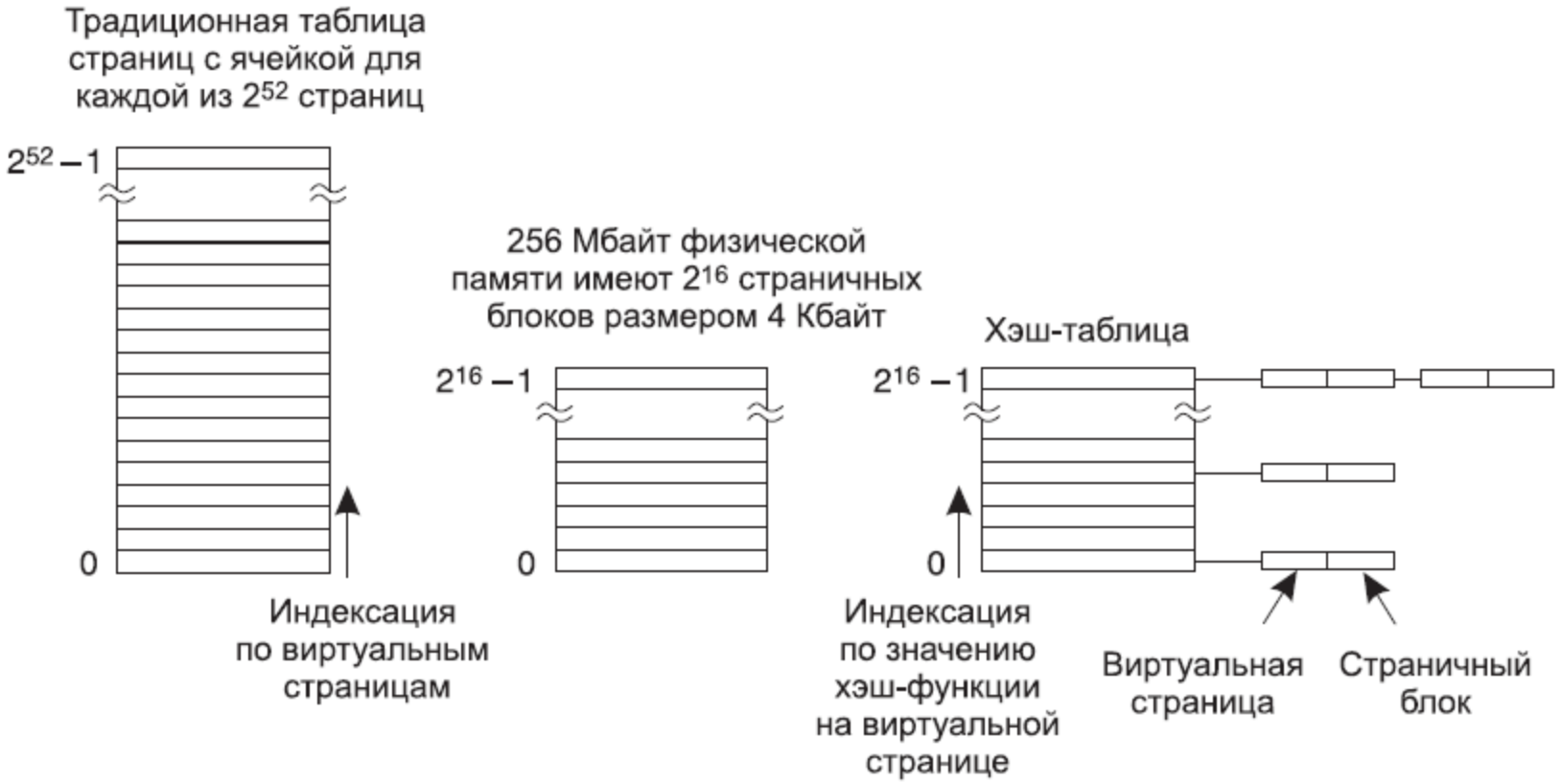
Подходы к решению проблем с увеличением объемов виртуальной памяти

