Санкт-Петербургский политехнический университет Петра Великого
Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе №3

Дисциплина: Основы вычислительной техники

Тема: Программирование RISC-V

Выполнил студент гр. 3530901/10005 _____ Бикир И. И.
(подпись)

Преподаватель _____ Коренев Д.А
(подпись)

“___”_____ 2022 г.

Санкт-Петербург

2022

1. ТЗ

В массиве чисел сделать перестановку – вначале все числа с нечетными индексами, потом – все с четными.

2. Метод решения

Для решения этой задачи нужно пройтись по нечетным индексам и передвинуть их в начало массива. Для этого нужно 2 цикла. В первом цикле будет i – указатель на нечетные индексы (с каждым проходом цикла увеличиваться на 2). Во втором цикле нужна j и k. j изначально равно i и с каждым проходом цикла уменьшается на 1, пока > k. k увеличивается на 1 каждый раз, когда i увеличивается на 2. Во вложенном цикле меняются элементы с индексами [j] и [j-1]. На примере для массива 0, 1, 2, 3. i = 1, k=0, j=1. Меняются a[1]-a[0]. Получается 1, 0, 2, 3. После этого i=3, j=3, k=1. И происходит две перестановки – a[3]-[2], a[2]-a[1]. Массив становится 1, 3, 0, 2. После этого i становится больше чем длина массива и происходит выход из цикла и завершение программы. В качестве i используется a2, j = a7, k = t0

3. Реализация программы

```
.text
__start:
.globl __start
la a3, array_length #}
lw a3, 0(a3) #} a3 = <длина массива>
la a4, array # a4 = <адрес 0-го элемента массива>
li t0, 1 # k(t0) = 1
li a2, 1 # a2 = 2
loop1:
  bgeu a2, a3, loop1_exit # if (a2 >= a3) goto loop1_exit
  mv a7, a2
    loop2:
    bltu a7, t0, loop2_exit # if (a7 < t0) goto loop2_exit
    slli a5, a7, 2 # a5 = a7 << 2 = a7 * 4
    add a5, a4, a5 # a5 = a4 + a5 = a4 + a2 * 4
    lw t1, -4(a5) # t1 = array[i-1]
    lw t2, 0(a5) # t2 = array[i]
    sw t1, 0(a5) # array[i] = t1
    sw t2, -4(a5) # array[i-1] = 2
    addi a7, a7, -1
    jal zero, loop2
  loop2_exit:
  addi a2, a2, 2 # a2 += 2
  addi t0, t0, 1 # t0 += 1
  jal zero, loop1
loop1_exit:
finish:
 li a0, 10 # x10 = 10
 li a1, 0 # x11 = 0
 ecall # ecall при значении x10 = 10 => останов. симулятора
 .rodata
 array_length:
 .word 12
.data
array:
 .word 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
```

