

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

**Отчёт по лабораторной работе № 4**

**Дисциплина:** Низкоуровневое программирование

**Тема:** Раздельная компиляция.

Выполнил студент гр. 3530901/10005 \_\_\_\_\_ Захаров В.А  
(подпись)  
Преподаватель \_\_\_\_\_ Коренев Д.А.  
(подпись) “ \_\_ ” \_\_\_\_\_ 2022 г.

Санкт-Петербург  
2022

## **1. Формулировка задачи**

1) На языке С 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-файлы для сборки библиотеки и использующей ее тестовой программы. Проанализировать ход сборки библиотеки и программы, созданные файлы зависимостей.

## **2. Вариант задания**

Определение наиболее часто встречающегося элемента в массиве.

## **3. Ход решения**

### **3.1 Текст программ, реализующих определенную вариантом задания функциональность**

*Листинг 1. Программа “mostCommon.c”*

```
int *mostCommon(int *array, int array_size) {
    int currentCounter = 0, biggestCounter = 0;
    static int result[2];
    //currentCounter - текущее число повторений
    //biggestCounter - наибольшее число повторений
    //result[0] - наиболее часто встречающийся элемент
    //result[1] - число повторений
    for (int i = 0; i < array_size - 1; i++) {
        for (int j = i; j < array_size; j++) {
            if (array[i] == array[j]) {
                currentCounter++;
            }
        }
        if (currentCounter >= biggestCounter) {
            biggestCounter = currentCounter;
            result[0] = array[i];
        }
        currentCounter = 0;
    }
    result[1] = biggestCounter;
```

```
        return result;
    }
```

*Листинг 2. Заголовочный файл “mostCommon.h”*

```
#ifndef MOSTCOMMON_H
```

```
#define MOSTCOMMON_H
```

```
extern int *mostCommon(int *array, int array_size);
```

```
#endif // MOSTCOMMON_H
```

*Листинг3. Программа “main.c”*

```
#include "mostCommon.h"
```

```
#include <stdio.h>
```

```
int main() {
```

```
    int array[] = {1, 2, 2, 4, 4, 80, 7, 80, 9, 4, 4};  
    int array_size = sizeof(array) / sizeof(int);
```

```
    int *result = mostCommon(array, array_size);
```

```
    printf("The most common value: %d.\nAmount of repetitions: %d.",  
          result[0], result[1]);
```

```
}
```

Пусть дано число из массива. Сравнить его со всеми остальными. В случае совпадения увеличить счетчик повторений на 1. Сравнить счетчик повторений с результатами работы программы на предыдущих этапах, если текущий результат оказался больше, запомнить текущее значение счетчика повторений и значение числа. Повторить для всех элементов массива.

### **3.2 Сборка программ «по шагам», анализ промежуточных и результирующих файлов**

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

```
riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -O1 -E mostCommon.c -o  
mostCommon.i
```

```
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\files>
riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -O1 -E main.c -o main.i
```

```
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\files>
riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -O1 -E mostCommon.c -o mostCommon.i
```

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.

Результаты работы препроцессора мало отличаются от исходных версий программ:

Листинг 4. Файл “mostCommon.i”

```
# 1 "mostCommon.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "mostCommon.c"
int *mostCommon(int *array, int array_size)
{
    int currentCounter = 0, biggestCounter = 0;
    static int result[2];

    for (int i = 0; i < array_size - 1; i++)
    {
        for (int j = i; j < array_size; j++)
        {
            if (array[i] == array[j])
            {
                currentCounter++;
            }
        }
        if (currentCounter >= biggestCounter)
        {
            biggestCounter = currentCounter;
            result[0] = array[i];
        }
        currentCounter = 0;
    }
    result[1] = biggestCounter;
    return result;
```

```
}
```

```
# 1 "main.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "main.c"
# 1 "mostCommon.h" 1
```

Листинг5. Файл “main.i”

```
extern int *mostCommon(int *array, int array_size);
# 2 "main.c" 2

# 5 "main.c"
int main()
{
    int array[] = {1, 2, 2, 4, 4, 80, 7, 80, 9, 4, 4};
    int array_size = sizeof(array) / sizeof(int);
    int *result = mostCommon(array, array_size);
    printf("The most common value: %d.\nAmount of repetitions: %d.",
    result[0], result[1]);
}
```

Появившиеся нестандартные директивы, начинающиеся с символа “#”, используются для передачи информации об исходном тексте из препроцессора в компилятор. Так, в файле “main.i” четвертая директива «# 1 “main.c”» информирует компилятор о том, что следующая строка является результатом обработки строки 1 исходного файла “main.c”. В этой строке стояла команда #include "mostCommon.h", поэтому препроцессор произвел вставку содержимого этого заголовочного файла, то есть определение функции mostCommon(). Далее же начинается описание самого содержимого файла, что происходит после директивы #2 "main.c" 2. Исходный код тестирующей функции main() после работы препроцессора остался без изменений, как и исходный код функции mostCommon () .

