**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургсчкий государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

**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. 3388 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Павлов А.Р. |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Молодцев Д.А |

Санкт-Петербург

2024

**Цель работы**

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.

**Задание:**

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

R = f (x, y, z),

выбираемое по таблице в соответствии с вашим вариантом(Таблица 1).

Таблица 1

|  |
| --- |
| Выражение (представленное в формате С/С++) |
| (a2 & (a3 - (a3 & (a4 \* a4))) + ((-a4) \* (a4 & a2))) |

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.

**Основные теоретические положения:**

**1. Описание состава используемых регистров:**

Первый регистр, x0 имеет специальное назначение, он содержит 0. Вне зависимости от того, какое значение вы в него записываете, при чтении из этого регистра вы всегда получите 0. Псевдонимом регистра x0 является zero.

Псевдонимы других регистров:

ra return address (адрес возврата). Используется для записи адреса возврата перед вызовом подпрограммы.

t0 - t6 temporary registers (регистры временных переменных). Подпрограммы не обязаны их сохранять.

s0 - s11 saved registers (сохраняемые регистры). Подпрограммы обязаны сохранять их состояние.

a0 - a7 function arguments (аргументы функций). Перед вызовом подпрограммы вы передаёте аргументы в эти регистры.

**Описание базового набора команд процессора RISC-V:**

1. Арифметические и логические операции: add, sub, and, or, xor и др.
2. Операции с памятью: load(lb, lh, lw) , store(sb, sh, sw), fence и др.
3. Управление переходами: branch(beq, bne, blt, bge, ...), jump(jal, jalr), call, return и др.
4. Управление регистрами: move (mv) , lui, auipc и др.
5. Привилегированные(системные) операции: ecall, ebreak и др.

**2. Краткие сведения по ассемблеру RISC-V.**

В концепции RISC инженеры устранили главный недостаток CISC-архитектур: использование большого числа функционально сложных команд. В RISC-архитектурах используется небольшое число команд фиксированной длины, но при этом увеличивается число регистров, чтобы иметь большее пространство для работы с данными и реже обращаться к памяти.

В архитектуре RISC-V имеется обязательное для реализации базовое подмножество в количестве 47 команд и несколько стандартных опциональных расширений. В базовый набор входят: минимальный набор команд арифметических/битовых операций на регистрах, команд для выполнения операций с памятью (load/store), команд условной и безусловной передачи управления/ветвления, а также небольшое число служебных инструкций (см. таблицу далее). Команды базового набора имеют длину 32 бита с выравниванием на границу 32-битного слова.

Операции условных переходов (ветвления) не используют каких-либо общих флагов, как результатов ранее выполненных операций сравнения, а непосредственно сравнивают свои регистровые операнды. Базис операций сравнения минимален, а для поддержки комплементарных операций операнды просто меняются местами.

**Выполнение работы**

1. Исходное задание было приведено выше. Формула с подставленными константами, по которой должны быть выполнены вычисления:

**(a2 & (a3 - (a3 & (a4 \* a4))) + ((-a4) \* (a4 & a2)))**

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(Таблица 2):

