

Средства симуляции ЦП и ОС и изучение поведения программ

Лекция №4



Державин Андрей
Шурыгин Антон

➤ **Квиз**

- Кодировка инструкций
- Формат ELF
- Адресное пространство Linux
- Загрузка ELF
- Домашнее задание №3

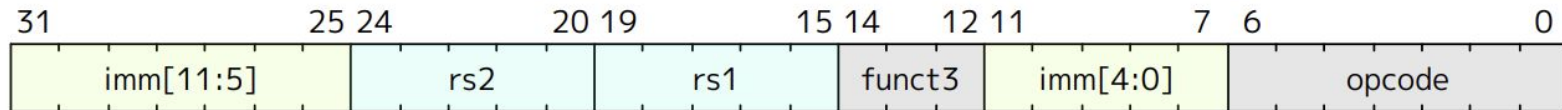
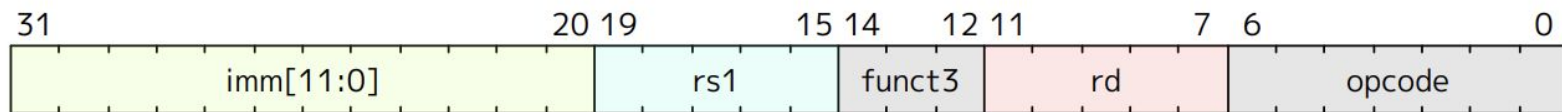
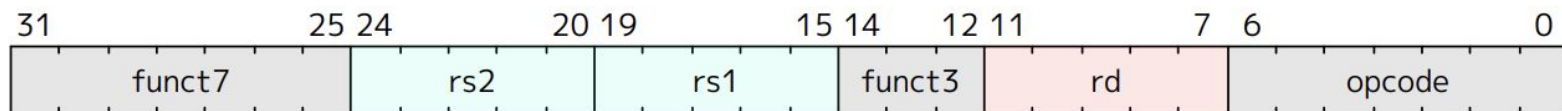
КВИЗ



- Квиз
- **Кодировка инструкций**
 - Формат ELF
 - Адресное пространство Linux
 - Загрузка ELF
 - Домашнее задание №3

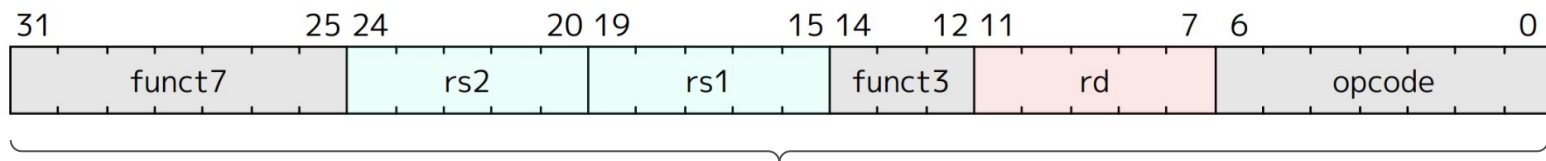
Кодировка инструкций

- Рассмотрим несколько примеров инструкций



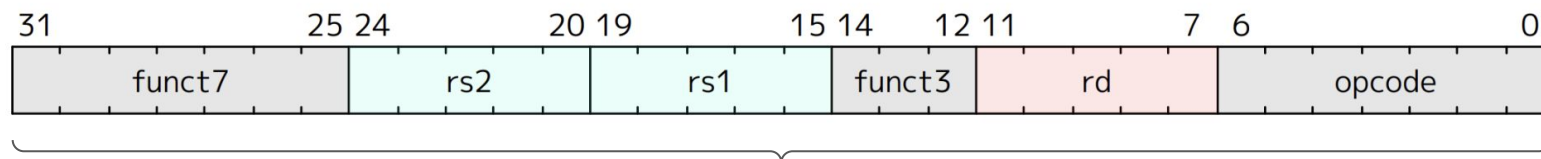
Кодировка инструкций

- Определяем тип кодировки



Кодировка инструкций

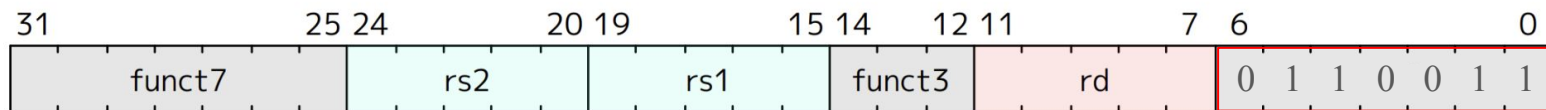
- Определяем тип кодировки



R-type

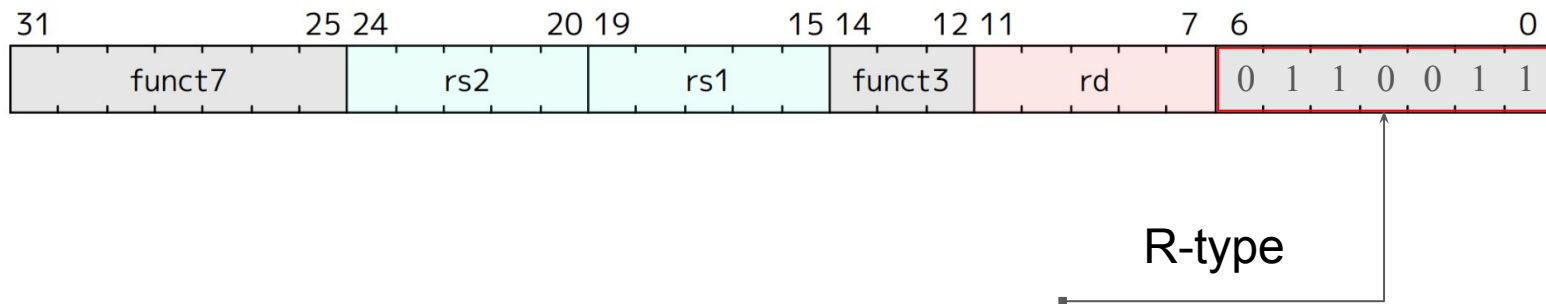
Кодировка инструкций

- Определяем тип кодировки



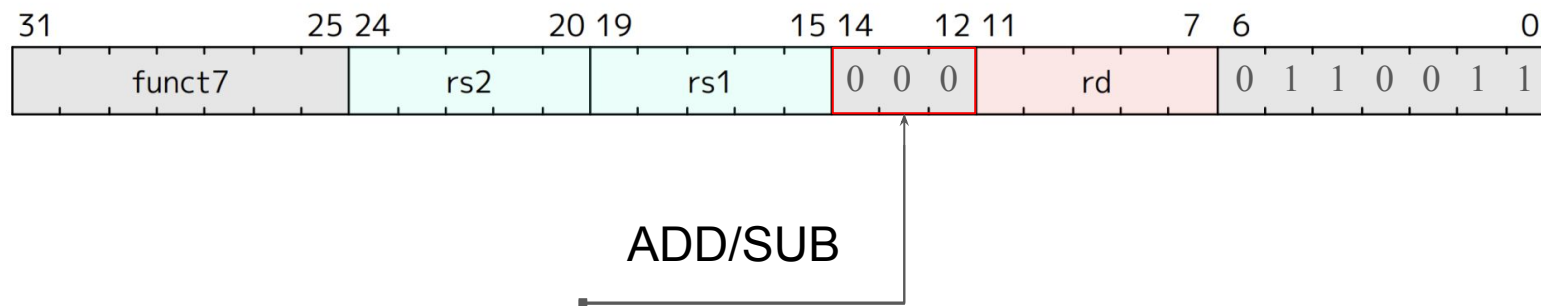
Кодировка инструкций

- Определяем тип кодировки
 - R-type: читаем из rs2, rs1, пишем в rd
 - funct3, funct7 однозначно определяют тип производимой операции



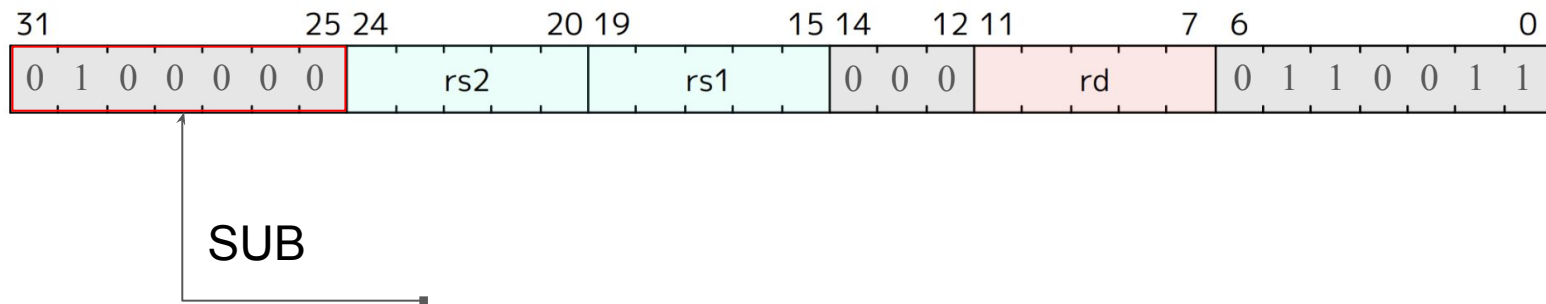
Кодировка инструкций

- Определяем тип кодировки
 - R-type: читаем из rs2, rs1, пишем в rd
 - func3, func7 однозначно определяют тип производимой операции