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

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

```
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\files>
riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -O1 -S mostCommon.i -o mostCommon.s
```

```
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\files>
riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -O1 -S main.i -o 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 “mostCommon.s”

```
.file "mostCommon.c"
.option nopic
.attribute arch, "rv32i2p0"
.attribute unaligned_access, 0
.attribute stack_align, 16
.text
.align    2
.globl   mostCommon
.type    mostCommon, @function
mostCommon:
    li      a5,1
    ble   a1,a5,.L9
    mv     a6,a0
    addi   t3,a1,-1
    slli   a5,a1,2
    add    a0,a0,a5
    li      a7,0
    li      t1,0
    lui    t4,%hi(result.2138)
    j     .L3
.L6:
    addi   a7,a7,1
    addi   a6,a6,4
    beq   a7,t3,.L2
.L3:
    li      a3,0
```

```

ble  a1,a7,.L8
lw   a2,0(a6)
mv   a4,a6
li   a3,0
.L5:
lw   a5,0(a4)
sub a5,a2,a5
seqz a5,a5
add  a3,a3,a5
addi a4,a4,4
bne a4,a0,.L5
.L8:
blt a3,t1,.L6
lw   a5,0(a6)
sw   a5,%lo(result.2138)(t4)
mv   t1,a3
j    .L6
.L9:
li   t1,0
.L2:
lui a0,%hi(result.2138)
addi a5,a0,%lo(result.2138)
sw   t1,4(a5)
addi a0,a0,%lo(result.2138)
ret
.size mostCommon, .-mostCommon
.section .sbss,"aw",@nobits
.align 2
.type result.2138, @object
.size result.2138, 8
result.2138:
.zero 8
.ident   "GCC: (SiFive GCC 8.3.0-2020.04.1) 8.3.0"

```

По метке `mostCommon` начинается тело самой функции. В `a0` передается `n`, которое перемещается в `a2`.

#### *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,-64
    sw    ra,60(sp)
    lui   a5,%hi(.LANCHOR0)
    addi a5,a5,%lo(.LANCHOR0)
    lw    t4,0(a5)
    lw    t3,4(a5)
    lw    t1,8(a5)
    lw    a7,12(a5)
    lw    a6,16(a5)
    lw    a0,20(a5)
    lw    a1,24(a5)
    lw    a2,28(a5)
    lw    a3,32(a5)
    lw    a4,36(a5)
    lw    a5,40(a5)
    sw    t4,4(sp)
    sw    t3,8(sp)
    sw    t1,12(sp)
    sw    a7,16(sp)
    sw    a6,20(sp)
    sw    a0,24(sp)
    sw    a1,28(sp)
    sw    a2,32(sp)
    sw    a3,36(sp)
    sw    a4,40(sp)
    sw    a5,44(sp)
    li    a1,11
    addi a0,sp,4
    call mostCommon
    lw    a2,4(a0)
    lw    a1,0(a0)
    lui   a0,%hi(.LC1)
    addi a0,a0,%lo(.LC1)
    call printf
    li    a0,0
    lw    ra,60(sp)
    addi sp,sp,64
    jr    ra
.size main, .-main
.section .rodata
.align      2
```

```

.set .LANCHOR0,. + 0
.LC0:
.word 1
.word 2
.word 2
.word 4
.word 4
.word 80
.word 7
.word 80
.word 9
.word 4
.word 4
.section .rodata.str1.4,"aMS",@progbits,1
.align 2
.LC1:
.string "The most common value: %d.\nAmount of repetitions:
%d."
.ident "GCC: (SiFive GCC 8.3.0-2020.04.1) 8.3.0"

```

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

```
riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -c mostCommon.s -o mostCommon.o
```

```
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\files>
riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -c main.s -o main.o
```

```
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\files>
riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -c mostCommon.s -o mostCommon.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:

```
riscv64-unknown-elf-objdump.exe -f mostCommon.o
```

```
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd>riscv64-unknown-elf-objdump.exe  
-f mostCommon.o  
  
mostCommon.o:      file format elf32-littleriscv  
architecture: riscv:rv32, flags 0x00000011:  
HAS_RELOC, HAS_SYMS  
start address 0x00000000  
  
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd>riscv64-unknown-elf-objdump.exe  
-f main.o  
  
main.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 флаги HAS\_RELOC).

