**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. 3344 | Тукалкин В.А. |
| Преподаватель | Фирсов М.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 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++) |
| 20 | ((z & b) + (x | (-a))) | (y - c) |

В выражениях используются следующие константы:

|  |  |
| --- | --- |
| Константа | Значение |
| a | 18 |
| b | 8 |
| c | 8 |

Выражение с подставленными значениями:

((z & 8) + (x | (-18))) | (y - 8)

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**

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

Базовый набор команд RISC-V включает в себя следующие группы команд:

* Арифметические операции: add, sub, mul, div.
* Логические операции: and, or, xor.
* Операции сдвига: sll (логический сдвиг влево), srl (логический сдвиг вправо), sra (арифметический сдвиг вправо).
* Операции сравнения: slt (установка меньше), sltu (установка меньше без знака).
* Операции загрузки и сохранения: lw (загрузка слова), sw (сохранение слова).
* Команды перехода: beq (переход, если равно), bne (переход, если не равно), jal (переход с сохранением адреса возврата), jalr (переход с сохранением адреса возврата через регистр).

Процессор RISC-V имеет 32 регистра общего назначения (x0-x31), каждый из которых имеет размер 32 бита. Некоторые регистры имеют специальное назначение:

* x0 (zero): Всегда содержит значение 0.
* x1 (ra): Используется для хранения адреса возврата при вызове подпрограмм.
* x2 (sp): Указатель стека.
* x8 (s0): Сохраненный регистр, используется для хранения временных данных.
* x9 (s1): Сохраненный регистр, используется для хранения временных данных.
* x10-x17 (a0-a7): Аргументы и возвращаемые значения функций.
* x18-x27 (s2-s11): Сохраненные регистры, используются для хранения временных данных.
* x28-x31 (t3-t6): Временные регистры.

Краткие сведения по ассемблеру RISC-V:

Язык ассемблера RISC-V — это низкоуровневый язык программирования, который используется для непосредственного управления процессором. Основные элементы языка:

* Команды: Основные инструкции процессора, такие как add, sub, lw, sw, jal, jalr.
* Псевдокоманды: Высокоуровневые команды, которые транслируются в одну или несколько машинных команд, например, li (загрузка константы), mv (пересылка данных), call (вызов процедуры), ret (возврат из процедуры).
* Директивы: Инструкции для ассемблера, такие как .data (секция данных), .text (секция кода), .word (объявление 32-битного слова).

**Выполнение работы**

1) Процедура main

Состоит из двух разделов. В разделе data хранятся значения переменных, которые впоследствии будут выводиться в консоль.

В разделе text содержится код программы. Производится инициализация констант a, b, c и загружается первые набор данных в регистры a2, a3, a4, далее вызывается процедура calculate с помощью call и сохраняется результат в a1 из a1. Загрузка второго набора данных в a5, a6, a7, вызывается процедура calculate и сохраняется результат в a2. Также происходит вывод формулы, значений и результатов.

2) Процедура calculate

Разработана процедура, вычисляющая выражение для целочисленных 32-битных входных переменных согласно варианту. Внутри вычисляется выражение для набора данных {x1, y1, z1}. Использованы инструкции or (побитовое «или»), and (побитовое «и»), sub (вычитание), а также псевдоинструкции mw (пересылка данных) и ret для возвращения из процедуры.

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 1 – результаты отладки программы в пошаговом режиме