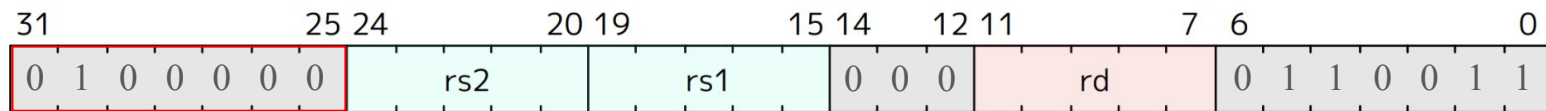
Кодировка инструкций

- Определяем тип кодировки
 - R-type: читаем из rs2, rs1, пишем в rd
 - func3, func7 однозначно определяют тип производимой операции



Кодировка инструкций

- Определяем тип кодировки
 - R-type: читаем из rs2, rs1, пишем в rd
 - func3, func7 однозначно определяют тип производимой операции

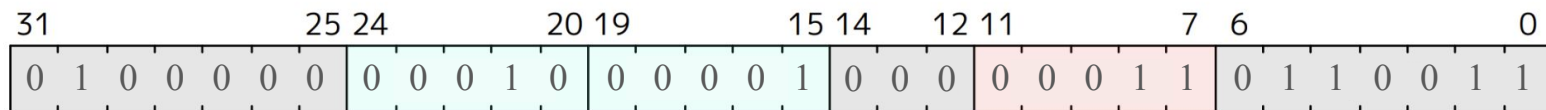


Кодировка инструкции SUB

- Назначим rs2, rs1, rd значения

Кодировка инструкций

- Определяем тип кодировки
 - R-type: читаем из rs2, rs1, пишем в rd
 - func3, func7 однозначно определяют тип производимой операции



- Назначим rs2, rs1, rd значения
- Получили инструкцию с байт-кодом **0x402081B3**
 - `sub x3, x1, x2`

Кодировка инструкций

```
.section ".text"
```

```
.global _start
```

```
start:
```

```
addi x1, x0, 1
```

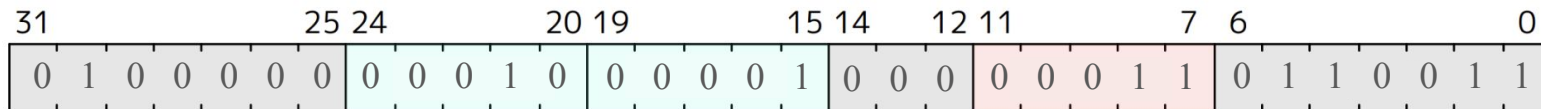
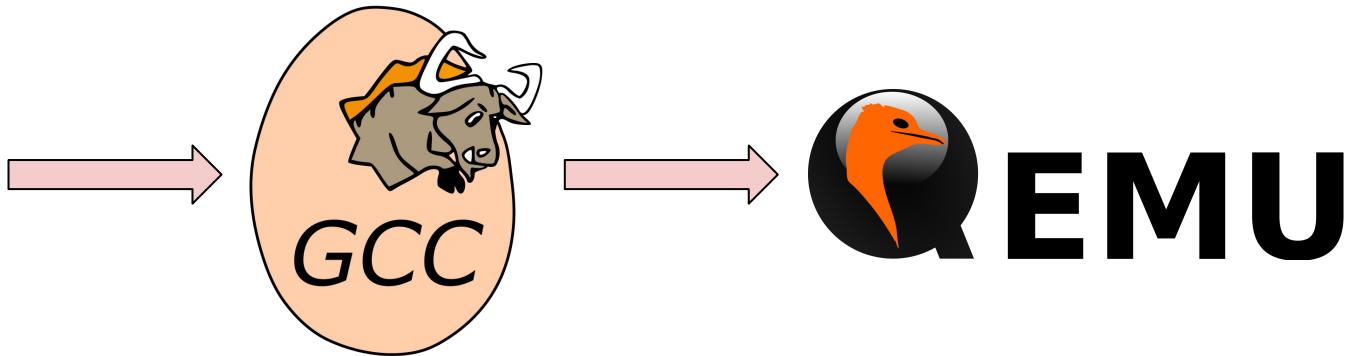
```
addi x2, x0, 2
```

```
sub x3, x1, x2
```

```
addi a0, x0, 0
```

```
addi a7, x0, 93
```

ecall



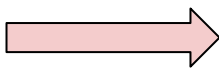
Кодировка инструкций

```
.section ".text"
.global _start

_start:
    addi x1, x0, 1
    addi x2, x0, 2

    sub x3, x1, x2

    addi a0, x0, 0
    addi a7, x0, 93
    ecall
```

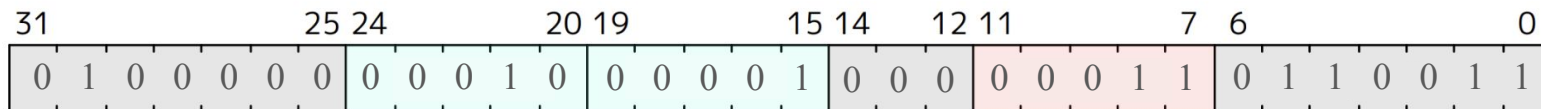


```
IN:
0x000100b8: 402001b3      sub      gp,ra,sp

pc: 000000000000100b8
x0/zero: 0000000000000000 x1/ra: 0000000000000001 x2/sp: 0000000000000002 x3/gp: 0000000000000000
x4/tp: 0000000000000000 x5/t0: 0000000000000000 x6/t1: 0000000000000000 x7/t2: 0000000000000000
x8/s0: 0000000000000000 x9/s1: 0000000000000000 x10/a0: 0000000000000000 x11/a1: 0000000000000000
x12/a2: 0000000000000000 x13/a3: 0000000000000000 x14/a4: 0000000000000000 x15/a5: 0000000000000000
x16/a6: 0000000000000000 x17/a7: 0000000000000000 x18/s2: 0000000000000000 x19/s3: 0000000000000000
x20/s4: 0000000000000000 x21/s5: 0000000000000000 x22/s6: 0000000000000000 x23/s7: 0000000000000000
x24/s8: 0000000000000000 x25/s9: 0000000000000000 x26/s10: 0000000000000000 x27/s11: 0000000000000000
x28/t3: 0000000000000000 x29/t4: 0000000000000000 x30/t5: 0000000000000000 x31/t6: 0000000000000000

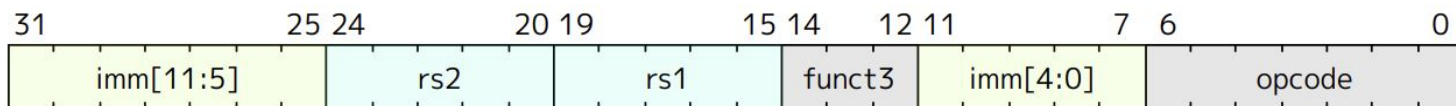
IN:
0x000100bc: 00000513      mv      a0,zero

pc: 000000000000100bc
x0/zero: 0000000000000000 x1/ra: 0000000000000001 x2/sp: 0000000000000002 x3/gp: ffffffff
x4/tp: 0000000000000000 x5/t0: 0000000000000000 x6/t1: 0000000000000000 x7/t2: 0000000000000000
x8/s0: 0000000000000000 x9/s1: 0000000000000000 x10/a0: 0000000000000000 x11/a1: 0000000000000000
x12/a2: 0000000000000000 x13/a3: 0000000000000000 x14/a4: 0000000000000000 x15/a5: 0000000000000000
x16/a6: 0000000000000000 x17/a7: 0000000000000000 x18/s2: 0000000000000000 x19/s3: 0000000000000000
x20/s4: 0000000000000000 x21/s5: 0000000000000000 x22/s6: 0000000000000000 x23/s7: 0000000000000000
x24/s8: 0000000000000000 x25/s9: 0000000000000000 x26/s10: 0000000000000000 x27/s11: 0000000000000000
x28/t3: 0000000000000000 x29/t4: 0000000000000000 x30/t5: 0000000000000000 x31/t6: 0000000000000000
```