Таблица 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Адресс инстр.** | **(Псевдо-) инстр.** | **Инструкция** | **16-ричный код инстр.** | **Содержимое регистров и ячеек памяти** | |
| **до вып. функции** | **после вып. функции** |
| 0 | la a0, formula | auipc x10 0x10000 | 10000517 | x10 0x00000000 | x10 0x10000000 |
| 4 | addi x10 x10 0 | 00050513 |  |  |
| 8 | li a7, 4 | addi x17 x0 4 | 00400893 | x17 0x00000000 | x17 0x00000000 |
| с | ecall | ecall | 00000073 |  |  |
| 10 | la a0, test\_1 | auipc x10 0x10000 | 10000517 | x10 0x00000000 | x10 0x00000000 |
| 14 | addi x10 x10 44 | 02c50513 | x10 0x00000000 | x10 0x10000000 |
| 18 | li a7, 4 | addi x17 x0 4 | 00400893 | x17 0x00000000 | x17 0x00000004 |
| 1с | ecall | ecall | 00000073 |  |  |
| 20 | li a0, x\_1 | addi x10 x0 0 | 00000513 | x10 0x10000000 | x10 0x10000010 |
| 24 | jal print\_with\_sep | Jal x1 200 <print\_with\_sep> | 0c8000ef | x1 0x00000000 | x1 0x000000010 |
| 28 | li a0, y\_1 | addi x10 x0 1 | 00100513 | x10 0x00000010 | x10 0x0000003b |
| 2c | jal print\_with\_sep | jal x1 192 <print\_with\_sep> | 0c0000ef | x1 0x00000010 | x1 0x00000000 |
| dc | li a7, 1 | addi x17 x0 1 | 00100893 | x17 0x00000004 | x17 0x00000004 |
| e0 | ecall | ecall | 00000073 |  |  |
| e4 | la a0, sep | auipc x10 0x10000 | 10000517 | x10 0x1000003c | x10 0x100000e4 |
| e8 | addi x10 x10 -156 | f6450513 | x10 0x100000e4 | x10 0x10000058 |
| ec | li a7, 4 | addi x17 x0 4 | 00400893 | x17 0x00000004 | x17 0x00000001 |
| f0 | ecall | ecall | 00000073 |  |  |
| f4 | ret | jalr x0 x1 0 | 00008067 |  |  |
| 30 | li a0, z\_1 | addi x10 x0 2 | 00200513 | x10 0x10000058 | x10 0x10000058 |
| 34 | li a7, 1 | addi x17 x0 1 | 00100893 | x17 0x00000001 | x17 0x00000004 |
| dc | li a7, 1 | addi x17 x0 4 | 00400893 | x17 0x00000004 | x17 0x00000001 |
| e0 | ecall | ecall | 00000073 |  |  |
| e4 | la a0, sep | auipc x10 0x10000 | 10000517 | x10 0x10000008 | x10 0x100000e4 |
| e8 | addi x10 x10 -156 | f6450513 | x10 0x100000e4 | x10 0x10000058 |
| ec | li a7, 4 | addi x17 x0 4 | 00400893 | x17 0x00000004 | x17 0x00000001 |
| f0 | ecall | ecall | 00000073 |  |  |
| f4 | ret | jalr x0 x1 0 | 00008067 |  |  |
| 3c | la a0, test\_2 | auipc x10 0x10000 | 10000517 | x10 0x10000058 | x10 0x1000000a |
| 40 | addi x10 x10 14 | 00e50513 | x10 0x100000a | x10 0x10000002 |
| 44 | li a7, 4 | addi x17 x0 4 | 00400893 | x17 0x00000004 | x17 0x00000001 |
| 48 | ecall | ecall | 00000073 |  |  |
| 4c | li a0, x\_2 | addi x10 x0 11 | 00b00513 | x10 0x10000002 | x10 0x1000003c |
| 50 | jal print\_with\_sep | jal x1 156 <print\_with\_sep> | 09c000ef | x1 0x00000000 | x1 0x00000030 |
| 54 | li a0, y\_2 | addi x10 x0 21 | 01500513 | x10 0x1000003c | x10 0x1000004a |
| 58 | jal print\_with\_sep | jal x1 148 <print\_with\_sep> | 094000ef | x1 0x00000030 | x1 0x00000054 |
| dc | li a7, 1 | addi x17 x0 4 | 00400893 | x17 0x00000004 | x17 0x00000001 |
| e0 | ecall | ecall | 00000073 |  |  |
| e4 | la a0, sep | auipc x10 0x10000 | 10000517 | x10 0x1000004a | x10 0x10000006 |
| e8 | addi x10 x10 -156 | f6450513 | x10 0x10000006 | x10 0x100000e4 |
| ec | li a7, 4 | addi x17 x0 4 | 00400893 | x17 0x00000004 | x17 0x00000001 |
| f0 | ecall | ecall | 00000073 |  |  |
| f4 | ret | jalr x0 x1 0 | 00008067 |  |  |
| 5c | li a0, z\_2 | addi x10 x0 278 | 11600513 | x10 0x10000058 | x10 0x10000058 |
