Домашняя работа №4

ISA. Ассемблер, дизассемблер

**Цель работы:** знакомство с архитектурой набора команд RISC-V.

**Инструментарий и требования к работе:** работа может быть выполнена на любом из следующих языков: C/C++, Python, Java.

**Порядок выполнения работы:**

1. Изучить:
   1. систему кодирования команд RISC-V;
   2. структуру elf-файла;
2. Написать программу-транслятор, с помощью которой можно преобразовывать машинный код в текст программы на языке ассемблера.

**Содержание отчета**

1. Теоретическая часть (порядок выполнения работы п.1).
2. Описание работы написанного кода (пункт 2 из Порядка выполнения, экспериментальная часть, с указанием языка программирования в самом начале части).
3. Пример результата работы написанной программы.
4. Листинг кода с указанием компилятора/интерпретатора (подробнее Оформление кода в отчёте).

**Примечания:**

* + - 1. Плагиат карается отрицательными баллами за всю работу;
      2. Код можно расшарить либо прикрепив исходники архивом в форму, либо как ссылку на репозиторий (git – репозиторий должен быть закрытым и расшаренным с RonoveRaum).

**Дополнительные сведения (код)**

1. Аргументы программе передаются через командную строку:

**hw4.exe <имя\_входного\_elf\_файла> <имя\_выходного\_файла>**

1. Корректно выделяется и освобождается память, закрываются файлы, есть обработка ошибок: не удалось открыть файл, формат файла не поддерживается.  
   Если программе передано значение, которое не поддерживается – следует сообщить об ошибке;
2. В программе можно вызывать только стандартные библиотеки (например, <bits/stdc++.h> таковой не является и ее использование влечет за собой потерю баллов);
3. Если программа использует библиотеки, которые явно не указаны в файле с исходным кодом (например, <algorithm>), то за это также будут снижаться баллы;
4. Если во входном файле встречается команда, которая не распознается программой, то следует выводить unknown\_command.

**Дополнительные сведения**

**ISA**: RISC-V RV32I, RV32M, RVC (нас интересуют только те команды, которые являются сжатой версией команд из RV32I и RV32M).

С RVC можно ознакомиться по ссылке: [waterman-ms.pdf (berkeley.edu)](http://people.eecs.berkeley.edu/~krste/papers/waterman-ms.pdf)

Кодирование: little endian.

ELF file: 32 бита.

Обрабатывать нужно только секции .text, .symtable.

Для каждой строки кода указывается её адрес в hex формате (16 СС).

Обозначение меток **нужно** найти в Symbol Table (.symtable). Если же название метки там не найдено, то используется следующее обозначение: LOC\_%05x, например, LOC\_00000, LOC\_00034.

Для каждой метки перед названием указывается адрес (пример ниже).

**Шаблон файла дизассемблера**

Файл должен состоять из двух частей: .text и .symtab, отделенных друг от друга одной пустой строкой.

Ниже приведены комментарии (строки, начинающиеся с ;) и форматы оформления.

**; формат строк указан по правилам printf (Си)**

**.text**

**; строки оформляются в следующем формате**

**; с меткой:** "%08x %10s: %s %s, %s, %s\n"

**; без метки: метка является пустой строкой**

**; числа - десятичная запись**

**; load/store**

**;** "%08x %10s: %s %s, %s(%s)\n"

**; для c.addi\*sp\* команд sp регистр прописывается явно**

**; примеры:**

00010078 \_start: addi a0, zero, 0

0001007a lui a1, 65536

00010080 lw a0, -24(s0)

00010088 c.addi4spn a0, sp, 8

**; между секциями text и symtab одна пустая строка**

**.symtab**

**; заголовок таблицы**

**;** "%s %-15s %7s %-8s %-8s %-8s %6s %s\n"

**; строки таблицы**

**;** "[%4i] 0x%-15X %5i %-8s %-8s %-8s %6s %s\n"

**; примеры:**

Symbol Value Size Type Bind Vis Index Name

[ 0] 0x0 0 NOTYPE LOCAL DEFAULT UNDEF

[ 1] 0x100b0 0 SECTION LOCAL DEFAULT 1

[ 6] 0x0 0 FILE LOCAL DEFAULT ABS test.c

[ 7] 0x11168 4 OBJECT LOCAL DEFAULT 3 counter.0

[ 8] 0x11967 0 NOTYPE GLOBAL DEFAULT ABS \_\_global\_pointer$

[ 9] 0x11167 0 NOTYPE GLOBAL DEFAULT 2 \_\_SDATA\_BEGIN\_\_

**Остальные комментарии:**

*Fence*: примеры дизасма fence можно найти [**здесь**](https://riscv.org/wp-content/uploads/2018/05/14.25-15.00-RISCVMemoryModelTutorial.pdf), но в рамках этой работы команды fence можно не обрабатывать (в тестах их также не будет).

*Псевдонимы команд*: псевдонимы команд парсить не нужно.

*Вывод меток*: если хочется всегда красиво выводить метки, особенно если их размер больше 10 символов, то можно посчитать размер метки и использовать вместо 10 полученное значение (это изменение не отразиться на баллах за работу, ибо проверяющий в случае чего, сможет поправить ваш формат вывода для тестов при необходимости).

**Update 29.11.** *Вывод регистров:* ABI.

**Update 4.12**

Возможные допустимые варианты вывода RVC:

1) c.or a1, a3

2) c.or a1, a1, a3

*System* команды (ecall, ebreak, csr...) являются частью RV32I и их тоже нужно дизассемблировать.

**Оформление кода в отчёте**

* + - 1. Никаких скринов кода – код в отчет добавляется только текстом;
      2. Шрифт: Consolas (размер 10-14 на ваше усмотрение);
      3. Выравнивание по левому краю;
      4. Подсветка кода допустима. Текст должен быть читаемым (а не светло-серый текст, который без выделения на белом не разобрать);
      5. В раздел Листинг код вставляется полностью в следующем виде:

**<Название файла>**<Его содержимое>  
Файлы исходных кодов разделяются новой строкой.  
Например,  **main.cpp**

int main()

{

return 0;

}

**tmain.cpp**

int tmain()

{

return 666;

}

* + - 1. Фон белый (актуально для тех, у кого копипаста кода идет вместе с фоном темной темы из IDE).

**Оформление дизассемблера в отчёте**

1. Результат работы программы оформляется Consolas (размер 10-14 на ваше усмотрение);
2. Интервал: 1.0;
3. Выравнивание по левому краю;
4. Остальное зафиксировано в Шаблоне файла дизассемблера