

**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 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 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

Main.c:

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

int main() {
    unsigned int result;
    unsigned int array[10] = {0, 1, 1, 1, 2, 2, 2, 1, 3, 4};

    size_t arraySize = sizeof(array)/sizeof(array[0]);
    result = mostCommon(array, arraySize);
    printf("result = %u", result);
    return 0;
}
```

mostCommon.h:

```
#include <stdio.h>

unsigned int mostCommon(unsigned int *array, size_t arraySize);
```

mostCommon.c:

```
#include <stdio.h>

unsigned int mostCommon(unsigned int *array, size_t arraySize) {
    int currentCounter = 0;
    int maxCounter = 0;
    unsigned int result = 0;

    for (int i = 0; i < arraySize - 1; i++) {
```

```

for (int j = i; j < arraySize; j++) {
    if (array[i] == array[j]) {
        currentCounter++;
    }
}
if (currentCounter > maxCounter) {
    maxCounter = currentCounter;
    result = array[i];
}
currentCounter = 0;
}
return result;
}

```

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\USER\Documents\экзамены\lab41>riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -O1 -E main.c -o main.i
C:\Users\USER\Documents\экзамены\lab41>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.

В начале файла main.i содержится порядка 1200 строк с инструкциями по линковке stdio.h к проекту, а затем следует код на С, который мало отличаются от исходных версий программ:

Main.i:

```
# 797 "c:\\users\\user\\documents\\lab4files\\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-mingw32\\riscv64-unknown-elf\\include\\stdio.h" 3
```

```
# 2 "mostCommon.h" 2
```

```
# 3 "mostCommon.h"
unsigned int mostCommon(unsigned int *array, size_t arraySize);
# 2 "main.c" 2
```

```
int main()
{
    unsigned int result;
    unsigned int array[10] = {0, 1, 1, 1, 2, 2, 2, 1, 3, 4};

    size_t arraySize = sizeof(array)/sizeof(array[0]);
    result = mostCommon(array, arraySize);
    printf("result = %u", result);
    return 0;
}
```

**mostCommon.i**

```
# 797 "c:\\users\\user\\documents\\lab4files\\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-w64-
mingw32\\riscv64-unknown-elf\\include\\stdio.h" 3
```

```
# 2 "mostCommon.c" 2
```

```
# 3 "mostCommon.c"
int mostCommon(unsigned int *array, size_t arraySize) {
    int currentCounter = 0;
    int maxCounter = 0;
    int result = 0;

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

Появившиеся нестандартные директивы, начинающиеся с символа “#”, используются для передачи информации об исходном тексте из препроцессора в компилятор. Так, в файле “main.i” вторая директива «# 2 “main.c”» информирует компилятор о том, что следующая строка является результатом обработки строки 2 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\USER\Documents\экзамены\lab41>riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -O1 -S mostCommon.i -o mostCommon.s
C:\Users\USER\Documents\экзамены\lab41>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:

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   t3,0(a5)
    lw   t1,4(a5)
    lw   a7,8(a5)
    lw   a6,12(a5)
    lw   a0,16(a5)
    lw   a1,20(a5)
    lw   a2,24(a5)
    lw   a3,28(a5)
    lw   a4,32(a5)
    lw   a5,36(a5)
    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 a1,10
addi a0,sp,8
call mostCommon
mv a1,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 0
.word 1
.word 1
.word 1
.word 2
.word 2
.word 2
.word 1
.word 3
.word 4
.section .rodata.str1.4,"aMS",@progbits,1
.align 2

```

.LC1:

```

.string "result = %u"
.ident "GCC: (SiFive GCC 8.3.0-2020.04.1) 8.3.0"

```

Массив array хранится под меткой LC0, в a0 – элементы массива, в a1 – длина массива и вызывается mostCommon

**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:

```

mv a6,a0
addi a0,a1,-1
beq a0,zero,.L1
mv a7,a6

```

```

        mv    t4,a0
        slli  a5,a1,2
        add   a6,a6,a5
        li    t1,0
        li    a0,0
        li    t3,0
        j     .L3
.L6:
        addi  t1,t1,1
        addi  a7,a7,4
        beq   t1,t4,.L1
.L3:
        li    a3,0
        bleu  a1,t1,.L8
        lw    a2,0(a7)
        mv    a4,a7
        li    a3,0
.L5:
        lw    a5,0(a4)
        sub   a5,a2,a5
        seqz  a5,a5
        add   a3,a3,a5
        addi  a4,a4,4
        bne   a6,a4,.L5
.L8:
        ble   a3,t3,.L6
        lw    a0,0(a7)
        mv    t3,a3
        j     .L6
.L1:
        ret
        .size mostCommon, .-mostCommon
        .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\USER\Documents\экзамены\lab41>riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -c mostCommon.s -o mostCommon.o
C:\Users\USER\Documents\экзамены\lab41>riscv64-unknown-elf-gcc.exe -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:

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

```

C:\Users\USER\Documents\экзамены\lab41>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\USER\Documents\экзамены\lab41>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 флаг HAS\_RELOC).

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