| 60 | li a7, 1 | addi x17 x0 1 | 00100893 | x17 0x00000001 | x17 0x00000004 |
| dc | li a7, 1 | addi x17 x0 4 | 00400893 | x17 0x00000004 | x17 0x00000001 |
| e0 | ecall | ecall | 00000073 |  |  |
| e4 | la a0, sep | auipc x10 0x10000 | 10000517 | x10 0x10000058 | x10 0x10000009 |
| e8 | addi x10 x10 -156 | f6450513 | x10 0x10000009 | x10 0x100000e4 |
| ec | li a7, 4 | addi x17 x0 4 | 00400893 | x17 0x00000004 | x17 0x00000001 |
| f0 | ecall | ecall | 00000073 |  |  |
| f4 | ret | jalr x0 x1 0 | 00008067 |  |  |
| 5c | li a0, z\_2 | addi x10 x0 278 | 11600513 | x10 0x100000e4 | x10 0x10000058 |
| 60 | li a7, 1 | addi x17 x0 1 | 00100893 | x17 0x00000001 | x17 0x00000004 |
| 64 | ecall | ecall | 00000073 |  |  |
| 68 | la a0, res | auipc x10 0x10000 | 10000517 | x10 0x10000058 | x10 0x10000058 |
| 6c | addi x10 x10 -13 | ff350513 | x10 0x10000058 | x10 0x10000116 |
| 70 | li a7, 4 | addi x17 x0 4 | 00400893 | x17 0x00000004 | x17 0x00000001 |
| 74 | ecall | ecall | 00000073 |  |  |
| 78 | li a2, x\_1 | addi x12 x0 0 | 00000613 | x12 0x00000000 | x12 0xfffffffb |
| 7c | li a3, y\_1 | addi x13 x0 1 | 00100693 | x13 0x00000000 | x13 0x00000008 |
| 80 | li a4, z\_1 | addi x14 x0 2 | 00200713 | x14 0x00000000 | x14 0x0000000a |
| 84 | li a5, x\_2 | addi x15 x0 11 | 00b00793 | x15 0x00000000 | x15 0x00000006 |
| 88 | li a6, y\_2 | addi x16 x0 21 | 01500813 | x16 0x00000000 | x16 0x00000009 |
| 8c | li a1, z\_2 | addi x11 x0 278 | 11600893 | x11 0x00000000 | x11 0x0000000e |
| 90 | call calculation | auipc x1 0x0 <main> | 00000097 | x1 0x00000054 | x1 0x0000005c |
| 94 | jalr x1 x1 56 | 038080e7 | x1 0x0000005c | x1 0x00000056 |
| b8 | mul t0, a4, a4 | mul x5 x14 x14 | 02e702b3 | x5 0x00000000 | x5 0x00000064 |
| bc | and t0, a3, t0 | and x5 x13 x5 | 0056f2b3 | x5 0x00000064 | x5 0x00000000 |
| c0 | sub t0, a3, t0 | sub x5 x13 x5 | 405682b3 | x5 0x00000000 | x5 0x00000008 |
| c4 | neg t1, a4 | sub x6 x0 x14 | 40e00333 | x6 0x00000000 | x6 0xfffffff6 |
| c8 | and t2, a4, a2 | and x7 x14 x12 | 00c773b3 | x7 0x00000000 | x7 0x0000000a |
| cc | mul t1, t1, t2 | mul x6 x6 x7 | 02730333 | x6 0xfffffff6 | x6 0xffffff9c |
| d0 | add t0, t0, t1 | add x5 x5 x6 | 006282b3 | x5 0x00000008 | x5 0xffffffa4 |
| d4 | and a0, a2, t0 | and x10 x12 x5 | 00c57533 | x10 0x10000000 | x10 0xffffffa0 |
| d8 | jr ra | jalr x0 x1 0 | 00008067 |  |  |
| 98 | mv a2, a5 | addi x12 x15 0 | 00078613 | x12 0x00000014 | x12 0x10000006 |
| 9c | mv a3, a6 | addi x13 x16 0 | 00080693 | x13 0x00000008 | x13 0x10000009 |
| a0 | mv a4, a7 | addi x14 x17 0 | 00088713 | x14 0x0000000a | x14 0x0000000e |
| a4 | jal calculation | jal x1 32 <calculation> | 020000ef | x1 0x00000056 | x1 0x00000098 |
| b8 | mul t0, a4, a4 | mul x5 x14 x14 | 02e702b3 | x5 0xffffffa4 | x5 0x000000c4 |
| bc | and t0, a3, t0 | and x5 x13 x5 | 0056f2b3 | x5 0x000000c4 | x5 0x00000000 |
| c0 | sub t0, a3, t0 | sub x5 x13 x5 | 405682b3 | x5 0x00000000 | x5 0x00000009 |
| c4 | neg t1, a4 | sub x6 x0 x14 | 40e00333 | x6 0xffffff9c | x6 0xfffffff2 |
| c8 | and t2, a4, a2 | and x7 x14 x12 | 00c773b3 | x7 0x0000000a | x7 0x00000006 |
| cc | mul t1, t1, t2 | mul x6 x6 x7 | 02730333 | x6 0 0xfffffff2 | x6 0xffffffac |
| d0 | add t0, t0, t1 | add x5 x5 x6 | 006282b3 | x5 0x00000009 | x5 0xffffffb5 |
| d4 | and a0, a2, t0 | and x10 x12 x5 | 00c57533 | x10 0x00000000 | x10 0x00000004 |
| d8 | jr ra | jalr x0 x1 0 | 00008067 |  |  |