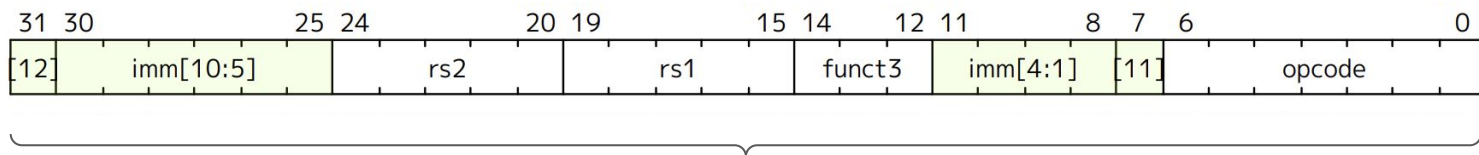
Кодировка инструкций

- Определяем тип кодировки



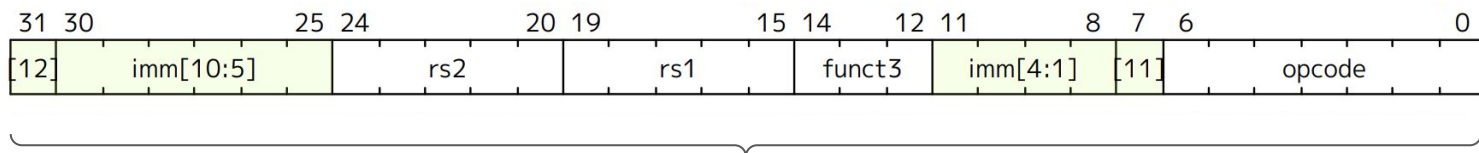
Кодировка инструкций

- Определяем тип кодировки



Кодировка инструкций

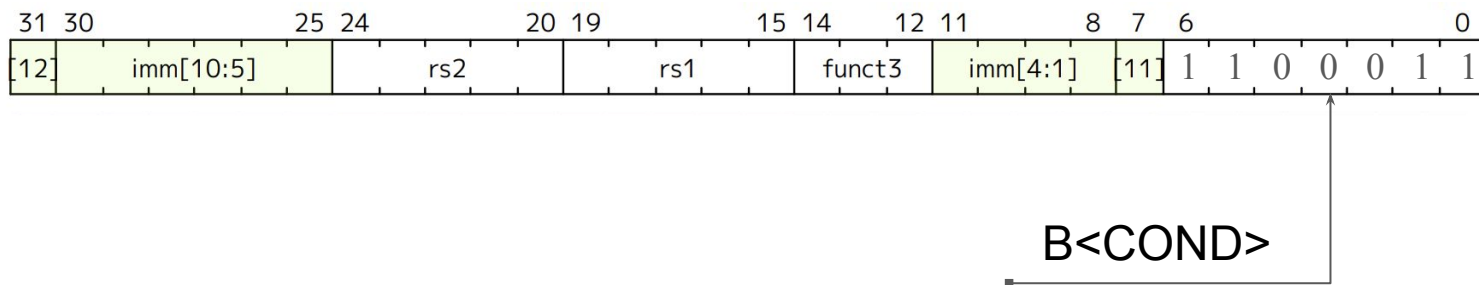
- Определяем тип кодировки



B-type

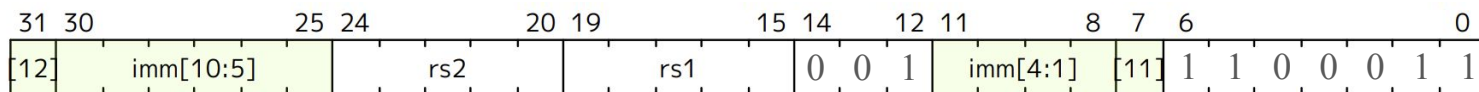
Кодировка инструкций

- Определяем тип кодировки
 - B-type: условный переход как результат сравнения rs1, rs2



Кодировка инструкций

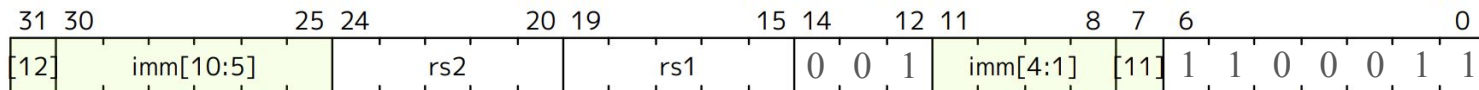
- Определяем тип кодировки
 - B-type: условный переход как результат сравнения rs1, rs2
 - funct3 определяет условие сравнения



NOT EQUAL

Кодировка инструкций

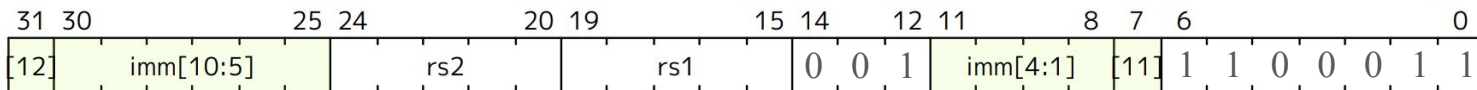
- Определяем тип кодировки
 - B-type: условный переход как результат сравнения rs1, rs2
 - funct3 определяет условие сравнения



Кодировка инструкции **BNE**

Кодировка инструкций

- Определяем тип кодировки
 - B-type: условный переход как результат сравнения rs1, rs2
 - funct3 определяет условие сравнения



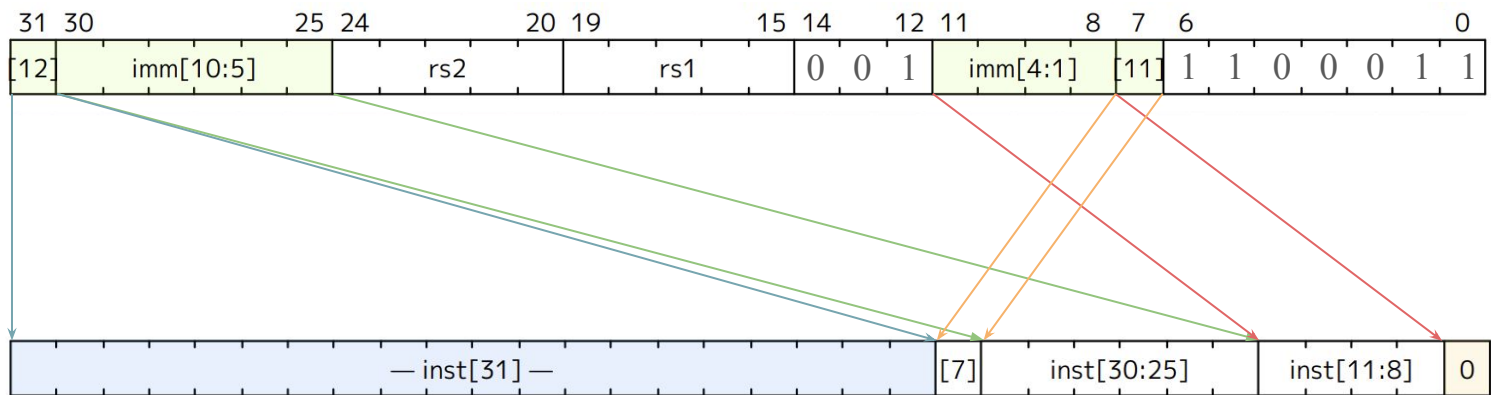
Кодировка инструкции **BNE**



Кодировка immediate через кодировку инструкции **BNE**

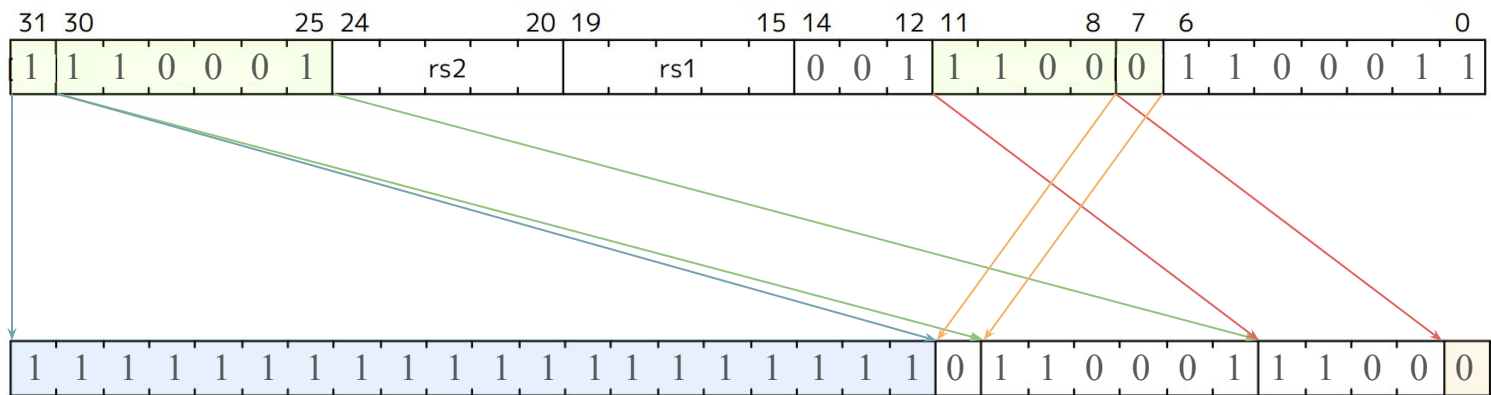
Кодировка инструкций

- Определяем тип кодировки
 - B-type: условный переход как результат сравнения rs1, rs2
 - funct3 определяет условие сравнения