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Адрес инстр. | (Псевдо-) инстр. | Инструк ция(и) | 16-ричный код инстр. | Содержимое регистров и ячеек памяти | |
| до вып. инстр. | после вып. инстр. |
| 0 | li s0, 18 | addi x8 x0 18 | 01700413 | x8 s0 0x00000000 | x8 s0 0x00000012 |
| 4 | li s1, 8 | addi x9 x0 8 | 00500493 | x9 s1 0x00000000 | x9 s1 0x00000008 |
| 8 | li s2, 8 | addi x18 x0 8 | 01700913 | x18 s2 0x00000000 | x18 s2 0x00000008 |
| c | li a2, x1 | addi x12 x0 10 | 00a00613 | x12 a2 0x00000000 | x12 a2 0x0000000a |
| 10 | li a3, y1 | addi x13 x0 20 | 01400693 | x13 a3 0x00000000 | x13 a3 0x00000014 |
| 14 | li a4, z1 | addi x14 x0 30 | 01e00713 | x14 a4 0x00000000 | x14 a4 0x0000001e |
| 18 | li a5, x2 | addi x15 x0 40 | 02800793 | x15 a5 0x00000000 | x15 a5 0x00000028 |
| 1c | li a6, y2 | lui x16 0x2 | 00002837 | x16 a6 0x00000000 | x16 a6 0x00002000 |
| 20 | addi x16 x16 1808 | 71080813 | x16 a6 0x00002000 | x16 a6 0x00002710 |
| 24 | li a7, z2 | addi x17 x0 60 | 03c00893 | x17 a7 0x00000000 | x17 a7 0x0000003c |
| 28 | li t5, 0 | addi x30 x0 0 | 00000f13 | x30 t5 0x00000000 | x30 t5 0x00000000 |
| 2c | li a7, 4 | addi x17 x0 4 | 00400893 | x17 a7 0x0000003c | x17 a7 0x00000004 |
| 30 | la a0, formula | auipc x10 0x10000 > | 10000517 | x10 a0 0x00000000 | x10 a0 0x10000030 |
| 34 | addi x10 x10 -36 | fdc50513 | x10 a0 0x10000030 | x10 a0 0x1000000c |
| 38 |  | ecall | 00000073 |  |  |
| 3c | la a0, val1 | auipc x10 0x10000 | 10000517 | x10 a0 0x1000000c | x10 a0 0x1000003c |
| 40 | addi x10 x10 -1 | fff50513 | x10 a0 0x1000003c | x10 a0 0x1000003b |
| 44 |  | ecall | 00000073 |  |  |
| 48 | li a7, 1 | addi x17 x0 1 | 00100893 | x17 a7 0x00000004 | x17 a7 0x00000001 |
| 4c | mv a0, a2 | addi x10 x12 0 | 00060513 | x10 a0 0x1000003b  x12 a2 0x0000000a | x10 a0 0x0000000a  x12 a2 0x0000000a |
| 50 |  | ecall | 00000073 |  |  |
| 54 | li a7, 4 | addi x17 x0 4 | 00400893 | x17 a7 0x00000001 | x17 a7 0x00000004 |
| 58 | la a0, space | auipc x10 0x10000 | 10000517 | x10 a0 0x0000000a | x10 a0 0x10000058 |
| 5c | addi x10 x10 3 | 00350513 | x10 a0 0x10000058 | x10 a0 0x1000005b |
| 60 |  | ecall | 00000073 |  |  |
| 64 | li a7, 1 | addi x17 x0 1 | 00100893 | x17 a7 0x00000004 | x17 a7 0x00000001 |
| 68 | mv a0, a3 | addi x10 x13 0 | 00068513 | x10 a0 0x1000005b  x13 a3 0x00000014 | x10 a0 0x00000014  x13 a3 0x00000014 |
| 6c |  | ecall | 00000073 |  |  |
| 70 | li a7, 4 | addi x17 x0 4 | 00400893 | x17 a7 0x00000001 | x17 a7 0x00000004 |
