

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Организация ЭВМ и систем»

Тема: 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.

Студент гр. 3384

Рудаков А.Л.

Преподаватель

Ковалев А.Д.

Санкт-Петербург

2024

Цель 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.
 - а) Ознакомиться 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 этапов компиляции и выполнения ассемблерной программы в автоматическом и отладочном режимах.

Задание.

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

$$R = f(x, y, z, a, b, c) = ((x + a) + (y \& c)) \& (z \& (-b)) \quad (\text{Var. 19})$$

выбираемое по таблице 1 в соответствии с вашим номером в списке группы.

2. Напишите программу, которая для двух наборов исходных данных x, y, z выполняет вычисление заданного выражения с помощью разработанной процедуры, сохраняет в регистрах и выводит на экран результаты вычислений.

Основные теоретические положения.

RISC-V — это открытая архитектура, что позволяет любому желающему использовать, модифицировать и разрабатывать свои

собственные процессоры на ее основе без необходимости платить лицензионные сборы.

RISC-V предлагает модульную архитектуру, где базовый набор инструкций может быть дополнен расширениями в зависимости от нужд конкретного приложения.

ARM — это закрытая архитектура, и компании должны платить лицензии для использования её технологий. Это создает дополнительные барьеры для входа.

В RISC-V набор инструкций относительно простой и понятный, что упрощает процесс разработки и оптимизации компиляторов.

ARM часто включает более сложные инструкции, что может облегчить задачи программирования, но увеличивает сложность реализации на уровне аппаратуры.

Модульность RISC-V позволяет разработчикам адаптировать процессоры под конкретные задачи, добавляя или исключая функциональные блоки. Это достигается за счет четко определенных расширений инструкционного набора, которые могут включать поддержку целочисленных операций, операций с плавающей точкой, атомарных инструкций и специализированных расширений для обработки сигналов, криптографии и т. д.

Выполнение работы.

В начале программы описан сегмент *.data*, в котором хранятся тестовые данные $\{x1, y1, z1\}$ и $\{x2, y2, z2\}$.

Далее в сегменте *.text* записываются значения $x1, y1, z1, a, b, c$ в регистры $a2, a3, a4, s0, s1, s2$ соответственно, после чего выполняется процедура f .

Внутри процедуры вначале выполняется сложение x и a , и записывается в $t0$, после y & c записывается в $t1$ и значения $t0$ и $t1$ складываются и записываются в $t0$. После идет инвертирование b и

выполнение $z \& (-b)$ с записью в $t1$. Далее выполняется $t0 \& t1$, что является результатом выражения и записывается в $a0$. Далее идет возврат при помощи `ret`.

После происходит вывод числа в $a0$ путем записи в $a7$ функции 1 и вызова функции вывода. Далее идет запись значений $x2, y2, z2$ в $a2, a3, a4$ и выполнение процедуры f для этих значений, после вывод результаты и завершение программы.

Вывод.

В ходе выполнения лабораторной работы была разработана программа, выполняющая арифметические и логические операции над целыми числами, путем работы с регистрами на ассемблере RISC-V.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ.

```
.data
    x1:  .word 10
    y1:  .word 5
    z1:  .word 3
    x2:  .word 15
    y2:  .word 12
    z2:  .word 6

.text
    lw    a2, x1
    lw    a3, y1
    lw    a4, z1
    li    s0, 20
    li    s1, 7
    li    s2, 9
    jal   f

    li    a7, 1
    ecall

    lw    a2, x2
    lw    a3, y2
    lw    a4, z2
    jal   f

    li    a7, 1
    ecall

    li    a7, 10
    ecall

f:
    add   t0, a2, s0
    and   t1, a3, s2
    add   t0, t0, t1
    neg   t2, s1
    and   t1, a4, t2
```

```
and a0, t0, t1  
ret
```