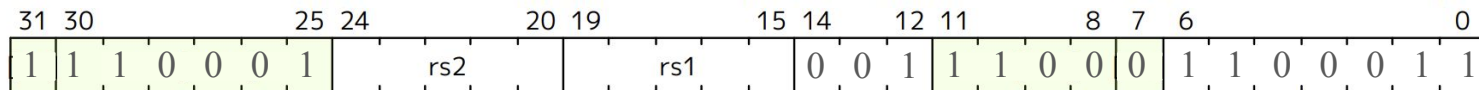
Кодировка инструкций

- Определяем тип кодировки
 - B-type: условный переход как результат сравнения rs1, rs2
 - funct3 определяет условие сравнения



Кодировка инструкций

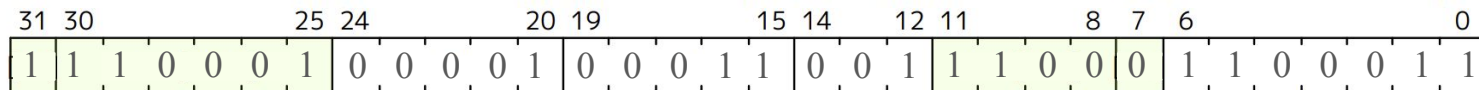
- Определяем тип кодировки
 - B-type: условный переход как результат сравнения rs1, rs2
 - funct3 определяет условие сравнения



- Назначим rs2, rs1 значения

Кодировка инструкций

- Определяем тип кодировки
 - B-type: условный переход как результат сравнения rs1, rs2
 - funct3 определяет условие сравнения



- Назначим rs2, rs1 значения
- Получили инструкцию с байт-кодом **0xE2119C63**
 - `bne x3, x1, -2504`

- Квиз
- Кодировка инструкций

➤ **Формат ELF**

- Адресное пространство Linux
- Загрузка ELF
- Домашнее задание №3

Формат ELF

- ELF (**E**xecutable and **L**inkable **F**ormat) - формат бинарных файлов
- В ELF можно хранить:
 - Исполняемые файлы (executable file)
 - Объектные файлы (relocatable file)
 - Разделяемые библиотеки (shared object)
- Два представления ELF-файла:
 - Секции - для разработки (ассемблер, компоновщик, отладчик)
 - .text, .data, .rodata...
 - Сегменты - для ОС
 - Позволяет оптимизировать время загрузки в ОС - загружаем только сегменты помеченные флагом PT_LOAD

Формат ELF

- ELF файл состоит из:
 - Заголовок
 - Содержимое
 - Таблица заголовков сегментов (“Program header”)
 - Таблица заголовков секций (“Section header”)
 - Данные

Формат ELF. Заголовок

```
typedef struct elf64_hdr {  
    unsigned char e_ident[EI_NIDENT];  
    Elf64_Half e_type;  
    Elf64_Half e_machine;  
    Elf64_Word e_version;  
    Elf64_Addr e_entry;  
    Elf64_Off e_phoff;  
    Elf64_Off e_shoff;  
    Elf64_Word e_flags;  
    Elf64_Half e_ehsize;  
    Elf64_Half e_phentsize;  
    Elf64_Half e_phnum;  
    Elf64_Half e_shentsize;  
    Elf64_Half e_shnum;  
    Elf64_Half e_shstrndx;  
} Elf64_Ehdr;
```

“ELF”

0x7F 0x45 0x4C 0x46

Разрядность: 32(1) или 64(2)

Порядок байт: little-(1) или big-(2) endian

Версия формата

Используемое ABI (0x0 - SystemV, 0x3 - Linux,...)

Версия ABI

Паддинг (нули)

Формат ELF. Заголовок

```
typedef struct elf64_hdr {  
    unsigned char e_ident[EI_NIDENT];  
    Elf64_Half e_type;  
    Elf64_Half e_machine;  
    Elf64_Word e_version;  
    Elf64_Addr e_entry;  
    Elf64_Off e_phoff;  
    Elf64_Off e_shoff;  
    Elf64_Word e_flags;  
    Elf64_Half e_ehsize;  
    Elf64_Half e_phentsize;  
    Elf64_Half e_phnum;  
    Elf64_Half e_shentsize;  
    Elf64_Half e_shnum;  
    Elf64_Half e_shstrndx;  
} Elf64_Ehdr;
```

Тип файла

Значение	Описание
0x01	Relocatable
0x02	Executable
0x03	Shared object
0x04	Core file

Формат ELF. Заголовок

```
typedef struct elf64_hdr {  
    unsigned char e_ident[EI_NIDENT];  
    Elf64_Half e_type;  
    Elf64_Half e_machine;  
    Elf64_Word e_version;  
    Elf64_Addr e_entry;  
    Elf64_Off e_phoff;  
    Elf64_Off e_shoff;  
    Elf64_Word e_flags;  
    Elf64_Half e_ehsize;  
    Elf64_Half e_phentsize;  
    Elf64_Half e_phnum;  
    Elf64_Half e_shentsize;  
    Elf64_Half e_shnum;  
    Elf64_Half e_shstrndx;  
} Elf64_Ehdr;
```

ISA

Значение	Описание
0x08	MIPS
...	...
0x3E	x86_64
...	...
0xF3	RISC-V

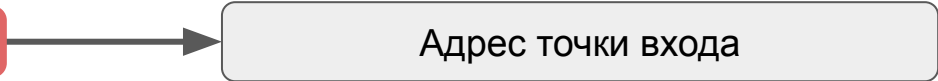
Формат ELF. Заголовок

```
typedef struct elf64_hdr {  
    unsigned char e_ident[EI_NIDENT];  
    Elf64_Half e_type;  
    Elf64_Half e_machine;  
    Elf64_Word e_version;  
    Elf64_Addr e_entry;  
    Elf64_Off e_phoff;  
    Elf64_Off e_shoff;  
    Elf64_Word e_flags;  
    Elf64_Half e_ehsize;  
    Elf64_Half e_phentsize;  
    Elf64_Half e_phnum;  
    Elf64_Half e_shentsize;  
    Elf64_Half e_shnum;  
    Elf64_Half e_shstrndx;  
} Elf64_Ehdr;
```

Версия ELF (опять?)

Формат ELF. Заголовок

```
typedef struct elf64_hdr {  
    unsigned char e_ident[EI_NIDENT];  
    Elf64_Half e_type;  
    Elf64_Half e_machine;  
    Elf64_Word e_version;  
    Elf64_Addr e_entry;  
    Elf64_Off e_phoff;  
    Elf64_Off e_shoff;  
    Elf64_Word e_flags;  
    Elf64_Half e_ehsize;  
    Elf64_Half e_phentsize;  
    Elf64_Half e_phnum;  
    Elf64_Half e_shentsize;  
    Elf64_Half e_shnum;  
    Elf64_Half e_shstrndx;  
} Elf64_Ehdr;
```

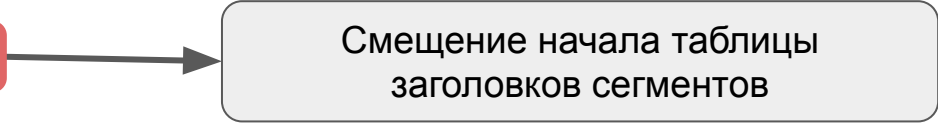


Адрес точки входа

The diagram consists of a red rectangular box highlighting the `Elf64_Addr e_entry;` line in the code block on the left. A black arrow points from this box to a light gray rounded rectangular box on the right containing the text "Адрес точки входа".

Формат ELF. Заголовок

```
typedef struct elf64_hdr {  
    unsigned char e_ident[EI_NIDENT];  
    Elf64_Half e_type;  
    Elf64_Half e_machine;  
    Elf64_Word e_version;  
    Elf64_Addr e_entry;  
    Elf64_Off e_phoff;  
    Elf64_Off e_shoff;  
    Elf64_Word e_flags;  
    Elf64_Half e_ehsize;  
    Elf64_Half e_phentsize;  
    Elf64_Half e_phnum;  
    Elf64_Half e_shentsize;  
    Elf64_Half e_shnum;  
    Elf64_Half e_shstrndx;  
} Elf64_Ehdr;
```




Смещение начала таблицы
заголовков сегментов

The diagram consists of a red rectangular box highlighting the `e_phoff` field in the ELF header structure. A black arrow points from this box to a grey rounded rectangular box on the right, which contains the Russian text explaining the field's meaning: 'Смещение начала таблицы заголовков сегментов' (Offset of the start of the segment header table).

