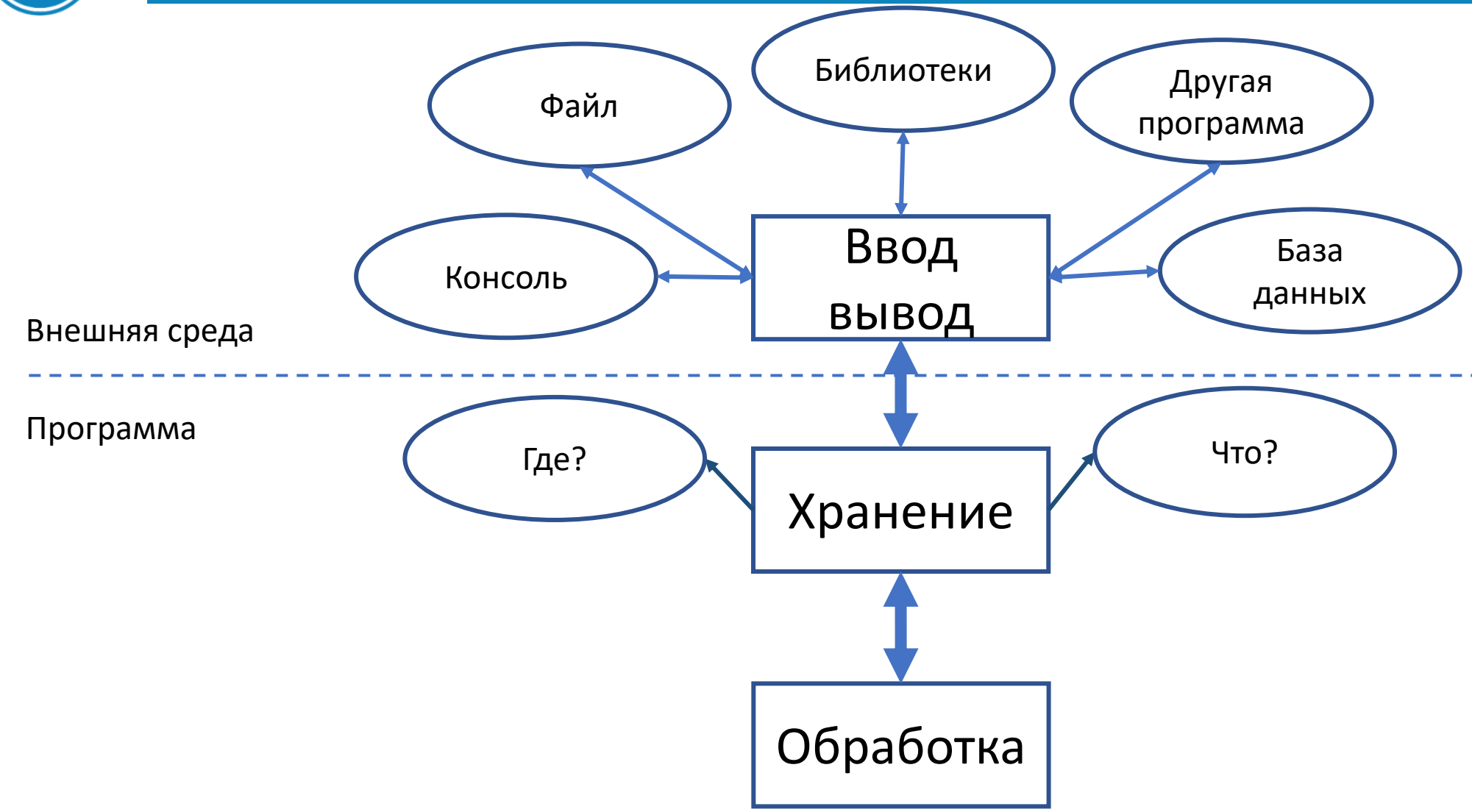
Факультатив по программированию на языке C

Занятие 5
Устройство памяти

План занятий

№	Тема	Описание
1	Введение в курс	Языки программирования. Основы работы с Linux.
2	Основы языка С	Написание и компиляция простейших программ с использованием gcc. Правила написания кода.
3	Компиляция	Разбиение программы на отдельные файлы. Make файлы. Компиляция.
4	Ввод данных. Библиотеки	Работа со вводом/выводом. Статические и динамические библиотеки.
5	Хранение данных. Память	Хранение процесса в памяти компьютера. Виртуальная память, сегментация. Секции программы.
6	Хранение данных.	Стек, куча. Типы данных. Преобразования типов. Gdb и отладка Хранение различных типов данных. Указатели. Передача аргументов в функцию по указателю.
7	Обработка данных	Безопасные функции. Битовые операции – сдвиги, логические операции. Битовые поля.
8	Язык ассемблера	Основы анализа программ на языке ассемблер.
9	Программирование под встраиваемые ОС	Работа с микрокомпьютером Raspberry Pi

Дерево языка

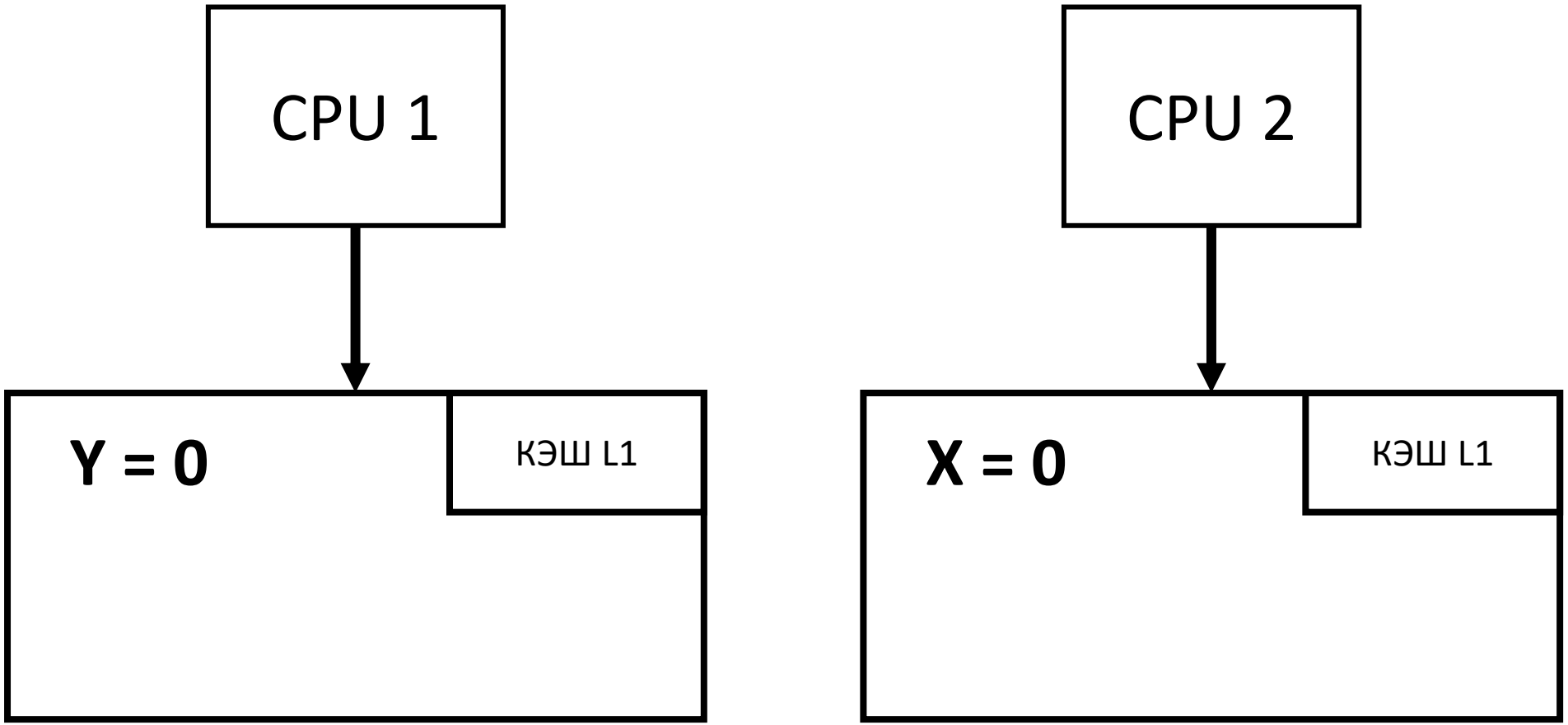


Store buffering problem

```
int x = 0;
int y = 0;

int r1 = 0;
int r2 = 0;

std::thread t1([&]() {
    x = 1;
    r1 = y;
});

std::thread t2([&]() {
    y = 1;
    r2 = x;
});
```

Чему могут быть равны r1 и r2 в результате работы программы?

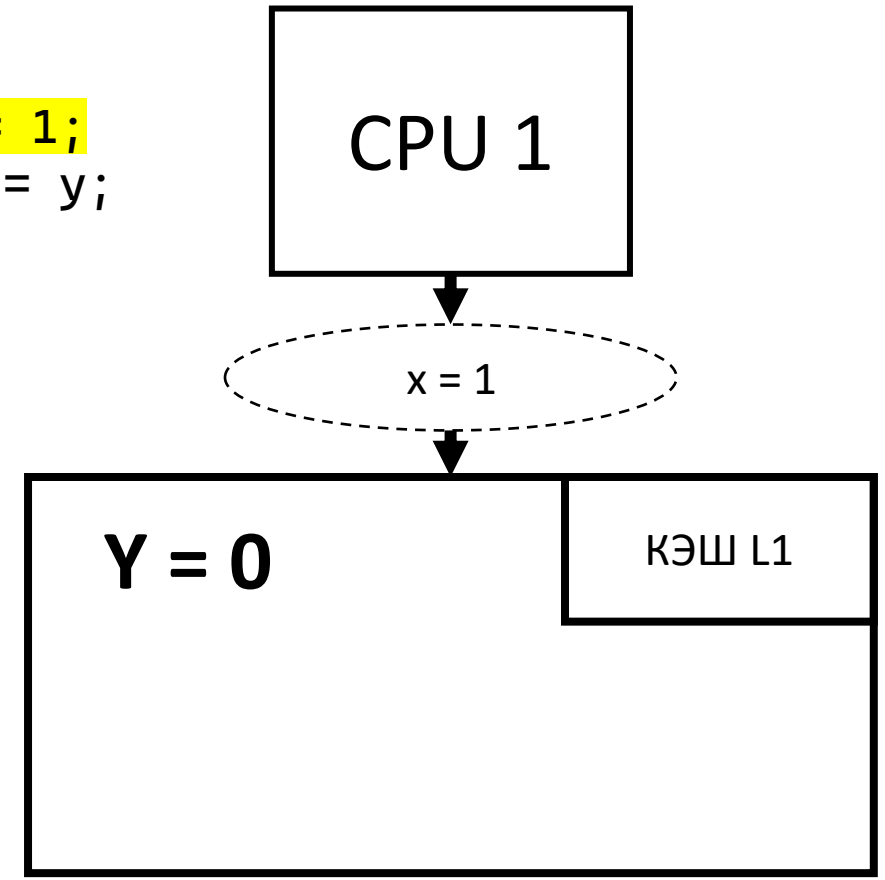
Store buffering problem

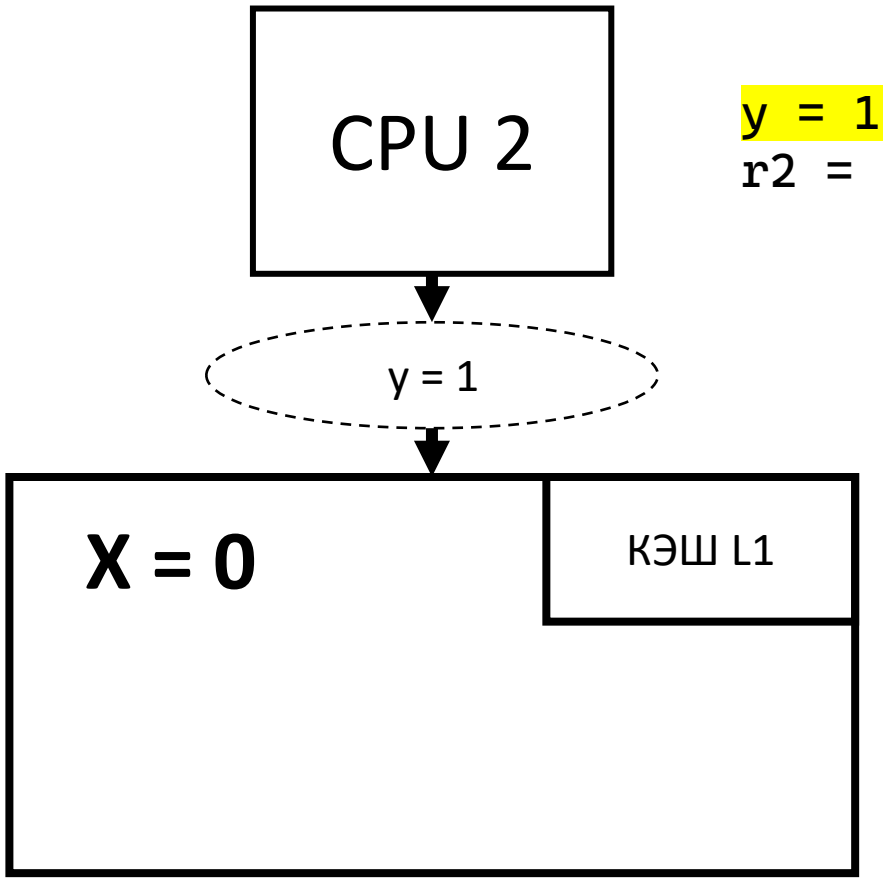


Программа выполняется на двух ядрах. Может получиться такая ситуация, что в первом кэше есть значение у, не X нет. На втором ядре наоборот.

