**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра математического обеспечения и применения ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №6**

**по дисциплине «Организация ЭВМ и систем»**

**Тема: Изучение режимов адресации в ассемблере RISC-V.**

|  |  |
| --- | --- |
| Студент гр. 3344 | Тукалкин В.А. |
| Преподаватель | Фирсов М.А. |

Санкт-Петербург

2024

**Цель работы**

1. Разработка программы преобразования данных для приобретения практических навыков программирования на языке ассемблера.

2. Закрепление знаний по режимам адресации в процессоре RISC-V.

**Задание**

1. Для заданного набора констант

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

a = 18

b = 8

c = 8

сформировать массив array из 10 элементов, в котором:

arr[0] = a+b+c

array[i+1] = arr[i] + a + b - c

Доступ к массиву (инициализация, запись, чтение) должен выполняться из памяти.

2. Написать программу, которая с использованием 4 режимов адресации: регистрового, непосредственного, базового и относительного к счётчику команд реализует вычисление выражения, выбираемого из таблицы 1 в соответствии с номером студента в списке группы.

Таблица 1

|  |  |  |
| --- | --- | --- |
| Номер | Вычисляемое выражение | Используемые регистры |
| 20 | ЕСЛИ (arr[4] + arr[2] + arr[1] > threshold) ТО (res1 = arr[5] - arr[4]) ИНАЧЕ (res2 = arr[6] + c) | threshold -> t1  res1 -> s7  res2 -> a5 |

Здесь threshold – заданный порог.

**Основные теоретические положения**

Режимы адресации в ассемблере RISC-V:

1. Регистровая адресация:

При регистровой адресации регистры используются для всех операндов источников и операндов-назначений (иными словами – для всех операндов и результата). Все инструкции типа R используют именно такой режим адресации.

Пример: add rd,rs1, rs2 # rd = rs1 + rs2

2. Непосредственная адресация:

При непосредственной адресации в качестве операндов наряду с регистрами используют константы (непосредственные операнды).

Пример: addi rd,rs1,12 # rd = rs1 + 12

Чтобы использовать константы большего размера, следует использовать инструкцию непосредственной записи в старшие разряды lui (load upper immediate), за которой следует инструкция непосредственного сложения addi.

Пример: lui s2, 0xABCDE # s2 = 0xABCDE000

addi s2, s2, 0x123 # s2 = 0xABCDE123

3. Базовая адресация:

Инструкции для доступа в память, такие как загрузка слова(чтение памяти) (lw) и сохранение слова(запись в память) (sw), используют базовую адресацию. Эффективный адрес операнда в памяти вычисляется путем сложения базового адреса в регистре rs1 и 12-битного смещения с расширенным знаком, являющегося непосредственным операндом.

Пример: lw rd, 36(rs1)

Поле rs1 указывает на регистр, содержащий базовый адрес, а поле rd указывает на регистр-назначение. Поле imm, хранящее непосредственный операнд, содержит 12-битное смещение, равное 36. В результате регистр rd содержит значение из ячейки памяти rs1+36

4. Адресация относительно счётчика команд:

Инструкции условного перехода (beq, bne, blt, bge, bltu, bgeu), или ветвления, а также jal (переход и связывание) используют адресацию относительно счетчика команд для определения нового значения счетчика команд в том случае, если нужно осуществить переход. Смещение со знаком прибавляется к счетчику команд (PC) для определения нового значения PC, поэтому тот адрес, куда будет осуществлен переход, называют адресом относительно счетчика команд.

Пример: beq rs1,rs2,imm # if(rs1 == rs2) PC += imm

**Выполнение работы**

1) Процедура main

Вычисляется a+b+c и записывается в array[0], далее вычисляется a+b-c, заполняется массив, выводится строка “Array =” и все значения в массиве. Выводится “array[4]+array[2]+array[1]” и значение этой суммы, далее происходит сравнение threshold с суммой и переход. В конце выводится “result = “ и число согласно условию ЕСЛИ (arr[4] + arr[2] + arr[1] > threshold) ТО (res1 = arr[5] - arr[4]) ИНАЧЕ (res2 = arr[6] + c).

**Тестирование**

Тестирование программы с разными параметрами представлено в таблице 1.

Таблица 1 – результаты тестирования.

|  |  |
| --- | --- |
| Входные данные | Полученные ответы |
| threshold = 200 | Array: 34 52 70 88 106 124 142 160 178 196  arr[4] + arr[2] + arr[1] = 228  result = 150 |
| threshold = 182 | Array: 34 52 70 88 106 124 142 160 178 196  arr[4] + arr[2] + arr[1] = 228  result = 150 |
| threshold = 20 | Array: 34 52 70 88 106 124 142 160 178 196  arr[4] + arr[2] + arr[1] = 228  result = 18 |
| threshold = -10 | Array: 34 52 70 88 106 124 142 160 178 196  arr[4] + arr[2] + arr[1] = 228  result = 18 |

**Вывод**

В ходе выполнения лабораторной работы получены и закреплены знания по режимам адресации в ассемблере процессора RISC-V. Разработана программа преобразования данных, работающая с памятью и использующая все типы режимов адресации.

