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

Отчёт по лабораторной работе № 4

Дисциплина: Низкоуровневое программирование
Тема: Раздельная компиляция
Вариант: 9

Выполнил студент гр. 3530901/00002 _____М.А. Разин
(подпись)
Принял преподаватель _____ Д.С. Степанов
(подпись)

“____”_____ 2021 г.

Санкт-Петербург
2021

Задача:

1. Изучить методические материалы, опубликованные на сайте курса.
2. Установить пакет средств разработки “SiFive GNU Embedded Toolchain” для RISC-V.
3. 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.
4. 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.
5. 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.

Вариант задания:

Вариант: 9 - Расчет биномиальных коэффициентов для данного показателя по треугольнику Паскаля.

1. Программа на C

```
#ifndef FINDCOEFFS_H
#define FINDCOEFFS_H
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
int *findCoeffs(int power);
#endif
```

Рис 1.1 Заголовочный файл findCoeffs.h

Директивы, указанные в начале файла, предназначены для однократного включения файла в проект.

```c
#include "findCoeffs.h"
int *findCoeffs(int power) {
    int a = 1;
    int *arr = (int*) calloc( Count: power+1,  Size: sizeof(int));
    for (int t=1; t<= power+1; t++){
        arr[t-1]=a;
        a=a*(power + 1 - t)/t;
    }
    return arr;
}
```

Рис 1.2 Файл findCoeffs.c

Так как размерность массива коэффициентов заранее неизвестна, используется функция динамического выделения памяти: calloc(число элементов, размер элемента в байтах), которая находит в оперативной памяти непрерывный участок требуемой длины и возвращает начальный адрес этого участка.

```c
#include "findCoeffs.h"

int main() {
    int power = 5;
    int *res = findCoeffs(power);
    for (int i = 0; i <= power; i++)
    {
        printf( Format: "%i \n", res[i]);
    }
    return 0;
}
```

Рис 1.3 Файл main.c (тестовая программа).

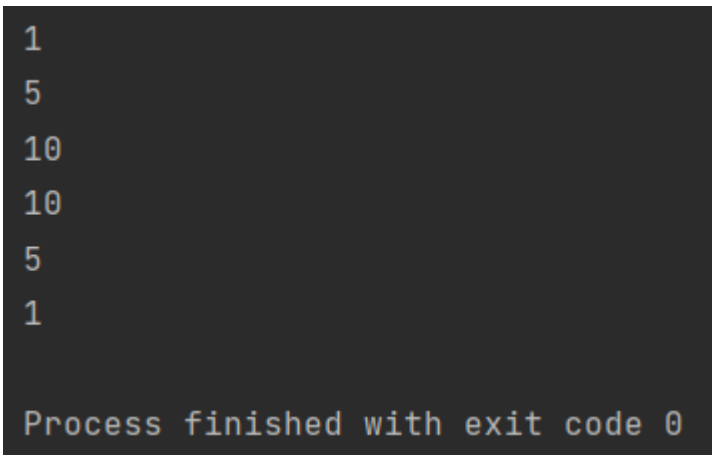


Рис 1.4 Результат работы программы.

2. Сборка по шагам

- **Препроцессирование**

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -E main.c -o main.i -v -E

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -E findCoeffs.c -o findCoeffs.i -v -E

Результат препроцессирования находится в файлах main.i и findCoeffs.i

```
# 6 "findCoeffs.h" 2

# 6 "findCoeffs.h"
int *findCoeffs(int power);
# 2 "main.c" 2
int main() {
    int power = 5;
    int *res = findCoeffs(power);
    for (int i = 0; i <= power; i++)
    {
        printf("%i \n", res[i]);
    }
    return 0;
}
```

Рис 2.1 Фрагмент файла main.i

```
# 6 "findCoeffs.h" 2

# 6 "findCoeffs.h"
int *findCoeffs(int power);
# 2 "findCoeffs.c" 2
int *findCoeffs(int power) {
    int a = 1;
    int *arr = (int*) calloc(power+1, sizeof(int));
    for (int t=1; t<= power+1; t++){
        arr[t-1]=a;
        a=a*(power + 1 - t)/t;
    }
    return arr;
}
```

