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. 3530901/90003 В.С. Андрианов

(подпись)

Преподаватель А.О. Алексюк

(подпись)

“ ” 2021 г.

Санкт-Петербург

2021

**Содержание**

**1. Техническое задание…………………………………………………………...3**

**2. Метод решения……...………………………………………………………. 3**

**3. Программ****ы на ЯП C…………………………………………………………3**

**4. Препроцессирование …………………………………………………………4**

**5. Компиляция файлов fibonacci.i и test.i ……………………………………7**

**6. Ассемблирование файлов fibonacci.s и test.s ………………………………9**

**7. Создание статической библиотеки и make-файлов ……………………27**

**8. Выводы…………………………………………………………………………31**

**1.** **Техническое задание**

1. Разработать на языке Си программу, которая будет выводить заданное число последовательности Фибоначчи.
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 исполнимом 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. Метод решения**

Алгоритм расчета заданного члена последовательности Фибоначчи приведен на ЯП Kotlin:

Листинг 1. Расчет заданного члена последовательности Фибоначчи

*fun* fib() {  
 *var n* = 5  
 *var f* = 1  
 *var s* = 0  
 *var t* = 0  
 *while* (*n* > 0) {  
 *t* = *f* + *s  
 f* = *s  
 s* = *t  
 n*--  
 }  
 print(*t*)  
}

**3. Программы на ЯП C**

Листинг 2. Программа fibonacci.c

#include "fibonacci.h"  
  
int fibonacci(int n) {  
 int first = 1;  
 int second = 0;  
 int third = 0;  
 while (n > 0) {  
 third = first + second;  
 first = second;  
 second = third;  
 n--;  
 }  
 return third;  
}

Листинг 3. Заголовочный файл fibonacci.h

#ifndef LOW\_LAB\_4\_FIBONACCI\_H  
#define LOW\_LAB\_4\_FIBONACCI\_H  
  
int fibonacci(int n);  
  
#endif

Листинг 4. Программа test.c

#include <stdio.h>  
#include "fibonacci.h"  
  
void test(int result, int n) {  
 int countedMember = fibonacci(n);  
 printf("Expected Result: %d\n", result);  
 printf("Received Result: %d\n\n", countedMember);  
}  
  
int main() {  
 int result1 = 610, n1 = 15;  
 test(result1, n1);  
  
 int result2 = 5, n2 = 5;  
 test(result2, n2);  
  
 int result3 = 1134903170, n3 = 45;  
 test(result3, n3);  
  
 return 0;  
}

4. Препроцессирование

Первым шагом сборки, написанных на Си программ, является препроцессирование файлов исходного текста fibonacci.c и test.c в файлы fibonacci.i и test.i.

Драйвер компилятора gcc riscv64-unknown-elf-gcc запускается с параметрами командной строки -march=rv32i и -mabi=ilp32. Эти параметры указывают, что целевым является процессор с базовой архитектурой системы команд RV32I. Параметр -O1 указывает, что надо выполнять простые оптимизации генерируемого кода, параметр -o сохраняет результат в fibonacci.i и test.i, параметр -E указывает, что надо остановить процесс сборки после препроцессирования.

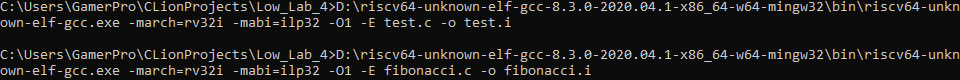


Рис. 1 Параметры запуска

Ниже представлен результат работы препроцессора, листинг файлов fibonacci.i и test.i.

Листинг 5. fibonacci.i

# 1 "fibonacci.c"  
# 1 "<built-in>"  
# 1 "<command-line>"  
# 1 "fibonacci.c"  
# 1 "fibonacci.h" 1  
  
  
  
int fibonacci(int n);  
# 2 "fibonacci.c" 2  
  
int fibonacci(int n) {  
 int first = 1;  
 int second = 0;  
 int third = 0;  
 while (n > 0) {  
 third = first + second;  
 first = second;  
 second = third;  
 n--;  
 }  
 return third;  
}

Листинг 6. test.i

# 1 "test.c"  
# 1 "<built-in>"  
# 1 "<command-line>"  
# 1 "test.c"  
# 1 "d:\\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86\_64-w64-mingw32\\riscv64-unknown-elf\\include\\stdio.h" 1 3

Далее идут линки библиотеки stdio.h, которая подключена к проекту.

# 2 "test.c" 2  
# 1 "fibonacci.h" 1  
  
  
  
  
# 4 "fibonacci.h"  
int fibonacci(int n);  
# 3 "test.c" 2  
  
void test(int result, int n) {  
 int countedMember = fibonacci(n);  
 printf("Expected Result: %d\n", result);  
 printf("Received Result: %d\n\n", countedMember);  
}  
  
int main() {  
 int result1 = 610, n1 = 15;  
 test(result1, n1);  
  
 int result2 = 5, n2 = 5;  
 test(result2, n2);  
  
 int result3 = 1134903170, n3 = 45;  
 test(result3, n3);  
  
 return 0;  
}

Начинающиеся с символа “#” директивы, используются для передачи информации об исходном тексте из препроцессора в компилятор. В файле test.i четвертая директива «# 1 “test.c”» информирует компилятор о том, что следующая строка является результатом обработки строки 1 исходного файла “test.c”. В этой строке стояла команда #include “fibonacci.h”, поэтому препроцессор произвел вставку содержимого этого заголовочного файла. Описание самого содержимого файла начинается после директивы # 2 “test.c” 2. Исходный код тестирующей функции main после работы препроцессора остался без изменений, как и исходный код функции fibonacci из файла fibonacci.i.

5. Компиляция файлов fibonacci.i и test.i

Параметр -S указывает, что нужно остановить процесс сборки после компиляции (без запуска ассемблера).

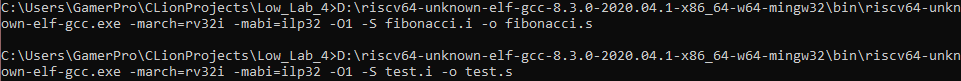


Рис. 2 Параметры запуска

Ниже представлен код на языке ассемблера, листинг fibonacci.s и test.s.

Листинг 7. fibonacci.s

.file "fibonacci.c"  
 .option nopic  
 .attribute arch, "rv32i2p0"  
 .attribute unaligned\_access, 0  
 .attribute stack\_align, 16  
 .text  
 .align 2  
 .globl fibonacci  
 .type fibonacci, @function  
fibonacci:  
 mv a5,a0  
 ble a0,zero,.L4  
 li a4,0  
 li a3,1  
.L3:  
 add a0,a3,a4  
 addi a5,a5,-1  
 mv a3,a4  
 mv a4,a0  
 bne a5,zero,.L3  
 ret  
.L4:  
 li a0,0  
 ret  
 .size fibonacci, .-fibonacci  
 .ident "GCC: (SiFive GCC 8.3.0-2020.04.1) 8.3.0"

