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 6

Выполнил студент гр. 3530901/90002 _____С.А. Колупаев
(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 6: Нахождение медианы in-place.

1.Постановка задачи

Необходимо смоделировать программу для RISC-V, которая найдет медиану неотсортированного массива(длина массива может быть как четной, так и нечетной).

Для реализации будем проходиться по всему массиву, сравнивая эл-ты друг с другом, записывая результаты сравнения в счётчики, исходя из показаний счётчика будем определять медиану.

2.Реализация программы.

С помощью необходимого набора инструкций составляем программу, для нахождения медианы (Рис.1):

```
.text
__start:
.global __start
  li a4, 0 #i = 0
  li a5, 0 #j = 0
  li a6, 0 #1
  
  lw a7, length  #длинна массива
  lw a1, result
  la t1, result

  la s2, array #нулевой элемент массива
  la s3, array #нулевой элемент массива

for_i:
  bgeu a4, a7, final #если i>=length, то идём в конец
  lw s0, 0(s2) #записываем в s0 элемент array[i]
for_j:
  bgeu a5, a7, total #если j>=length, то идём в конец цикла for_i
  lw s1, 0(s3) #записываем в s1 элемент array[j]
  beq s0, s1, c2plus #если array[i] == array[j], идём в count2++
  bge s0, s1, c1plus#если array[i] >= array[j], идём в count1++
  blt s0, s1, c1minus#если array[i] < array[j], идём в count1--

c1plus:
  addi a2, a2, 1 #увеличиваем первый счётчик на 1
```

```
27    jal zero, end_forj #переход в конец цикла
28  c1minus:
29    addi a2, a2, -1#уменьшаем первый счётчик на 1
30    jal zero, end_forj #переход в конец цикла
31  c2plus:
32    addi a3, a3, 1 #увеличиваем второй счётчик на 1
33    jal zero, end_forj #переход в конец цикла
34
35  end_forj:
36    addi s3, s3, 4
37    addi a5, a5, 1 #увеличиваем j на 1
38    jal zero, for_j #переход в for_j
39  total:
40    beq a3, a7, pre_final
41    beq a2, a6, pre_final
42    addi a6, a6, 1
43    beq a2, a6, division
44    addi a6, a6, -2
45    beq a2, a6, division
46    jal zero, end_fori
47  division:
48    srli s0, s0 1
49    add a1, a1, s0
50  end_fori:
51    addi s2, s2, 4 #переход к след. элементу массива
52    addi a4, a4, 1 #увеличиваем i на 1
```

```
53    la s3, array #обновляем цикл for_j, записав снова нулевой элемент в s3
54    li a2, 0 #count1 = 0 первый счётчик
55    li a3, 0 #count2 = 0 второй счётчик
56    li a5, 0
57    li a6, 0
58    jal zero, for_i #переход в for_i
59  pre_final:
60    add a1, a1, s0
61  final:
62    sw a1, 0(t1)
63    li a0, 10
64    ecall
65
66   .data #секция изменяемых данных
67  result:
68    .word 0
69
70
71    .rodata #секция неизменяемых данных
72  length:
73   .word 6
74  array:
75   .word 8, 1, 2, 20, 7, 11
```

Рис.1 Программа

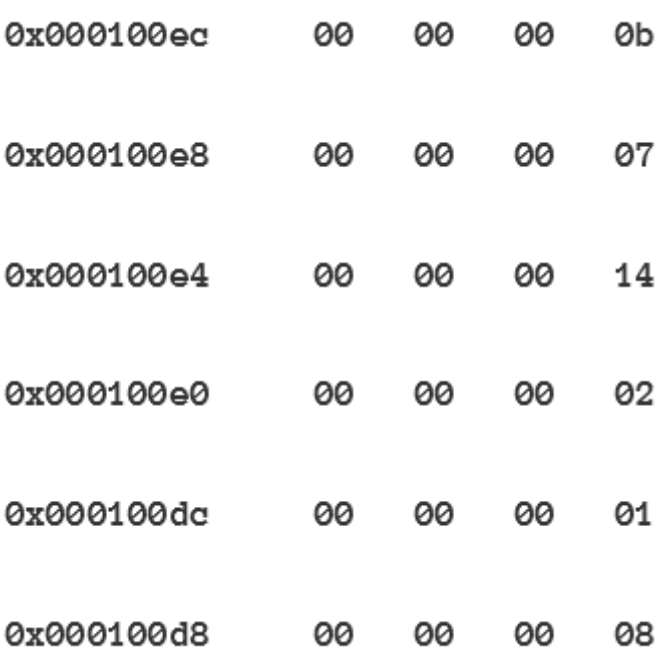

Рис.3. Массив входных данных

Рис.4. Ответ

Как видно по рис. 4, программа правильно определила медиану и записала её в память.

