Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа информатики и вычислительной техники

**Лабораторная работа № 4**

**Дисциплина**: Низкоуровневое программирование

**Тема:** Раздельная компиляция

Выполнил студент гр. 3530901/10005 Тучков Д.А.

Преподаватель Коренев Д. А.

«30» ноября 2022 г.

Санкт-Петербург

2022

Оглавление

[**Техническое задание** 3](#_Toc120300519)

[**1. Программа на языке C** 3](#_Toc120300520)

[**2. Сборка программы «по шагам»** 5](#_Toc120300521)

[Препроцессирование 6](#_Toc120300522)

[Компиляция 6](#_Toc120300523)

[Ассемблирование 11](#_Toc120300524)

[Компоновка 13](#_Toc120300525)

[**3. Создание статической библиотеки и make-файлов** 16](#_Toc120300526)

[**Вывод** 20](#_Toc120300527)

Техническое задание

1. **Формулировка задачи**

1) На языке C разработать функцию, реализующую определенную вариантом задания функциональность. Поместить определение функции в отдельный исходный файл, оформить заголовочный файл. Разработать тестовую программу на языке C.

2) Собрать программу «по шагам». Проанализировать выход препроцессора и компилятора. Проанализировать состав и содержимое секций, таблицы символов, таблицы перемещений и отладочную информацию, содержащуюся в объектных файлах иисполняемом файле.

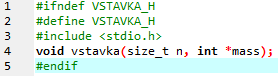
3) Выделить разработанную функцию в статическую библиотеку. Разработать make-файлыд ля сборки библиотеки и использующей ее тестовой программы. Проанализировать ход сборки библиотеки и программы, созданные файлы зависимостей.

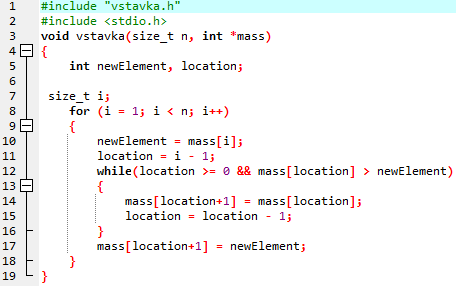
1. **Вариант задания**

Сортировка массива методом вставки.

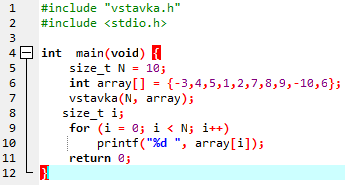
1. **Ход решения**

Листинг 1.1. Заголовочный файл vstavka.h

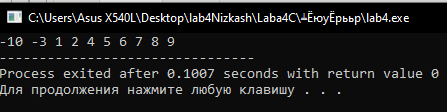
Листинг 1.2. Основной файл vstavka.c



Листинг 1.3. Тестовая программа main.c



Работа программы



**2. Сборка программы по шагам**

Препроцессирование, компиляция и ассемблирование

riscv64-unknown-elf-gcc --save-temps -march=rv32i -mabi=ilp32 -O1 -v main.c >log 2>&1

riscv64-unknown-elf-gcc --save-temps -march=rv32i -mabi=ilp32 -O1 -v vstavka.c >log 2>&1

В файлах main.i и vstavka.i содержится результат препроцессирования

Листинг 2.1. Файл main.i

# 797 "c:\\users\\asus x540l\\ld\\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86\_64-w64-mingw32\\riscv64-unknown-elf\\include\\stdio.h" 3

# 4 "vstavka.h" 2

# 4 "vstavka.h"

void vstavka(size\_t n, int \*mass);

# 2 "main.c" 2

int main(void) {

size\_t N = 10;

int array[] = {-3,4,5,1,2,7,8,9,-10,6};

vstavka(N, array);

size\_t i;

for (i = 0; i < N; i++)

printf("%d ", array[i]);

return 0;