Листинг 8. test.s

.file "test.c"  
 .option nopic  
 .attribute arch, "rv32i2p0"  
 .attribute unaligned\_access, 0  
 .attribute stack\_align, 16  
 .text  
 .align 2  
 .globl test  
 .type test, @function  
test:  
 addi sp,sp,-16  
 sw ra,12(sp)  
 sw s0,8(sp)  
 sw s1,4(sp)  
 mv s1,a0  
 mv a0,a1  
 call fibonacci  
 mv s0,a0  
 mv a1,s1  
 lui a0,%hi(.LC0)  
 addi a0,a0,%lo(.LC0)  
 call printf  
 mv a1,s0  
 lui a0,%hi(.LC1)  
 addi a0,a0,%lo(.LC1)  
 call printf  
 lw ra,12(sp)  
 lw s0,8(sp)  
 lw s1,4(sp)  
 addi sp,sp,16  
 jr ra  
 .size test, .-test  
 .align 2  
 .globl main  
 .type main, @function  
main:  
 addi sp,sp,-16  
 sw ra,12(sp)  
 li a1,15  
 li a0,610  
 call test  
 li a1,5  
 li a0,5  
 call test  
 li a1,45  
 li a0,1134903296  
 addi a0,a0,-126  
 call test  
 li a0,0  
 lw ra,12(sp)  
 addi sp,sp,16  
 jr ra  
 .size main, .-main  
 .section .rodata.str1.4,"aMS",@progbits,1  
 .align 2  
.LC0:  
 .string "Expected Result: %d\n"  
 .zero 3  
.LC1:  
 .string "Received Result: %d\n\n"  
 .ident "GCC: (SiFive GCC 8.3.0-2020.04.1) 8.3.0"

Метка main показывает главную программу, которая вызывает программу test, в свою очередь программа test вызывает программу fibonacci. Из-за того, что не проведено ассемблирование fibonacci нигде не определена.

**6. Ассемблирование файлов fibonacci.s и test.s**

Параметр -c указывает, что необходимо остановить процесс сборки после ассемблирования. Результат работы ассемблера – объектный код – сохраняется в файлах с расширением .o.

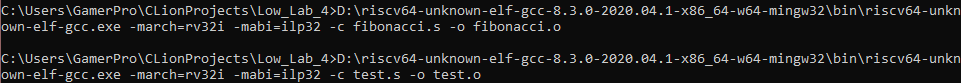


Рис. 3 Параметры запуска

Для просмотра содержимого файлов fibonacci.o и test.o используется objdump.

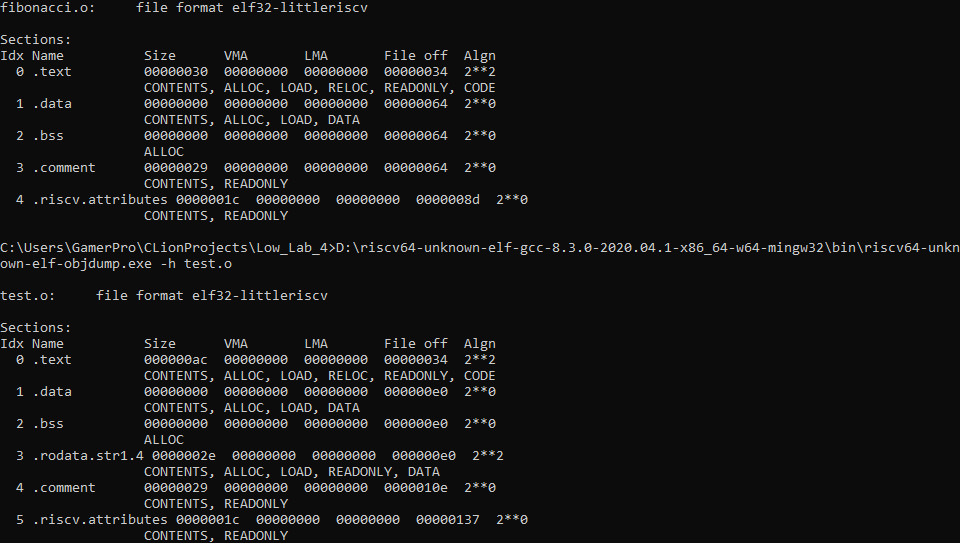


Рис. 4 Заголовки секций файлов fibonacci.o и test.o

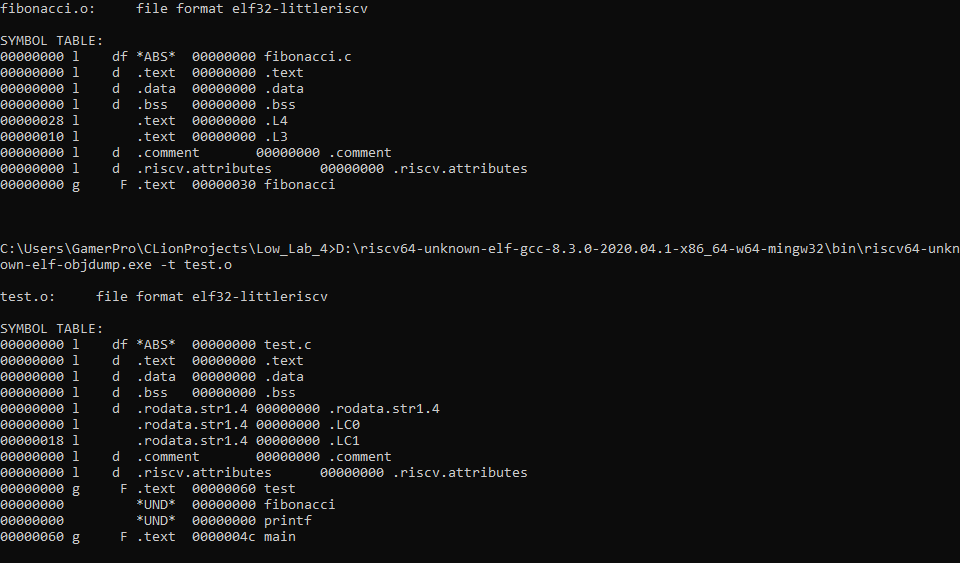


Рис. 5 Таблицы символов файлов fibonacci.o и test.o

Из рисунка 5 видно, что в таблице символов файла fibonacci.o содержится один глобальный (g) символ типа функция (F) – fibonacci.

В таблице символов файла test.o содержатся два глобальных (g) символа типа функции (F) – main и test. Два символа типа \*UND\* (undefined) – fibonacci и printf. Появление символов типа \*UND\* связано с тем, что эти символы использовались в ассемблерном коде, из которого получен объектный файл, но не были определены. Ассемблер сделал вывод о том, что символы должны быть определены в другом месте, поэтому отразил это в таблице кодов.

Ниже представлены секции .text объектных файлов – секций кода, в которых содержатся коды инструкций.