Выведем все заголовки секций объектных файлов (команда riscv64-unknown-elf-objdump.exe -h main.o):

```
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd>riscv64-unknown-elf-objdump.exe
-h main.o

main.o:      file format elf32-littleriscv

Sections:
Idx Name      Size    VMA      LMA      File off  Align
 0 .text      000000a0 00000000 00000000 00000034 2**2
              CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data      00000000 00000000 00000000 000000d4 2**0
              CONTENTS, ALLOC, LOAD, DATA
 2 .bss       00000000 00000000 00000000 000000d4 2**0
              ALLOC
 3 .rodata     0000002c 00000000 00000000 000000d4 2**2
              CONTENTS, ALLOC, LOAD, READONLY, DATA
 4 .rodata.str1.4 00000036 00000000 00000000 00000100 2**2
              CONTENTS, ALLOC, LOAD, READONLY, DATA
 5 .comment    00000029 00000000 00000000 00000136 2**0
              CONTENTS, READONLY
 6 .riscv.attributes 0000001c 00000000 00000000 0000015f 2**0
              CONTENTS, READONLY

C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd>riscv64-unknown-elf-objdump.exe
-h mostCommon.o

mostCommon.o:      file format elf32-littleriscv

Sections:
Idx Name      Size    VMA      LMA      File off  Align
 0 .text      0000008c 00000000 00000000 00000034 2**2
              CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data      00000000 00000000 00000000 000000c0 2**0
              CONTENTS, ALLOC, LOAD, DATA
 2 .bss       00000000 00000000 00000000 000000c0 2**0
              ALLOC
 3 .sbss      00000008 00000000 00000000 000000c0 2**2
              ALLOC
 4 .comment    00000029 00000000 00000000 000000c0 2**0
              CONTENTS, READONLY
 5 .riscv.attributes 0000001c 00000000 00000000 000000e9 2**0
              CONTENTS, READONLY
```

Далее представлены таблицы символов файлов main.o и mostCommon.o.

```
riscv64-unknown-elf-objdump.exe -t main.o
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd>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    00000000 .rodata
00000000 l  .rodata    00000000 .LANCHOR0
00000000 l  d  .rodata.str1.4 00000000 .rodata.str1.4
00000000 l  .rodata.str1.4 00000000 .LC1
00000000 l  d  .comment   00000000 .comment
00000000 l  d  .riscv.attributes  00000000 .riscv.attributes
00000000 g  F .text    000000a0 main
00000000     *UND*  00000000 mostCommon
00000000     *UND*  00000000 printf

C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd>riscv64-unknown-elf-objdump.exe
-t mostCommon.o

mostCommon.o:      file format elf32-littleriscv

SYMBOL TABLE:
00000000 l  df *ABS*  00000000 mostCommon.c
00000000 l  d  .text  00000000 .text
00000000 l  d  .data  00000000 .data
00000000 l  d  .bss   00000000 .bss
00000000 l  0 .sbss   00000008 result.2138
00000000 l  d  .sbss   00000000 .sbss
00000074 l  .text    00000000 .L9
00000034 l  .text    00000000 .L3
00000078 l  .text    00000000 .L2
00000060 l  .text    00000000 .L8
00000048 l  .text    00000000 .L5
00000028 l  .text    00000000 .L6
00000000 l  d  .comment   00000000 .comment
00000000 l  d  .riscv.attributes  00000000 .riscv.attributes
00000000 g  F .text    0000008c mostCommon
```

В каждой таблице только одни глобальный (флаг “g”) символ типа «функция» (“F”) – “mostCommon” и “main” соответственно.

Проанализируем секции .text объектных файлов – секций кода, в которых содержатся коды инструкций:

```
riscv64-unknown-elf-objdump.exe -d -M no=aliases -j .text mostCommon.o
```

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

Disassembly of section .text:

```

00000000 <mostCommon>:
    0:      Riscv64-unknown-elf-objdump.exe: unrecognized disassembler
option: no=aliases
00100793           li      a5,1
    4: 06b7d863          bge    a5,a1,74 <.L9>
    8: 00050813          mv     a6,a0
   c: fff58e13          addi   t3,a1,-1
   10: 00259793         slli   a5,a1,0x2
   14: 00f50533         add    a0,a0,a5
   18: 00000893         li     a7,0
   1c: 00000313         li     t1,0
   20: 00000eb7         lui    t4,0x0
   24: 0100006f         j     34 <.L3>

00000028 <.L6>:
   28: 00188893         addi   a7,a7,1
   2c: 00480813         addi   a6,a6,4
   30: 05c88463         beq    a7,t3,78 <.L2>

00000034 <.L3>:
   34: 00000693         li     a3,0
   38: 02b8d463         bge   a7,a1,60 <.L8>
   3c: 00082603         lw    a2,0(a6)
   40: 00080713         mv    a4,a6
   44: 00000693         li     a3,0

00000048 <.L5>:
   48: 00072783         lw    a5,0(a4)
   4c: 40f607b3         sub   a5,a2,a5
   50: 0017b793         seqz  a5,a5
   54: 00f686b3         add   a3,a3,a5
   58: 00470713         addi  a4,a4,4
   5c: fea716e3         bne   a4,a0,48 <.L5>

00000060 <.L8>:
   60: fc66c4e3         blt   a3,t1,28 <.L6>
   64: 00082783         lw    a5,0(a6)
   68: 00fea023         sw    a5,0(t4) # 0 <mostCommon>
   6c: 00068313         mv    t1,a3
   70: fb9ff06f         j     28 <.L6>

00000074 <.L9>:
   74: 00000313         li     t1,0

