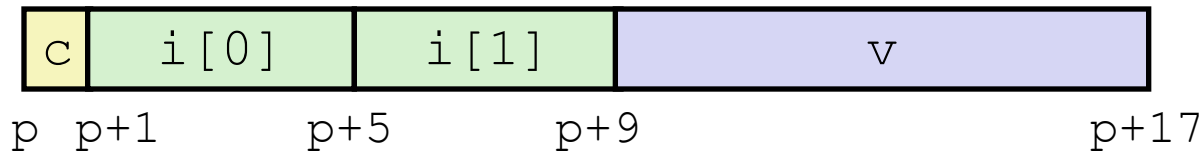


# Лекция 10

14 марта

# Выравнивание полей в структурах

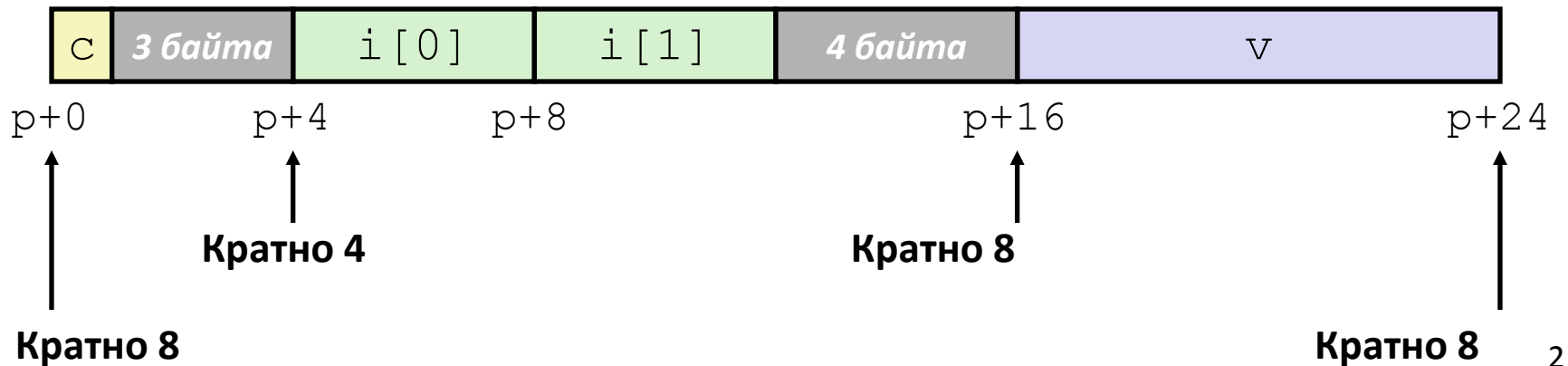
- Невыровненные данные



```
struct S1 {
    char c;
    int i[2];
    double v;
} *p;
```

- Выровненные данные

- Если примитивный тип данных требует  $K$  байт
- Адрес должен быть кратен  $K$



# Почему выравнивают данные

- Выровненные данные
  - Примитивный тип данных требует  $K$  байт
  - Адрес должен быть кратен  $K$
  - Обязательно для некоторых архитектур
  - Для IA32 требование имеет рекомендательный характер
    - Требования **различаются** для IA32 Linux, x86-64 Linux, и Windows
- Причины
  - Доступ к физической памяти осуществляется блоками (выровненными) по 4 или 8 байт (зависит от аппаратуры)
    - Эффективность теряется при обращении к данным, расположенным в двух блоках
    - Виртуальная память...
- Компилятор
  - Расставляет пропуски между полями для сохранения выравнивания

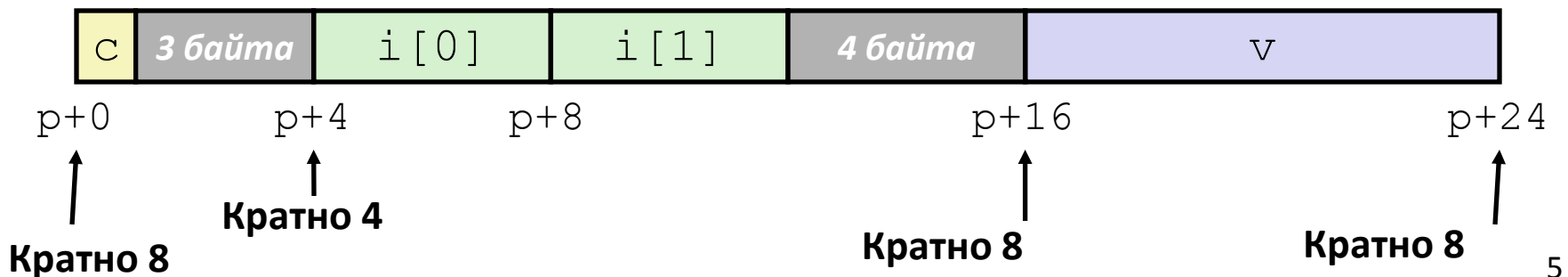
# Правила выравнивания (IA32)

