# **Лабораторная работа № 1.**

**Знакомство с рабочей средой эмулятора Ripes для работы с процессором RISC-V. Базовый ISA, система команд, состав регистров. Разработка и выполнение простой программы на ассемблере RISC-V.**

* 1. **Цель и задачи работы**

Целями работы являются:

1. Освоение работы с эмулятором Ripes: установка, настройка, трансляция ассемблерной программы, выполнение программы в автоматическом и отладочном режимах.

2. Изучение архитектуры RISC-V, базового набора инструкций и разработка простых программ на ассемблере.

Задачи работы:

а) Ознакомление с основными компонентами компьютера RISC-V.

б) Освоение взаимосвязи между аппаратными и программными средствами компьютера на основе архитектуры системы команд (ISA- Instruction Set Architecture).

в) Изучение языка ассемблера RISC-V и кодирования операторов ассемблера в машинном коде.

г) Освоение компиляции и выполнения ассемблерной программы в автоматическом и отладочном режимах.

**1.2. Основные теоретические сведения**

1. Инструкция по работе с эмулятором Ripes.

2. Описание состава используемых регистров и базового набора команд процессора RISC-V.

3. Краткие сведения по ассемблеру RISC-V.

**1.3. Задание к лабораторной работе**

1. Разработайте процедуру на ассемблере, которая для целочисленных 32-битных входных переменных x, y, z и констант a, b, c вычисляет выражение

R = f (x, y, z, a, b, c) ,

выбираемое по таблице 1 в соответствии с вашим вариантом.

Таблица 1

|  |  |
| --- | --- |
| Вариант | Выражение (представленное в формате С/С++) |
| 1 | ((x >> a) - (z | (-b))) - (y << c) |
| 2 | ((x | b) + (z + c)) | (y | (-a)) |
| 3 | ((x << a) + (z | b)) - (y | (-c)) |
| 4 | ((x & (-a)) ^ (y >> c)) + (z << b) |
| 5 | ((z + a) + (x + c)) + (y ^ (-b)) |
| 6 | ((y + a) ^ (z | (-b))) + (x - c) |
| 7 | ((x << c) + (y & (-a))) | (z - b) |
| 8 | ((z + c) ^ (y ^ (-a))) | (x >> b) |
| 9 | ((x & (-a)) & (y | b)) - (z - c) |
| 10 | ((x >> c) + (z & (-a))) & (y << b) |
| 11 | ((x ^ (-c)) ^ (z | b)) ^ (y - a) |
| 12 | ((z >> b) | (y | (-a))) | (x - c) |
| 13 | ((y << c) ^ (x ^ (-a))) & (z & b) |
| 14 | ((x & (-c)) | (z - a)) - (y + b) |
| 15 | ((x << a) ^ (y & (-c))) | (z >> b) |
| 16 | ((z ^ (-b)) ^ (x & c)) & (y << a) |
| 17 | ((x ^ (-a)) - (y - c)) & (z & b) |
| 18 | ((y >> b) ^ (z << a)) ^ (x & (-c)) |
| 19 | ((x + a) + (y & c)) & (z & (-b)) |
| 20 | ((z & b) + (x | (-a))) | (y - c) |
| 21 | ((y ^ a) - (z >> b)) - (x ^ (-c)) |
| 22 | ((x >> a) & (y + b)) + (z | (-c)) |
| 23 | ((x << a) | (y & (-c))) ^ (z + b) |
| 24 | ((x ^ (-b)) & (z ^ c)) | (y + a) |
| 25 | ((x ^ b) - (z & (-c))) ^ (y << a) |
| 26 | ((y ^ a) ^ (z | (-c))) - (x - b) |
| 27 | ((y & a) & (x >> b)) & (z | (-c)) |
| 28 | ((y >> a) - (z | b)) | (x & (-c)) |
| 29 | ((y ^ (-a)) + (x >> b)) + (z | c) |
| 30 | ((z + a) - (y | (-c))) | (x - b) |

