**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО 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.**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3342 |  | Смирнова Е.С. |
| Преподаватель |  | Морозов С.М. |

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.

Задачи работы:

1. Ознакомление с основными компонентами компьютера RISC-V.
2. Освоение взаимосвязи между аппаратными и программными средствами компьютера на основе архитектуры системы команд (ISA-Instruction Set Architecture).
3. Изучение языка ассемблера RISC-V и кодирования операторов ассемблера в машинном коде.
4. Освоение компиляции и выполнения ассемблерной программы в автоматическом и отладочном режимах.

**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)

В выражениях используются следующие константы:

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

Вариант 22

R = ((x >> a) & (y + b)) + (z | (-c))

Константы:

a = 18 (№ студ. билета 334224)

b = 8 (Смирнова)

c = 9 (Елизавета) Формула для вычисления:

R = ((x >> 18) & (y + 8)) + (z | (-9))

1. 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.

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.

**Выполнение работы.**

1. Процедура calc\_expression

Разработана процедура, выполняющая вычисления для двух наборов значений и возвращает два результата r1 и r2. Выражение для каждого набора входных данных вычисляется с использованием правого сдвига (sra), побитового И (and) и ИЛИ (or), а также сложения (add). а также псевдоинструкции (neg) (обратное число) и (ret) для возвращения из процедуры.

1. Основная программа

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

В разделе text содержится код программы. С помощью директивы equ определяются константы a, b, c, далее они загружаются в регистры инструкцией addi. Псевдо-инструкцией li в регистры загружаются значения для двух наборов

{x1, y1, z1} и {x2, y2, z2}, для больших чисел данная псевдо-инструкция преобразуется в две: lui (загружает константу в старшие биты регистра) и addi (складывает тот же регистр с младшими 12-ю битами). Далее с помощью псевдо-инструкции la в регистры загружаются адреса строк для печати, они выводятся с помощью системного вызова printString. Символы и числа для печати загружаются в регистр а0 после чего выводятся системными вызовами PrintChar и PrintInt соответственно. Процедура calc\_expression вызывается псевдо-инструкцией call, которая преобразуется в две инструкции: auipc и jalr. После вывода результатов, полученных в процедуре, программа завершается системным вызовом Exit.

Результаты отладки в пошаговом режиме под управлением отладчика с фиксацией содержимого используемых регистров и ячеек памяти до и после выполнения команд представлены в таблице 1. Тестирование программы на 3-х наборах для {x1 , y1 , z1 } и 3-х наборах для {x2, y2, z2 } представлены в таблице

2. Исходный код в приложении А.