1. Многоуровневые таблицы страниц
2. Инвертированные таблицы страниц

Многоуровневые таблицы страниц



Инвертированные таблицы страниц



Алгоритмы замещения страниц

1. Оптимальный
2. Алгоритм исключения недавно использовавшейся страницы
3. Алгоритм FIFO
4. Алгоритм «Второй шанс»
5. Алгоритм «часы»
6. Алгоритм замещения наименее востребованной страницы
7. Алгоритм «Рабочий набор»
8. Алгоритм WSClock

Оптимальный

1. Каждая загруженная страница помечается числом, обозначающим через сколько команд к ней будет обращение
2. Удаляется страница имеющая наибольшее число
3. Прост в понимании, невозможен в реализации
4. Бывает нужен для получения наилучшей замены страниц при тестировании реальных алгоритмов

Исключение давно используемой страницы (NRU)

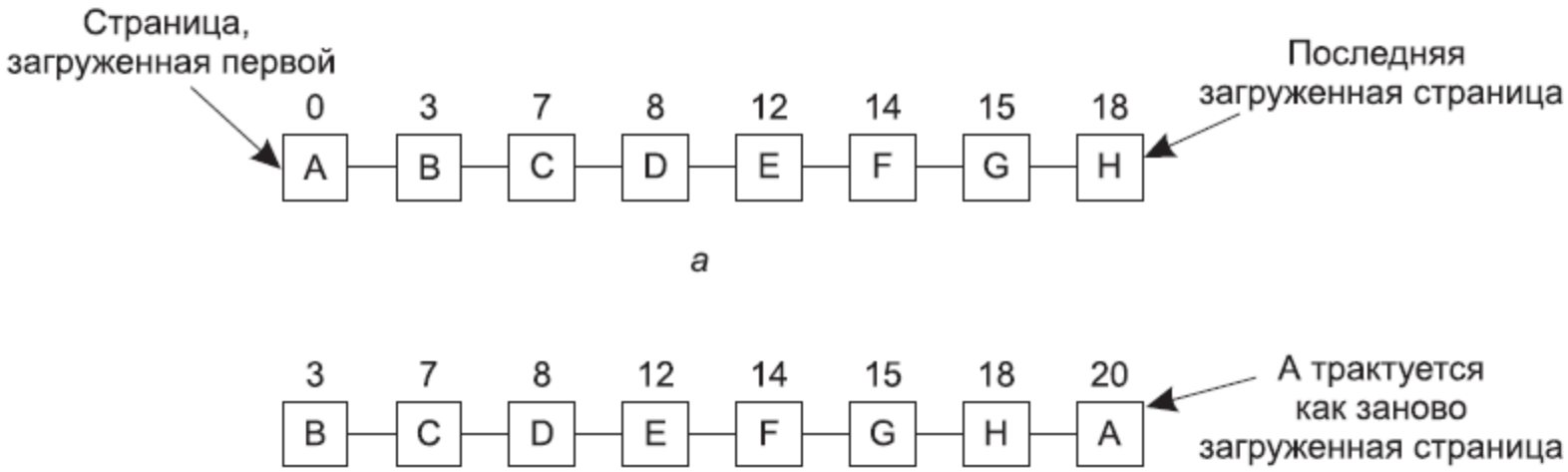
1. Алгоритм основан на битах чтения (R) и изменения (M)
2. По таймеру бит R сбрасывается в ноль
3. Каждая страница относится к одному из классов:
   a) R:0 M:0
   b) R:0 M:1
   c) R:1 M:0
   d) R:1 M:1
4. Пытаемся удалить страницы в порядке a->b->c->d