Рис 2.2 Фрагмент файла findCoeffs.i

- **Компиляция**

  riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -S -fpreprocessed main.i -o main.s

  riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -S -fpreprocessed findCoeffs.i -o findCoeffs.s

На этом этапе препроцессированный файл компилируется в asm RISC-V. Можно заметить, что main.s содержит обращение к подпрограмме findCoeffs(с сохранением адреса возврата в стеке).

```
    .file   "main.c"
    .option nopic
    .attribute arch, "rv64i2p0_a2p0_c2p0"
    .attribute unaligned_access, 0
    .attribute stack_align, 16
    .text
    .section    .rodata.str1.8,"aMS",@progbits,1
    .align  3
.LC0:
    .string "%i \n"
    .text
    .align  1
    .globl  main
    .type   main, @function
main:
    addi    sp,sp,-32
    sd  ra,24(sp)
    sd  s0,16(sp)
    sd  s1,8(sp)
    sd  s2,0(sp)
    li  a0,5
    call    findCoeffs
    mv  s0,a0
    addi    s2,a0,24
    lui s1,%hi(.LC0)
```

```
.L2:
    lw  a1,0(s0)
    addi    a0,s1,%lo(.LC0)
    call    printf
    addi    s0,s0,4
    bne s0,s2,.L2
    li  a0,0
    ld  ra,24(sp)
    ld  s0,16(sp)
    ld  s1,8(sp)
    ld  s2,0(sp)
    addi    sp,sp,32
    jr  ra
    .size   main, .-main
    .ident  "GCC: (SiFive GCC-Metal 10.2.0-2020.12.8) 10.2.0"
```

Рис 2.3 Файл main.s

```
    .file   "findCoeffs.c"
    .option nopic
    .attribute arch, "rv64i2p0_a2p0_c2p0"
    .attribute unaligned_access, 0
    .attribute stack_align, 16
    .text
    .globl  __muldi3
    .globl  __divdi3
    .align  1
    .globl  findCoeffs
    .type   findCoeffs, @function
```

```
findCoeffs:
    addi    sp,sp,-48
    sd  ra,40(sp)
    sd  s0,32(sp)
    sd  s1,24(sp)
    sd  s2,16(sp)
    sd  s3,8(sp)
    sd  s4,0(sp)
    mv  s2,a0
    addiw   s0,a0,1
    li  a1,4
    mv  a0,s0
    call    calloc
    mv  s4,a0
    ble s0,zero,.L1
    mv  s1,a0
    addiw   s3,s2,2
    li  s0,1
    li  a0,1
    addiw   s2,s2,1
.L3:
    sw  a0,0(s1)
    subw    a1,s2,s0
    call    __muldi3
    mv  a1,s0
    sext.w  a0,a0
    call    __divdi3
    sext.w  a0,a0
    addiw   s0,s0,1
    addi    s1,s1,4
    bne s0,s3,.L3
```

Рис 2.4 Фрагмент файла findCoeff.s

- **Ассемблирование**

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -v -c main.s -o main.o
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -v -c findCoeffs.s -o findCoeffs.o

Команда для вывода заголовков секций файла main.o:
riscv64-unknown-elf-objdump -h main.o

```
Sections:
Idx Name          Size      VMA               LMA               File off  Algn
  0 .text         00000040  0000000000000000  0000000000000000  00000040  2**1
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  0000000000000000  0000000000000000  00000080  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  0000000000000000  0000000000000000  00000080  2**0
                  ALLOC
  3 .rodata.str1.8 00000005  0000000000000000  0000000000000000  00000080  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .comment      00000031  0000000000000000  0000000000000000  00000085  2**0
                  CONTENTS, READONLY
  5 .riscv.attributes 00000026  0000000000000000  0000000000000000  000000b6  2**0
                  CONTENTS, READONLY
```