```

```

00000078 <.L2>:
78: 00000537          lui    a0,0x0
7c: 00050793          mv     a5,a0
80: 0067a223          sw     t1,4(a5)
84: 00050513          mv     a0,a0
88: 00008067          ret

```

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

Disassembly of section .text:

```

00000000 <main>:
0:      Riscv64-unknown-elf-objdump.exe: unrecognized disassembler
option: no=aliases
fc010113          addi   sp,sp,-64
4: 02112e23          sw     ra,60(sp)
8: 000007b7          lui    a5,0x0
c: 00078793          mv     a5,a5
10: 0007ae83         lw     t4,0(a5) # 0 <main>
14: 0047ae03         lw     t3,4(a5)
18: 0087a303         lw     t1,8(a5)
1c: 00c7a883         lw     a7,12(a5)
20: 0107a803         lw     a6,16(a5)
24: 0147a503         lw     a0,20(a5)
28: 0187a583         lw     a1,24(a5)
2c: 01c7a603         lw     a2,28(a5)
30: 0207a683         lw     a3,32(a5)
34: 0247a703         lw     a4,36(a5)
38: 0287a783         lw     a5,40(a5)
3c: 01d12223         sw     t4,4(sp)
40: 01c12423         sw     t3,8(sp)
44: 00612623         sw     t1,12(sp)
48: 01112823         sw     a7,16(sp)
4c: 01012a23         sw     a6,20(sp)
50: 00a12c23         sw     a0,24(sp)
54: 00b12e23         sw     a1,28(sp)
58: 02c12023         sw     a2,32(sp)
5c: 02d12223         sw     a3,36(sp)
60: 02e12423         sw     a4,40(sp)
64: 02f12623         sw     a5,44(sp)
68: 00b00593         li     a1,11
6c: 00410513         addi   a0,sp,4
70: 00000097         auipc  ra,0x0
74: 000080e7         jalr   ra # 70 <main+0x70>

```

```
78: 00452603      lw      a2,4(a0)
7c: 00052583      lw      a1,0(a0)
80: 00000537      lui     a0,0x0
84: 00050513      mv     a0,a0
88: 00000097      auipc  ra,0x0
8c: 000080e7      jalr   ra # 88 <main+0x88>
90: 00000513      li     a0,0
94: 03c12083      lw      ra,60(sp)
98: 04010113      addi   sp,sp,64
9c: 00008067      ret
```

Дизассемблированный код практически идентичен сгенерированному (за исключением псевдоинструкций).

Секции .data объектных файлов – секции инициализированных данных – не содержат данных, размер секций, как было выведено выше, равен нулю:

```
riscv64-unknown-elf-objdump.exe -d -M no=aliases -j .data main.o
mostCommon.o

C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd>
riscv64-unknown-elf-objdump.exe -d -M no=aliases -j .data main.o mostCommon.o

main.o:    file format elf32-littleriscv

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

Секции .bss объектных файлов – секции данных, инициализированных нулями – таким же образом пусты:

```
riscv64-unknown-elf-objdump.exe -d -M no=aliases -j .bss main.o mostCommon.o
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd>riscv64-unknown-elf-objdump.exe
-d -M no=aliases -j .bss main.o mostCommon.o

main.o:    file format elf32-littleriscv

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

Секция .comment – секция данных о версиях – и для одного и для другого файла содержит одни и те же значения – сведения о GCC версии 8.3.0 от SiFive:

```
riscv64-unknown-elf-objdump.exe -s -j .comment main.o mostCommon.o
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd>riscv64-unknown-elf-objdump.exe
-s -j .comment main.o mostCommon.o

main.o:      file format elf32-littleriscv

Contents of section .comment:
0000 00474343 3a202853 69466976 65204743  .GCC: (SiFive GC
0010 4320382e 332e302d 32303230 2e30342e  C 8.3.0-2020.04.
0020 31292038 2e332e30 00                  1) 8.3.0.

mostCommon.o:      file format elf32-littleriscv

Contents of section .comment:
0000 00474343 3a202853 69466976 65204743  .GCC: (SiFive GC
0010 4320382e 332e302d 32303230 2e30342e  C 8.3.0-2020.04.
0020 31292038 2e332e30 00                  1) 8.3.0.
```