Алгоритм FIFO

1. Имеет много вырожденных случаев
2. В чистом виде практически не используется

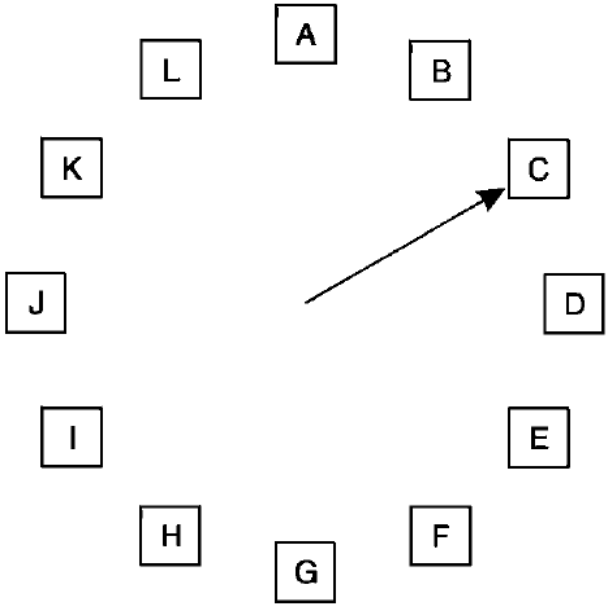
«Второй шанс»

1. Модификация FIFO
2. При вытеснении страницы производят следующие действия:
   a) Если бит R установлен в 0, то просто вытесняем
   b) Если бит R установлен в 1, то устанавливаем его в 0, а страницу перемещаем в конец очереди и пытаемся снова вытеснить «голову» очереди



«Часы»

1. Модификация алгоритма «Второй шанс»
2. Аналогично «Второму шансу», но вместо очереди используем закольцованный список и ходим по нему



Замещение наименее востребованной страницы (LRU)

1. Ведем 64-разрядный счетчик команд C
2. После каждого обращения к странице записываем значение C в таблицу страниц для этой записи, относящейся к этой странице
3. Замещаем страницу с наименьшим значением этого поля
4. Сложности в реализации из-за величины счетчика

Страница

	0	1	2	3
0	0	1	1	1
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

а

Страница

0	1	2	3
0	0	1	1
1	0	1	1
0	0	0	0
0	0	0	0

б

Страница

0	1	2	3
0	0	0	1
1	0	0	1
1	1	0	1
0	0	0	0

в

Страница

0	1	2	3
0	0	0	0
1	0	0	0
1	1	0	0
1	1	1	0

г

Страница

0	1	2	3
0	0	0	0
1	0	0	0
1	1	0	1
1	1	0	0

д

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

е

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

ж

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

з

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

и

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

к

Нечастого востребования (NFU), как приближение LRU

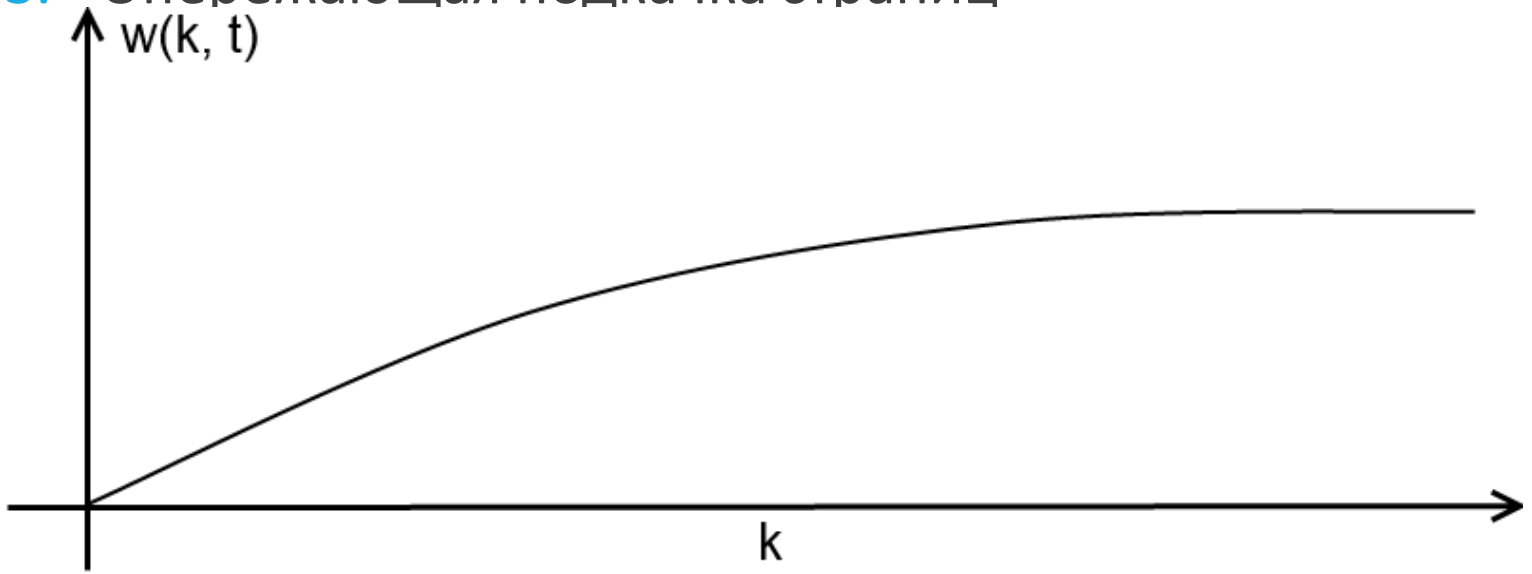
1. При каждой странице ведем счетчик
2. При каждом прерывание таймера прибавляем к счетчику значение R
3. Замещается страница с наименьшим значением счетчика

