Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Лабораторная работа № 2**

Программирование RISC-V

по дисциплине «Низкоуровневое программирование»

Выполнил

студент гр. 3530901/90004

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Остапчук А.С.

(подпись)

Руководитель

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

(подпись)

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

Санкт-Петербург   
2021

**Задача**

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**

Вариант: 9 - Расчет биномиальных коэффициентов для данного показателя по треугольнику Паскаля.

**Выполнение работы**

Массив – под меткой «array»

Показатель – под меткой «param»

Изначально мы имеем массив из 2 элементов, равных 1. Суть алгоритма заключается в том, чтобы проходить по массиву, складывать значения двух соседних элементов и перезаписывать их и увеличивать массив в размерах. По массиву мы проходим столько раз, сколько необходимо для вычисления коэффициентов для заданного показателя.

Код программы для задачи 1 с комментариями:  
  
.text

start:

.globl start

la a3, param #}

lw a3, 0(a3) #} a3 = <параметр>

la a4, array # a4 = <адрес 0-го элемента массива>

li a2, 1 # a2 = 0

loop1:

bgeu a2, a3, loop\_exit1 # if( a2 >= a3 ) goto loop\_exit

addi a2, a2, 1

li a7, 0 # a7 = 0

slli a5, a7, 2 # a5 = a2 << 2 = a2 \* 4

add a5, a4, a5 # a5 = a4 + a5 = a4 + a2 \* 4

addi a6, a5, 4 # a6 = a5 + 4

lw t1, 0(a6) # t1 = array[i+1]

lw t0, 0(a5) # t0 = array[i]

add t0, t1, t0 # t0 = array[0] + array[1]

loop2:

bgeu a7, a2, loop\_exit2 # if( a2 >= a3 ) goto loop\_exit

addi a7, a7, 1 # a7 += 1

slli a5, a7, 2 # a5 = a7 << 2 = a7 \* 4

add a5, a4, a5 # a5 = a4 + a5 = a4 + a2 \* 4

addi a6, a5, 4 # a6 = a5 + 4

lw t1, 0(a5) # t1 = array[i]

sw t0, 0(a5) # array[i] = t0

lw t0, 0(a6) # t0 = array[i+1]

add t0, t1, t0 # t0 = array[i+1] + t1

jal zero, loop2 # goto loop

loop\_exit2:

jal zero, loop1 # goto loop

loop\_exit1:

finish:

li a0, 10 # x10 = 10

li a1, 0 # x11 = 0

ecall # ecall при значении x10 = 10 => останов симулятора

.rodata

param:

.word 5

.data

array:

.word 1, 1

Аналогичный алгоритм используется для задачи 2 и в целом часть подпрограмма базируется на изменённой версии кода для задачи 1. Так же разработана вызывающая ее тестовую программа. Адрес обрабатываемого массива данных и показатель передаются через регистр a0 и a1 соответственно.  
  
Код программы задачи 2 состоит из трёх файлов ниже представлены эти файлы с комментариями:

Файл 1

# setup.s

.text

start:

.globl start

call main

finish:

mv a1, a0 # a1 = a0

li a0, 17 # a0 = 17

ecall # выход с кодом завершения

Файл2  
# 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, param # } find\_koeffs( array, param );

call find\_koeffs # }

li a0, 0 # }

lw ra, 12(sp) # восстановление ra

addi sp, sp, 16 # освобождение памяти в стеке

ret # } return 0;

.rodata

param:

.word 5

.data

array:

.word 1, 1

Файл3

# find\_koeffs

.text

find\_koeffs:

.globl find\_koeffs

li a2, 1 # a2 = 0

loop1:

bgeu a2, a1, loop\_exit1 # if( a2 >= a1 ) goto loop\_exit

addi a2, a2, 1

li a7, 0 # a7 = 0

slli a5, a7, 2 # a5 = a2 << 2 = a2 \* 4

add a5, a0, a5 # a5 = a0 + a5 = a4 + a2 \* 4

addi a6, a5, 4 # a6 = a5 + 4

lw t1, 0(a6) # t1 = array[i+1]

lw t0, 0(a5) # t0 = array[i]

add t0, t1, t0 # t0 = array[0] + array[1]

loop2:

bgeu a7, a2, loop\_exit2 # if( a2 >= a3 ) goto loop\_exit

addi a7, a7, 1 # a7 += 1

slli a5, a7, 2 # a5 = a7 << 2 = a7 \* 4

add a5, a0, a5 # a5 = a0 + a5 = a4 + a2 \* 4

addi a6, a5, 4 # a6 = a5 + 4

lw t1, 0(a5) # t1 = array[i]

sw t0, 0(a5) # array[i] = t0

lw t0, 0(a6) # t0 = array[i+1]

add t0, t1, t0 # t0 = array[i+1] + t1

jal zero, loop2 # goto loop

loop\_exit2:

jal zero, loop1 # goto loop

loop\_exit1:

ret

**Вывод**

В ходе выполнения лабораторной работы была разработана программа на языке ассемблера RISC-V, выполняющая расчет биномиальных коэффициентов для данного показателя по треугольнику Паскаля и выполняющая запуск как цельной программы, так и подпрограммы, организованную в соответствии с ABI, разработана использующая её тестовая программа.