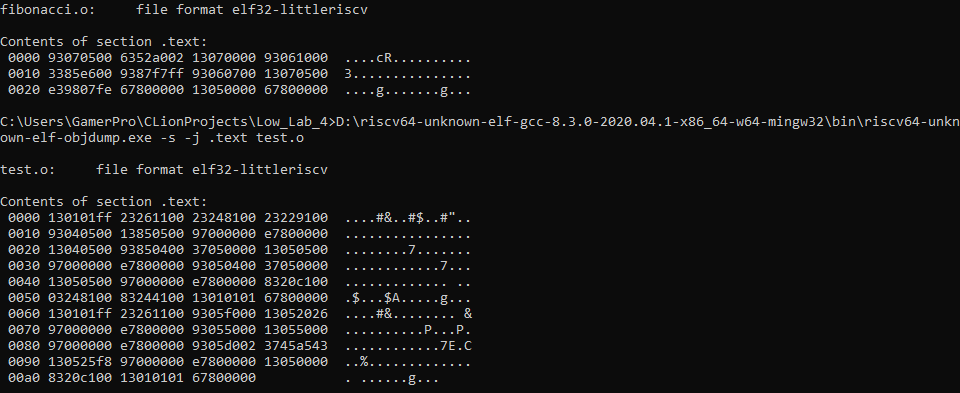


Рис. 6 Секции .text файлов fibonacci.o и test.o

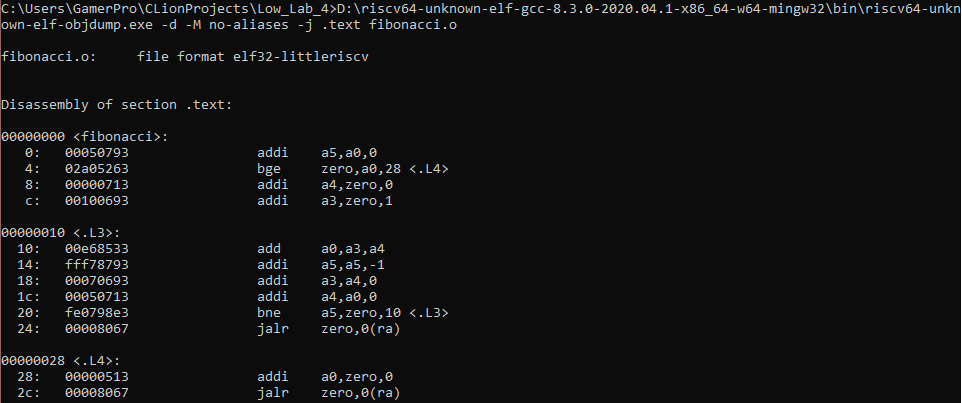


Рис. 7 Дизассемблированный код fibonacci.o

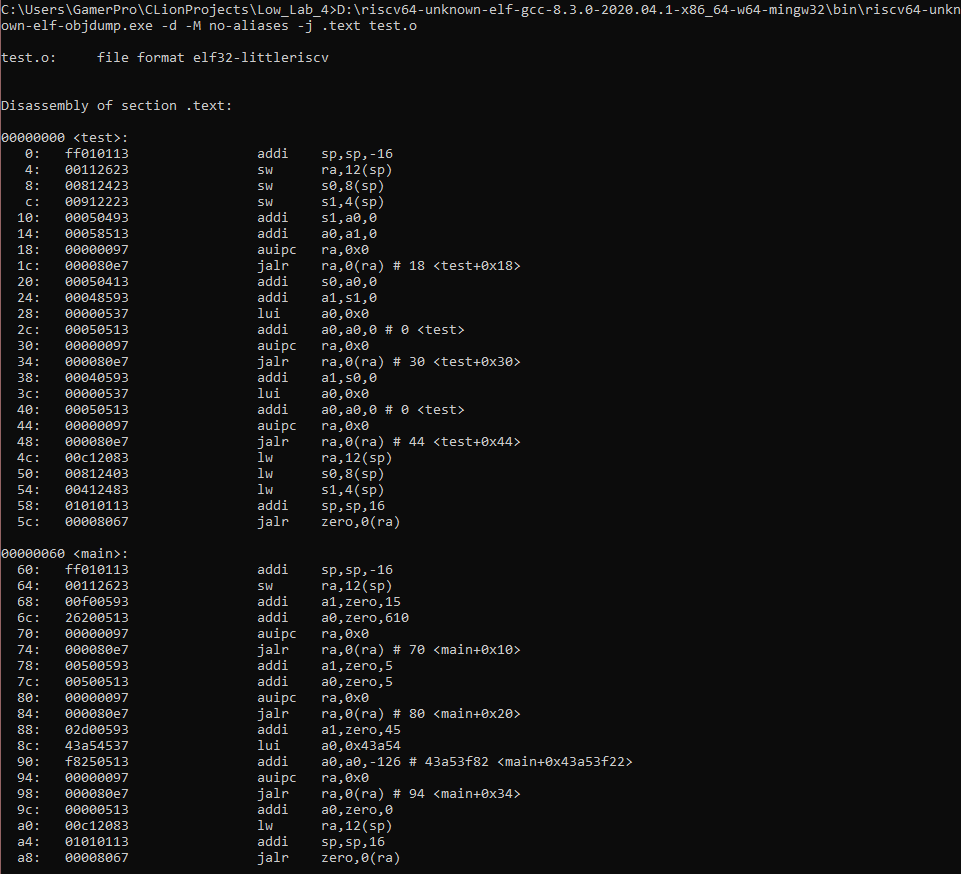


Рис. 8 Дизассемблированный код test.o

Из рисунков 7 и 8 видно, что дизассемблированный код практически идентичен сгенерированном, только он без псевдо-инструкций (call, mv).

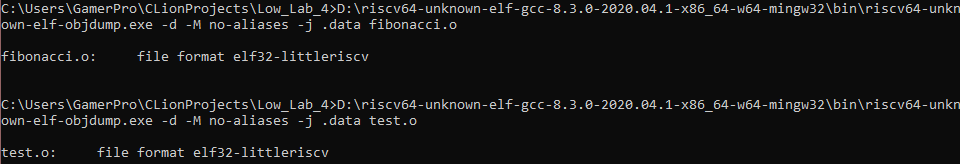


Рис. 9 Секция .data файлов fibonacci.o и test.o

Из рисунка 9 видно, что секция .data (секция инициализированных данных) файлов fibonacci.o и test.o не содержит данных.

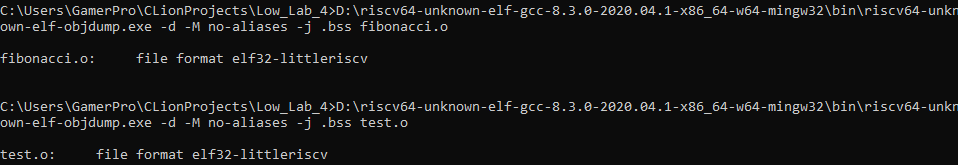


Рис. 10 Секция .bss файлов fibonacci.o и test.o

Из рисунка 10 видно, что секция .bss (секция данных, инициализированных нулями) файлов fibonacci.o и test.o тоже пустые.

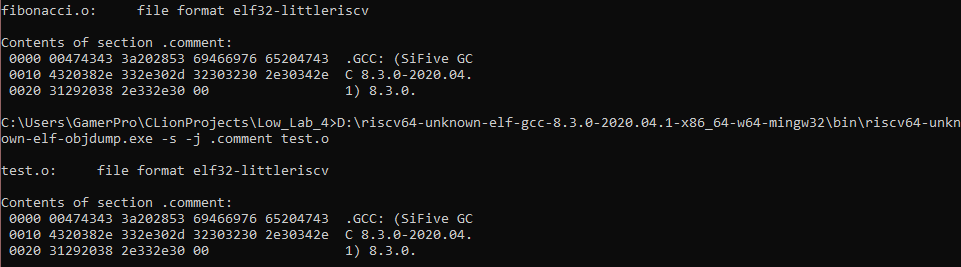


Рис. 11 Секция .comment файлов fibonacci.o и test.o

Из рисунка 11 видно, что секция .comment (секция данных о версиях) одинакова для файлов fibonacci.o и test.o.