Рис 2.5 Заголовки секций файла main.o

```
Sections:
Idx Name          Size      VMA               LMA               File off  Algn
  0 .text         00000000  0000000000000000  0000000000000000  00000040  2**1
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000000  0000000000000000  0000000000000000  00000040  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  0000000000000000  0000000000000000  00000040  2**0
                  ALLOC
  3 .comment      00000031  0000000000000000  0000000000000000  00000040  2**0
                  CONTENTS, READONLY
  4 .riscv.attributes 00000026  0000000000000000  0000000000000000  00000071  2**0
                  CONTENTS, READONLY
```

Рис 2.6 Заголовки секций файла findCoeffs.o

Команда для вывода таблицы символов:
riscv64-unknown-elf-objdump -t findCoeffs.o main.o

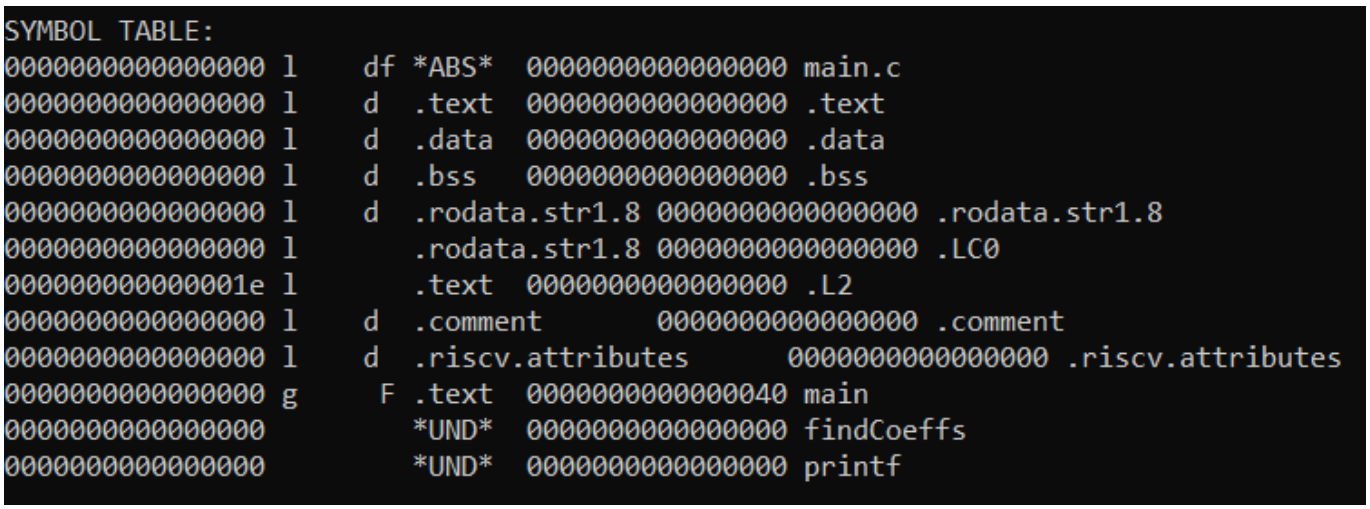


Рис 2.7 Таблица символов файла main.o

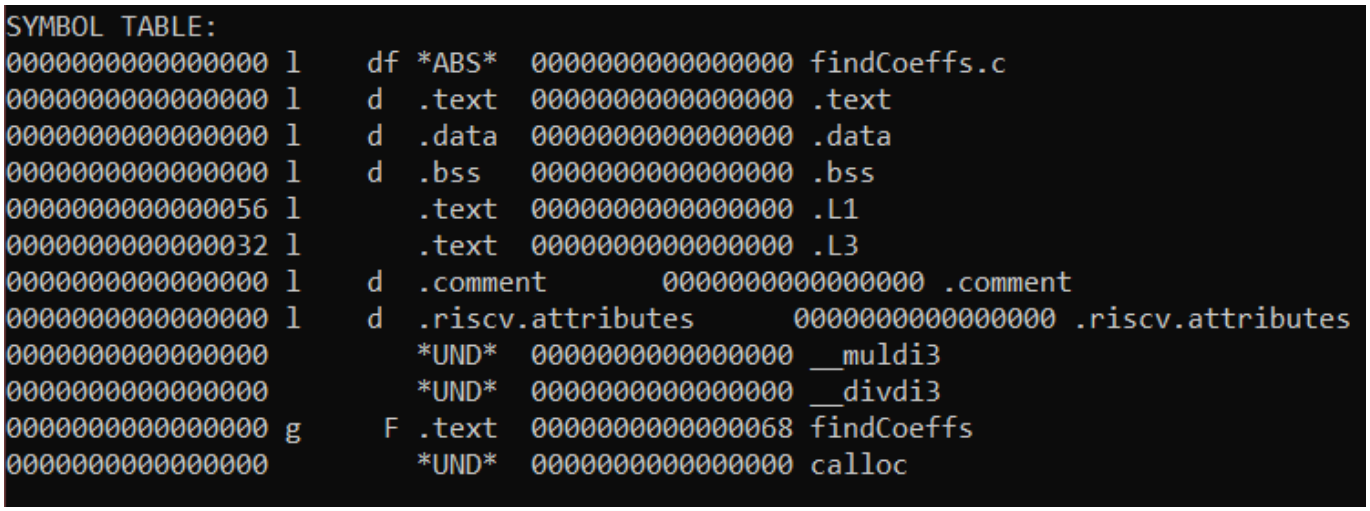
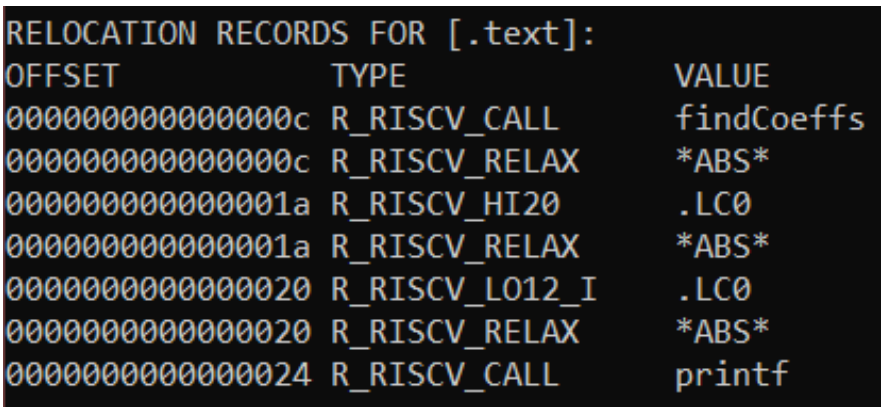


Рис 2.8 Таблица символов файла findCoeffs.o

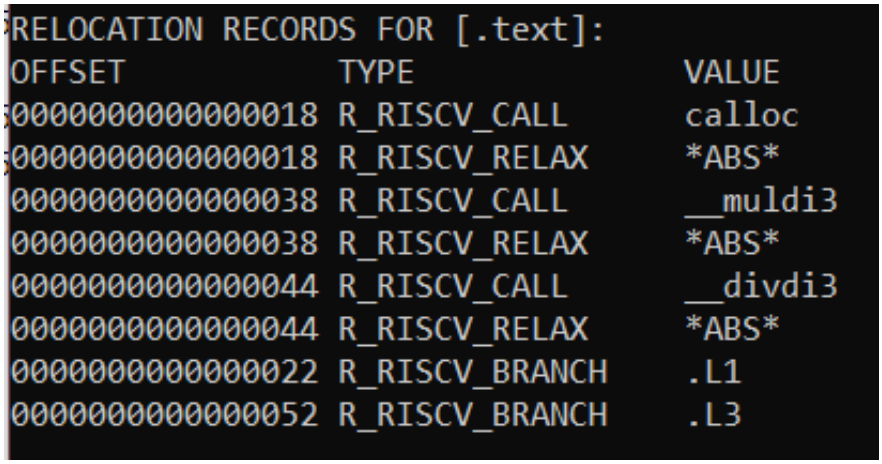
Как и следовало ожидать, таблицы содержат один глобальный (флаг “g”) символ типа «функция» (“F”) – символ “main” и символ “findCoeffs”. Тип *UND* указывает на то, что символ findCoeffs (printf, calloc и т.д) был использован в ассемблерном коде, но не был определен, поэтому ассемблер указал, что символ должен быть определен где-то ещё.

