Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчёт по лабораторной работе**

**Дисциплина**: Низкоуровневое программирование

**Тема**: RISC-V

Вариант: 6

Выполнил студент гр. 5130901/20003 Гусев М.М.

(подпись)

Преподаватель Целищева А.А.

(подпись)

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Санкт-Петербург

2023

1. **Задание на работу.**
2. Разработать программу на языке ассемблера RISC-V, реализующую определение медианы массива чисел in-place, отладить программу в симуляторе VSim/Jupiter. Массив (массивы) данных и другие параметры (преобразуемое число, длина массива, параметр статистики и пр.) располагаются в памяти по фиксированным адресам.
3. Выделить определенную вариантом задания функциональность в подпрограмму, организованную в соответствии с ABI, разработать использующую ее тестовую программу. Адрес обрабатываемого массива данных и другие значения передавать через параметры подпрограммы в соответствии с ABI. Тестовая программа должна состоять из инициализирующего кода, кода завершения, подпрограммы main и тестируемой подпрограммы.
4. **Описание алгоритма**

На вход подаётся отсортированный по возрастанию массив целых чисел.

Проверяется длина.

ЕСЛИ длина массива нечётная, ТО она делится на 2 нацело и берётся элемент с полученным индексом, являющийся результатом работы программы.

ИНАЧЕ длина также делится нацело на два, берётся элемент с полученным индексом, берётся предшествующий ему элемент и находится их среднее арифметическое, являющееся результатом работы программы.

1. **Реализация на Python**

b = [1, 2, 3, 4, 5]

res = 0

if len(b) % 2 != 0:

res = b[len(b) // 2]

else:

res = (b[len(b) // 2 - 1] + b[len(b) // 2]) / 2

print(res**)**

1. **RISC-V программа**

.rodata

array:

.word 0, 4, 7, 12, 43, 45, 60, 100

array\_len:

.word 8

.data

int\_res:

.word 10

fract\_res:

.word 10

.text

start:

.globl start

la a0, array # считываем адрес первого элемента массива в регистр 10

lw a1, array\_len # считываем длину массива в регистр 11

li t0, 2

li t1, 4

rem a1, a1, t0 # получаем остаток от деления длины массива на 2

bnez a1, odd # если остаток 0, выполняется код далее (even), если нет выполняется код блока odd

even:

lw a1, array\_len # берём длину массива

div a1, a1, t0 # делим нацело на 2

mul a1, a1, t1 # умножаем на 4

add a2, a0, a1 # прибавляем к адресу первого элемента массива полученный индекс и запоминаем его в x12

lw a3, 0(a2) # берём элемент, адрес которого был получен на предыдущем шаге, и запоминаем его в x13

lw a4, -4(a2) # берём предыдущий элемент и запоминаем его в x14

add a4, a4, a3 # складываем элементы

div a0, a4, t0 # делим сумму на 2 и сохраняем результат в int\_res

li a1, 5 # x11 = 5, устанавливаем регистр x11 в значение дробной части результата

la a2, fract\_res # берём x12 адрес ячейки для дробной части результата

sw a1, 0(a2) # записываем в память дробную часть результата

beqz x0, exit

odd:

lw a1, array\_len # берём длину массива

div a1, a1, t0 # делим нацело на 2

mul a1, a1, t1 # умножаем на 4

add a2, a0, a1 # прибавляем к адресу первого элемента массива полученный индекс и запоминаем его в x12

lw a0, 0(a2) # берём элемент находящийся по этому адресу и сохраняем в x10

beqz x0, exit

exit:

la a2, int\_res # берём в x12 адрес ячейки для целой части результата

sw a0, 0(a2) # записываем в память целую часть результата

li a0, 17 # x10 = 17

ecall # ecall при значении x10 = 17 => останов симулятора

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

*Рис. 1 – Входные данные для программы. Секция rodata ( массив [0, 4, 7, 12, 43, 45, 60, 100], его длина (8)) с ячейки 0x10000000 до ячейки 0x10000020, далее через разделение секция data с двумя ячейками под ответ.*

*Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание*

*Рис. 2 – Выходные данные для программы. Результат для данного массива – 27,5.*

1. **RISC-V подпрограмма**

Файл setup.s:

.text

start:

.globl start

call main

finish:

mv a1, a0 # a1 = a0

li a0, 17 # a0 = 17

ecall

Файл main.s:

.text

main:

.globl main

addi sp, sp, -16 # выделение памяти в стеке

sw ra, 12(sp) # сохранение ra

la a0, array # }

lw a1, array\_len # } median(array, array\_length)

call median # }

# результат выполнения программы лежит в регистрах a0 и a1

la a2, int\_res # берём адрес для целой части результата

sw a0, 0(a2) # записываем целую часть в память

la a2, fract\_res # берём адрес для дробной части результата

sw a1, 0(a2) # записываем дробную часть в память

lw ra, 12(sp) # восстановление ra

addi sp, sp, 16 # освобождение памяти в стеке

li a0, 0

ret # jalr zero, ra, 0

.rodata

array:

.word 0, 1, 2, 3, 4, 5, 6, 7, 8

array\_len:

.word 9

.data

int\_res:

.word 10

fract\_res:

.word 10

Файл median.s:

.text

median:

.globl median

# в a0 - адрес 0-го элемента массива

# в a1 - длина массива

li t0, 2 # в t0 помещаем 2

li t1, 4 # в t1 помещаем 4

rem a2, a1, t0 # получаем остаток от деления длины массива на 2

bnez a2, odd # если остаток 0, выполняется код далее (even), если нет выполняется код блока odd

even:

div a2, a1, t0 # делим нацело на 2

mul a2, a2, t1 # умножаем на 4

add a2, a0, a2 # прибавляем к адресу первого элемента массива полученный индекс и запоминаем его в x12

lw a3, 0(a2) # берём элемент, адрес которого был получен на предыдущем шаге, и запоминаем его в x13

lw a4, -4(a2) # берём предыдущий элемент и запоминаем его в x14

add a3, a4, a3 # складываем элементы

div a0, a3, t0 # делим сумму нацело на 2 и сохраняем результат в x10

li a1, 5 # x11 = 5, устанавливаем регистр x11 в значение дробной части результат

beqz x0, exit # if x0 = 0 (always true) - exit

odd:

div a2, a1, t0 # делим нацело на 2

mul a2, a2, t1 # умножаем на 4

add a2, a0, a2 # прибавляем к адресу первого элемента массива полученный индекс и запоминаем его в x12

lw a0, 0(a2) # берём элемент находящийся по этому адресу и сохраняем в x10

li a1, 0 # т.к. результат целый, подпрограмма должна возвращать в регистре для дробной части 0

beqz x0, exit # if x0 = 0 (always true) - exit

*Изображение выглядит как текст, снимок экрана, меню

Автоматически созданное описание* exit:

ret # jalr zero, ra, 0

*Рис. 3 – Входные данные для тестовой программы. Секция rodata (массив [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], его длина (10)) с ячейки 0x10000000 до ячейки 0x10000028, далее через разделение секция data с двумя ячейками под ответ.*

*Изображение выглядит как текст, снимок экрана

Автоматически созданное описаниеРис. 4 – Выходные данные для тестовой программы. Результат для данного массива – 4,5.*

1. **Вывод**

В ходе выполнения работы была разработана программа на языке ассемблера RISC-V для определения медианы массива целых чисел и выполнена её отладка в симуляторе V-Sim. Далее написанная программа была выделена в подпрограмму, организованную в соответствие с ABI. Разработана вызывающая её тестовая программа. Результаты полностью соответствуют ожидаемым.