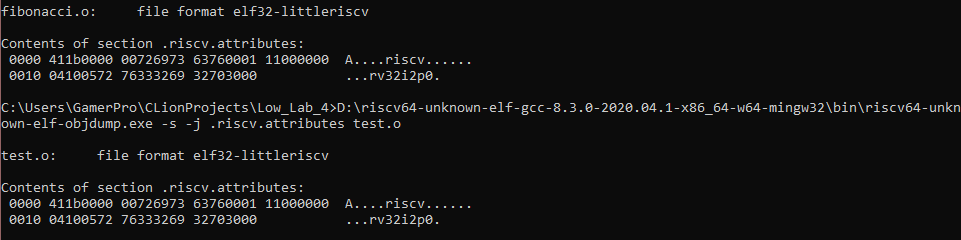


Рис. 12 Секция .riscv.attributes файлов fibonacci.o и test.o

Из рисунка 12 видно, что секция .riscv.attributes содержит информацию об используемой архитектуре команд RV32I.

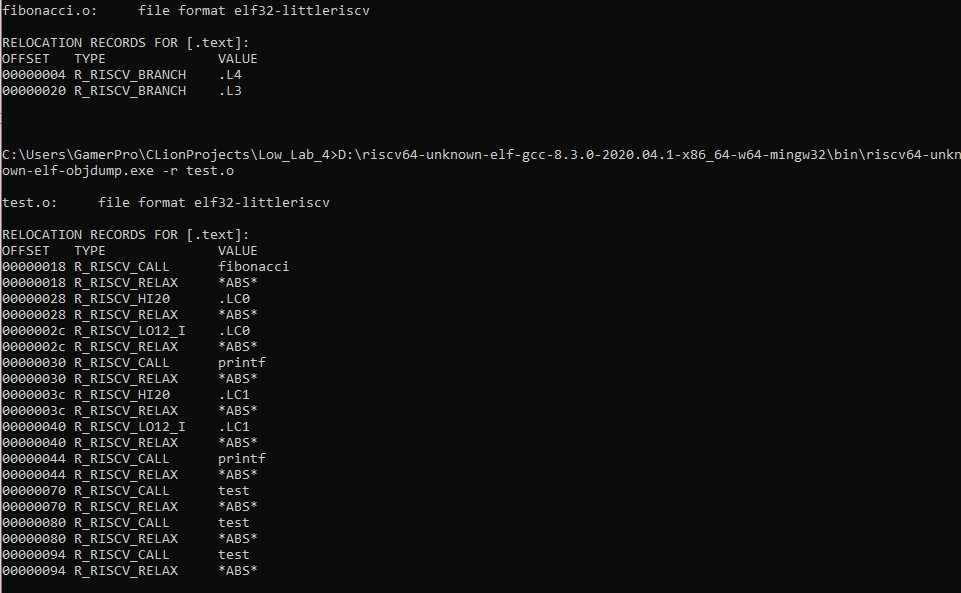


Рис. 13 Таблицы перемещения файлов fibonacci.o и test.o

Из рисунка 13 видно, что в таблице перемещения fibonacci.o находиться информация о ветвлениях R\_RISCV\_BRANCH. В таблице перемещения test.o видно вызов R\_RISCV\_CALL. Записи типа R\_RISCV\_RELAX заносятся в таблицу перемещений в дополнение к записям типа R\_RISCV\_CALL.

**7. Компоновка**

С помощью *\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86\_64-w64-mingw32\bin\riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 fibonacci.o test.o -o fibonacci.out* компоновщиком был сформирован файл fibonacci.out.

Для просмотра содержимого файлов fibonacci.out используется objdump.

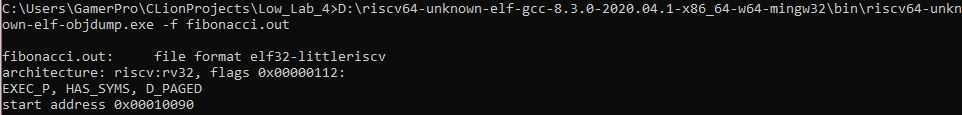


Рис. 14 Файл fibonacci.out

Из рисунка 14 видно, что флаг EXEC\_P указывает, что файл действительно является исполняемым, после загрузки его выполнение должно начаться с адреса 0x00010090 (entrypoint).

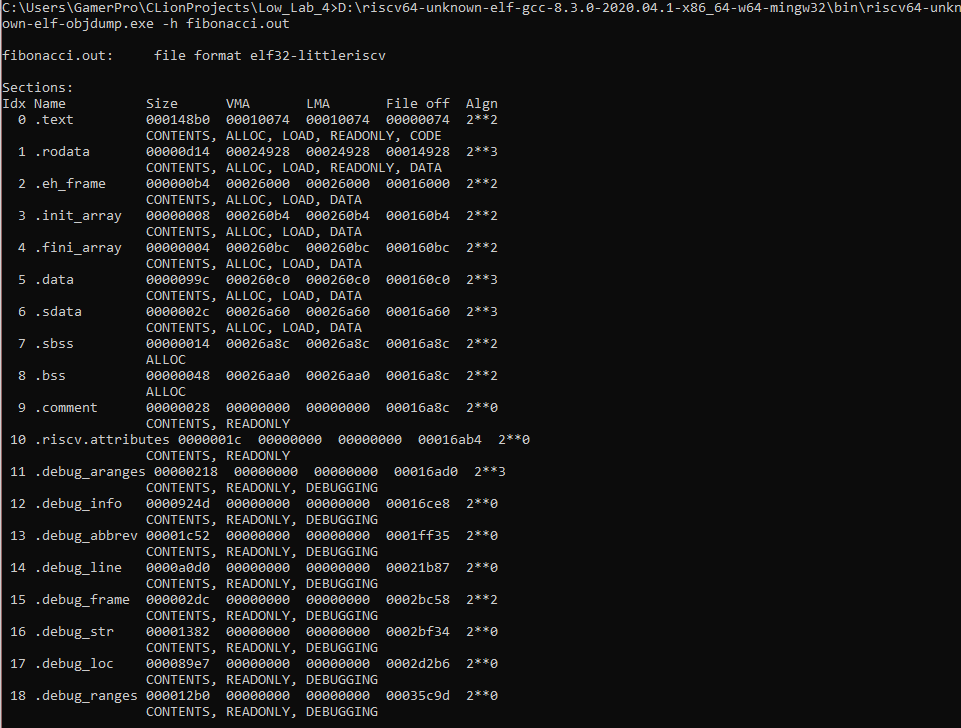


Рис. 15 Секции файла fibonacci.out

Из рисунка 15 видно, что в файле fibonacci.out е производится слияние содержания секций обоих объектных файлов.

Для анализа секции .text был создан файл log, ниже представлен листинг некоторых его частей.

Листинг 8. Log

*fibonacci.out*: file format elf32-littleriscv  
  
  
Disassembly of section .text:  
  
*00010074* <register\_fini>:  
 *10074*: ffff0797 auipc a5,0xffff0  
 *10078*: f8c78793 addi a5,a5,-*116* # *0* <register\_fini-0x10074>  
 1007c: *00078863* beq a5,zero,1008c <register\_fini+0x18>  
 *10080*: *00000517* auipc a0,0x0  
 *10084*: 1d050513 addi a0,a0,*464* # *10250* <\_\_libc\_fini\_array>  
 *10088*: 1800006f jal zero,*10208* <atexit>  
 1008c: *00008067* jalr zero,*0*(ra)  
  
*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: 29c000ef jal ra,*10344* <memset>  
 100ac: *00000517* auipc a0,0x0  
 100b0: 1a450513 addi a0,a0,*420* # *10250* <\_\_libc\_fini\_array>  
 100b4: 154000ef jal ra,*10208* <atexit>  
 100b8: 1f8000ef jal ra,102b0 <\_\_libc\_init\_array>  
 100bc: *00012503* lw a0,*0*(sp)  
 100c0: *00410593* addi a1,sp,*4* 100c4: *00000613* addi a2,zero,*0* 100c8: 100000ef jal ra,101c8 <main>  
 100cc: 1500006f jal zero,1021c <exit>

. . .

