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 Ухов А.Д.

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. **Ходрешения**

Листинг 1.1. Заголовочный файл sdvig.h

#ifndef SDVIG\_H

#define SDVIG\_H

void sdvig(int n, int \*mass);

#endif

Листинг 1.2. Основной файл sdvig.c

#include "sdvig.h"

void read\_array(int n, int\*\* values) {

int i = 0;

for (i; i < n; ++i) {

printf("values[%d] = ", i);

scanf("%d", &((\*values)[i]));

}

}

void print\_array(int n, int\* values) {

int i = 0;

for (i; i < n; ++i) {

printf("values[%d] = %d\n", i, values[i]);

}

}

void shift\_left(int \*a, int n) {

if (0 == n)

return;

int first = a[0];

int i = 1;

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

a[i-1] = a[i];

a[n-1] = first;

}

void shift\_left\_on(int \*a, int n, int shift\_size) {

shift\_size = shift\_size % n;

shift\_size = n - shift\_size;

int i = 0;

for (i; i < shift\_size; ++i) {

shift\_left(a, n);

}

}Листинг 1.3. Тестовая программа main.c

#include "sdvig.h"

int main() {

int n, m;

int \*values;

printf("n: ");

scanf("%d", &n);

printf("m: ");

scanf("%d", &m);

int array\_size = n+m;

values = malloc(sizeof(int) \* array\_size);

read\_array(array\_size, &values);

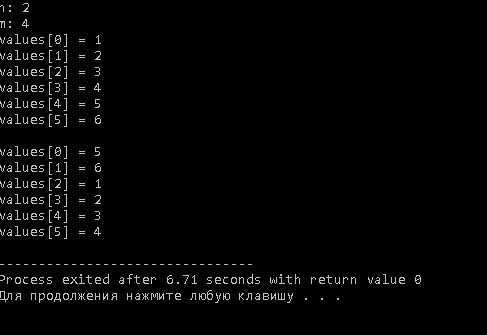
shift\_left\_on(values, array\_size, m);

printf("\n");

print\_array(array\_size, values);

return 0;

}



**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 sdvig.c >log 2>&1

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

Листинг 2.1. Файл main.i

# 2 "main.c" 2

int main() {

int n, m;

int \*values;

printf("n: ");

scanf("%d", &n);

printf("m: ");

scanf("%d", &m);

int array\_size = n+m;

values = malloc(sizeof(int) \* array\_size);

read\_array(array\_size, &values);

shift\_left\_on(values, array\_size, m);

printf("\n");

print\_array(array\_size, values);

return 0;

}

Листинг 2.2. Файл sdvig.i

# 1 "sdvig.c"

# 1 "<built-in>"

# 1 "<command-line>"

# 1 "sdvig.c"

# 1 "sdvig.h" 1

void sdvig(int n, int \*mass);

# 2 "sdvig.c" 2

void read\_array(int n, int\*\* values) {

int i = 0;

for (i; i < n; ++i) {

printf("values[%d] = ", i);

scanf("%d", &((\*values)[i]));

}

}

void print\_array(int n, int\* values) {

int i = 0;

for (i; i < n; ++i) {

printf("values[%d] = %d\n", i, values[i]);

}

}

void shift\_left(int \*a, int n) {

if (0 == n)

return;

int first = a[0];

int i = 1;

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

a[i-1] = a[i];

a[n-1] = first;

}

void shift\_left\_on(int \*a, int n, int shift\_size) {

int i = 0;

for (i; i < shift\_size; ++i) {

shift\_left(a, n);

}

}

Компиляция

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

.file "main.c"

.option nopic

.attribute arch, "rv32i2p0"

.attribute unaligned\_access, 0

.attribute stack\_align, 16

.text

.align 2

.globl main

.type main, @function

main:

addi sp,sp,-32

sw ra,28(sp)

sw s0,24(sp)

lui a0,%hi(.LC0)

addi a0,a0,%lo(.LC0)

call printf

addi a1,sp,12

lui s0,%hi(.LC1)

addi a0,s0,%lo(.LC1)

call scanf

lui a0,%hi(.LC2)

addi a0,a0,%lo(.LC2)

call printf

addi a1,sp,8

addi a0,s0,%lo(.LC1)

call scanf

lw s0,12(sp)

lw a5,8(sp)

add s0,s0,a5

slli a0,s0,2

call malloc

sw a0,4(sp)

addi a1,sp,4

mv a0,s0

call read\_array

lw a2,8(sp)

mv a1,s0

lw a0,4(sp)

call shift\_left\_on

li a0,10

call putchar

lw a1,4(sp)

mv a0,s0

call print\_array

li a0,0

lw ra,28(sp)

lw s0,24(sp)

addi sp,sp,32

jr ra

.size main, .-main

.section .rodata.str1.4,"aMS",@progbits,1

.align 2

.LC0:

.string "n: "

.LC1:

.string "%d"

.zero 1

.LC2:

.string "m: "

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

Листинг 2.4. Файл sdvig.s

.file "sdvig.c"

.option nopic

.attribute arch, "rv32i2p0"

.attribute unaligned\_access, 0

.attribute stack\_align, 16

.text

.align 2

.globl read\_array

.type read\_array, @function

read\_array:

ble a0,zero,.L6

addi sp,sp,-32

sw ra,28(sp)

sw s0,24(sp)

sw s1,20(sp)

sw s2,16(sp)

sw s3,12(sp)

sw s4,8(sp)

mv s2,a1

mv s1,a0

li s0,0

lui s4,%hi(.LC0)

lui s3,%hi(.LC1)

.L3:

mv a1,s0

addi a0,s4,%lo(.LC0)

call printf

slli a5,s0,2

lw a1,0(s2)

add a1,a1,a5

addi a0,s3,%lo(.LC1)

call scanf

addi s0,s0,1

bne s1,s0,.L3

lw ra,28(sp)

lw s0,24(sp)

lw s1,20(sp)

lw s2,16(sp)

lw s3,12(sp)

lw s4,8(sp)

addi sp,sp,32

jr ra

.L6:

ret

.size read\_array, .-read\_array

.align 2

.globl print\_array

.type print\_array, @function

print\_array:

ble a0,zero,.L14

addi sp,sp,-32

sw ra,28(sp)

sw s0,24(sp)

sw s1,20(sp)

sw s2,16(sp)

sw s3,12(sp)

mv s2,a0

mv s1,a1

li s0,0

lui s3,%hi(.LC2)

.L11:

lw a2,0(s1)

mv a1,s0

addi a0,s3,%lo(.LC2)

call printf

addi s0,s0,1

addi s1,s1,4

bne s2,s0,.L11

lw ra,28(sp)

lw s0,24(sp)

lw s1,20(sp)

lw s2,16(sp)

lw s3,12(sp)

addi sp,sp,32

jr ra

.L14:

ret

.size print\_array, .-print\_array

.align 2

.globl shift\_left

.type shift\_left, @function

shift\_left:

beq a1,zero,.L17

lw a2,0(a0)

li a5,1

ble a1,a5,.L19

addi a5,a0,4

slli a3,a1,2

add a3,a0,a3

.L20:

lw a4,0(a5)

sw a4,-4(a5)

addi a5,a5,4

bne a5,a3,.L20

.L19:

slli a1,a1,2

add a1,a0,a1

sw a2,-4(a1)

.L17:

ret

.size shift\_left, .-shift\_left

.align 2

.globl shift\_left\_on

.type shift\_left\_on, @function

shift\_left\_on:

ble a2,zero,.L27

addi sp,sp,-32

sw ra,28(sp)

sw s0,24(sp)

sw s1,20(sp)

sw s2,16(sp)

sw s3,12(sp)

mv s1,a2

mv s3,a1

mv s2,a0

li s0,0

.L24:

mv a1,s3

mv a0,s2

call shift\_left

addi s0,s0,1

bne s1,s0,.L24

lw ra,28(sp)

lw s0,24(sp)

lw s1,20(sp)

lw s2,16(sp)

lw s3,12(sp)

addi sp,sp,32

jr ra

.L27:

ret

.size shift\_left\_on, .-shift\_left\_on

.section .rodata.str1.4,"aMS",@progbits,1

.align 2

.LC0:

.string "values[%d] = "

.zero 2

.LC1:

.string "%d"

.zero 1

.LC2:

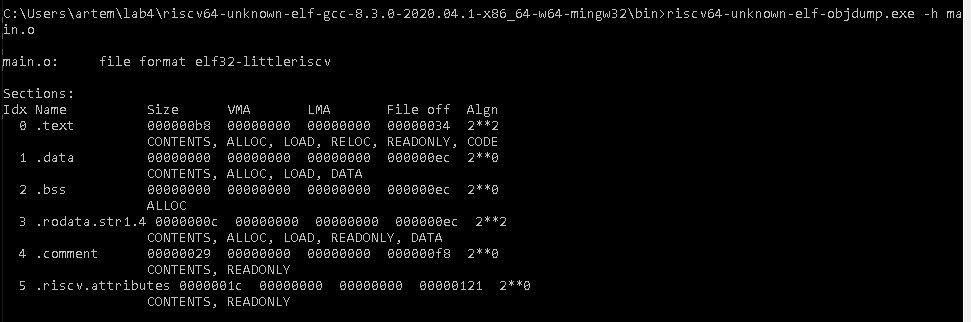
.string "values[%d] = %d\n"