Store buffering problem

```
x = 1;
r1 = y;
```

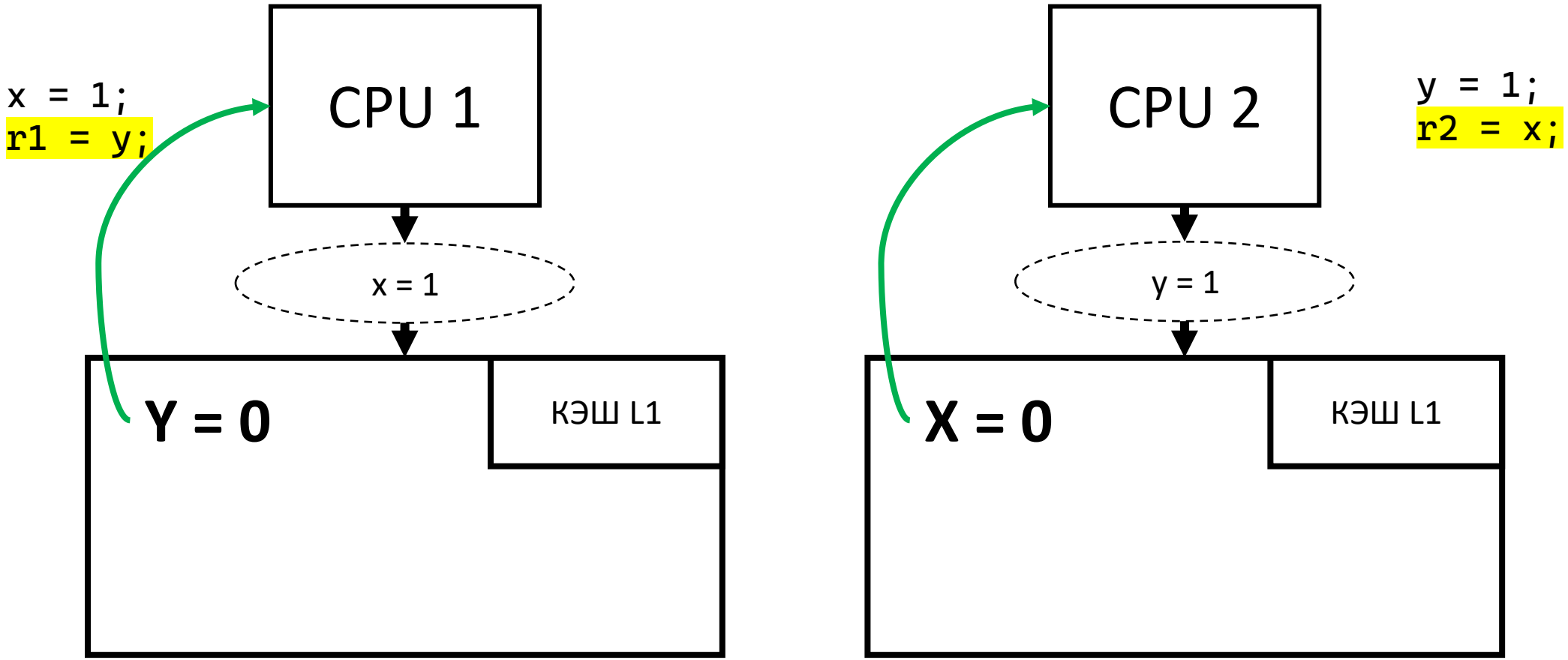





```
y = 1;
r2 = x;
```

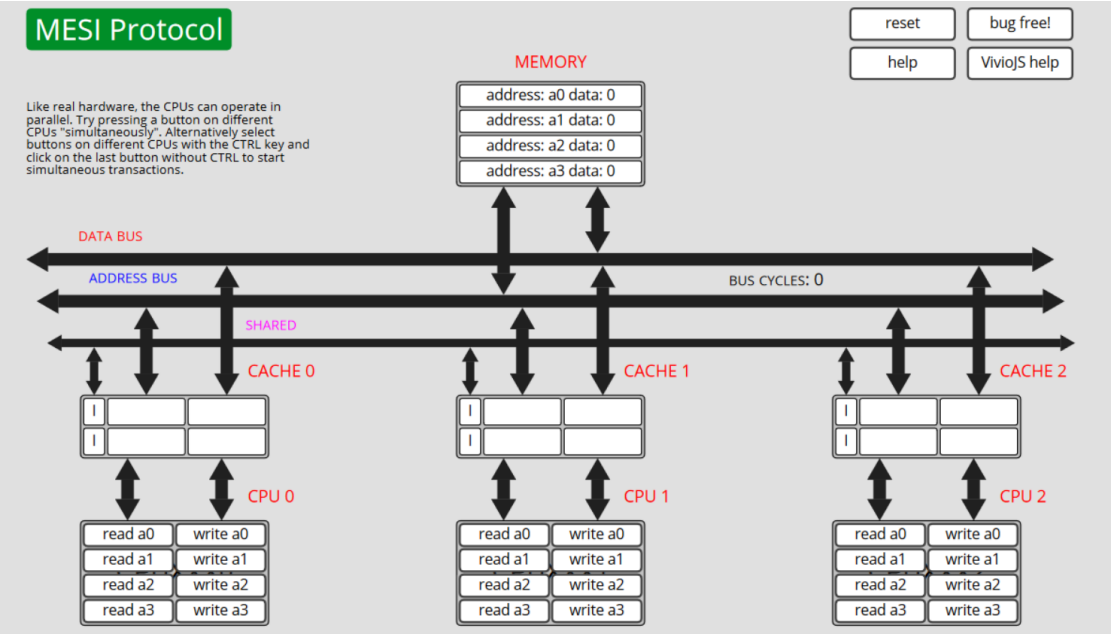
Т.к. значения X на ядре нет, то оно не сразу пишется в кэш, т.к. это затормозит всю систему (придется менять флаги во всех ядрах), а положится в промежуточный буфер. Далее программа продолжит последовательно выполняться.

Store buffering problem



Программа возьмет значения из кэша, которые равны 0. Архитектура допускает такой вариант, т.к. это следствие (путь и необычное) гонки данных, которая в программе не допускается

Интересные ссылки про кэш



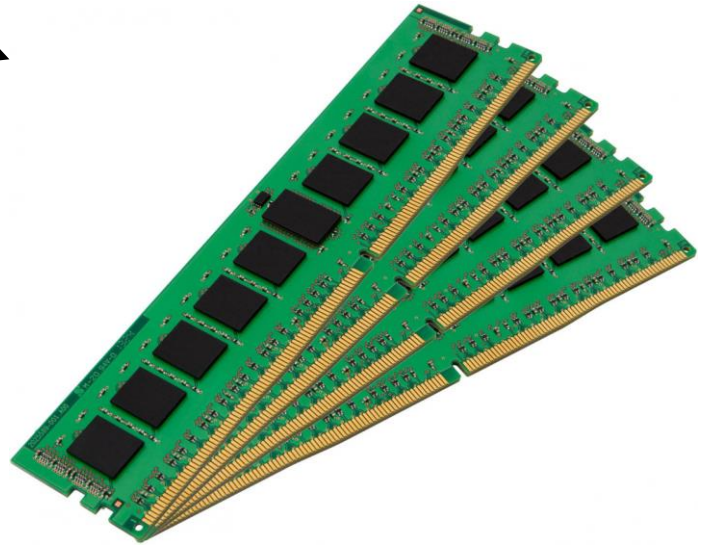
https://www.scss.tcd.ie/Jeremy.Jones/VivioJS/caches/MESI.htm

https://www.youtube.com/watch?v=_p__rtZ-cjI

A Primer on Memory Consistency and Cache Coherence, Daniel J. Sorin, Mark D. Hill, David A. Wood

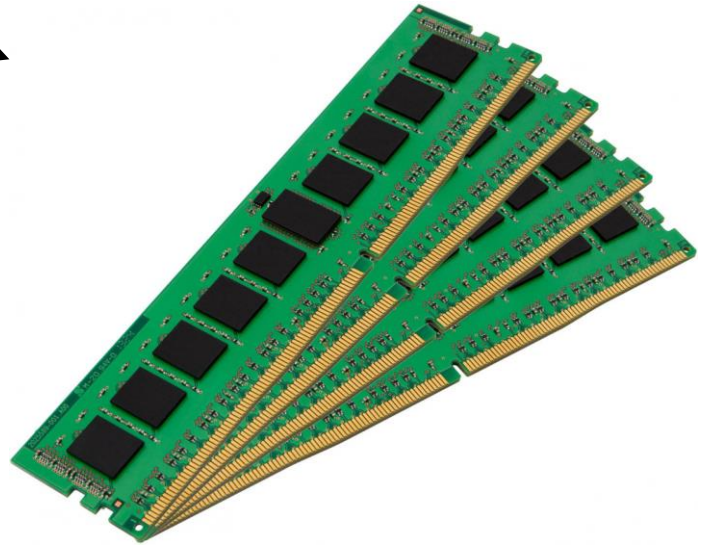
Где?

0x00004edc



Где?

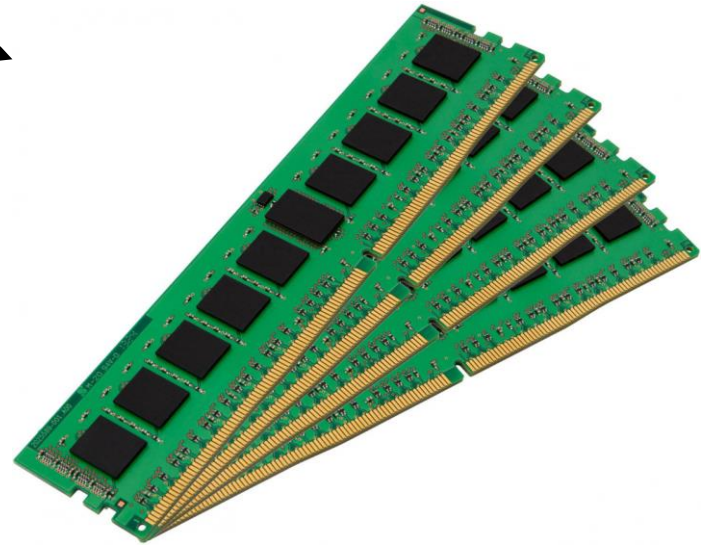
0x00004edc



Если это реальные адреса жесткого диска, то тогда зачем нужна оперативная память?

Где?