*00010144* <fibonacci>:  
 *10144*: *00050793* addi a5,a0,*0  
 10148*: 02a05263 bge zero,a0,1016c <fibonacci+0x28>  
 1014c: *00000713* addi a4,zero,*0  
 10150*: *00100693* addi a3,zero,*1  
 10154*: 00e68533 add a0,a3,a4  
 *10158*: fff78793 addi a5,a5,-*1* 1015c: *00070693* addi a3,a4,*0  
 10160*: *00050713* addi a4,a0,*0  
 10164*: fe0798e3 bne a5,zero,*10154* <fibonacci+0x10>  
 *10168*: *00008067* jalr zero,*0*(ra)  
 1016c: *00000513* addi a0,zero,*0  
 10170*: *00008067* jalr zero,*0*(ra)

*00010174* <test>:  
 *10174*: ff010113 addi sp,sp,-*16  
 10178*: *00112623* sw ra,*12*(sp)  
 1017c: *00812423* sw s0,*8*(sp)  
 *10180*: *00912223* sw s1,*4*(sp)  
 *10184*: *00050493* addi s1,a0,*0  
 10188*: *00058513* addi a0,a1,*0* 1018c: fb9ff0ef jal ra,*10144* <fibonacci>  
 *10190*: *00050413* addi s0,a0,*0  
 10194*: *00048593* addi a1,s1,*0  
 10198*: *00025537* lui a0,0x25  
 1019c: *92850513* addi a0,a0,-*1752* # *24928* <\_\_clzsi2+0x50>  
 101a0: 2c4000ef jal ra,*10464* <printf>  
 101a4: *00040593* addi a1,s0,*0* 101a8: *00025537* lui a0,0x25  
 101ac: *94050513* addi a0,a0,-*1728* # *24940* <\_\_clzsi2+0x68>  
 101b0: 2b4000ef jal ra,*10464* <printf>  
 101b4: 00c12083 lw ra,*12*(sp)  
 101b8: *00812403* lw s0,*8*(sp)  
 101bc: *00412483* lw s1,*4*(sp)  
 101c0: *01010113* addi sp,sp,*16* 101c4: *00008067* jalr zero,*0*(ra)  
  
000101c8 <main>:  
 101c8: ff010113 addi sp,sp,-*16* 101cc: *00112623* sw ra,*12*(sp)  
 101d0: 00f00593 addi a1,zero,*15* 101d4: *26200513* addi a0,zero,*610* 101d8: f9dff0ef jal ra,*10174* <test>  
 101dc: *00500593* addi a1,zero,*5* 101e0: *00500513* addi a0,zero,*5* 101e4: f91ff0ef jal ra,*10174* <test>  
 101e8: 02d00593 addi a1,zero,*45* 101ec: 43a54537 lui a0,0x43a54  
 101f0: f8250513 addi a0,a0,-*126* # 43a53f82 <\_\_BSS\_END\_\_+0x43a2d49a>  
 101f4: f81ff0ef jal ra,*10174* <test>  
 101f8: *00000513* addi a0,zero,*0* 101fc: 00c12083 lw ra,*12*(sp)  
 *10200*: *01010113* addi sp,sp,*16  
 10204*: *00008067* jalr zero,*0*(ra)

0001021c <exit>:  
 1021c: ff010113 addi sp,sp,-*16  
 10220*: *00000593* addi a1,zero,*0  
 10224*: *00812423* sw s0,*8*(sp)  
 *10228*: *00112623* sw ra,*12*(sp)  
 1022c: *00050413* addi s0,a0,*0  
 10230*: 050030ef jal ra,*13280* <\_\_call\_exitprocs>  
 *10234*: 1b818793 addi a5,gp,*440* # 26a78 <\_global\_impure\_ptr>  
 *10238*: 0007a503 lw a0,*0*(a5)  
 1023c: 03c52783 lw a5,*60*(a0)  
 *10240*: *00078463* beq a5,zero,*10248* <exit+0x2c>  
 *10244*: 000780e7 jalr ra,*0*(a5)  
 *10248*: *00040513* addi a0,s0,*0* 1024c: 5080f0ef jal ra,1f754 <\_exit>

. . .

0001f754 <\_exit>:  
 1f754: *00000593* addi a1,zero,*0* 1f758: *00000613* addi a2,zero,*0* 1f75c: *00000693* addi a3,zero,*0* 1f760: *00000713* addi a4,zero,*0* 1f764: *00000793* addi a5,zero,*0* 1f768: 05d00893 addi a7,zero,*93* 1f76c: *00000073* ecall  
 1f770: *00054463* blt a0,zero,1f778 <\_exit+0x24>  
 1f774: 0000006f jal zero,1f774 <\_exit+0x20>  
 1f778: ff010113 addi sp,sp,-*16* 1f77c: *00812423* sw s0,*8*(sp)  
 1f780: *00050413* addi s0,a0,*0* 1f784: *00112623* sw ra,*12*(sp)  
 1f788: *40800433* sub s0,zero,s0  
 1f78c: 2f0000ef jal ra,1fa7c <\_\_errno>  
 1f790: *00852023* sw s0,*0*(a0)  
 1f794: 0000006f jal zero,1f794 <\_exit+0x40>

Компоновщик все переходы сочетания инструкций auipc и jalr заменил на инструкцию jal и корректным адресом перехода.

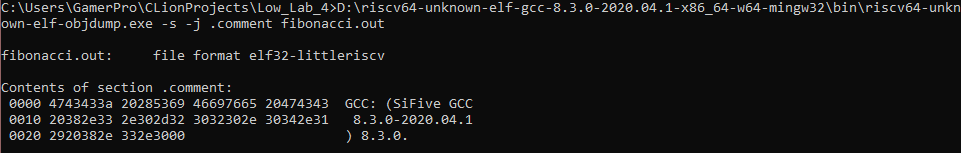


Рис. 16 Секция .comment файла fibonacci.out

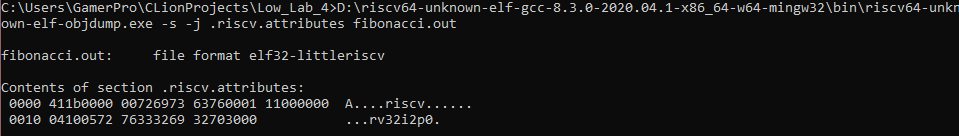


Рис. 17 Секция .riscv.attributes файла fibonacci.out

Секция .riscv.attributes содержит информацию о том, что используется архитектура команд RV32I.