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

Ассемблирование

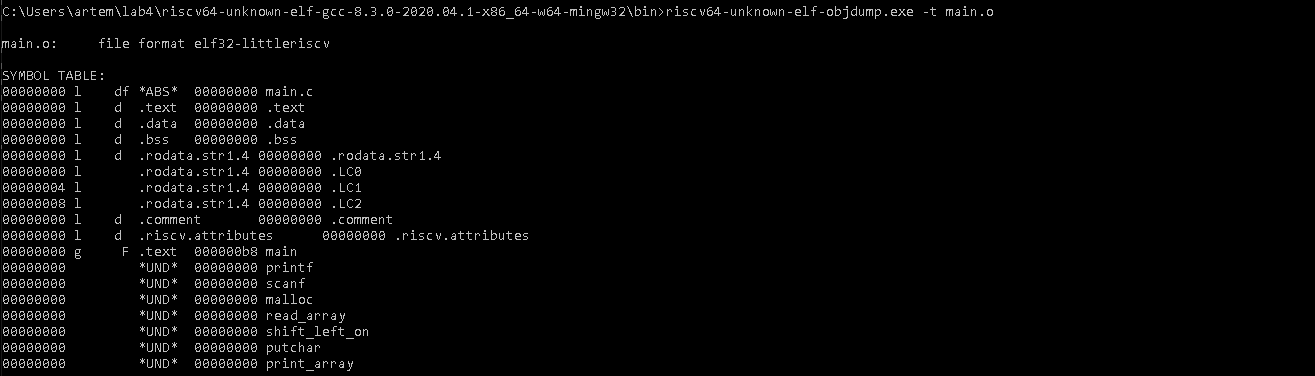
Листинг 2.5. Заголовки секций файла main.o

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



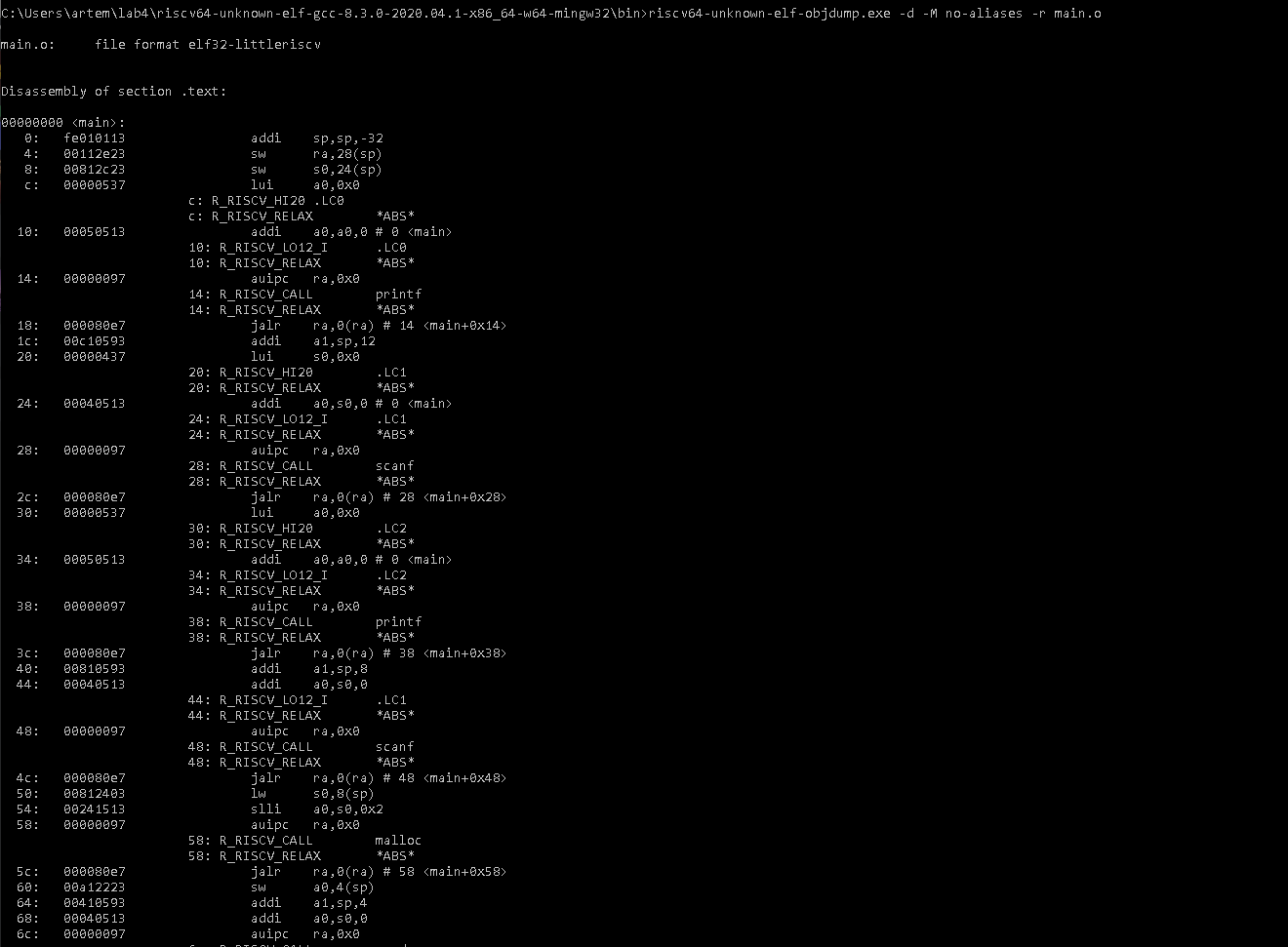
Листинг 2.6. Таблица символов файла main.o