0x00004edc



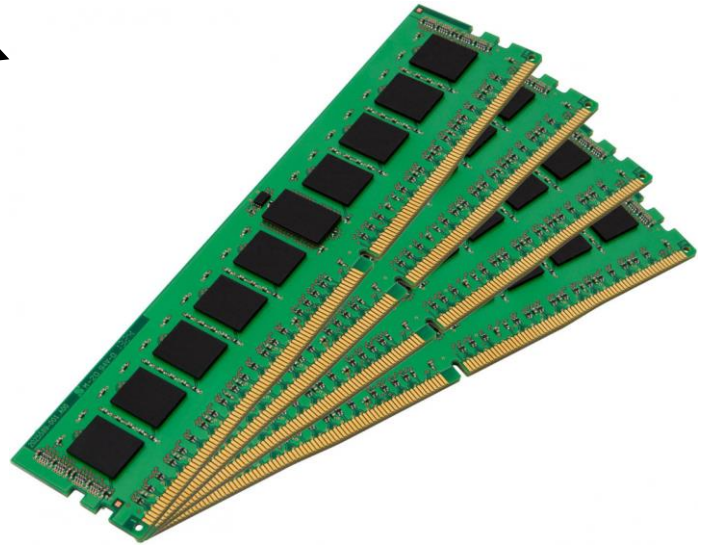
Если это реальные адреса жесткого диска, то тогда зачем нужна оперативная память?

Если это реальные адреса в оперативной памяти, то как тогда организовать одновременную работу двух процессов?

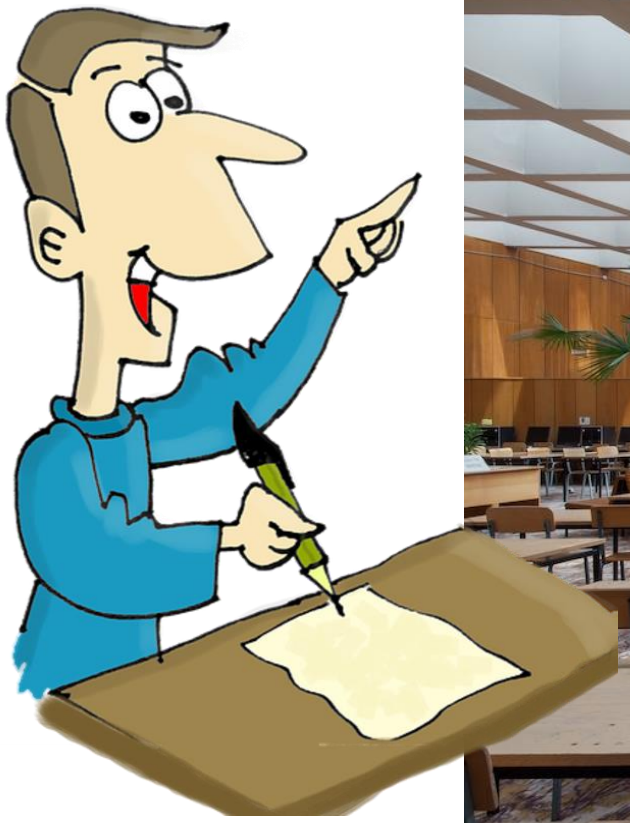
Где?

0x00004edc

?

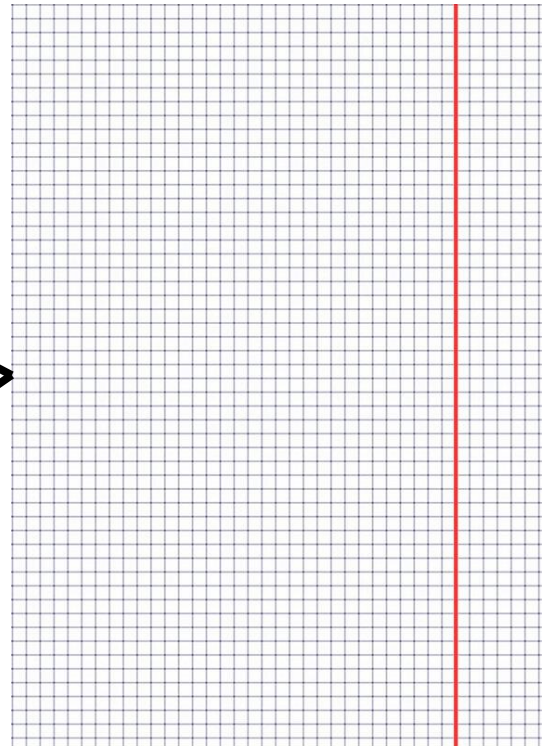


Пример - библиотека



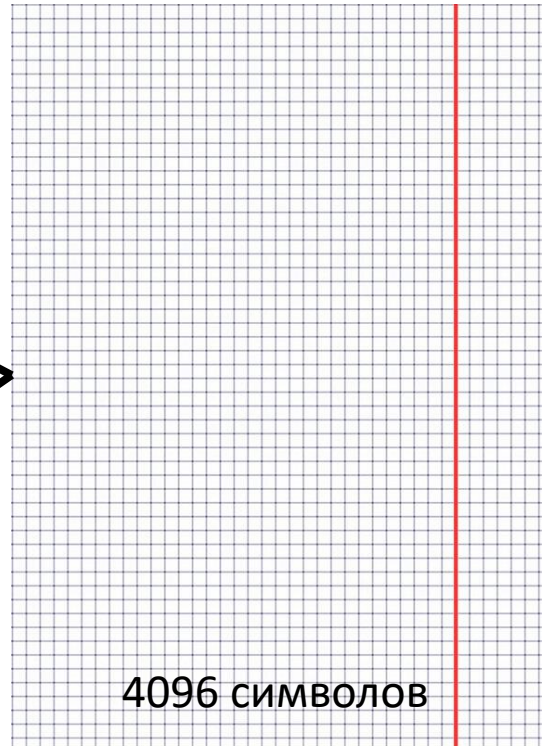
Пример - библиотека

С точки зрения **библиотекаря**



Пример

~ 2 млн страниц

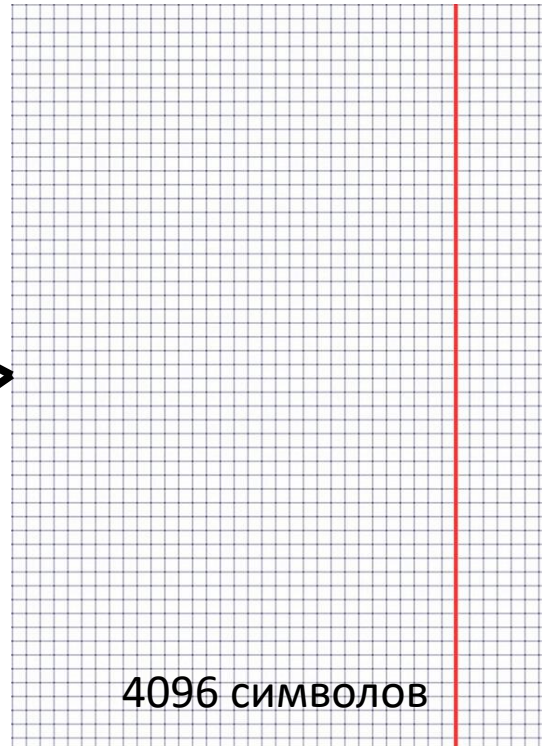


~ 2 млрд страниц

Пример

~ 2 млн страниц

~ 2 млрд страниц

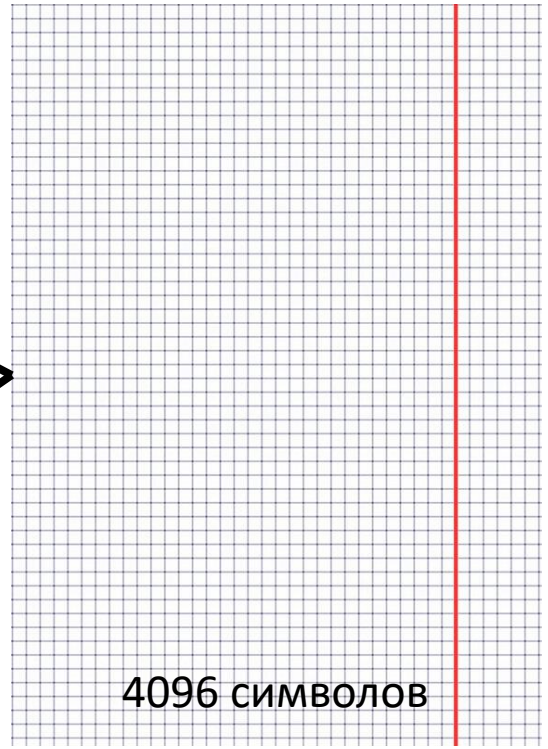


Что будет, если нужно записать 4097 символов?

Пример

~ 2 млн страниц

~ 2 млрд страниц



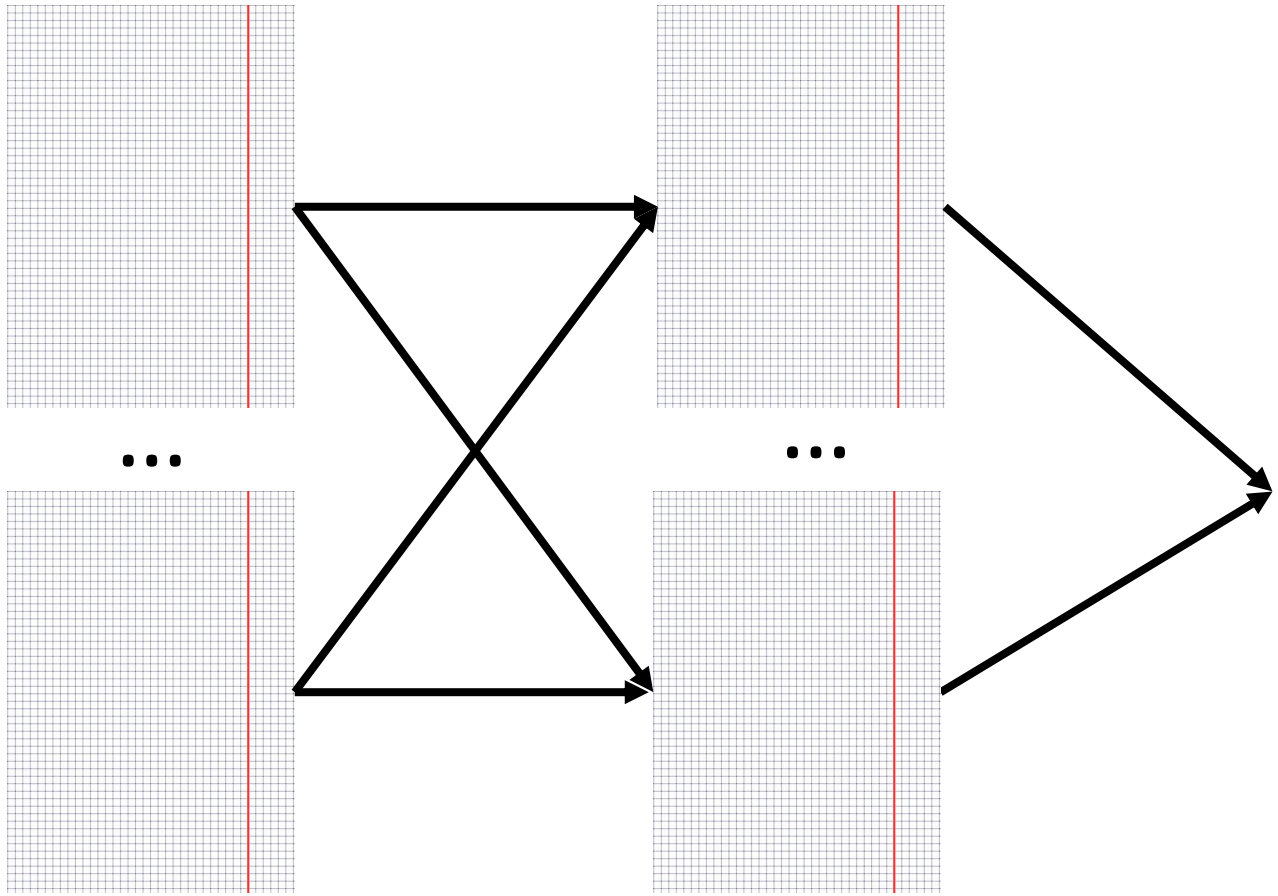
Что будет, если нужно записать 4097 символов?

Как не запутаться?