}

Листинг 2.2. Файл vstavka.i

# 797 "c:\\users\\asus x540l\\ld\\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86\_64-w64-mingw32\\riscv64-unknown-elf\\include\\stdio.h" 3

# 4 "vstavka.h" 2

# 4 "vstavka.h"

void vstavka(size\_t n, int \*mass);

# 2 "vstavka.c" 2

void vstavka(size\_t n, int \*mass)

{

int newElement, location;

size\_t i;

for (i = 1; i < n; i++)

{

newElement = mass[i];

location = i - 1;

while(location >= 0 && mass[location] > newElement)

{

mass[location+1] = mass[location];

location = location - 1;

}

mass[location+1] = newElement;

}

}

Компиляция

В файле main.s, так как в нем можно заметить обращение к подпрограмме vstavka.

.file "main.c"

.option nopic

.attribute arch, "rv64i2p0\_m2p0\_a2p0\_f2p0\_d2p0\_c2p0"

.attribute unaligned\_access, 0

.attribute stack\_align, 16

.text

.align 1

.globl main

.type main, @function

main:

addi sp,sp,-80

sd ra,72(sp)

sd s0,64(sp)

sd s1,56(sp)

sd s2,48(sp)

lui a5,%hi(.LANCHOR0)

addi a5,a5,%lo(.LANCHOR0)

ld a1,0(a5)

ld a2,8(a5)

ld a3,16(a5)

ld a4,24(a5)

ld a5,32(a5)

sd a1,8(sp)

sd a2,16(sp)

sd a3,24(sp)

sd a4,32(sp)

sd a5,40(sp)

addi a1,sp,8

li a0,10

call vstavka

addi s0,sp,8

addi s2,sp,48

lui s1,%hi(.LC1)

.L2:

lw a1,0(s0)

addi a0,s1,%lo(.LC1)

call printf

addi s0,s0,4

bne s0,s2,.L2

li a0,0

ld ra,72(sp)

ld s0,64(sp)

ld s1,56(sp)

ld s2,48(sp)

addi sp,sp,80

jr ra

.size main, .-main

.section .rodata

.align 3

.set .LANCHOR0,. + 0

.LC0:

.word -3

.word 4

.word 5

.word 1

.word 2

.word 7

.word 8

.word 9

.word -10

.word 6

.section .rodata.str1.8,"aMS",@progbits,1

.align 3

.LC1:

.string "%d "

.ident "GCC: (SiFive GCC 8.3.0-2020.04.1) 8.3.0"

Листинг 2.4. Файл vstavka.s

.file "vstavka.c"

.option nopic

.attribute arch, "rv32i2p0"

.attribute unaligned\_access, 0

.attribute stack\_align, 16

.text

.align 2

.globl vstavka

.type vstavka, @function

vstavka:

li a5,1

bleu a0,a5,.L1

addi a6,a1,4

addi a7,a0,-1

li a0,0

j .L5

.L3:

addi a5,a5,1

slli a5,a5,2

add a5,a1,a5

sw a2,0(a5)

addi a0,a0,1

addi a6,a6,4

beq a0,a7,.L1

.L5:

lw a2,0(a6)

mv a5,a0

blt a0,zero,.L3

lw a3,-4(a6)

bge a2,a3,.L3

mv a4,a6

.L4:

sw a3,0(a4)

addi a5,a5,-1

blt a5,zero,.L3

addi a4,a4,-4

lw a3,-4(a4)

bgt a3,a2,.L4

j .L3

.L1:

ret

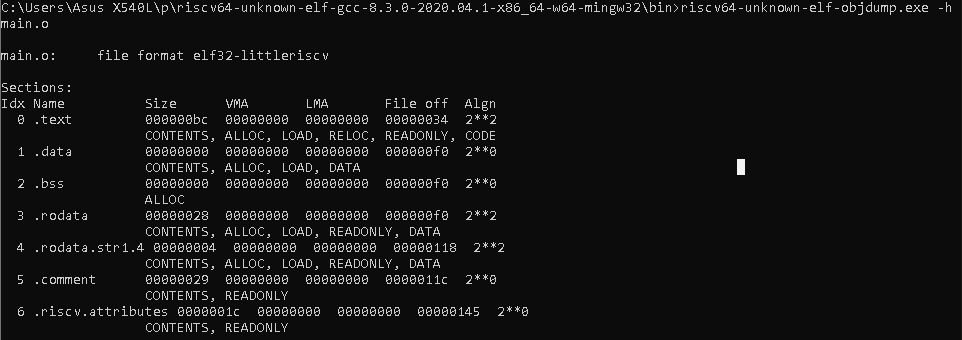
.size vstavka, .-vstavka

.ident "GCC: (SiFive GCC 8.3.0-2020.04.1) 8.3.0"

Ассемблирование

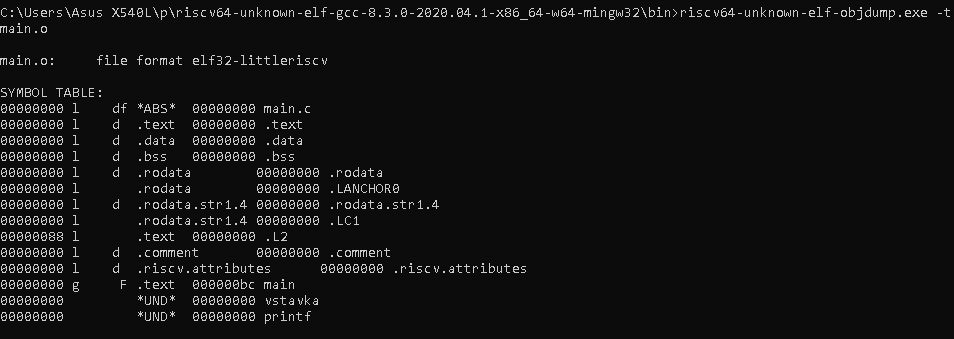
Листинг 2.5. Заголовки секций файла main.o

riscv64-unknown-elf-objdump.exe -h main.o



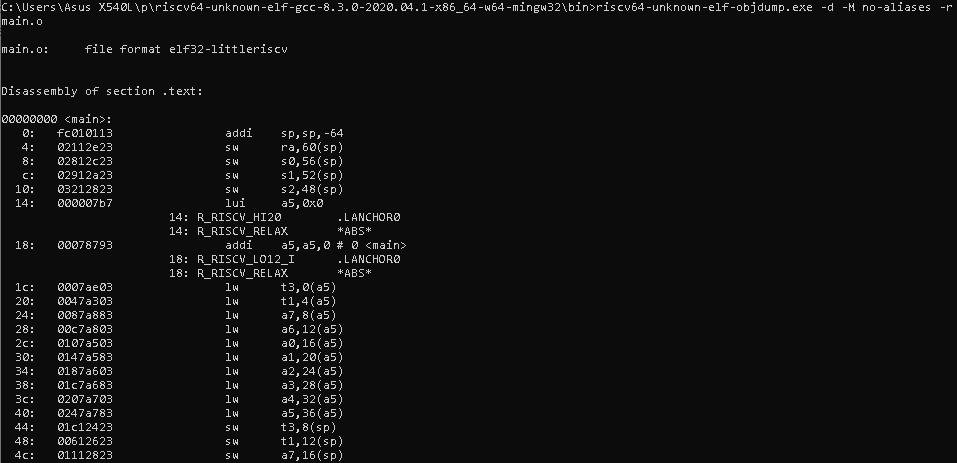
Листинг 2.6. Таблица символов файла main.o

