

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  
Туфаногуллари Севдат

\_\_\_\_\_

Преподаватель  
\_\_Коренев Д. А.

14, 12, 2022 г.

Санкт-Петербург

2022

## **Оглавление**

Техническое задание

1. Формулировка задачи

2. Вариант задания

    Препроцессирование

    Компиляция

    Ассемблирование

    Компоновка

3. Создание статической библиотеки и make-файлов

Вывод

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

### 2. Вариант задания

Реализовать нахождение наибольшего общего делителя (НОД) для массива чисел.

#### Ход решения

##### Листинг 2.1. Заголовочный файл fun.h

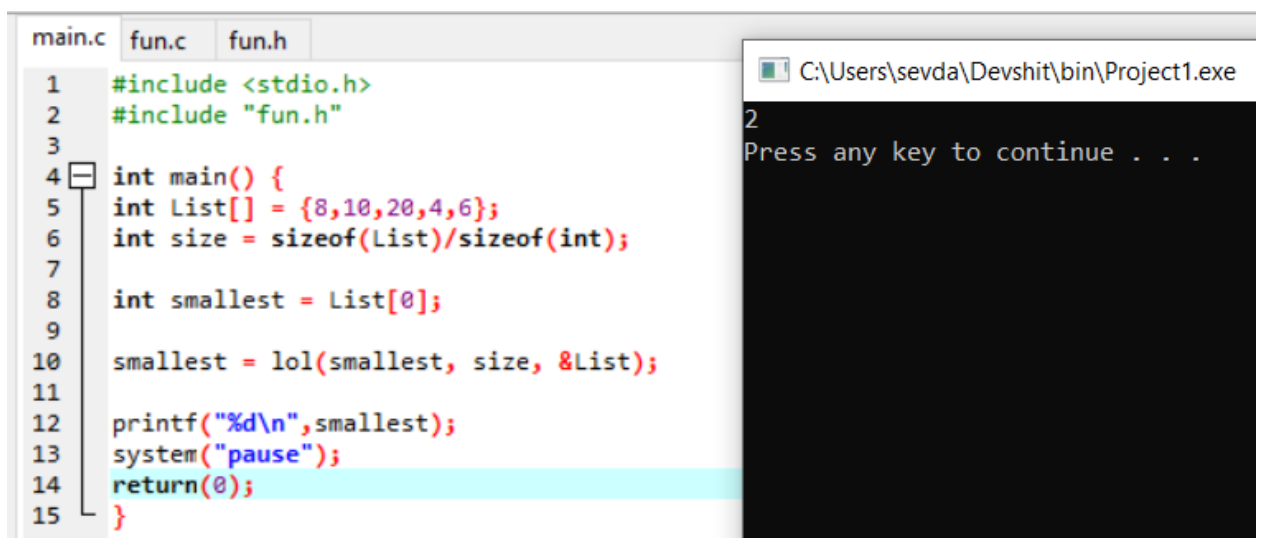
```
#ifndef FUN_H
#define FUN_H
int lol(int smallest, int size, int* List);
#endif
```

##### Листинг 2.2. Основной файл fun.c

```
#include <stdio.h>
#include "fun.h"
int main() {
    int List[] = {8,10,20,4,6};
    int size = sizeof(List)/sizeof(int);
    int smallest = List[0];
    smallest = lol(smallest, size, &List);
    printf("%d\n", smallest);
    system("pause");
    return(0);
}
```

### Листинг 2.3. Тестовая программа main.c

```
#include <stdio.h>
#include "fun.h"
int main() {
    int List[] = {8,10,20,4,6};
    int size = sizeof(List)/sizeof(int);
    int smallest = List[0];
    smallest = lol(smallest, size, &List);
    printf("%d\n",smallest);
    system("pause");
    return(0);
}
```



The screenshot displays a code editor with three tabs: 'main.c', 'fun.c', and 'fun.h'. The 'main.c' tab is active, showing the following code:

```
1  #include <stdio.h>
2  #include "fun.h"
3
4  int main() {
5      int List[] = {8,10,20,4,6};
6      int size = sizeof(List)/sizeof(int);
7
8      int smallest = List[0];
9
10     smallest = lol(smallest, size, &List);
11
12     printf("%d\n",smallest);
13     system("pause");
14     return(0);
15 }
```

Below the code editor, a console window titled 'C:\Users\sevda\Devshit\bin\Project1.exe' is open. It displays the output of the program:

```
2
Press any key to continue . . .
```

## Сборка программы по шагам

Чтобы начать этот отчет, мы сначала должны записать коды в формате блокнота, которые являются: (main.i), (fun.i), (main.c) и (fun.c)

```
riscv64-unknown-elf-gcc.exe -march-rv32i -mabi - ilp32 -01 -E fun.c -o fun.i  
riscv64-unknown-elf-gcc.exe -march-rv32i -mabi - ilp32 -01 -E main.c - main.i
```

Параметры командной строки (-march=rv32i -mabi=ilp32) указывают, что это драйвер компилятора gcc– riscv64-unknown-elf-gcc с архитектурой командной системы RV32L. Система команд O1 предназначена для оптимизации сгенерированного кода, а -E - для завершения процесса после завершения.

### Листинг 2.4. Файл main.i

```
# 2 "main.c" 2  
# 1 "fun.h" 1  
  
# 3 "fun.h"  
int lol(int smallest, int size,int* List);  
# 3 "main.c" 2  
  
int main() {  
  
int List[] = {8,10,20,4,6};  
  
int size = sizeof(List)/sizeof(int);  
  
int smallest = List[0];  
  
smallest = lol(smallest, size, &List);  
  
printf("%d\n",smallest);  
system("pause");  
return(0);  
}
```

## Листинг 2.5 Файл fun.i

```
# 1 "fun.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "fun.c"
# 1 "fun.h" 1
int lol(int smallest, int size,int* List);
# 2 "fun.c" 2

int lol(int smallest, int size,int* List){
int count = 0;
for (count; count < size; count++) {
if (smallest > List[count]) smallest = List[count];
}
int sizeCopy = -(size + 1);

while (sizeCopy < 0) {
sizeCopy += 1;
if (List[size + sizeCopy] % smallest != 0) {
smallest--;
sizeCopy = -(size + 1);
}
}
return smallest;
}
```

Символ (#), означающий нестандартную директиву, используется для передачи информации компилятору.

