Виртуальная память – 2

Основы информатики.

Компьютерные основы программирования

goo.gl/X7evF

На основе **CMU** 15-213/18-243:
Introduction to Computer Systems

goo.gl/TDDVV

Лекция **14**, **21** Мая, 2017

Лектор:

Дмитрий Северов, кафедра информатики 608 КПМ

dseverov@mail.mipt.ru

cs.mipt.ru/wp/?page_id=346

Виртуальная память – 2: Системы

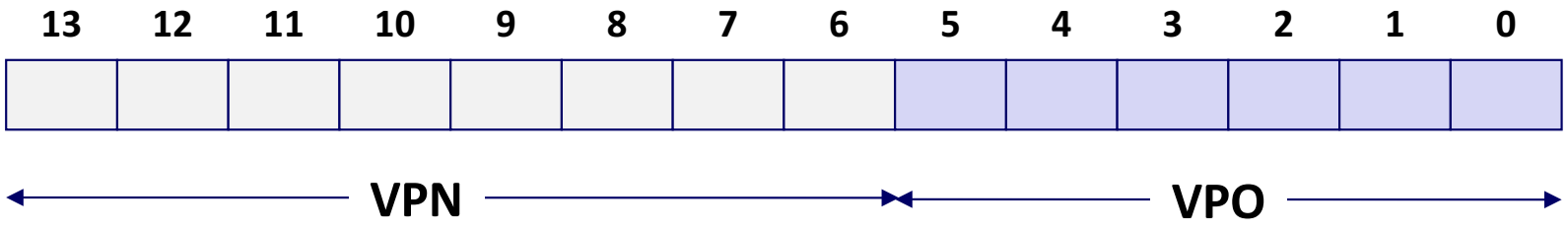
- **Пример простой подсистемы памяти**
- **Пример подсистемы памяти Core i7/Linux**
- **Отображение памяти**

Термины и обозначения

- **Основные параметры**
  - **N = 2^n** : размер виртуального адресного пространства
  - **M = 2^m** : Размер физического адресного пространства
  - **P = 2^p** : Размер страницы (в байтах)
- **Части виртуального адреса**
  - **TLBI**: TLB index – индекс ближайшего буфера трансляции
  - **TLBT**: TLB tag – метка ближайшего буфера трансляции
  - **VPO**: Virtual page offset – сдвиг в виртуальной странице – СВС
  - **VPN**: Virtual page number – номер виртуальной страницы – НВС
- **Части физического адреса**
  - **PPO**: Physical page offset – сдвиг в физической странице – СФС
  - **PPN:** Physical page number – номер физической страницы – НФС
  - **CO**: Cache offset – сдвиг байта (от начала) линии кеша
  - **CI:** Cache index – индекс кеша
  - **CT**: Cache tag – метка кеша

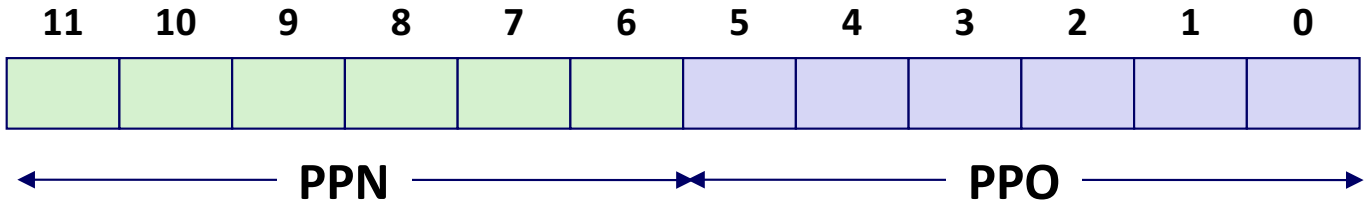
Пример простой подсистемы памяти

- **Адресация**
  - 14-битный виртуальный адрес
  - 12-битный физический адрес
  - Размер страницы = 64 байта



Номер виртуальной страницы

Сдвиг виртуальной страницы



Номер физической страницы

Сдвиг физической страницы

Таблица страниц простой подсистемы

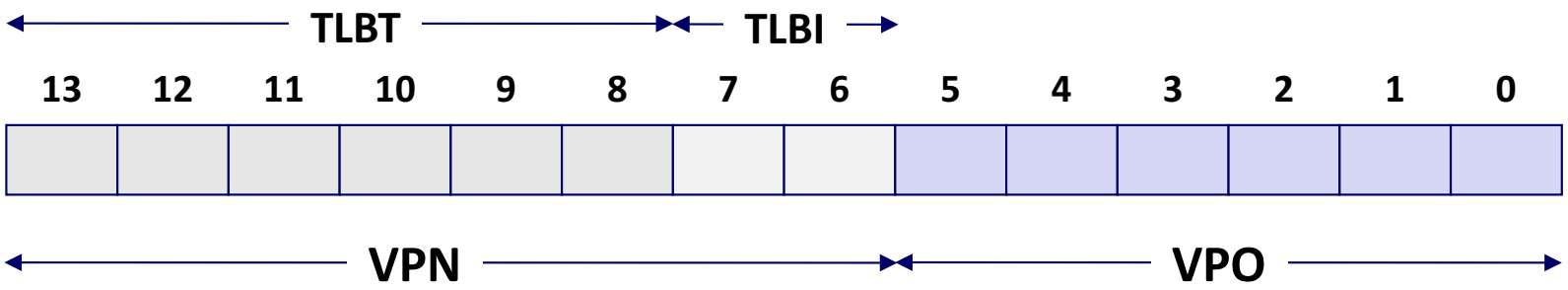
Показаны только первые 16 записей (из 256)

VPN	PPN	в кеше
00	28	1
01	–	0
02	33	1
03	02	1
04	–	0
05	16	1
06	–	0
07	–	0

VPN	PPN	в кеше
08	13	1
09	17	1
0A	09	1
0B	–	0
0C	–	0
0D	2D	1
0E	11	1
0F	0D	1

TLB простой подсистемы

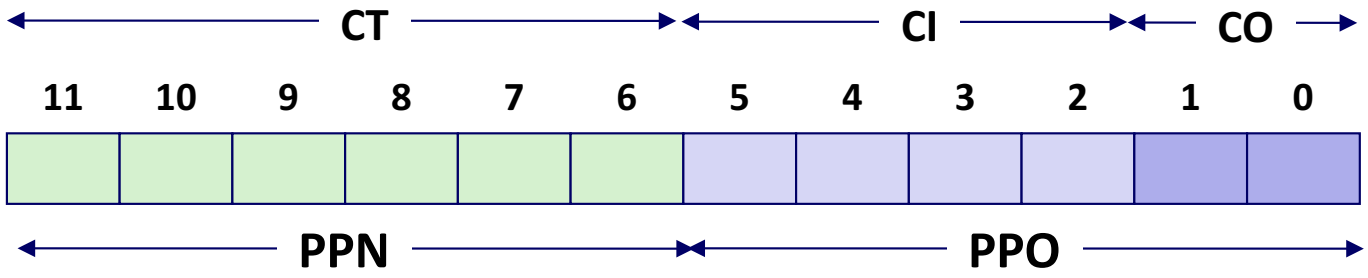
- 16 записей
- 4-вариантный



набор	метка	PPN	в кеше	метка	PPN	в кеше	метка	PPN	в кеше	метка	PPN	в кеше
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0

Кеш простой подсистемы

- 16 линий, размер блока 4 байта (B0, B1, B2, B3)
- Физическая адресация
- Прямое отображение (без вариантов)

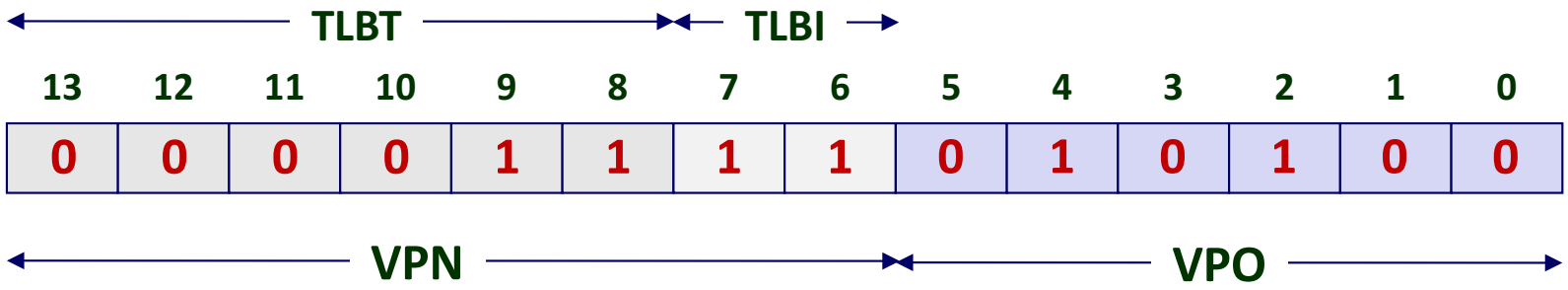


индекс	метка	в кеше	B0	B1	B2	B3
0	19	1	99	11	23	11
1	15	0	–	–	–	–
2	1B	1	00	02	04	08
3	36	0	–	–	–	–
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	–	–	–	–
7	16	1	11	C2	DF	03

индекс	метка	в кеше	B0	B1	B2	B3
8	24	1	3A	00	51	89
9	2D	0	–	–	–	–
A	2D	1	93	15	DA	3B
B	0B	0	–	–	–	–
C	12	0	–	–	–	–
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	–	–	–	–