| 74 | la a0, space | auipc x10 0x10000 | 10000517 | x10 a0 0x00000014 | x10 a0 0x10000074 |
| 78 | addi x10 x10 -25 | fe750513 | x10 a0 0x10000074 | x10 a0 0x1000005b |
| 7c |  | ecall | 00000073 |  |  |
| 80 | li a7, 1 | addi x17 x0 1 | 00100893 | x17 a7 0x00000004 | x17 a7 0x00000001 |
| 84 | mv a0, a4 | addi x10 x14 0 | 00070513 | x10 a0 0x1000005b | x10 a0 0x0000001e |
| 88 |  | ecall | 00000073 |  |  |
| 8c | li a7, 4 | addi x17 x0 4 | 00400893 | x17 a7 0x00000001 | x17 a7 0x00000004 |
| 90 | la a0, newline | auipc x10 0x10000 | 10000517 | x10 a0 0x0000001e | x10 a0 0x10000090 |
| 94 | addi x10 x10 -144 | f7050513 | x10 a0 0x10000090 | x10 a0 0x10000000 |
| 98 |  | ecall | 00000073 |  |  |
| 9c | la a0, val2 | auipc x10 0x10000 | 10000517 | x10 a0 0x10000000 | x10 a0 0x1000009c |
| a0 | addi x10 x10 -81 | faf50513 | x10 a0 0x1000009c | x10 a0 0x1000004b |
| a4 |  | ecall | 00000073 |  |  |
| a8 | li a7, 1 | addi x17 x0 1 | 00100893 | x17 a7 0x00000004 | x17 a7 0x00000001 |
| ac | mv a0, a5 | addi x10 x15 0 | 00078513 | x10 a0 0x1000004b  x15 a5 0x00000028 | x10 a0 0x00000028  x15 a5 0x00000028 |
| b0 |  | ecall | 00000073 |  |  |
| b4 | li a7, 4 | addi x17 x0 4 | 00400893 | x17 a7 0x00000001 | x17 a7 0x00000004 |
| b8 | la a0, space | auipc x10 0x10000 | 10000517 | x10 a0 0x00000028 | x10 a0 0x100000b8 |
| bc | addi x10 x10 -93 | fa350513 | x10 a0 0x100000b8 | x10 a0 0x1000005b |
| c0 |  | ecall | 00000073 |  |  |
| c4 | li a7, 1 | addi x17 x0 1 | 00100893 | x17 a7 0x00000004 | x17 a7 0x00000001 |
| c8 | mv a0, a6 | addi x10 x16 0 | 00080513 | x10 a0 0x1000005b | x10 a0 0x00002710 |
| cc |  | ecall | 00000073 |  |  |
| d0 | li a7, 4 | addi x17 x0 4 | 00400893 | x17 a7 0x00000001 | x17 a7 0x00000004 |
| d4 | la a0, space | auipc x10 0x10000 | 10000517 | x10 a0 0x00002710 | x10 a0 0x100000d4 |
| d8 | addi x10 x10 -121 | f8750513 | x10 a0 0x100000d4 | x10 a0 0x1000005b |
| dc |  | ecall | 00000073 |  |  |
| e0 | li a7, 1 | addi x17 x0 1 | 00100893 | x17 a7 0x00000004 | x17 a7 0x00000001 |
| e4 | li a0, z2 | addi x10 x0 60 | 03c00513 | x10 a0 0x1000005b | x10 a0 0x0000003c |
| e8 |  | ecall | 00000073 |  |  |
| ec | li a7, 4 | addi x17 x0 4 | 00400893 | x17 a7 0x00000001 | x17 a7 0x00000004 |
| f0 | la a0, newline | auipc x10 0x10000 | 10000517 | x10 a0 0x0000003c | x10 a0 0x100000f0 |
| f4 | addi x10 x10 -240 | f1050513 | x10 a0 0x0000003c | x10 a0 0x00000000 |
| f8 |  | ecall | 00000073 |  |  |
