Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Лабораторная работа № 3**

Программирование Risc-V

по дисциплине «Низкоуровневое программирование»

Выполнил

студент гр. 3530901/90004

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Антунович П.Ю.

(подпись)

Руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Алексюк А.О.

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Санкт-Петербург   
2021

**Постановка задачи**

1. Разработать программу на языке ассемблера RISC-V, реализующую определенную вариантом задания функциональность, 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.

2. 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.

**Вариант задания:** 13) расчет значения многочлена по схеме Горнера с «длинным» результатом

**Общие сведения**

Схема Горнера для деления многочлена — это алгоритм упрощения вычисления значения многочлена f(x) при определённой величине x = x0 методом деления многочлена на одночлены (многочлены 1ой степени). Каждый одночлен включает в себя максимум один процесс умножения и один процесс сложения. Результат, полученный из одного одночлена, прибавляют к результату, полученному от следующего одночлена и так далее в аккумулятивной манере. Такой процесс деления также называют синтетическим делением.

Т.е. если задан многочлен 4-й степени:

То его можно представить в форме:

Где x = x0 – значение, при котором мы ищем значение многочлена.

Наглядно алгоритм можно представить в виде таблицы:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| счетчик | 4 | 3 | 2 | 1 | 0 |
| вычисление |  |  |  |  |  |
| результат |  |  |  |  |  |

Значение b0 и является искомым.

**Ход работы**

Напишем программу, реализующую метод Горнера с «длинным» результатом. Тестировать код будем на примере многочлена:

Код программы с комментариями представлен ниже:

.rodata

array\_length:

.word 5 # кол-во коэффициентов

x:

.word 2 # x0 - значение x, по которому ищется значение многочлена

msg1:

.string "f(x) = " # строка для вывода на консоль

array:

.word 1, 3, 5, 7, 9 # массив коэффициентов начиная со старшего

.text

start:

.globl start

la a3, array\_length

lw a3, 0(a3)

la a7, x

lw a7, 0(a7)

li a2, 1 # счетчик итераций

la a4, array

lw a4, 0(a4) # загрузка первого элемента массива

la a5, array

loop:

slli a6, a2, 2 # a6 = a2 << 2 = a2 \* 4

add a6, a5, a6 # a6 = a5 + a6 = a5 + a2 \* 4

lw a6, 0(a6)

mul a4, a4, a7 # a4 = a4 \* a7

add a4, a4, a6 # a4 = a4 + a6

addi a2, a2, 1

bltu a2, a3, loop # if (a2 < a3) goto loop

finish:

li a0, 4

la a1, msg1

ecall # вывод текста на консоль

li a0, 1

mv a1, a4

ecall # вывод результата на консоль

li a0, 10

ecall # конец программы

Код для задачи 2 практически идентичен коду для 1 задачи. Рабочий код был выведен в подпрограмму, разработана тестовая программа, вызывающая подпрограмму main, которая запускает тестируемую подпрограмму.

Исходное значение ra следует сохранять перед псевдоинструкцией call, и восстанавливать перед псевдоинструкцией ret. Это необходимо для избегания зацикливания программы, т.к иначе значение ra изменяется в результате выполнения псевдоинструкции call: в ra будет записан адрес возврата для вызываемой подпрограммы trapz, то есть адрес следующей за call псевдоинструкции, в данном случае - инструкции возврата из подпрограммы main. Таким образом, результатом выполнения инструкции возврата, соответствующе псевдоинструкции ret, будет переход на эту же инструкцию.

Файл setup.s:

.text

start:

.globl start

call main

finish:

mv a1, a0 # a1 = a0

li a0, 17 # a0 = 17

ecall

Файл main.s:

.rodata

array\_length:

.word 5 # кол-во коэффициентов

x:

.word 2 # x0 - значение x, по которому ищется значение многочлена

msg1:

.string "f(x) = " # строка для вывода на консоль

array:

.word 1, 3, 5, 7, 9 # массив коэффициентов начиная со старшего

.text

main:

.globl main

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

sw ra, 12(sp) # запись ra в стек

la a3, array\_length

la a7, x

la a4, array

la a5, array

call gorner

li a0, 4

la a1, msg1

ecall # вывод текста на консоль

li a0, 1

mv a1, a4

ecall # вывод результата на консоль

lw ra, 12(sp) # восстановление ra

addi sp, sp, 16 # освобождение стека

ret

Файл gorner.s:

.text

gorner:

.global gorner

lw a3, 0(a3)

lw a7, 0(a7)

li a2, 1 # счетчик итераций

lw a4, 0(a4) # загрузка первого элемента массива

loop:

slli a6, a2, 2 # a6 = a2 << 2 = a2 \* 4

add a6, a5, a6 # a6 = a5 + a6 = a5 + a2 \* 4

lw a6, 0(a6)

mul a4, a4, a7 # a4 = a4 \* a7

add a4, a4, a6 # a4 = a4 + a6

addi a2, a2, 1

bltu a2, a3, loop # if (a2 < a3) goto loop

ret

**Вывод:** в ходе работы были разработаны программа расчета значений многочлена по схеме Горнера с «длинным» результатом для ассемблера Risc-V, а также разработана и протестирована подпрограмма с данным функционалом.