Алгоритм старения, улучшение NFU

	Биты R для страниц 0–5, такт 0	Биты R для страниц 0–5, такт 1	Биты R для страниц 0—5, такт 2	Биты R для страниц 0–5, такт 3	Биты R для страниц 0–5, такт 4
	1 0 1 0 1 1	1 1 0 0 1 0	1 1 0 1 0 1	1 0 0 0 1 0	0 1 1 0 0 0
Страница					
0	10000000	11000000	11100000	11110000	01111000
1	00000000	10000000	11000000	01100000	10110000
2	10000000	01000000	00100000	00100000	10001000
3	00000000	00000000	10000000	01000000	00100000
4	10000000	11000000	01100000	10110000	01011000
5	10000000	01000000	10100000	01010000	00101000

«Рабочий набор»

1. Загрузка страниц только по мере надобности
2. W(k, t) – рабочий набор, k – количество последних обращений к памяти, t – время
3. Опережающая подкачка страниц



«Рабочий набор». Реализация

1. В качестве времени берется виртуальное время
2. Вводим время устаревания (τ) и время такта (за время устаревания должно проходить несколько тактов)
3. После каждого такта, страницы имеющие R:1 устанавливают обновленное время
4. При возникновении ошибки отсутствия страницы происходит очистка таблицы страниц
   a) R:1 – не трогаем страницу
   b) R:0 и возраст меньше τ – не трогаем страницу
   c) R:0 и возраст больше τ – выгружаем страницу