Таблица была приведена не целиком, она завершается после четырёх инструкций после второго вызова функции вычисления.

3. Лог тестирования программы в виде таблицы с входными данными и результатами(Таблица 3):

Таблица 3

|  |  |  |  |
| --- | --- | --- | --- |
| № | Исходные данные | Ожидаемые результаты | Полученный результат |
| 1 | x1 = 1, y1 = 1, z1 = 2  x2 = 11, y2 = 21, z2 = 278 | r1 = 0  r2 = 0 | r1 = 0  r2 = 0 |
| 2 | x1 = 0, y1 = 0, z1 = -10  x2 = 0, y2 = -5, z2 = 0 | r1 = -28  r2 = 0 | r1 = -28  r2 = 0 |
| 3 | x1 = 3, y1 = 7, z1 = -10  x2 = 0, y2 = 0, z2 = -1000000 | r1 = 4  r2 = -2082432 | r1 = 4  r2 =-2082432 |
| 4 | x1 = -5, y1 = 8, z1 = 10  x2 = 6, y2 = 9, z2 = 14 | r1 = -96  r2 = 4 | r1 = -96  r2 = 4 |

4. Исходный код разработанной программы представлен в приложении.

**Вывод**

В результате выполнения лабораторной работы были получены навыки работы с эмулятором Ripes, включающие выполнение ассемблерной программы в автоматическом и отладочном режимах. Также была изучена архитектура RISC-V, базовый набор инструкций. Была разработана программа вычисления значений по заданной формуле на ассемблере RISC-V.