| fc | li a7, z2 | addi x17 x0 60 | 03c00893 | x17 a7 0x00000004 | x17 a7 0x0000003c |
| 100 | call calculate | auipc x1 0x0 <main> | 00000097 | x1 ra 0x00000000 | x1 ra 0x00000100 |
| 104 | jalr x1 x1 96 | 060080e7 | x1 ra 0x00000100 | x1 ra 0x00000108 |
| 160 | mv t0, a2 | addi x5 x12 0 | 00060293 | x5 t0 0x00000000 | x5 t0 0x0000000a |
| 164 | mv t1, a3 | addi x6 x13 0 | 00068313 | x6 t1 0x00000000 | x6 t1 0x00000014 |
| 168 | mv t2, a4 | addi x7 x14 0 | 00070393 | x7 t2 0x00000000 | x7 t2 0x0000001e |
| 16c |  | beq x30 x0 16 <continue> | 000f0863 | x30 t5 0x00000000 | x30 t5 0x00000000 |
| 17c |  | and x28 x7 x9 | 0093fe33 | x28 t3 0x00000000 | x28 t3 0x00000008 |
| 180 | mv t4, s0 | addi x29 x8 0 | 00040e93 | x29 t4 0x00000000 | x29 t4 0x00000012 |
| 184 |  | or x30 x12 x29 | 01d66f33 | x30 t5 0x00000000 | x30 t5 0x0000001a |
| 188 |  | add x31 x28 x30 | 01ee0fb3 | x31 t6 0x00000000 | x31 t6 0x00000022 |
| 18c |  | sub x23 x6 x18 | 41230bb3 | x23 s7 0x00000000 | x23 s7 0x0000000c |
| 190 |  | or x10 x31 x23 | 017fe533 | x10 a0 0x00000000 | x10 a0 0x00000000 |
| 194 | ret | jalr x0 x1 0 | 00008067 | x1 ra 0x00000024 | x1 ra 0x00000024 |
| 108 | mv a1, a0 | addi x11 x10 0 | 00050593 | x11 a1 0x00000000  x10 a0 0x0a000000 | x11 a1 0x0a000000  x10 a0 0x0a000000 |
| 10c |  | jal x1 84 <calculate > | 054000ef | x1 ra 0x00000108 | x1 ra 0x00000110 |
| 160 | mv t0, a2 | addi x5 x12 0 | 00060293 | x5 t0 0x0000000a | x5 t0 0x0000000a |
| 164 | mv t1, a3 | addi x6 x13 0 | 00068313 | x6 t1 0x00000014 | x6 t1 0x00000014 |
| 168 | mv t2, a4 | addi x7 x14 0 | 00070393 | x7 t2 0x0000001e | x7 t2 0x0000001e |
| 16c |  | beq x30 x0 16 <continue > | 000f0863 | x30 t5 0x000001 | x30 t5 0x00000001 |
| 170 | mv t0, a5 | addi x5 x15 0 | 00078293 | x5 t0 0x0000000a | x5 t0 0x00000028 |
| 174 | mv t1, a6 | addi x6 x16 0 | 00080313 | x6 t1 0x00000014 | x6 t1 0x00002710 |
| 178 | mv t2, a7 | addi x7 x17 0 | 00088393 | x7 t2 0x0000001e | x7 t2 0x0000003c |
| 17c |  | and x28 x7 x9 | 0093fe33 | x28 t3 0x00000000 | x28 t3 0x00000008 |
| 180 | mv t4, s0 | addi x29 x8 0 | 00040e93 | x29 t4 0x00000000 | x29 t4 0x00000012 |
| 184 |  | or x30 x12 x29 | 01d66f33 | x30 t5 0x00000000 | x30 t5 0x0000001a |
| 188 |  | add x31 x28 x30 | 01ee0fb3 | x31 t6 0x00000000 | x31 t6 0x00000022 |
| 18c |  | sub x23 x6 x18 | 41230bb3 | x23 s7 0x00000000 | x23 s7 0x0000000c |
| 190 |  | or x10 x31 x23 | 017fe533 | x10 a0 0x00000000 | x10 a0 0x00000000 |
| 194 | ret | jalr x0 x1 0 | 00008067 | x1 ra 0x00000024 | x1 ra 0x00000024 |

