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: Программирование RISC-V
Вариант 7

Выполнил студент гр. 3530901/90002 _____Д.С. Ковалевский
(подпись)

Принял старший преподаватель _____Д.С. Степанов
(подпись)

“___” _____ 2021 г.

Санкт-Петербург
2021

Цели работы:

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 7: Определение k-й порядковой статистики in-place.

1.Постановка задачи

Необходимо смоделировать программу для RISC-V, которая определит такой элемент неупорядоченного массива, если бы он был k-ым в упорядоченном.

Для реализации будем использовать сортировку пузырьком и вывод k-ого (k от 0 до n) элемента уже отсортированного массива.

2.Реализация программы.

С помощью необходимого набора инструкций, составляем программу, для поиска k-ой порядковой статистики(Рис.1 и 2):

```
.text
start:
.globl start
 la a0, array_size # Берем ссылку на размер массива
 lw a0, 0(a0) # Запоминаем значение по ссылке
 la a1, array # Адрес первого элемента

 addi a4, a1, 0 # адрес первой ячейки массива
 addi a5, a1, 4 # адрес второй ячейки массива

next_loop:
 li a2, 1 # iter = 1
 li a3, 0 # bool = 0

next_step:
 lw t0, 0(a4) # Запоминаем i-ое значение
 lw t1, 0(a5) # Запоминаем значение i+1
 bgeu t1, t0, no_change # Если i+1 >= i, то не меняем их местами
 sw t0, 0(a5) # Записываем i в ячейку i + 1
 sw t1, 0(a4) # Записываем i+1 в ячейку i
 li a3, 1 # bool = 1

no_change:
 addi a4, a4, 4 # Прибавляем 4 к адресу i, переход к следующей ячейке
 addi a5, a5, 4 # Прибавляем 4 к адресу i+1, переход к следующей ячейке
 addi a2, a2, 1 # iter += 1
 bne a2, a0, next_step # Если iter != array_size, то идем на слудующий шаг

 addi a4, a1, 0 # Сброс в начальное значение
 addi a5, a1, 4 # Сброс в начальное значение
 bnez a3, next_loop # Если bool != 0, то переход к следующей итерации

 la a6, k # получаем ссылку на k
 lw a6, 0(a6) # получаем значение k
 slli a6, a6, 2 # полчаем 4k
 add a4, a4, a6 # ищем нужное k-ое число
 lw a4, 0(a4) # получаем его значение
 slli a0, a0, 2 # размер массива * 4
 add a1, a1, a0 # адрес первого эл + array_size * 4, получаем адрес ячейки после массива
 sw a4, 0(a1) # записываем в нее ответ
```

Рис.1

```
41
42  loop_exit:
43  finish:
44   li a0, 10
45   li a1, 0
46   ecall # останов симулятора
47  .rodata # read-only data
48  array_size:
49   .word 7 # Размер массива
50  k:
51   .word 3 # Число k
52  .data #изменяемые данные
53  array:
54   .word 5, 3, 10, 10, 11, 1, 8  # Массив данных
```

Рис.2

```
>>> memory 0x1000000c 3
             Value (+0) Value (+4) Value (+8) Value (+c)
[0x1000000c] 0x00000005 0x00000003 0x0000000a 0x0000000a
[0x1000001c] 0x0000000b 0x00000001 0x00000008 0x00000000
[0x1000002c] 0x00000000 0x00000000 0x00000000 0x00000000
```

Рис.3. Массив данных до начала работы

```
>>> memory 0x1000000c 3
             Value (+0) Value (+4) Value (+8) Value (+c)
[0x1000000c] 0x00000001 0x00000003 0x00000005 0x00000008
[0x1000001c] 0x0000000a 0x0000000a 0x0000000b 0x00000008
[0x1000002c] 0x00000000 0x00000000 0x00000000 0x00000000
```

Рис.4. Массив данных после работы и ответ.

Для расширения массива необходимо ввести еще данные в 54 строку и в 49 изменить размер массива на текущий:

```
>>> memory 0x1000000c 3
             Value (+0) Value (+4) Value (+8) Value (+c)
[0x1000000c] 0x00000005 0x00000003 0x0000000a 0x0000000a
[0x1000001c] 0x0000000b 0x00000001 0x00000008 0x00000005
[0x1000002c] 0x00000002 0x00000000 0x00000000 0x00000000
>>> breakpoint 0x00010084
>>> c
>>> memory 0x1000000c 3
             Value (+0) Value (+4) Value (+8) Value (+c)
[0x1000000c] 0x00000001 0x00000002 0x00000003 0x00000005
[0x1000001c] 0x00000005 0x00000008 0x0000000a 0x0000000a
[0x1000002c] 0x0000000b 0x00000005 0x00000000 0x00000000
```

Рис.5. Расширение массива.

Как видим по рис.5, расширение массива не нарушило работу программы, в конце так же правильный ответ (k = 3, k от 0 до n).

