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

Инструментарий и требования к работе: Python 3.8

Теоретическая часть

Описание кодирования команд RISC-V:

Чтобы ускорить декодирование, в базовом наборе RISC-V наиболее важные поля находятся в одном и том же месте. Рассмотрим общие положения для кодирования команд:

1. Основной код операции находится в 0-6 битах (здесь и далее подразумевается, что границы включаются), он частично указывает на тип форматов инструкций.
2. Номер регистра-назначения, куда будет записан результат, находится в 7-11 битах.
3. Номер регистра-источника, где содержится первый операнд, находится в 15-19 битах.
4. Номер регистра-источника, где содержится второй операнд, находится в 20-24 битах.
5. 12-14 и 25-31 биты используются для второстепенного кода операции или других данных для инструкций, которые уточняют, какая операция должна быть выполнена.

Нужные данные не всегда могут находиться в регистрах-источниках, такие данные называются immediates. Например, если мы хотим сложить данные из регистра с каким-то произвольным числом, то такое число — immediate, будем обозначать его imm. Количество бит, отведенное для imm, зависит от типа формата инструкции (по сути, от числа регистров в инструкции). Рассмотрим типы формата инструкций:

1. **R-тип:** инструкции с одним регистром назначения и двумя регистрами-источниками без imm. Например, операция сложения данных из двух регистров.
2. **I-тип**: инструкции с одним регистром назначения и одним-регистром источником, для кодирования 0-11 битов imm отводятся 20-31 биты инструкции. Например, уже вышеупомянутая операция сложения данных из регистра и произвольного числа.
3. **S-тип**: инструкции без регистра назначения и с двумя регистрами-источниками, для кодирования 0-11 битов imm отводятся 7-11 и 25-31 биты инструкции. Например, store instructions.
4. **U-тип**: инструкции с регистром назначения без двух регистров-источников и без второстепенного кода операции, для кодирования 12-31 битов imm отводятся 12-31 биты инструкции.

Существуют еще 2 типа формата инструкций — **B-тип** и **J-тип**.

Когда хранятся младшие 12 бит imm, оставшиеся 20 полагаются равными знаковому – imm[11]. Когда хранятся старшие 20 бит imm, оставшиеся 12 полагаются равными 0.

B-тип отличается от S-типа тем, что imm используются для кодирования заведомо четных оффсетов переходов, поэтому хранятся не 0-11, а 1-12 биты imm. Вместо того, чтобы сдвигать все биты влево на 1, средние биты и знаковый бит остаются в фиксированных позициях, а бит, который в S-типе кодировал imm[0], кодирует imm[11]. imm[0] = 0.

В J-типе imm аналогично четный, поэтому imm[0] = 0, в битах инструкции в определенном порядке хранятся 1-20 биты imm, старшие биты равны imm[20].

На рисунках 1 и 2 наглядно представлено, как из битов инструкций соответствующих типов получаются значения imm, opcode, регистров-источников и регистров назначения.