```

C:\Users\USER\Documents\экзамены\lab41>riscv64-unknown-elf-objdump.exe -h 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        00000028  00000000  00000000  000000c8  2**2
    CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .rodata.str1.4 0000000c  00000000  00000000  000000f0  2**2
    CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .comment       00000029  00000000  00000000  000000fc  2**0
    CONTENTS, READONLY
  6 .riscv.attributes 0000001c  00000000  00000000  00000125  2**0
    CONTENTS, READONLY

C:\Users\USER\Documents\экзамены\lab41>riscv64-unknown-elf-objdump.exe -h mostCommon.o

mostCommon.o:      file format elf32-littleriscv

Sections:
Idx Name          Size      VMA           LMA           File off  Algn
  0 .text          00000078  00000000  00000000  00000034  2**2
    CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data          00000000  00000000  00000000  000000ac  2**0
    CONTENTS, ALLOC, LOAD, DATA
  2 .bss           00000000  00000000  00000000  000000ac  2**0
    ALLOC
  3 .comment       00000029  00000000  00000000  000000ac  2**0
    CONTENTS, READONLY
  4 .riscv.attributes 0000001c  00000000  00000000  000000d5  2**0
    CONTENTS, READONLY

```



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

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

```
C:\Users\USER\Documents\экзамены\lab41>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     d  .rodata 00000000 .LANCHOR0
00000000 l     d  .rodata.str1.4 00000000 .rodata.str1.4
00000000 l     d  .rodata.str1.4 00000000 .LC1
00000000 l     d  .comment 00000000 .comment
00000000 l     d  .riscv.attributes 00000000 .riscv.attributes
00000000 g     F  .text 00000094 main
00000000      *UND* 00000000 mostCommon
00000000      *UND* 00000000 printf

C:\Users\USER\Documents\экзамены\lab41>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
00000074 l     d  .text 00000000 .L1
00000038 l     d  .text 00000000 .L3
00000064 l     d  .text 00000000 .L8
0000004c l     d  .text 00000000 .L5
0000002c l     d  .text 00000000 .L6
00000000 l     d  .comment 00000000 .comment
00000000 l     d  .riscv.attributes 00000000 .riscv.attributes
00000000 g     F  .text 00000078 mostCommon
```

В каждой таблице только один 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

```
00050813      mv      a6,a0
4:      fff58513      addi     a0,a1,-1
8:      06050663      beqz     a0,74 <.L1>
c:      00080893      mv      a7,a6
10:     00050e93      mv      t4,a0
14:     00259793      slli     a5,a1,0x2
18:     00f80833      add      a6,a6,a5
1c:     00000313      li       t1,0
20:     00000513      li       a0,0
24:     00000e13      li       t3,0
28:     0100006f      j        38 <.L3>
```

0000002c <.L6>:

```
2c:     00130313      addi     t1,t1,1
30:     00488893      addi     a7,a7,4
34:     05d30063      beq      t1,t4,74 <.L1>
```

00000038 <.L3>:

```
38:     00000693      li       a3,0
3c:     02b37463      bgeu     t1,a1,64 <.L8>
40:     0008a603      lw       a2,0(a7)
44:     00088713      mv       a4,a7
48:     00000693      li       a3,0
```

0000004c <.L5>:

```
4c:     00072783      lw       a5,0(a4)
50:     40f607b3      sub      a5,a2,a5
54:     0017b793      seqz     a5,a5
58:     00f686b3      add      a3,a3,a5
5c:     00470713      addi     a4,a4,4
60:     fee816e3      bne      a6,a4,4c <.L5>
```

00000064 <.L8>:

```
64:     fcde54e3      bge      t3,a3,2c <.L6>
68:     0008a503      lw       a0,0(a7)
6c:     00068e13      mv       t3,a3
70:     fbdff06f      j        2c <.L6>
```

00000074 <.L1>:

```
74:     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:   0007ae03      lw      t3,0(a5) # 0 <main>
14:   0047a303      lw      t1,4(a5)
18:   0087a883      lw      a7,8(a5)
1c:   00c7a803      lw      a6,12(a5)
20:   0107a503      lw      a0,16(a5)
24:   0147a583      lw      a1,20(a5)
28:   0187a603      lw      a2,24(a5)
2c:   01c7a683      lw      a3,28(a5)
30:   0207a703      lw      a4,32(a5)
34:   0247a783      lw      a5,36(a5)
38:   01c12423      sw      t3,8(sp)
3c:   00612623      sw      t1,12(sp)
40:   01112823      sw      a7,16(sp)
44:   01012a23      sw      a6,20(sp)
48:   00a12c23      sw      a0,24(sp)
4c:   00b12e23      sw      a1,28(sp)
50:   02c12023      sw      a2,32(sp)
54:   02d12223      sw      a3,36(sp)
58:   02e12423      sw      a4,40(sp)
5c:   02f12623      sw      a5,44(sp)
60:   00a00593      li      a1,10
64:   00810513      addi     a0,sp,8
68:   00000097      auipc    ra,0x0
6c:   000080e7      jalr     ra # 68 <main+0x68>
70:   00050593      mv      a1,a0
74:   00000537      lui      a0,0x0
78:   00050513      mv      a0,a0
7c:   00000097      auipc    ra,0x0
80:   000080e7      jalr     ra # 7c <main+0x7c>
84:   00000513      li      a0,0
88:   03c12083      lw      ra,60(sp)
8c:   04010113      addi     sp,sp,64
90:   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\USER\Documents\экзамены\lab41>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\USER\Documents\экзамены\lab41>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
```