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 |  | addi x8 x0 18 | 01200413 | x8 s0 0x00000000 | x8 s0 0x00000012 |
| 4 |  | addi x9 x0 8 | 00800493 | x9 s1 0x00000000 | x9 s1 0x00000008 |
| 8 |  | addi x18 x0 9 | 00900913 | x18 s2 0x00000000 | x18 s2 0x00000009 |
| c | li a2, -17 | addi x12 x0 -17 | fef00613 | x12 a2 0x00000000 | x12 a2 0xffffffef |
| 10 | li a3, 5695 | lui x13 0x1 | 000016b7 | x13 a3 0x00000000 | x13 a3 0x00001000 |
| 14 | addi x13 x13 1599 | 63f68693 | x13 a3 0x00001000 | x13 a3 0x0000163f |
| 18 | li a4, 9 | addi x14 x0 9 | 00900713 | x14 a4 0x00000000 | x14 a4 0x00000009 |
| 1c | li a5, 132 | addi x15 x0 132 | 08400793 | x15 a5 0x00000000 | x15 a5 0x00000084 |
| 20 | li a6, -11 | addi x16 x0 -11 | ff500813 | x16 a6 0x00000000 | x16 a6 0xfffffff5 |
| 24 | li a7, 310 | addi x17 x0 310 | 13600893 | x17 a7 0x00000000 | x17 a7 0x00000136 |
| 28 | mv s4, a7 | addi x20 x17 0 | 00088a13 | x20 s4 0x00000000 x17 a7 0x00000136 | x20 s4 0x00000136 x17 a7 0x00000136 |
| 2c | la a0,  formula | auipc x10 0x10000 | 10000517 | x10 a0 0x00000000 | x10 a0 0x1000002c |
| 30 | addi x10 x10 -44 | fd450513 | x10 a0 0x1000002c | x10 a0 0x10000000 |
| 34 |  | addi x17 x0 4 | 00400893 | x17 a7 0x00000136 | x17 a7 0x00000004 |
| 38 |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| 3c | mv a0, a2 | addi x10 x12 0 | 00060513 | x10 a0 0x10000000  x12 a2 0xffffffef | x10 a0 0xffffffef x12 a2 0xffffffef |
| 40 |  | addi x17 x0 1 | 00100893 | x17 a7 0x00000004 | x17 a7 0x00000001 |
| 44 |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| e0 | call calc\_expr ession | auipc x1 0x0  <\_start> | 00000097 | x1 ra 0x00000000 | x1 ra 0x000000e0 |
| e4 | Jalr x1 x1 72 | 048080e7 | x1 ra 0x000000e4 | x1 ra 0x000000e8 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 128 | neg s2, s2 | sub x18 x0 x18 | 41200933 | x18 s2 0x00000009 | x18 s2 0xfffffff7 |
| 12c |  | srl x5 x12 x8 | 008652b3 | x5 t0 0x00000000 | x5 t0 0xffffffff |
| 130 |  | add x6 x13 x9 | 00968333 | x6 t1 0x00000000  x13 a3 0x0000163f | x6 t1 0x00001647  x13 a3 0x0000163f |
| 134 |  | and x7 x5 x6 | 0062f3b3 | x7 t2 0x00000000 x5 s2 0x00003fff | x7 t2 0x00001647  x5 s2 0xffffffff |
| 138 |  | or x28 x14 x18 | 01276e33 | x28 t3 0x00000000  x14 t2 0x00000009 | x28 t3 0xffffffff  x14 t2 0x00000009 |
| 13c |  | add x11 x7 x28 | 01c385b3 | x11 a1 0x00000000  x28 t3 0xffffffff | x11 a1 0x00001646  x28 t3 0xffffffff |
| 154 | ret | jalr x0 x1 0 | 00008067 | x1 ra 0x000000e8 | x1 ra 0x000000e8 |
| 11c |  | addi x10 x0 1 | 00100513 | x10 a0 0xfffffff7 | x10 a0 0x00000001 |
| 120 |  | addi x17 x0 93 | 05d00893 | x17 a7 0x00000001 | x17 a7 0x0000005d |
| 124 |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |

Таблица 2 – результаты тестирования

|  |  |
| --- | --- |
| Входные данные | Результаты |
| x1 = -17 | Formula: ((x >> 18) & (y + 8)) + (z | (-9)) |
| y1 = 5695 | Input data: |
| z1 = 9 | {x1, y1, z1} = -17 5695 9 |
| x2 = 132 | {x2, y2, z2} = 132 -11 310 |
| y2 = -11 | Results: |
| z2 = 310 | 5702 |
|  | -9 |
| x1 = 3847 | Formula: ((x >> 18) & (y + 8)) + (z | (-9)) |
| y1 = 5 | Input data: |
| z1 = -19 | {x1, y1, z1} = 3847 5 -19 |
| x2 = 74 | {x2, y2, z2} = 74 9088 662 |
| y2 = 90888 | Results: |
| z2 = 662 | -1 |
|  | -9 |
| x1 = -8 | Formula: ((x >> 18) & (y + 8)) + (z | (-9)) |
| y1 = 47 | Input data: |
| z1 = -109 | {x1, y1, z1} = -8 47 -109 |
| x2 = -500 | {x2, y2, z2} = -500 -76 -29 |
| y2 = -76 | Results: |
| z2 = -29 | 46 |
|  | -77 |
| x1 = -13 | Formula: ((x >> 18) & (y + 8)) + (z | (-9)) |
| y1 = 40 | Input data: |

|  |  |
| --- | --- |
| z1 = -1 | {x1, y1, z1} = -13 40 -1 |
| x2 = -34 | {x2, y2, z2} = -34 7 2 |
| y2 = 7 | Results: |
| z2 = 2 | 47 |
|  | 6 |

