Санкт-Петербургский политехнический университет Петра Великого

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчёт по лабораторной работе №3**

**Дисциплина**: Низкоуровневое программирование

**Тема**: программирование на языке riscV

Выполнил студент гр. 3530901/90003 А.А. Ундольский

(подпись)

Преподаватель А. О. Алексюк

(подпись)

“ ” 2021 г.

Санкт-Петербург

2021

Оглавление

[1. Техническое задание 3](#_Toc68519279)

[2. Метод решения 3](#_Toc68519280)

[3. Программа nodPrg 3](#_Toc68519281)

[4. Работа программы nodPrg 5](#_Toc68519282)

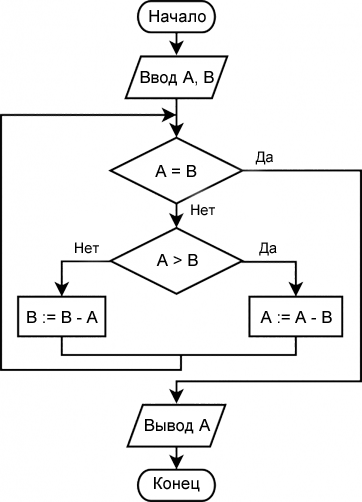
[5. Программа nodSub 5](#_Toc68519283)

[6. Работа программы nodSub 7](#_Toc68519284)

# Техническое задание

* Разработать программу на языке ассемблера RISC-V находящую НОД массива, 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.
* Выделить нахождение НОД массива 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.

# Метод решения

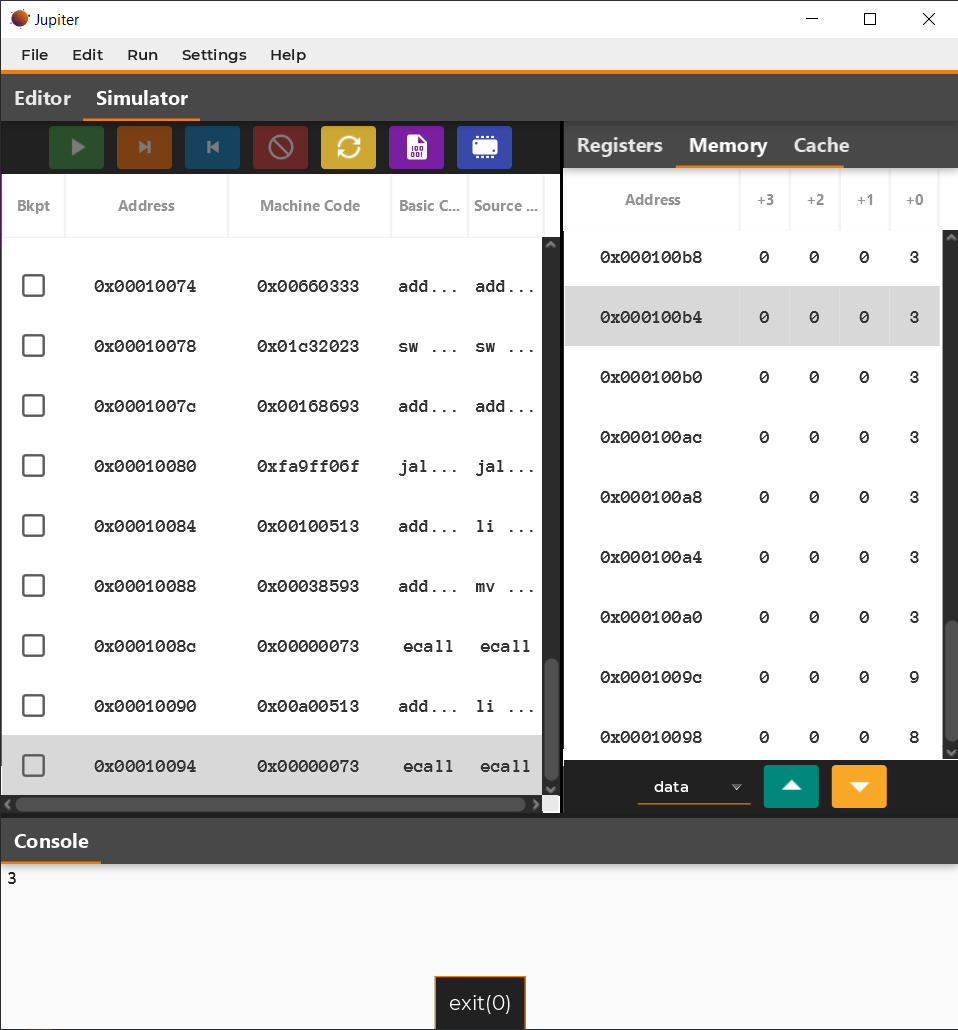
Нахождение НОД осуществляется по алгоритму Евклида   
(методом вычитания):

Если A=B, то НОД найден и он равен одному из чисел, иначе необходимо большее из двух чисел заменить разностью его и меньшего.

