

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

Лабораторная работа № 4

Дисциплина: Низкоуровневое программирование

Тема: Раздельная компиляция

Выполнил студент гр. 3530901/10005 _____ Кузичева П. Д.

Преподаватель _____ Коренев Д.А.

«30» ноября 2022 г.

Санкт-Петербург

2022

Раздельная компиляция

Оглавление

ФОРМУЛИРОВКА ЗАДАЧИ	2
ВАРИАНТ ЗАДАНИЯ	3
ХОД РЕШЕНИЯ	4
ТЕКСТ ПРОГРАММ, РЕАЛИЗУЮЩИХ ОПРЕДЕЛЕННУЮ ВАРИАНТОМ ЗАДАНИЯ ФУНКЦИОНАЛЬНОСТЬ	4
СБОРКА ПРОГРАММ «ПО ШАГАМ», АНАЛИЗ ПРОМЕЖУТОЧНЫХ И РЕЗУЛЬТИРУЮЩИХ ФАЙЛОВ	5
<i>Перепроцессирование</i>	5
<i>Компиляция</i>	7
<i>Ассемблирование</i>	9
<i>Компоновка</i>	13
ФОРМИРОВАНИЕ СТАТИЧЕСКОЙ БИБЛИОТЕКИ, РАЗРАБОТКА MAKE-ФАЙЛОВ ДЛЯ СБОРКИ БИБЛИОТЕКИ	15
ВЫВОДЫ	18

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 С.

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 исполняемом файле.

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

Вариант задания

Разработать на языке С, реализующую реверс массива, который осуществляется путем перестановки элементов массива. Сначала местами меняются первый и последний, потом второй и предпоследний и так далее. Если число элементов массива нечетное, то элемент посередине не меняет своего расположения.

Ход решения

Текст программ, реализующих определенную вариантом задания функциональность

Листинг 1. Программа “reverse.c”

```
1  #include "stdlib.h"
2  #include "reverse.h"
3  void reverse(unsigned *array, size_t array_length) {
4      unsigned t = 0;
5      size_t i;
6      for (i = 0; i < array_length / 2; i += 1) {
7          t = array[i];
8          array[i] = array [array_length - i - 1];
9          array[array_length - i - 1] = t;
10     }
11
12 }
```

Листинг 2. Заголовочный файл “reverse.h”

```
1  #include "stdlib.h"
2  #ifndef REVERSE_H
3  #define REVERSE_H
4  void reverse(unsigned *array, size_t array_length);
5  #endif
```

Листинг3. Программа “main.c”

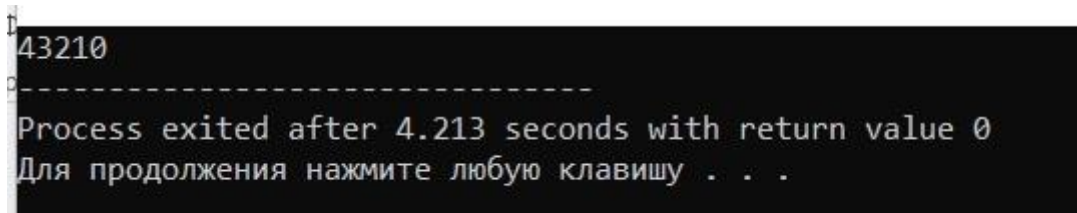
```
1  #include <stdio.h>
2  #include "reverse.h"
3
4  int main(void) {
5      unsigned array[] = {0,1,2,3,4};
6      reverse(array, sizeof(array)/sizeof(array[0]));
7      size_t i;
8      for (i = 0; i < sizeof(array)/sizeof(array[0]); i++) {
9          printf("%d", array[i]);
10     }
11     return 0;
12 }
```

В файле “reverse.c” реализована функция reverse(), в которую передаётся массив и его длина. Функция осуществляет перестановку элементов массива.

Заголовочный файл “reverse.h” содержит в себе определение функции reverse(). В дальнейшем для использования этой функции в другой программе,

необходимо организовывать подключение этого заголовочного файла, а также компиляцию исходного файла “reverse.c” вместе с использующей ее программой.