Листинг 9. Таблица символов файла fibonacci.out

C:\Users\GamerPro\CLionProjects\Low\_Lab\_4>D:\riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86\_64-w64-mingw32\bin\riscv64-unknown-elf-objdump.exe -t fibonacci.out

fibonacci.out: file format elf32-littleriscv

SYMBOL TABLE:

00010074 l d .text 00000000 .text

00024928 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 0000001c register\_fini

00000000 l df \*ABS\* 00000000 crtstuff.c

00026000 l O .eh\_frame 00000000 \_\_EH\_FRAME\_BEGIN\_\_

000100d0 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

00010120 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 fibonacci.c

00000000 l df \*ABS\* 00000000 test.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 vfprintf.c

00012fbc l F .text 000000c0 \_\_sbprintf

00024ac4 l O .rodata 00000010 blanks.4504

00024ad4 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 fflush.c

00000000 l df \*ABS\* 00000000 findfp.c

00013680 l F .text 00000008 \_\_fp\_lock

00013694 l F .text 00000184 \_\_sinit.part.0

00013818 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

00013f88 l F .text 00000068 eshdn1

00013ff0 l F .text 00000070 eshup1

00014060 l F .text 000000e0 m16m

00014140 l F .text 00000024 eisnan.part.0

00014164 l F .text 0000004c eneg

000141b0 l F .text 00000040 eisneg

000141f0 l F .text 000000e8 emovi

000142d8 l F .text 0000011c ecmp

000143f4 l F .text 0000001c eisinf.part.1

00014410 l F .text 000001ac eshift.part.3

000145bc l F .text 0000017c enormlz

00014738 l F .text 00000408 emdnorm

00014b40 l F .text 00000164 eiremain

00014ca4 l F .text 000000b0 emovo.isra.6

00014d54 l F .text 00000408 emul

0001515c l F .text 00000504 ediv

00015660 l F .text 00000144 e113toe.isra.8

00024b28 l O .rodata 00000014 ezero

00024b3c l O .rodata 00000014 eone

00024b50 l O .rodata 00000104 etens

00024c54 l O .rodata 00000104 emtens

00024d58 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

00024db8 l O .rodata 0000000c p05.3298

00000000 l df \*ABS\* 00000000 sbrkr.c

00000000 l df \*ABS\* 00000000 s\_frexp.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

