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. 3530901/90002 _____ Г.А. Сухоруков
(подпись)

Преподаватель _____ Д.С. Степанов
(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) #} a1 = <длина массива>
 la a0, array # a0 = <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 = a0 + a5 = a0 + a2 * 4 (array[i])
 sw a2, 0(a5) # array[i] = a6
 addi a2, a2, 1 # a2 += 1
 #}
loop_check1:
 bltu a2, a1, loop1 # if( a2 < a1 ) goto loop1
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 = a0 + a5 = a0 + 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 < a1 ) 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
```

Используется алгоритм генерации кода Грея через операцию XOR. Коды Грея легко получаются из двоичных чисел путём побитовой операции «Исключающее ИЛИ» с тем же числом, сдвинутым вправо на один бит и в котором старший разряд заполняется нулём. В результате получается массив из десятичных чисел в коде Грея, что подтверждают тесты:

0x000100ac	00	00	00	04
0x000100a8	00	00	00	05
0x000100a4	00	00	00	07
0x000100a0	00	00	00	06
0x0001009c	00	00	00	02
0x00010098	00	00	00	03
0x00010094	00	00	00	01
0x00010090	00	00	00	00
0x0001008c	00	00	00	08

Тестирование программы.

Код Грея разрядности 3.

3. Реализация подпрограммы

Листинг 3.1. Подпрограмма код Грея.

Программа setup вызывает тестовую программу main, а та в свою очередь вызывает gray

```
.text
__start:
.globl __start
 call main

 finish:
  li a0, 10
  ecall
```

Setup.

```
.text
main:
.globl main
la a1, array_length #}
 lw a1, 0(a1) #} a1 = <длина массива>
 la a0, array # a0 = <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 = a0 + a5 = a0 + a2 * 4 (array[i])
 sw a2, 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
```

```
27 loop:
28 #{
29  slli a5, a2, 2 # a5 = a2 << 2 = a2 * 4
30  add a5, a0, a5 # a5 = a4 + a5 = a4 + a2 * 4
31  lw t1, 0(a5) # t1 = array[i]
32  #получение кода грея
33  srli t3, t1, 1 # t3 = t1 >> 1
34  xor t1, t1, t3
35  sw t1, 0(a5) # array[i] = t1
36  #
37  addi a2, a2, 1 # a2 += 1
38 #}
39 loop_check:
40  bltu a2, a1, loop # if( a2 < a3 ) goto loop
41 loop_exit:
42 li a0, 10 # x10 = 10
43  li a1, 0 # x11 = 0
44  ecall # ecall при значении x10 = 10 => останов симулятора
45
46 .rodata
47 array_length:
48  .word 8 # разрядность n(2^n)
49 .data
50 array:
51  .word 0, 0, 0, 0, 0, 0, 0, 0
```

Main и Gray.

4. Руководство программиста

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