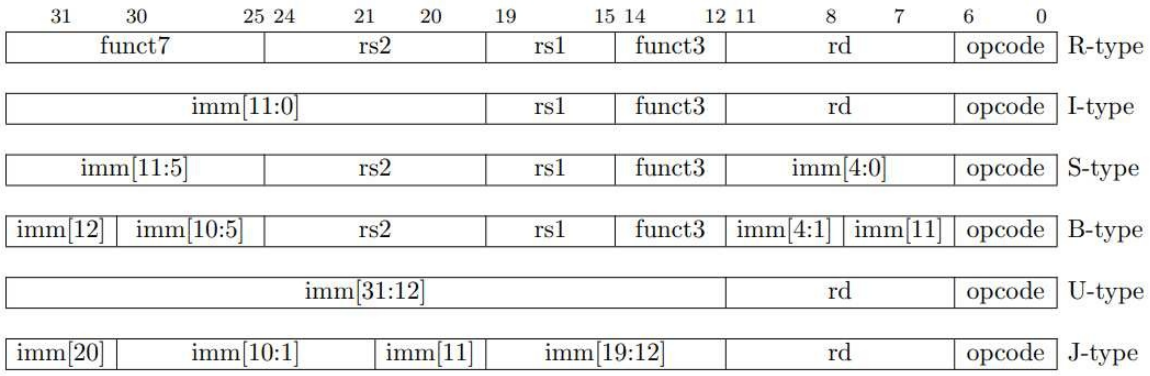

Рисунок №1 – Формат инструкций разных типов

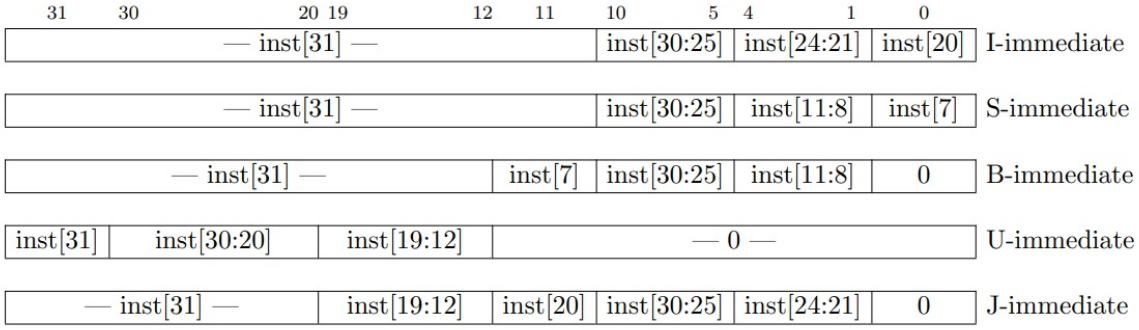

Рисунок №2 – Сопоставление битов imm битам инструкции

Также рассмотрим сопоставление основных команд RISC-V и типов.

FENCE, FENCE.I, ECALL, EBREAK не относятся ни к какому из типов.

FENCE.I, ECALL, EBREAK являются константами. FENCE в 20-23 битах находится множество операций, следующих за FENCE, а в 24-27 битах — множество операций, предшествующих FENCE.

Остальные команды распределены в соответствие с таблицей 1.

Таблица №1 – Соответствие инструкций и типов

R-тип	I-тип	S-тип	U-тип	B-тип	J-тип
ADD, SUB; SLT, SLTU; AND, OR, XOR; SLL, SRL, SRA; MUL MULH, MULHSU, MULHU; DIV(U), REM(U);	ADDI, SUBI; SLTI, SLTUI; ANDI, ORI, XORI; SLLI, SRLI, SRAI; JALR; LW(U), LH(U), LB(U); CSRRW(I), CSRRS(I), CSRRC(I);	SW, SH, SB;	LUI, AUIPC;	BEQ, BNR; BLT, BLTU; BGE, BGEU;	JAL

Структура ELF-файла:

Составляющие любого ELF-файла:

1. ELF-заголовок, в котором указаны общие характеристики файла.
2. Таблица программных заголовков, описывающая сегменты файла.
3. Таблицы заголовков секций, характеризующая секции файла.

Сегменты делятся на секции, типичный кодовый сегмент состоит из секций .init (процедуры инициализации), .plt (секции связок), .text (основной код программы), .data(константы), .finit (процедуры финализации).

Структура ELF-заголовка:

ELF-заголовок находится строго в начале файла и содержит тип, версию формата, архитектуру процессора, виртуальный адрес точки входа, размеры и смещение относительно начала файла таблиц заголовков секций и программы, индекс секции имен секций (.shstrtab) в таблице заголовков секций.

Структура таблицы заголовков программы (program header table):

Находится на позиции e_phoff от начала файла. Содержит заголовки сегментов, которые содержат информацию об отдельном сегменте программы: тип сегмента и действия операционной системы с данным сегментом, расположение сегмента, точка входа сегмента, размер сегмента и флаги доступа к сегменту.

Структура таблицы заголовков секций (section header table):

Находится на позиции e_shoff от начала файла. В последовательных участках размера e_shentsize содержит информацию о каждой секции: имя как оффсет относительно начала .shstrtab, тип секции, флаги секции, виртуальный адрес точки входа, размер секции, связки с другими секциями и дополнительную информацию о секции.

Секции могут хранить строковые константы, отделенные друг от друга байтом 0x00. Пример: .data, .strtab, .shstrtab.

Секции сами могут быть таблицами, как, например, .symtab. Из них можно аналогично получить для каждого элемента его параметры.

Практическая часть

Описание работы программы:

1. Программа проверяет elf-файл на корректность – является ли он elf-файлом и использует ли 32-битный набор команд RISC-V.
2. Затем читает его заголовки, чтобы найти раздел section header table, с помощью которого находится секция с кодом программы (.text), секция с адресами меток (.symtab) и названиями меток (.strtab).
3. Для поиска нужных секций для каждого section header в .shstrtab находится строковое представление имени.
4. Формируется список меток и словарь, в котором адресу метки соответствует ее имя (аналогично находится из .strtab).
5. Каждые 4 байта в .text интерпретируются как команда и ее аргументы в соответствии со спецификацией RISC-V, команды сохраняются в списке команд.
6. Выводится каждая команда из списка. Адрес для первой команды берется из соответствующего параметра секции .text. Если для текущего адреса в словаре есть метка, перед командой выводится имя метки.