- 1 байт : `char`, ...
  - Ограничений нет
- 2 байта : `short`, ...
  - Младший бит адреса должен быть  $0_2$
- 4 байта : `int`, `float`, `char *`, ...
  - Два младших бита адреса должны быть  $00_2$
- 8 байт : `double`, ...
  - Windows ( и другие ...):
    - Младшие три бита адреса должны быть  $000_2$
  - Linux:
    - Два младших бита адреса должны быть  $00_2$
    - Т.е. рассматриваются как и 4-байтные примитивные типы данных
- 12 байт : `long double` (gcc)
  - Windows, Linux:
    - Два младших бита должны быть  $00_2$
    - Т.е. рассматриваются как и 4-байтные примитивные типы данных

# Выполнение правил выравнивания для полей

- Внутри структуры
  - Выравнивание должно выполняться для каждого поля
- Размещение всей структуры
  - Для каждой структуры определяется требование по выравниванию в **К** байт
    - **К** = Наибольшее выравнивание среди всех полей
  - Начальный адрес структуры и ее длина должны быть кратны **К**
- Пример (для Windows или x86-64):
  - **К** = 8, из-за присутствия поля типа double

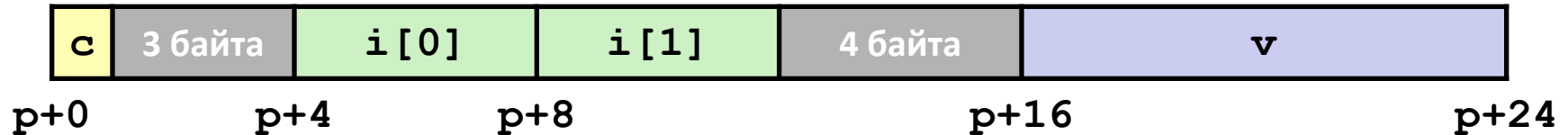
```
struct S1 {
    char c;
    int i[2];
    double v;
} *p;
```



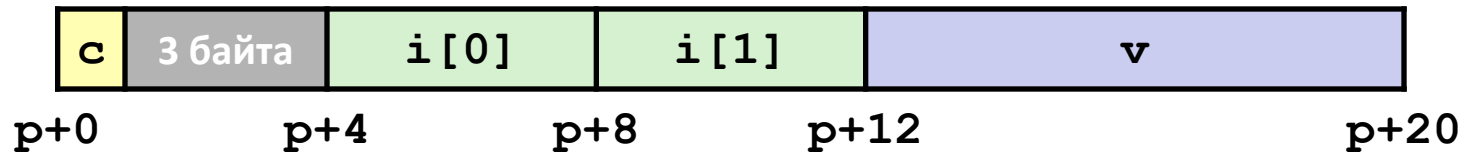
# Различные соглашения о выравнивании

- x86-64 или IA32 Windows:
  - $K = 8$ , из-за наличия поля типа **double**

```
struct S1 {
    char c;
    int i[2];
    double v;
} *p;
```



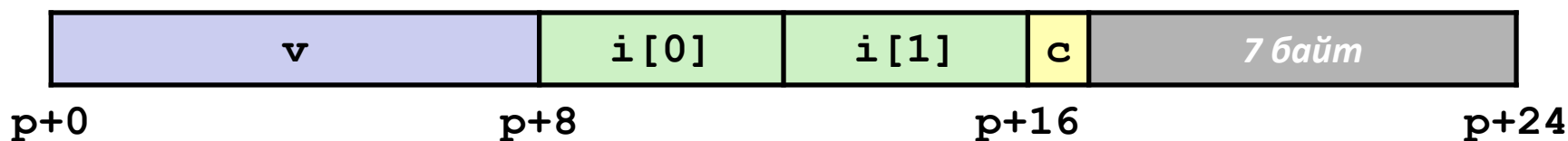
- IA32 Linux
  - $K = 4$ ; double рассматривается аналогично 4-байтным типам данных



# Выравнивание всей структуры

- Определяется требование к выравниванию в К байт
- Общий размер структуры должен быть кратен К