Пример

Каталог оглавлений

Оглавление



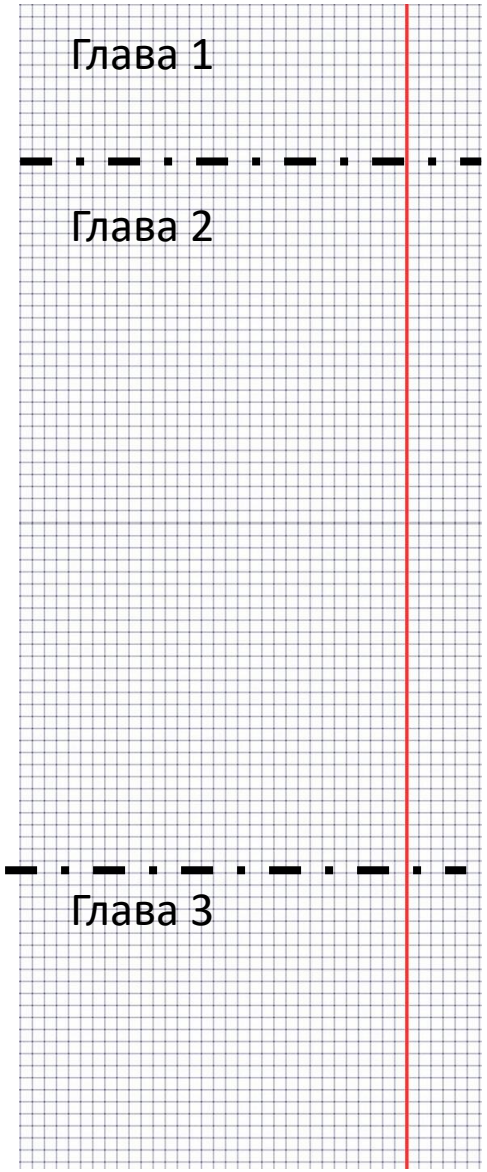
Пример

Каталог	Таблица	Смещение

Каталог оглавлений

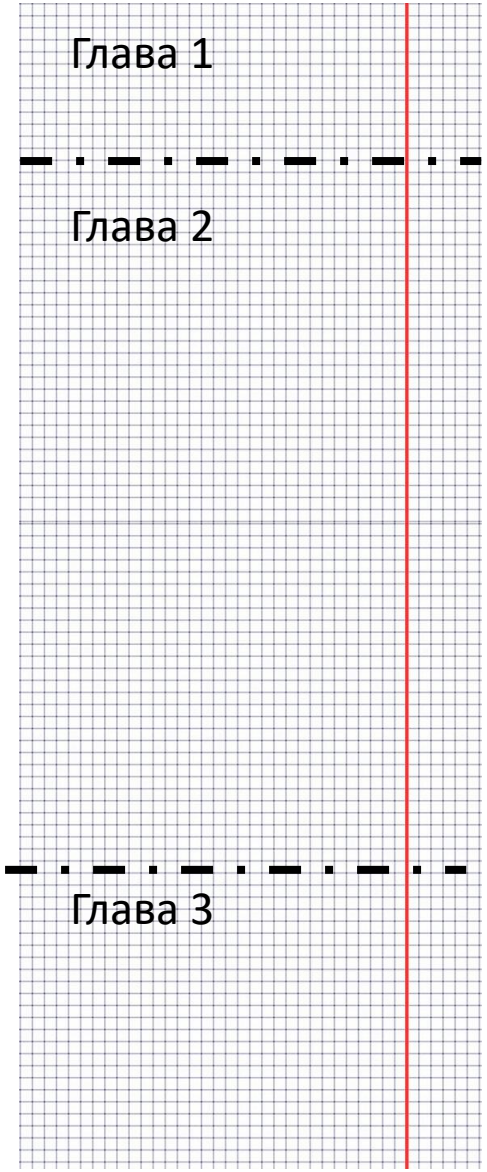
Оглавление

Пример



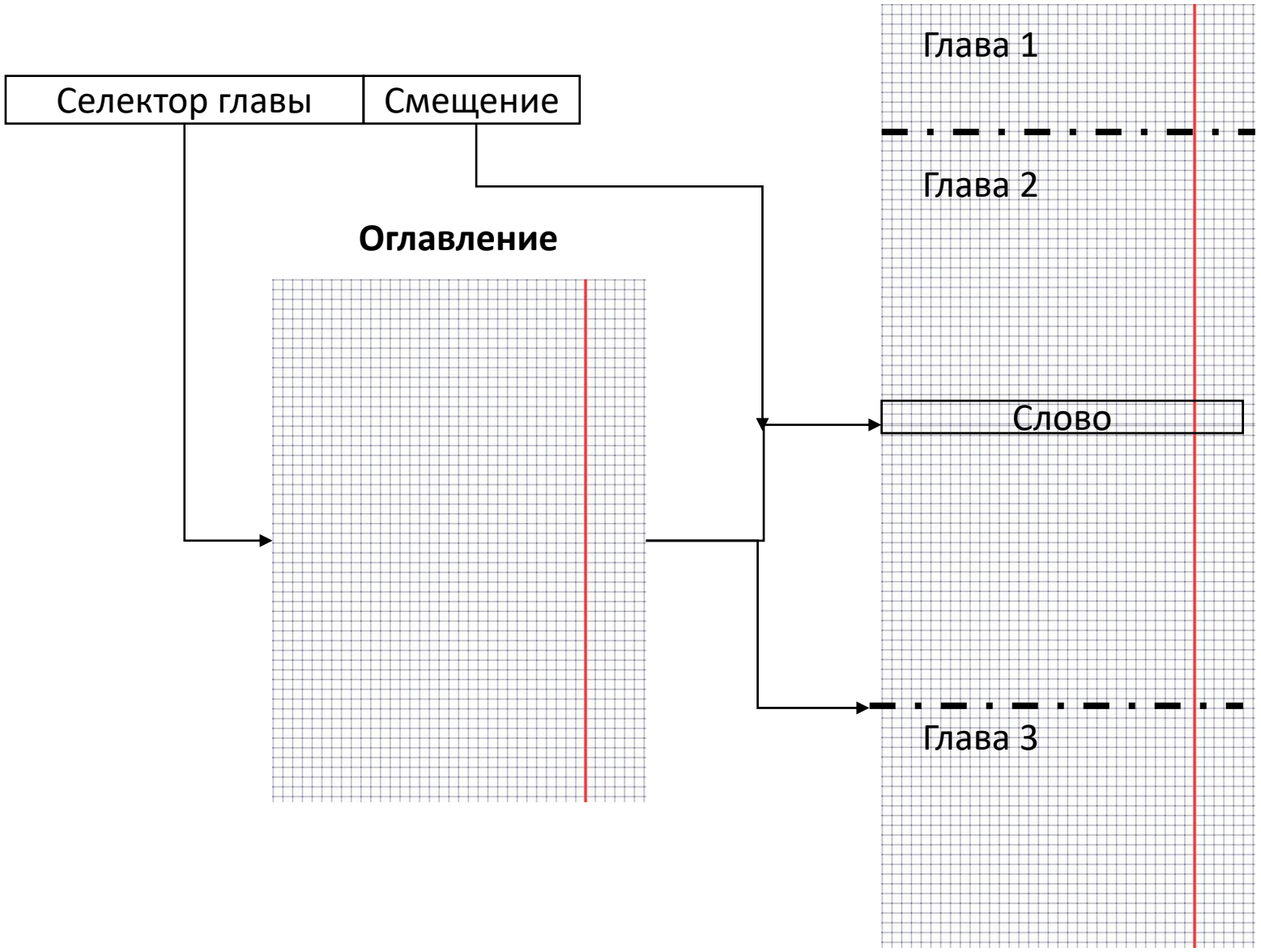
С точки зрения **писателя**

Пример

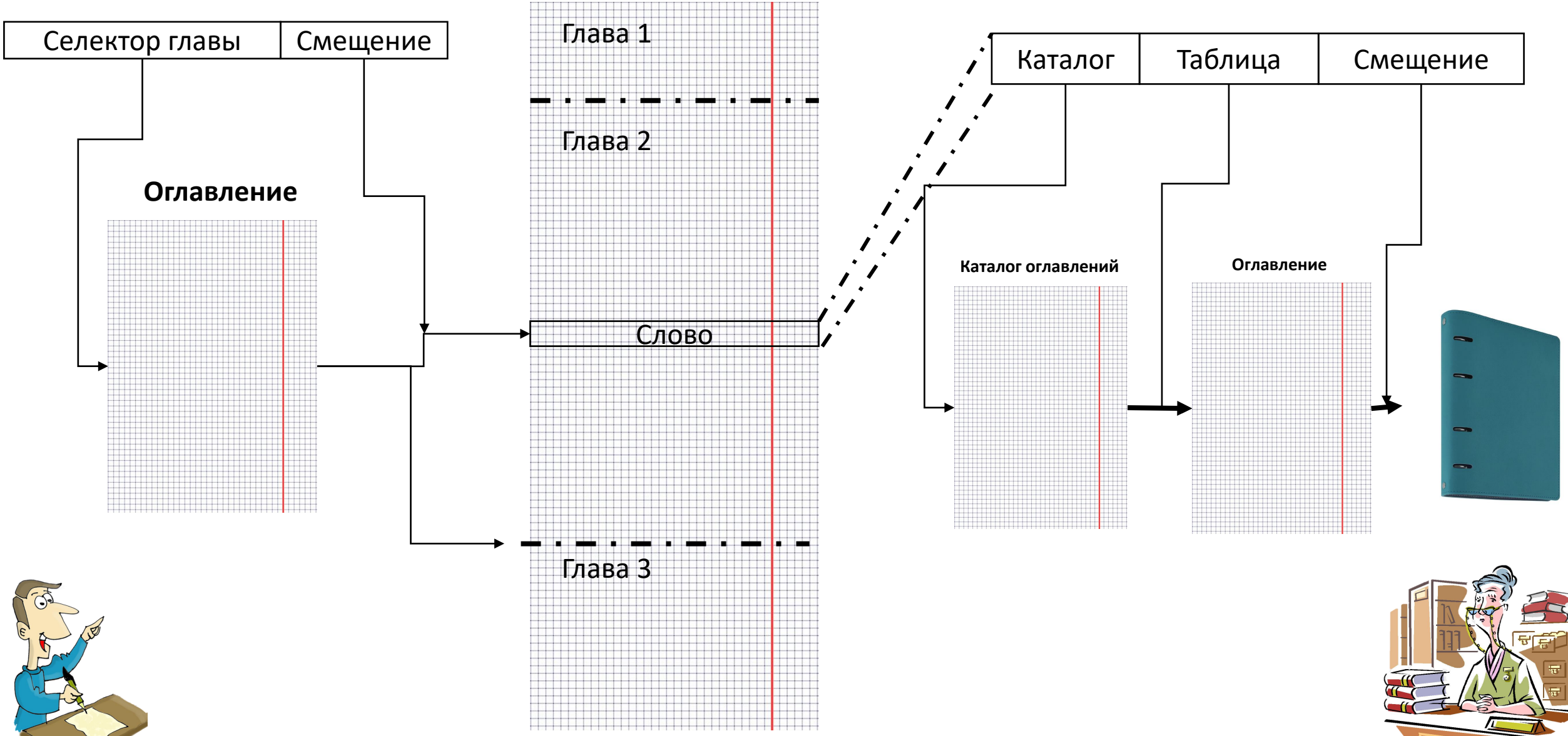


Размеры глав различаются!!!