Формат ELF. Заголовок

```
typedef struct elf64_hdr {  
    unsigned char e_ident[EI_NIDENT];  
    Elf64_Half e_type;  
    Elf64_Half e_machine;  
    Elf64_Word e_version;  
    Elf64_Addr e_entry;  
    Elf64_Off e_phoff;  
    Elf64_Off e_shoff;  
    Elf64_Word e_flags;  
    Elf64_Half e_ehsize;  
    Elf64_Half e_phentsize;  
    Elf64_Half e_phnum;  
    Elf64_Half e_shentsize;  
    Elf64_Half e_shnum;  
    Elf64_Half e_shstrndx;  
} Elf64_Ehdr;
```



Смещение начала таблицы
заголовков секций


Формат ELF. Заголовок

```
typedef struct elf64_hdr {  
    unsigned char e_ident[EI_NIDENT];  
    Elf64_Half e_type;  
    Elf64_Half e_machine;  
    Elf64_Word e_version;  
    Elf64_Addr e_entry;  
    Elf64_Off e_phoff;  
    Elf64_Off e_shoff;  
    Elf64_Word e_flags;  
    Elf64_Half e_ehsize;  
    Elf64_Half e_phentsize;  
    Elf64_Half e_phnum;  
    Elf64_Half e_shentsize;  
    Elf64_Half e_shnum;  
    Elf64_Half e_shstrndx;  
} Elf64_Ehdr;
```

Флаги (зависят от архитектуры)

Формат ELF. Заголовок

```
typedef struct elf64_hdr {  
    unsigned char e_ident[EI_NIDENT];  
    Elf64_Half e_type;  
    Elf64_Half e_machine;  
    Elf64_Word e_version;  
    Elf64_Addr e_entry;  
    Elf64_Off e_phoff;  
    Elf64_Off e_shoff;  
    Elf64_Word e_flags;  
    Elf64_Half e_ehsize;  
    Elf64_Half e_phentsize;  
    Elf64_Half e_phnum;  
    Elf64_Half e_shentsize;  
    Elf64_Half e_shnum;  
    Elf64_Half e_shstrndx;  
} Elf64_Ehdr;
```



Размер заголовка:
64(64) либо 52(32) байт

Формат ELF. Заголовок

```
typedef struct elf64_hdr {  
    unsigned char e_ident[EI_NIDENT];  
    Elf64_Half e_type;  
    Elf64_Half e_machine;  
    Elf64_Word e_version;  
    Elf64_Addr e_entry;  
    Elf64_Off e_phoff;  
    Elf64_Off e_shoff;  
    Elf64_Word e_flags;  
    Elf64_Half e_ehsize;  
    Elf64_Half e_phentsize;  
    Elf64_Half e_phnum;  
    Elf64_Half e_shentsize;  
    Elf64_Half e_shnum;  
    Elf64_Half e_shstrndx;  
} Elf64_Ehdr;
```

Размер заголовка сегмента:
64(64) либо 40(32) байт

Количество заголовков сегментов

Формат ELF. Заголовок


```
typedef struct elf64_hdr {  
    unsigned char e_ident[EI_NIDENT];  
    Elf64_Half e_type;  
    Elf64_Half e_machine;  
    Elf64_Word e_version;  
    Elf64_Addr e_entry;  
    Elf64_Off e_phoff;  
    Elf64_Off e_shoff;  
    Elf64_Word e_flags;  
    Elf64_Half e_ehsize;  
    Elf64_Half e_phentsize;  
    Elf64_Half e_phnum;  
    Elf64_Half e_shentsize;  
    Elf64_Half e_shnum;  
    Elf64_Half e_shstrndx;  
} Elf64_Ehdr;
```

Размер заголовка секции:
40(64) либо 32(32) байт

Количество заголовков секций

Формат ELF. Заголовок

```
typedef struct elf64_hdr {  
    unsigned char e_ident[EI_NIDENT];  
    Elf64_Half e_type;  
    Elf64_Half e_machine;  
    Elf64_Word e_version;  
    Elf64_Addr e_entry;  
    Elf64_Off e_phoff;  
    Elf64_Off e_shoff;  
    Elf64_Word e_flags;  
    Elf64_Half e_ehsize;  
    Elf64_Half e_phentsize;  
    Elf64_Half e_phnum;  
    Elf64_Half e_shentsize;  
    Elf64_Half e_shnum;  
    Elf64_Half e_shstrndx;  
} Elf64_Ehdr;
```



Номер секции в таблице секций, в которой содержатся имена секций

Формат ELF. Заголовок сегмента

```
typedef struct elf64_phdr {  
    Elf64_Word p_type;  
    Elf64_Word p_flags;  
    Elf64_Off p_offset;  
    Elf64_Addr p_vaddr;  
    Elf64_Addr p_paddr;  
    Elf64_Xword p_filesz;  
    Elf64_Xword p_memsz;  
    Elf64_Xword p_align;  
} Elf64_Phdr;
```

Тип сегмента

Значение	Описание
...	
0x1	Загружаемый
0x2	Информация для динамической линковки
0x3	Информация для интерпретатора
...	

Формат ELF. Заголовок сегмента

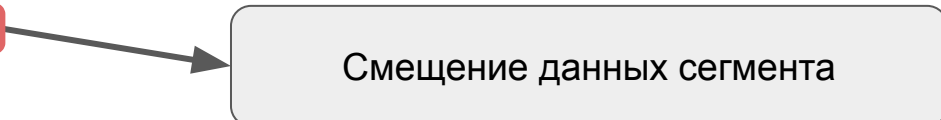
```
typedef struct elf64_phdr {  
    Elf64_Word p_type;  
    Elf64_Word p_flags;  
    Elf64_Off p_offset;  
    Elf64_Addr p_vaddr;  
    Elf64_Addr p_paddr;  
    Elf64_Xword p_filesz;  
    Elf64_Xword p_memsz;  
    Elf64_Xword p_align;  
} Elf64_Phdr;
```

Флаги доступа

Значение	Описание
0x1	Исполнение
0x2	Запись
0x4	Чтение

Формат ELF. Заголовок сегмента

```
typedef struct elf64_phdr {  
    Elf64_Word p_type;  
    Elf64_Word p_flags;  
    Elf64_Off p_offset;  
    Elf64_Addr p_vaddr;  
    Elf64_Addr p_paddr;  
    Elf64_Xword p_filesz;  
    Elf64_Xword p_memsz;  
    Elf64_Xword p_align;  
} Elf64_Phdr;
```

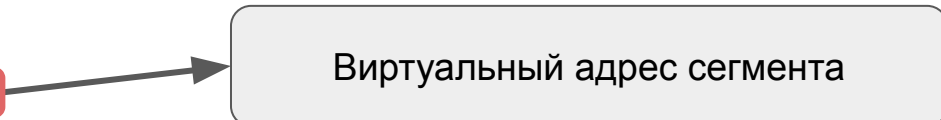


Смещение данных сегмента

The diagram consists of a red rectangular box highlighting the 'Elf64_Off p_offset;' line in the code block on the left. A black arrow points from this box to a light gray rounded rectangular box on the right containing the Russian text 'Смещение данных сегмента'.

Формат ELF. Заголовок сегмента

```
typedef struct elf64_phdr {  
    Elf64_Word p_type;  
    Elf64_Word p_flags;  
    Elf64_Off p_offset;  
    Elf64_Addr p_vaddr;  
    Elf64_Addr p_paddr;  
    Elf64_Xword p_filesz;  
    Elf64_Xword p_memsz;  
    Elf64_Xword p_align;  
} Elf64_Phdr;
```



Виртуальный адрес сегмента

The diagram consists of a red rectangular box highlighting the `Elf64_Addr p_vaddr;` line in the code block on the left. A black arrow points from this box to a light gray rounded rectangular box on the right containing the text "Виртуальный адрес сегмента".

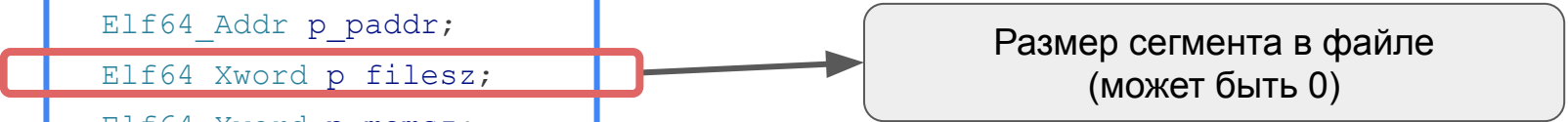
Формат ELF. Заголовок сегмента

```
typedef struct elf64_phdr {  
    Elf64_Word p_type;  
    Elf64_Word p_flags;  
    Elf64_Off p_offset;  
    Elf64_Addr p_vaddr;  
    Elf64_Addr p_paddr;  
    Elf64_Xword p_filesz;  
    Elf64_Xword p_memsz;  
    Elf64_Xword p_align;  
} Elf64_Phdr;
```

Физический адрес сегмента
(обычно не используется)

Формат ELF. Заголовок сегмента

```
typedef struct elf64_phdr {  
    Elf64_Word p_type;  
    Elf64_Word p_flags;  
    Elf64_Off p_offset;  
    Elf64_Addr p_vaddr;  
    Elf64_Addr p_paddr;  
    Elf64_Xword p_filesz;  
    Elf64_Xword p_memsz;  
    Elf64_Xword p_align;  
} Elf64_Phdr;
```