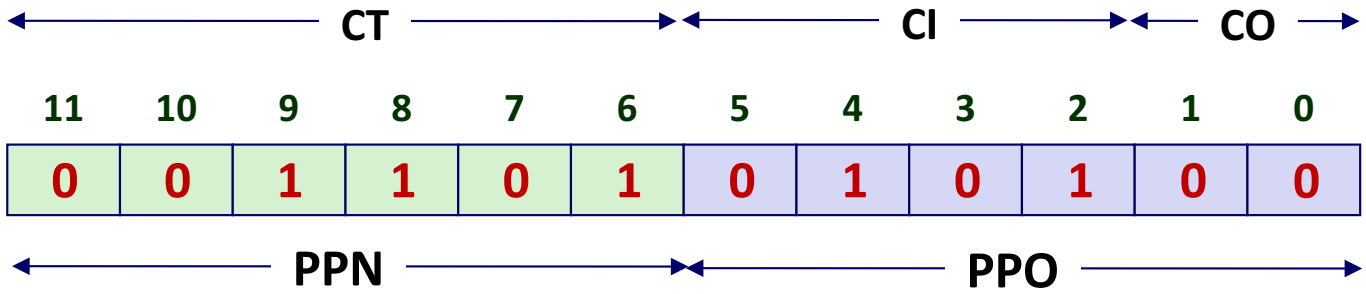
Пример трансляции адресов №1

Виртуальный адрес: 0x03D4



VPN 0x0F TLBI 0x3 TLBT 0x03 Попадание TLB? ДА Page Fault? НЕТ PPN: 0x0D

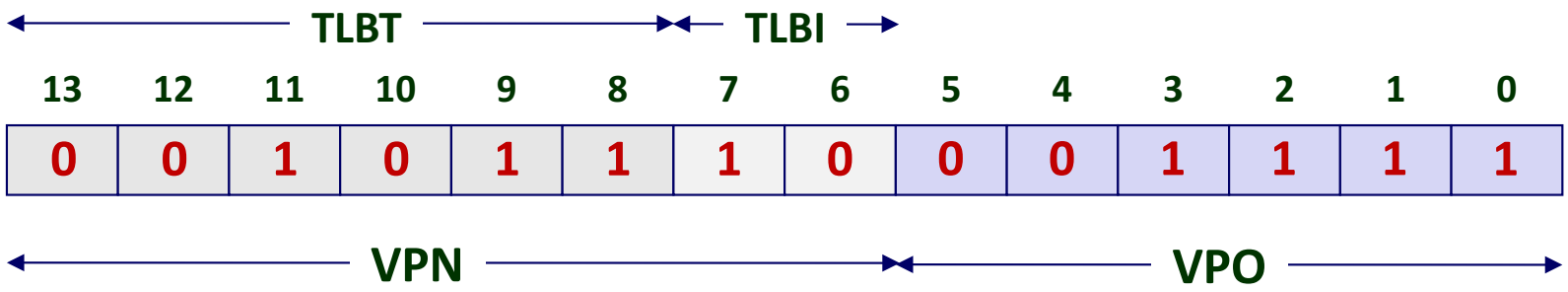
Физический адрес



CO 0 CI 0x5 CT 0x0D Попадание? ДА Байт: 0x36

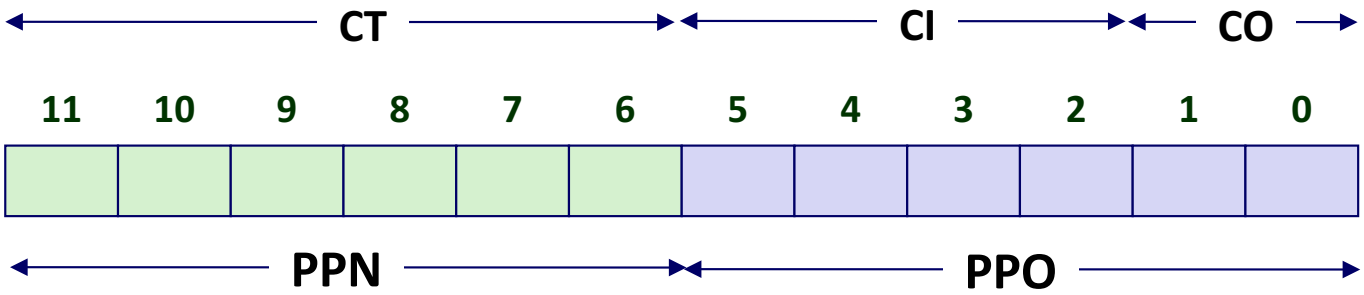
Пример трансляции адресов №2

Виртуальный адрес: `0x0B8F`



VPN **0x2E** TLBI **2** TLBT **0x0B** Попадание TLB? **НЕТ** Page Fault? **ДА** PPN: **???**