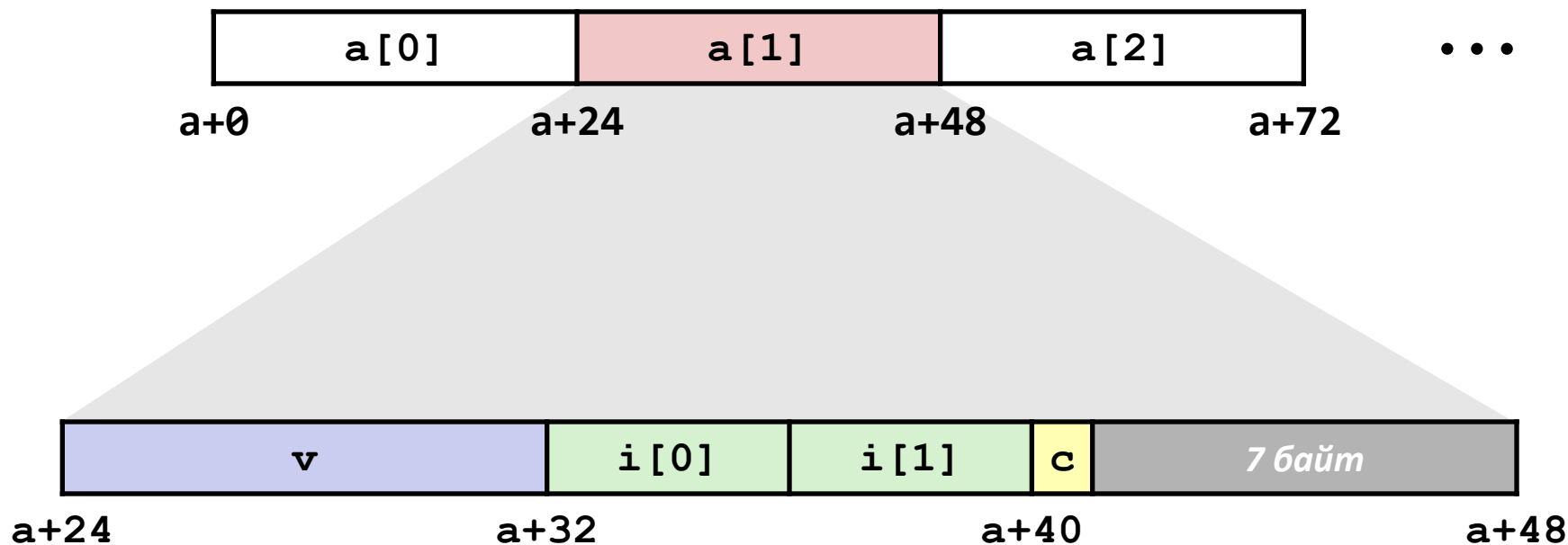
```
struct S2 {  
    double v;  
    int i[2];  
    char c;  
} *p;
```



# Массивы структур

- Размер всей структуры кратен  $K$
- Для каждого элемента массива производится выравнивание

```
struct S2 {  
    double v;  
    int i[2];  
    char c;  
} a[10];
```

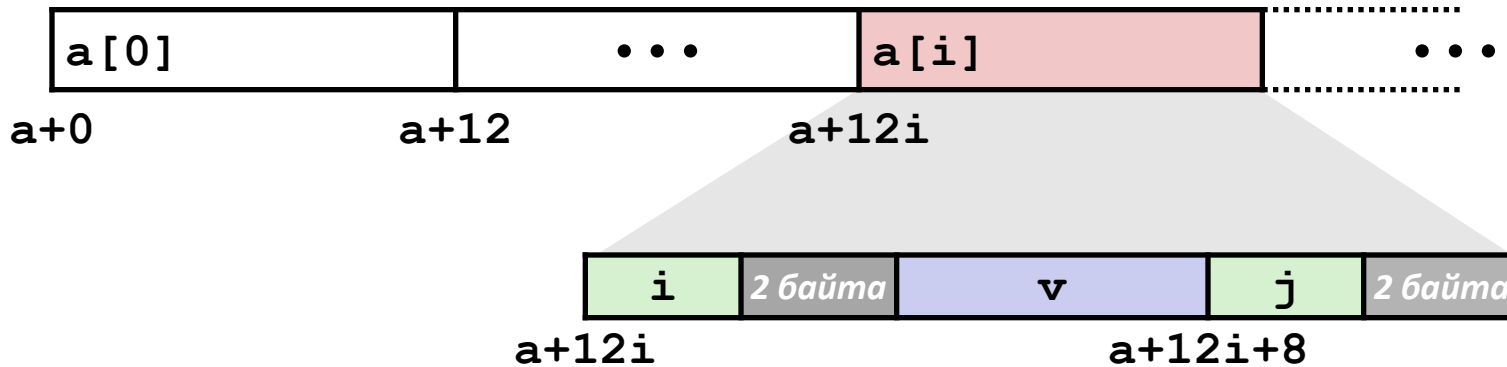




# Доступ к элементам массива

- Вычисляем смещение в массиве
  - Вычисляем `sizeof(S3)`, учитывая пропуски
- Вычисляем смещение внутри структуры
  - Поле `j` расположено со смещением 8 внутри структуры

```
struct S3 {
    short i;
    float v;
    short j;
} a[10];
```



```
short get_j(int idx)
{
    return a[idx].j;
}
```

```
; eax = idx
lea eax, [eax + 2 * eax] ; 3*idx
movsx eax, [a + 4 * eax + 8]
```

# Как сохранить место

- Размещаем большие типы первыми

```
struct S4 {
  char c;
  int i;
  char d;
} *p;
```



```
struct S5 {
  int i;
  char c;
  char d;
} *p;
```

- Результат (K=4)