0002504c l O .rodata 00000010 blanks.4489

0002505c l O .rodata 00000010 zeroes.4490

00000000 l df \*ABS\* 00000000 vfprintf.c

0001b8d4 l F .text 000000f0 \_\_sprint\_r.part.0

0001ccc4 l F .text 000000c0 \_\_sbprintf

000251d8 l O .rodata 00000010 blanks.4480

000251e8 l O .rodata 00000010 zeroes.4481

00000000 l df \*ABS\* 00000000 wctomb\_r.c

00000000 l df \*ABS\* 00000000 writer.c

00000000 l df \*ABS\* 00000000 mallocr.c

00000000 l df \*ABS\* 00000000 closer.c

00000000 l df \*ABS\* 00000000 ctype\_.c

00000000 l df \*ABS\* 00000000 fclose.c

00000000 l df \*ABS\* 00000000 fputwc.c

00000000 l df \*ABS\* 00000000 fstatr.c

00000000 l df \*ABS\* 00000000 fvwrite.c

00000000 l df \*ABS\* 00000000 isattyr.c

00000000 l df \*ABS\* 00000000 lseekr.c

00000000 l df \*ABS\* 00000000 memcpy.c

00000000 l df \*ABS\* 00000000 memmove-stub.c

00000000 l df \*ABS\* 00000000 readr.c

00000000 l df \*ABS\* 00000000 mallocr.c

00000000 l df \*ABS\* 00000000 reent.c

00000000 l df \*ABS\* 00000000 vfprintf.c

00025468 l O .rodata 00000010 blanks.4466

00025478 l O .rodata 00000010 zeroes.4467

00000000 l df \*ABS\* 00000000 wbuf.c

00000000 l df \*ABS\* 00000000 wcrtomb.c

00000000 l df \*ABS\* 00000000 sys\_close.c

00000000 l df \*ABS\* 00000000 sys\_exit.c

00000000 l df \*ABS\* 00000000 sys\_fstat.c

00000000 l df \*ABS\* 00000000 sys\_isatty.c

00000000 l df \*ABS\* 00000000 sys\_lseek.c

00000000 l df \*ABS\* 00000000 sys\_read.c

00000000 l df \*ABS\* 00000000 sys\_sbrk.c

00026a9c l O .sbss 00000004 heap\_end.1862

00000000 l df \*ABS\* 00000000 sys\_write.c

00000000 l df \*ABS\* 00000000 sys\_conv\_stat.c

00000000 l df \*ABS\* 00000000 errno.c

00000000 l df \*ABS\* 00000000 libgcc2.c

00000000 l df \*ABS\* 00000000 libgcc2.c

00000000 l df \*ABS\* 00000000 divdf3.c

00000000 l df \*ABS\* 00000000 muldf3.c

00000000 l df \*ABS\* 00000000 eqtf2.c

00000000 l df \*ABS\* 00000000 getf2.c

00000000 l df \*ABS\* 00000000 letf2.c

00000000 l df \*ABS\* 00000000 multf3.c

00000000 l df \*ABS\* 00000000 subtf3.c

00000000 l df \*ABS\* 00000000 fixtfsi.c

00000000 l df \*ABS\* 00000000 floatsitf.c

00000000 l df \*ABS\* 00000000 extenddftf2.c

00000000 l df \*ABS\* 00000000 trunctfdf2.c

00000000 l df \*ABS\* 00000000 libgcc2.c

00000000 l df \*ABS\* 00000000 libgcc2.c

00000000 l df \*ABS\* 00000000 crtstuff.c

000260b0 l O .eh\_frame 00000000 \_\_FRAME\_END\_\_

00000000 l 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

000185f4 g F .text 0000009c \_mprec\_log10

000186f8 g F .text 00000078 \_\_any\_on

0001d750 g F .text 00000054 \_isatty\_r

00024e90 g O .rodata 00000028 \_\_mprec\_tinytens

00018c34 g F .text 000000b0 strcpy

0001e070 g F .text 00000040 cleanup\_glue

0001d7a4 g F .text 00000060 \_lseek\_r

00021458 g F .text 00000144 .hidden \_\_getf2

0002138c g F .text 000000cc .hidden \_\_eqtf2

00010464 g F .text 00000054 printf

000268c0 g \*ABS\* 00000000 \_\_global\_pointer$

0001f5f4 g F .text 00000078 \_wcrtomb\_r

00018a48 g F .text 00000068 \_\_sseek

000139b0 g F .text 00000010 \_\_sinit

0001f45c g F .text 00000184 \_\_swbuf\_r

00016b14 g F .text 0000007c \_setlocale\_r

00013820 g F .text 00000078 \_\_sfmoreglue

000176bc g F .text 00000004 \_\_malloc\_unlock

0002413c g F .text 00000188 .hidden \_\_floatsitf

0001d920 g F .text 00000120 memmove

0001399c g F .text 00000014 \_cleanup

000176c0 g F .text 000000a8 \_Balloc

0001fa7c g F .text 0000000c \_\_errno

0001f9d8 g F .text 000000a4 \_conv\_stat

00016afc g F .text 00000008 \_\_localeconv\_l

0001d238 g F .text 0000005c \_fstat\_r

00026ae4 g O .bss 00000004 errno

000189b8 g F .text 00000008 \_\_seofread

00026a60 g .sdata 00000000 \_\_SDATA\_BEGIN\_\_

0001d804 g F .text 0000011c memcpy

00013688 g F .text 0000000c \_cleanup\_r

00018e3c g F .text 00002a98 \_svfprintf\_r

00018550 g F .text 000000a4 \_\_ratio

00010420 g F .text 00000044 \_printf\_r

0002159c g F .text 00000144 .hidden \_\_letf2

0002482c g F .text 00000048 .hidden \_\_udivsi3

0001f798 g F .text 00000070 \_fstat

00024dc8 g O .rodata 000000c8 \_\_mprec\_tens

00026a98 g O .sbss 00000004 \_\_malloc\_top\_pad

00026a7c g O .sdata 00000000 .hidden \_\_dso\_handle

00016b04 g F .text 00000008 \_localeconv\_r

0001d294 g F .text 000004bc \_\_sfvwrite\_r

00017b20 g F .text 00000034 \_\_i2b

00018770 g F .text 00000054 \_sbrk\_r

0001da40 g F .text 00000060 \_read\_r

0001cf3c g F .text 00000110 \_fclose\_r

00013658 g F .text 00000028 fflush

00026a94 g O .sbss 00000004 \_\_malloc\_max\_sbrked\_mem

000182f4 g F .text 00000118 \_\_b2d

0002005c g F .text 000004f0 .hidden \_\_umoddi3

0001f808 g F .text 00000040 \_isatty

00026a78 g O .sdata 00000004 \_global\_impure\_ptr

0001daa0 g F .text 000005d0 \_realloc\_r

000102b0 g F .text 00000094 \_\_libc\_init\_array

0001fa88 g F .text 000005d4 .hidden \_\_udivdi3

0001d188 g F .text 0000002c \_fputwc\_r

00024eb8 g O .rodata 00000028 \_\_mprec\_bigtens

000178e4 g F .text 00000110 \_\_s2b

0001f8e8 g F .text 000000a0 \_sbrk

00018054 g F .text 0000005c \_\_mcmp

000139d0 g F .text 00000014 \_\_fp\_lock\_all

00010250 g F .text 00000060 \_\_libc\_fini\_array

0001e0b0 g F .text 00000110 \_reclaim\_reent

000179f4 g F .text 00000074 \_\_hi0bits

00023fec g F .text 00000150 .hidden \_\_fixtfsi

00017dc0 g F .text 00000148 \_\_pow5mult

00024874 g F .text 00000010 .hidden \_\_umodsi3

000248d8 g F .text 0000004c .hidden \_\_clzsi2

000139c0 g F .text 00000004 \_\_sfp\_lock\_acquire

000175dc g F .text 000000dc memchr

00018874 g F .text 0000006c \_sprintf\_r

00013b20 g F .text 000002f8 \_free\_r

00016b90 g F .text 00000008 \_\_locale\_mb\_cur\_max

00013280 g F .text 0000011c \_\_call\_exitprocs

**00010174 g F .text 00000054 test**

00026a84 g O .sdata 00000004 \_\_malloc\_sbrk\_base

00010090 g F .text 00000040 \_start

0001f848 g F .text 00000050 \_lseek

00017f08 g F .text 0000014c \_\_lshift

0002159c g F .text 00000144 .hidden \_\_lttf2

0002138c g F .text 000000cc .hidden \_\_netf2

0001e1c0 g F .text 000001bc \_\_ssprint\_r

000131e4 g F .text 0000009c \_\_register\_exitproc

00016a98 g F .text 00000064 \_ldcheck

00017b54 g F .text 0000026c \_\_multiply

00024800 g F .text 00000024 .hidden \_\_mulsi3

00018d70 g F .text 000000cc strncpy

00026abc g O .bss 00000028 \_\_malloc\_current\_mallinfo

0001840c g F .text 00000144 \_\_d2b

000242c4 g F .text 00000208 .hidden \_\_extenddftf2

0001cee8 g F .text 00000054 \_close\_r

0001307c g F .text 00000168 \_\_swsetup\_r

0002054c g F .text 000007d0 .hidden \_\_divdf3

00013898 g F .text 00000104 \_\_sfp

00018690 g F .text 00000068 \_\_copybits

00026ae8 g .bss 00000000 \_\_BSS\_END\_\_

00026654 g O .data 00000408 \_\_malloc\_av\_

000139cc g F .text 00000004 \_\_sinit\_lock\_release

