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: Архитектура RISC-V

Вариант: 18

Выполнил студент гр. 3530901/90002 _____Н.С. Якубец
(подпись)

Принял преподаватель _____ Д.С. Степанов
(подпись)

“_____” _____ 2021 г.

Санкт-Петербург
2021

Задание

Разработать программу на языке ассемблера RISC-V, реализующую заданную функциональность, а также выделить реализованный алгоритм в подпрограмму, создать тестовую программу, вызывающую подпрограмму.

Вариант 18: Циклический сдвиг массива чисел.

Алгоритм

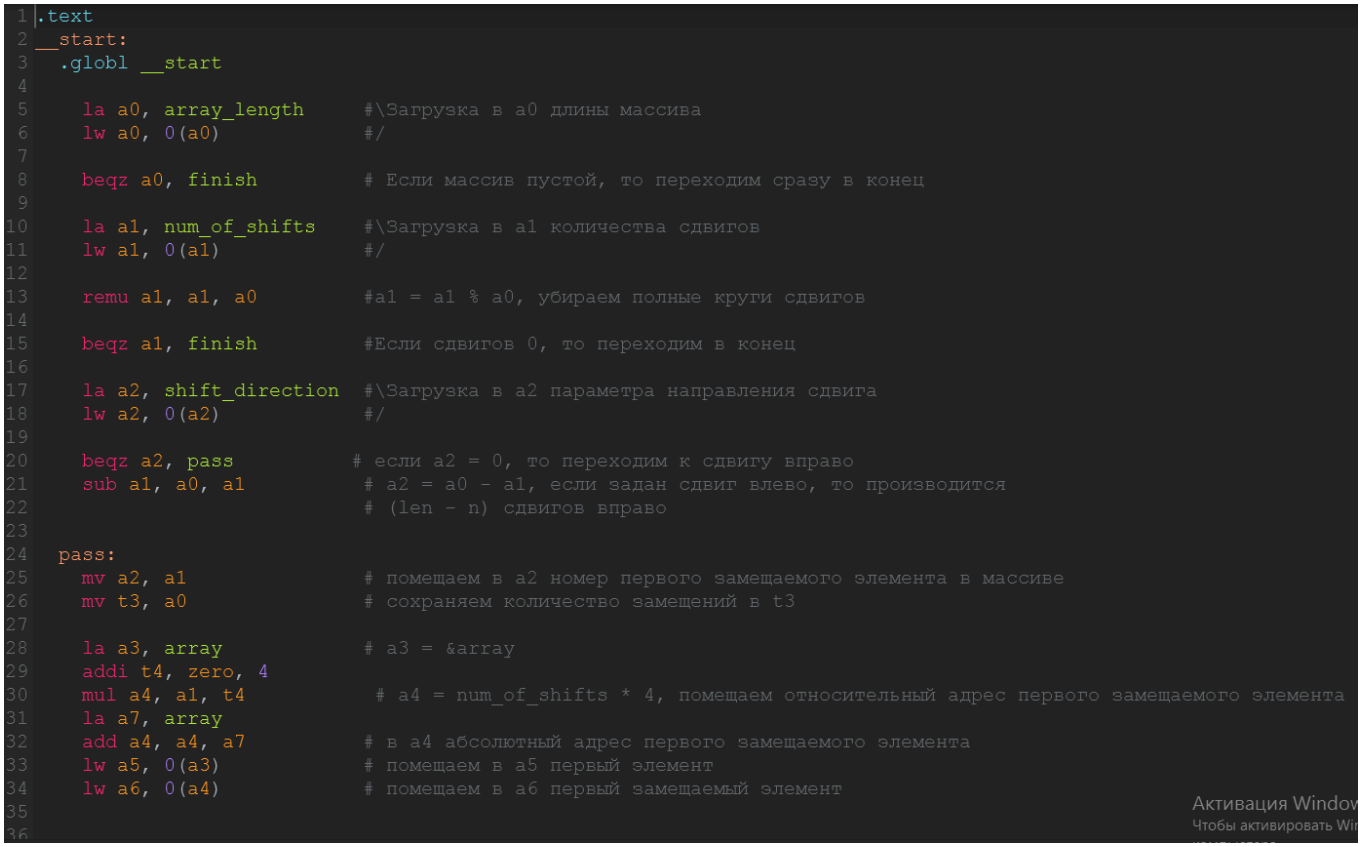
На каждом шаге своей работы алгоритм замещает элемент el1 на el2. Адрес el2 на каждом шаге считается следующим образом:

el2_addr = (el1_addr + num_of_shifts) % array_length

После замещения в массиве el2 на el1, el2 помещается в регистр, хранящий el1, и, вычисляя адрес нового el2, программа помещает новый el2 в соответствующий регистр. Такие шаги производятся array_length раз, так как каждый элемент изменил свое положение в результате сдвигов и должен быть заменен в исходном массиве.