Команда получения таблицы перемещений:
riscv64-unknown-elf-objdump -r findCoeffs.o main.o



```
RELOCATION RECORDS FOR [.text]:
OFFSET           TYPE              VALUE
000000000000000c R_RISCV_CALL      findCoeffs
000000000000000c R_RISCV_RELAX     *ABS*
000000000000001a R_RISCV_HI20      .LC0
000000000000001a R_RISCV_RELAX     *ABS*
0000000000000020 R_RISCV_LO12_I    .LC0
0000000000000020 R_RISCV_RELAX     *ABS*
0000000000000024 R_RISCV_CALL      printf
```

Рис 2.9 Таблица перемещений файла main.o



```
RELOCATION RECORDS FOR [.text]:
OFFSET           TYPE              VALUE
0000000000000018 R_RISCV_CALL      calloc
0000000000000018 R_RISCV_RELAX     *ABS*
0000000000000038 R_RISCV_CALL      __muldi3
0000000000000038 R_RISCV_RELAX     *ABS*
0000000000000044 R_RISCV_CALL      __divdi3
0000000000000044 R_RISCV_RELAX     *ABS*
0000000000000022 R_RISCV_BRANCH    .L1
0000000000000052 R_RISCV_BRANCH    .L3
```

Рис 2.10 Таблица перемещений файла findCoeffs.o

- **Компановка**

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -v main.o findCoeffs.o -o main.out

Доступ: riscv64-unknown-elf-objdump -j .text -d -M no-aliases main.out > a.ds

```
00000000000100c4 <_start>:
   100c4:   0000f197            auipc   gp,0xf
   100c8:   3cc18193            addi    gp,gp,972 # 1f490 <__global_pointer$>
   100cc:   76818513            addi    a0,gp,1896 # 1fbf8 <__malloc_max_total_mem>
   100d0:   00010617            auipc   a2,0x10
   100d4:   bc060613            addi    a2,a2,-1088 # 1fc90 <__BSS_END__>
   100d8:   8e09                    c.sub   a2,a0
   100da:   4581                    c.li    a1,0
   100dc:   1af000ef            jal ra,10a8a <memset>
   100e0:   00009517            auipc   a0,0x9
   100e4:   5aa50513            addi    a0,a0,1450 # 1968a <atexit>
   100e8:   c519                    c.beqz  a0,100f6 <_start+0x32>
   100ea:   00003517            auipc   a0,0x3
   100ee:   eb250513            addi    a0,a0,-334 # 12f9c <__libc_fini_array>
   100f2:   598090ef            jal ra,1968a <atexit>
   100f6:   304000ef            jal ra,103fa <__libc_init_array>
   100fa:   4502                    c.lwsp  a0,0(sp)
   100fc:   002c                    c.addi4spn  a1,sp,8
   100fe:   4601                    c.li    a2,0
   10100:   056000ef            jal ra,10156 <main>
   10104:   ace1                    c.j 103dc <exit>
```

Рис 2.11 Фрагмент <_start> исполняемого файла main.out

Код с метки <_start> обеспечивает инициализацию памяти, регистров процессора и среды времени выполнения, после чего передает управление main.

```
0000000000010156 <main>:
   10156:   1101                    c.addi  sp,-32
   10158:   ec06                    c.sdsp  ra,24(sp)
   1015a:   e822                    c.sdsp  s0,16(sp)
   1015c:   e426                    c.sdsp  s1,8(sp)
   1015e:   e04a                    c.sdsp  s2,0(sp)
   10160:   4515                    c.li    a0,5
   10162:   02a000ef            jal ra,1018c <findCoeffs>
   10166:   842a                    c.mv    s0,a0
   10168:   01850913            addi    s2,a0,24
   1016c:   64f5                    c.lui   s1,0x1d
   1016e:   400c                    c.lw    a1,0(s0)
   10170:   b3048513            addi    a0,s1,-1232 # 1cb30 <__clzdi2+0x30>
   10174:   1e7000ef            jal ra,10b5a <printf>
   10178:   0411                    c.addi  s0,4
   1017a:   ff241ae3            bne s0,s2,1016e <main+0x18>
   1017e:   4501                    c.li    a0,0
   10180:   60e2                    c.ldsp  ra,24(sp)
   10182:   6442                    c.ldsp  s0,16(sp)
   10184:   64a2                    c.ldsp  s1,8(sp)
   10186:   6902                    c.ldsp  s2,0(sp)
   10188:   6105                    c.addi16sp  sp,32
   1018a:   8082                    c.jr    ra
```