```
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.
```

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\USER\Documents\экзамены\lab41>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\USER\Documents\экзамены\lab41>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_P` указывает, 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\USER\Documents\экзамены\lab41>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              000148f0  00010074  00010074  00000074  2**2
    CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .rodata            00000d14  00024968  00024968  00014968  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            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.

Изучим содержимое секции “.text”:

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

В результате выполнения получили файл “*mostCommon.ds*”. Изучим его.

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: 22818613      addi  a2,gp,552 # 26ae8 <__BSS_END__>
100a0: 40a60633      sub   a2,a2,a0
```

100a4:	00000593	addi a1,zero,0
100a8:	2dc000ef	jal ra,10384 <memset>
100ac:	00000517	auipc a0,0x0
100b0:	1e450513	addi a0,a0,484 # 10290 <__libc_fini_array>
100b4:	194000ef	jal ra,10248 <atexit>
100b8:	238000ef	jal ra,102f0 <__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:	1900006f	jal zero,1025c <exit>

“\_start” – “точка входа” в нашу программу. Код, начинающийся с метки “\_start” обеспечивает инициализацию памяти, регистров процессора и среды времени выполнения, после чего передаёт управление определённой нами функции main.

0001025c <exit>:

1025c:	ff010113	addi sp,sp,-16
10260:	00000593	addi a1,zero,0
10264:	00812423	sw s0,8(sp)
10268:	00112623	sw ra,12(sp)
1026c:	00050413	addi s0,a0,0
10270:	050030ef	jal ra,132c0 <__call_exitprocs>
10274:	1b818793	addi a5,gp,440 # 26a78 <_global_impure_ptr>
10278:	0007a503	lw a0,0(a5)
1027c:	03c52783	lw a5,60(a0)
10280:	00078463	beq a5,zero,10288 <exit+0x2c>
10284:	000780e7	jalr ra,0(a5)
10288:	00040513	addi a0,s0,0
1028c:	5080f0ef	jal ra,1f794 <_exit>

Можно видеть, что в конце “exit” управление передается на символ “\_exit”

### 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
```

Сборка с помощью Makefile:

```
C:\Users\USER\Documents\экзамены\lab41\makekмак>mingw32-make.exe
gcc -c main.c
gcc -c mostCommon.c
ar -rsc mostlib.a mostCommon.o
gcc main.o mostlib.a -o output

C:\Users\USER\Documents\экзамены\lab41\makekмак>dir
Том в устройстве C не имеет метки.
Серийный номер тома: 1288-F267

Содержимое папки C:\Users\USER\Documents\экзамены\lab41\makekмак

20.12.2022  16:18    <DIR>          .
20.12.2022  16:18    <DIR>          ..
19.12.2022  20:42             297 main.c
20.12.2022  16:18             970 main.o
20.12.2022  16:18             231 makefile
20.12.2022  03:38             540 mostCommon.c
19.12.2022  20:42              83 mostCommon.h
20.12.2022  16:18             796 mostCommon.o
20.12.2022  16:18             944 mostlib.a
20.12.2022  16:18           41 542 output.exe
```

Демонстрация работы программы:

```
C:\Users\USER\Documents\экзамены\lab41\makekмак>output.exe
result = 1
```



Очистка:

```
C:\Users\USER\Documents\экзамены\lab41\makekmake>mingw32-make.exe clean
del *.o *.a

C:\Users\USER\Documents\экзамены\lab41\makekmake>dir
Том в устройстве C не имеет метки.
Серийный номер тома: 1288-F267

Содержимое папки C:\Users\USER\Documents\экзамены\lab41\makekmake

20.12.2022  16:20    <DIR>          .
20.12.2022  16:20    <DIR>          ..
19.12.2022  20:42             297 main.c
20.12.2022  16:18             231 makefile
20.12.2022  03:38             540 mostCommon.c
19.12.2022  20:42              83 mostCommon.h
20.12.2022  16:18           41 542 output.exe
```

Что происходит в Makefile:

1. Создаём объектный файл *main.o* из исходного *main.c*
2. Создаём объектный файл *mostCommon.o* из исходного *mostCommon.c*
3. Архивируем объектный файл *mostCommon.o* (создаём статическую библиотеку *mostlib.a*)
4. Компонуем статическую библиотеку *mostlib.a* с объектным файлом *main.o* и получаем исполняемый файл *output*

ВЫВОД

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

Он состоит из:

1. **Препроцессирования:** исходного .c файл препроцессуем 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.