# Программа nodPrg

|  |
| --- |
| #nodPrg  #Техническое задание – Нахождение НОД массива  .globl \_\_start  #Входные данные:  .rodata  array\_length: #длина массива  .word 8  .data  array: #массив  .word 9,18,30,60,120,36,33,24  .text #Секция кода  \_\_start:  la a1, array\_length  lw a1, 0(a1) #a1 := array\_length  li a5, 0  addi a5, a1, -1 # a5 := array\_length-1  la a2, array #a2 = a[0]  li a3, 0 # a3 = 0    loop:  beq a3, a5, loop\_exit # if (a3 == a5) goto loop\_exit1    #t1:=addr(a[i]);  slli t1, a3, 2 # t1= a3 << 2 = a3 \* 4  add t1, a2, t1 # t1 = a2 + t1 = a2 + a3 \* 4    lw t2, 0(t1) #t2 := array[i]    addi a3, a3, 1 #a3++    slli t1, a3, 2 # t1= a3 << 2 = a3 \* 4  add t1, a2, t1 # t1 = a2 + t1 = a2 + a3 \* 4    lw t3, 0(t1) #t3 := array[i+1]    loop\_nod:  beq t2, t3, loop\_nod\_exit # if (t2 == t3) got loop\_nod\_exit  bgtu t2, t3, a\_minus\_b # if (t2 > t3) goto a\_minus\_b  #(t2 <= t3)  sub t4, t3, t2 # b = b - a  mv t3, t4  jal zero, loop\_nod  a\_minus\_b:  #(t2 > t3)  sub t4, t2, t3 #a = a - b  mv t2, t4  jal zero, loop\_nod  loop\_nod\_exit:    sw t2, 0(t1) #array[i+1]:= t2    addi a3, a3, -1 #a3--    slli t1, a3, 2 # t1 = a3 << 2 = a3 \* 4  add t1, a2, t1 # t1 = a2 + t1 = a2 + a3 \* 4    sw t3, 0(t1) # array[i]:= t3    addi a3, a3, 1 # a3++  jal zero, loop  loop\_exit:  # prints the result in t2  li a0, 1  mv a1, t2  ecall    finish:  li a0, 10 # x10 := 10  ecall # ecall при значении x10 = 10 => остановка |

# Работа программы nodPrg



# Программа nodSub

Основная программа

|  |
| --- |
| # main.s  .global \_\_start  #Определим данные:  .rodata  array\_length:  .word 8  #Секция изменяемых данных:  .data  array: #массив  .word 9,18,30,60,120,36,33,24    .text  \_\_start:  addi sp, sp, -16 # выделение памяти в стеке  sw ra, 12(sp) # сохранение ra  lw a1, array\_length # \  la a2, array # } nodSub(array\_length, array);  call nodSub # /    lw ra, 12(sp) # восстановление ra  addi sp, sp, 16 # освобождение памяти в стеке  li a1, 0  li a0, 17  ecall |

Подпрограмма для поиска НОД массива

|  |
| --- |
| # nodSub.s  # Подпрограмма для нахождения НОД массива  .global nodSub  .text  nodSub:  #Arguments    #array\_length -> a1  #Addr(a[0]) -> a2    li a5, 0  addi a5, a1, -1 # a5 := array\_length-1  li a3, 0 # a3 = 0    loop:  beq a3, a5, loop\_exit # if (a3 == a5) goto loop\_exit1    #t1:=addr(a[i]);  slli t1, a3, 2 # t1= a3 << 2 = a3 \* 4  add t1, a2, t1 # t1 = a2 + t1 = a2 + a3 \* 4    lw t2, 0(t1) #t2 := array[i]    addi a3, a3, 1 #a3++    slli t1, a3, 2 # t1= a3 << 2 = a3 \* 4  add t1, a2, t1 # t1 = a2 + t1 = a2 + a3 \* 4    lw t3, 0(t1) #t3 := array[i+1]    loop\_nod:  beq t2, t3, loop\_nod\_exit # if (t2 == t3) got loop\_nod\_exit  bgtu t2, t3, a\_minus\_b # if (t2 > t3) goto a\_minus\_b  #(t2 <= t3)  sub t4, t3, t2 # b = b - a  mv t3, t4  jal zero, loop\_nod  a\_minus\_b:  #(t2 > t3)  sub t4, t2, t3 #a = a - b  mv t2, t4  jal zero, loop\_nod  loop\_nod\_exit:    sw t2, 0(t1) #array[i+1]:= t2    addi a3, a3, -1 #a3--    slli t1, a3, 2 # t1 = a3 << 2 = a3 \* 4  add t1, a2, t1 # t1 = a2 + t1 = a2 + a3 \* 4    sw t3, 0(t1) # array[i]:= t3    addi a3, a3, 1 # a3++  jal zero, loop  loop\_exit:  # prints the result in t2  li a0, 1  mv a1, t2  ecall  ret # jalr zero, ra, 0 |

# Работа программы nodSub

