Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Низкоуровневое программирование

Отчет по лабораторной работе №3

Программирование RISC-V

Работу выполнил:
Аникин Д.А.
Группа:
3530901/90004
Преподаватель:
Алексюк А.О.

Санкт-Петербург
2021

Содержание

1. Формулировка задания

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.

Формулировка варианта задания

Интегрирование табличной функции методом трапеций с «длинным» результатом. *Примечание: переполнение разрядной сетки предотвращается пользователем масштабированием параметра шага сетки.*

$$\int_a^b f(x)\,dx = \sum_{i=0}^{n-1} \frac{f(x_i) + f(x_{i+1})}{2}(x_{i+1} - x_i)$$

2. Организация программы

Входные и выходные данные

Входными данными для программы являются

1. Длина массивов X и Y, записанное в форме 32 битного двоичного слова (word). Записывается в регистр a5 и используется в дальнейшем как счетчик итераций.

2. Адреса массивов X и Y, записанные в форме 32 битного двоичного слова (word). Записываются в регистры a3 и a4 соответственно и используются в дальнейшем для указания адреса i-го элемента.

3. Элементы массивов X и Y хранятся в памяти в формате с плавающей запятой одинарной точности.

Выходным значением программы является значение определенного интеграла, полученное относительно X и Y. Записывается в регистр fa4.

Регулировка шага сетки предполагается модификацией значений X.

Алгоритм работы программы

На старте программы загружаются адреса массивов X и Y в регистры a3 и a4. Кроме этого загружается адрес, содержащий двойку, используемую при делении. Необходимость хранить 2 в памяти вызвана отсутствием аналога псевдоинструкии *li* для чисел в формате с плавающей запятой.

На этом месте можно оптимизировать программу - достаточно хранить адрес divisor и обращаться к остальным элементам .rodata через него, но это непрактично из-за необходимости расчета положения элементов относиельно divisor.

Далее происходит загрузка счетчика в регистр a5, декрементируется на единицу для формирования значения n - 1 и загружается 2 в регистр fa0.

На следующем этапе в цикле просчитывается значение слагаемого. Рабочими регистрами являются fa1 и fa2, где хранятся значения сначала x_{i+1} и x_i, а затем y_{i+1} и y_i. Промежуточное значение (x_{i+1} - x_i) хранится в регистре fa3, а (y_{i+1} - y_i) - в fa1. Значение полученного слагаемого складывается с предыдущим из регистра fa4 и записывается туда же.

В конце декрементируется счетчик, и инкрементируется адреса массивов для перехода к следующим элементам. Цикл повторяется, пока не обнулиться счетчик.

Код программы

Листинг 1: trapz.s

```
1  .globl __start
2
3  .rodata
4      divisor: .float 2
5
6      length: .word 5
7      datax: .float 1.91, 2.64, 3, 8.237, 9.001
8      datay: .float 4.3853, 2.854, 6.345, 3.1415, 8
9
10 .text
11     __start:
12         # Загрузка адресов
13         la a2, divisor
14         la a3, datax
15         la a4, datay
16
17         Загрузка# счетчика
18         lw a5, 4(a2)
19         addi a5, a5, -1
20
21         Загрузка# 2 в регистр
22         flw fa0, 0(a2)
```

```
23
24      loop:
25          flw fa1, 4(a3) #xi+1
26          flw fa2, 0(a3) #xi
27          fsub.s fa3, fa1, fa2
28
29          flw fa1, 4(a4) #yi+1
30          flw fa2, 0(a4) # yi
31          fadd.s fa1, fa2, fa1
32
33          fdiv.s fa1, fa1, fa0 # / 2
34          fmadd.s fa4, fa1, fa3, fa4
35
36          addi a5, a5, -1 # dec counter
37          addi a3, a3, 4 # inc datax addr
38          addi a4, a4, 4 # inc datay addr
39
40          bnez a5 loop
41
42      finish:
43          li a0, 10
44          ecall
```

3. Организация подпрограммы

Далее нам потребуется написать подпрограмму main, реализовав в ней функциональность тестовой программы. В то же время, код, обеспечивающий вызов main и завершение работы, может использоваться «как есть» в самых разных программах. Учитывая это, мы разобьем текст программы на 2 файла: setup.s и main.s.

Листинг 2: subprog/setup.s

```
1  .globl __start
2
3  .text
4      __start:
5          call main
6
7      finish:
8          mv a1, a0
9          li a0, 17 # выход с кодом завершения
10         ecall
```

Листинг 3: subprog/main.s

```
1  .globl main
2
3  .rodata
```

```
4       length: .word 5
5       datax: .float 1.91, 2.64, 3, 8.237, 9.001
6       datay: .float 4.3853, 2.854, 6.345, 3.1415, 8
7
8   .text
9       main:
10          addi sp, sp, -16 # выделение памяти в стеке
11          sw ra, 12(sp) # запись ra в стек
12
13          la a0, datax
14          la a1, datay
15          lw a2, length
16
17          call trapz
18
19          lw ra, 12(sp) # восстановление ra
20          addi sp, sp, 16 # освобождение стека
21
22          li a0, 0
23          ret
```

Подпрограмма main содержит входные данные:длина массивов X и Y и их адреса и элементы. Согласно ABI входные данные должны записываться в регистры аргументов: длина - в a2, адресс массива X - в a0, Y - в a1.

Исходное значение ra следует сохраняется перед псевдоинструкцией call, и восстанавлиается перед псевдоинструкцией ret. Это необходимо для избежания зацикливания программы, т.к иначе значение ra изменяется в результате выполнения псевдоинструкции call: в ra будет записан адрес возврата для вызываемой подпрограммы trapz, то есть адрес следующей за call псевдоинструкции, в данном случае - инструкции возврата из подпрограммы main. Таким образом, результатом выполнения инструкции возврата, соответствующе псевдоинструкции ret, будет переход на эту же инструкцию.

Рабочими регистрами подпрограммы trapz.s являются *temporary* регистры ft0 и ft1 для хранения x_{i+1} и x_i, а затем y_{i+1} и y_i. Промежуточное значение (x_{i+1} - x_i) хранится в регистре ft3, а (y_{i+1} - y_i) - в ft0. Значение полученного слагаемого складывается с предыдущим из регистра ft4 и записывается туда же. Двойка хранится в ft5.

Результат работы записывается в регистр fa0.

Листинг 4: subprog/trapz.s

```
1   .globl trapz
2
3   .rodata
4       divisor: .float 2
5
6   .text
7       trapz:
8           la t1, divisor
```

```
9
10          #counter
11          addi a2, a2, -1
12
13          #const 2
14          flw ft5, 0(t1)
15
16      loop:
17          flw ft0, 4(a0) #xi+1
18          flw ft1, 0(a0) #xi
19          fsub.s ft3, ft0, ft1
20
21          flw ft0, 4(a1)
22          flw ft1, 0(a1)
23          fadd.s ft0, ft0, ft1
24
25          fdiv.s ft0, ft0, ft5
26          fmadd.s ft4, ft0, ft3, ft4
27
28          addi a2, a2, -1
29          addi a0, a0, 4
30          addi a1, a1, 4
31
32          bnez a2 loop
33
34          fmv.s fa0, ft4
35          ret
```

4. Выводы

В ходе выполнения лабораторной работы была разработана программа на языке ассемблера RISC-V, выполняющая расчет значения определенного интеграла методом трапеций. Так же была создана тестовая программа, вызывающая подпрограмму, работающую с загруженными в нее данными о длине массивов и их адресами.