Секция .riscv.attributes обоих объектных файлов содержит одну и ту же информацию об используемой архитектуре команд RV32I:

```
riscv64-unknown-elf-objdump.exe -s -j .riscv.attributes main.o mostCommon.o
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd>riscv64-unknown-elf-objdump.exe
-s -j .riscv.attributes main.o mostCommon.o

main.o:      file format elf32-littleriscv

Contents of section .riscv.attributes:
0000 411b0000 00726973 63760001 11000000  A....riscv.....
0010 04100572 76333269 32703000          ...rv32i2p0.

mostCommon.o:      file format elf32-littleriscv

Contents of section .riscv.attributes:
0000 411b0000 00726973 63760001 11000000  A....riscv.....
0010 04100572 76333269 32703000          ...rv32i2p0.
```

Проанализируем таблицы перемещений объектных файлов:

```
riscv64-unknown-elf-objdump.exe -r main.o mostCommon.o
```

```
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd>riscv64-unknown-elf-objdump.exe
-r main.o mostCommon.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_L012_I   .LANCHOR0
0000000c R_RISCV_RELAX   *ABS*
00000070 R_RISCV_CALL    mostCommon
00000070 R_RISCV_RELAX   *ABS*
00000080 R_RISCV_HI20    .LC1
00000080 R_RISCV_RELAX   *ABS*
00000084 R_RISCV_L012_I   .LC1
00000084 R_RISCV_RELAX   *ABS*
00000088 R_RISCV_CALL    printf
00000088 R_RISCV_RELAX   *ABS*

mostCommon.o:      file format elf32-littleriscv

RELOCATION RECORDS FOR [.text]:
OFFSET   TYPE        VALUE
00000020 R_RISCV_HI20    result.2138
00000020 R_RISCV_RELAX   *ABS*
00000068 R_RISCV_L012_S   result.2138
00000068 R_RISCV_RELAX   *ABS*
00000078 R_RISCV_HI20    result.2138
00000078 R_RISCV_RELAX   *ABS*
0000007c R_RISCV_L012_I   result.2138
0000007c R_RISCV_RELAX   *ABS*
00000084 R_RISCV_L012_I   result.2138
00000084 R_RISCV_RELAX   *ABS*
00000004 R_RISCV_BRANCH  .L9
00000024 R_RISCV_JAL     .L3
00000030 R_RISCV_BRANCH  .L2
00000038 R_RISCV_BRANCH  .L8
0000005c R_RISCV_BRANCH  .L5
00000060 R_RISCV_BRANCH  .L6
00000070 R_RISCV_JAL     .L6
```

В таблицах перемещения mostCommon.o содержится информация о переходах (R\_RISCV\_JAL) и ветвлениях (R\_RISCV\_BRANCH). В таблицах перемещения main.o, есть R\_RISCV\_CALL, чтобы информация о переходах соответствовала mostCommon. Записи типа R\_RISCV\_RELAX заносятся в таблицу перемещений в дополнение к записям типа R\_RISCV\_CAL.

Следующим шагом является компоновка и формирование исполняемых файлов программ:

```
riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 main.o mostCommon.o -
o mostCommon.out
```

```
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd>Riscv64-unknown-elf-gcc.exe  
-march=rv32i -mabi=ilp32 main.o mostCommon.o -o mostCommon.out
```

Сформированный компоновщиком файл “mostCommon.out”, также является «бинарным»:

```
riscv64-unknown-elf-objdump.exe -f mostCommon.out
```

```
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd>riscv64-unknown-elf-objdump.exe  
-f mostCommon.out
```

```
mostCommon.out:      file format elf32-littleriscv  
architecture: riscv:rv32, flags 0x00000112:  
EXEC_P, HAS_SYMS, D_PAGED  
start address 0x00010090
```

Флаг EXEC\_Руказывает, что файл действительно является исполняемым, после загрузки его выполнение должно начаться с адреса 0x00010090 (entrypoint).

Перечислим секции исполняемого файла:

```
riscv64-unknown-elf-objdump.exe -h mostCommon.out
```

```
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd>riscv64-unknown-elf-objdump.exe
-h mostCommon.out

mostCommon.out:      file format elf32-littleriscv

Sections:
Idx Name      Size    VMA      LMA      File off  Algn
 0 .text      00014908 00010074 00010074 00000074 2**2
                CONTENTS, ALLOC, LOAD, READONLY, CODE
 1 .rodata     000000d44 00024980 00024980 00014980 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       0000009c 000260c0 000260c0 000160c0 2**3
                CONTENTS, ALLOC, LOAD, DATA
 6 .sdata      00000002c 00026a60 00026a60 00016a60 2**3
                CONTENTS, ALLOC, LOAD, DATA
 7 .sbss       00000001c 00026a8c 00026a8c 00016a8c 2**2
                ALLOC
 8 .bss        000000048 00026aa8 00026aa8 00016a8c 2**2
                ALLOC
 9 .comment    00000028 00000000 00000000 00016a8c 2**0
                CONTENTS, READONLY
10 .riscv.attributes 0000001c 00000000 00000000 00016ab4 2**0
                CONTENTS, READONLY
11 .debug_aranges 00000218 00000000 00000000 00016ad0 2**3
                CONTENTS, READONLY, DEBUGGING
12 .debug_info   0000924d 00000000 00000000 00016ce8 2**0
                CONTENTS, READONLY, DEBUGGING
13 .debug_abbrev 00001c52 00000000 00000000 0001ff35 2**0
                CONTENTS, READONLY, DEBUGGING
14 .debug_line   0000a0d0 00000000 00000000 00021b87 2**0
                CONTENTS, READONLY, DEBUGGING
15 .debug_frame  000002dc 00000000 00000000 0002bc58 2**2
                CONTENTS, READONLY, DEBUGGING
16 .debug_str    00001382 00000000 00000000 0002bf34 2**0
                CONTENTS, READONLY, DEBUGGING
17 .debug_loc    000089e7 00000000 00000000 0002d2b6 2**0
                CONTENTS, READONLY, DEBUGGING
18 .debug_ranges 000012b0 00000000 00000000 00035c9d 2**0
                CONTENTS, READONLY, DEBUGGING
```