## Компиляция

После получения (main.i и fun.i) код преобразуется в язык ассемблера, который станет (main.s и fun.s) и используя кодов командной строки:

**riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -O1 -S main.i -o main.s**

**riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -O1 -S fun.i -o fun.s**

Параметры командной строки (-march=rv32i -mabi=ilp32) указывают, что это драйвер компилятора gcc- riscv64-unknown-elf-gcc с архитектурой командной системы RV32L. Система команд O1 предназначена для оптимизации сгенерированного кода, а -S - для завершения процесса после завершения.

```
main.c: In function 'main':
main.c:12:32: warning: passing argument 3 of 'lol' from incompatible pointer type [-Wincompatible-pointer-types]
   12 | smallest = lol(smallest, size, &List);
      |                               ^~~~~~
      |                               |
      |                               int (*)[5]
In file included from main.c:2:
fun.h:3:37: note: expected 'int *' but argument is of type 'int (*)[5]'
     3 | int lol(int smallest, int size, int* List);
      |                               ~~~~~^~~~~
main.c:15:1: warning: implicit declaration of function 'system' [-Wimplicit-function-declaration]
   15 | system("pause");
      | ^~~~~~
```

## Листинг 2.6. Файл main.s

```
.file "main.c"
.option nopic
.attribute arch, "rv32i2p0"
.attribute unaligned_access, 0
.attribute stack_align, 16
.text
.section .rodata.str1.4,"aMS",@progbits,1
.align 2
.LC1:
.string "%d\n"
.align 2
.LC2:
.string "pause"
.text
.align 2
```

```

.globl main
.type main, @function
main:
    addi sp,sp,-48
    sw ra,44(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)
    addi a2,sp,12
    li a1,5
    li a0,8
    call lol
    mv a1,a0
    lui a0,%hi(.LC1)
    addi a0,a0,%lo(.LC1)
    call printf
    lui a0,%hi(.LC2)
    addi a0,a0,%lo(.LC2)
    call system
    li a0,0
    lw ra,44(sp)
    addi sp,sp,48
    jr ra
.size main,.-main
.section .rodata
.align 2
.set .LANCHOR0, . + 0
.LC0:
    .word 8
    .word 10
    .word 20
    .word 4
    .word 6
.ident "GCC: (SiFive GCC-Metal 10.2.0-2020.12.8) 10.2.0"

```



## Ассемблирование

Следующим шагом является ассемблирование файлов “fun.s” и “main.s” в объектные файлы “fun.o” и “main.o”:

**riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -c main.s -o main.o**

**riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -c fun.s -o fun.o**

Параметры командной строки (-march=rv32i -mabi=ilp32) указывают, что это драйвер компилятора gcc– riscv64-unknown-elf-gcc с архитектурой командной системы RV32L. Система команд O1 предназначена для оптимизации сгенерированного кода, а -c - для завершения процесса после завершения.

### Листинг 2.7 Файл fun.s

```
.file "fun.c"
.option nopic
.attribute arch, "rv32i2p0"
.attribute unaligned_access, 0
.attribute stack_align, 16
.text
.globl __modsi3
.align 2
.globl lol
.type lol, @function
lol:
    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,a0
    mv s4,a2
    ble a1,zero,.L2
    mv a5,a2
    slli a3,a1,2
    add a3,a3,a2
    j .L4
.L3:
    addi a5,a5,4
    beq a5,a3,.L2
.L4:
    lw a4,0(a5)
    ble s2,a4,.L3
    mv s2,a4
```

```

    j    .L3
.L2:
    not  s3,a1
.L5:
    mv   s1,s4
    mv   s0,s3
.L6:
    bge  s0,zero,.L12
    addi s0,s0,1
    mv   a1,s2
    lw   a0,0(s1)
    call __modsi3
    addi s1,s1,4
    beq  a0,zero,.L6
    addi s2,s2,-1
    j    .L5
.L12:
    mv   a0,s2
    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
.size   lol,.-lol
.ident  "GCC: (SiFive GCC-Metal 10.2.0-2020.12.8) 10.2.0"

```

## Листинг 2.8 Файл main.o и main.o

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

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

```
main.o:      file format elf32-littleriscv
architecture: riscv:rv32, flags 0x00000011:
HAS_RELOC, HAS_SYMS
start address 0x00000000
```

**riscv64-unknown-elf-objdump.exe -h fun.o**

```
fun.o:       file format elf32-littleriscv
architecture: riscv:rv32, flags 0x00000011:
HAS_RELOC, HAS_SYMS
start address 0x00000000
```

Оба файла содержат таблицу перемещений (в списке флагов есть флага HAS\_RELOC)

## Листинг 2.9 Заголовки секций файла main.o и fun.o

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

```
main.o:      file format elf32-littleriscv

Sections:
Idx Name          Size      VMA           LMA           File off  Algn
  0 .text          00000080  00000000  00000000  00000034  2**2
    CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data          00000000  00000000  00000000  000000b4  2**0
    CONTENTS, ALLOC, LOAD, DATA
  2 .bss           00000000  00000000  00000000  000000b4  2**0
    ALLOC
  3 .rodata        00000014  00000000  00000000  000000b4  2**2
    CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .rodata.str1.4 0000000a  00000000  00000000  000000c8  2**2
    CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .comment       00000029  00000000  00000000  000000d2  2**0
    CONTENTS, READONLY
  6 .riscv.attributes 0000001c  00000000  00000000  000000fb  2**0
    CONTENTS, READONLY
```

**riscv64-unknown-elf-objdump -h fun.o**

```
fun.o:      file format elf32-littleriscv

Sections:
Idx Name          Size      VMA           LMA           File off  Algn
  0 .text          000000a8  00000000  00000000  00000034  2**2
    CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data          00000000  00000000  00000000  000000dc  2**0
    CONTENTS, ALLOC, LOAD, DATA
  2 .bss           00000000  00000000  00000000  000000dc  2**0
    ALLOC
  3 .comment       00000031  00000000  00000000  000000dc  2**0
    CONTENTS, READONLY
  4 .riscv.attributes 0000001c  00000000  00000000  0000010d  2**0
    CONTENTS, READONLY
```

## Листинг 2.10 Таблица символов файла main.o и fun.o

Выведем таблицы символов файлов fun.o и main.o

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

```
main.o:      file format elf32-littleriscv

SYMBOL TABLE:
00000000 l      df *ABS*  00000000 main.c
00000000 l      d  .text 00000000 .text
00000000 l      d  .data 00000000 .data
00000000 l      d  .bss 00000000 .bss
00000000 l      d  .rodata.str1.4 00000000 .rodata.str1.4
00000000 l      d  .rodata      00000000 .rodata
00000000 l      .rodata      00000000 .LANCHOR0
00000000 l      .rodata.str1.4 00000000 .LC1
00000004 l      .rodata.str1.4 00000000 .LC2
00000000 l      d  .comment      00000000 .comment
00000000 l      d  .riscv.attributes 00000000 .riscv.attributes
00000000 g      F  .text 00000080 main
00000000      *UND*  00000000 lol
00000000      *UND*  00000000 printf
00000000      *UND*  00000000 system
```

riscv64-unknown-elf-objdump.exe -t fun.o

```
fun.o:      file format elf32-littleriscv

SYMBOL TABLE:
00000000 l      df *ABS*  00000000 fun.c
00000000 l      d  .text 00000000 .text
00000000 l      d  .data 00000000 .data
00000000 l      d  .bss 00000000 .bss
00000050 l      .text 00000000 .L2
00000040 l      .text 00000000 .L4
00000038 l      .text 00000000 .L3
00000084 l      .text 00000000 .L12
0000005c l      .text 00000000 .L6
00000054 l      .text 00000000 .L5
00000000 l      d  .comment 00000000 .comment
00000000 l      d  .riscv.attributes 00000000 .riscv.attributes
00000000      *UND*  00000000 __modsi3
00000000 g      F .text 000000a8 lol
```