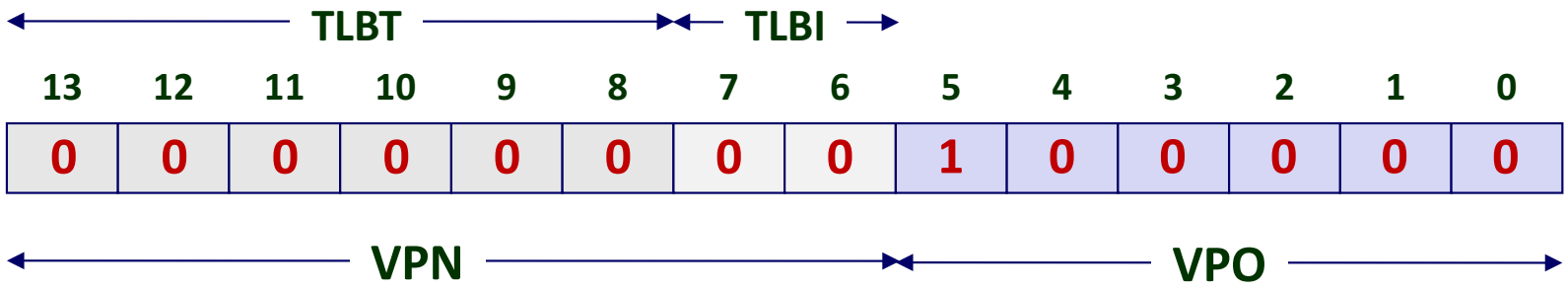
Физический адрес



CO ___ CI___ CT ____ Попадание? ___ Байт: ____

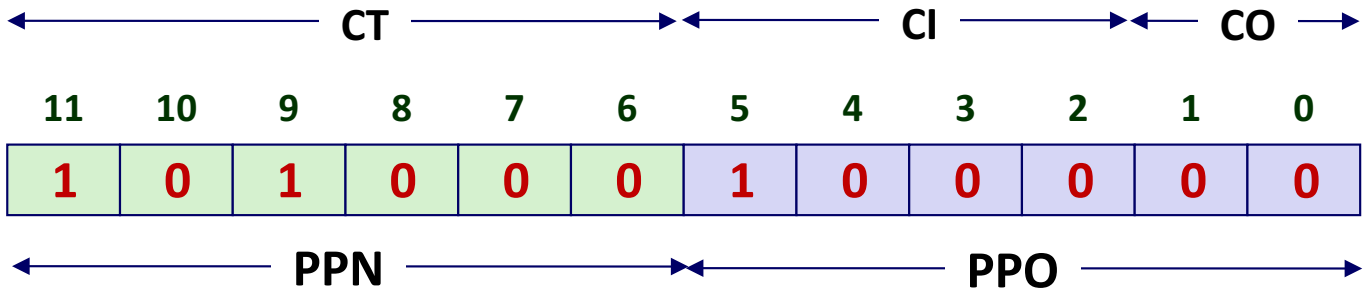
Пример трансляции адресов №3

Виртуальный адрес: 0x0020



VPN 0x00 TLBI 0 TLBT 0x00 Попадание TLB? НЕТ Page Fault? НЕТ PPN: 0x28

Физический адрес

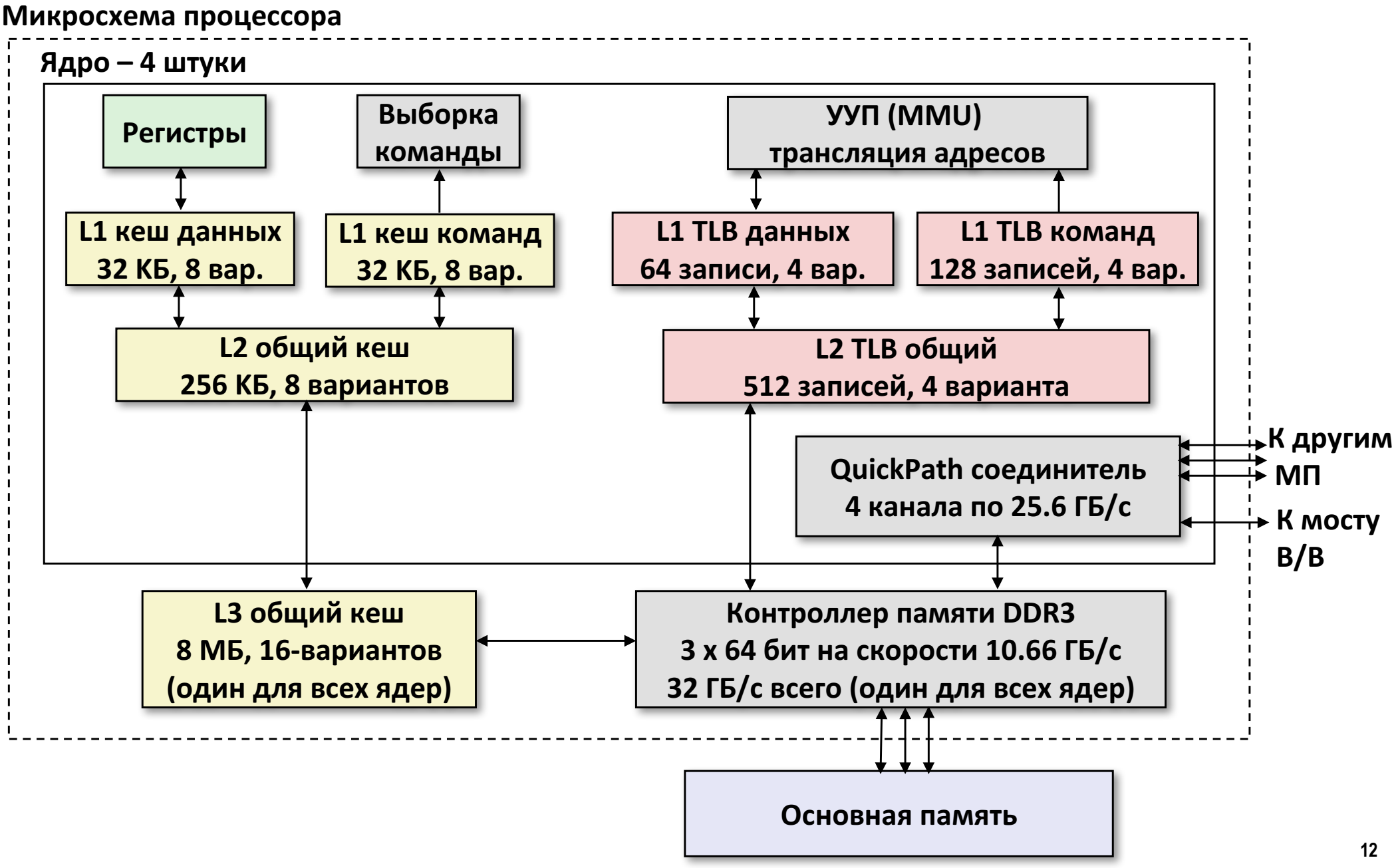


CO 0 CI 0x8 CT 0x28 Попадание? НЕТ Байт: в памяти

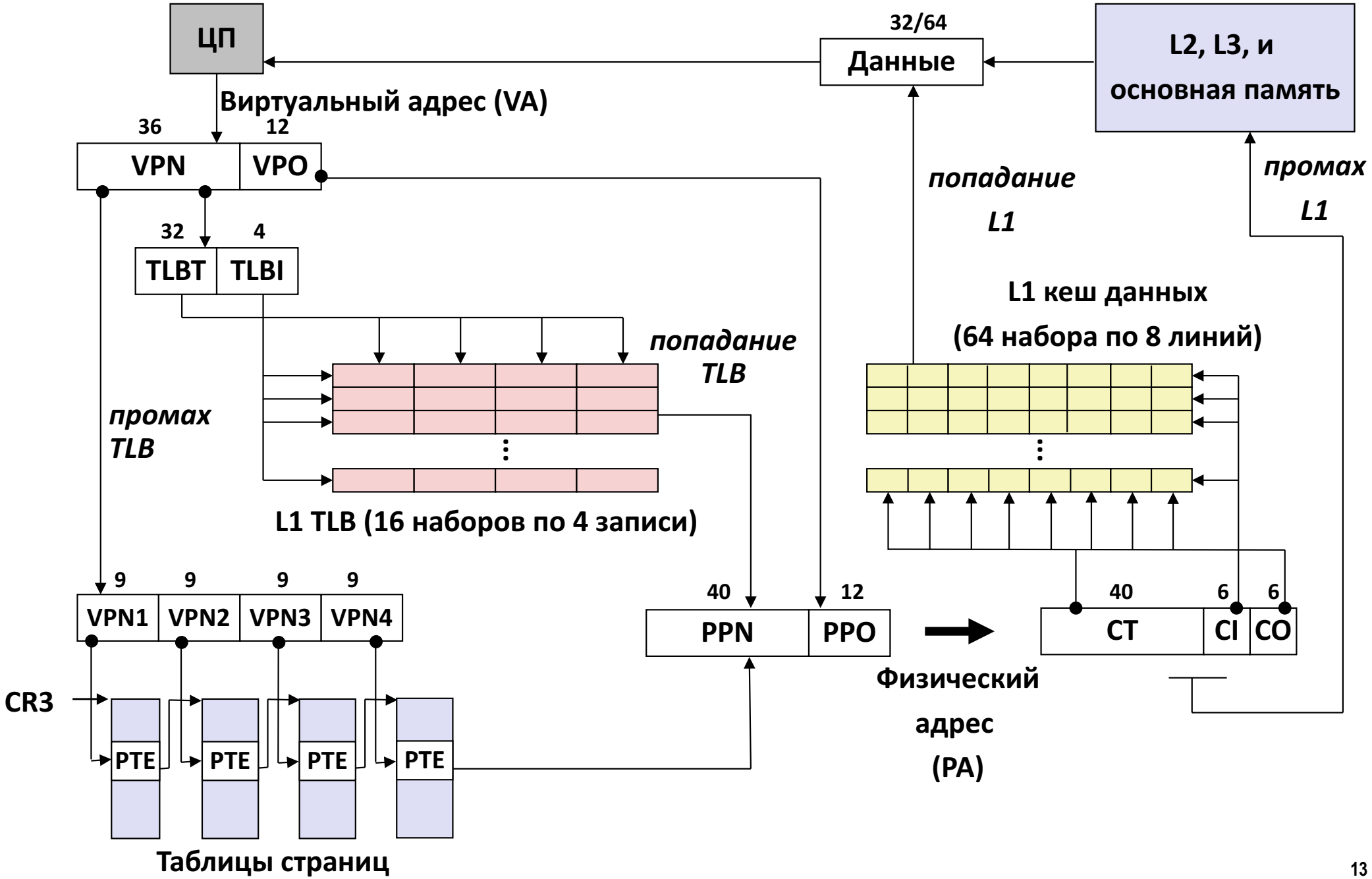
Виртуальная память – 2: Системы

- Пример простой подсистемы памяти
- Пример подсистемы памяти Core i7/Linux
- Отображение памяти

Подсистема памяти Intel Core i7



Трансляция адресов Core i7 полностью



Табличные записи Core i7 уровней 1-3

Каждая запись обозначает дочернюю таблицу страниц размером 4КБ

P: присутствие в физической памяти дочерней таблицы страниц

R/W: разрешение на чтение-запись или только чтение для всех достижимых станиц

U/S: разрешение доступа пользователю или ядру ко всем достижимым станицам

WT: политика записи в кеш дочерней таблицы страниц (отложенная или сквозная)

CD: запрет или разрешение кеширования дочерней таблицы страниц

A: признак обращения (устанавливается УУП при чтении и записи, сбрасывается ПО).

PS: размер страницы 4 КБ или 4 МБ (только для записей уровня 1).

G: признак глобальности таблицы страниц (не откачивается из TLB при переключении задач)

Основа физического адреса таблицы страниц: 40 старших бит физического адреса таблицы страниц (принудительное выравнивание на границу 4КБ)

Табличные записи Core i7 уровня 4

Каждая запись обозначает дочернюю страницу размером 4КБ

P: присутствие в физической памяти дочерней страницы

R/W: разрешение на чтение-запись или только чтение для дочерней страницы

U/S: разрешение доступа пользователю или ядру для дочерней страницы

WT: политика записи в кеш дочерней страницы (отложенная или сквозная)

CD: разрешение или запрет кеширования дочерней страницы

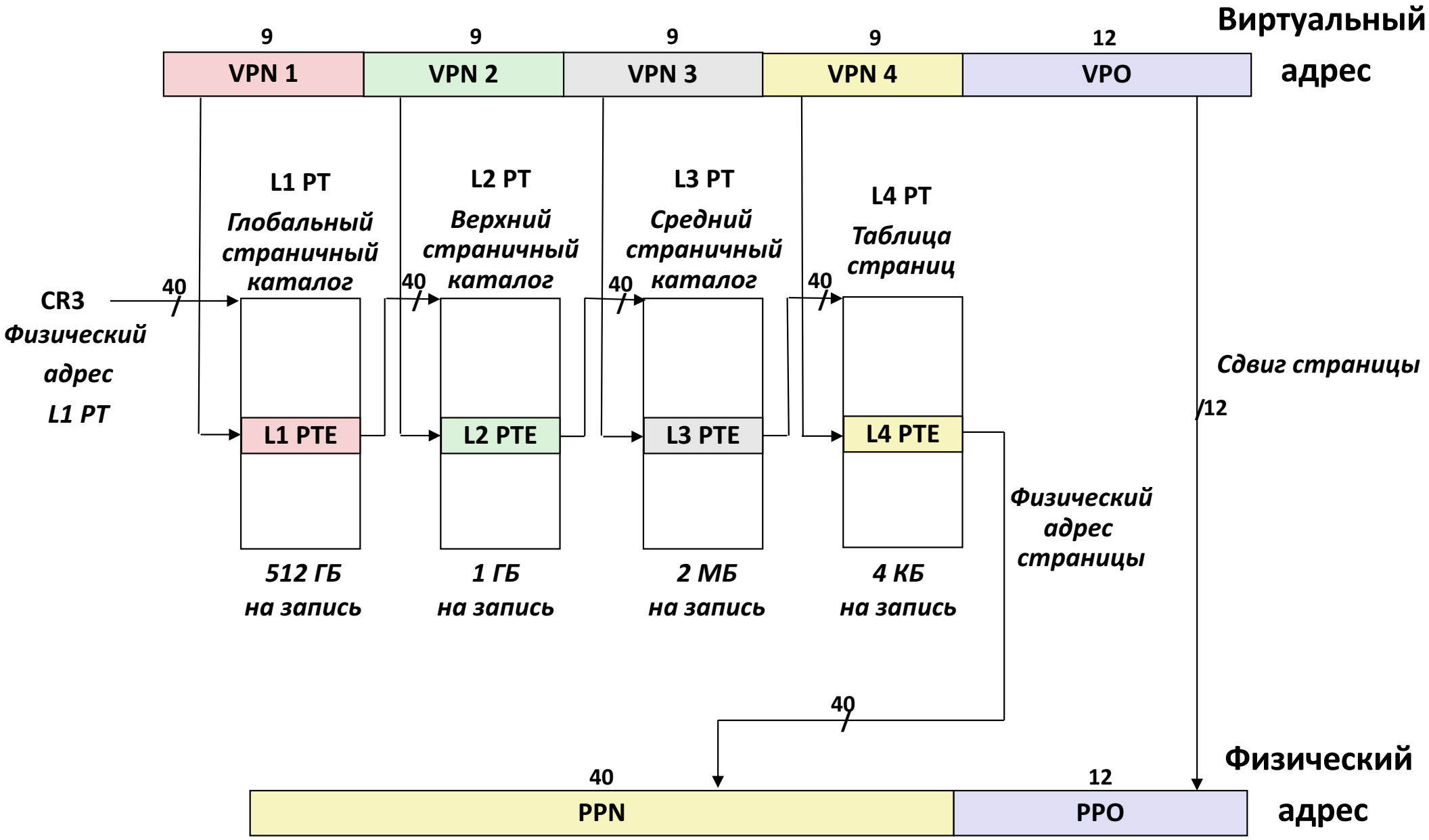
A: признак обращения (устанавливается УУП при чтении и записи, сбрасывается ПО).