**Тестирование**

Тестирование программы с разными параметрами представлено в таблице 2.

Таблица 2 – результаты тестирования.

|  |  |
| --- | --- |
| Входные данные | Полученные ответы |
| x1 = 0  y1 = 0  z1 = 0  x2 = 0  y2 = 0  z2 = 0 | -1 -1 |
| x1 = 10  y1 = 20  z1 = 30  x2 = 10  y2 = 20  z2 = 30 | -1 -1 |
| x1 = 123  y1 = 244  z1 = 35  x2 = 123  y2 = 244  z2 = 35 | 253 253 |
| x1 = 152  y1 = 5413  z1 = 61  x2 = 152  y2 = 5413  z2 = 61 | 5406 5406 |

**Вывод**

В ходе выполнения лабораторной работы установлена, настроена и изучена среда эмулятора Ripes. Получены знания об архитектуре RISC-V, базовом наборе инструкций, псевдо-инструкциях и регистрах. Разработана простая программа на ассемблере.

**Приложение**

Код программы lr5.s:

.data

newline: .asciz "\n"

output: .asciz "res = "

formula: .asciz "((z & 8) + (x | (-18))) | (y - 8)\n"

val1: .asciz "(x1, y1, z1) = "

val2: .asciz "(x2, y2, z2) = "

space: .asciz " "

.equ x1, 10

.equ y1, 20

.equ z1, 30

.equ x2, 123

.equ y2, 244

.equ z2, 35

.text

.globl main

main:

# Загрузка констант

li s0, 18 # s0 = a (18)

li s1, 8 # s1 = b (8)

li s2, 8 # s2 = c (8)

# Загрузка первого набора данных

li a2, x1 # a2 = x1

li a3, y1 # a3 = y1

li a4, z1 # a4 = z1

# Загрузка второго набора данных

li a5, x2 # a5 = x2

li a6, y2 # a6 = y2

li a7, z2 # a7 = z2

li t5, 0 # Установка флага для первого вызова

# Вывод формулы

li a7, 4 # Установка системного вызова для вывода строки

la a0, formula # Загрузка адреса строки в a0

ecall # Вывод строки

# Вывод первого набора

la a0, val1 # Загрузка адреса строки в a0

ecall # Вывод строки

# Вывод x1

li a7, 1 # Установка системного вызова для вывода числа

mv a0, a2 # Передача x1 в a0

ecall # Вывод x1

# Вывод пробела

li a7, 4

la a0, space # Загрузка адреса пробела в a0

ecall # Вывод пробела

# Вывод y1

li a7, 1

mv a0, a3 # Передача y1 в a0

ecall # Вывод y1

# Вывод пробела

li a7, 4

la a0, space # Загрузка адреса пробела в a0

ecall # Вывод пробела

# Вывод z1

li a7, 1

mv a0, a4 # Передача z1 в a0

ecall # Вывод z1

# Вывод enter

li a7, 4

la a0, newline # Загрузка адреса новой строки в a0

ecall # Вывод новой строки

# Вывод второго набора

la a0, val2 # Загрузка адреса строки в a0

ecall # Вывод строки

# Вывод x2

li a7, 1

mv a0, a5 # Передача x2 в a0

ecall # Вывод x2

# Вывод пробела

li a7, 4

la a0, space # Загрузка адреса пробела в a0

ecall # Вывод пробела

# Вывод y2

li a7, 1

mv a0, a6 # Передача y2 в a0

ecall # Вывод y2

# Вывод пробела

li a7, 4

la a0, space # Загрузка адреса пробела в a0

ecall # Вывод пробела

# Вывод z2

li a7, 1

li a0, z2 # Передача z2 в a0

ecall # Вывод z2

# Вывод новой строки

li a7, 4

la a0, newline # Загрузка адреса новой строки в a0

ecall

li a7, z2 # Загрузка z2 в a7

call calculate # Вызов процедуры compute

mv a1, a0 # Сохранение результата первого вызова в a1

li t5, 1

jal calculate # Вызов процедуры compute через jal

mv a2, a0 # Сохранение результата второго вызова в a2

# Вывод результатов

li a7, 4

la a0, output

ecall

li a7, 1

mv a0, a1

ecall

li a7, 4

la a0, newline

ecall

la a0, output

ecall

li a7, 1

mv a0, a2

ecall

li a7, 10 # Установка системного вызова для завершения программы

ecall # Завершение работы программы

calculate:

mv t0, a2 # Копирование x1 в t0 (t0 = x1)

mv t1, a3 # Копирование y1 в t1 (t1 = y1)

mv t2, a4 # Копирование z1 в t2 (t2 = z1)

beq t5, x0, continue # Переход, если флаг равен 0

mv t0, a5 # Копирование x2 в t0 для второго вызова (t0 = x2)

mv t1, a6 # Копирование y2 в t1 для второго вызова (t1 = y2)

mv t2, a7 # Копирование z2 в t2 для второго вызова (t2 = z2)

continue:

# Вычисление выражения

and t3, t2, s1 # t3 = z1 & b

mv t4, s0 # t4 = -a

or t5, a2, t4 # t5 = x1 | (-a)

add t6, t3, t5 # t6 = (z1 & b) + (x1 | (-a))

sub s7, t1, s2 # s7 = y1 - c

or a0, t6, s7 # a0 = ((z1 & b) + (x1 | (-a))) | (y1 - c)

ret # Возврат из процедуры