## Листинг 2.11 файл main.o и fun.o

Получение выходных данных объектных файлов с помощью файлов, содержащих коды инструкций:

**riscv64-unknown-elf-objdump.exe -d -M no-aliases -r 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: 000007b7      lui     a5,0x0
                8: R_RISCV_HI20 .LANCHOR0
                8: R_RISCV_RELAX *ABS*
 c: 00078793      addi    a5,a5,0 # 0 <main>
                c: R_RISCV_LO12_I .LANCHOR0
                c: R_RISCV_RELAX *ABS*
10: 0007a583      lw      a1,0(a5)
14: 0047a603      lw      a2,4(a5)
18: 0087a683      lw      a3,8(a5)
1c: 00c7a703      lw      a4,12(a5)
20: 0107a783      lw      a5,16(a5)
24: 00b12623      sw      a1,12(sp)
28: 00c12823      sw      a2,16(sp)
2c: 00d12a23      sw      a3,20(sp)
30: 00e12c23      sw      a4,24(sp)
34: 00f12e23      sw      a5,28(sp)
38: 00c10613      addi    a2,sp,12
3c: 00500593      addi    a1,zero,5
40: 00800513      addi    a0,zero,8
44: 00000097      auipc   ra,0x0
                44: R_RISCV_CALL    lol
                44: R_RISCV_RELAX    *ABS*
48: 000080e7      jalr    ra,0(ra) # 44 <main+0x44>
4c: 00050593      addi    a1,a0,0
50: 00000537      lui     a0,0x0
                50: R_RISCV_HI20 .LC1
                50: R_RISCV_RELAX *ABS*
54: 00050513      addi    a0,a0,0 # 0 <main>
                54: R_RISCV_LO12_I .LC1
                54: R_RISCV_RELAX *ABS*
58: 00000097      auipc   ra,0x0
                58: R_RISCV_CALL    printf
                58: R_RISCV_RELAX    *ABS*
5c: 000080e7      jalr    ra,0(ra) # 58 <main+0x58>
60: 00000537      lui     a0,0x0
                60: R_RISCV_HI20 .LC2
                60: R_RISCV_RELAX *ABS*
```

```
5c: 000080e7      jalr    ra,0(ra) # 58 <main+0x58>
60: 00000537      lui     a0,0x0
                60: R_RISCV_HI20 .LC2
                60: R_RISCV_RELAX *ABS*
64: 00050513      addi    a0,a0,0 # 0 <main>
                64: R_RISCV_LO12_I .LC2
                64: R_RISCV_RELAX *ABS*
68: 00000097      auipc   ra,0x0
                68: R_RISCV_CALL    system
                68: R_RISCV_RELAX    *ABS*
6c: 000080e7      jalr    ra,0(ra) # 68 <main+0x68>
70: 00000513      addi    a0,zero,0
74: 02c12083      lw      ra,44(sp)
78: 03010113      addi    sp,sp,48
7c: 00008067      jalr    zero,0(ra)
```

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

```
fun.o:      file format elf32-littleriscv

Disassembly of section .text:

00000000 <lol>:
 0: fe010113      addi    sp,sp,-32
 4: 00112e23      sw      ra,28(sp)
 8: 00812c23      sw      s0,24(sp)
 c: 00912a23      sw      s1,20(sp)
10: 01212823      sw      s2,16(sp)
14: 01312623      sw      s3,12(sp)
18: 01412423      sw      s4,8(sp)
1c: 00050913      addi    s2,a0,0
20: 00060a13      addi    s4,a2,0
24: 02b05663      bge     zero,a1,50 <.L2>
    24: R_RISCV_BRANCH .L2
28: 00060793      addi    a5,a2,0
2c: 00259693      slli    a3,a1,0x2
30: 00c686b3      add     a3,a3,a2
34: 00c0006f      jal     zero,40 <.L4>
    34: R_RISCV_JAL .L4

00000038 <.L3>:
38: 00478793      addi    a5,a5,4
3c: 00d78a63      beq     a5,a3,50 <.L2>
    3c: R_RISCV_BRANCH .L2

00000040 <.L4>:
40: 0007a703      lw      a4,0(a5)
44: ff275ae3      bge     a4,s2,38 <.L3>
    44: R_RISCV_BRANCH .L3
48: 00070913      addi    s2,a4,0
4c: fedff06f      jal     zero,38 <.L3>
    4c: R_RISCV_JAL .L3

00000050 <.L2>:
50: fff5c993      xori    s3,a1,-1

00000054 <.L5>:
54: 000a0493      addi    s1,s4,0
58: 00098413      addi    s0,s3,0

0000005c <.L6>:
5c: 02045463      bge     s0,zero,84 <.L12>
    5c: R_RISCV_BRANCH .L12
60: 00140413      addi    s0,s0,1
```

```
60: 00140413      addi    s0,s0,1
64: 00090593      addi    a1,s2,0
68: 0004a503      lw      a0,0(s1)
6c: 00000097      auipc   ra,0x0
    6c: R_RISCV_CALL __modsi3
    6c: R_RISCV_RELAX *ABS*
70: 000080e7      jalr    ra,0(ra) # 6c <.L6+0x10>
74: 00448493      addi    s1,s1,4
78: fe0502e3      beq     a0,zero,5c <.L6>
    78: R_RISCV_BRANCH .L6
7c: fff90913      addi    s2,s2,-1
80: fd5ff06f      jal     zero,54 <.L5>
    80: R_RISCV_JAL .L5

00000084 <.L12>:
84: 00090513      addi    a0,s2,0
88: 01c12083      lw      ra,28(sp)
8c: 01812403      lw      s0,24(sp)
90: 01412483      lw      s1,20(sp)
94: 01012903      lw      s2,16(sp)
98: 00c12983      lw      s3,12(sp)
9c: 00812a03      lw      s4,8(sp)
a0: 02010113      addi    sp,sp,32
a4: 00008067      jalr    zero,0(ra)
```

Разобранный код похож на сгенерированный код, но в нем отсутствуют псевдо инструкции.



Размер секции.data данных равны нулю

**riscv64-unknown-elf-objdump.exe -d -M no-aliases -j .data main.o**

**riscv64-unknown-elf-objdump.exe -d -M no-aliases -j .data fun.o**