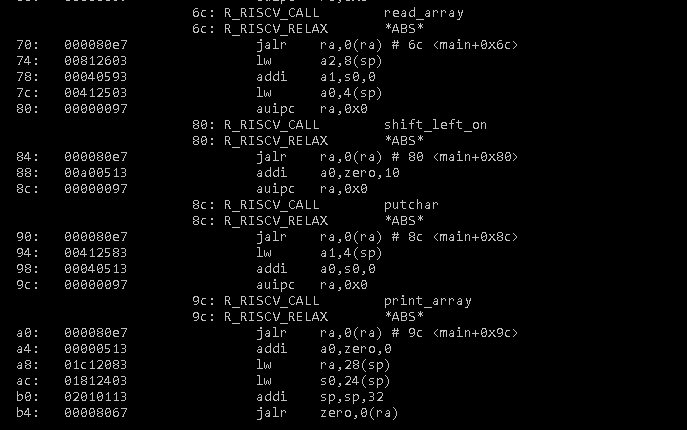
riscv64-unknown-elf-objdump.exe –t main.o



Листинг 2.7. Таблица перемещений файла main.o

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

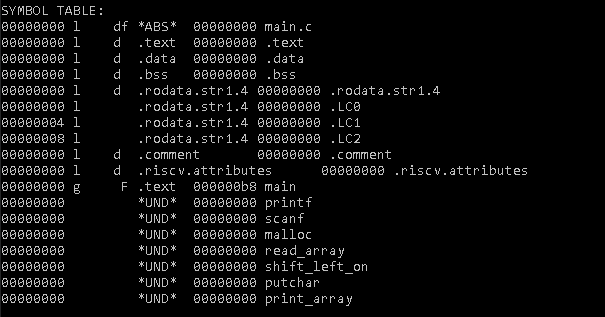




Листинг 2.8. Заголовки секций файла sdvig.o



Листинг 2.9. Таблица символов файла sdvig.o



Компановка

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



Листинг 2.10. Исполняемый файл a.out

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

a.out: file format elf64-littleriscv

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 <sdvig>

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 <sdvig>:

101aa: 4785 c.li a5,1

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

101ce: 00082603 lw a2,0(a6)

101d2: 0008879b addiw a5,a7,0

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

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

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

101ec: 1771 c.addi a4,-4

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

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

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

101f8: 8082 c.jr ra

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

Сделаем из sdvig.c статическую библиотеку sdvi, тестовую программу main.c оставим без изменений.

получим объектный файлsdvig.o

riscv64-unknown-elf-gcc.exe -O1 -c sdvig.c -o sdvi.o

из получившегося файла делаем библиотеку

riscv64-unknown-elf-ar.exe -rsc sdvi.a sdvig.o

собираем исполняющий файл

riscv64-unknown-elf-gcc.exe -O1 --save-temps main.c sdvi.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= sdvig.c

#Вызываемые приложения

AR = riscv64-unknown-elf-ar.exe

CC = riscv64-unknown-elf-gcc.exe

# Файл библиотеки

MYLIBNAME = sdvi.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)

# $^ = (sdvig.o)

$(MYLIBNAME): sdvi.o

$(AR) -rsc $@ $^

Листинг 3.3. Makefile для сборки исполняемого файла

# Цели

.PHONY: all clean

# Исходные файлы для сборки библиотеки

OBJS= main.c \

sdvi.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 выполнения данной лабораторной работы была разработана функция на языке C, реализующая заданную вариантом задания функциональность (Разработать программу для нахождения суммы всех элементов массива. Если сумма меньше 50 – увеличиваем значения всех элементов на 7.). Определение функции было помещено в отдельный исходный файл, также был оформлен заголовочный файл. Была разработана тестовая программа на языке C.

Были изучены особенности каждого этапа пошаговой сборки набора программ, а также инструменты, позволяющие выделить разработанные программы в статическую библиотеку и автоматизировать сборку этой библиотеки.

Проанализированы ход сборки библиотеки и программы, созданные файлы зависимостей.