3.Реализация подпрограммы.

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

```
1 .text
2 __start:
3 .global __start
4   call main
5
6 finish:
7   li a0, 10
8   ecall
```

Рис.6. Тестирующая программа.

```
3 .global __start
4  la t2, result
5  la t3, array
6  lw t4, length
7
8  addi sp, sp, -16
9  sw ra, 12(sp)
10
11 call Medians
12
13 lw ra, 12(sp)
14 addi sp, sp, -16
15
16 li a0, 0
17 ret
18
19  .data #секция изменяемых данных
20 result:
21   .word 0
22
23
24   .rodata #секция неизменяемых данных
25 length:
26  .word 6
27 array:
28  .word 8, 1, 2, 20, 7, 11
```

Рис.7. Подпрограмма main.

Тестирующая программа вызовет подпрограмму main, которая в свою очередь вызовет основную программу Median.

```
.text
__start:
.global __start
  li a4, 0 #i = 0
  li a5, 0 #j = 0
  li a6, 0 #1

  mv t4, t4  #длинна массива


  mv t1, t2 #результат
  mv s2, t3 #нулевой элемент массива
  mv s3, t3 #нулевой элемент массива
  lw a1, t1
for_i:
  bgeu a4, a7, final #если i>=length, то идём в конец
  lw s0, 0(s2) #записываем в s0 элемент array[i]
for_j:
  bgeu a5, a7, total #если j>=length, то идём в конец цикла for_i
  lw s1, 0(s3) #записываем в s1 элемент array[j]
  beq s0, s1, c2plus #если array[i] == array[j], идём в count2++
  bge s0, s1, c1plus#если array[i] >= array[j], идём в count1++
  blt s0, s1, c1minus#если array[i] < array[j], идём в count1--

c1plus:
  addi a2, a2, 1 #увеличиваем первый счётчик на 1
  jal zero, end_forj #переход в конец цикла
c1minus:
  addi a2, a2, -1#уменьшаем первый счётчик на 1
  jal zero, end_forj #переход в конец цикла
c2plus:
  addi a3, a3, 1 #увеличиваем второй счётчик на 1
  jal zero, end_forj #переход в конец цикла

end_forj:
  addi s3, s3, 4
  addi a5, a5, 1 #увеличиваем j на 1
  jal zero, for_j #переход в for_j
total:
  beq a3, a7, pre_final
  beq a2, a6, pre_final
  addi a6, a6, 1
  beq a2, a6, division
  addi a6, a6, -2
  beq a2, a6, division
  jal zero, end_fori
division:
  srli s0, s0 1
  add a1, a1, s0
end_fori:
  addi s2, s2, 4 #переход к след. элементу массива
  addi a4, a4, 1 #увеличиваем i на 1
```

```
53   la s3, array #обновляем цикл for_j, записав снова нулевой элемент в s3
54   li a2, 0 #count1 = 0 первый счётчик
55   li a3, 0 #count2 = 0 второй счётчик
56   li a5, 0
57   li a6, 0
58   jal zero, for_i #переход в for_i
59 pre_final:
60   add a1, a1, s0
61 final:
62   sw a1, 0(t1)
63   li a0, 10
64   ret
```

Рис.8. Подпрограмма Median

4. Алгоритм работы нахождения медианы.

1 шаг. Запоминаем необходимые адреса(результат, нулевой эл-нт массива), а также значения(длина массива). Также устанавливаем нули в регистры, необходимые для работы

2 шаг. Записываем в две переменные эл-ты массива(a[i] и a[j]). Начинаем сравнивать их друг с другом.

3 шаг. Если a[i]< a[j] уменьшаем значение счётчика на 1, иначе увеличиваем. Если эл-ты равны, увеличиваем на 1 второй счётчик(это нужно, чтобы определять медиану в массиве, в котором все эл-ты одинаковые)

4 шаг. Исходя из результата счётчика в конце каждого цикла for_j определяем медиану. При нечётном массиве счётчик будет равен 0. При чётном 1/-1. Если массив чётный, также делим эл-нт массива на 2.

5 шаг. Если медиана не была найдена, обновляем цикл for_j и закидываем след. эл-нт a[i] и снова сравниваем.

6 шаг. Запись результата в память.

5.Вывод

Составленные программа и подпрограмма для RISC-V удовлетворяют варианту и находят медиану in-place.