Рис 2.12 Фрагмент <main> исполняемого файла main.out

3. Создание статической библиотеки и make-файлов

Объединю findCoeffs.c в libFindCoeffs.a:
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -c findCoeffs.c -o findCoeffs.o
riscv64-unknown-elf-ar -rsc libFindCoeffs.a findCoeffs.o

Команда для просмотра списка символов созданной библиотеки:
riscv64-unknown-elf-nm libFindCoefffs.a

```
findCoeffs.o:
0000000000000056 t .L1
0000000000000032 t .L3
                 U __divdi3
                 U __muldi3
                 U calloc
0000000000000000 T findCoeffs
```

Рис 3.1 Список символов библиотеки libFindCoeffs.a

Статическая библиотека является набором объектных файлов, среди которых компоновщик выбирает «полезные» для данной программы. Соберу исполняемый файл программы, используя полученную библиотеку:

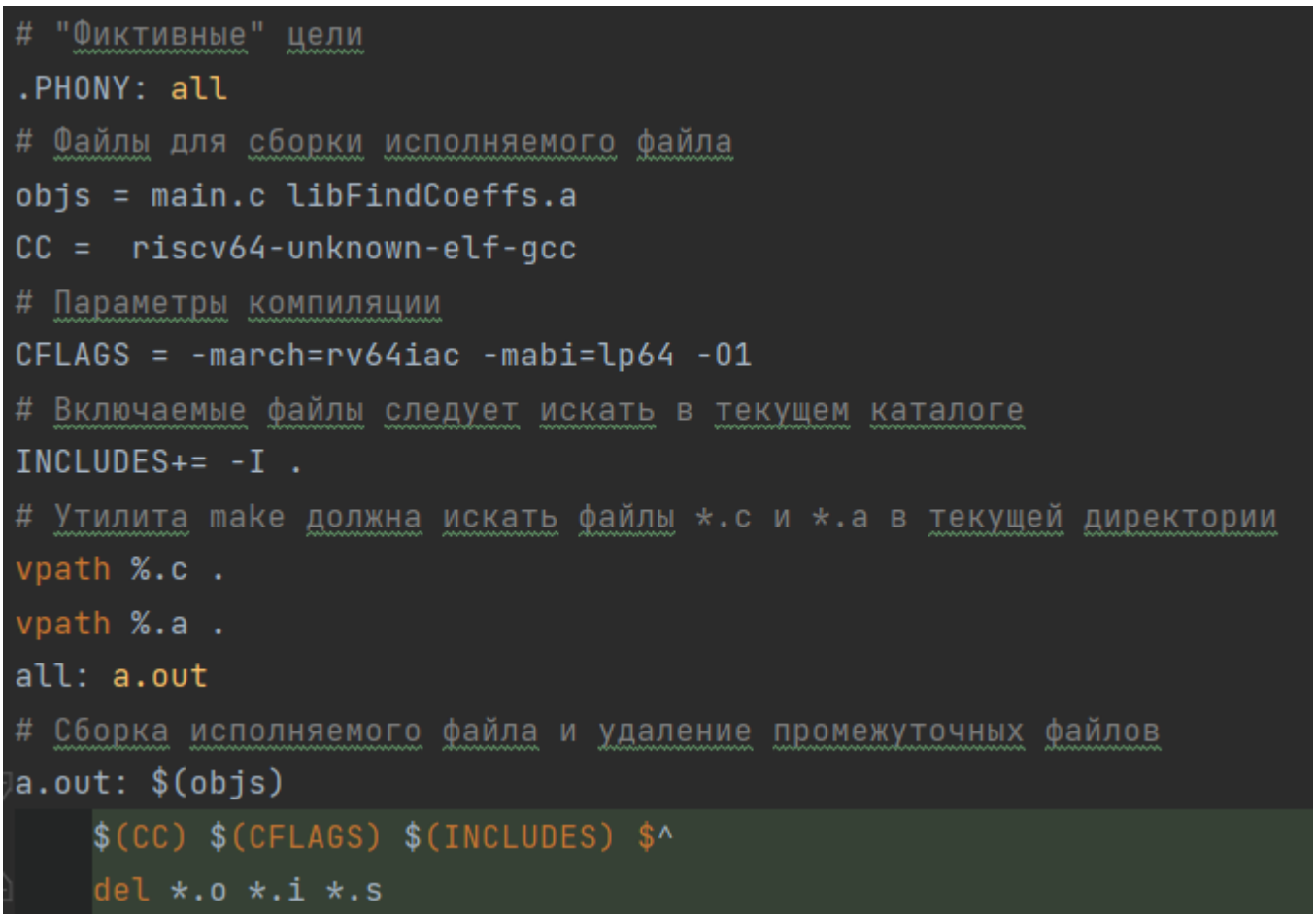
riscv64-unknown-elf-gcc -O1 --save-temps main.c libfindCoeffs.a

