Санкт-Петербургский политехнический университет Петра Великого Институт компьютерных наук и технологий Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчёт по лабораторной работе №3**

Дисциплина: Низкоуровневое программирование

Тема: Программирование RISC-V

Вариант: 2

Выполнил студент группы 3530901/00002 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Г. Антонов

(подпись)

Принял преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.С. Степанов

(подпись)

«\_\_» \_\_\_\_\_\_\_\_\_\_ 2021г.

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.

**Задание**

Формирование в памяти десятичного представления дробного числа (-1;1).

**Выполнение работы**

Для реализации было принято решение сохранять число в его ieee754 представлении в 2/10 коде. Знак и экспонента не нуждаются в переводе в 2/10 код, т.к занимают в памяти =< 1 байта. А для перевода мантиссы был выбран алгоритм, представленный на рис. 1.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 1

Код для части 1 задания находится в приложении 1, код для части 2 задания находится в приложении 2.

**Описание работы**

**Приложение 1.**

.rodata

input: .float 3.1415

.data

sign: .word 0

eps: .word 0

mant: .word 0

mant210: .zero 32

.text

.globl \_\_start

\_\_start:

la a1, input

lw a0, 0(a1)

if\_sign: bgez a0, end\_if\_sign # запись знака

la a1, sign

li t0, 1

sw t0, 0(a1)

end\_if\_sign:

srli t0, a0, 23

li t1, 0xFF

and t0, t0, t1

la a1, eps

sw t0, 0(a1) # записываем эпсилон

li t1, 0x7FFFFF

and t0, a0, t1

la a1, mant

sw t0, 0(a1) # записываем мантиссу

while\_mantissa\_nez: beqz t0, finish

li t2, 2

rem a0, t0, t2

srli t0, t0, 1

addi sp, sp, -4

sw t0, 0(sp)

la t0, mant210

la t1, mant210

addi t0, t0, 96

lb t2, 0(t0)

add t2, t2, a0

mv a0, t2

mv a1, zero

update: beq t0, t1, end\_update # цикл обновления результата

addi sp, sp, -12

sw t0, 0(sp)

sw t1, 4(sp)

sw ra, 8(sp)

li t2, 2

rem t1, a0, t2

srli a0, a0, 1

shift\_if: beqz a1, end\_shift\_if

addi a0, a0, 5

end\_shift\_if:

mv a1, t1

lw t0, 0(sp)

lw t1, 4(sp)

lw ra, 8(sp)

addi sp, sp, 12

sw a0, 0(t0)

addi t0, t0, -4

lb a0, 0(t0)

j update

end\_update:

lw t0, 0(sp)

addi sp, sp, 4

j while\_mantissa\_nez

finish:

li a0, 1

lw a1, sign

ecall # печатаем знак

li a0, 1

lw a1, eps

ecall # печатаем эпсилон

li a0, 10

ecall # завершаем программу

**Приложение 2.**

# файл 1(riscv.s)

.text

.globl \_\_start

\_\_start:

call main

finish:

li a0, 10

ecall

# файл 2(main.s)

.rodata

input: .float 3.1415

.data

sign: .word 0

eps: .word 0

mant: .word 0

mant210: .zero 32

.text

.globl main

main:

la a1, input

lw a0, 0(a1)

if\_sign: bgez a0, end\_if\_sign # запись знака

la a1, sign

li t0, 1

sw t0, 0(a1)

end\_if\_sign:

srli t0, a0, 23

li t1, 0xFF

and t0, t0, t1

la a1, eps

sw t0, 0(a1) # записываем эпсилон

li t1, 0x7FFFFF

and t0, a0, t1

la a1, mant

sw t0, 0(a1) # записываем мантиссу

la a5, mant210

addi sp, sp, -16

sw ra, 12(sp)

call subf

lw ra, 12(sp)

addi sp, sp, 16

end:

li a0, 1

lw a1, sign

ecall

li a0, 1

lw a1, eps

ecall

ret

# файл 3(subf)

.text

.globl subf

subf:

while\_mantissa\_nez: beqz t0, end

li t2, 2

rem a0, t0, t2

srli t0, t0, 1

addi sp, sp, -4

sw t0, 0(sp)

mv t0, a5

mv t1, a5

addi t0, t0, 96

lb t2, 0(t0)

add t2, t2, a0

mv a0, t2

mv a1, zero

update: beq t0, t1, end\_update # цикл обновления результата

addi sp, sp, -12

sw t0, 0(sp)

sw t1, 4(sp)

sw ra, 8(sp)

li t2, 2

rem t1, a0, t2

srli a0, a0, 1

shift\_if: beqz a1, end\_shift\_if

addi a0, a0, 5

end\_shift\_if:

mv a1, t1

lw t0, 0(sp)

lw t1, 4(sp)

lw ra, 8(sp)

addi sp, sp, 12

sw a0, 0(t0)

addi t0, t0, -4

lb a0, 0(t0)

j update

end\_update:

lw t0, 0(sp)

addi sp, sp, 4

j while\_mantissa\_nez

end:

ret

**Вывод**

Программирование на языке ассемблера с набором инструкций risc-v влечёт за собой некоторые трудности (например, невозможность указания необходимых аргументов для подпрограмм, что усложняет поиск ошибок). Однако программы для данного набора инструкций являются намного более читаемыми, чем программы для EDSAC. Ещё важнее наличие возможности перехода по адресам меток, что позволяет не переписывать половину программы после незначительных изменений в самом её начале (чем запомнилось программирование для EDSAC)