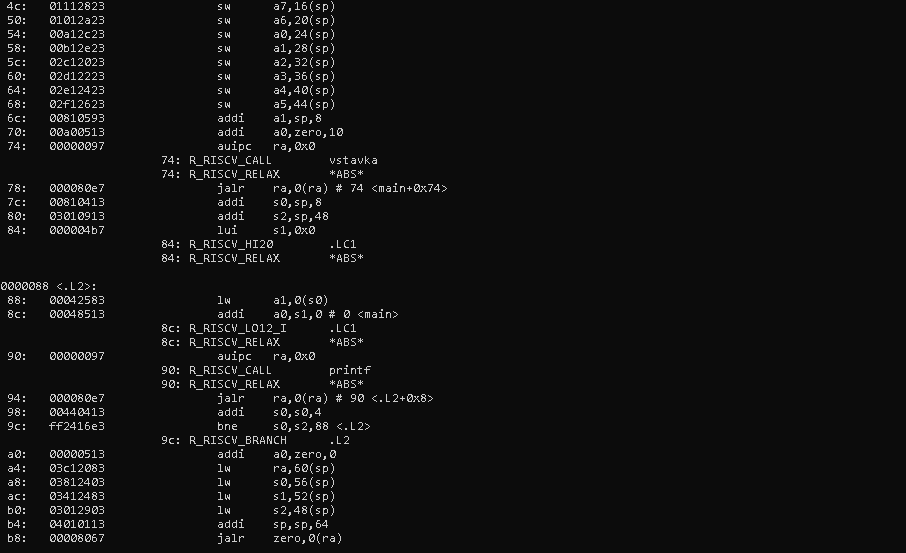
riscv64-unknown-elf-objdump.exe –t main.o



Листинг 2.7. Таблица перемещений файла main.o

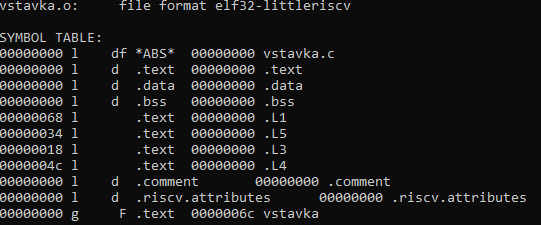
riscv64-unknown-elf-objdump.exe -d -M no-aliases -r main.o





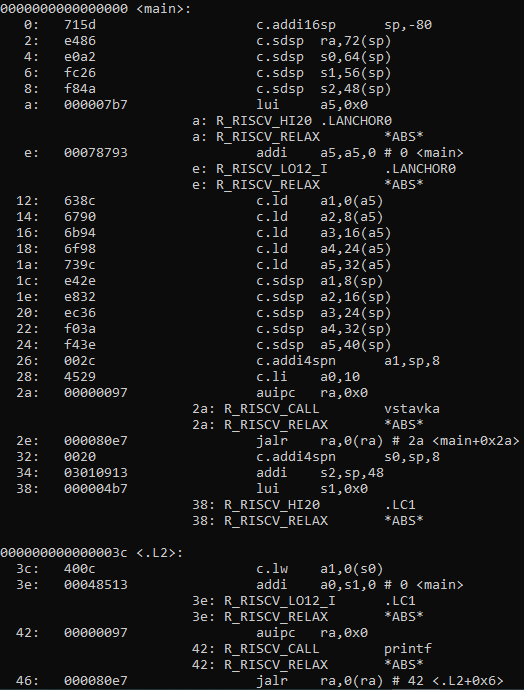
Листинг 2.8. Заголовки секций файла vstavka.o

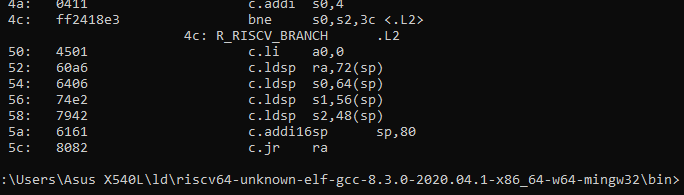
riscv64-unknown-elf-objdump -t vstavka.o