D: признак записи (устанавливается УУП при записи, сбрасывается ПО)

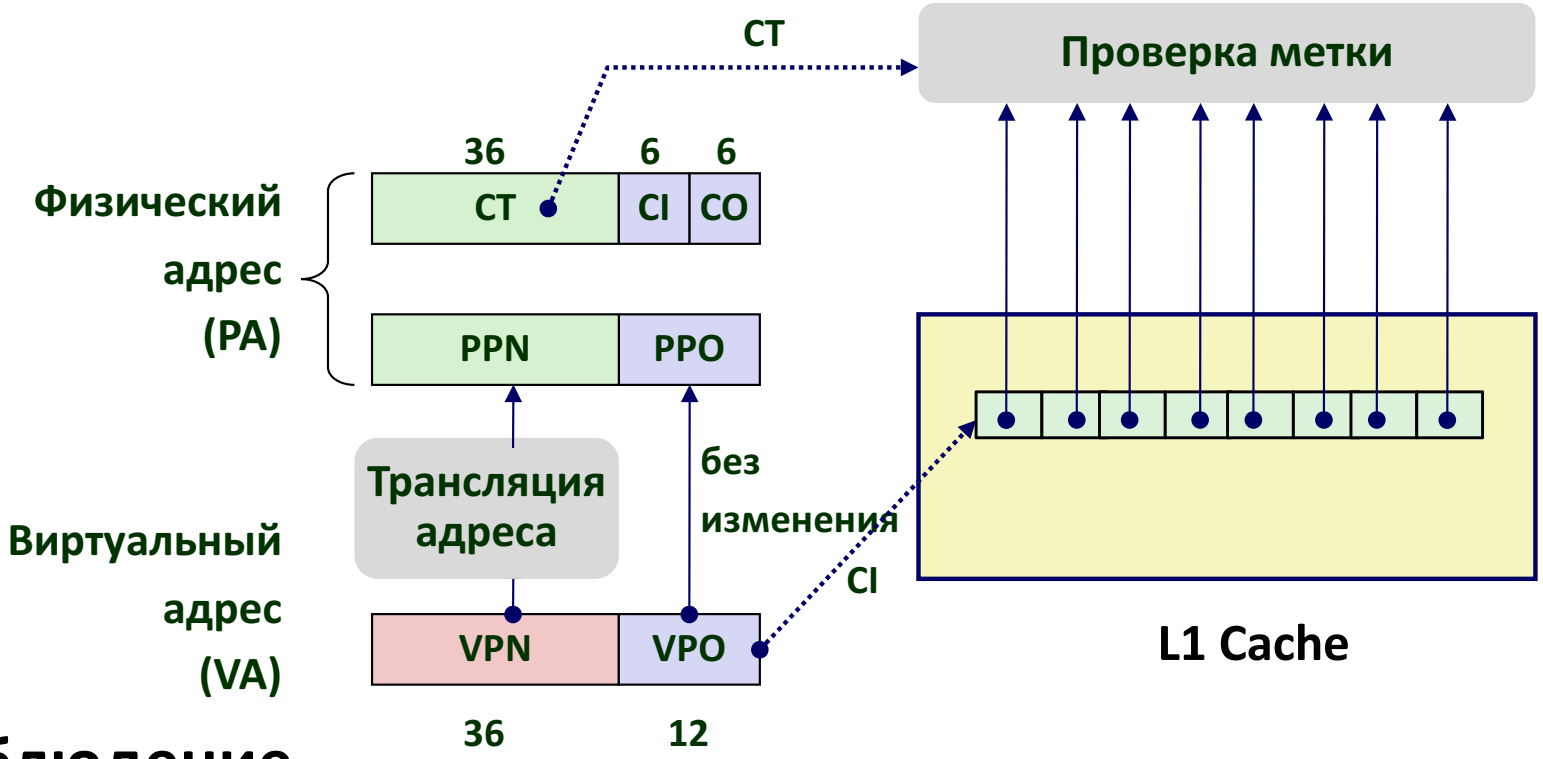
G: признак глобальности страницы (не откачивается из TLB при переключении задач)

Основа физического адреса страницы: 40 старших бит физического адреса страницы (принудительное выравнивание на границу 4КБ)

Трансляция страничных таблиц Core i7

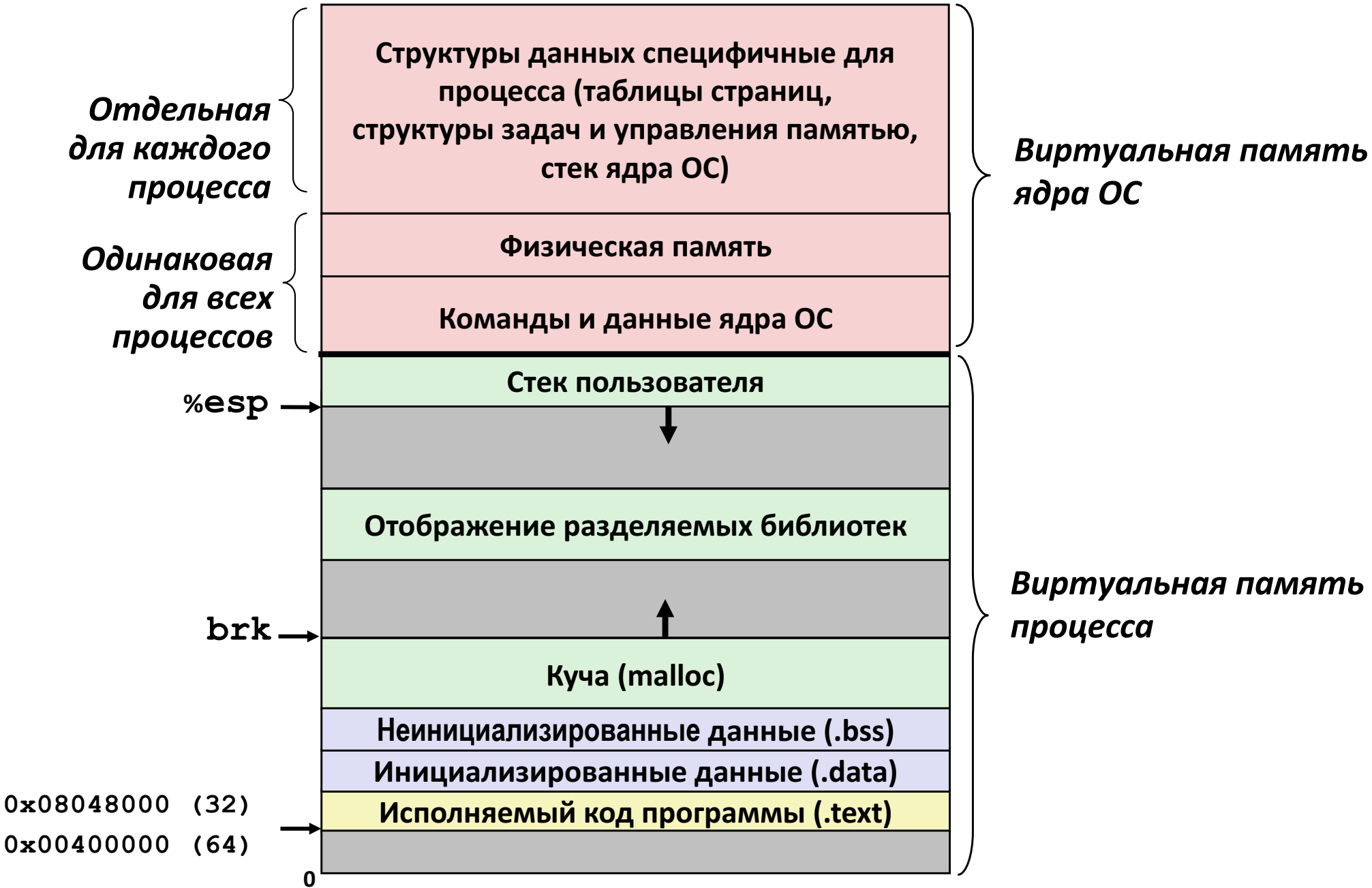


Ускорение доступа к L1

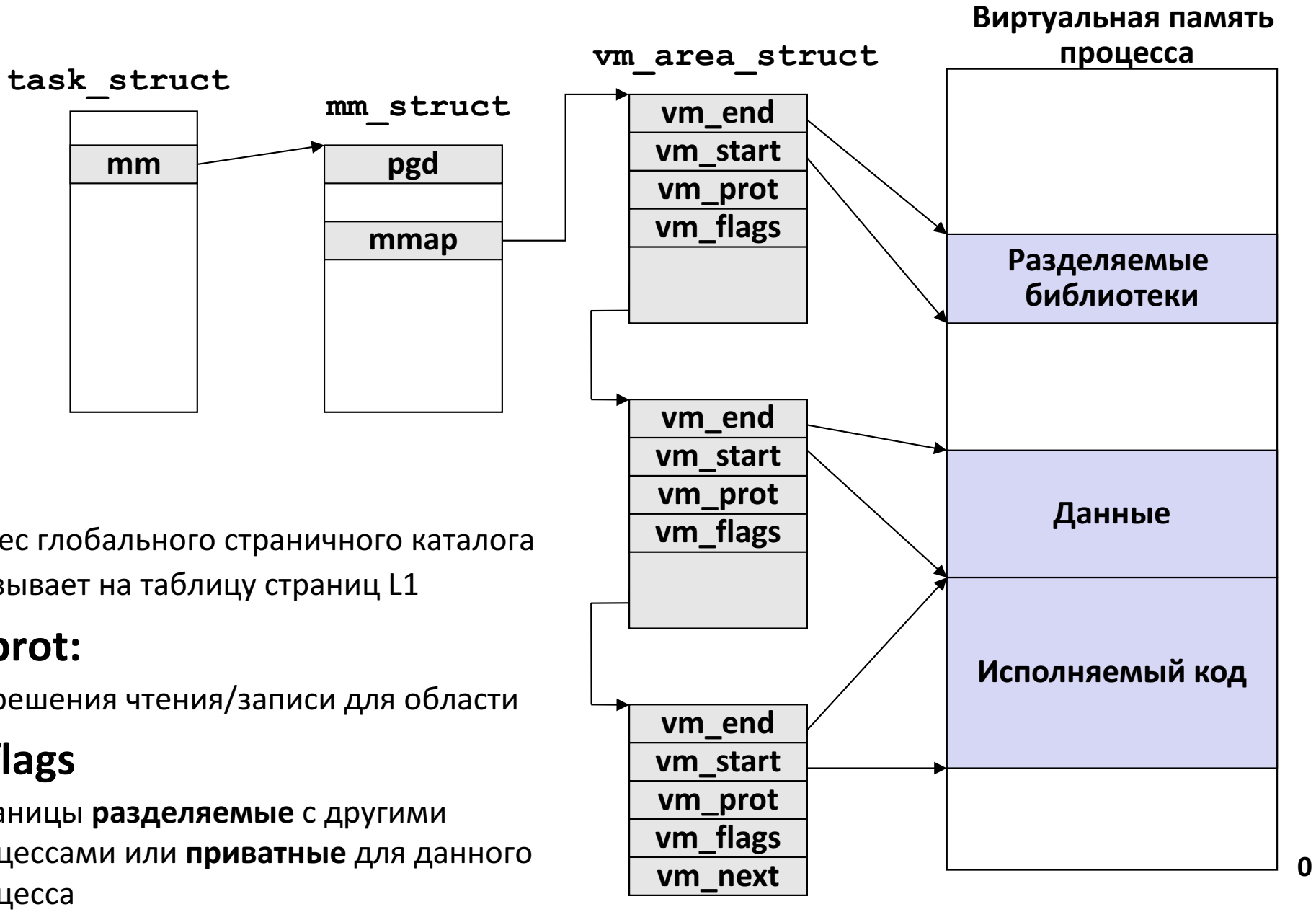


- **Наблюдение**
  - Биты CI идентичны в физическом и виртуальном адресах
  - Индексирование в кеше возможно при трансляции адресов
  - Обычно попадание в TLB, и биты PPN (CT) доступны немедленно
  - “Виртуально индексирован, физически размечен”
  - Чтобы обеспечить такое, размеры кеша аккуратно подобраны

