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

Отчёт по лабораторной работе

Дисциплина: Низкоуровневое программирование

Тема: Программирование RISC-5 (Вариант 1)

Выполнил студент гр. 3530901/00002 _____ А.Э. А.Л.Малак
(подпись)
Преподаватель _____ Д.С Степанов
(подпись)

“__” _____2021 г.

Санкт-Петербург

2021

Задачи:

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.

Задание по варианту:
Формирование в памяти десятичного представления целого числа со знаком.

1 Часть

Код без подпрограмм :

```
.text

start:

.global start

lw a1, number# a1 = number

la a2, array # a2 = array[0]

la a5, const

li a3, 10

li a4, 1

bltz a1, minus #a1 < 0, то мы переходим на minus

bgez a1, plus#a1 >= 0, то переходим на метку plus

minus:

  lw t0, 0(a2) #t0 = array[0] = 0

  addi t0, t0, 45 #t0 = array[0] = 45 = 0x2d

  sw t0, 0(a2) # array[0] = t0 = 45
```

```
  li a7, 0 # a5 = number
  sub a1, a7, a1 # a1 = 0 - number
  addi a2, a2, 4 # a2 = a2 + 4 - переход к следующему элементу в массиве
  bgez a1, loopConst # a1 >= 0, то идём в цикл
plus:
  lw t0, 0(a2) # t0 = array[0]
  addi t0, t0, 43 #t0 = array[0] = 43 = 0x2b
  sw t0, 0(a2) # array[0] = t0 = 43 = 0x2b
  addi a2, a2, 4 # a2 = a2 + 4 - переход к следующему элементу в массиве
loopConst:
   lw t1, 0(a5) #t1 = const[0]
  loopVal:
   lw t0, 0(a2) # t0 = array[1]
   sub a1, a1, t1 #a1 = a1 - a3 = a1 - 1000000000
   bltz a1, new1 #если a1<0, то выходим из второго цикла
   addi t0, t0, 1 #сюда прибавляем в ячейку +1 для разряда
   sw t0, 0(a2)# array[1] = t0
   bgez a1, loopVal # если a1>=0,то следующая итерация цикла
 new1:# loop1 -> loop2
   addi a2, a2, 4 # a2 = a2 + 4 - переход к следующему элементу в массиве
   add a1, a1, t1 # a1 = a1 + a3 = a1 + 1000000000 - восстановление числа
   addi a5, a5, 4 # a5 = a5 + 4 - переход к следующему элементу в массиве
   bgt a1, a3, loopConst


new9: # выход из loop10 и сохранение последнего разряда
```

```
  lw t0, 0(a2) # t0 = array[10]
  add t0, t0, a1 # t0 = t0 + a1
  sw t0, 0(a2) # array[10] = t0
finish:#выход из программы
  ecall
# данные
.data # изменяемые данные
number:
 .word -2147483640 # исходное число [-2147483648;2147483647]
array:
 .word 0,0,0,0,0,0,0,0,0,0
const:
 .word 1000000000,100000000,10000000,1000000,100000,10000,1000,100,10,0
```

Дано число -2147483647. В конце при выполнении этой программы мы должны получить массив array = [2d, 2, 1, 4, 7, 4, 8, 3, 6, 4, 7]

Если число положительное, то в первом элементе массива будет 2b, в обратном же случае 2d.

Убедимся в этом.