Вывод программы:

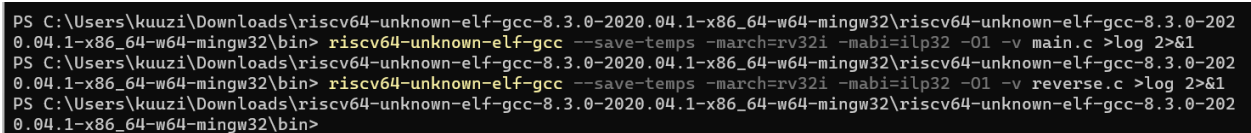


```
43210
-----
Process exited after 4.213 seconds with return value 0
Для продолжения нажмите любую клавишу . . .
```

Сборка программ «по шагам», анализ промежуточных и результатирующих файлов

Перепроцессирование

Начнем сборку созданных программ на языке С по шагам. Первым шагом является препроцессирование файлов исходного текста “reverse.c” и “main.c” в файлы “reverse.i” и “main.i”:



```
PS C:\Users\kuuzi\Downloads\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\bin> riscv64-unknown-elf-gcc --save-temps -march=rv32i -mabi=ilp32 -O1 -v main.c >log 2>&1
PS C:\Users\kuuzi\Downloads\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\bin> riscv64-unknown-elf-gcc --save-temps -march=rv32i -mabi=ilp32 -O1 -v reverse.c >log 2>&1
PS C:\Users\kuuzi\Downloads\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\bin>
```

Драйвер компилятора gcc– riscv64-unknown-elf-gcc– запускается с параметрами командной строки “-march=rv32i -mabi=ilp32”, указывающих что целевым является процессор с базовой архитектурой системы команд RV32I;-O1 – указание выполнять простые оптимизации генерируемого кода; -E – указание остановить процесс сборки после препроцессирования.

По причине того, что main.c содержит заголовочный файл стандартной библиотеки языка С stdio.h, результат препроцессирования этого файла имеет достаточно много добавочных строк.

Результаты работы препроцессора мало отличаются от исходных версий программ:

Листинг 4. Файл “reverse.i”

```
void * aligned_alloc(size_t, size_t) __attribute__((__malloc__)) __attribute__((__alloc_align__(1)))
__attribute__((__alloc_size__(2))) __attribute__((__warn_unused_result__));
int at_quick_exit(void (*)(void));
_Noreturn void
quick_exit(int);

# 2 "reverse.c" 2
# 1 "reverse.h" 1

# 4 "reverse.h"
void reverse(unsigned *array, size_t array_length);
# 3 "reverse.c" 2
void reverse(unsigned *array, size_t array_length) {
    unsigned t = 0;
    size_t i;
    for (i = 0; i < array_length / 2; i +=1) {
        t = array[i];
        array[i] = array [array_length - i - 1];
        array[array_length - i - 1] = t;
    }
}
```

Листинг5. Файл “main.i”

```
# 3 "main.c" 2
# 1 "reverse.h" 1

# 4 "reverse.h"
void reverse(unsigned *array, size_t array_length);
# 4 "main.c" 2

int main(void) {
    unsigned array[] = {0,1,2,3,4};
    reverse(array, sizeof(array)/sizeof(array[0]));
    size_t i;
    for (i = 0; i < sizeof(array)/sizeof(array[0]); i++) {
        printf("%d", array[i]);
    }
    return 0;
}
```

Появившиеся нестандартные директивы, начинающиеся с символа “#”, используются для передачи информации об исходном тексте из препроцессора в компилятор.

Компиляция

Следующим шагом является компиляция файлов “reverse.i” и “main.i” в код на языке ассемблера “reverse.s” и “main.s”:

Драйвер компилятора riscv64-unknown-elf-gcc запускается с параметрами командной строки “-march=rv32i -mabi=ilp32”, указывающих что целевым является процессор с базовой архитектурой системы команд RV32I; -O1 – указание выполнять простые оптимизации генерируемого кода; -S – указание остановить процесс сборки после компиляции (без запуска ассемблера).