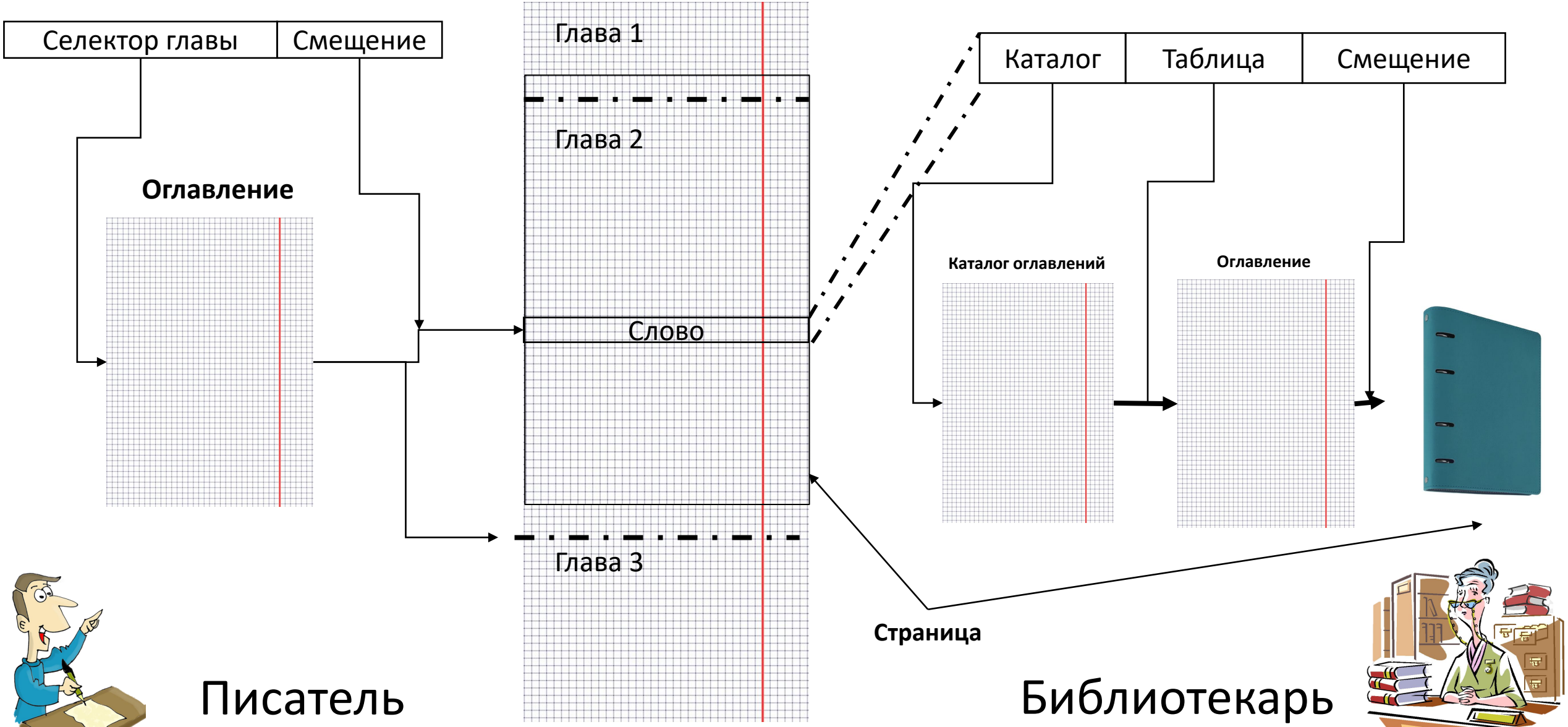
Пример


Селектор главы
Смещение
Оглавление
Глава 1
Глава 2
Слово
Глава 3

Пример


Селектор главы
Смещение
Оглавление
Глава 1
Глава 2
Слово
Глава 3
Каталог
Таблица
Смещение
Каталог оглавлений
Оглавление

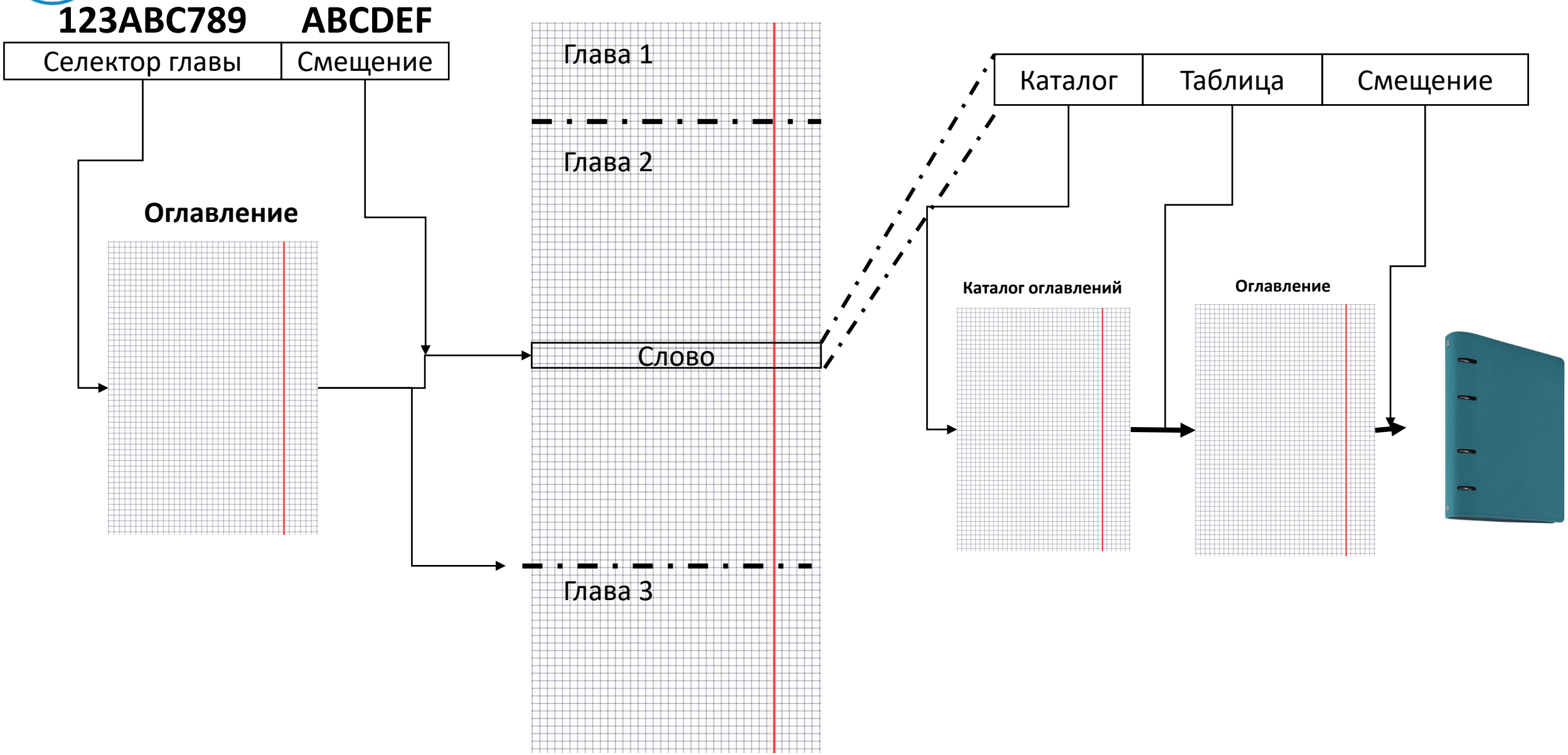
Пример



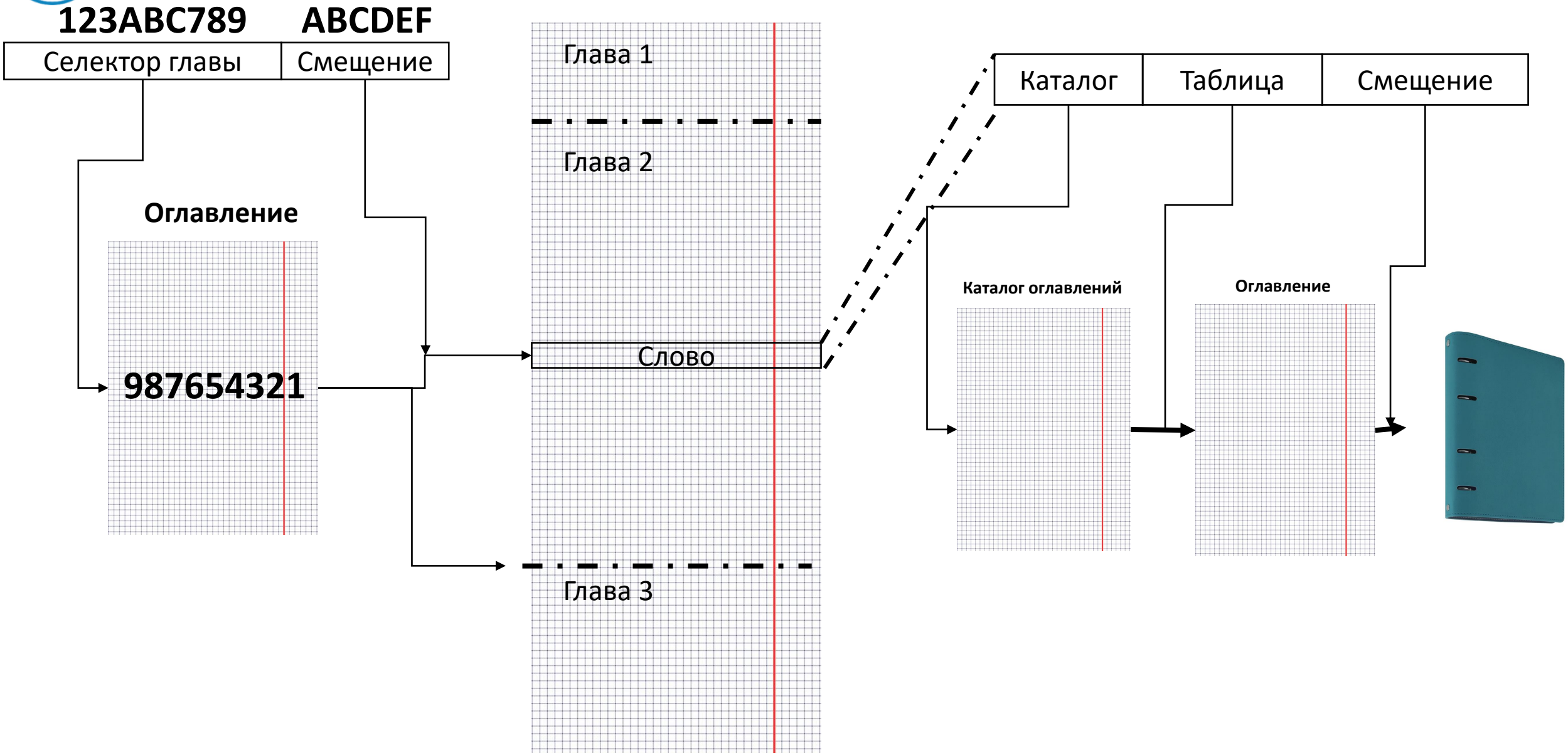
Писатель

Библиотекарь

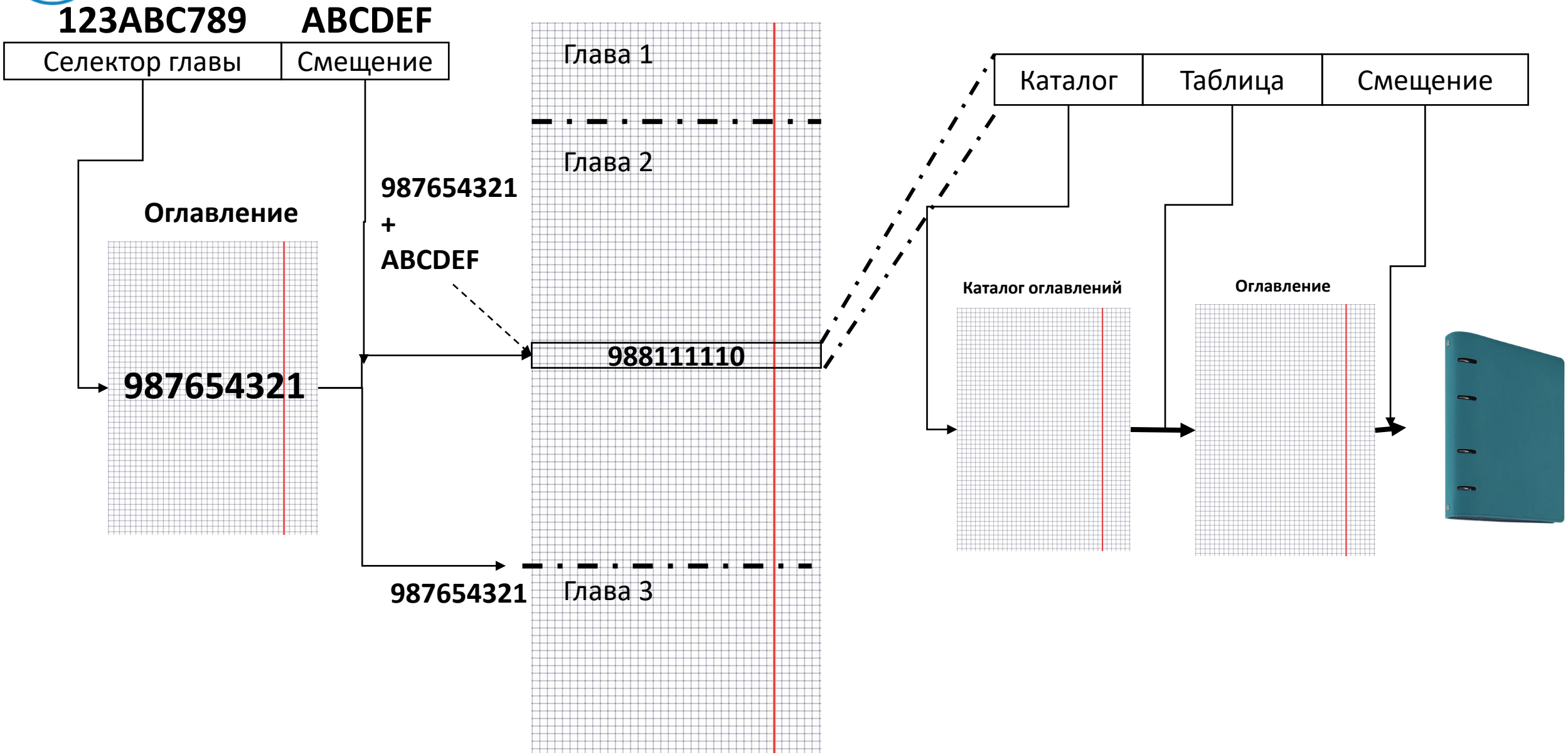
Пример


123ABC789
ABCDEF
Селектор главы
Смещение
Оглавление
Глава 1
Глава 2
Слово
Глава 3
Каталог
Таблица
Смещение
Каталог оглавлений
Оглавление