```
C:\Users\sevda\Devshit\bin>riscv64-unknown-elf-objdump.exe -d -M no-aliases -j .data fun.o
fun.o:      file format elf32-littleriscv

C:\Users\sevda\Devshit\bin>riscv64-unknown-elf-objdump.exe -d -M no-aliases -j .data main.o
main.o:     file format elf32-littleriscv
```

Размер файлов section .bss равен нулю

**riscv64-unknown-elf-objdump.exe -d -M no-aliases -j .bss main.o**

**riscv64-unknown-elf-objdump.exe -d -M no-aliases -j .bss fun.o**

```
C:\Users\sevda\Devshit\bin>riscv64-unknown-elf-objdump.exe -d -M no-aliases -j .bss main.o
main.o:     file format elf32-littleriscv

C:\Users\sevda\Devshit\bin>riscv64-unknown-elf-objdump.exe -d -M no-aliases -j .bss fun.o
fun.o:      file format elf32-littleriscv
```

## Листинг 2.12 таблицы перемещений main.o и fun.o

В этой таблице выводится сгенерированное перемещение объектных файлов.

**riscv64-unknown-elf-objdump.exe -r fun.o main.o**

```
fun.o:      file format elf32-littleriscv

RELOCATION RECORDS FOR [.text]:
OFFSET      TYPE          VALUE
00000006c R_RISCV_CALL      __modsi3
00000006c R_RISCV_RELAX      *ABS*
000000024 R_RISCV_BRANCH     .L2
000000034 R_RISCV_JAL        .L4
00000003c R_RISCV_BRANCH     .L2
000000044 R_RISCV_BRANCH     .L3
00000004c R_RISCV_JAL        .L3
00000005c R_RISCV_BRANCH     .L12
000000078 R_RISCV_BRANCH     .L6
000000080 R_RISCV_JAL        .L5
```

**riscv64-unknown-elf-objdump.exe -r fun.o main.o**

```
main.o:      file format elf32-littleriscv

RELOCATION RECORDS FOR [.text]:
OFFSET      TYPE             VALUE
00000008 R_RISCV_HI20             .LANCHOR0
00000008 R_RISCV_RELAX             *ABS*
0000000c R_RISCV_LO12_I           .LANCHOR0
0000000c R_RISCV_RELAX             *ABS*
00000044 R_RISCV_CALL              lol
00000044 R_RISCV_RELAX             *ABS*
00000050 R_RISCV_HI20             .LC1
00000050 R_RISCV_RELAX             *ABS*
00000054 R_RISCV_LO12_I           .LC1
00000054 R_RISCV_RELAX             *ABS*
00000058 R_RISCV_CALL              printf
00000058 R_RISCV_RELAX             *ABS*
00000060 R_RISCV_HI20             .LC2
00000060 R_RISCV_RELAX             *ABS*
00000064 R_RISCV_LO12_I           .LC2
00000064 R_RISCV_RELAX             *ABS*
00000068 R_RISCV_CALL              system
00000068 R_RISCV_RELAX             *ABS*
```

These tablets contain information about the (R\_RISCV\_JUL) and (R\_RISCV\_BRANCH) transistors.

## Компоновка

На этом шаге показано, как формируются и выполняются программные файлы.

**riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 fun.o main.o -o fun.out**

Файл, сгенерированный fun.out, является двоичным

**riscv64-unknown-elf-objdump.exe -f fun.out**

```
fun.out:      file format elf32-littleriscv
architecture: riscv:rv32, flags 0x00000112:
EXEC_P, HAS_SYMS, D_PAGED
start address 0x0001008c
```

Строка EXEC flag\_ P означает, что файлы являются исполняемыми

В этой таблице перечислены исполняемые файлы.

### riscv64-unknown-elf-objdump -h fun.out

```
fun.out:      file format elf32-littleriscv

Sections:
Idx Name          Size      VMA           LMA           File off  Algn
  0 .text          00014958  00010074  00010074  00000074  2**2
    CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .rodata        00000d04  000249d0  000249d0  000149d0  2**3
    CONTENTS, ALLOC, LOAD, READONLY, DATA
  2 .eh_frame      000000b4  00026000  00026000  00016000  2**2
    CONTENTS, ALLOC, LOAD, DATA
  3 .init_array    00000008  000260b4  000260b4  000160b4  2**2
    CONTENTS, ALLOC, LOAD, DATA
  4 .fini_array    00000004  000260bc  000260bc  000160bc  2**2
    CONTENTS, ALLOC, LOAD, DATA
  5 .data          0000099c  000260c0  000260c0  000160c0  2**3
    CONTENTS, ALLOC, LOAD, DATA
  6 .sdata         0000002c  00026a60  00026a60  00016a60  2**3
    CONTENTS, ALLOC, LOAD, DATA
  7 .sbss          00000014  00026a8c  00026a8c  00016a8c  2**2
    ALLOC
  8 .bss           00000048  00026aa0  00026aa0  00016a8c  2**2
    ALLOC
  9 .comment       00000058  00000000  00000000  00016a8c  2**0
    CONTENTS, READONLY
10 .riscv.attributes 0000001c  00000000  00000000  00016ae4  2**0
    CONTENTS, READONLY
11 .debug_aranges  00000218  00000000  00000000  00016b00  2**3
    CONTENTS, READONLY, DEBUGGING
12 .debug_info     0000924d  00000000  00000000  00016d18  2**0
    CONTENTS, READONLY, DEBUGGING
13 .debug_abbrev   00001c52  00000000  00000000  0001ff65  2**0
    CONTENTS, READONLY, DEBUGGING
14 .debug_line     0000a0d0  00000000  00000000  00021bb7  2**0
    CONTENTS, READONLY, DEBUGGING
15 .debug_frame    000002dc  00000000  00000000  0002bc88  2**2
    CONTENTS, READONLY, DEBUGGING
16 .debug_str      00001382  00000000  00000000  0002bf64  2**0
    CONTENTS, READONLY, DEBUGGING
17 .debug_loc      000089e7  00000000  00000000  0002d2e6  2**0
    CONTENTS, READONLY, DEBUGGING
18 .debug_ranges   000012b0  00000000  00000000  00035ccd  2**0
    CONTENTS, READONLY, DEBUGGING
```

Содержимое исполняемых файлов объединено.

### riscv64-unknown-elf-objdump.exe -d -M no-aliases -j .text fun.out