Размер сегмента в файле
(может быть 0)

Формат ELF. Заголовок сегмента

```
typedef struct elf64_phdr {  
    Elf64_Word p_type;  
    Elf64_Word p_flags;  
    Elf64_Off p_offset;  
    Elf64_Addr p_vaddr;  
    Elf64_Addr p_paddr;  
    Elf64_Xword p_filesz;  
    Elf64_Xword p_memsz;  
    Elf64_Xword p_align;  
} Elf64_Phdr;
```

Размер сегмента в памяти
(может быть 0)

Формат ELF. Заголовок сегмента

```
typedef struct elf64_phdr {  
    Elf64_Word p_type;  
    Elf64_Word p_flags;  
    Elf64_Off p_offset;  
    Elf64_Addr p_vaddr;  
    Elf64_Addr p_paddr;  
    Elf64_Xword p_filesz;  
    Elf64_Xword p_memsz;  
    Elf64_Xword p_align;  
} Elf64_Phdr;
```

Выравнивание

Формат ELF. Заголовок секции

```
typedef struct elf64_shdr {  
    Elf64_Word sh_name;  
    Elf64_Word sh_type;  
    Elf64_Xword sh_flags;  
    Elf64_Addr sh_addr;  
    Elf64_Off sh_offset;  
    Elf64_Xword sh_size;  
    Elf64_Word sh_link;  
    Elf64_Word sh_info;  
    Elf64_Xword sh_addralign;  
    Elf64_Xword sh_entsize;  
} Elf64_Shdr;
```

Смещение имени секции в .shstrtab

Формат ELF. Заголовок секции

```
typedef struct elf64_shdr {  
    Elf64_Word sh_name;  
    Elf64_Word sh_type;  
    Elf64_Xword sh_flags;  
    Elf64_Addr sh_addr;  
    Elf64_Off sh_offset;  
    Elf64_Xword sh_size;  
    Elf64_Word sh_link;  
    Elf64_Word sh_info;  
    Elf64_Xword sh_addralign;  
    Elf64_Xword sh_entsize;  
} Elf64_Shdr;
```

Тип секции (данные, строки, таблица символов...)

Формат ELF. Заголовок секции

```
typedef struct elf64_shdr {  
    Elf64_Word sh_name;  
    Elf64_Word sh_type;  
    Elf64_Xword sh_flags;  
    Elf64_Addr sh_addr;  
    Elf64_Off sh_offset;  
    Elf64_Xword sh_size;  
    Elf64_Word sh_link;  
    Elf64_Word sh_info;  
    Elf64_Xword sh_addralign;  
    Elf64_Xword sh_entsize;  
} Elf64_Shdr;
```

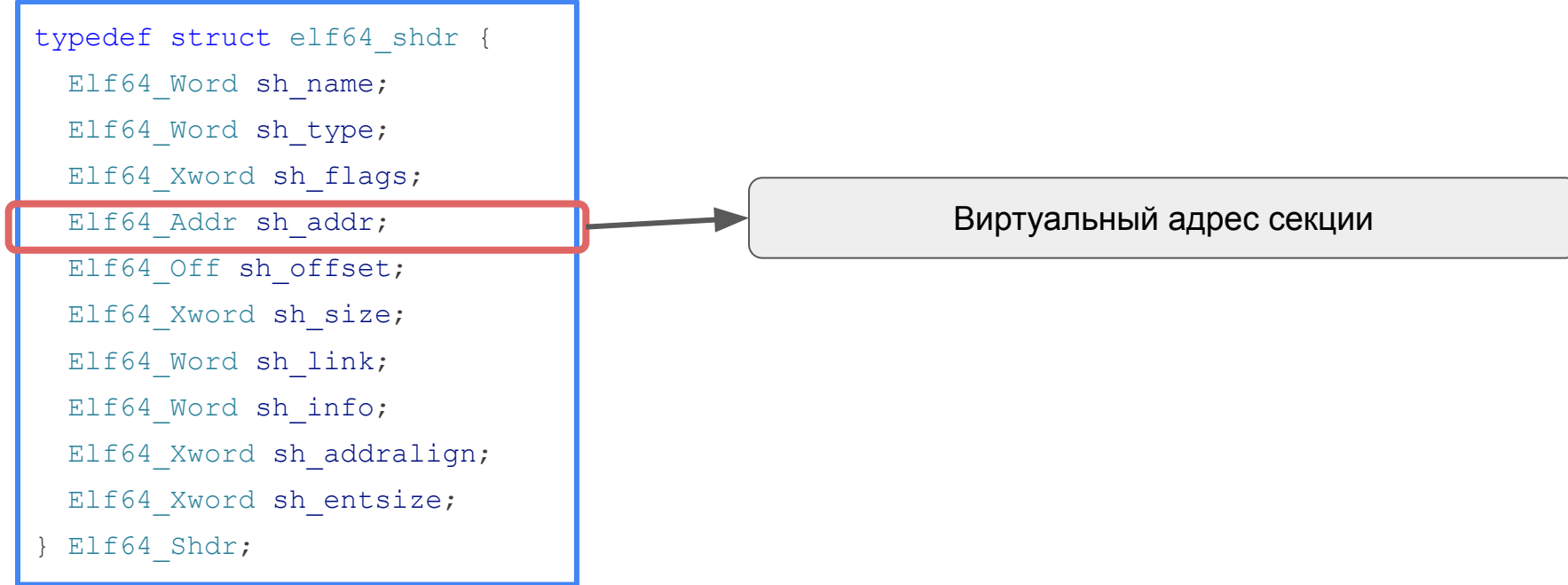


Флаги

The diagram consists of a light gray rounded rectangle with the word 'Флаги' (Flags) inside. A black arrow points from the 'sh_flags' field in the code block to this rectangle.

Формат ELF. Заголовок секции

```
typedef struct elf64_shdr {  
    Elf64_Word sh_name;  
    Elf64_Word sh_type;  
    Elf64_Xword sh_flags;  
    Elf64_Addr sh_addr;  
    Elf64_Off sh_offset;  
    Elf64_Xword sh_size;  
    Elf64_Word sh_link;  
    Elf64_Word sh_info;  
    Elf64_Xword sh_addralign;  
    Elf64_Xword sh_entsize;  
} Elf64_Shdr;
```



Виртуальный адрес секции

The diagram illustrates the relationship between the `sh_addr` field in the ELF section header and its virtual address. A blue box encloses the entire `elf64_shdr` struct definition. Within this box, the line `Elf64_Addr sh_addr;` is highlighted with a red rectangle. A black arrow points from this red rectangle to a light gray rounded rectangular box on the right, which contains the text "Виртуальный адрес секции" (Virtual address of the section).

Формат ELF. Заголовок секции

```
typedef struct elf64_shdr {  
    Elf64_Word sh_name;  
    Elf64_Word sh_type;  
    Elf64_Xword sh_flags;  
    Elf64_Addr sh_addr;  
    Elf64_Off sh_offset;  
    Elf64_Xword sh_size;  
    Elf64_Word sh_link;  
    Elf64_Word sh_info;  
    Elf64_Xword sh_addralign;  
    Elf64_Xword sh_entsize;  
} Elf64_Shdr;
```

Смещение данных секции

Формат ELF. Заголовок секции

```
typedef struct elf64_shdr {  
    Elf64_Word sh_name;  
    Elf64_Word sh_type;  
    Elf64_Xword sh_flags;  
    Elf64_Addr sh_addr;  
    Elf64_Off sh_offset;  
    Elf64_Xword sh_size;  
    Elf64_Word sh_link;  
    Elf64_Word sh_info;  
    Elf64_Xword sh_addralign;  
    Elf64_Xword sh_entsize;  
} Elf64_Shdr;
```

Размер секции (может быть 0)

Формат ELF. Заголовок секции

```
typedef struct elf64_shdr {  
    Elf64_Word sh_name;  
    Elf64_Word sh_type;  
    Elf64_Xword sh_flags;  
    Elf64_Addr sh_addr;  
    Elf64_Off sh_offset;  
    Elf64_Xword sh_size;  
    Elf64_Word sh_link;  
    Elf64_Word sh_info;  
    Elf64_Xword sh_addralign;  
    Elf64_Xword sh_entsize;  
} Elf64_Shdr;
```

Оptionальные поля
(индекс связанной секции и дополнительная информация)

Формат ELF. Заголовок секции

```
typedef struct elf64_shdr {  
    Elf64_Word sh_name;  
    Elf64_Word sh_type;  
    Elf64_Xword sh_flags;  
    Elf64_Addr sh_addr;  
    Elf64_Off sh_offset;  
    Elf64_Xword sh_size;  
    Elf64_Word sh_link;  
    Elf64_Word sh_info;  
    Elf64_Xword sh_addralign;  
    Elf64_Xword sh_entsize;  
} Elf64_Shdr;
```