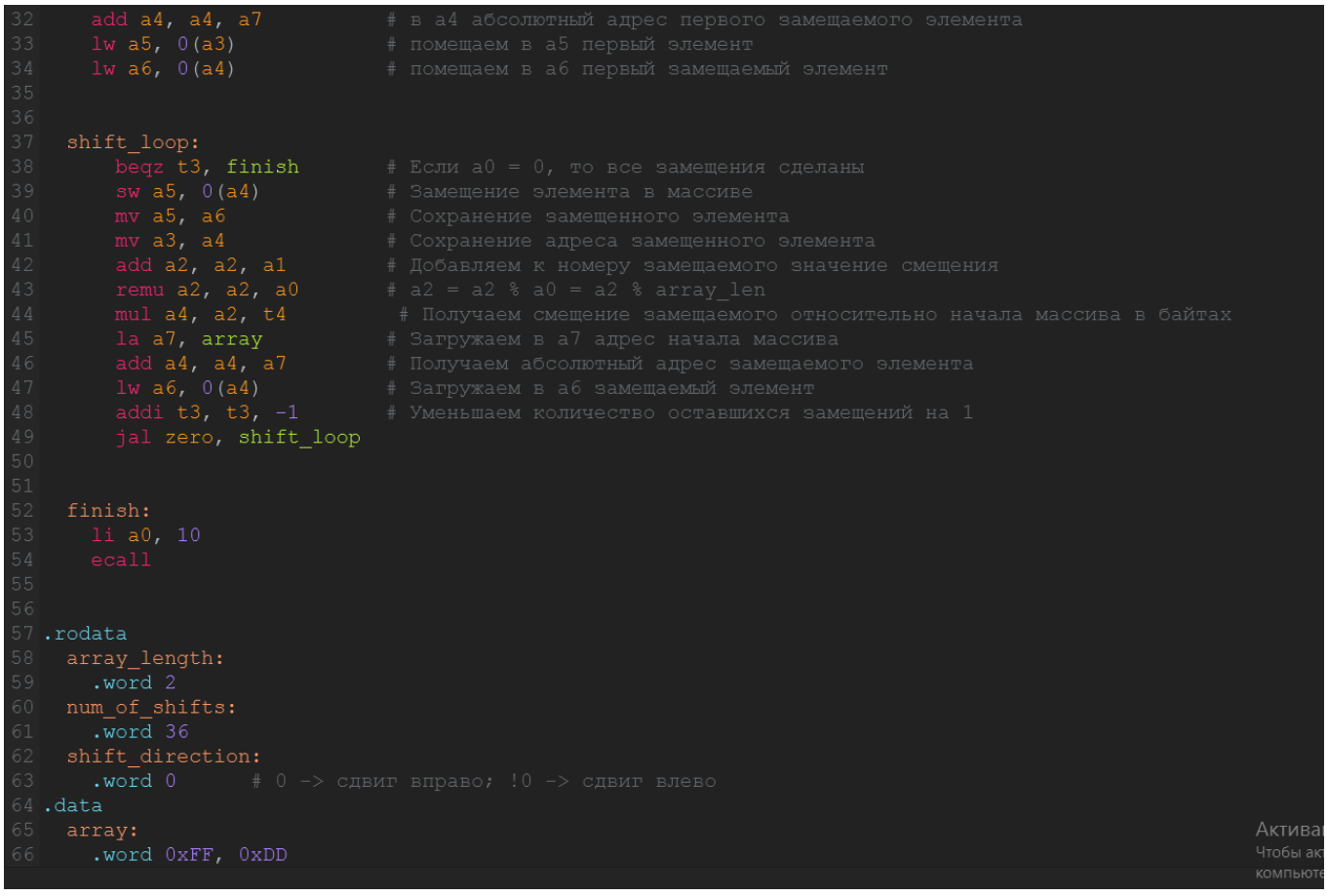
Программа

Была реализована программа prog в файле prog.s, циклически сдвигающая указанный массив указанное число раз в указанном направлении. Код программы приводится на рис. 1-2.



```
.text
__start:
  .globl __start

    la a0, array_length     #\Загрузка в a0 длины массива
    lw a0, 0(a0)            #/

    beqz a0, finish         # Если массив пустой, то переходим сразу в конец

    la a1, num_of_shifts    #\Загрузка в a1 количества сдвигов
    lw a1, 0(a1)            #/

    remu a1, a1, a0         #a1 = a1 % a0, убираем полные круги сдвигов

    beqz a1, finish         #Если сдвигов 0, то переходим в конец

    la a2, shift_direction  #\Загрузка в a2 параметра направления сдвига
    lw a2, 0(a2)            #/

    beqz a2, pass          # если a2 = 0, то переходим к сдвигу вправо
    sub a1, a0, a1          # a2 = a0 - a1, если задан сдвиг влево, то производится
                            # (len - n) сдвигов вправо

  pass:
    mv a2, a1               # помещаем в a2 номер первого замещаемого элемента в массиве
    mv t3, a0               # сохраняем количество замещений в t3

    la a3, array            # a3 = &array
    addi t4, zero, 4
    mul a4, a1, t4           # a4 = num_of_shifts * 4, помещаем относительный адрес первого замещаемого элемента
    la a7, array
    add a4, a4, a7          # в a4 абсолютный адрес первого замещаемого элемента
    lw a5, 0(a3)            # помещаем в a5 первый элемент
    lw a6, 0(a4)            # помещаем в a6 первый замещаемый элемент
```

Рис. 1 Исходный код программы часть 1



```
    add a4, a4, a7          # в a4 абсолютный адрес первого замещаемого элемента
    lw a5, 0(a3)            # помещаем в a5 первый элемент
    lw a6, 0(a4)            # помещаем в a6 первый замещаемый элемент


  shift_loop:
      beqz t3, finish       # Если a0 = 0, то все замещения сделаны
      sw a5, 0(a4)          # Замещение элемента в массиве
      mv a5, a6             # Сохранение замещенного элемента
      mv a3, a4             # Сохранение адреса замещенного элемента
      add a2, a2, a1        # Добавляем к номеру замещаемого значение смещения
      remu a2, a2, a0       # a2 = a2 % a0 = a2 % array_len
      mul a4, a2, t4         # Получаем смещение замещаемого относительно начала массива в байтах
      la a7, array          # Загружаем в a7 адрес начала массива
      add a4, a4, a7        # Получаем абсолютный адрес замещаемого элемента
      lw a6, 0(a4)          # Загружаем в a6 замещаемый элемент
      addi t3, t3, -1       # Уменьшаем количество оставшихся замещений на 1
      jal zero, shift_loop


  finish:
    li a0, 10
    ecall


.rodata
  array_length:
    .word 2
  num_of_shifts:
    .word 36
  shift_direction:
    .word 0       # 0 -> сдвиг вправо; !0 -> сдвиг влево
.data
  array:
    .word 0xFF, 0xDD
```

Рис. 2 Исходный код программы часть 2

Таблица 1 Назначение используемых регистров

Регистр	Назначение
a0	array_length, хранение значения длины массива
a1	num_of_shifts, хранение числа сдвигов
a2	direction, хранение числа, определяющего направление сдвига, а позже хранение номера замещаемого элемента
a3	addr_to_move, хранение адреса элемента, который заместит следующий замещаемый элемент
a4	addr_to_save, хранение адреса замещаемого элемента
a5	elem_to_move, хранение элемента, который заместит следующий замещаемый элемент
a6	elem_to_move, хранение замещаемого элемента
a7	Используется как переменная для хранения адреса array
t3	Хранит количество замещений, которые осталось сделать. Изначально t3 = array_length
t4	Хранит число 4 для операций умножения для получения смещения элементов в байтах относительно начала массива