**Выводы.**

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 А. ИСХОДНЫЙ КОД ПРОГРАММЫ.**

Название файла lb5.s:

.text

.global \_start

\_start:

# Константы

.equ a, 18

.equ b, 8

.equ c, 9

addi s0, x0, a # s0 = a

addi s1, x0, b # s1 = b

addi s2, x0, c # s2 = c

# Входные данные {x1, y1, z1}

li a2, -1 # x1

li a3, -1 # y1

li a4, -1 # z1

# Входные данные {x2, y2, z2}

li a5, 1 # x2

li a6, 1 # y2

li a7, 1 # z2

mv s4, a7

# Вывод формулы

la a0, formula

addi a7, x0, 4 # PrintString

ecall

# Вывод входных данных

mv a0, a2 # x1

addi a7, x0, 1 # PrintInt

ecall

li a0, 32

addi a7, x0, 11 # PrintChar (space)

ecall

mv a0, a3 # y1

addi a7, x0, 1

ecall

li a0, 32

addi a7, x0, 11

ecall

mv a0, a4 # z1

addi a7, x0, 1

ecall

li a0, 10

addi a7, x0, 11 # PrintChar (newline)

ecall

# Вывод второго набора данных

la a0, data

addi a7, x0, 4 # PrintString

ecall

mv a0, a5 # x2

addi a7, x0, 1 # PrintInt

ecall

li a0, 32

addi a7, x0, 11 # PrintChar (space)

ecall

mv a0, a6 # y2

addi a7, x0, 1

ecall

li a0, 32

addi a7, x0, 11

ecall

mv a7, s4 # z2

mv a0, a7

addi a7, x0, 1

ecall

li a0, 10

addi a7, x0, 11 # PrintChar (newline)

ecall

# Вызов процедуры вычислений

call calc\_expression

# Вывод результатов

la a0, results

addi a7, x0, 4 # PrintString

ecall

mv a0, a1 # r1

addi a7, x0, 1 # PrintInt

ecall

li a0, 10

addi a7, x0, 11 # PrintChar (newline)

ecall

mv a0, a2 # r2

addi a7, x0, 1 # PrintInt

ecall

addi a0, x0, 1 # завершение программы системным вызовом

addi a7, x0, 93 # Exit (a7=93) с кодом возврата в регистре а0

ecall

calc\_expression:

# Вычисление для {x1, y1, z1}

neg s2, s2 # s2 = -c

sra t0, a2, s0 # t0 = x1 >> a

add t1, a3, s1 # t1 = y1 + b

and t2, t0, t1 # t2 = (x1 >> a) & (y1 + b)

or t3, a4, s2 # t3 = z1 | c

add a1, t2, t3 # r1 = ((x1 >> a) & (y1 + b)) + (z1 | c)

# Вычисление для {x2, y2, z2}

sra t0, a5, s0 # t0 = x2 >> a

add t1, a6, s1 # t1 = y2 + b

and t2, t0, t1 # t2 = (x2 >> a) & (y2 + b)

or t3, s4, s2 # t3 = z2 | c

add a2, t2, t3 # r2 = ((x2 >> a) & (y2 + b)) + (z2 | c)

ret # возврат из процедуры

.data

formula: .asciz "Formula: ((x >> 18) & (y + 8)) + (z | -9)\nInput data:\n{x1, y1, z1} = "

data: .asciz "{x2, y2, z2} = "

results: .asciz "Results:\n"