Учитывая тот факт, что при изменении одного файла функции приходится пересобирать весь проект, удобнее воспользоваться make файлом, позволяющим пересобирать только определенные компоненты программы, которые были изменены.

Каждый make файл состоит из:
Цель: зависимости
[tab] команда

```
# "Фиктивные" цели
.PHONY: all
# Файлы для сборки исполняемого файла
objs = findCoeffs.c
CC =  riscv64-unknown-elf-gcc
AR = riscv64-unknown-elf-ar
# Параметры компиляции
CFLAGS = -march=rv64iac -mabi=lp64 -O1 -c
# Включаемые файлы следует искать в текущем каталоге
INCLUDES+= -I .
# Утилита make должна искать файлы *.c и *.h в текущей директории
vpath %.c .
vpath %.h .
# Сборка объектных файлов
%.o : $(objs)
    $(CC) $(CFLAGS) $(INCLUDES) $< -o $@
all : libFindCoeffs.a
libFindCoeffs.a : findCoeffs.o
    $(AR) -rsc $@ $<
    del *.o *.i *.s
```

Рис. 3.2 Файл Makelib для создания статической библиотеки.



```
# "Фиктивные" цели
.PHONY: all
# Файлы для сборки исполняемого файла
objs = main.c libFindCoeffs.a
CC =  riscv64-unknown-elf-gcc
# Параметры компиляции
CFLAGS = -march=rv64iac -mabi=lp64 -O1
# Включаемые файлы следует искать в текущем каталоге
INCLUDES+= -I .
# Утилита make должна искать файлы *.c и *.a в текущей директории
vpath %.c .
vpath %.a .
all: a.out
# Сборка исполняемого файла и удаление промежуточных файлов
a.out: $(objs)
	$(CC) $(CFLAGS) $(INCLUDES) $^
	del *.o *.i *.s
```

Рис. 3.3 Makefile для сборки исполняемого файла.

Для запуска Makelib и Makefile использую программу mingw32-make.exe, команды:

mingw32-make -f Makelib
mingw32-make -f Makefile

Вывод:

В ходе лабораторной работы была выполнена пошаговая раздельная компиляция программы на языке С. Также была создана статическая библиотека и произведена сборка программы с помощью Makefile.

Список использованных источников:

http://kspt.icc.spbstu.ru/media/files/2018/lowlevelprog/cle.pdf

https://www.sifive.com/software

https://guides.hexlet.io/makefile-as-task-runner/

https://www.youtube.com/watch?v=HPni4P9ahHo&t