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:** «Создание программ для RISC-V»

**Вариант №12**

Выполнил студент гр. 3530901/90004 \_\_\_\_\_\_\_\_\_\_ Сидоров А.А.

*(подпись)*

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

*(подпись)*

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Санкт-Петербург

2021

1. Постановка задачи

1. Разработать программу на языке ассемблера RISC-V, реализующую, формирование последовательности чисел в коде Грея заданной разрядности, отладить программу в симуляторе Jupiter. Массив данных и другие параметры (адрес массива, длина массива) 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.

2. Реализация программы

**Листинг 2.1.** Программа код Грея.

.text  
start:  
.globl start  
 la a1, array\_length #}  
 lw a1, 0(a1) #} a3 = <длина массива>  
 la a0, array # a4 = <адрес 0-го элемента массива>  
 li a2, 1 # a2 = 1  
 jal zero, loop\_check1 # goto loop\_check  
loop1:  
 #{  
 slli a5, a2, 2 # a5 = a2 << 2 = a2 \* 4  
 add a5, a0, a5 # a5 = a4 + a5 = a4 + a2 \* 4 (array[i])  
 addi a6, a5, -4 # a6 = a5 + (-4) = a5 - 4 (array[i - 1])  
 lw a6, 0(a6)   
 addi a6, a6, 1 # a6 += 1  
 sw a6, 0(a5) # array[i] = a6  
 addi a2, a2, 1 # a2 += 1  
 #}  
loop\_check1:  
 bltu a2, a1, loop1 # if( a2 < a3 ) goto loop  
loop\_exit1:  
li a2, 1 # a2 = 1  
jal zero, loop\_check # goto loop\_check  
loop:  
#{  
 slli a5, a2, 2 # a5 = a2 << 2 = a2 \* 4  
 add a5, a0, a5 # a5 = a4 + a5 = a4 + a2 \* 4  
 lw t1, 0(a5) # t1 = array[i]  
 #получение кода грея  
 srli t3, t1, 1 # t3 = t1 >> 1  
 xor t1, t1, t3  
 sw t1, 0(a5) # array[i] = t1  
 #  
 addi a2, a2, 1 # a2 += 1  
#}  
loop\_check:  
 bltu a2, a1, loop # if( a2 < a3 ) goto loop  
loop\_exit:  
# запись массива в регистры  
lw s2, 0(a0)  
lw s3, 4(a0)  
lw s4, 8(a0)  
lw s5, 12(a0)  
lw s6, 16(a0)  
lw s7, 20(a0)  
lw s8, 24(a0)  
lw s9, 28(a0)  
#  
finish:  
 li a0, 10 # x10 = 10  
 li a1, 0 # x11 = 0  
 ecall # ecall при значении x10 = 10 => останов симулятора  
.rodata  
array\_length:  
 .word 8 # разрядность 3(2^3)  
.data  
array:  
 .word 0, 0, 0, 0, 0, 0, 0, 0

3. Реализация подпрограммы

**Листинг 3.1.** Сортировка обменом в виде подпрограммы.

.text  
main:  
.globl main  
la a1, array\_length #}  
 lw a1, 0(a1) #} a3 = <длина массива>  
 la a0, array # a4 = <адрес 0-го элемента массива>  
 li a2, 1 # a2 = 1  
call gray  
li a0, 0  
ret  
  
.globl gray  
gray:  
jal zero, loop\_check1 # goto loop\_check  
loop1:  
 #{  
 slli a5, a2, 2 # a5 = a2 << 2 = a2 \* 4  
 add a5, a0, a5 # a5 = a4 + a5 = a4 + a2 \* 4 (array[i])  
 addi a6, a5, -4 # a6 = a5 + (-4) = a5 - 4 (array[i - 1])  
 lw a6, 0(a6)   
 addi a6, a6, 1 # a6 += 1  
 sw a6, 0(a5) # array[i] = a6  
 addi a2, a2, 1 # a2 += 1  
 #}  
loop\_check1:  
 bltu a2, a1, loop1 # if( a2 < a3 ) goto loop  
loop\_exit1:  
li a2, 1 # a2 = 1  
jal zero, loop\_check # goto loop\_check  
loop:  
#{  
 slli a5, a2, 2 # a5 = a2 << 2 = a2 \* 4  
 add a5, a0, a5 # a5 = a4 + a5 = a4 + a2 \* 4  
 lw t1, 0(a5) # t1 = array[i]  
 #получение кода грея  
 srli t3, t1, 1 # t3 = t1 >> 1  
 xor t1, t1, t3  
 sw t1, 0(a5) # array[i] = t1  
 #  
 addi a2, a2, 1 # a2 += 1  
#}  
loop\_check:  
 bltu a2, a1, loop # if( a2 < a3 ) goto loop  
loop\_exit:  
# запись массива в регистры  
lw s2, 0(a0)  
lw s3, 4(a0)  
lw s4, 8(a0)  
lw s5, 12(a0)  
lw s6, 16(a0)  
lw s7, 20(a0)  
lw s8, 24(a0)  
lw s9, 28(a0)  
#  
  
ret  
  
.rodata  
array\_length:  
 .word 8 # разрядность 3(2^3)  
.data  
array:  
 .word 0, 0, 0, 0, 0, 0, 0, 0

4. Руководство программиста

Подпрограмма gray вызывается согласно соглашениям ABI. В регистре ra ожидается адрес возврата. Аргументами подпрограммы являются регистры a0, a1, где а0 – адрес исходного массива, а1 – количество элементов исходного массива.

Вывод

В ходе работы была создана программа, формирующая последовательности чисел в коде Грея заданной разрядности для языка ассемблера RISC-V. Также эта программа была представлена, как подпрограмма, из-за чего её можно использовать несколько раз в других программах.