**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

**отчет**

**по лабораторной работе №6**

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

**Тема: Типы адресации**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3388 |  | Лексин М.В. |
| Преподаватель |  | Молодцев Д.А. |

Санкт-Петербург

2024

## Цель работы.

Научиться программировать на языке ассемблера RISC-V и освоить различные режимы адресации этого процессора. Разработать программу для необходимых вычислений, где будут применены полученные знания.

## Задание.

Требуется написать программу, которая использует разные режимы адресации для вычислений по массиву данных в памяти. Результатом выполнения вашей программы будет измененный массив в памяти.

В качестве исходных данных дается стартовый адрес в памяти для хранения массива, количество элементов в массиве и формула для требуемых вычислений (Вычисления включают изменения каждого элемента массива в зависимости от условия). Порядок операций в формуле соответствует порядку операций в языке Си.

При автоматической проверке вашей программы исходные данные располагаются в регистрах следующим образом:

- a1 - адрес памяти, где расположен массив

- a2 - количество элементов в массиве

Считайте, что массив уже инициализирован и заполнен данными. Размер каждого элемента массива - 8 байт (dword).

Ваша программа должна иметь следующую структуру:

.globl solution

solution:

# при старте данной метки ваша программа должна выполнить

# необходимые вычисления и изменить элементы массива согласно ветке условия и формуле в ней

ret

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

Формула для вычислений будет выведена ниже (arr[i] - элемент массива, считаем что arr[-1] == 0):

ЕСЛИ ((arr[5] - arr[4] ^ arr[7]) <= 592)

ТО (arr[i] = arr[i - 1] & 17)

ИНАЧЕ (arr[i] = arr[i] & 93)

Ваш seed = 5323866464

## Выполнение работы.

Код разделен на две части: данные и логика. В секции .data описаны массив из 10 элементов, его длина, а также строки, которые выводятся перед и после обработки массива. Секция .text содержит основную программу и вспомогательные функции.

Программа сначала выводит "Исходный массив", затем печатает элементы массива через функцию print\_array. После этого вызывается функция solution, которая обрабатывает массив: для каждого элемента проверяется условие, основанное на арифметических и логических операциях. В зависимости от результата условного перехода происходит либо одно преобразование элемента, либо другое. Результаты сохраняются обратно в массив. После обработки выводится "Результат", а затем преобразованный массив.

В данной работе используются 4 разных типа адресации: регистра для работы с временными данными, непосредственной для констант, базовой для доступа к массиву и относительной для переходов в циклах.

Разработанный программный код см. в приложении А.

## Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
| 1. | -124 942 321 5424 215 52356 -1247 821 935 11230 | 4 12 65 16 85 4 0 0 0 0 | Верно |
| 2. | -45231 78654 -89012 34567 -67890 91234 -23456 56789 -12345 88901 | 81 28 76 5 76 64 0 0 0 0 | Верно |
| 3. | -45678 89234 -12345 67890 -98765 34567 -23456 78901 -56789 91234 | 16 16 69 16 17 5 1 1 1 1 | Верно |

## Выводы.

В ходе работы были изучены основные режимы адресации в RISC-V и их применение на практике. Программа успешно выполняет обработку массива, демонстрируя использование регистровой, базовой, непосредственной и относительной адресации. Результаты показали правильность работы алгоритма, подтвердив понимание работы с памятью, условными переходами и вычислениями в RISC-V.

# Приложение А Исходный код программы

.data

dataArray: .dword -124, 942, 321, 5424, 215, 52356, -1247, 821, 935, 11230

arrayLength: .dword 10

originalMsg: .string "Исходный массив: "

resultMsg: .string "Результат: "

.text

.globl Start

Start:

# Вывод строки "Исходный массив: "

la a0, originalMsg # Загрузка адреса строки originalMsg в a0

li a7, 4 # Системный вызов для печати строки

ecall

# Загрузка массива и его длины

la a1, dataArray # Загрузка адреса массива в a1

ld a2, arrayLength # Загрузка длины массива в a2

# Вывод исходного массива

jal DisplayArray

# Вызов функции решения для обработки массива

jal solution

# Вывод строки "Результат: "

la a0, resultMsg # Загрузка адреса строки resultMsg в a0

li a7, 4 # Системный вызов для печати строки

ecall

# Вывод результирующего массива

la a1, dataArray # Повторная загрузка адреса массива в a1

ld a2, arrayLength # Повторная загрузка длины массива в a2

jal DisplayArray

# Завершение программы

li a7, 10 # Системный вызов для выхода

ecall

DisplayArray:

mv t0, a1 # t0 = адрес текущего элемента массива

li t1, 0 # Инициализация счетчика элементов

PrintLoop:

beq t1, a2, PrintEnd # Если все элементы выведены, выйти из цикла

ld a0, 0(t0) # Загрузка текущего элемента массива в a0

li a7, 1 # Системный вызов для печати целого числа

ecall

li a0, 32 # Символ пробела (ASCII 32)

li a7, 11 # Системный вызов для печати символа

ecall

addi t0, t0, 8 # Переход к следующему элементу массива (8 байт)

addi t1, t1, 1 # Увеличение счетчика

j PrintLoop # Переход к следующей итерации цикла

PrintEnd:

ret # Возврат из функции DisplayArray

solution:

mv t0, a1

li t2, 0

process\_loop:

beq t2, a2, end

check\_condition:

addi t1, a1, 40

ld t3, 0(t1)

addi t1, a1, 32

ld t4, 0(t1)

addi t1, a1, 56

ld t5, 0(t1)

sub t6, t3, t4

xor t6, t6, t5

li t1, 592

bgt t6, t1, else\_part

then\_part:

beqz t2, zero\_case

ld t3, -8(t0)

li t1, 17

and t4, t3, t1

jal x0, save\_result

zero\_case:

li t4, 0

jal x0, save\_result

else\_part:

ld t3, 0(t0)

li t1, 93

and t4, t3, t1

save\_result:

sd t4, 0(t0)

addi t0, t0, 8

addi t2, t2, 1

jal x0, process\_loop

end:

ret