3.Реализация подпрограммы.

Для реализации подпрограммы необходимо написать тестирующую программу, а также подпрограмму main, уже вызывающую программу прошлого пункта.

```
# test.s
.text
start:
.globl start
 call main # Вызываем  main
finish:
 li a0, 10 # x10 = 10
 ecall # Если значение x10 == 10, то ecall вызывает останов
```

Рис.6. Тестирующая программа.

Тестирующая программа вызовет подпрограмму main:

```
# main.s
.text
main:
.globl main
 la a0, array_size # Берем ссылку на размер массива
 lw a0, 0(a0) # Запоминаем значение по ссылке
 la a1, array # Адрес первого элемента
 li a6, 3

 addi a4, a1, 0 # Адрес первой ячейки массива
 addi a5, a1, 4 # Адрес второй ячейки массива

 addi sp, sp, -16 # Выделение памяти в стеке
 sw ra, 12(sp) # Записываем ra (адрес возврата)

 call second # Вызываем подпрограмму

 lw ra, 12(sp) # Восстанавливаем ra
 addi sp, sp, 16 # Всвобождение памяти в стеке

 li a0, 0
 ret

 .rodata # read-only data
array_size:
 .word 7
k:
 .word 3
.data # Изменяемые данные
array:
 .word 5, 3, 10, 10, 11, 1, 8
```

Рис.7. Подпрограмма main.

main в свою очередь вызовет подпрограмму second:

```
# second.s
.text
second:
.globl second

next_loop:
 li a2, 1 # iter = 1
 li a3, 0 # bool = 0

next_step:
 lw t0, 0(a4) # Запоминаем i-ое значение
 lw t1, 0(a5) # Запоминаем значение i+1
 bgeu t1, t0, no_change # Если i+1 >= i, то не меняем их местами
 sw t0, 0(a5) # Записываем i в ячейку i + 1
 sw t1, 0(a4) # Записываем i+1 в ячейку i
 li a3, 1 # bool = 1

no_change:
 addi a4, a4, 4 # Прибавляем 4 к адресу i, переход к следующей ячейке
 addi a5, a5, 4 # Прибавляем 4 к адресу i+1, переход к следующей ячейке
 addi a2, a2, 1 # iter += 1
 bne a2, a0, next_step # Если iter != array_size, то идем на слудующий шаг

 addi a4, a1, 0 # Сброс в начальное значение
 addi a5, a1, 4 # Сброс в начальное значение
 bnez a3, next_loop # Если bool != 0, то переход к следующей итерации

 slli a6, a6, 2 # полчаем 4k
 add a4, a4, a6 # ищем нужное к-ое число
 lw a4, 0(a4) # получаем его значение
 slli a0, a0, 2 # размер массива * 4
 add a1, a1, a0 # адрес первого эл + array_size * 4, получаем адрес ячейки после массива
 sw a4, 0(a1) # записываем в нее ответ

loop_exit:
 ret
```

Рис.8. Подпрограмма second.

Как видно из рисунка 7 и 8, начальные данные преобразования можно вынести в подпрограмму main.

Результаты работы:

```
>>> memory 0x1000000c 3
            Value (+0) Value (+4) Value (+8) Value (+c)
[0x1000000c] 0x00000005 0x00000003 0x0000000a 0x0000000a
[0x1000001c] 0x0000000b 0x00000001 0x00000008 0x00000000
[0x1000002c] 0x00000000 0x00000000 0x00000000 0x00000000
>>> breakpoint 0x000100ac
>>> c
>>> memory 0x1000000c 3
            Value (+0) Value (+4) Value (+8) Value (+c)
[0x1000000c] 0x00000001 0x00000003 0x00000005 0x00000008
[0x1000001c] 0x0000000a 0x0000000a 0x0000000b 0x00000008
[0x1000002c] 0x00000000 0x00000000 0x00000000 0x00000000
```

Рис.9. Результат работы, входные и выходные данные.

Как видим по рис.9, массив от отсортирован, а в конце массива правильный ответ (k = 3, k от 0 до n).

4. Алгоритм работы определения k-ой статистики

1 шаг. Запоминаем необходимые адреса и значения (размер массива, адрес первой и второй ячейки массива)

2 шаг. Устанавливаем iter = 1, bool = 0.

3 шаг. Сравниваем i и i+1 число, если второе больше или равно первому, то 5 шаг.

4 шаг. Меняем i и i+1 местами в массиве, bool = 1.

5 шаг. i = i+1, i+1=i+2 (продвигаемся дальше по массиву)

6 шаг. iter +=1, если iter != array_size, то шаг 3

7 шаг. Возвращаем i и i+1 адреса 1 и 2 элемента массива

8 шаг. Если bool !=0, то шаг 2.

9 шаг. Вывод результата в следующую после массива ячейку памяти.

5.Вывод

Составленные программа и подпрограмма для RISC-V удовлетворяют варианту и находят k-ую порядковую статистику in-place.