САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Архитектура ЭВМ

# Отчет

по домашней работе № 4

**«ISA»**

Выполнил: Султанов Мирзомансурхон Махсудович

студ. гр. М313Д

Санкт-Петербург

2020

**Цель работы:** знакомство со системой набора команд RISC-V.

**Инструментарий и требования к работе:** Java.

## Теоретическая часть

Система кодирования в RISC-V строится следующим образом, как показано на рисунке 1. Rd (register destination) - регистр, в который направляется результат, rs (register source) – регистр, откуда берётся нужное значение, opcode (operation code) и fuct определяют выполняемую операцию, imm (immediate) в терминах RISC это константа, которую мы можем получить из кода, не обращаясь к памяти.

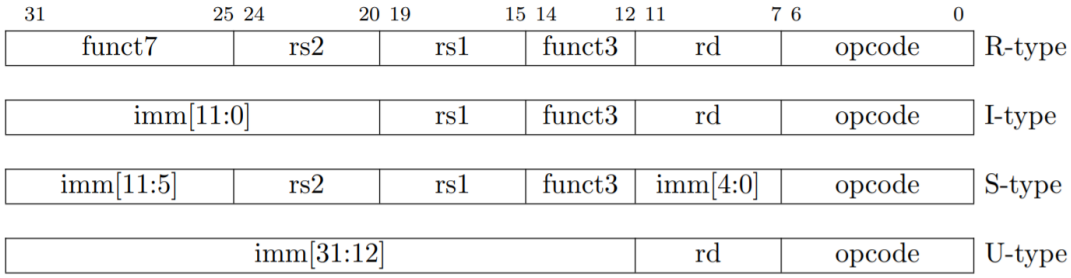


Рисунок 1 – Базовые конструкции кодировки в RISC-V

Так, например, на рисунке 2 показаны все конструкции типа R, где различные значения funct7 + funct3 определяют разные команды.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0000000 | rs2 | rs1 | 000 | rd | 0110011 | ADD |
| 0100000 | rs2 | rs1 | 000 | rd | 0110011 | SUB |
| 0000000 | rs2 | rs1 | 001 | rd | 0110011 | SLL |
| 0000000 | rs2 | rs1 | 010 | rd | 0110011 | SLT |
| 0000000 | rs2 | rs1 | 011 | rd | 0110011 | SLTU |
| 0000000 | rs2 | rs1 | 100 | rd | 0110011 | XOR |
| 0000000 | rs2 | rs1 | 101 | rd | 0110011 | SRL |
| 0100000 | rs2 | rs1 | 101 | rd | 0110011 | SRA |
| 0000000 | rs2 | rs1 | 110 | rd | 0110011 | OR |
| 0000000 | rs2 | rs1 | 111 | rd | 0110011 | AND |

Рисунок 2 – Все конструкции RV32 типа R

Аналогичным образом строятся команды и других типов.

ELF (Executable and Linkable Format) – дословно формат исполняемых и связываемых файлов. Он определяет структуру бинарных файлов, библиотек, и файлов ядра. Спецификация формата позволяет операционной системе корректно интерпретировать содержащиеся в файле машинные команды. Файл ELF, как правило, является выходным файлом компилятора или линкера и имеет двоичный формат. С помощью подходящих инструментов он может быть проанализирован и изучен.

ELF-файл состоит из заголовка ELF и собственно данных. Разберём структуру заголовка на примере, который показан на рисунке 3.

```
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x4013e2
  Start of program headers:          64 (bytes into file)
  Start of section headers:          25376 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         9
  Size of section headers:           64 (bytes)
  Number of section headers:         28
  Section header string table index: 27
```

Рисунок 3 – Пример заголовка ELF-файла

Как видно из рисунка, заголовок начинается с “магического числа”. “Магическое число” даёт общую информацию о файле. Первые 4 байта показывают, что это ELF-файл (45 = E, 4c = L, 46 = F, перед ними стоит значение 7f). Далее идёт поле класса, для которого выделен лишь один байт. Если у нас значение 01, то это означает, что мы работаем с 32-битной архитектуре, если же 02 – то с 64-битной. В нашем же примере ELF-файл использует 64-битную архитектуру. Далее идёт поле “данные”, в котором

два варианта, обозначающие тип endian, который мы используем. Для 01 – little-endian, а для 02 – big-endian. Замечу, что RISC-V не сможет работать с big-endian файлом, так как он предназначен для little-endian. И последнее “магическое число” это версия ELF. На данный момент имеется лишь версия 01, поэтому это число не имеет большого значения. Однако это не полный заголовок. Если рассматривать полный заголовок, то можно получить другую важную информацию, которая приведена на рисунке 4.