**ПРИЛОЖЕНИЕ**

**Код программы**

.data

# Константы

.equ x\_1, -5

.equ y\_1, 8

.equ z\_1, 10

.equ x\_2, 6

.equ y\_2, 9

.equ z\_2, 14

# Формула для вывода в консоль

formula: .string "Formula: (a2 & (a3 - (a3 & (a4 \* a4))) + ((-a4) \* (a4 & a2)))"

test\_1: .string "\nx1, y1, z1: "

test\_2: .string "\nx2, y2, z2: "

sep: .string ", "

res: .string "\nres: "

.text

main:

# Печать формулы

la a0, formula # Загрузка адреса строки формулы в a0

li a7, 4 # Код системного вызова для печати строки

ecall # Вызов системного вызова

# Печать первого набора данных

la a0, test\_1 # Загрузка адреса строки test\_1 в a0

li a7, 4 # Код системного вызова для печати строки

ecall # Вызов системного вызова

li a0, x\_1 # Загрузка значения константы x\_1 в a0

jal print\_with\_sep # Печать значения с разделителем

li a0, y\_1 # Загрузка значения константы y\_1 в a0md

jal print\_with\_sep # Печать значения с разделителем

li a0, z\_1 # Загрузка значения константы z\_1 в a0

li a7, 1 # Код системного вызова для печати целого числа

ecall # Вызов системного вызова

# Печать второго набора данных

la a0, test\_2 # Загрузка адреса строки test\_2 в a0

li a7, 4 # Код системного вызова для печати строки

ecall # Вызов системного вызова

li a0, x\_2 # Загрузка значения константы x\_2 в a0

jal print\_with\_sep # Печать значения с разделителем

li a0, y\_2 # Загрузка значения константы y\_2 в a0

jal print\_with\_sep # Печать значения с разделителем

li a0, z\_2 # Загрузка значения константы z\_2 в a0

jal print\_with\_sep # Печать значения с разделителем

la a0, res # Загрузка адреса строки res в a0

li a7, 4 # Код системного вызова для печати строки

ecall # Вызов системного вызова

# Загрузка констант для вычислений

li a2, x\_1 # Загрузка значения константы x\_1 в регистр a2 (a2 будет -5)

li a3, y\_1 # Загрузка значения константы y\_1 в регистр a3 (a3 будет 8)

li a4, z\_1 # Загрузка значения константы z\_1 в регистр a4 (a4 будет 10)

li a5, x\_2 # Загрузка значения константы x\_2 в регистр a5 (a5 будет 6)

li a6, y\_2 # Загрузка значения константы y\_2 в регистр a6 (a6 будет 9)

li a1, z\_2 # Загрузка значения константы z\_2 в регистр a1 (a1 будет 14)

call calculation # Вызов процедуры расчета

jal print\_with\_sep # Печать результата с разделителем

# Второй набор данных для вычислений

mv a2, a5 # Копирование значения регистра x12 (x\_2) в регистр a2 (переносим первый результат)

mv a3, a6 # Копирование значения регистра x15 (y\_2) в регистр a3

mv a4, a1 # Копирование значения регистра x16 (z\_2) в регистр а4

jal calculation # Вызов процедуры расчета для второго набора данных

jal print\_with\_sep # Печать результата с разделителем

li a7, 10 # Код системного завершения программы

ecall # Вызов системного завершения

# Процедура расчета выражения

calculation:

mul t0, a4, a4 # Вычисление: t0 = a4 \* a4

and t0, a3, t0 # Вычисление: t0 = а3 & (a4 \* а4)

sub t0, a3, t0 # Вычисление: t0 = а3 - (а3 & (а4 \* а4))

neg t1, a4 # Вычисление: t1 = -a4

and t2, a4, a2 # Вычисление: t2 = а4 & а2

mul t1, t1, t2 # Вычисление: t1 = (-a4) \* (а4 & а2)

add t0, t0, t1 # Cложение результатов: t0 = (а3 - (а3 & (а4 \* а4))) + ((-a4) \* (а4 & а2))

and a0, a2, t0 # Финальный результат: а0 = а2 & t0

jr ra # Возврат из процедуры расчета

# Процедура печати числа с разделителем

print\_with\_sep:

li a7, 1 # Код системного вывода для печати целого числа

ecall # Вызов системного вывода

la a0, sep # Загрузка адреса строки sep (", ") в а0

li a7, 4 # Код системного вызова для печати строки

ecall # Вызов системного вывода

ret # Возврат из процедуры печати