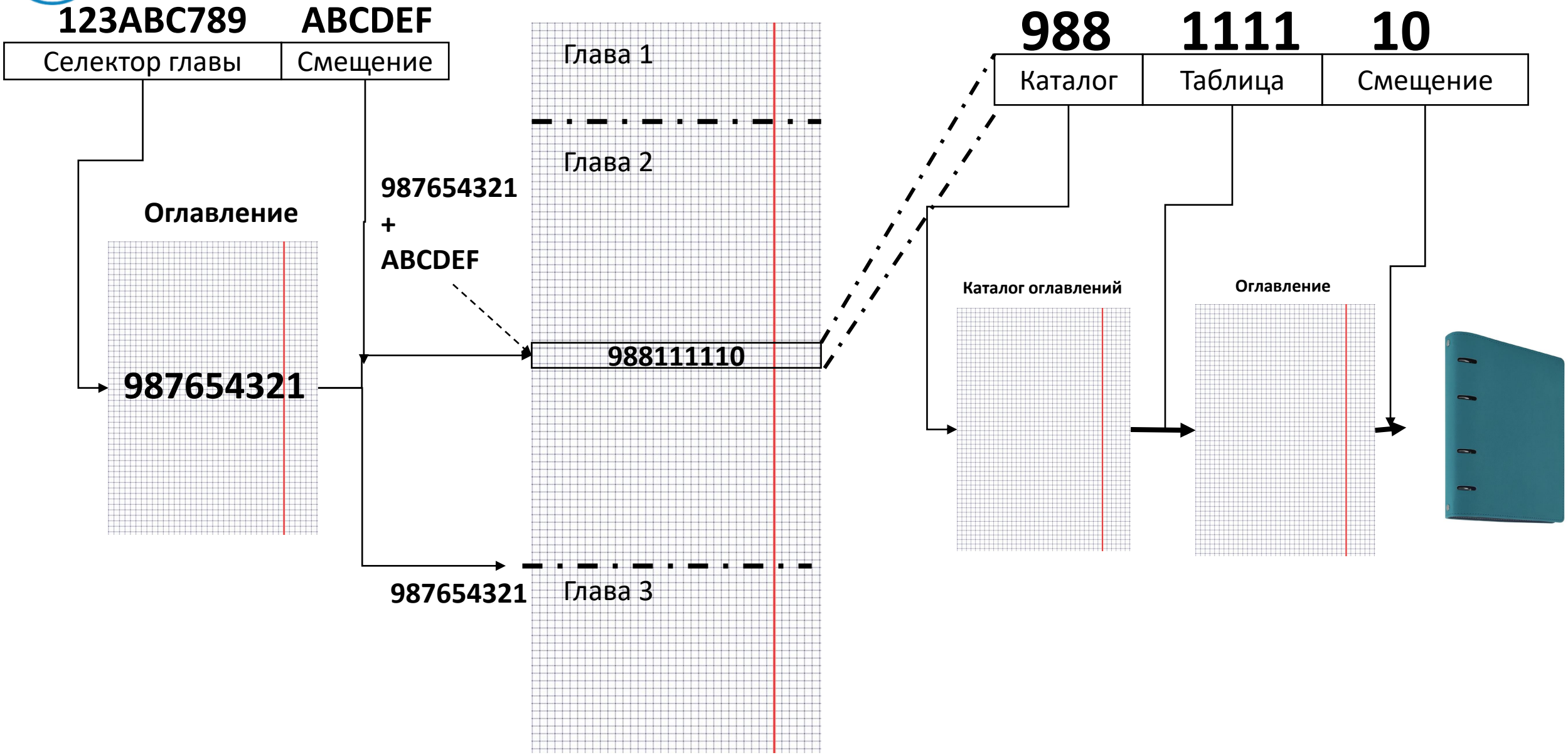
Пример


123ABC789
ABCDEF
Селектор главы
Смещение
Оглавление
987654321
Глава 1
Глава 2
Слово
Глава 3
Каталог
Таблица
Смещение
Каталог оглавлений
Оглавление

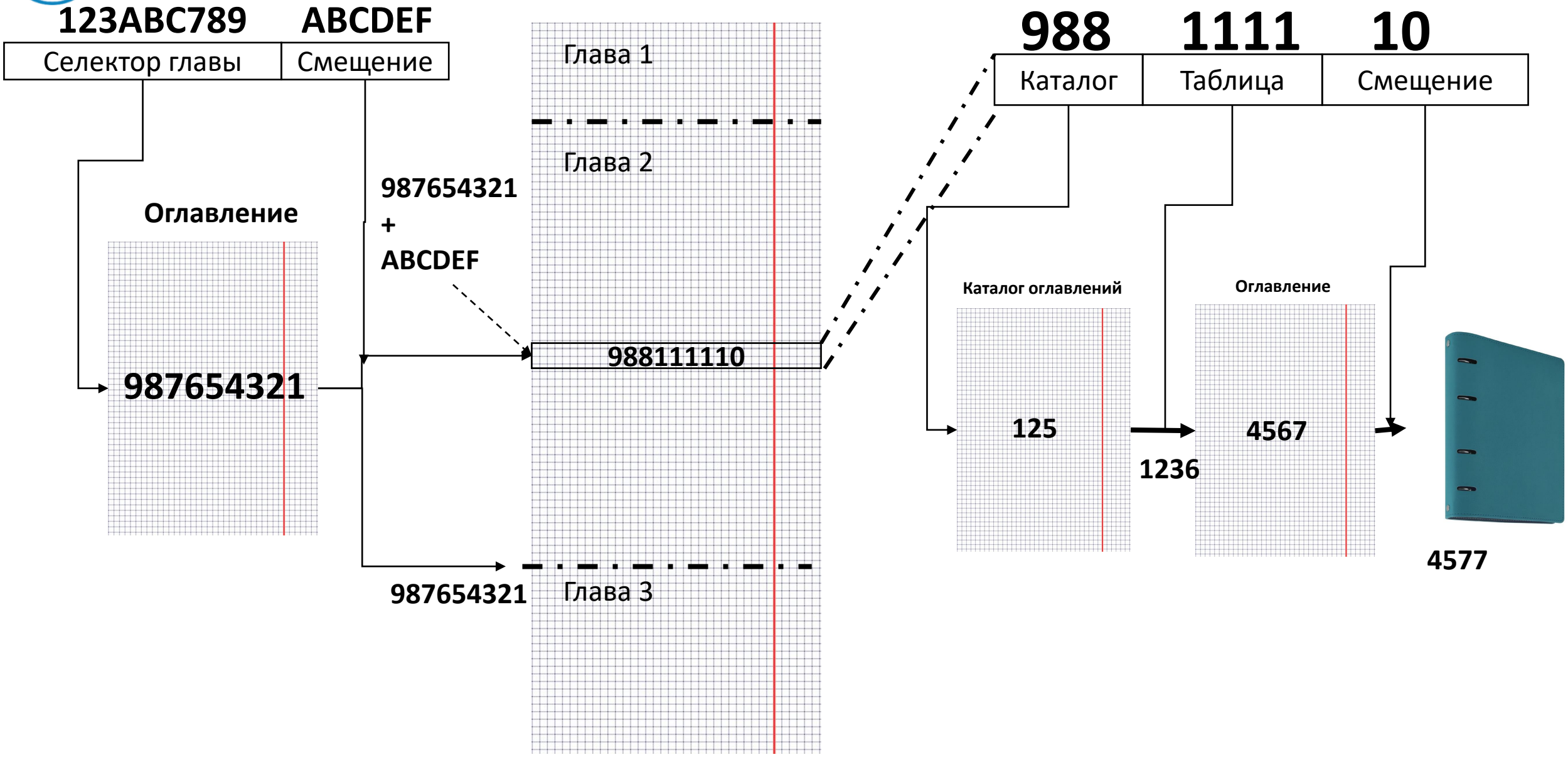
Пример


123ABC789
ABCDEF
Селектор главы
Смещение
Оглавление
987654321
987654321
+
ABCDEF
Глава 1
Глава 2
988111110
987654321
Глава 3
Каталог
Таблица
Смещение
Каталог оглавлений
Оглавление

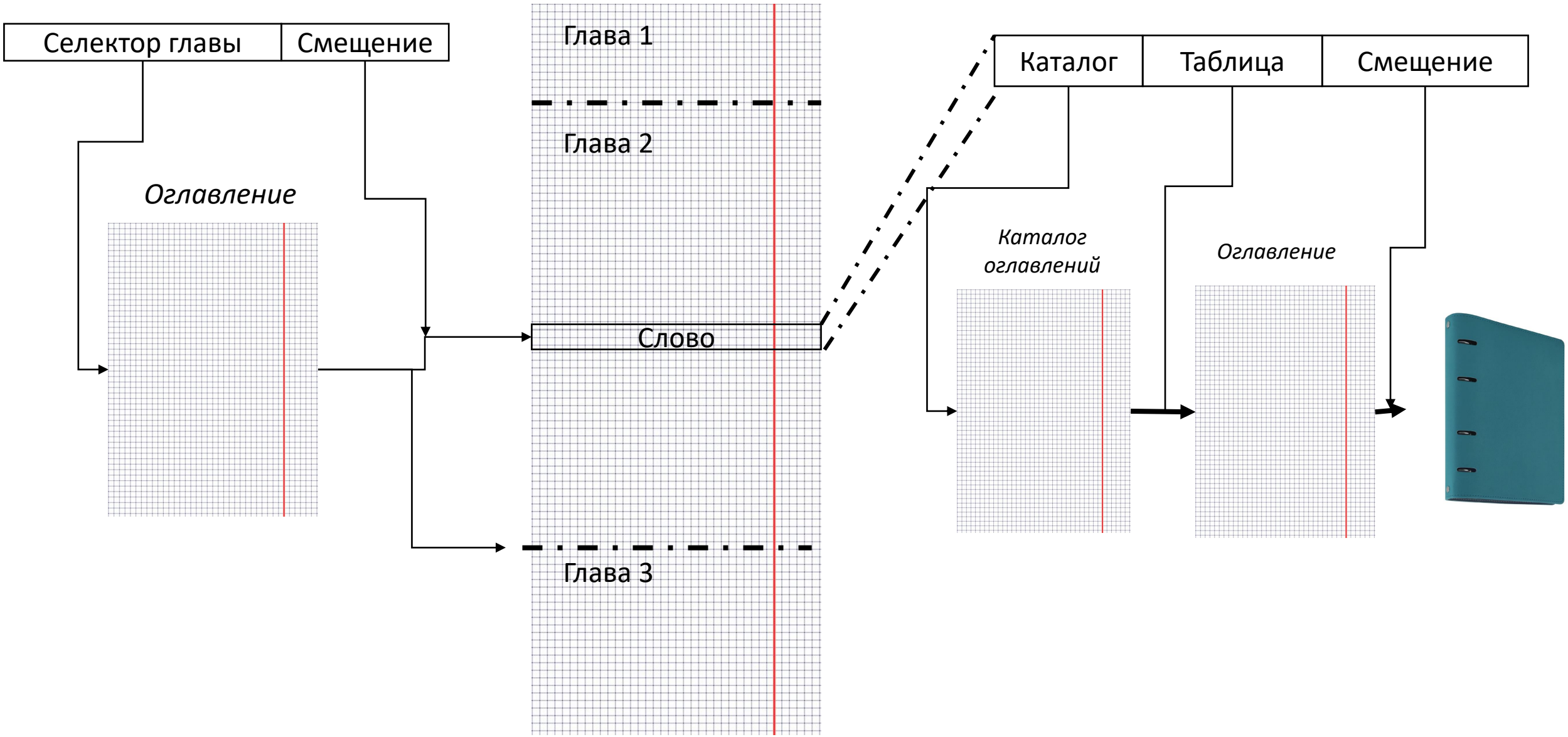
Пример


123ABC789
ABCDEF
Селектор главы
Смещение
Оглавление
987654321
987654321
+
ABCDEF
Глава 1
Глава 2
988111110
987654321
Глава 3
988
1111
10
Каталог
Таблица
Смещение
Каталог оглавлений
Оглавление