4. Работа с подпрограммой

```
# setup.s
.text
start:
.globl start
 call main
finish:
 mv a1, a0 # a1 = a0
 li a0, 17 # a0 = 17
 ecall # выход с кодом завершения
```

```
# 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, array_length # } swap_pairs(array, array_length);
 call swap_pairs # }
 li a0, 0 # }
 lw ra, 12(sp) # } восстановление ra
 addi sp, sp, 16 # освобождение памяти в стеке
 ret # } return 0;
.rodata
array_length:
 .word 11
.data
array:
 .word 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

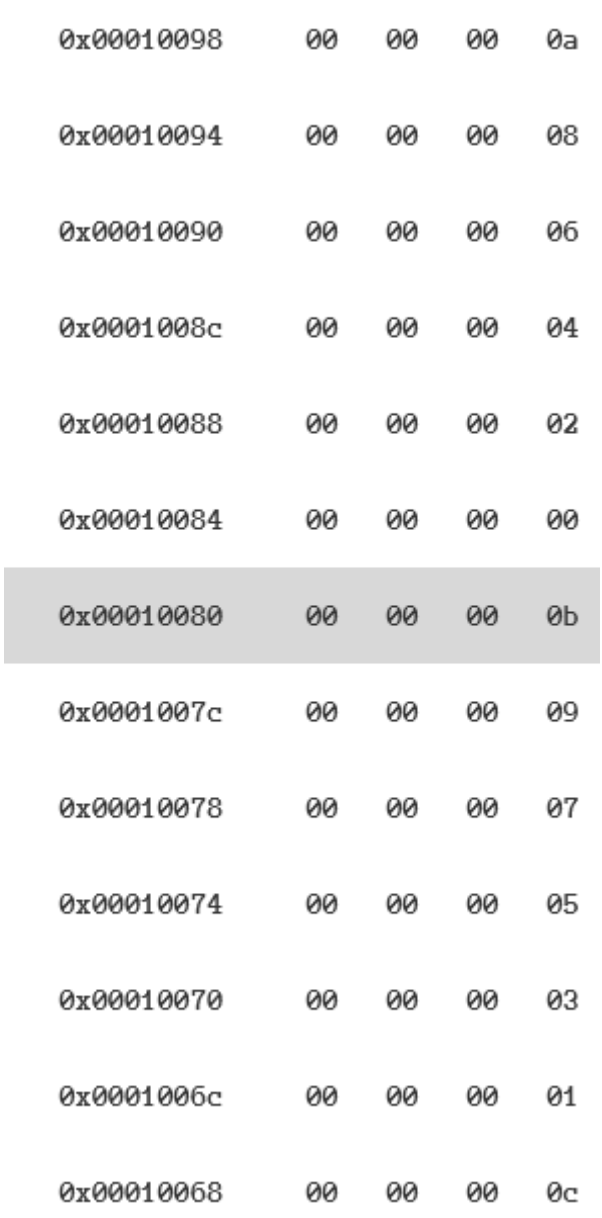
```
#swap_pairs
.text
swap_pairs:
.globl swap_pairs
 # в a0 - адрес 0-го элемента массива чисел типа unsigned
 # в a1 - длина массива
 li t0, 1 # k(t0) = 1
 li a2, 1 # a2 - 2
loop1:
  bgeu a2, a1, loop1_exit #if(a2 >= a3) goto loop1_exit
  mv a7, a2
  loop2:
    bltu a7, t0, loop2_exit #if (a7 < t0) goto loop2_exit
    slli a5, a7, 2 # a5 = a7 << 2 = a7 * 4
    add a5, a0, a5 # a5 = a4 + a5 = a4 + a2 * 4
    lw t1, -4(a5) # t1 = array[i-1]
    lw t2, 0(a5) # t2 = array[i]
    sw t1, 0(a5) # array [i] = t1
    sw t2, -4(a5) # array[i-1] = t2
    addi a7, a7, -1
    jal zero, loop2
  loop2_exit:
  addi a2, a2, 2 # a2 += 2
  addi t0, t0, 1 # t0 += 1
  jal zero, loop1
loop1_exit:
 ret
```

5. Руководство программисту

Начальные данные к программе: адрес нулевого элемента массива (и соответственно сам массив) и его длина. В реализации без подпрограммы адрес и длина хранятся в регистрах a4 и a3соответственно. В реализации через подпрограмму предполагается, что нулевым аргументом (регистр a0) передается адрес нулевого элемента массива и первым аргументов (регистр a1) – длина массива.

6. Испытание программ

Без подпрограммы:



Адрес 0x00010068 – длина массива – далее следуют элементы массива

С подпрограммой:

```
0x000100bc     00   00   00   0a

0x000100b8     00   00   00   08

0x000100b4     00   00   00   06

0x000100b0     00   00   00   04

0x000100ac     00   00   00   02

0x000100a8     00   00   00   00

0x000100a4     00   00   00   09

0x000100a0     00   00   00   07

0x0001009c     00   00   00   05

0x00010098     00   00   00   03

0x00010094     00   00   00   01

0x00010090     00   00   00   0b
```

Адрес 0x00010090 – длина массива – далее следуют элементы массива