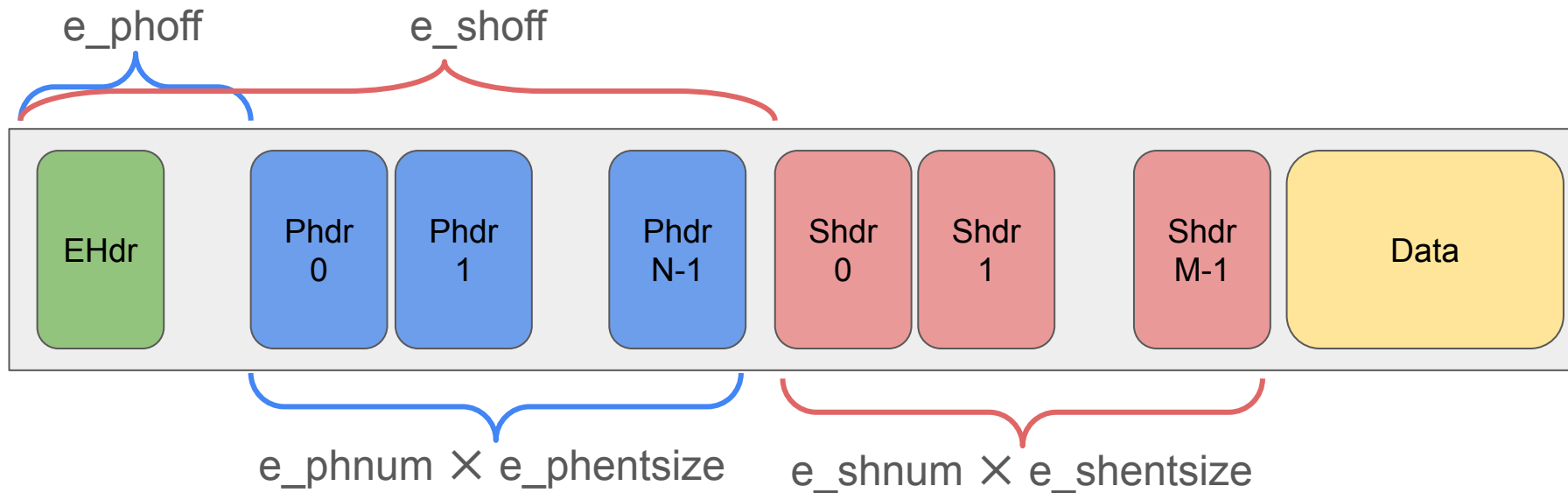
Выравнивание

Формат ELF. Заголовок секции

```
typedef struct elf64_shdr {  
    Elf64_Word sh_name;  
    Elf64_Word sh_type;  
    Elf64_Xword sh_flags;  
    Elf64_Addr sh_addr;  
    Elf64_Off sh_offset;  
    Elf64_Xword sh_size;  
    Elf64_Word sh_link;  
    Elf64_Word sh_info;  
    Elf64_Xword sh_addralign;  
    Elf64_Xword sh_entsize;  
} Elf64_Shdr;
```

Размер элемента таблицы (опционально).
Напр. нужен для таблицы символов

Формат ELF. Общая структура

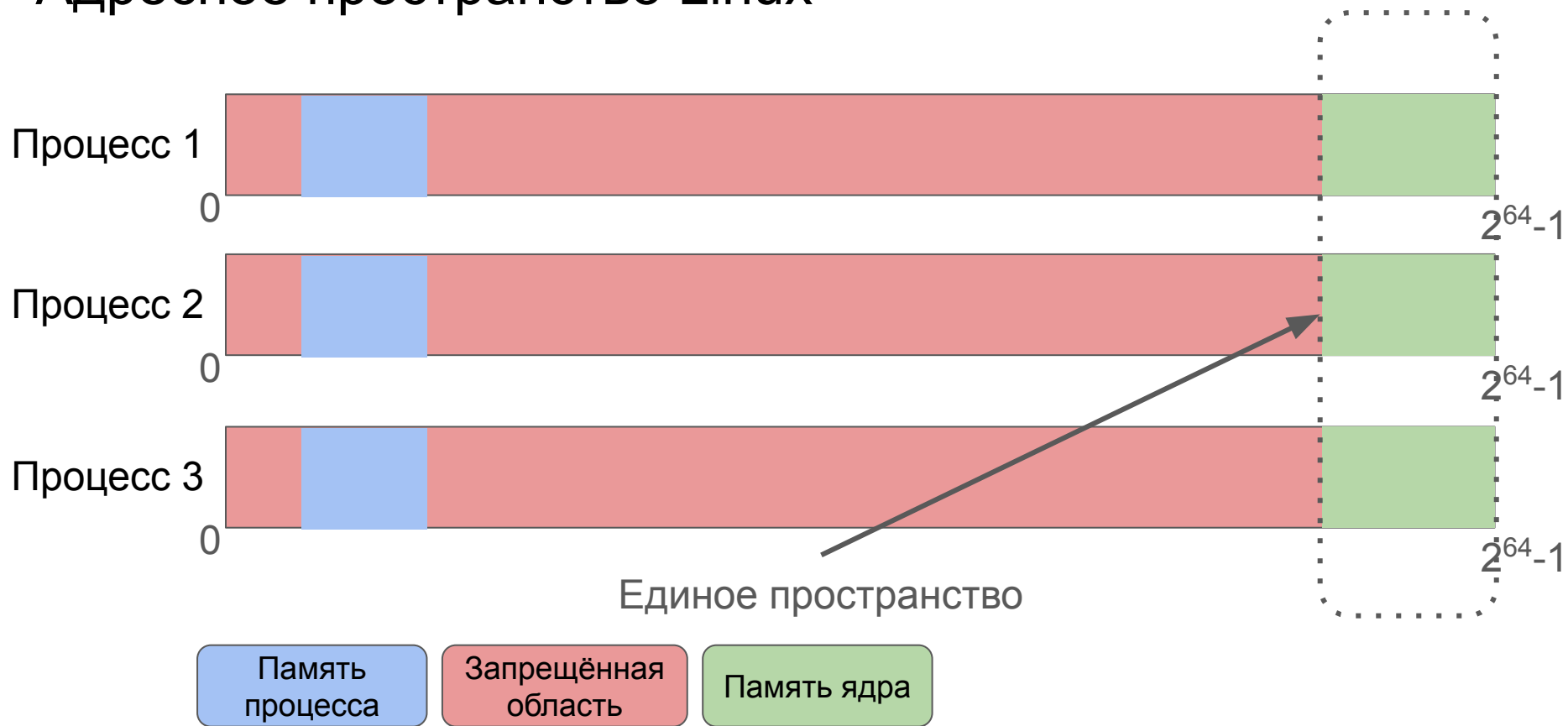


- Квиз
- Кодировка инструкций
- Формат ELF
- **Адресное пространство Linux**
 - Загрузка ELF
 - Домашнее задание №3

Адресное пространство Linux

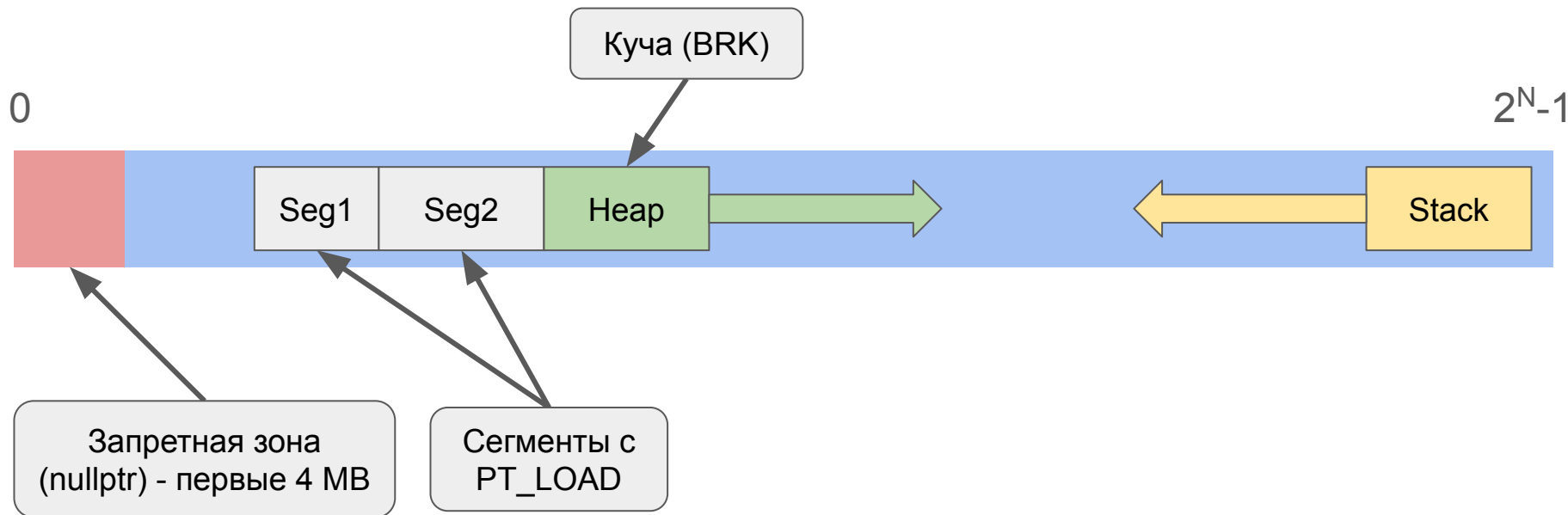
- Как устроена память процесса в 64-битном Linux?