Проанализируем получившийся код на языке ассемблера:

Листинг 6. Файл “reverse.s”

```
.file "reverse.c"
.option nopic
.attribute arch, "rv32i2p0"
.attribute unaligned_access, 0
.attribute stack_align, 16
.text
.align      2
.globl      reverse
.type reverse, @function
reverse:
    srli a6,a1,1
    li    a5,1
    bleu  a1,a5,.L1
    mv    a5,a0
    slli  a1,a1,2
    add   a0,a0,a1
    li    a4,0
.L3:
    lw    a3,0(a5)
    lw    a2,-4(a0)
    sw    a2,0(a5)
    sw    a3,-4(a0)
    addi  a4,a4,1
    addi  a5,a5,4
    addi  a0,a0,-4
    bltu  a4,a6,.L3
.L1:
    ret
```

```
.size reverse, .-reverse
.ident      "GCC: (SiFive GCC 8.3.0-2020.04.1) 8.3.0"
```

Листинг 7. Файл "main.s"

```
    .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,-48
    sw      ra,44(sp)
    sw      s0,40(sp)
    sw      s1,36(sp)
    sw      s2,32(sp)
    lui     a5,%hi(.LANCHOR0)
    addi    a5,a5,%lo(.LANCHOR0)
    lw      a1,0(a5)
    lw      a2,4(a5)
    lw      a3,8(a5)
    lw      a4,12(a5)
    lw      a5,16(a5)
    sw      a1,12(sp)
    sw      a2,16(sp)
    sw      a3,20(sp)
    sw      a4,24(sp)
    sw      a5,28(sp)
    li      a1,5
    addi    a0,sp,12
    call    reverse
    addi    s0,sp,12
    addi    s2,sp,32
    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
    lw      ra,44(sp)
    lw      s0,40(sp)
```



```

        lw    s1,36(sp)
        lw    s2,32(sp)
        addi  sp,sp,48
        jr    ra
        .size main, .-main
        .section .rodata
        .align 2
        .set .LANCHOR0, . + 0
.LC0:
        .word 0
        .word 1
        .word 2
        .word 3
        .word 4
        .section .rodata.str1.4,"aMS",@progbits,1
        .align 2
.LC1:
        .string "%d"
        .ident  "GCC: (SiFive GCC 8.3.0-2020.04.1) 8.3.0"

```

Наибольший интерес представляет файл `main.s`, так как в нем можно заметить обращение к подпрограмме `reverse` (значение регистра `ra`, содержащее адрес возврата из `main`, сохраняется на время вызова в стеке).

Ассемблирование

Для ассемблирования выполним следующие команды:

```

PS C:\Users\kuuzi\Downloads\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\bin> riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -c reverse.s -o reverse.o
PS C:\Users\kuuzi\Downloads\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\bin> riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -c main.s -o main.o

```

Драйвер компилятора `riscv64-unknown-elf-gcc` запускается с параметрами командной строки “`-march=rv32i -mabi=ilp32`”, указывающих что целевым является процессор с базовой архитектурой системы команд RV32I; `-c` – указание остановить процесс сборки после ассемблирования.

Объектный файл не является текстовым, для изучения его содержимого используем утилиту `objdump`:

```

PS C:\Users\kuuzi\Downloads\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\bin> riscv64-unknown-elf-objdump -t reverse.o main.o

```

Листинг 8. Таблица символов файла main.o и reverse.o

SYMBOL TABLE:

```
00000000 1  df *ABS* 00000000 reverse.c
00000000 1  d  .text 00000000 .text
00000000 1  d  .data 00000000 .data
00000000 1  d  .bss 00000000 .bss
0000003c 1  .text 00000000 .L1
0000001c 1  .text 00000000 .L3
00000000 1  d  .comment 00000000 .comment
00000000 1  d  .riscv.attributes 00000000 .riscv.attributes
00000000 g  F .text 00000040 reverse
```

main.o: file format elf32-littleriscv

SYMBOL TABLE:

```
00000000 1  df *ABS* 00000000 main.c
00000000 1  d  .text 00000000 .text
00000000 1  d  .data 00000000 .data
00000000 1  d  .bss 00000000 .bss
00000000 1  d  .rodata 00000000 .rodata
00000000 1  .rodata 00000000 .LANCHOR0
00000000 1  d  .rodata.str1.4 00000000 .rodata.str1.4
00000000 1  .rodata.str1.4 00000000 .LC1
00000060 1  .text 00000000 .L2
00000000 1  d  .comment 00000000 .comment
00000000 1  d  .riscv.attributes 00000000 .riscv.attributes
00000000 g  F .text 00000094 main
00000000  *UND* 00000000 reverse
00000000  *UND* 00000000 printf
```

В таблице символов main.o имеется запись: символ “reverse” типа *UND*. Эта запись означает, что символ “reverse” использовался в ассемблерном коде, из которого был получен данный объектный файл, но не был определен; ассемблер сделал вывод о том, что символ должен быть определен где-то еще,