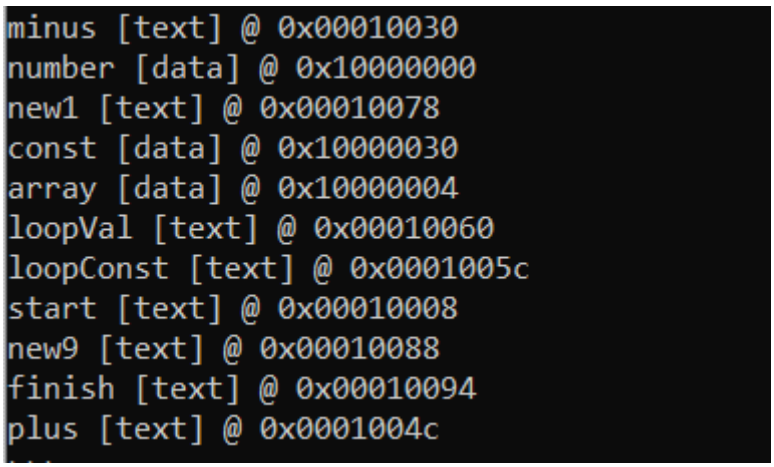

Рис.1 Все метки и данные в программе.

```
>>> memory 0x10000004
            Value (+0) Value (+4) Value (+8) Value (+c)
[0x10000004] 0x00000000 0x00000000 0x00000000 0x00000000
[0x10000014] 0x00000000 0x00000000 0x00000000 0x00000000
[0x10000024] 0x00000000 0x00000000 0x00000000 0x00000000
[0x10000034] 0x00000000 0x00000000 0x00000000 0x00000000
[0x10000044] 0x00000000 0x00000000 0x00000000 0x00000000
[0x10000054] 0x00000000 0x00000000 0x00000000 0x00000000
[0x10000064] 0x00000000 0x00000000 0x00000000 0x00000000
[0x10000074] 0x00000000 0x00000000 0x00000000 0x00000000
[0x10000084] 0x00000000 0x00000000 0x00000000 0x00000000
[0x10000094] 0x00000000 0x00000000 0x00000000 0x00000000
[0x100000a4] 0x00000000 0x00000000 0x00000000 0x00000000
[0x100000b4] 0x00000000 0x00000000 0x00000000 0x00000000
```

Рис.2 array до выполнения программы.

```
>>> breakpoint 0x00010094
>>> c
>>> memory 0x10000004
             Value (+0) Value (+4) Value (+8) Value (+c)
[0x10000004] 0x0000002d 0x00000002 0x00000001 0x00000004
[0x10000014] 0x00000007 0x00000004 0x00000008 0x00000003
[0x10000024] 0x00000006 0x00000004 0x00000000 0x3b9aca00
[0x10000034] 0x05f5e100 0x00989680 0x000f4240 0x000186a0
[0x10000044] 0x00002710 0x000003e8 0x00000064 0x0000000a
[0x10000054] 0x00000000 0x00000000 0x00000000 0x00000000
[0x10000064] 0x00000000 0x00000000 0x00000000 0x00000000
[0x10000074] 0x00000000 0x00000000 0x00000000 0x00000000
[0x10000084] 0x00000000 0x00000000 0x00000000 0x00000000
[0x10000094] 0x00000000 0x00000000 0x00000000 0x00000000
[0x100000a4] 0x00000000 0x00000000 0x00000000 0x00000000
[0x100000b4] 0x00000000 0x00000000 0x00000000 0x00000000
```

Рис.3. array до выполнения программы

2 Часть

Разделим нашу программу на несколько подпрограмм

Код вызывающей подпрограммы

```
.text
start:
.globl start
  call main # вызов подпрограммы main
finish:
  ecall
```

Код подпрограммы (файл main.s с данными)

```
#main.s
.text
main:
.globl main
lw a1, number# a1 = number
la a2, array # a2 = array[0]
la a5, const
li a3, 10
li a4, 1

addi sp, sp, -16
sw ra, 12(sp) # сохранение ссылки на возврат в прошлую подпрограмму

call subf #вызов подпрограммы subf

lw ra, 12(sp) # возврат ссылки на прошлую вызывающую подрограмму
addi sp, sp, 16

end_main:# выход из подпрограммы
```

```
  ret


# данные
.data # изменяемые данные
number:
 .word -2147483647# исходное число [-2147483648;2147483647]
array:
 .word 0,0,0,0,0,0,0,0,0,0
const:
 .word 1000000000,100000000,10000000,1000000,100000,10000,1000,100,10,0
```