Адресное пространство Linux



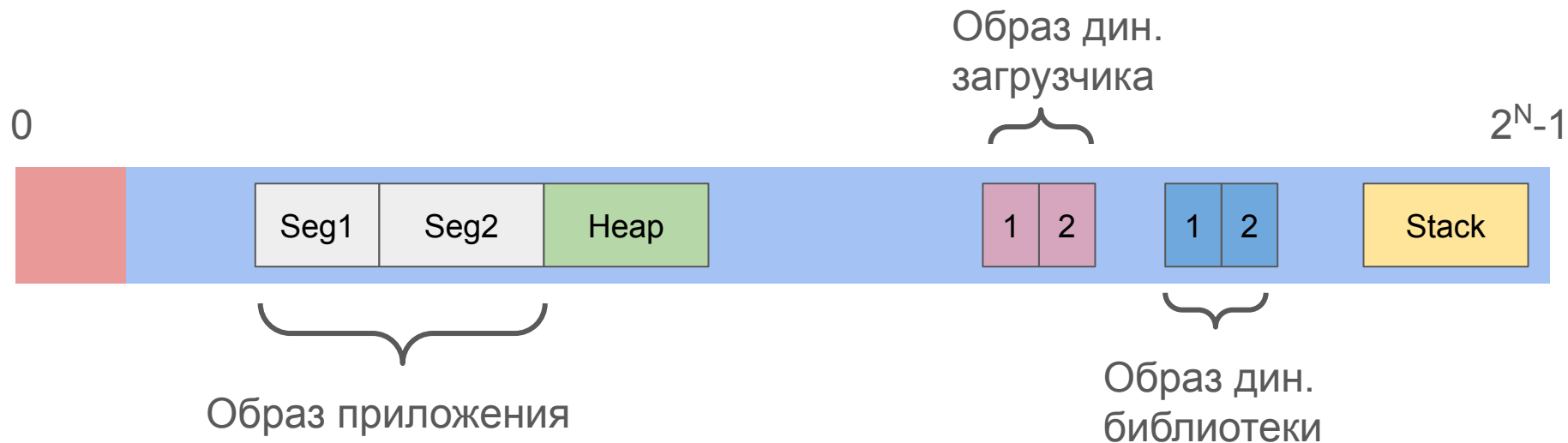
Адресное пространство Linux

- Рассмотрим память процесса



Адресное пространство Linux

➤ Динамическая загрузка

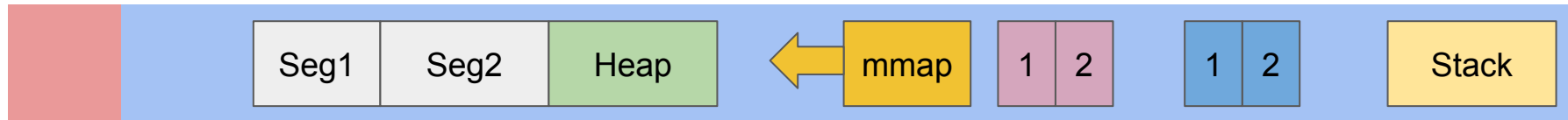


Адресное пространство Linux

- Системный вызов `mmap()`
 - Выделение/освобождение памяти

0

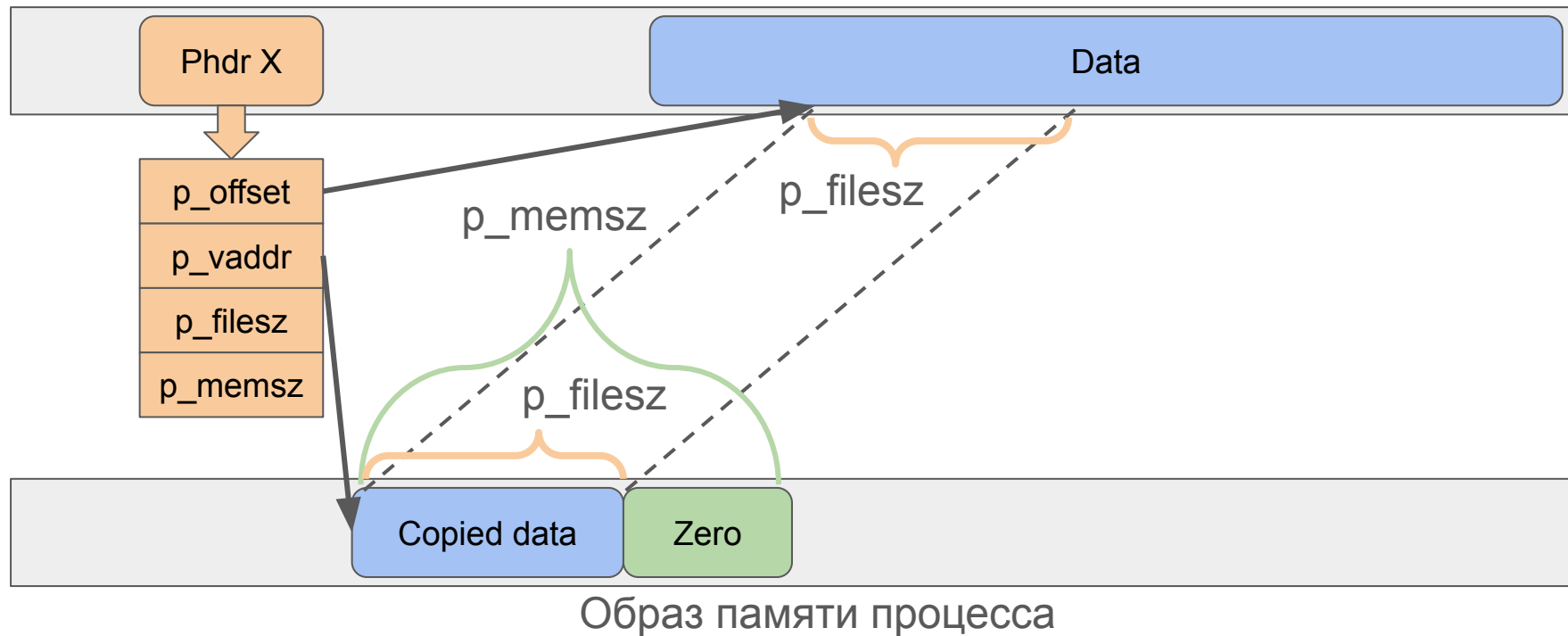
$2^N - 1$



- Квиз
- Кодировка инструкций
- Формат ELF
- Адресное пространство Linux
- **Загрузка ELF**
- Домашнее задание №3

Загрузка ELF

- Загружаем сегменты с PT_LOAD



Загрузка ELF. Библиотеки

- Мы знаем:
 - Как устроен ELF
 - Какие данные и куда нужно загружать из файла
- Будем читать структуру заголовка из файла, искать таблицу заголовков сегментов по смещению
- Читаем таблицу заголовков сегментов
- Ищем нужные сегменты...
- А может есть библиотека для этого?

Загрузка ELF. Библиотеки

- Elfio - <https://github.com/serge1/ELFIO>
- ELFIO::elfio – класс для работы с ELF файлом
 - load(filename) – загрузка существующего файла
 - get_class() – возвращает тип файла (ELFCLASS32, ELFCLASS64)
 - get_encoding() – порядок байтов (ELFDATA2LSB, ELFDATA2MSB)
 - segments – массив сегментов
- ELFIO::segment – класс для работы с сегментами
 - get_type() - Возвращает тип сегмента
 - get_virtual_address() - виртуальный адрес сегмента
 - get_file_size() - размер сегмента в файле
 - get_memory_size() - размер сегмента в памяти
 - get_data() - указатель на начало сегмента

Elfio hello world

```
int main(int argc, char* argv[]) {  
    using namespace ELFIO;  
    elfio reader; // Create elfio reader  
    // Print ELF file segments info  
    Elf_Half seg_num = reader.segments.size();  
    std::cout << "Number of segments: " << seg_num << std::endl;  
    for ( int i = 0; i < seg_num; ++i ) {  
        const segment* pseg = reader.segments[i];  
        std::cout << " [" << i << "] 0x" << std::hex  
        << pseg->get_flags() << "\t0x"  
        << pseg->get_virtual_address() << "\t0x"  
        << pseg->get_file_size() << "\t0x"  
        << pseg->get_memory_size() << std::endl;  
  
        const char* p = reader.segments[i]->get_data(); // Access segments's data  
    }  
}
```

- Квиз
 - Кодировка инструкций
 - Формат ELF
 - Адресное пространство Linux
 - Загрузка ELF
- **Домашнее задание №3**

Домашнее задание №3

- Реализуйте в вашем симуляторе загрузку программы из ELF файла. Можно использовать библиотеку elfio/libelf.