Виртуальная память в процессах Linux

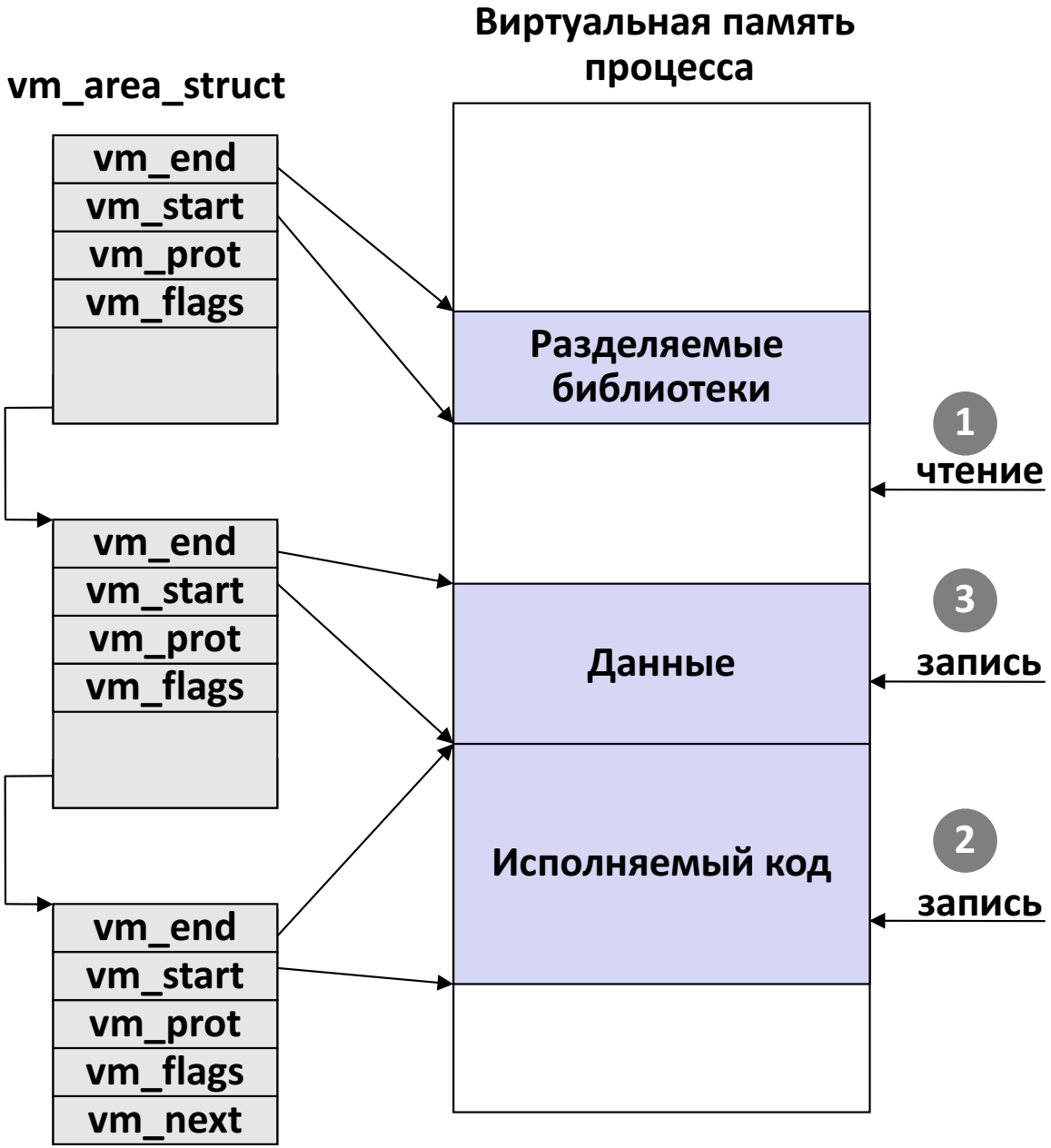


В Linux ВП – коллекция “областей”



- **pgd:**
  - Адрес глобального страничного каталога
  - Указывает на таблицу страниц L1
- **vm_prot:**
  - Разрешения чтения/записи для области
- **vm_flags**
  - Страницы **разделяемые** с другими процессами или **приватные** для данного процесса

Обработка страничного сбоя в Linux



Segmentation fault:
обращение к несуществующей странице

Обычный страничный сбой

Защитное исключение:
например, попытка записи в область только для чтения (Linux диагностирует как Segmentation fault)

Виртуальная память – 2: Системы

- Пример простой подсистемы памяти
- Пример подсистемы памяти Core i7/Linux
- Отображение памяти

Отображение памяти

- **Области ВП инициализируются соотнесением их с объектами на диске**
  - Процесс называется ***отображение памяти (memory mapping)***

- **Области могут быть инициализированы содержимым…**
  - ***…обычных файлов на диске Regular*** (например файла исполняемого объекта)
    - Байты, инициализирующие страницы, поступают из разделов файла
  - ***…анонимных файлов*** (то есть из ниоткуда)
    - первый сбой занимает обнулённую страницу (***demand-zero page***)
    - после записи в страницу (***dirtied***), она обрабатывается как и прочие

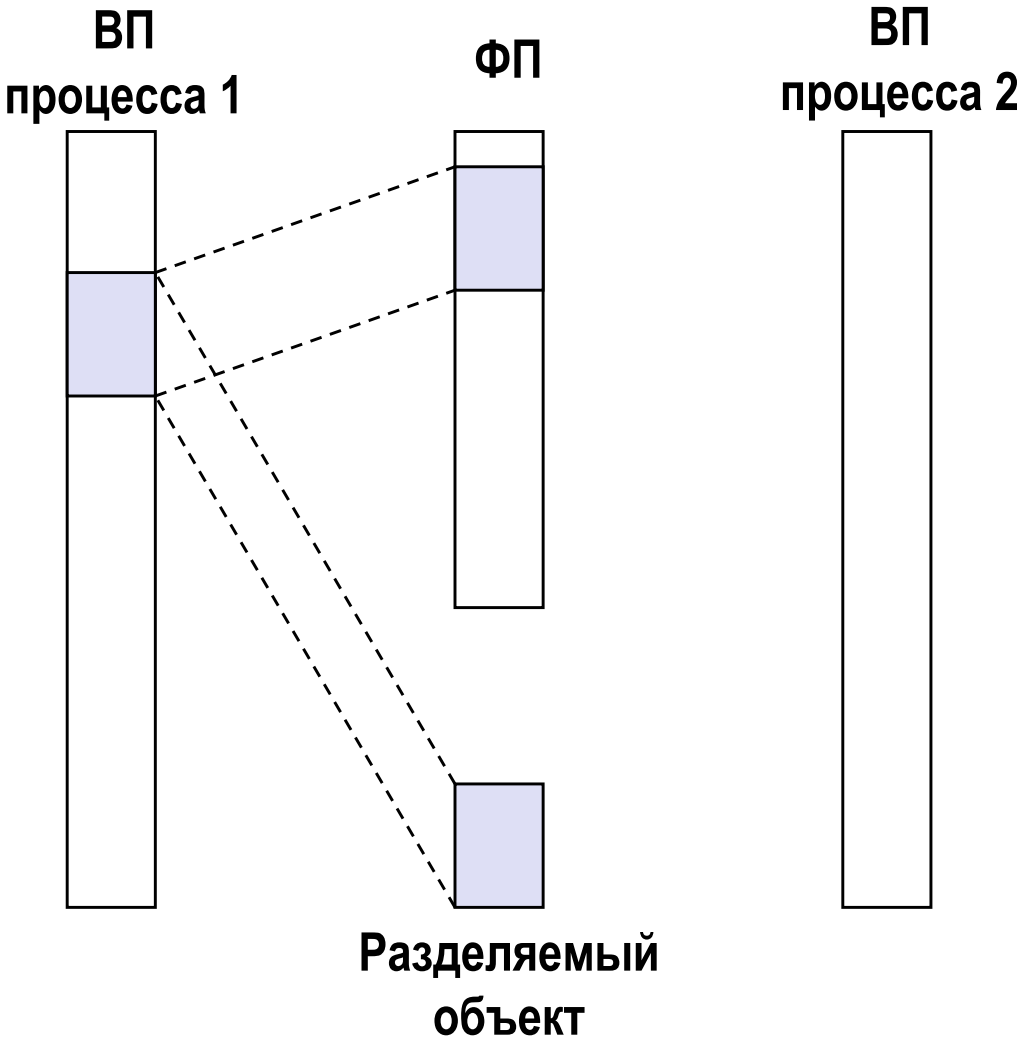
- **Страницами после записи обмениваются память и специальный *файл подкачки (swap file)***

Подкачка по требованию

- ***Ключевая идея:*** **виртуальные страницы не копируются в физическую память пока к ним не обращаются!**
  - Называется ***подкачка по требованию (demand paging)***