и отразил это в таблице символов. То же самое относится и к символу “printf”.

```
PS C:\Users\kuuzi\Downloads\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\bin> riscv64-unknown-elf-objdump.exe -h main.o
```

Листинг 9. Заголовки секций файла main.o

main.o: file format elf32-littleriscv

Sections:

Idx	Name	Size	VMA	LMA	File off	Algn
0	.text	00000094	00000000	00000000	00000034	2**2
	CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE					
1	.data	00000000	00000000	00000000	000000c8	2**0
	CONTENTS, ALLOC, LOAD, DATA					
2	.bss	00000000	00000000	00000000	000000c8	2**0
	ALLOC					
3	.rodata	00000014	00000000	00000000	000000c8	2**2
	CONTENTS, ALLOC, LOAD, READONLY, DATA					
4	.rodata.str1.4	00000003	00000000	00000000	000000dc	2**2
	CONTENTS, ALLOC, LOAD, READONLY, DATA					
5	.comment	00000029	00000000	00000000	000000df	2**0
	CONTENTS, READONLY					
6	.riscv.attributes	0000001c	00000000	00000000	00000108	2**0
	CONTENTS, READONLY					

Листинг 10. Таблица перемещений файла main.o

main.o: file format elf32-littleriscv

Disassembly of section .text:

00000000 <main>:

0:	fd010113	addi	sp,sp,-48
4:	02112623	sw	ra,44(sp)
8:	02812423	sw	s0,40(sp)
c:	02912223	sw	s1,36(sp)
10:	03212023	sw	s2,32(sp)
14:	000007b7	lui	a5,0x0
14:	R_RISCV_HI20		.LANCHOR0

```

14: R_RISCV_RELAX    *ABS*
18: 00078793          addi  a5,a5,0 # 0 <main>
18: R_RISCV_LO12_I    .LANCHOR0
18: R_RISCV_RELAX    *ABS*
1c: 0007a583          lw    a1,0(a5)
20: 0047a603          lw    a2,4(a5)
24: 0087a683          lw    a3,8(a5)
28: 00c7a703          lw    a4,12(a5)
2c: 0107a783          lw    a5,16(a5)
30: 00b12623          sw    a1,12(sp)
34: 00c12823          sw    a2,16(sp)
38: 00d12a23          sw    a3,20(sp)
3c: 00e12c23          sw    a4,24(sp)
40: 00f12e23          sw    a5,28(sp)
44: 00500593          addi  a1,zero,5
48: 00c10513          addi  a0,sp,12
4c: 00000097          auipc ra,0x0
4c: R_RISCV_CALL      reverse
4c: R_RISCV_RELAX    *ABS*
50: 000080e7          jalr  ra,0(ra) # 4c <main+0x4c>
54: 00c10413          addi  s0,sp,12
58: 02010913          addi  s2,sp,32
5c: 000004b7          lui   s1,0x0
5c: R_RISCV_HI20      .LC1
5c: R_RISCV_RELAX    *ABS*

```

00000060 <.L2>:

```

60: 00042583          lw    a1,0(s0)
64: 00048513          addi  a0,s1,0 # 0 <main>
64: R_RISCV_LO12_I    .LC1
64: R_RISCV_RELAX    *ABS*
68: 00000097          auipc ra,0x0
68: R_RISCV_CALL      printf
68: R_RISCV_RELAX    *ABS*
6c: 000080e7          jalr  ra,0(ra) # 68 <.L2+0x8>
70: 00440413          addi  s0,s0,4
74: ff2416e3          bne   s0,s2,60 <.L2>
74: R_RISCV_BRANCH     .L2
78: 00000513          addi  a0,zero,0

```

```

7c: 02c12083      lw    ra,44(sp)
80: 02812403      lw    s0,40(sp)
84: 02412483      lw    s1,36(sp)
88: 02012903      lw    s2,32(sp)
8c: 03010113      addi  sp,sp,48
90: 00008067      jalr  zero,0(ra)

```

Листинг 10. Заголовки секций файла reverse.o

```

PS C:\Users\kuuzi\Downloads\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\bin> riscv64-unknown-elf-objdump.exe -h reverse.o

```

reverse.o: file format elf32-littleriscv

Sections:

Idx	Name	Size	VMA	LMA	File off	Algn
0	.text	00000040	00000000	00000000	00000034	2**2
	CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE					
1	.data	00000000	00000000	00000000	00000074	2**0
	CONTENTS, ALLOC, LOAD, DATA					
2	.bss	00000000	00000000	00000000	00000074	2**0
	ALLOC					
3	.comment	00000029	00000000	00000000	00000074	2**0
	CONTENTS, READONLY					
4	.riscv.attributes	0000001c	00000000	00000000	0000009d	2**0
	CONTENTS, READONLY					

Компоновка

Следующим шагом является компоновка и формирование исполняемых фалов программ:

```

PS C:\Users\kuuzi\Downloads\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\bin> riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 reverse.o main.o -o main.out

```

Посмотрим на результат компоновки, выполнив следующую команду:

```

PS C:\Users\kuuzi\Downloads\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\bin> riscv64-unknown-elf-objdump -j .text -d -M no-aliases main.out >main.ds

```

В файле main.ds интересны только некоторые фрагменты, описывающие разработанные функции:

00010144 <reverse>:

```
10144: 0015d813
10148: 00100793
1014c: 02b7fa63
10150: 00050793
10154: 00259593
10158: 00b50533
1015c: 00000713
10160: 0007a683
10164: ffc52603
10168: 00c7a023
1016c: fed52e23
10170: 00170713
10174: 00478793
10178: ffc50513
1017c: ff0762e3
10180: 00008067
```

```
srli a6,a1,0x1
addi a5,zero,1
bgeu a5,a1,10180 <reverse+0x3c>
addi a5,a0,0
slli a1,a1,0x2
add a0,a0,a1
addi a4,zero,0
lw a3,0(a5)
lw a2,-4(a0)
sw a2,0(a5)
sw a3,-4(a0)
addi a4,a4,1
addi a5,a5,4
addi a0,a0,-4
bltu a4,a6,10160 <reverse+0x1c>
jalr zero,0(ra)
```

00010184 <main>:

```
10184: fd010113
10188: 02112623
1018c: 02812423
10190: 02912223
10194: 03212023
10198: 000257b7
1019c: 93078793
101a0: 0007a583
101a4: 0047a603
101a8: 0087a683
101ac: 00c7a703
101b0: 0107a783
101b4: 00b12623
101b8: 00c12823
101bc: 00d12a23
101c0: 00e12c23
101c4: 00f12e23
101c8: 00500593
101cc: 00c10513
101d0: f75ff0ef
101d4: 00c10413
101d8: 02010913
101dc: 000254b7
```

```
addi sp,sp,-48
sw ra,44(sp)
sw s0,40(sp)
sw s1,36(sp)
sw s2,32(sp)
lui a5,0x25
addi a5,a5,-1744 # 24930 <__clzsi2+0x50>
lw a1,0(a5)
lw a2,4(a5)
lw a3,8(a5)
lw a4,12(a5)
lw a5,16(a5)
sw a1,12(sp)
sw a2,16(sp)
sw a3,20(sp)
sw a4,24(sp)
sw a5,28(sp)
addi a1,zero,5
addi a0,sp,12
jal ra,10144 <reverse>
addi s0,sp,12
addi s2,sp,32
lui s1,0x25
```

101e0: 00042583	lw a1,0(s0)
101e4: 94448513	addi a0,s1,-1724 # 24944 <__clzsi2+0x64>
101e8: 284000ef	jal ra,1046c <printf>
101ec: 00440413	addi s0,s0,4
101f0: ff2418e3	bne s0,s2,101e0 <main+0x5c>
101f4: 00000513	addi a0,zero,0
101f8: 02c12083	lw ra,44(sp)
101fc: 02812403	lw s0,40(sp)
10200: 02412483	lw s1,36(sp)
10204: 02012903	lw s2,32(sp)
10208: 03010113	addi sp,sp,48
1020c: 00008067	jalr zero,0(ra)

Формирование статической библиотеки, разработка make-файлов для сборки библиотеки

Статическая библиотека (staticlibrary) является, по сути, архивом (набором, коллекцией) объектных файлов, среди которых компоновщик выбирает «полезные» для данной программы: объектный файл считается «полезным», если в нем определяется еще не разрешенный компоновщиком символ.

Сделаем из reverse.c статическую библиотеку revlib, тестовую программу main.c оставим без изменений.