В исполняемом файле действительно производится слияние содержания секций обоих объектных файлов, а также значительное расширение списка секций новыми блоками.

Проанализируем содержимое секции .text исполняемого файла:

```
riscv64-unknown-elf-objdump.exe -d -M no-aliases -j .text mostCommon.out
```

```
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd>riscv64-unknown-elf-objdump.exe
-d -M no-aliases -j .text mostCommon.out
```

Секция кода теперь содержит намного большее количество строк, поэтому рассмотрим только самые важные участки:

```

00010090 <_start>:
10090:    00017197      auipc   gp,0x17
10094:    83018193      addi    gp,gp,-2000 # 268c0 <__global_pointer$>
10098:    1cc18513      addi    a0,gp,460 # 26a8c <_edata>
1009c:    23018613      addi    a2,gp,560 # 26af0 <_BSS_END__>
100a0:    40a60633      sub     a2,a2,a0
100a4:    00000593      addi    a1,zero,0
100a8:    2f4000ef      jal    ra,1039c <memset>
100ac:    00000517      auipc   a0,0x0
100b0:    1fc50513      addi    a0,a0,508 # 102a8 <__libc_fini_array>
100b4:    1ac000ef      jal    ra,10260 <atexit>
100b8:    250000ef      jal    ra,10308 <__libc_init_array>
100bc:    00012503      lw     a0,0(sp)
100c0:    00410593      addi    a1,sp,4
100c4:    00000613      addi    a2,zero,0
100c8:    07c000ef      jal    ra,10144 <main>
100cc:    1a80006f      jal    zero,10274 <exit>

000101dc <mostCommon>:
101dc:    00100793      addi    a5,zero,1
101e0:    06b7d663      bge     a5,a1,1024c <mostCommon+0x70>
101e4:    00050813      addi    a6,a0,0
101e8:    fff58e13      addi    t3,a1,-1
101ec:    00259793      slli    a5,a1,0x2
101f0:    00f50533      add    a0,a0,a5
101f4:    00000893      addi    a7,zero,0
101f8:    00000313      addi    t1,zero,0
101fc:    0100006f      jal    zero,1020c <mostCommon+0x30>
10200:    00188893      addi    a7,a7,1
10204:    00480813      addi    a6,a6,4
10208:    05c88463      beq    a7,t3,10250 <mostCommon+0x74>
1020c:    00000693      addi    a3,zero,0
10210:    02b8d463      bge    a7,a1,10238 <mostCommon+0x5c>
10214:    00082603      lw     a2,0(a6)
10218:    00080713      addi    a4,a6,0
1021c:    00000693      addi    a3,zero,0
10220:    00072783      lw     a5,0(a4)
10224:    40f607b3      sub    a5,a2,a5
10228:    0017b793      sltiu   a5,a5,1
1022c:    00f686b3      add    a3,a3,a5
10230:    00470713      addi    a4,a4,4
10234:    fea716e3      bne    a4,a0,10220 <mostCommon+0x44>
10238:    fc66c4e3      blt    a3,t1,10200 <mostCommon+0x24>
1023c:    00082783      lw     a5,0(a6)
10240:    1cf1a623      sw     a5,460(gp) # 26a8c <_edata>
10244:    00068313      addi    t1,a3,0
10248:    fb9ff06f      jal    zero,10200 <mostCommon+0x24>
1024c:    00000313      addi    t1,zero,0
10250:    1cc18793      addi    a5,gp,460 # 26a8c <_edata>
10254:    0067a223      sw     t1,4(a5)
10258:    1cc18513      addi    a0,gp,460 # 26a8c <_edata>
1025c:    00008067      jalr   zero,0(ra)