```
0001008c <_start>:
1008c: 00017197      auipc      gp,0x17
10090: 83418193      addi      gp,gp,-1996 # 268c0 <__global_pointer$>
10094: 1cc18513      addi      a0,gp,460 # 26a8c <_edata>
10098: 23018613      addi      a2,gp,560 # 26af0 <__BSS_END__>
1009c: 40a60633      sub       a2,a2,a0
100a0: 00000593      addi      a1,zero,0
100a4: 39c000ef      jal       ra,10440 <memset>
100a8: 00000517      auipc     a0,0x0
100ac: 2a450513      addi      a0,a0,676 # 1034c <__libc_fini_array>
100b0: 254000ef      jal       ra,10304 <atexit>
100b4: 2f8000ef      jal       ra,103ac <__libc_init_array>
100b8: 00012503      lw        a0,0(sp)
100bc: 00410593      addi      a1,sp,4
100c0: 00000613      addi      a2,zero,0
100c4: 118000ef      jal       ra,101dc <main>
100c8: 2500006f      jal       zero,10318 <exit>
```

00010138 <lol>:

10138:	fe010113	addi	sp,sp,-32
1013c:	00112e23	sw	ra,28(sp)
10140:	00812c23	sw	s0,24(sp)
10144:	00912a23	sw	s1,20(sp)
10148:	01212823	sw	s2,16(sp)
1014c:	01312623	sw	s3,12(sp)
10150:	01412423	sw	s4,8(sp)
10154:	00050913	addi	s2,a0,0
10158:	00060a13	addi	s4,a2,0
1015c:	02b05663	bge	zero,a1,10188 <lol+0x50>
10160:	00060793	addi	a5,a2,0
10164:	00259693	slli	a3,a1,0x2
10168:	00c686b3	add	a3,a3,a2
1016c:	00c0006f	jal	zero,10178 <lol+0x40>
10170:	00478793	addi	a5,a5,4
10174:	00d78a63	beq	a5,a3,10188 <lol+0x50>
10178:	0007a703	lw	a4,0(a5)
1017c:	ff275ae3	bge	a4,s2,10170 <lol+0x38>
10180:	00070913	addi	s2,a4,0
10184:	fedff06f	jal	zero,10170 <lol+0x38>
10188:	fff5c993	xori	s3,a1,-1
1018c:	000a0493	addi	s1,s4,0
10190:	00098413	addi	s0,s3,0
10194:	02045263	bge	s0,zero,101b8 <lol+0x80>
10198:	00140413	addi	s0,s0,1
1019c:	00090593	addi	a1,s2,0
101a0:	0004a503	lw	a0,0(s1)
101a4:	130000ef	jal	ra,102d4 <__modsi3>
101a8:	00448493	addi	s1,s1,4
101ac:	fe0504e3	beq	a0,zero,10194 <lol+0x5c>
101b0:	fff90913	addi	s2,s2,-1
101b4:	fd9ff06f	jal	zero,1018c <lol+0x54>
101b8:	00090513	addi	a0,s2,0
101bc:	01c12083	lw	ra,28(sp)
101c0:	01812403	lw	s0,24(sp)
101c4:	01412483	lw	s1,20(sp)
101c8:	01012903	lw	s2,16(sp)
101cc:	00c12983	lw	s3,12(sp)
101d0:	00812a03	lw	s4,8(sp)
101d4:	02010113	addi	sp,sp,32
101d8:	00008067	jalr	zero,0(ra)

```

000101dc <main>:
 101dc:      fd010113          addi    sp,sp,-48
 101e0:      02112623          sw      ra,44(sp)
 101e4:      000257b7          lui     a5,0x25
 101e8:      9dc78793          addi    a5,a5,-1572 # 249dc <__clzsi2+0x5c>
 101ec:      0007a583          lw      a1,0(a5)
 101f0:      0047a603          lw      a2,4(a5)
 101f4:      0087a683          lw      a3,8(a5)
 101f8:      00c7a703          lw      a4,12(a5)
 101fc:      0107a783          lw      a5,16(a5)
 10200:      00b12623          sw      a1,12(sp)
 10204:      00c12823          sw      a2,16(sp)
 10208:      00d12a23          sw      a3,20(sp)
 1020c:      00e12c23          sw      a4,24(sp)
 10210:      00f12e23          sw      a5,28(sp)
 10214:      00c10613          addi    a2,sp,12
 10218:      00500593          addi    a1,zero,5
 1021c:      00800513          addi    a0,zero,8
 10220:      f19ff0ef          jal     ra,10138 <lol>
 10224:      00050593          addi    a1,a0,0
 10228:      00025537          lui     a0,0x25
 1022c:      9d050513          addi    a0,a0,-1584 # 249d0 <__clzsi2+0x50>
 10230:      330000ef          jal     ra,10560 <printf>
 10234:      00025537          lui     a0,0x25
 10238:      9d450513          addi    a0,a0,-1580 # 249d4 <__clzsi2+0x54>
 1023c:      3a8000ef          jal     ra,105e4 <system>
 10240:      00000513          addi    a0,zero,0
 10244:      02c12083          lw      ra,44(sp)
 10248:      03010113          addi    sp,sp,48
 1024c:      00008067          jalr    zero,0(ra)

```

```

00010318 <exit>:
 10318:      ff010113          addi    sp,sp,-16
 1031c:      00000593          addi    a1,zero,0
 10320:      00812423          sw      s0,8(sp)
 10324:      00112623          sw      ra,12(sp)
 10328:      00050413          addi    s0,a0,0
 1032c:      0b0030ef          jal     ra,133dc <__call_exitprocs>
 10330:      1b818793          addi    a5,gp,440 # 26a78 <_global_impure_ptr>
 10334:      0007a503          lw      a0,0(a5)
 10338:      03c52783          lw      a5,60(a0)
 1033c:      00078463          beq     a5,zero,10344 <exit+0x2c>
 10340:      000780e7          jalr    ra,0(a5)
 10344:      00040513          addi    a0,s0,0
 10348:      5740f0ef          jal     ra,1f8bc <_exit>

```

## Листинг 2.13 Таблица символов исполняемого файла

riscv64-unknown-elf-objdump -t fun.out

fun.out: file format elf32-littleriscv

### SYMBOL TABLE:

```
00010074 l d .text 00000000 .text
000249d0 l d .rodata 00000000 .rodata
00026000 l d .eh_frame 00000000 .eh_frame
000260b4 l d .init_array 00000000 .init_array
000260bc l d .fini_array 00000000 .fini_array
000260c0 l d .data 00000000 .data
00026a60 l d .sdata 00000000 .sdata
00026a8c l d .sbss 00000000 .sbss
00026aa0 l d .bss 00000000 .bss
00000000 l d .comment 00000000 .comment
00000000 l d .riscv.attributes 00000000 .riscv.attributes
00000000 l d .debug_aranges 00000000 .debug_aranges
00000000 l d .debug_info 00000000 .debug_info
00000000 l d .debug_abbrev 00000000 .debug_abbrev
00000000 l d .debug_line 00000000 .debug_line
00000000 l d .debug_frame 00000000 .debug_frame
00000000 l d .debug_str 00000000 .debug_str
00000000 l d .debug_loc 00000000 .debug_loc
00000000 l d .debug_ranges 00000000 .debug_ranges
00000000 l df*ABS* 00000000 __call_atexit.c
00010074 l F.text 00000018 register_fini
00000000 l df*ABS* 00000000 crtstuff.c
00026000 l O.eh_frame 00000000 __EH_FRAME_BEGIN__
000100cc l F.text 00000000 __do_global_dtors_aux
00026aa0 l O.bss 00000001 completed.5434
000260bc l O.fini_array 00000000 __do_global_dtors_aux_fini_array_entry
00010118 l F.text 00000000 frame_dummy
00026aa4 l O.bss 00000018 object.5439
000260b8 l O.init_array 00000000 __frame_dummy_init_array_entry
00000000 l df*ABS* 00000000 fun.c
00000000 l df*ABS* 00000000 main.c
00000000 l df*ABS* 00000000 atexit.c
00000000 l df*ABS* 00000000 exit.c
00000000 l df*ABS* 00000000 fini.c
00000000 l df*ABS* 00000000 impure.c
000260c0 l O.data 00000428 impure_data
00000000 l df*ABS* 00000000 init.c
00000000 l df*ABS* 00000000 printf.c
00000000 l df*ABS* 00000000 system.c
00000000 l df*ABS* 00000000 vfprintf.c
00013118 l F.text 000000c0 __sbprintf
00024b5c l O.rodata 00000010 blanks.4504
00024b6c l O.rodata 00000010 zeroes.4505
00000000 l df*ABS* 00000000 wsetup.c
00000000 l df*ABS* 00000000 __atexit.c
```



```

00000000 l df*ABS* 00000000 errno.c
00000000 l df*ABS* 00000000 fflush.c
00000000 l df*ABS* 00000000 findfp.c
000137e8 l F.text 00000008 __fp_lock
000137fc l F.text 00000184 __sinit.part.0
00013980 l F.text 00000008 __fp_unlock
00000000 l df*ABS* 00000000 mallocr.c
00000000 l df*ABS* 00000000 fwalk.c
00000000 l df*ABS* 00000000 ldtoa.c
000140f0 l F.text 00000068 eshdn1
00014158 l F.text 00000070 eshup1
000141c8 l F.text 000000e0 m16m
000142a8 l F.text 00000024 eisnan.part.0
000142cc l F.text 0000004c eneg
00014318 l F.text 00000040 eisneg
00014358 l F.text 000000e8 emovi
00014440 l F.text 0000011c ecmp
0001455c l F.text 0000001c eisinf.part.1
00014578 l F.text 000001ac efun.part.3
00014724 l F.text 0000017c enormlz
000148a0 l F.text 00000408 emdnorm
00014ca8 l F.text 00000164 eiremain
00014e0c l F.text 000000b0 emovo.isra.6
00014ebc l F.text 00000408 emul
000152c4 l F.text 00000504 ediv
000157c8 l F.text 00000144 e113toe.isra.8
00024bc0 l O.rodata 00000014 ezero
00024bd4 l O.rodata 00000014 eone
00024be8 l O.rodata 00000104 etens
00024cec l O.rodata 00000104 emtens
00024df0 l O.rodata 00000022 bmask
00000000 l df*ABS* 00000000 localeconv.c
00000000 l df*ABS* 00000000 locale.c
00000000 l df*ABS* 00000000 makebuf.c
00000000 l df*ABS* 00000000 mallocr.c
00000000 l df*ABS* 00000000 mbtowc_r.c
00000000 l df*ABS* 00000000 memchr.c
00000000 l df*ABS* 00000000 mlock.c
00000000 l df*ABS* 00000000 mprec.c
00024e50 l O.rodata 0000000c p05.3298
00000000 l df*ABS* 00000000 sbrkr.c
00000000 l df*ABS* 00000000 s_frex.c
00000000 l df*ABS* 00000000 sprintf.c
00000000 l df*ABS* 00000000 stdio.c
00000000 l df*ABS* 00000000 strcpy.c
00000000 l df*ABS* 00000000 strlen.c
00000000 l df*ABS* 00000000 strncpy.c
00000000 l df*ABS* 00000000 vfprintf.c
000250e4 l O.rodata 00000010 blanks.4489
000250f4 l O.rodata 00000010 zeroes.4490
00000000 l df*ABS* 00000000 vfprintf.c
0001ba3c l F.text 000000f0 __sprint_r.part.0

```

```

0001ce2c 1 F.text 000000c0 __sbrprintf
00025270 1 O.rodata 00000010 blanks.4480
00025280 1 O.rodata 00000010 zeroes.4481
00000000 1 df*ABS* 00000000 wctomb_r.c
00000000 1 df*ABS* 00000000 writer.c
00000000 1 df*ABS* 00000000 mallocr.c
00000000 1 df*ABS* 00000000 closer.c
00000000 1 df*ABS* 00000000 ctype_.c
00000000 1 df*ABS* 00000000 fclose.c
00000000 1 df*ABS* 00000000 fputwc.c
00000000 1 df*ABS* 00000000 fstatr.c
00000000 1 df*ABS* 00000000 fvwrite.c
00000000 1 df*ABS* 00000000 isatty.c
00000000 1 df*ABS* 00000000 lseekr.c
00000000 1 df*ABS* 00000000 memcpy.c
00000000 1 df*ABS* 00000000 memmove-stub.c
00000000 1 df*ABS* 00000000 readr.c
00000000 1 df*ABS* 00000000 mallocr.c
00000000 1 df*ABS* 00000000 reent.c
00000000 1 df*ABS* 00000000 vfprintf.c
00025500 1 O.rodata 00000010 blanks.4466
00025510 1 O.rodata 00000010 zeroes.4467
00000000 1 df*ABS* 00000000 wbuf.c
00000000 1 df*ABS* 00000000 wctomb.c
00000000 1 df*ABS* 00000000 sys_close.c
00000000 1 df*ABS* 00000000 sys_exit.c
00000000 1 df*ABS* 00000000 sys_fstat.c
00000000 1 df*ABS* 00000000 sys_isatty.c
00000000 1 df*ABS* 00000000 sys_lseek.c
00000000 1 df*ABS* 00000000 sys_read.c
00000000 1 df*ABS* 00000000 sys_sbrk.c
00026a9c 1 O.sbss 00000004 heap_end.1862
00000000 1 df*ABS* 00000000 sys_write.c
00000000 1 df*ABS* 00000000 sys_conv_stat.c
00000000 1 df*ABS* 00000000 libgcc2.c
00000000 1 df*ABS* 00000000 libgcc2.c
00000000 1 df*ABS* 00000000 divdf3.c
00000000 1 df*ABS* 00000000 muldf3.c
00000000 1 df*ABS* 00000000 eqtf2.c
00000000 1 df*ABS* 00000000 getf2.c
00000000 1 df*ABS* 00000000 letf2.c
00000000 1 df*ABS* 00000000 multf3.c
00000000 1 df*ABS* 00000000 subtf3.c
00000000 1 df*ABS* 00000000 fixtfsi.c
00000000 1 df*ABS* 00000000 floatsitf.c
00000000 1 df*ABS* 00000000 extenddfft2.c
00000000 1 df*ABS* 00000000 truncdfdf2.c
00000000 1 df*ABS* 00000000 libgcc2.c
00000000 1 df*ABS* 00000000 libgcc2.c
00000000 1 df*ABS* 00000000 crtstuff.c
000260b0 1 O.eh_frame 00000000 __FRAME_END__
00000000 1 df*ABS* 00000000

```