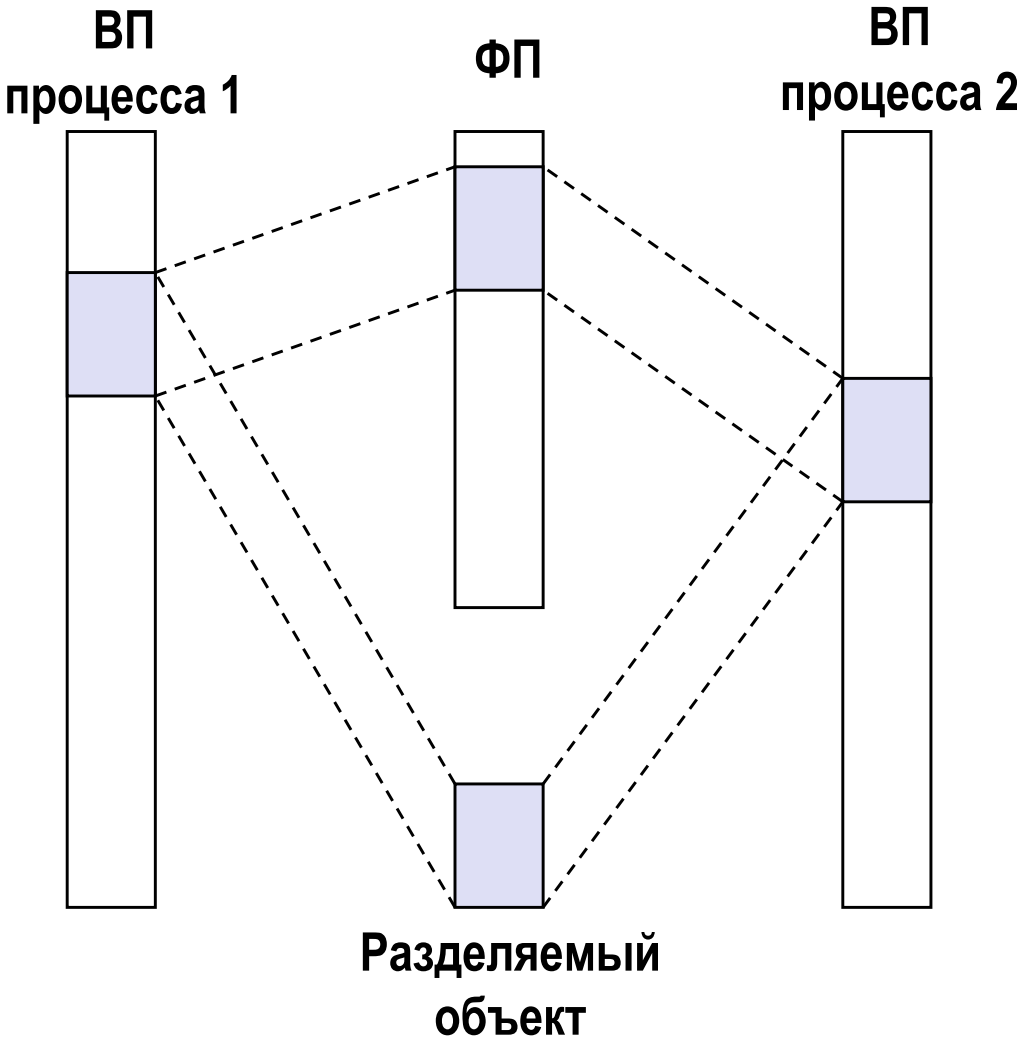
- **Критически важно для временной и пространственной эффективности**

Вновь о разделении: разделяемые объекты



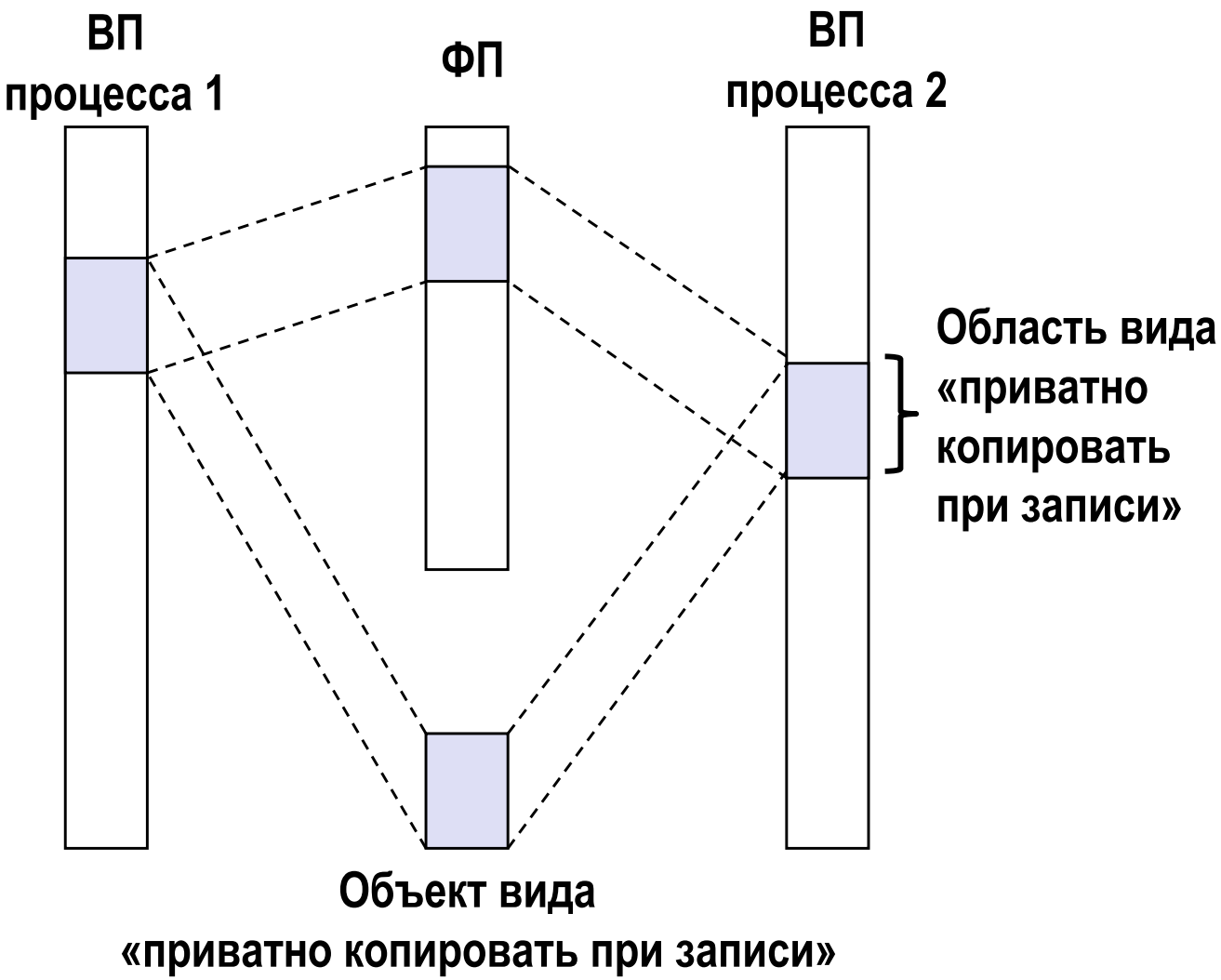
- Процесс 1 отображает разделяемый объект

Вновь о разделении: разделяемые объекты



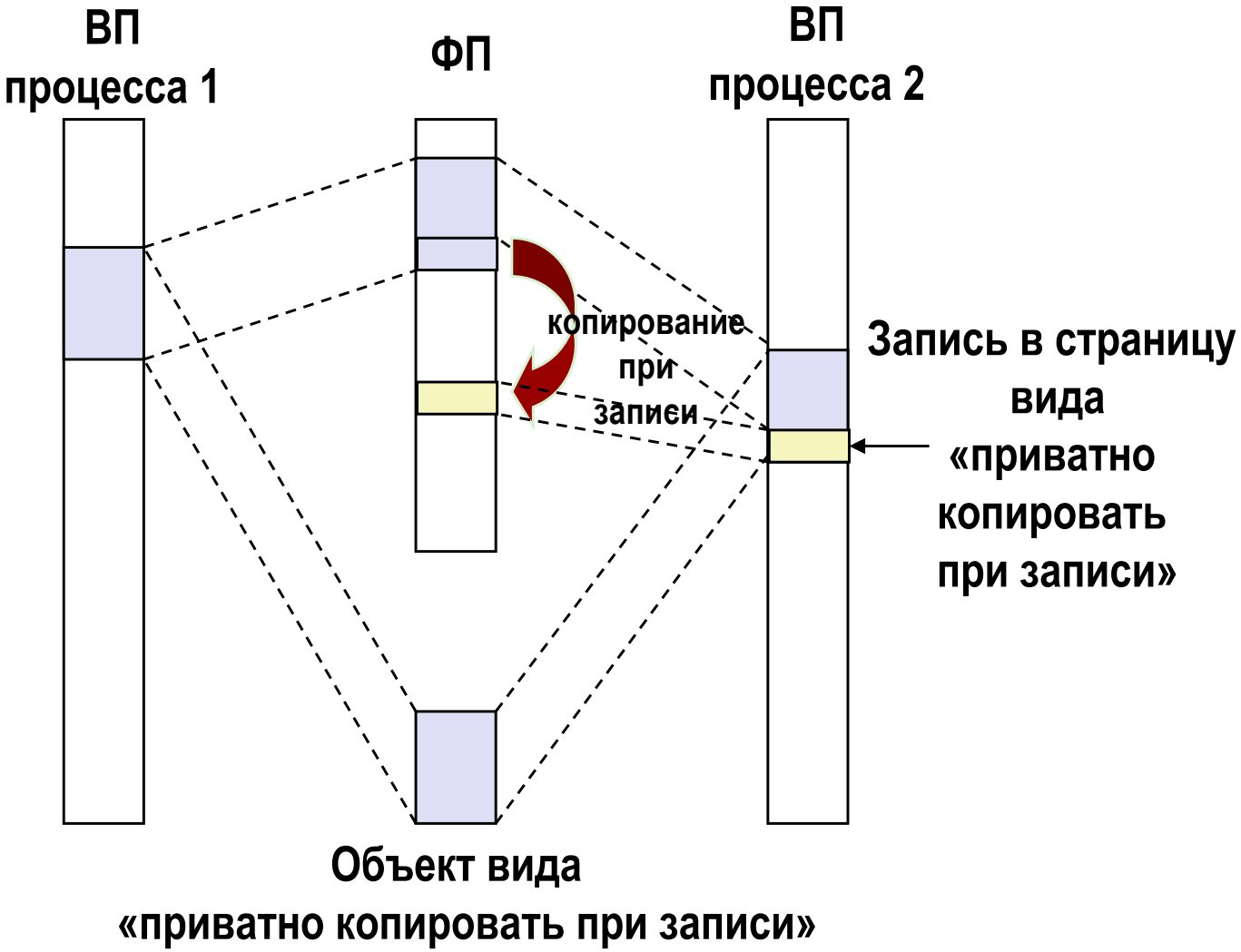
- Процесс 2 отображает разделяемый объект
- Заметьте, что виртуальные адреса могут быть разными!