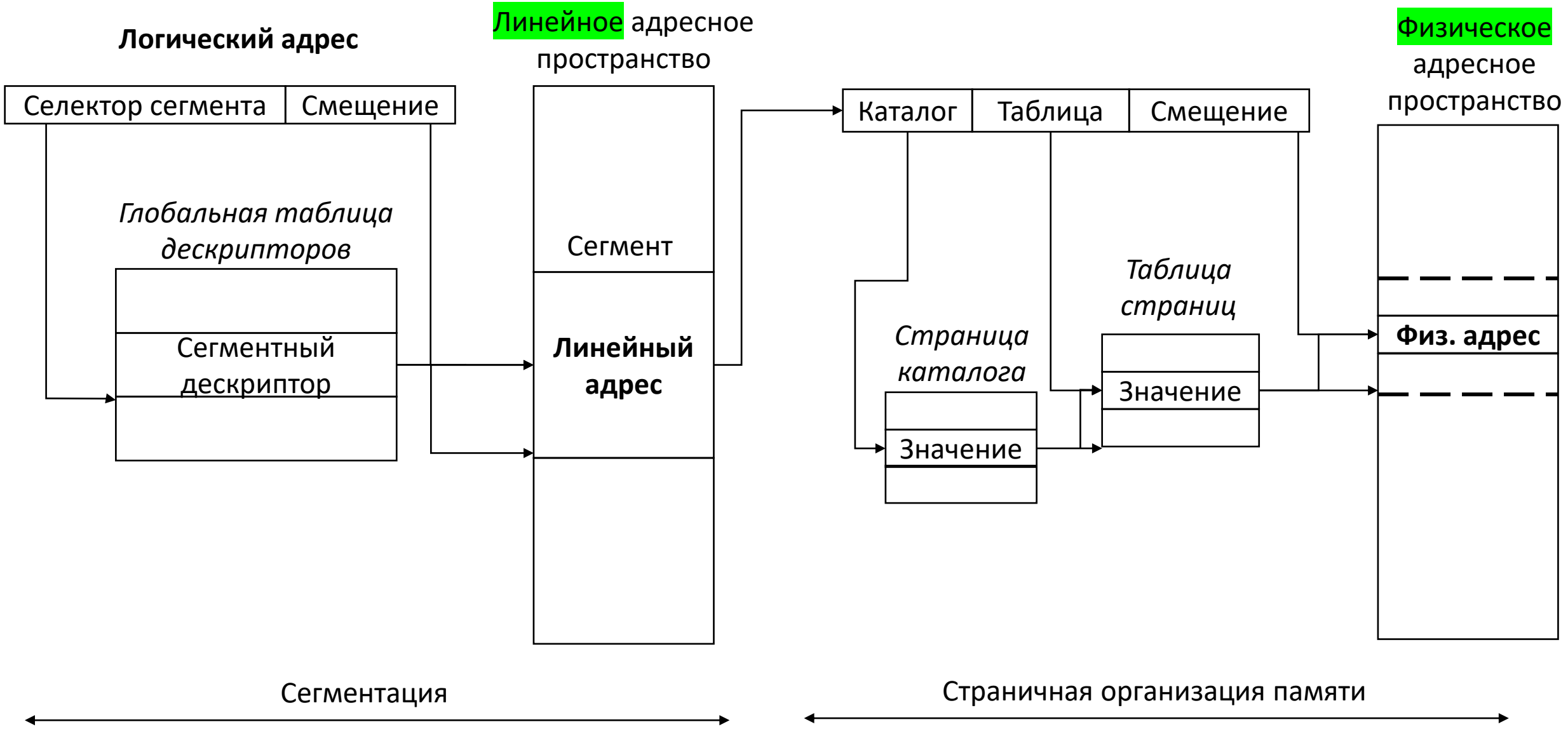
Пример


123ABC789
ABCDEF
Селектор главы
Смещение
Оглавление
987654321
987654321
+
ABCDEF
Глава 1
Глава 2
988111110
987654321
Глава 3
988
1111
10
Каталог
Таблица
Смещение
Каталог оглавлений
125
1236
Оглавление
4567
4577

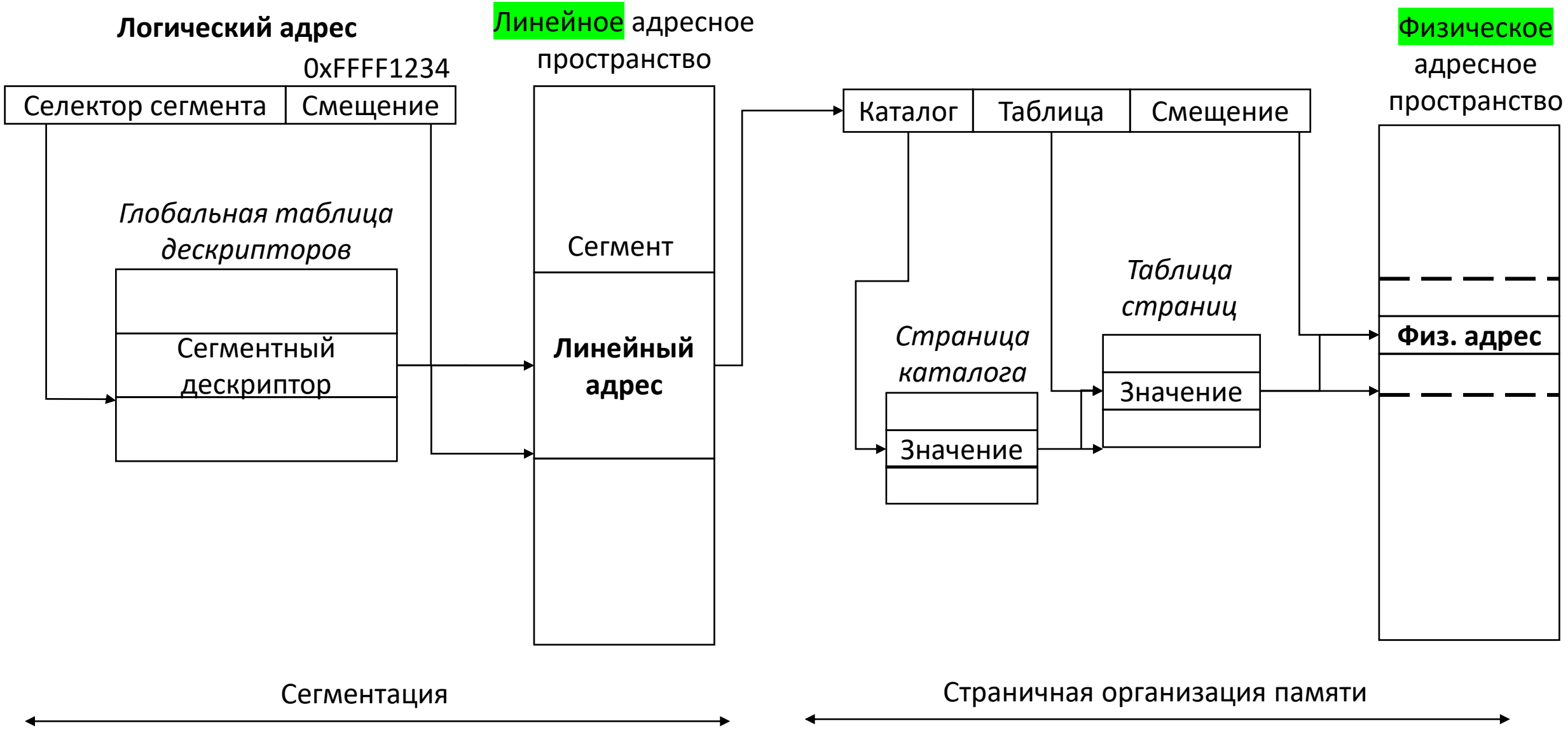
Пример


Селектор главы
Смещение
Оглавление
Глава 1
Глава 2
Слово
Глава 3
Каталог
Таблица
Смещение
Каталог оглавлений
Оглавление

Устройство памяти (x86, 32 bit)



Устройство памяти (x86, 32 bit)



Промежуточные выводы!

1) Память поделена на страницы
2) Преобразование:

Логический адрес -> Линейный адрес -> Физический адрес

Модульное программирование

Работа с портом

- Настройка порта
- …
- Чтение данных
- …
- Отправка сигнала завершения
- …

Работа с буфером

- Запись данных в буфер
- …
- Чтение данных из буфера
- …

Работа с файлом

- Открытие файла
- …
- Запись данных
- …
- Закрытие файла
- …

main

Анализ данных

…

Удобно работать с программой

Модульное программирование

Работа с портом

- Настройка порта
- …
- Чтение данных
- …
- Отправка сигнала завершения
- …

Работа с буфером

- Запись данных в буфер
- …
- Чтение данных из буфера
- …

Работа с файлом

- Открытие файла
- …
- Запись данных
- …
- Закрытие файла
- …

main

Анализ данных

…

Как бы вы предложили разбить текст программы для удобства работы загрузчика?

Удобно работать с программой

Секции программы

```
#include <stdio.h>
#include <stdlib.h>

static int bss[1024];
static int data[1024] = {1024};
static const char rodata[8192] = {"rodata"};

int main(void)
{
	int stack;
	int *heap = (int*)malloc(1024*sizeof(int));
	free(heap);
	return 0;
}
```

Секции программы

К
О
Д

```
#include <stdio.h>
#include <stdlib.h>

static int bss[1024];
static int data[1024] = {1024};
static const char rodata[8192] = {"rodata"};

int main(void)
{
    int stack;
    int *heap = (int*)malloc(1024*sizeof(int));
    free(heap);
    return 0;
}
```

→ Неинициализированная глобальная переменная

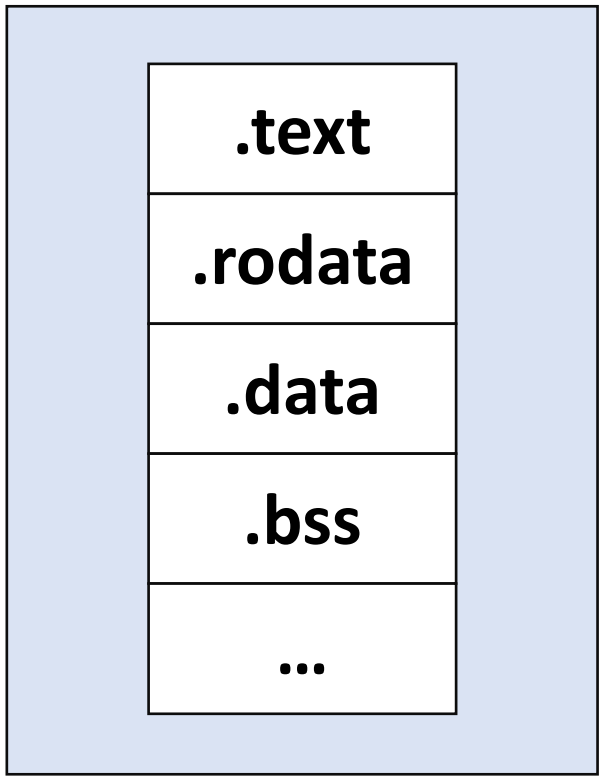
→ Инициализированная глобальная переменная