В выражениях используются следующие константы:

|  |  |
| --- | --- |
| Константа | Значение |
| a | [Сумма цифр студ. билета] |
| b | [Количество букв в фамилии] |
| c | [Количество букв в полном имени] |

2. Напишите программу, которая для двух наборов исходных данных x, y, z выполняет вычисление заданного выражения с помощью разработанной процедуры (вызывая её два раза), сохраняет в регистрах и выводит на экран результаты вычислений. Один вызов процедуры должен выполняться с помощью псевдоинструкции call, другой – с помощью инструкции jal.

Начальные значения {x1, y1, z1} расположить в регистрах a2, a3, a4; значения {x2, y2, z2} расположить в регистрах a5, a6, a7; значения констант a, b, c расположить в регистрах s0, s1, s2. Результаты вычисления {r1, r2} записать в регистры а1, а2.

В исходном коде обязательно должны быть употреблены следующие псевдоинструкции: call (ровно 1 раз), ret (ровно 1 раз), mv (как минимум 1 раз), li (как минимум 2 раза: 1 раз – преобразующаяся в две инструкции; 1 раз – преобразующаяся в одну инструкцию).

Моделируемые вычисления (формула, входные данные, результаты) должны выводиться в консоль.

**Отчёт по лабораторной работе должен содержать:**

1) Исходное задание и формулу с подставленными константами, по которой должны быть выполнены вычисления.

2) Результаты отладки программы в пошаговом режиме под управлением отладчика с фиксацией содержимого используемых регистров и ячеек памяти до и после выполнения команды в форме таблицы:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Адрес инстр. | (Псевдо-) инстр. | Инструкция(и) | 16-ричный код инстр. | Содержимое регистров и ячеек памяти | |
| до вып. инструкции | после вып. инструкции |

Если одной псевдоинструкции соответствует более одной инструкции, то в таблице должны быть отражены результаты выполнения каждой инструкции.

Таблица может приводиться не целиком, а завершаться первой (псевдо‑)инструкцией, которая будет выполняться в результате второго вызова вычисления выражения.

3) Лог тестирования программы в виде таблицы с входными данными и результатами. Должно быть представлено не менее 4 результатов вычисления значения по формуле.

4) Код программы. Код должен быть с подробными комментариями: после каждой вычислительной команды должно быть указано полученное к текущему моменту значение.

**Контрольные вопросы для защиты работы.**

1. Какие группы команд входят в состав базового набора команд процессоров RISC-V?

2. Назначение и отличие команд безусловного перехода JAL и JALR.

3. Как реализуется простой безусловный переход, не связанный с вызовом процедуры?

4. Назовите несколько основных отличий систем команд процессоров Intel x86 и RISC-V.

5. Назовите типы основных директив ассемблера RISC-V.

6. Состав и назначение регистров процессора RISC-V.

7. Назначение и особенности реализации псевдокоманд ассемблера RISC-V.

8. Как реализуются (расширяются) псевдокоманды вызова процедуры и возврата из процедуры.

9. Напишите возможные варианты псевдокоманд с соответствующими им командами:

9а) пересылки данных между регистрами;

9б) умножения регистра на -1;

9в) загрузки в регистр 32-битной константы.

10. Какая информация (какие поля) есть в 32-битном коде машинной инструкции RISC-V?

11. Что делает команда <некоторая команда из исходного кода>?

**Прочие вопросы (не для защиты, но для подготовки к экзамену).**

1. В чём состоят особенности концепции процессора RISC-V и её отличие от концепции процессоров ARM?

2. В чём заключаются и что обеспечивают модульность и расширяемость процессоров RISC-V?

3. В чём преимущество трёхадресного формата инструкций (команд) процессоров RISC-V?

4. Что такое ассемблер, язык ассемблера и машинный код?

5. Отличие назначения и использования директив и команд ассемблера RISC-V.