Подпрограмма

Была реализована подпрограмма subprog в файле subprog.s, циклически сдвигающая указанный массив указанное число раз в указанном направлении, вызывающая ее подпрограмма main в файле main.s и главная программа start в файле start.s, вызывающей. Код всех файлов приводится на рис. 3-.

```
.text
subprog:
  .globl subprog

    beqz a0, return         # Если массив пустой, то переходим сразу в конец

    remu a1, a1, a0         #a1 = a1 % a0, убираем полные круги сдвигов

    beqz a1, return         #Если сдвигов 0, то переходим в конец

    mv t5, a3

    beqz a2, pass           # если a2 = 0, то переходим к сдвигу вправо
    sub a1, a0, a1          # a2 = a0 - a1, если задан сдвиг влево, то производится
                            # (len - n) сдвигов вправо

  pass:
    mv a2, a1               # помещаем в a2 номер первого замещаемого элемента в массиве
    mv t3, a0               # сохраняем количество замещений в t3

    mv a3, t5               # a3 = &array
    addi t4, zero, 4
    mul a4, a1, t4          # a4 = num_of_shifts * 4, помещаем относительный адрес первого замещаемого элемента
    mv a7, t5
    add a4, a4, a7          # в a4 абсолютный адрес первого замещаемого элемента
    lw a5, 0(a3)            # помещаем в a5 первый элемент
    lw a6, 0(a4)            # помещаем в a6 первый замещаемый элемент


  shift_loop:
      beqz t3, return       # Если a0 = 0, то все замещения сделаны
      sw a5, 0(a4)          # Замещение элемента в массиве
      mv a5, a6             # Сохранение замещенного элемента
      mv a3, a4             # Сохранение адреса замещенного элемента
```

Рис. 3 Код подпрограммы subprog, часть 1

```
28
29
30   shift_loop:
31       beqz t3, return        # Если a0 = 0, то все замещения сделаны
32       sw a5, 0(a4)           # Замещение элемента в массиве
33       mv a5, a6              # Сохранение замещенного элемента
34       mv a3, a4              # Сохранение адреса замещенного элемента
35       add a2, a2, a1         # Добавляем к номеру замещаемого значение смещения
36       remu a2, a2, a0        # a2 = a2 % a0 = a2 % array_len
37       mul a4, a2, t4         # Получаем смещение замещаемого относительно начала массива в байтах
38       #mv a7, t5              # Загружаем в a7 адрес начала массива
39       add a4, a4, a7         # Получаем абсолютный адрес замещаемого элемента
40       lw a6, 0(a4)           # Загружаем в a6 замещаемый элемент
41       addi t3, t3, -1        # Уменьшаем количество оставшихся замещений на 1
42       jal zero, shift_loop
43
44
45   return:
46       ret
47
48
49
50
```

Рис. 4 Код подпрограммы subprog, часть 2

```
.text
__main:
  .globl __main

    addi sp, sp, -16
    sw ra, 12(sp)

    la a0, array_length     #\Загрузка в a0 длины массива
    lw a0, 0(a0)            #/

    la a1, num_of_shifts    #\Загрузка в a1 количества сдвигов
    lw a1, 0(a1)            #/

    la a2, shift_direction  #\Загрузка в a2 параметра направления сдвига
    lw a2, 0(a2)            #/

    la a3, array            #Загрузка в a3 указателя на массив

    call subprog

    lw ra, 12(sp)
    addi sp, sp, 16

    ret


.rodata
  array_length:
    .word 11
  num_of_shifts:
    .word 37
  shift_direction:
    .word 0       # 0 -> сдвиг вправо; !0 -> сдвиг влево
.data
  array:
```

Рис. 5 Код подпрограммы main, часть 1

```
10
11      la a1, num_of_shifts    #\Загрузка в a1 количества сдвигов
12      lw a1, 0(a1)            #/
13
14      la a2, shift_direction  #\Загрузка в a2 параметра направления сдвига
15      lw a2, 0(a2)            #/
16
17      la a3, array            #Загрузка в a3 указателя на массив
18
19      call subprog
20
21      lw ra, 12(sp)
22      addi sp, sp, 16
23
24      ret
25
26
27  .rodata
28    array_length:
29      .word 11
30    num_of_shifts:
31      .word 37
32    shift_direction:
33      .word 0       # 0 -> сдвиг вправо; !0 -> сдвиг влево
34  .data
35    array:
36  #       .word 0xFF, 0xDD
37      .word 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

Рис. 6 Код подпрограммы main, часть 2

```
1 .text
2 __start:
3   .globl __start
4
5     call __main
6
7     li a0, 10
8     ecall
```

Рис. 7 Код программы start

Программа main вызывает подпрограмму subprog, которая использует те же самые регистры что и программа prog из первой части отчета, за исключением регистра t5, который подпрограмма использует для хранения в нем адреса массива array.

Таблица 2 Использование регистров при передаче параметров

Регистр	Назначение
a0	array_length, передача длины массива
a1	num_of_shifts, передача числа сдвигов
a2	direction, передача числа, определяющего направление сдвига
a3	array, передача адреса массива