→ Инициализированная глобальная константа

→ Статическая переменная

→ Динамический массив

Секции программы



Код программы

Глобальная константы

Инициализированная глобальные переменные

Неинициализированная глобальные переменные

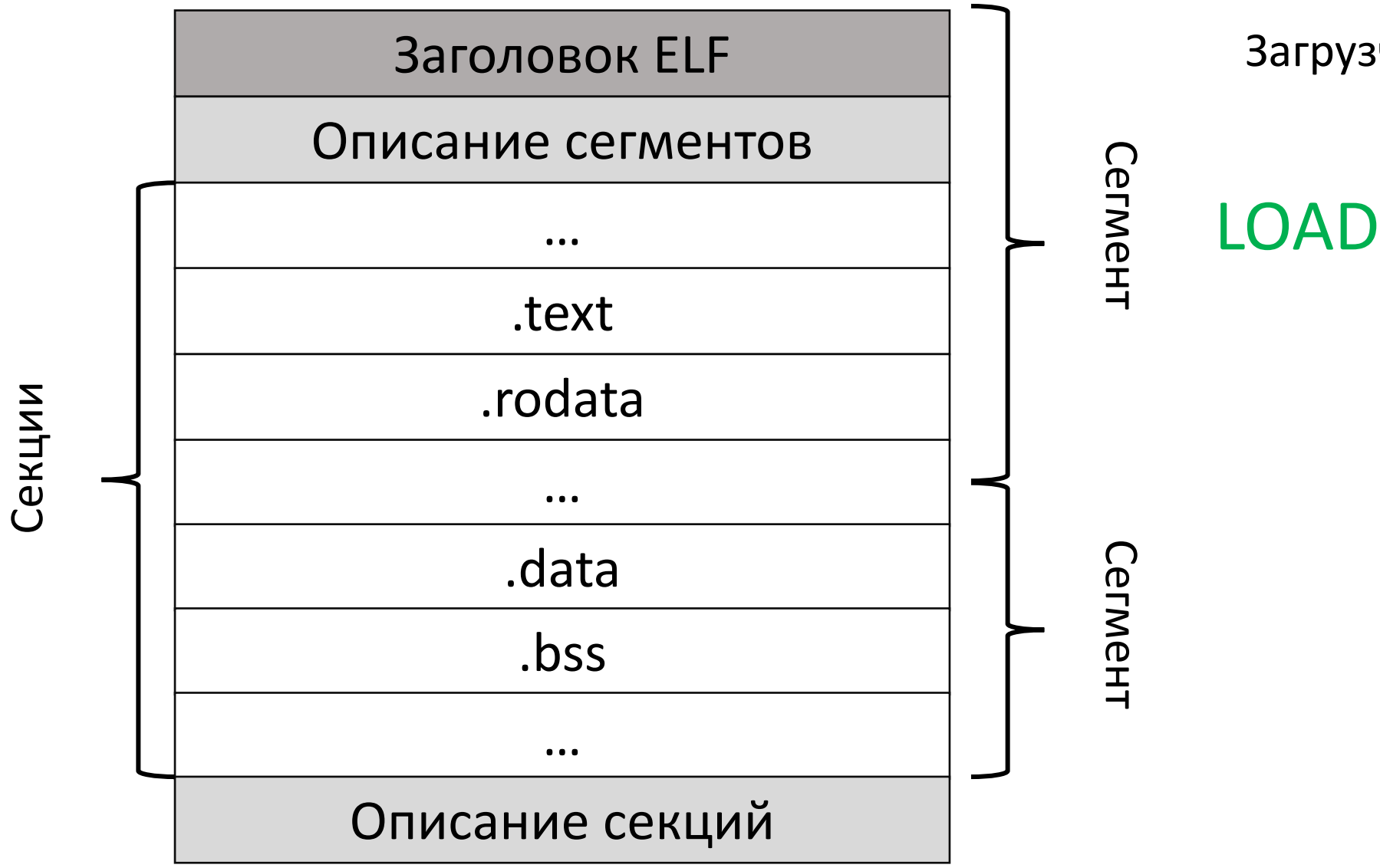
stack Стек

heap Куча

Секции и сегменты программы

Компоновщик

Загрузчик



Исходный код программ

https://github.com/SergeyBalabaev

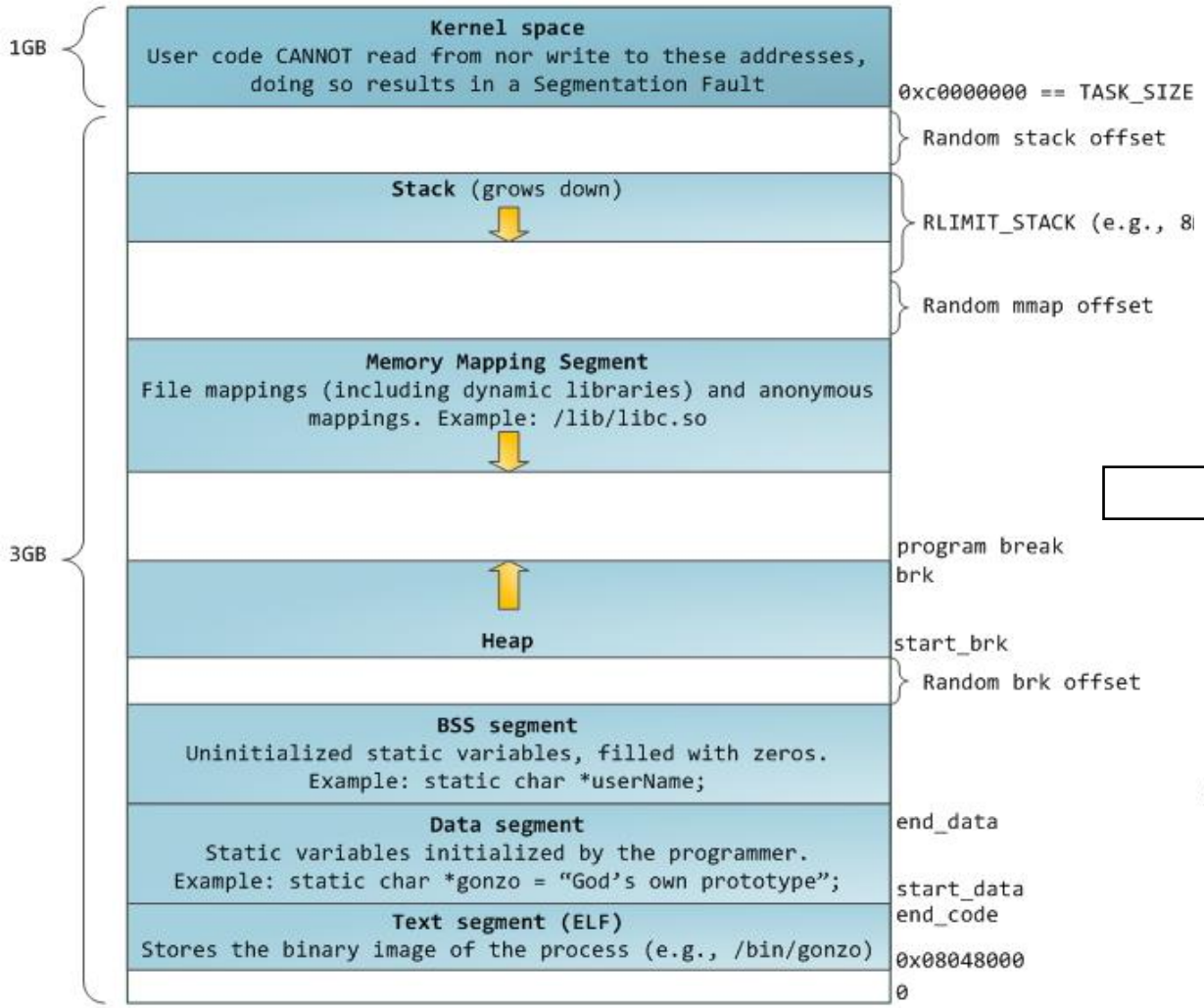
Elective-C-Programming-Language → *Lesson5/Memory*

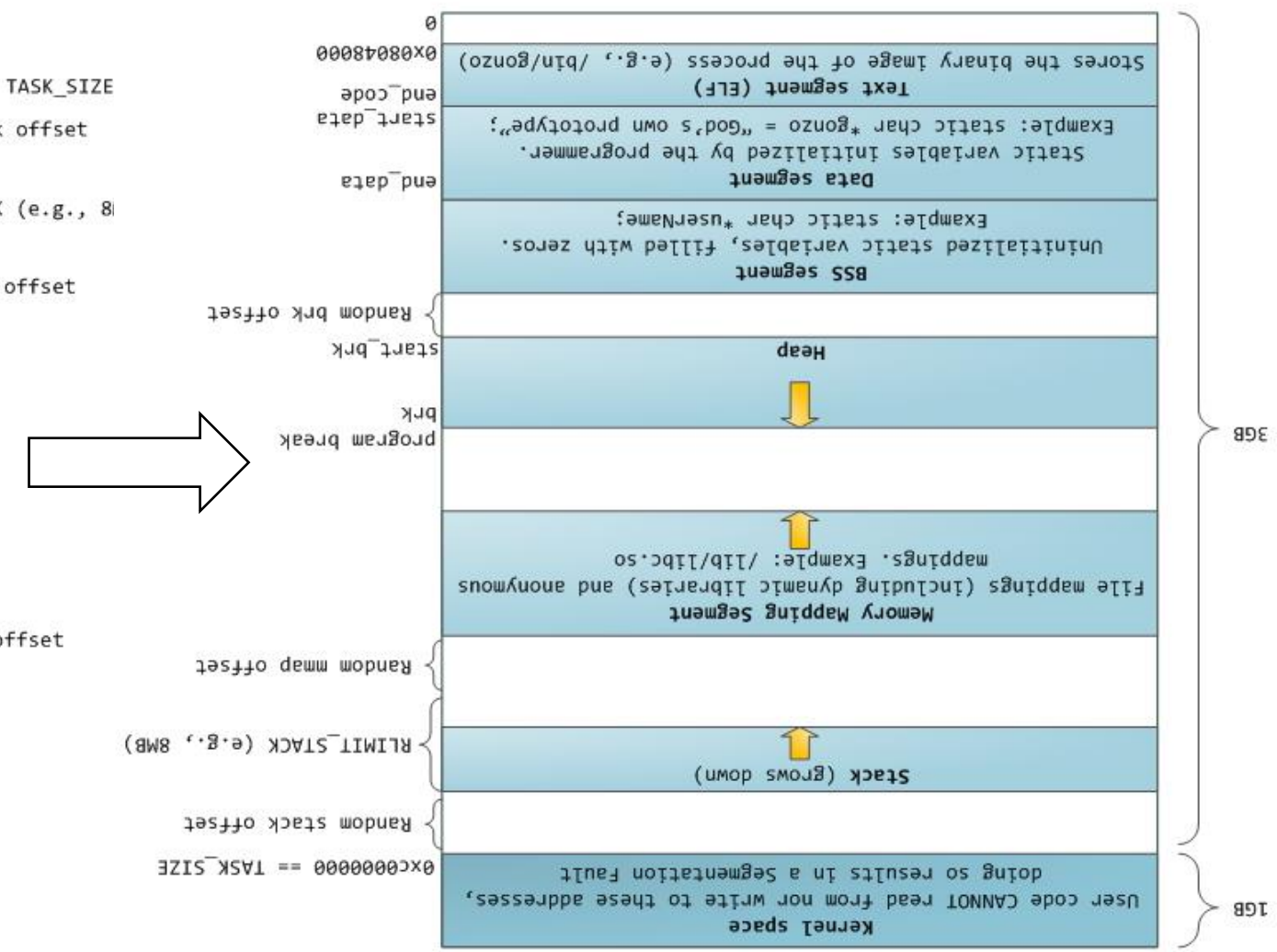
Пример

```
gcc memory1.c -o memory
size memory
```

Изменим размер bss, data, rodata
Выполним команду size

Секции программы





Рассмотрим пример

Elective -> lesson5 -> Memory

```
gcc memory.c -o memory
./memory
```

Спасибо за внимание!