Для создания статической библиотеки получим объектный файл reverse.o.

```
PS C:\Users\kuuzi\Downloads\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\bin> riscv64-unknown-elf-gcc.exe -O1 -c reverse.c -o reverse.o
```

Сделаем из получившегося файла библиотеку следующей командой:

```
PS C:\Users\kuuzi\Downloads\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\bin> riscv64-unknown-elf-ar.exe -rsc revlib.a reverse.o
```

Используя получившуюся библиотеку, соберем исполняемый файл программы следующей командой:

```
PS C:\Users\kuuzi\Downloads\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\bin> riscv64-unknown-elf-gcc.exe -O1 --save-temps main.c revlib.a
```

Листинг 11. Таблица символов исполняемого файла (фрагмент)

```
PS C:\Users\kuuzi\Downloads\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\bin> riscv64-unknown-elf-objdump.exe -t a.out

a.out:      file format elf64-littleriscv

SYMBOL TABLE:
00000000000100b0 l d .text 0000000000000000 .text
000000000001c0f0 l d .rodata 0000000000000000 .rodata
000000000001d000 l d .eh_frame 0000000000000000 .eh_frame
000000000001d008 l d .init_array 0000000000000000 .init_array
000000000001d018 l d .fini_array 0000000000000000 .fini_array
000000000001d020 l d .data 0000000000000000 .data
000000000001e120 l d .sdata 0000000000000000 .sdata
000000000001e178 l d .sbss 0000000000000000 .sbss
000000000001e1a0 l d .bss 0000000000000000 .bss
```

```
0000000000015534 g F .text 0000000000000012 _Bfree
```

```
0000000000010158 g F .text 000000000000004a main
```

```
00000000000101a2 g F .text 000000000000002a reverse
```

Процесс выполнения команд выше можно заменить make-файлами, которые произведут создание библиотеки и сборку программы.

Листинг 12. Make-файл для создания статической библиотеки

```
# "Фиктивные" цели
.PHONY: all clean

# Исходные файлы, необходимые для сборки библиотеки
OBJS= reverse.c

#Вызываемые приложения
AR = riscv64-unknown-elf-ar.exe
CC = riscv64-unknown-elf-gcc.exe

# Файл библиотеки
MYLIBNAME = revlib.a

# Параметры компиляции
CFLAGS= -O1

# Включаемые файлы следует искать в текущем каталоге
INCLUDES+= -I .

# Make должна искать файлы *.h и *.c в текущей директории
vpath %.h .
vpath %.c .

# Построение объектного файла из исходного текста
# $< = %.c
# $@ = %.o
%.o: %.c
    $(CC) -MD $(CFLAGS) $(INCLUDES) -c $< -o $@

# Чтобы достичь цели "all", требуется построить библиотеку
all: $(MYLIBNAME)

# $^ = (reverse.o)
$(MYLIBNAME): reverse.o
    $(AR) -rsc $@ $^
```


Листинг 12. Make-файл для сборки исполняемого файла

```
# "Фиктивные" цели
.PHONY: all clean

# Файлы для сборки исполнимого файла
OBJS= main.c \
      revlib.a

#Вызываемые приложения
CC = riscv64-unknown-elf-gcc.exe

# Параметры компиляции
CFLAGS= -O1 --save-temps

# Включаемые файлы следует искать в текущем каталоге
INCLUDES+= -I .

# Make должна искать файлы *.c и *.a в текущей директории
vpath %.c .
vpath %.a .

# Чтобы достичь цели "all", требуется собрать исполнимый файл
all: a.out

# Сборка исполнимого файла и удаление мусора
a.out: $(OBJS)
      $(CC) $(CFLAGS) $(INCLUDES) $^
      del *.o *.i *.s *.d
```

Запуск make – файлов осуществляется с использованием утилиты mingw32-make.exe.

```
PS C:\Users\kuuzi\IdeaProjects\3sem\comp> mingw32-make.exe -f Makefile.win Makefile1.win
```

Выводы

В ходе выполнения лабораторной работы были закреплены знания языка C, ассемблера RISC-V, получены навыки работы с препроцессором, компилятором, ассемблером и компоновщиком пакета GCC и драйвером компилятора riscv64-unknown-elf-gcc. Были изучены особенности каждого этапа пошаговой сборки набора программ, а также инструменты, позволяющие выделить разработанные программы в статическую библиотеку и автоматизировать сборку этой библиотеки.

Была реализована поставленная задача – «реверс массива», а затем проведена проверка правильности перевода программы решения этой задачи в набор инструкций, выполняемых процессором.