Код подпрограммы (файл subf.s с циклами)

```
.text
subf:
.globl subf
bltz a1, minus #a1 < 0, то мы переходим на minus
bgez a1, plus#a1 >= 0, то переходим на метку plus
minus:
  lw t0, 0(a2) #t0 = array[0] = 0
  addi t0, t0, 45 #t0 = array[0] = 45 = 0x2d
  sw t0, 0(a2) # array[0] = t0 = 45
  li a7, 0 # a5 = number
  sub a1, a7, a1 # a1 = 0 - number
  addi a2, a2, 4 # a2 = a2 + 4 - переход к следующему элементу в массиве
  bgez a1, loopConst # a1 >= 0, то идём в цикл
plus:
  lw t0, 0(a2) # t0 = array[0]
  addi t0, t0, 43 #t0 = array[0] = 43 = 0x2b
  sw t0, 0(a2) # array[0] = t0 = 43 = 0x2b
  addi a2, a2, 4 # a2 = a2 + 4 - переход к следующему элементу в массиве
loopConst:
```

```
  lw t1, 0(a5) #t1 = const[0]
 loopVal:
  lw t0, 0(a2) # t0 = array[1]
  sub a1, a1, t1 #a1 = a1 - a3 = a1 - 1000000000
  bltz a1, new1 #если a1<0, то выходим из второго цикла
  addi t0, t0, 1 #сюда прибавляем в ячейку +1 для разряда
  sw t0, 0(a2)# array[1] = t0
  bgez a1, loopVal # если a1>=0,то следующая итерация цикла
new1:# loop1 -> loop2
  addi a2, a2, 4 # a2 = a2 + 4 - переход к следующему элементу в массиве
  add a1, a1, t1 # a1 = a1 + a3 = a1 + 1000000000 - восстановление числа
  addi a5, a5, 4 # a5 = a5 + 4 - переход к следующему элементу в массиве
  bgt a1, a3, loopConst
new9: # выход из loop10 и сохранение последнего разряда
 lw t0, 0(a2) # t0 = array[10]
 add t0, t0, a1 # t0 = t0 + a1
 sw t0, 0(a2) # array[10] = t0
fin:
 ret
```

```
number [data] @ 0x10000000
const [data] @ 0x10000030
array [data] @ 0x10000004
main [text] @ 0x00010014
end_main [text] @ 0x0001004c
C:\Users\mahon\Desktop\risc-5\subf.s
subf [text] @ 0x00010050
minus [text] @ 0x00010058
new1 [text] @ 0x000100a0
loopVal [text] @ 0x00010088
loopConst [text] @ 0x00010084
new9 [text] @ 0x000100b0
fin [text] @ 0x000100bc
plus [text] @ 0x00010074
C:\Users\mahon\Desktop\risc-5\np2_1.s
start [text] @ 0x00010008
finish [text] @ 0x00010010
```

Рис.4. Метки и данные в подпрограммах.

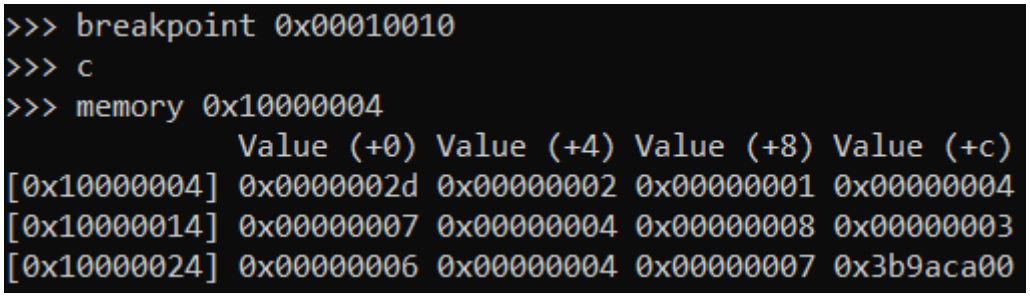

Рис.5. array после прохождения через все подпрограммы.

Вывод: В ходе выполнения лабораторной работы была реализована программа на RISCV, реализующая слияние двух отсортированных массивов, и работающая корректно. Также эта программа была представлена, как подпрограмма, из-за чего её можно использовать несколько раз в других программах.