```

000260c0 l .fini_array 00000000 __fini_array_end
000260bc l .fini_array 00000000 __fini_array_start
000260bc l .init_array 00000000 __init_array_end
000260b4 l .init_array 00000000 __preinit_array_end
000260b4 l .init_array 00000000 __init_array_start
000260b4 l .init_array 00000000 __preinit_array_start
0001875c g F .text 0000009c __mprec_log10
00018860 g F .text 00000078 __any_on
0001d8b8 g F .text 00000054 __isatty_r
00024f28 g O .rodata 00000028 __mprec_tinytens
00018d9c g F .text 000000b0 strcpy
0001e1d8 g F .text 00000040 cleanup_glue
0001d90c g F .text 00000060 __lseek_r
000215b4 g F .text 00000144 .hidden __getf2
000214e8 g F .text 000000cc .hidden __eqtf2
00010560 g F .text 00000054 printf
000268c0 g *ABS* 00000000 __global_pointer$
0001f75c g F .text 00000078 __wrtomb_r
00018bb0 g F .text 00000068 __sseek
00013b18 g F .text 00000010 __sinit
0001f5c4 g F .text 00000184 __swbuf_r
00016c7c g F .text 0000007c __setlocale_r
00013988 g F .text 00000078 __sfmoreglue
00017824 g F .text 00000004 __malloc_unlock
00024298 g F .text 00000188 .hidden __floatsitf
0001da88 g F .text 00000120 memmove
00013b04 g F .text 00000014 __cleanup
00017828 g F .text 000000a8 __Balloc
000134f8 g F .text 0000000c __errno
0001fb40 g F .text 000000a4 __conv_stat
00016c64 g F .text 00000008 __localeconv_l
0001d3a0 g F .text 0000005c __fstat_r
00026ae4 g O .bss 00000004 errno
00018b20 g F .text 00000008 __seofread
00026a60 g .sdata 00000000 __SDATA_BEGIN__
0001d96c g F .text 0000011c memcpy
000137f0 g F .text 0000000c __cleanup_r
00018fa4 g F .text 00002a98 __svfprintf_r
000186b8 g F .text 000000a4 __ratio
000105e4 g F .text 00000030 system
0001051c g F .text 00000044 __printf_r
000216f8 g F .text 00000144 .hidden __letf2
00010258 g F .text 00000048 .hidden __udivsi3
0001f900 g F .text 00000070 __fstat
00024e60 g O .rodata 000000c8 __mprec_tens
00026a98 g O .sbss 00000004 __malloc_top_pad
00026a7c g O .sdata 00000000 .hidden __dso_handle
00016c6c g F .text 00000008 __localeconv_r
0001d3fc g F .text 000004bc __sfvwrite_r
00017c88 g F .text 00000034 __i2b
000188d8 g F .text 00000054 __sbrk_r
0001dba8 g F .text 00000060 __read_r

```

```

0001d0a4 g F .text 00000110 _fclose_r
000137c0 g F .text 00000028 fflush
00026a94 g O .sbss 00000004 __malloc_max_sbrked_mem
000105b4 g F .text 00000030 _system_r
0001845c g F .text 00000118 __b2d
000201b8 g F .text 000004f0 .hidden __umoddi3
0001f970 g F .text 00000040 _isatty
00026a78 g O .sdata 00000004 _global_impure_ptr
0001dc08 g F .text 000005d0 _realloc_r
000103ac g F .text 00000094 __libc_init_array
0001fbe4 g F .text 000005d4 .hidden __udivdi3
0001d2f0 g F .text 0000002c _fputwc_r
00024f50 g O .rodata 00000028 __mprec_bigtens
00017a4c g F .text 00000110 __s2b
0001fa50 g F .text 000000a0 _sbrk
000181bc g F .text 0000005c __mcmp
00013b38 g F .text 00000014 __fp_lock_all
0001034c g F .text 00000060 __libc_fini_array
0001e218 g F .text 00000110 _reclaim_reent
00017b5c g F .text 00000074 __hi0bits
00024148 g F .text 00000150 .hidden __fixtfsi
00017f28 g F .text 00000148 __pow5mult
000102a0 g F .text 00000010 .hidden __umodsi3
00024980 g F .text 0000004c .hidden __clzsi2
00013b28 g F .text 00000004 __sfp_lock_acquire
00017744 g F .text 000000dc memchr
000189dc g F .text 0000006c _sprintf_r
00013c88 g F .text 000002f8 _free_r
00016cf8 g F .text 00000008 __locale_mb_cur_max
000133dc g F .text 0000011c __call_exitprocs
00026a84 g O .sdata 00000004 __malloc_sbrk_base
0001008c g F .text 00000040 _start
0001f9b0 g F .text 00000050 _lseek
00018070 g F .text 0000014c __lfun
000216f8 g F .text 00000144 .hidden __ltf2
000214e8 g F .text 000000cc .hidden __netf2
0001e328 g F .text 000001bc __ssprint_r
00013340 g F .text 0000009c __register_exitproc
00016c00 g F .text 00000064 _ldcheck
00017cbc g F .text 0000026c __multiply
0002495c g F .text 00000024 .hidden __mulsi3
00018ed8 g F .text 000000cc strncpy
00026abc g O .bss 00000028 __malloc_current_mallinfo
00018574 g F .text 00000144 __d2b
00024420 g F .text 00000208 .hidden __extenddftf2
0001d050 g F .text 00000054 _close_r
000131d8 g F .text 00000168 __swsetup_r
000206a8 g F .text 000007d0 .hidden __divdf3
00013a00 g F .text 00000104 __sfp
000187f8 g F .text 00000068 __copybits
00026af0 g .bss 00000000 __BSS_END__
00026654 g O .data 00000408 __malloc_av_

```