Вновь о разделении: приватная копия объекта при записи



- Два процесса отображают объект вида *приватно копировать при записи (COW)*
- Область помечена соответственно
- Страничные записи в приватных областях помечены «только чтение»

Вновь о разделении: приватная копия объекта при записи



- Команда записи в приватную страницу даёт сбой защиты
- Обработчик создаёт новую страницу вида чтение/запись
- Команда перезапускается по выходе из обработчика
- Копирование откладывается насколько возможно!

Вновь о функции `fork`

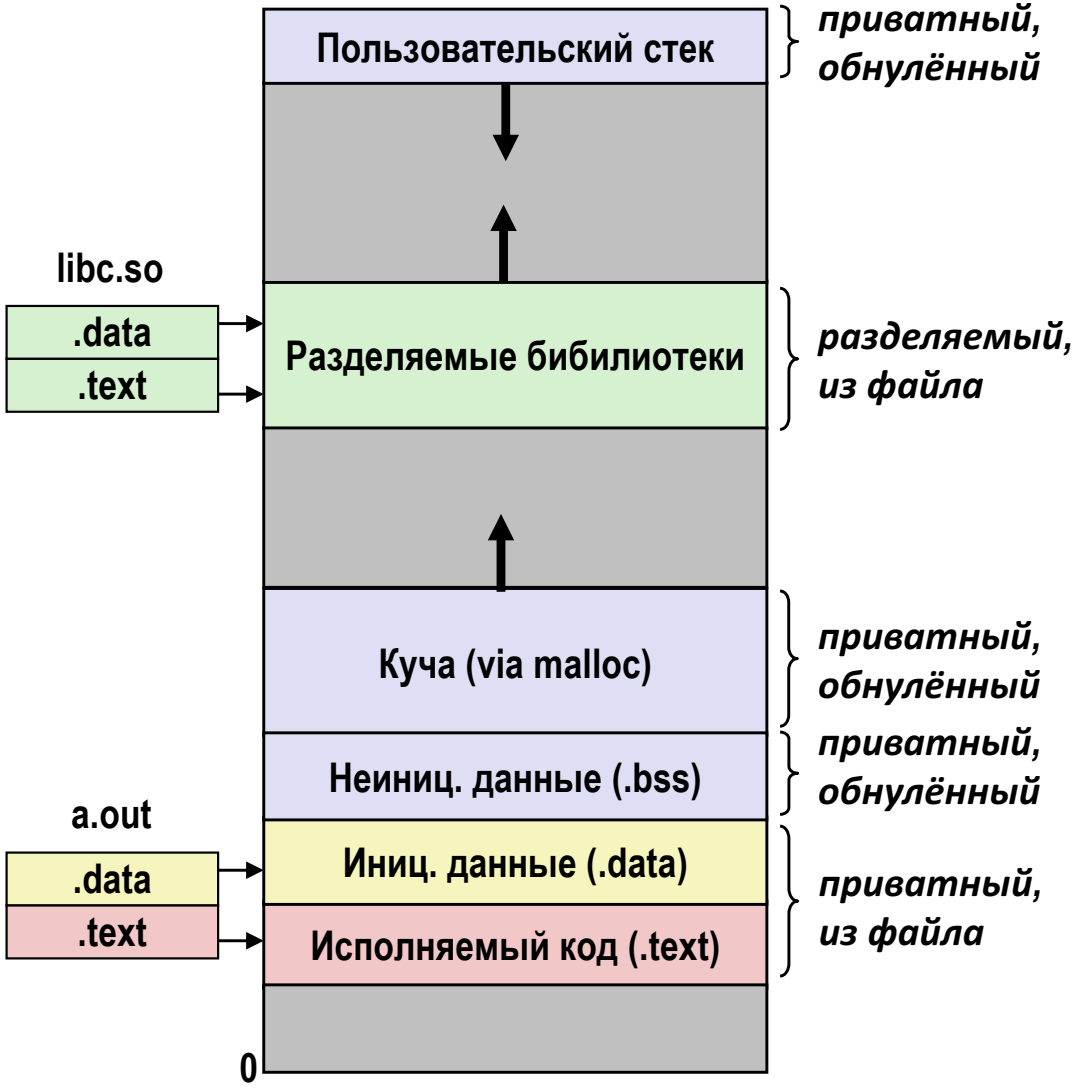
- **ВП и отображение памяти проясняют как `fork` обеспечивает приватное адресное пространство для каждого процесса**

- **При создании нового процесса**
  - В точности копируем текущие `mm_struct`, `vm_area_struct`, и таблицы страниц.
  - Помечаем «только чтение» каждую страницу в обоих процессах
  - Помечаем «приватное копирование при записи» каждую `vm_area_struct` в обоих процессах

- **При возврате, у каждого процесса – точная копия ВП**

- **Очередные записи создают новые страницы с помощью механизма копирования при записи (COW)**

Вновь о функции execve



- **Для загрузки и исполнения новой программы `a.out` в текущем процессе используем `execve`:**
- **Освобождаем старые `vm_area_struct` и таблицы страниц**
- **Создаём новые `vm_area_struct`'s и таблицы страниц**
  - Код и инициализированные данные из объектных файлов
  - `.bss` и стек – из безымянных файлов
- **Устанавливаем PC на точку входа в `.text`**
  - Linux при необходимости будет обслуживать сбои страниц кода и данных.

Отображение памяти пользователем

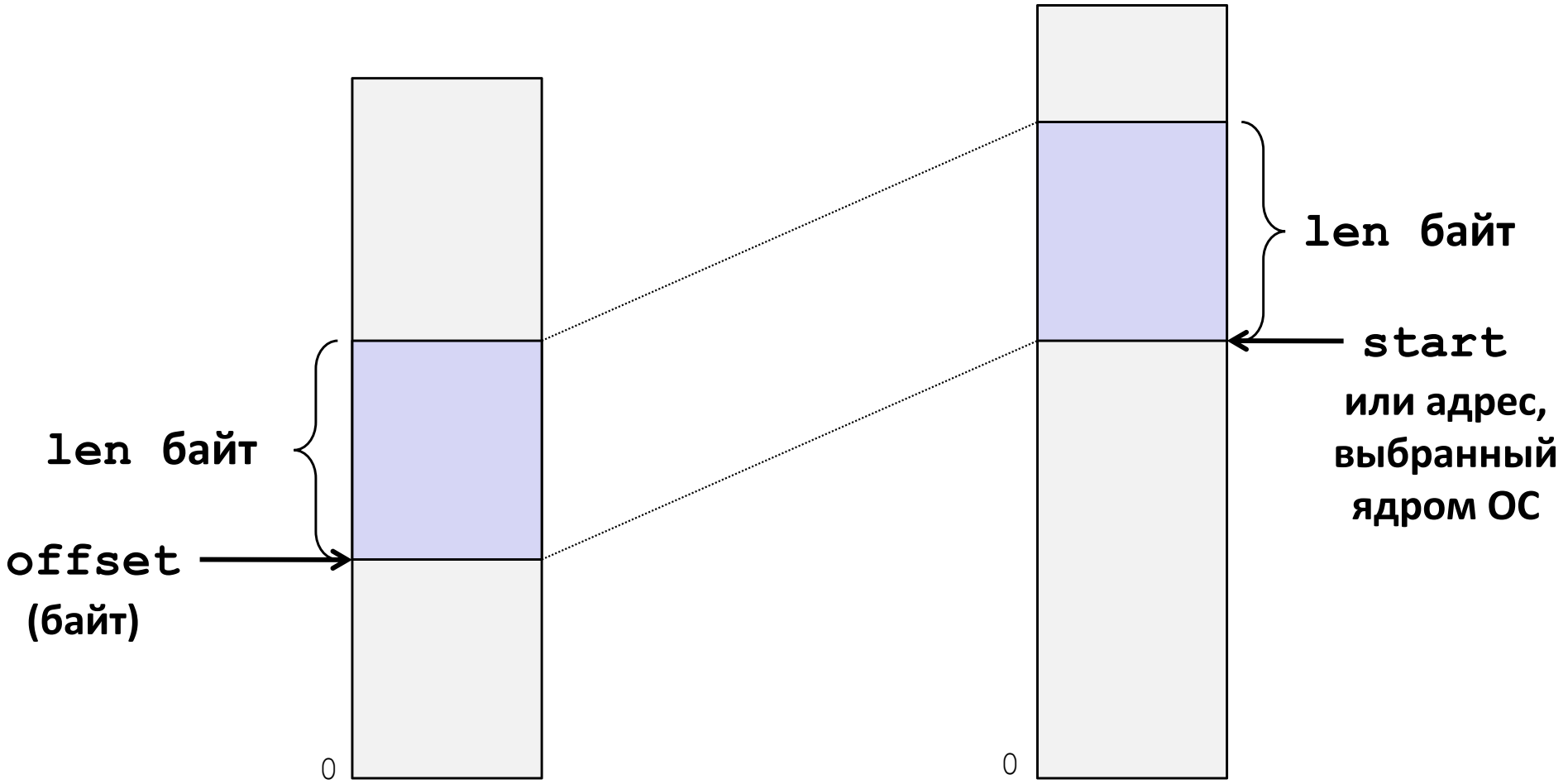
```
void *mmap(void *start, int len,
           int prot, int flags, int fd, int offset)
```

- **Отображает `len` байт начиная со смещения `offset` в файле, указанном файловым дескриптором `fd`, предпочтительно с адреса `start`**
  - **`start`:** может быть 0 для “выбери адрес”
  - **`prot`**: PROT_READ, PROT_WRITE, ...
  - **`flags`**: MAP_ANON, MAP_PRIVATE, MAP_SHARED, ...