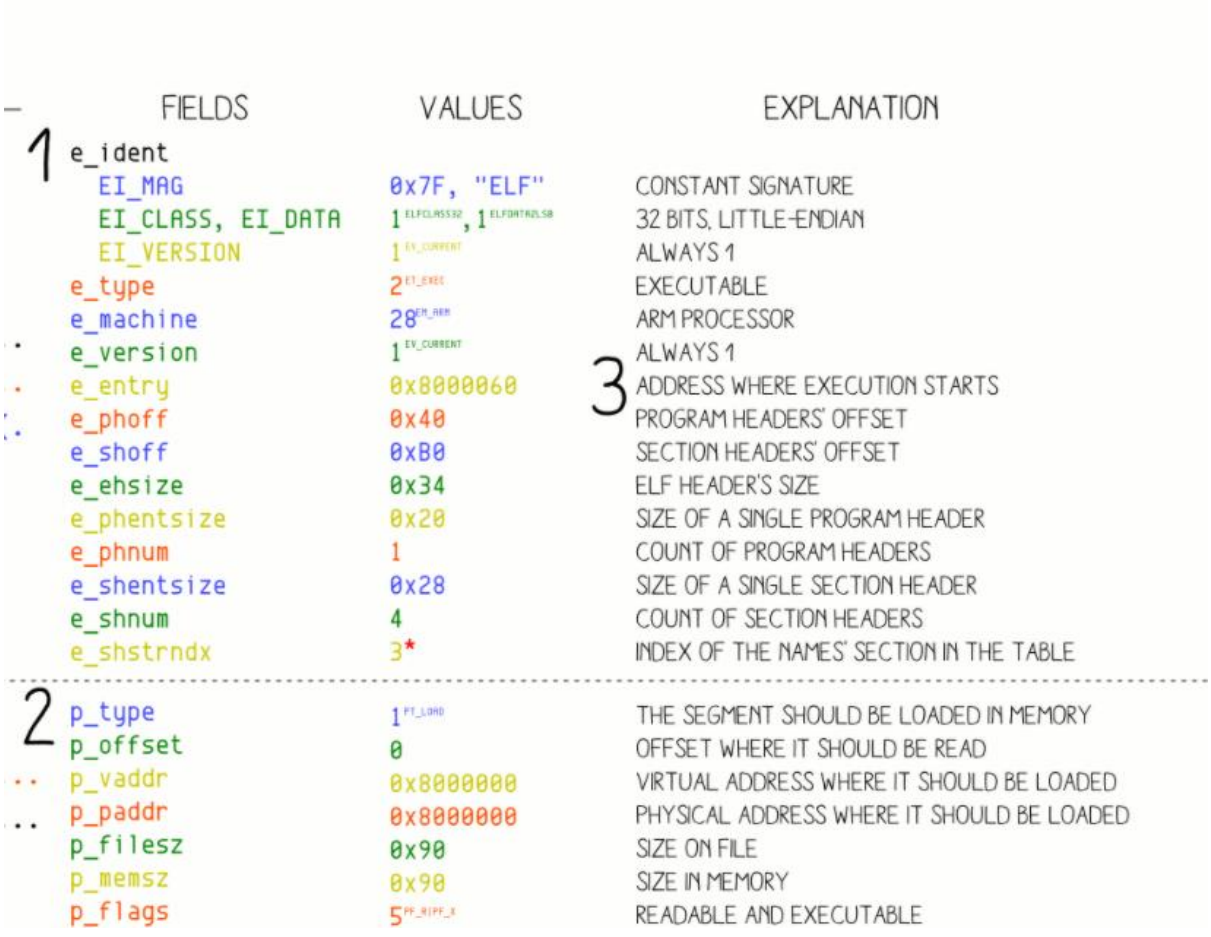


Рисунок 4 – Вся информация, хранящаяся в заголовке ELF-файла

Помимо заголовка, файлы ELF состоят ещё из трёх частей: программные заголовки или сегменты, заголовки секций или секции и собственно данные.

Файл ELF состоит из нуля или более сегментов, и описывает, как создать процесс, образ памяти для исполнения в реальном времени. Когда

ядро видит эти сегменты, оно размещает их в виртуальном адресном пространстве, используя системный вызов mmap(2). Другими словами, конвертирует заранее подготовленные инструкции в образ в памяти. Если ELF-файл является обычным бинарным файлом, он требует эти программные заголовки, иначе он просто не будет работать. Эти заголовки используются, вместе с соответствующими структурами данных, для формирования процесса.

Сегмент же в свою очередь может иметь 0 или более секций. Для исполняемых файлов существует четыре главных секций: text, data, rodata, bss. Каждая из этих секций загружается с различными правами доступа.

Text будет упакован в сегмент с правами на чтение и на исполнение, так как он содержит исполняемый код. Он загружается один раз, и его содержание не изменяется.

Data – это инициализированные данные с правами на чтение и запись.

Rodata – это инициализированные данные с правами только на чтение.

Bss – это неинициализированные данными с правами на чтение/запись.