```

```

00010144 <main>:
 10144:    fc010113      addi   sp,sp,-64
 10148:    02112e23      sw     ra,60(sp)
 1014c:    000257b7      lui    a5,0x25
 10150:    98078793      addi   a5,a5,-1664 # 24980 <__clzsi2+0x50>
 10154:    0007ae83      lw     t4,0(a5)
 10158:    0047ae03      lw     t3,4(a5)
 1015c:    0087a303      lw     t1,8(a5)
 10160:    00c7a883      lw     a7,12(a5)
 10164:    0107a803      lw     a6,16(a5)
 10168:    0147a503      lw     a0,20(a5)
 1016c:    0187a583      lw     a1,24(a5)
 10170:    01c7a603      lw     a2,28(a5)
 10174:    0207a683      lw     a3,32(a5)
 10178:    0247a703      lw     a4,36(a5)
 1017c:    0287a783      lw     a5,40(a5)
 10180:    01d12223      sw     t4,4(sp)
 10184:    01c12423      sw     t3,8(sp)
 10188:    00612623      sw     t1,12(sp)
 1018c:    01112823      sw     a7,16(sp)
 10190:    01012a23      sw     a6,20(sp)
 10194:    00a12c23      sw     a0,24(sp)
 10198:    00b12e23      sw     a1,28(sp)
 1019c:    02c12023      sw     a2,32(sp)
 101a0:    02d12223      sw     a3,36(sp)
 101a4:    02e12423      sw     a4,40(sp)
 101a8:    02f12623      sw     a5,44(sp)
 101ac:    00b00593      addi   a1,zero,11
 101b0:    00410513      addi   a0,sp,4
 101b4:    028000ef      jal    ra,101dc <mostCommon>
 101b8:    00452603      lw     a2,4(a0)
 101bc:    00052583      lw     a1,0(a0)
 101c0:    00025537      lui    a0,0x25
 101c4:    9ac50513      addi   a0,a0,-1620 # 249ac <__clzsi2+0x7c>
 101c8:    2f4000ef      jal    ra,104bc <printf>
 101cc:    00000513      addi   a0,zero,0
 101d0:    03c12083      lw     ra,60(sp)
 101d4:    04010113      addi   sp,sp,64
 101d8:    00008067      jalr   zero,0(ra)

00010274 <exit>:
 10274:    ff010113      addi   sp,sp,-16
 10278:    00000593      addi   a1,zero,0
 1027c:    00812423      sw     s0,8(sp)
 10280:    00112623      sw     ra,12(sp)
 10284:    00050413      addi   s0,a0,0
 10288:    050030ef      jal    ra,132d8 <__call_exitprocs>
 1028c:    1b818793      addi   a5,gp,440 # 26a78 <_global_impure_ptr>
 10290:    0007a503      lw     a0,0(a5)
 10294:    03c52783      lw     a5,60(a0)
 10298:    00078463      beq   a5,zero,102a0 <exit+0x2c>
 1029c:    000780e7      jalr   ra,0(a5)
 102a0:    00040513      addi   a0,s0,0
 102a4:    5080f0ef      jal    ra,1f7ac <_exit>

```

```

0001f7ac <_exit>:
 1f7ac:    00000593      addi    a1,zero,0
 1f7b0:    00000613      addi    a2,zero,0
 1f7b4:    00000693      addi    a3,zero,0
 1f7b8:    00000713      addi    a4,zero,0
 1f7bc:    00000793      addi    a5,zero,0
 1f7c0:    05d00893      addi    a7,zero,93
 1f7c4:    00000073      ecall
 1f7c8:    00054463      blt    a0,zero,1f7d0 <_exit+0x24>
 1f7cc:    0000006f      jal    zero,1f7cc <_exit+0x20>
 1f7d0:    ff010113      addi    sp,sp,-16
 1f7d4:    00812423      sw     s0,8(sp)
 1f7d8:    00050413      addi    s0,a0,0
 1f7dc:    00112623      sw     ra,12(sp)
 1f7e0:    40800433      sub    s0,zero,s0
 1f7e4:    2f0000ef      jal    ra,1fad4 <__errno>
 1f7e8:    00852023      sw     s0,0(a0)
 1f7ec:    0000006f      jal    zero,1f7ec <_exit+0x40>

```

Компоновщик все переходы auipc+jalr, заменил на одну инструкцию jal и корректным адресом перехода.

Секция .comment хранит все те же сведения о GCCверсии 8.3.0 от SiFive:

```

riscv64-unknown-elf-objdump.exe -s -j .comment mostCommon.out
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd>riscv64-unknown-elf-objdump.exe
-s -j .comment mostCommon.out

mostCommon.out:      file format elf32-littleriscv

Contents of section .comment:
 0000 4743433a 20285369 46697665 20474343  GCC: (SiFive GCC
 0010 20382e33 2e302d32 3032302e 30342e31   8.3.0-2020.04.1
 0020 2920382e 332e3000                   ) 8.3.0.