- **Возвращает указатель на начало отображённой области (возможно не `start`)**

Отображение памяти пользователем

```
void *mmap(void *start, int len,
           int prot, int flags, int fd, int offset)
```



Дисковый файл, указанный файловым дескриптором `fd`

Виртуальная память процесса

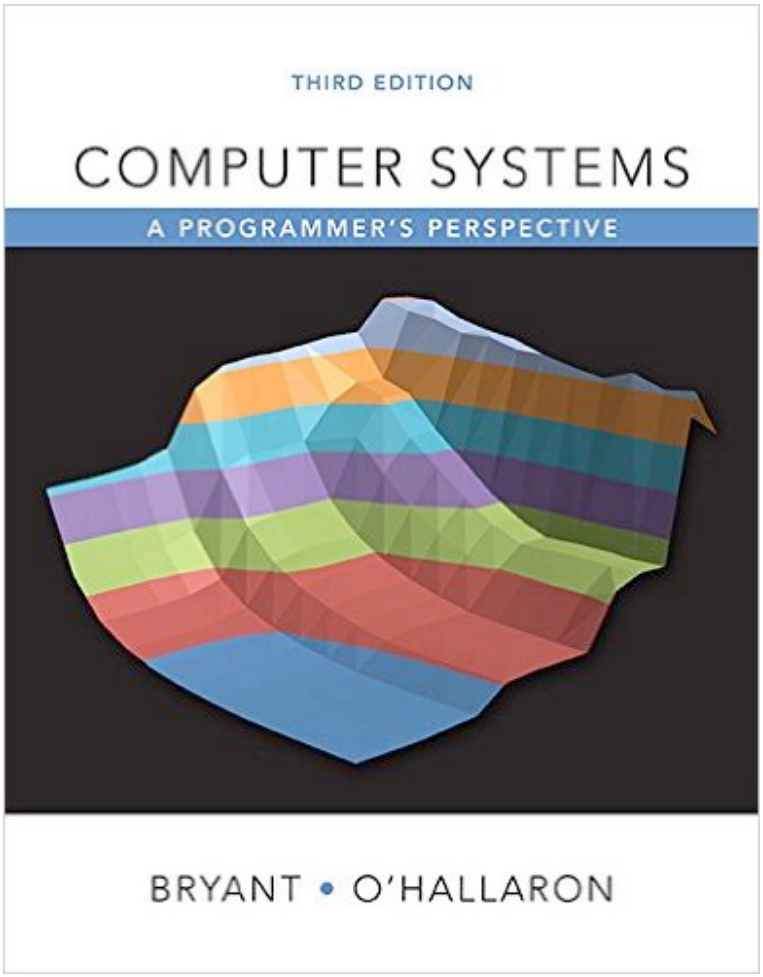
Использование mmap для копирования файлов

- Копирование без передачи данных в пространство пользователя

```
#include "csapp.h"

/*
 * mmapcopy - использует mmap
 *            для копирования
 *            файла fd в stdout
 */
void mmapcopy(int fd, int size)
{

   /* Указатель на область ВП
      отображённую в ФП */
    char *bufp;

    bufp = mmap(NULL, size,
                PROT_READ,
                MAP_PRIVATE, fd, 0);
    write(1, bufp, size);
    return;
}
```

```
/* Пускатель mmapcopy */
int main(int argc, char **argv)
{
    struct stat stat;
    int fd;

    /* Проверка необходимых параметров*/
    if (argc != 2) {
        printf("usage: %s <filename>\n”,
                argv[0]);
        exit(0);
    }

    /* Копируем параметр в stdout */
    fd = Open(argv[1], O_RDONLY, 0);
    fstat(fd, &stat);
    mmapcopy(fd, stat.st_size);
    exit(0);
}
```

Первоисточники



- http://www.cs.cmu.edu/afs/cs/academic/class/15213-f15/www/index.html

Немного о компьютерах

- Представление и исполнение программ
- Представление информации и манипуляции с ней
- Машинный уровень представления программ
- Архитектура процессора
- Оптимизация производительности программ
- Иерархия памяти
- Связывание
- Поток управления с исключительными ситуациями
- Виртуальная память
- Ввод-вывод на уровне системы
- Сетевое программирование
- Параллельное программирование

Немного о компьютерах. Осталось

- Представление информации и манипуляции с ней
- Машинный уровень представления программ
- **Архитектура процессора – Y86**
- Оптимизация производительности программ
- Иерархия памяти

- Связывание
- Поток управления с исключительными ситуациями
- Виртуальная память

- **Ввод-вывод на уровне системы**
- **Сетевое программирование**
- **Параллельное программирование**

Представление и исполнение программ

- **Машинный уровень представления программ**
  - Генерация кода с высокими уровнями оптимизации
  - Архитектура подсистем с плавающей точкой x87, SSE
- **Оптимизация производительности программ**
  - Команды SIMD
  - Профилирование
- **Иерархия памяти**
  - Блокирование для улучшения временной локальности

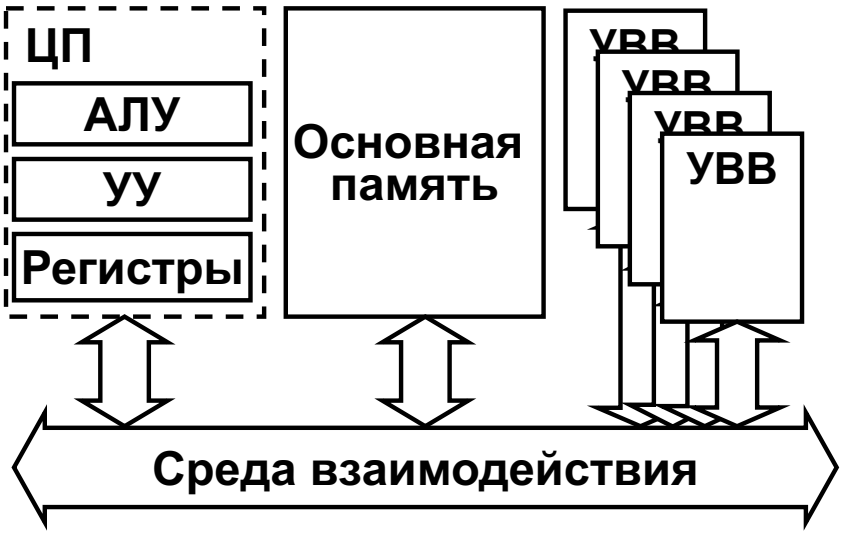
Запуск программы в системе

- **Связывание**
  - Код не зависящий от размещения в памяти
  - Системные средства работы с объектными файлами
- **Поток управления с исключительными ситуациями**
  - Обработка ошибок обращений к системе
  - Нелокальные переходы
  - Сигналы
  - Системные средства работы с процессами
- **Виртуальная память**
  - Динамическое выделение памяти
  - Сборка мусора
  - Частые ошибки работы с памятью в С-программах

Взаимодействие программ

- **Ввод-вывод на уровне системы**
  - **Файлы, их метаданные**
  - **Надёжный ввод/вывод, совместное использование файлов**
  - **Перенаправление, стандартный ввод/вывод**
- **Сетевое программирование**
  - **Клиент-серверная модель**
  - **Сети, интернет, sockets, web**
  - **Простейший веб-сервер**
- **Параллельное программирование**
  - Сервер, преимущества и недостатки процессов
  - Общий ввод/вывод
  - Нити, общие переменные, синхронизация семафорами
  - Вызовы параллелизма – гонки, взаимоблокировки

Масштабы событий



Крупнее: среда исполнения

- Загрузка кода программы
- Выделение ресурсов
- Нормирование работы и реагирование на запросы
- Освобождение ресурсов

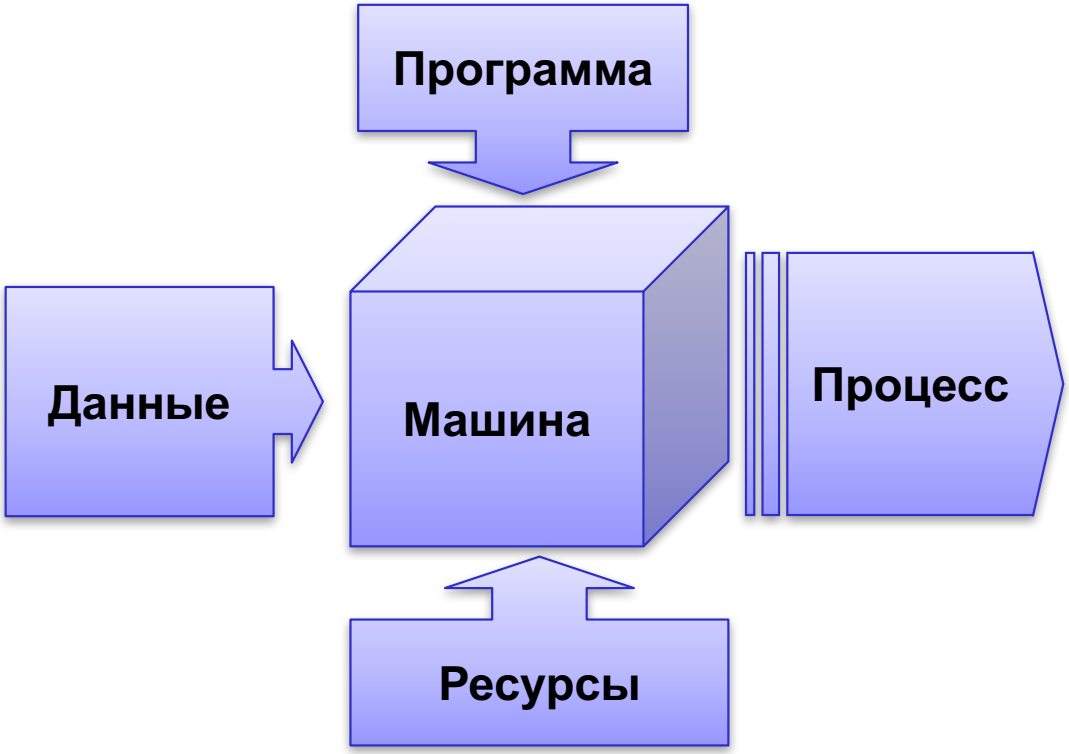
Программа

- Ваш процесс
  - Выполнение команд
  - Изменение состояния
- Другие процессы

Мельче: команда

- Выборка кода операции
- Декодирование операции
- Выборка операндов
- Выполнение операции
- Сохранение результата

Понятие машинных вычислений



Программа

определённый набор предписаний

Данные

неопределённый набор бит

Процесс

ожидаемый набор событий

Ресурсы

- Энергия
- Время
- Пространство