Листинг 2.9. файл main.o

riscv64-unknown-elf-objdump.exe -d -M no-aliases -r main.o





Компановка

riscv64-unknown-elf-gcc.exe -v main.o vstavka.o



Листинг 2.10. Исполняемый файл a.out

riscv64-unknown-elf-objdump.exe –j .text –d –M no-aliases a.out >a.ds

0000000000010158 <main>:

10158: 715d c.addi16sp sp,-80

1015a: e486 c.sdsp ra,72(sp)

1015c: e0a2 c.sdsp s0,64(sp)

1015e: fc26 c.sdsp s1,56(sp)

10160: f84a c.sdsp s2,48(sp)

10162: 67f1 c.lui a5,0x1c

10164: 11078793 addi a5,a5,272 # 1c110 <\_\_clzdi2+0x30>

10168: 638c c.ld a1,0(a5)

1016a: 6790 c.ld a2,8(a5)

1016c: 6b94 c.ld a3,16(a5)

1016e: 6f98 c.ld a4,24(a5)

10170: 739c c.ld a5,32(a5)

10172: e42e c.sdsp a1,8(sp)

10174: e832 c.sdsp a2,16(sp)

10176: ec36 c.sdsp a3,24(sp)

10178: f03a c.sdsp a4,32(sp)

1017a: f43e c.sdsp a5,40(sp)

1017c: 002c c.addi4spn a1,sp,8

1017e: 4529 c.li a0,10

10180: 02a000ef jal ra,101aa <vstavka>

10184: 0020 c.addi4spn s0,sp,8

10186: 03010913 addi s2,sp,48

1018a: 64f1 c.lui s1,0x1c

1018c: 400c c.lw a1,0(s0)

10186: 03010913 addi s2,sp,48

1018a: 64f1 c.lui s1,0x1c

1018c: 400c c.lw a1,0(s0)

10186: 03010913 addi s2,sp,48

1018a: 64f1 c.lui s1,0x1c

1018c: 400c c.lw a1,0(s0)

1018e: 13848513 addi a0,s1,312 # 1c138 <\_\_clzdi2+0x58>

10192: 204000ef jal ra,10396 <printf>

10196: 0411 c.addi s0,4

10198: ff241ae3 bne s0,s2,1018c <main+0x34>

1019c: 4501 c.li a0,0

1019e: 60a6 c.ldsp ra,72(sp)

101a0: 6406 c.ldsp s0,64(sp)

101a2: 74e2 c.ldsp s1,56(sp)

101a4: 7942 c.ldsp s2,48(sp)

101a6: 6161 c.addi16sp sp,80

101a8: 8082 c.jr ra

00000000000101aa <vstavka>:

101aa: 4785 c.li a5,1

101ac: 04a7d663 bge a5,a0,101f8 <vstavka+0x4e>

101b0: 00458813 addi a6,a1,4

101b4: fff5031b addiw t1,a0,-1

101b8: 4881 c.li a7,0

101ba: 557d c.li a0,-1

101bc: a809 c.j 101ce <vstavka+0x24>

101be: 0785 c.addi a5,1

101c0: 078a c.slli a5,0x2

101c2: 97ae c.add a5,a1

101c4: c390 c.sw a2,0(a5)

101c6: 2885 c.addiw a7,1

101c8: 0811 c.addi a6,4

101ca: 02688763 beq a7,t1,101f8 <vstavka+0x4e>

101ce: 00082603 lw a2,0(a6)

101d2: 0008879b addiw a5,a7,0

101d6: fe07c4e3 blt a5,zero,101be <vstavka+0x14>

101da: ffc82683 lw a3,-4(a6)

101de: fed650e3 bge a2,a3,101be <vstavka+0x14>

101e2: 8742 c.mv a4,a6

101e4: c314 c.sw a3,0(a4)