```

Секция .riscv.attributes по-прежнему содержит информацию об используемой архитектуре команд RV32I:

```

riscv64-unknown-elf-objdump.exe -s -j .riscv.attributes mostCommon.out
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd>riscv64-unknown-elf-objdump.exe
-s -j .riscv.attributes mostCommon.out

mostCommon.out:      file format elf32-littleriscv

Contents of section .riscv.attributes:
 0000 411b0000 00726973 63760001 11000000  A....riscv.....
 0010 04100572 76333269 32703000           ...rv32i2p0.

```

Проанализируем таблицу символов исполняемого файла:

```
riscv64-unknown-elf-objdump.exe -t mostCommon.out
```

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

SYMBOL TABLE:

```
...
0001ce78 g    F .text  000000c8 _calloc_r
00026a8c g    .sbss   00000000 __bss_start
0001039c g    F .text  000000dc memset
00010144 g    F .text  00000098 main
00026a98 g    O .sbss   00000004 __malloc_max_total_mem
0001f638 g    F .text  00000014 __swbuf
00018b08 g    F .text  00000008 __sclose
...
00024f18 g    O .rodata       00000028 __mprec_tinytens
00018c8c g    F .text  000000b0 strcpy
0001e0c8 g    F .text  00000040 cleanup_glue
000101dc g    F .text  00000084 mostCommon
0001d7fc g    F .text  00000060 _lseek_r
000214b0 g    F .text  00000144 .hidden __getf2
000213e4 g    F .text  000000cc .hidden __eqtf2
```

Таблица символов содержит множество дополнительных вхождений, однако в целом определяет все нужные секции, метки и адреса. Функции mostCommon и main так же помечены флагом F, но в отличие от стадии ассемблирования, все они являются определенными и содержатся по корректным адресам для успешного вызова этих функций из других участков программ.

Проанализируем таблицу перемещений исполняемого файла:

```
riscv64-unknown-elf-objdump.exe -r mostCommon.out
```

```
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd>riscv64-unknown-elf-objdump.exe
-r mostCommon.out
```

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

Таблица перемещений оказывается пуста, все необходимые релокации, оптимизации и замены инструкций были успешно проведены компоновщиком.

Итогом сборки программ на языке С по шагам является исполняемый

на процессорах архитектуры RISC-V файл, решающий задачу поиска наиболее часто встречающегося числа в массиве простого числа до числа и проверяющий корректность решения этой задачи на тестовом примере.

### 3.3 Формирование статической библиотеки, разработка make-файлов для сборки библиотеки

Текст makefile:

```
output: main.o mostlib.a  
        gcc main.o mostlib.a -o output  
main.o: main.c  
        gcc -c main.c
```

```
mostlib.a: mostCommon.o mostCommon.h  
        ar -rsc mostlib.a mostCommon.o
```

mostCommon.o:

```
        gcc -c mostCommon.c
```

clean:

```
        del *.o *.a
```

Запуск Мейкфайла и процесс сборки:

```
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd\Новая папка>  
mingw32-make.exe  
gcc -c main.c  
gcc -c mostCommon.c  
ar -rsc mostlib.a mostCommon.o  
gcc main.o mostlib.a -o output
```

Директория после запуска:

```
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd\Новая папка>
dir
Том в устройстве C имеет метку OS
Серийный номер тома: 3638-0C85

Содержимое папки C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\
4\3rd\Новая папка

05.11.2021  01:41    <DIR>      .
05.11.2021  01:41    <DIR>      ..
04.11.2021  20:59          324 main.c
05.11.2021  01:41          1 034 main.o
05.11.2021  01:38          232 makefile
04.11.2021  21:00          913 mostCommon.c
04.11.2021  21:00          125 mostCommon.h
05.11.2021  01:41          857 mostCommon.o
05.11.2021  01:41          1 006 mostlib.a
05.11.2021  01:41          41 573 output.exe
                           8 файлов        46 064 байт
                           2 папок   42 217 148 416 байт свободно
```

Результат работы exe файла для массива 1, 2, 2, 4, 4, 80, 7, 80, 9, 4, 4:

```
C:\Users\aleks\Documents\учеба\3_сем\lowlvl_prog\labs\4\3rd\Новая папка>output.exe
The most common value: 4.
Amount of repetitions: 4.
```

## 4. Выводы

В данной лабораторной работе мы познакомились с процессом сборки проекта на языке С.

Он состоит из:

1. **Препроцессирования:** исходного .c файл препроцессируем в .i файл
2. **Компиляции:** полученный .i файл компилируется в файл ассемблера .s
3. **Ассемблирования:** файл .s ассемблируется в объектный файл .o
4. **Компоновки:** объектный файл .o компонуется в исполняемый файл

Также мы ознакомились в *makefile*'ами, которые упрощают процесс сборки. Утилита Make позволяет собирать проекты, состоящие из большого количества файлов, вместо использования PS/SH скриптов, и прописывания файлов вручную.