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. 3530901/00002

_____ Тарсуков Ф.Д.
(подпись)

Руководитель

_____ Степанов Д.С.
(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: 12 – Вывод последовательности в коде Грея заданной разрядности.

Выполнение работы

Пользователем задается число разрядов. Алгоритм по заданному числу строит счетчик итераций и, контролируя их число, переводит последовательно числа из двоичного кода в код Грея. Вывод двоичного числа реализован при помощи маски и логического умножения, позволяющих выводить число по разрядам.

Код программы для задачи 1 с комментариями:

```
.text
__start:
.globl __start

lw a3, size   #a3 = числу разрядов

li a2, 0      #a2 = 0 - изменяемое число
li a4, 1      #a4 = 1 - число шагов
li a5, 0      #a5 = 0 - счетчик

bgeu a3, a4, end_of_the_check
   jal zero, finish    #переход к завершению, если число разрядов меньше 1
end_of_the_check:

sll a4, a4, a3    #a4 умножаем на 2 на каждый разряд

loop:
   bgeu a5, a4, finish    #счетчик итераций
   srli a6, a2, 1    #вводим вспомогательную переменную для получения числа в коде Грэя
```

```
    xor a6, a2, a6    #число в коде Грея
    print_code:
        srli a7, a4, 1    #создаем маску
        addi s4, a3, -1    #корректор для вывода
        digit:
            beq a7, zero, new_line    #счетчик выводимых разрядов
            and s2, a6, a7    #логическое умножение маски на число в коде Грея
            srl s2, s2, s4    #сдвиг для корректного вывода
            li a0, 1
            addi a1, s2, 0    #вывод разряда
            ecall
            srli a7, a7, 1    #сдвиг маски
            addi s4, s4, -1    #изменение величины корректора
            jal zero, digit    #безусловный возврат для вывода следующего разряда
    new_line:
        li a0, 4
        la a1, newline    #сдвиг на новую строку
        ecall
    addi a5, a5, 1    #увеличиваем счетчик
    addi a2, a2, 1    #увеличиваем изменяемое число
    jal zero, loop    #безусловный возврат в начало цикла

finish:
    li a0, 10
    ecall

.data
size:
    .word 7    #число разрядов
newline:
    .string "\n"    #константа новой строки
```

Аналогичный алгоритм используется для задачи 2 и в целом часть подпрограммы базируется на изменённой версии кода для задачи 1. Так же разработана вызывающая ее тестовую программа. Число разрядов передается через регистр s5.

Код программы задачи 2 состоит из трёх файлов ниже представлены эти файлы с комментариями:

Файл 1
```
.text
__start:
.globl __start

  call main

finish:
  li a0, 10
  ecall
```

Файл2
```
.text
main:
.globl main

addi sp, sp, -16    #add sp -= 16
sw ra, 12(sp)    #сохранение ссылки на возврат в прошлую подпрограмму в стек

lw s5, size
call graycode

lw ra, 12(sp)    #возврат ссылки на прошлую вызывающую подрограмму
addi sp, sp, 16    #(add) sp += 16  (возвращение sp в норму)
ret

.data
size:
    .word 7    #число разрядов
newline:
    .string "\n"    #константа новой строки
```

Файл3
```
.text
graycode:
.globl graycode

mv a3, s5    #a3 = числу разрядов

li a2, 0       #a2 = 0 - изменяемое число
li a4, 1       #a4 = 1 - число шагов
li a5, 0       #a5 = 0 - счетчик

bgeu a3, a4, end_of_the_check
    jal zero, finish    #переход к завершению, если число разрядов меньше 1
```

```
end_of_the_check:

sll a4, a4, a3    #a4 умножаем на 2 на каждый разряд

loop:
   bgeu a5, a4, finish    #счетчик итераций
   srli a6, a2, 1    #вводим вспомогательную переменную для получения числа в коде Грэя
   xor a6, a2, a6    #число в коде Грея
   print_code:
      srli a7, a4, 1    #создаем маску
      addi s4, a3, -1    #корректор для вывода
      digit:
         beq a7, zero, new_line    #счетчик выводимых разрядов
         and s2, a6, a7    #логическое умножение маски на число в коде Грея
         srl s2, s2, s4    #сдвиг для корректного вывода
         li a0, 1
         addi a1, s2, 0    #вывод разряда
         ecall
         srli a7, a7, 1    #сдвиг маски
         addi s4, s4, -1    #изменение величины корректора
         jal zero, digit    #безусловный возврат для вывода следующего разряда
   new_line:
      li a0, 4
      la a1, newline    #сдвиг на новую строку
      ecall
   addi a5, a5, 1    #увеличиваем счетчик
   addi a2, a2, 1    #увеличиваем изменяемое число
   jal zero, loop    #безусловный возврат в начало цикла

finish:
   ret
```

Вывод

В ходе выполнения лабораторной работы была разработана программа на языке ассемблера RISC-V, выполняющая вывод последовательности чисел в коде Грея и выполняющая запуск как цельной программы, так и подпрограммы, организованную в соответствии с ABI.