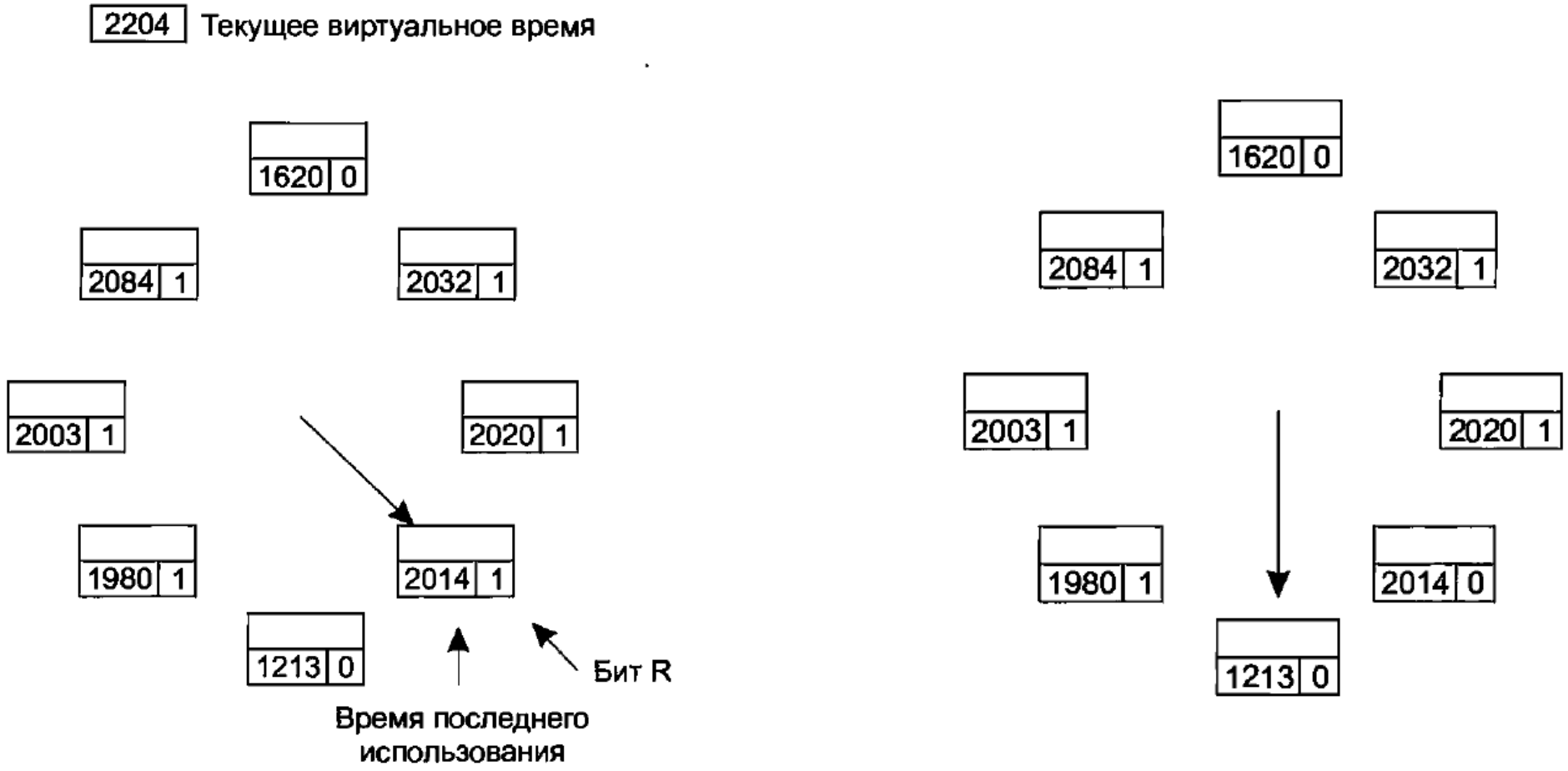
«Рабочий набор». Анализ

1. Поддерживается довольно точно актуальный набор страниц в общих случаях
2. Трудоемкий процесс очистки

WSClock

1. Совмещение алгоритма «Рабочий набор» и «Часы»
2. Время последнего использования берется из алгоритма «Рабочий набор»
3. При ошибке отсутствия страницы:
   a) Если R:1, то устанавливаем R:0 и стрелка перемещается дальше
   b) Если R:0, W:1 и возраст > τ, то планируем запись на диск и перемещаем стрелку далее
   c) Если R:0, W:0 и возраст> τ, то освобождаем страницу
   d) Если R:0 и возраст<τ, то перемещаем стрелку далее