101e6: 37fd c.addiw a5,-1

101e8: fca78be3 beq a5,a0,101be <vstavka+0x14>

101ec: 1771 c.addi a4,-4

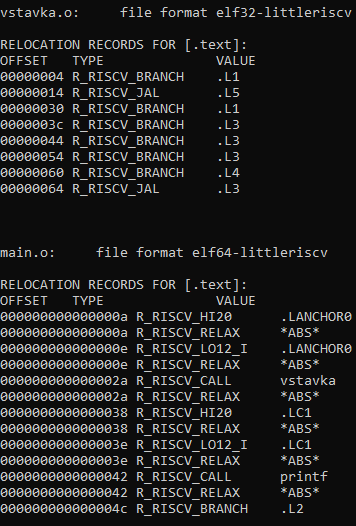
101ee: ffc72683 lw a3,-4(a4)

101f2: fed649e3 blt a2,a3,101e4 <vstavka+0x3a>

101f6: b7e1 c.j 101be <vstavka+0x14>

101f8: 8082 c.jr ra

Листинг 2.11. Исполняемый файл a.out



**3. Создание статической библиотеки и make-файлов**

Сделаем из vstavka.c статическую библиотеку vstav, тестовую программу main.c оставим без изменений.

получим объектный файл vstavka.o

riscv64-unknown-elf-gcc.exe -O1 -c vstavka.c -o vstav.o

из получившегося файла делаем библиотеку

riscv64-unknown-elf-ar.exe -rsc vstav.a vstavka.o

собираем исполняющий файл

riscv64-unknown-elf-gcc.exe -O1 --save-temps main.c vstav.a

Листинг 3.1. Таблица символов исполняемого файла



Листинг 3.2. Makefile для создания статической библиотеки

# Цели

.PHONY: all clean

# Исходные файлы для сборки библиотеки

OBJS= vstavka.c

#Вызываемые приложения

AR = riscv64-unknown-elf-ar.exe

CC = riscv64-unknown-elf-gcc.exe

# Файл библиотеки

MYLIBNAME = vstav.a

# Параметры компиляции

CFLAGS= -O1

# файлы искать в данном каталоге

INCLUDES+= -I .

# Make ищет файлы ... .h и ... .c в текущей директории

vpath %.h .

vpath %.c .

# $< = %.c

# $@ = %.o

%.o: %.c

$(CC) -MD $(CFLAGS) $(INCLUDES) -c $< -o $@

# Для того чтобы выполнить задачу "all", требуется построить библиотеку

all: $(MYLIBNAME)

# $^ = (vstavka.o)

$(MYLIBNAME): vstavka.o

$(AR) -rsc $@ $^

Листинг 3.3. Makefile для сборки исполняемого файла

# Цели

.PHONY: all clean

# Исходные файлы для сборки библиотеки

OBJS= main.c \

vstav.a

#Вызываемые приложения

CC = riscv64-unknown-elf-gcc.exe

# Компиляция

CFLAGS= -O1 --save-temps

#файлы искать в данном каталоге

INCLUDES+= -I .

# Make ищет файлы … .c и … .a в текущей директории

vpath %.c .

vpath %.a .

# Для того чтобы выполнить задачу "all", требуется построить библиотеку

all: a.out

# Сборка файла

a.out: $(OBJS)

$(CC) $(CFLAGS) $(INCLUDES) $^

del \*.o \*.i \*.s \*.d

Листинг 3.4. Запуск Makefile

Запускаем Makefile.win, потом Makefile1.win со сборкой исполняемого файла.

mingw32-make.exe -f Makefile.win Makefile1.win

Листинг 3.5. Таблица символов исполняемого файла, созданного с помощью Makefile

Созданный файл идентичен файлу созданному ранее

Вывод

В ходе работы была разработана программа на языке С и выполнены все задачи в соответствии с техническим заданием. Получены навыки работы с препроцессором, компилятором, ассемблером и компоновщиком.