00020d1c g F .text 00000670 .hidden \_\_muldf3

0001895c g F .text 0000005c \_\_sread

**00010144 g F .text 00000030 fibonacci**

000176b8 g F .text 00000004 \_\_malloc\_lock

000135f8 g F .text 00000060 \_fflush\_r

0001ce20 g F .text 000000c8 \_calloc\_r

00026a8c g .sbss 00000000 \_\_bss\_start

00010344 g F .text 000000dc memset

**000101c8 g F .text 00000040 main**

00026a90 g O .sbss 00000004 \_\_malloc\_max\_total\_mem

0001f5e0 g F .text 00000014 \_\_swbuf

00018ab0 g F .text 00000008 \_\_sclose

0001d04c g F .text 00000010 fclose

00016d9c g F .text 000007cc \_malloc\_r

0001cd90 g F .text 00000030 \_\_ascii\_wctomb

00013e18 g F .text 000000b0 \_fwalk

00017568 g F .text 0000000c \_mbtowc\_r

00024824 g F .text 00000084 .hidden \_\_divsi3

000139f8 g F .text 00000128 \_malloc\_trim\_r

00018ab8 g F .text 0000017c strcmp

0001ccac g F .text 00000018 vfiprintf

000216e0 g F .text 0000136c .hidden \_\_multf3

000188e0 g F .text 0000007c sprintf

0002553c g O .rodata 00000100 .hidden \_\_clz\_tab

00026a8c g O .sbss 00000004 \_PathLocale

00010208 g F .text 00000014 atexit

0001cdc0 g F .text 00000060 \_write\_r

00016b98 g F .text 00000014 setlocale

00026a80 g O .sdata 00000004 \_impure\_ptr

0001339c g F .text 0000025c \_\_sflush\_r

00021458 g F .text 00000144 .hidden \_\_gttf2

0001e37c g F .text 000010e0 \_svfiprintf\_r

00017574 g F .text 00000068 \_\_ascii\_mbtowc

00022a4c g F .text 000015a0 .hidden \_\_subtf3

00018294 g F .text 00000060 \_\_ulp

000139e4 g F .text 00000014 \_\_fp\_unlock\_all

00016b0c g F .text 00000008 localeconv

00016bac g F .text 000000d0 \_\_swhatbuf\_r

000260c0 g .data 00000000 \_\_DATA\_BEGIN\_\_

0001f988 g F .text 00000050 \_write

00026a8c g .sdata 00000000 \_edata

00026ae8 g .bss 00000000 \_end

0001d05c g F .text 0000012c \_\_fputwc

000189c0 g F .text 00000088 \_\_swrite

00026a88 g O .sdata 00000004 \_\_malloc\_trim\_threshold

0001021c g F .text 00000034 exit

0001b9dc g F .text 000012d0 \_vfiprintf\_r

00013ec8 g F .text 000000c0 \_fwalk\_reent

000180b0 g F .text 000001e4 \_\_mdiff

000248a8 g F .text 00000030 .hidden \_\_modsi3

000139c4 g F .text 00000004 \_\_sfp\_lock\_release

000157a4 g F .text 000012f4 \_ldtoa\_r

000251f8 g O .rodata 00000101 \_ctype\_

0001f898 g F .text 00000050 \_read

0001f754 g F .text 00000044 \_exit

00016c7c g F .text 00000120 \_\_smakebuf\_r

00018ce4 g F .text 0000008c strlen

0001b9c4 g F .text 00000018 \_\_sprint\_r

0001cd84 g F .text 0000000c \_wctomb\_r

000104b8 g F .text 00002aec \_vfprintf\_r

00017a68 g F .text 000000b8 \_\_lo0bits

0001f66c g F .text 00000090 wcrtomb

000187c4 g F .text 000000b0 frexp

000264e8 g O .data 0000016c \_\_global\_locale

00012fa4 g F .text 00000018 vfprintf

000244cc g F .text 00000334 .hidden \_\_trunctfdf2

0001d1b4 g F .text 00000084 fputwc

0001f6fc g F .text 00000058 \_close

000139c8 g F .text 00000004 \_\_sinit\_lock\_acquire

0001778c g F .text 00000158 \_\_multadd

00017768 g F .text 00000024 \_Bfree

Из листинга 9 видно, что таблица символов содержит дополнительные вхождения, но при этом определяет все нужные секции, сетки и адреса. Функции fibonacci, test, main помечены флагом функций F. При этом все они являются определенными и содержатся по корректным адресам для успешного вызова этих функций из других участков программ.

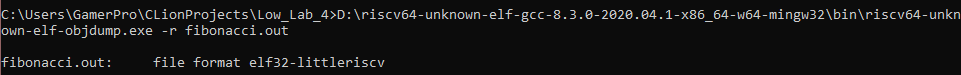


Рис. 18 Таблица перемещений файла fibonacci.out

Из рисунка 18 видно, что таблица перемещений не имеет данных, это означает, что все необходимые релокации, оптимизации и замены инструкций были успешно проведены компоновщиком.

Итогом сборки “по шагам” программы на ЯП C является файл, который исполняется на процессорах архитектуры RISC-V. Полученный файл рассчитывает заданный член последовательности Фибоначчи.

7. Создание статической библиотеки и make-файлов

Статическая библиотека (staticlibrary) – это архив объектных файлов, среди которых компоновщик выбирает «полезные» для данной программы. Объектный файл считается «полезным», если в нем определяется еще не разрешенный компоновщиком символ. Разработанная функция поиска члена последовательности Фибоначчи содержится в единственном исходном файле на ЯП C. Выделим этот файл в статическую библиотеку.

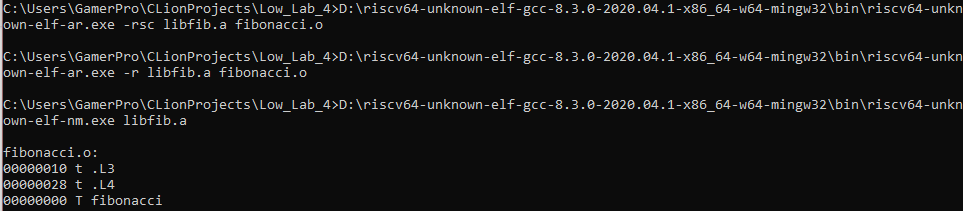


Рис. 19 Создание библиотеки libfib.a

В выводе утилиты nm кодом T обозначаются символы, определенные в соответствующем объектном файле. Символ функции fibonacci является основным символом, определяемым в этом объектном файле, остальные символы определяют лишь локальные для этого файла метки.



Рис. 20 Поиск библиотеки и сборка программы

После поиска библиотеки и сборки программы получается файл fibonacci.

**Сборка через makefile**

Makefile – это набор инструкций для программы make, которая позволяет собирать проекты, состоящие из большого числа файлов с расширениями .c и .h. Обычно эта программа используется в связке с системами сборки, например cmake, позволяя вести проекты модульно (т.е. проект с включенными подпроектами).

Листинг 10. Makefile

fibonacci: test.o libfib.a  
 gcc test.o libfib.a -o fibonacci  
  
test.o: test.c  
 gcc -c test.c  
  
libfib.a: fibonacci.o fibonacci.h  
 ar -rsc libfib.a fibonacci.o  
  
fibonacci.o:  
 gcc -c fibonacci.c  
  
clean:  
 del \*.o \*.a \*.exe

Что происходит в makefile:

* Создаём объектный файл test.o из исходного test.c
* Создаём объектный файл fibonacci.o из исходного fibonacci.c
* Архивируем объектный файл fibonacci.o (создаём статическую библиотеку libfib.a)
* Компонуем статическую библиотеку libfib.a с объектным файлом test.o и получаем исполняемый файл fibonacci.exe

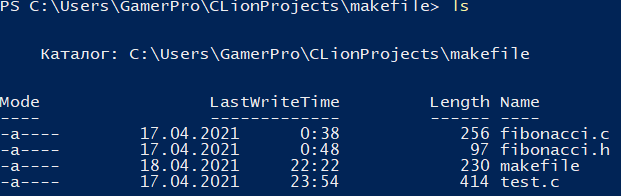


Рис. 21 Директория с makefile

Из рисунка 21 видно, что в директории 4 файла.

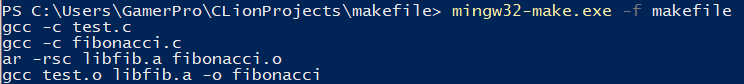


Рис. 22 Запуск makefile и процесс сборки

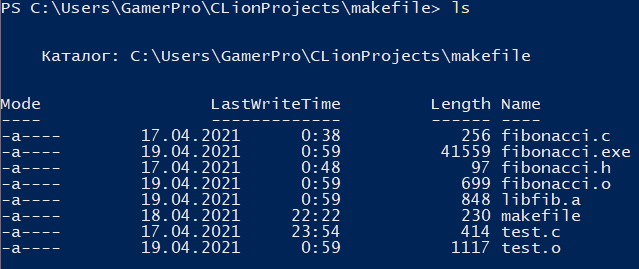


Рис. 23 Состояние директории после запуска makefile

Из рисунка 23 видно, что в директории стало больше файлов.

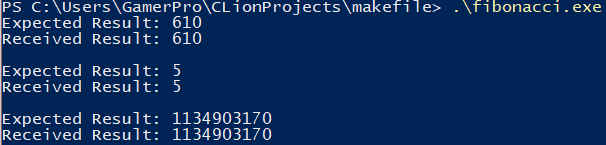


Рис. 24 Запуск fibonacci.exe

Из листинга 12 видно, что в makefile добавлена команда clean. Команда clean чистит директорию с файлами от построенных файлов, оставляя только исходные.

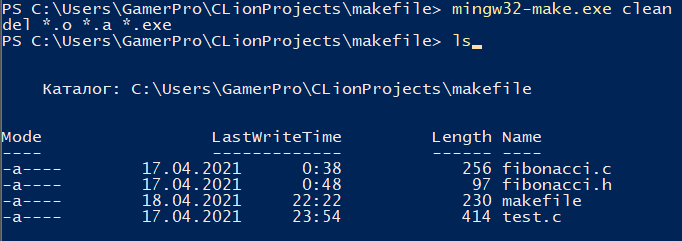


Рис. 25 Очистка и состояние директории

**8. Выводы**

В ходе выполнения лабораторной работы были получены навыки работы с препроцессором, компилятором, компоновщиком пакета GCC и драйвером компилятора riscv64-unknown-elf-gcc.

Также были изучены особенности каждого этапа пошаговой сборки набора программ. Разработанные функции были выделены в статическую библиотеку с помощью пакета средств разработки “SiFive GNU Embedded Toolchain”, а также с помощью make-файлов.