WSClock. Иллюстрация



Оптимизация по работе со страничной памятью

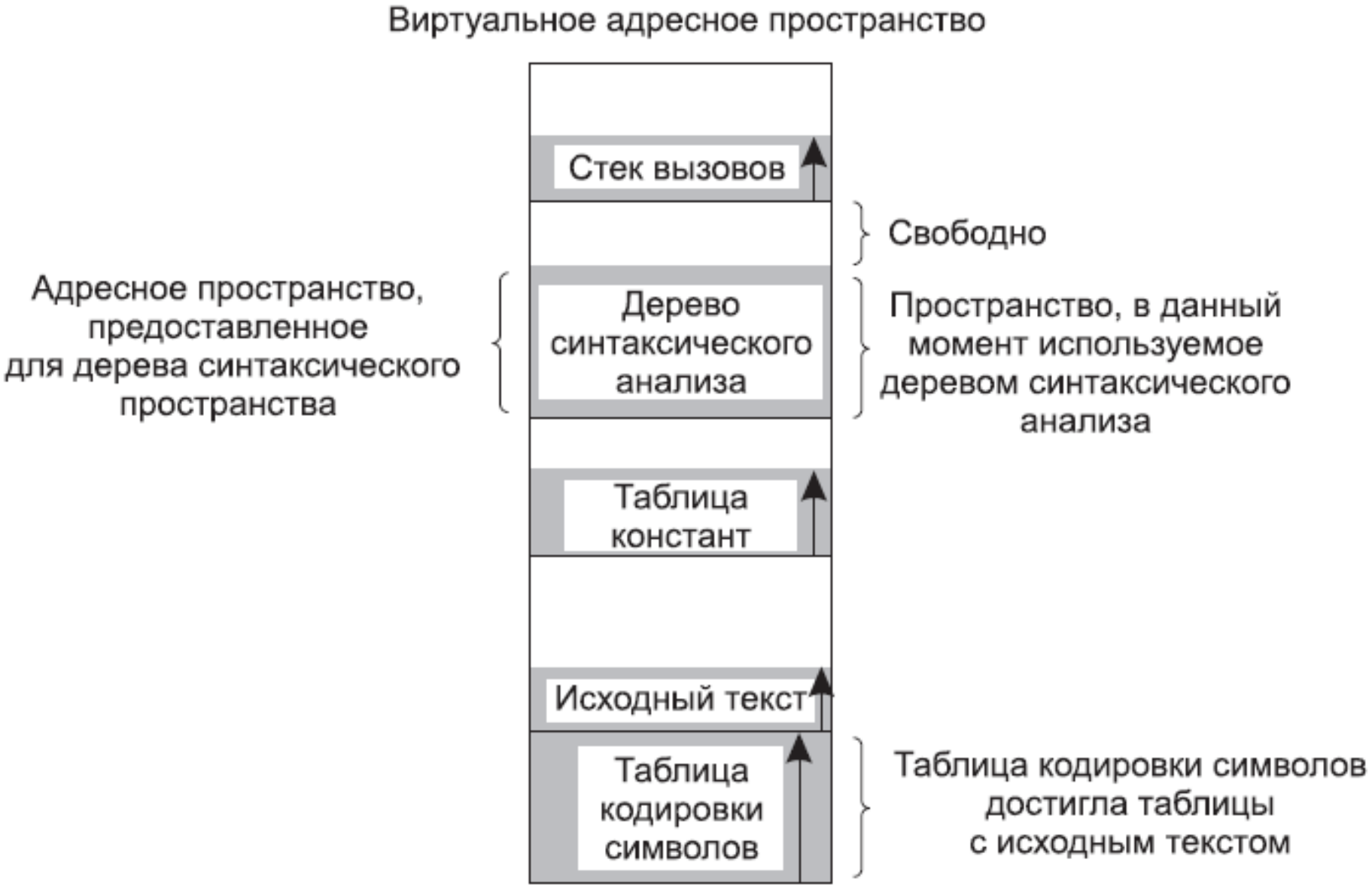
1. Использование локальных и глобальных политик (PFF)
2. Использование свопинга (когда сумма рабочих наборов превышает ОЗУ)
3. Оптимальный размер страниц
   издержки = (s/p)*e + p/2
4. Совместно используемые страницы и библиотеки
5. Отображаемые файлы
6. Своевременная очистка страниц (страничный демон)

События инициализирующие подкачку страниц

1. Создание процесса
2. Процесс планируется на выполнение
3. Возникновение ошибки отсутствия страницы
4. Завершение процесса

Сегментирование

1. Работа идет с разрастающимися и сужающимися структурами данных
2. Сегмент – это логический объект
3. Сегмент не имеет фиксированного размера


Виртуальное адресное пространство
Стек вызовов
Свободно
Адресное пространство, предоставленное для дерева синтаксического пространства
Дерево синтаксического анализа
Пространство, в данный момент используемое деревом синтаксического анализа
Таблица констант
Исходный текст
Таблица кодировки символов
Таблица кодировки символов достигла таблицы с исходным текстом