**Приложение**

Код программы lr5.s:

.data

newline: .asciz "\n" # Строка для новой строки

space: .asciz " " # Строка для пробела

str\_array: .asciz "Array: " # Строка для вывода "Array: "

str\_index: .asciz "arr[4] + arr[2] + arr[1] = " # Строка для вывода "arr[4] + arr[2] + arr[1] = "

result: .asciz "result = " # Строка для вывода "result = "

array: .word 0,0,0,0,0,0,0,0,0,0 # Массив из 10 элементов

.equ a, 18 # Константа a

.equ b, 8 # Константа b

.equ c, 8 # Константа c

.equ threshold, 1000 # Константа threshold

.text

.globl main

main:

# Определение переменных

li s0, a # Загрузка значения a в регистр s0

li s1, b # Загрузка значения b в регистр s1

li s2, c # Загрузка значения c в регистр s2

add s3, s0, s1 # Сложение s0 и s1, результат в s3 (a+b)

add s3, s3, s2 # Сложение s3 и s2, результат в s3 (a+b+c)

la t5, array # Загрузка адреса массива в регистр t5

sw s3, 0(t5) # Сохранение значения s3 в первый элемент массива

sub s3, s3, s2 # Вычитание s2 из s3, результат в s3 (a+b)

sub s3, s3, s2 # Вычитание s2 из s3, результат в s3 (a+b-c)

li s0, 1 # Инициализация счетчика цикла в регистр s0

li s1, 10 # Загрузка количества элементов массива (10) в регистр s1

init\_array:

lw t4, 0(t5) # Загрузка элемента массива в регистр t4

add t4, t4, s3 # Сложение t4 и s3, результат в t4

sw t4, 4(t5) # Сохранение значения t4 в следующий элемент массива

addi t5, t5, 4 # Увеличение адреса на 4 (размер слова)

addi s0, s0, 1 # Увеличение счетчика цикла на 1

blt s0, s1, init\_array # Если счетчик меньше 10, переход на метку init\_array

# Вывод строки "Array: "

la a0, str\_array # Загрузка адреса строки "Array: " в регистр a0

li a7, 4 # Загрузка кода системного вызова для вывода строки (4) в регистр a7

ecall # Вызов системного вызова для вывода строки

# Вывод массива

la t5, array # Загрузка адреса начала массива в регистр t5

li s0, 0 # Инициализация счетчика цикла в регистр s0

li s1, 10 # Загрузка количества элементов массива (10) в регистр s1

print\_array:

lw a0, 0(t5) # Загрузка элемента массива в регистр a0

li a7, 1 # Загрузка кода системного вызова для вывода числа (1) в регистр a7

ecall # Вызов системного вызова для вывода числа

# Вывод пробела

la a0, space # Загрузка адреса строки пробела в регистр a0

li a7, 4 # Загрузка кода системного вызова для вывода строки (4) в регистр a7

ecall # Вызов системного вызова для вывода строки

addi t5, t5, 4 # Увеличение адреса на 4 (размер слова)

addi s0, s0, 1 # Увеличение счетчика цикла на 1

blt s0, s1, print\_array # Если счетчик меньше 10, переход на метку print\_array

# Вывод новой строки

la a0, newline # Загрузка адреса строки новой строки в регистр a0

li a7, 4 # Загрузка кода системного вызова для вывода строки (4) в регистр a7

ecall # Вызов системного вызова для вывода строки

# Вычисление выражения

la t5, array # Загрузка адреса начала массива в регистр t5

lw s9, 16(t5) # Загрузка элемента массива a[4] в регистр s9

lw s1, 8(t5) # Загрузка элемента массива a[2] в регистр s1

lw s4, 4(t5) # Загрузка элемента массива a[1] в регистр s4

add s3, s9, s1 # Сложение s9 и s1, результат в s3

add s3, s3, s4 # Сложение s3 и s4, результат в s3

# Вывод строки "arr[4] + arr[2] + arr[1] = "

la a0, str\_index # Загрузка адреса строки "arr[4] + arr[2] + arr[1] = " в регистр a0

li a7, 4 # Загрузка кода системного вызова для вывода строки (4) в регистр a7

ecall # Вызов системного вызова для вывода строки

# Вывод arr[4] + arr[2] + arr[1]

mv a0, s3 # Перемещение значения суммы из регистра s3 в регистр a0

li a7, 1 # Загрузка кода системного вызова для вывода числа (1) в регистр a7

ecall # Вызов системного вызова для вывода числа

# Вывод новой строки

la a0, newline # Загрузка адреса строки новой строки в регистр a0

li a7, 4 # Загрузка кода системного вызова для вывода строки (4) в регистр a7

ecall # Вызов системного вызова для вывода строки

# Вывод строки "result = "

la a0, result # Загрузка адреса строки "result = " в регистр a0

li a7, 4 # Загрузка кода системного вызова для вывода строки (4) в регистр a7

ecall # Вызов системного вызова для вывода строки

li t1, threshold # Загрузка значения threshold в регистр t1

bgt s3, t1, if\_branch # Если сумма больше threshold, переход на метку if\_branch

else\_branch:

lw a5, 24(t5) # Загрузка элемента массива a[6] в регистр a5

add a5, a5, s2 # Сложение a5 и s2, результат в a5

# Вывод результата

mv a0, a5 # Перемещение значения результата из регистра a5 в регистр a0

li a7, 1 # Загрузка кода системного вызова для вывода числа (1) в регистр a7

ecall # Вызов системного вызова для вывода числа

j end # Переход на метку end

if\_branch:

lw s7, 20(t5) # Загрузка элемента массива a[5] в регистр s7

sub s7, s7, s9 # Вычитание s9 из s7, результат в s7

# Вывод результата

mv a0, s7 # Перемещение значения результата из регистра s7 в регистр a0

li a7, 1 # Загрузка кода системного вызова для вывода числа (1) в регистр a7

ecall # Вызов системного вызова для вывода числа

end:

# Завершение программы

li a7, 10 # Загрузка кода системного вызова для завершения программы (10) в регистр a7

ecall # Вызов системного вызова для завершения программы