```

00013b34 g F .text 00000004 __sinit_lock_release
00020e78 g F .text 00000670 .hidden __muldf3
00010138 g F .text 000000a4 lol
00018ac4 g F .text 0000005c __sread
00017820 g F .text 00000004 __malloc_lock
00013760 g F .text 00000060 _fflush_r
0001cf88 g F .text 000000c8 _calloc_r
00026a8c g .sbss 00000000 __bss_start
00010440 g F .text 000000dc memset
000101dc g F .text 00000074 main
00026a90 g O .sbss 00000004 __malloc_max_total_mem
0001f748 g F .text 00000014 __swbuf
00018c18 g F .text 00000008 __sclose
0001d1b4 g F .text 00000010 fclose
00016f04 g F .text 000007cc _malloc_r
0001cef8 g F .text 00000030 __ascii_wctomb
00013f80 g F .text 000000b0 _fwalk
000176d0 g F .text 0000000c _mbtowc_r
00010250 g F .text 00000084 .hidden __divsi3
00013b60 g F .text 00000128 _malloc_trim_r
00018c20 g F .text 0000017c strcmp
0001ce14 g F .text 00000018 vfiprintf
0002183c g F .text 0000136c .hidden __multf3
00018a48 g F .text 0000007c sprintf
000255d4 g O .rodata 00000100 .hidden __clz_tab
00026a8c g O .sbss 00000004 _PathLocale
00010304 g F .text 00000014 atexit
0001cf28 g F .text 00000060 _write_r
00016d00 g F .text 00000014 setlocale
00026a80 g O .sdata 00000004 _impure_ptr
00013504 g F .text 0000025c __sflush_r
000215b4 g F .text 00000144 .hidden __gttf2
0001e4e4 g F .text 000010e0 _svfiprintf_r
000176dc g F .text 00000068 __ascii_mbtowc
00022ba8 g F .text 000015a0 .hidden __subtf3
000183fc g F .text 00000060 __ulp
00013b4c g F .text 00000014 __fp_unlock_all
00016c74 g F .text 00000008 localeconv
00016d14 g F .text 000000d0 __swatbuf_r
000260c0 g .data 00000000 __DATA_BEGIN__
0001faf0 g F .text 00000050 _write
00026a8c g .sdata 00000000 _edata
00026af0 g .bss 00000000 _end
0001d1c4 g F .text 0000012c __fputwc
00018b28 g F .text 00000088 __swrite
00026a88 g O .sdata 00000004 __malloc_trim_threshold
00010318 g F .text 00000034 exit
0001bb44 g F .text 000012d0 _vfiprintf_r
00014030 g F .text 000000c0 _fwalk_reent
00018218 g F .text 000001e4 __mdiff
000102d4 g F .text 00000030 .hidden __modsi3
00013b2c g F .text 00000004 __sfp_lock_release

```

```

0001590c g F .text 000012f4 _ldtoa_r
00025290 g O .rodata 00000101 _ctype_
0001fa00 g F .text 00000050 _read
0001f8bc g F .text 00000044 _exit
00016de4 g F .text 00000120 __smakebuf_r
00018e4c g F .text 0000008c strlen
0001bb2c g F .text 00000018 __sprint_r
0001ceec g F .text 0000000c _wctomb_r
00010614 g F .text 00002aec _vfprintf_r
00017bd0 g F .text 000000b8 __lo0bits
0001f7d4 g F .text 00000090 wctomb
0001892c g F .text 000000b0 frexp
000264e8 g O .data 0000016c __global_locale
00013100 g F .text 00000018 vfprintf
00024628 g F .text 00000334 .hidden __trunctdf2
0001d31c g F .text 00000084 fputwc
0001f864 g F .text 00000058 _close
00013b30 g F .text 00000004 __sinit_lock_acquire
000178f4 g F .text 00000158 __multadd
000178d0 g F .text 00000024 _Bfree

```

В code .comment используется GCC версии 8.3.0:

Проанализируем таблицу перемещений исполняемого файла

**riscv64-unknown-elf-objdump.exe -r fun.out**

```

C:\Users\sevda\Devshit\bin>riscv64-unknown-elf-objdump.exe -r fun.out
fun.out:      file format elf32-littleriscv

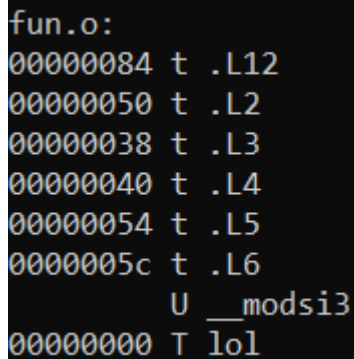
```

Таблица смещений пуста, что означает, что процесс компиляции прошел успешно.

### 3. Формирование статической библиотеки, разработка make-файлов для сборки библиотеки

#### Листинг 3.1. файл fun.o

```
riscv64-unknown-elf-ar.exe -rsc libfun.a fun.o  
riscv64-unknown-elf-ar.exe -t libfun.a fun.o  
riscv64-unknown-elf-nm.exe libfun.a
```



```
fun.o:  
00000084 t .L12  
00000050 t .L2  
00000038 t .L3  
00000040 t .L4  
00000054 t .L5  
0000005c t .L6  
          U __modsi3  
00000000 T lol
```

#### Листинг 3.2. Makefile для создания статической библиотеки

```
# Цели  
.PHONY: all clean  
  
# Исходные файлы для сборки библиотеки  
OBJS= fun.c  
  
#Вызываемые приложения  
AR = riscv64-unknown-elf-ar.exe  
CC = riscv64-unknown-elf-gcc.exe  
  
# Файл библиотеки  
MYLIBNAME = Libfun.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)

# $^ = (fun.o)

$(MYLIBNAME): fun.o

$(AR) -rsc $@ $^

```

### Листинг 3.3. Makefile для сборки исполняемого файла

```

# Цели

.PHONY: all clean

# Исходные файлы для сборки библиотеки

OBJS= main.c \

    Libfun.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

```
SYMBOL TABLE:
00000000000100b0 1 d .text 0000000000000000 .text
000000000001c110 1 d .rodata 0000000000000000 .rodata
000000000001d000 1 d .eh_frame 0000000000000000 .eh_frame
000000000001d008 1 d .init_array 0000000000000000 .init_array
000000000001d018 1 d .fini_array 0000000000000000 .fini_array
000000000001d020 1 d .data 0000000000000000 .data
000000000001e120 1 d .sdata 0000000000000000 .sdata
000000000001e178 1 d .sbss 0000000000000000 .sbss
000000000001e1a0 1 d .bss 0000000000000000 .bss
0000000000000000 1 d .comment 0000000000000000 .comment
0000000000000000 1 d .riscv.attributes 0000000000000000 .riscv.attributes
0000000000000000 1 d .debug_aranges 0000000000000000 .debug_aranges
0000000000000000 1 d .debug_info 0000000000000000 .debug_info
0000000000000000 1 d .debug_abbrev 0000000000000000 .debug_abbrev
0000000000000000 1 d .debug_line 0000000000000000 .debug_line
0000000000000000 1 d .debug_frame 0000000000000000 .debug_frame
0000000000000000 1 d .debug_str 0000000000000000 .debug_str
```

Созданный файл идентичен файлу созданному ранее

### Вывод

В этой курсовой работе была разработана функция на языке C, которая находит наименьшее число в списке, и с помощью символа (%) мы находим, можно ли его разделить на 0 или нет, что, если все значения равны, приведет к 0 .

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.