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 \_\_\_\_\_\_\_\_\_\_\_\_\_ Е.Е. Ремнева

(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 shift.i и test.i ……………………………………7**

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

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

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

**1.** **Техническое задание**

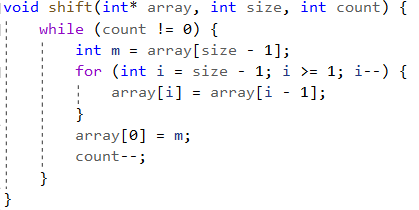
1. Разработать на языке Си программу, которая будет сдвигать массив вправо на заданное число разрядов.
2. Собрать программу «по шагам». Проанализировать выход препроцессора и компилятора. Проанализировать состав и содержимое секций, таблицы символов, таблицы перемещений и отладочную информацию, содержащуюся 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**

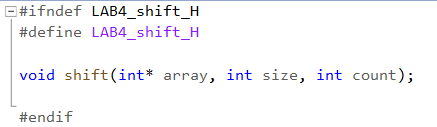
Чтобы сдвинуть массив элементов вправо, будет запоминаться последний элемент массива и с помощью цикла, который будет идти с конца значение разряда array[i] будет сохраняться в разряде массива array[i+1]. После этого сохраненный последний элемент массива возвращается на место нулевого элемента массива.

**3. Программы на ЯП C**

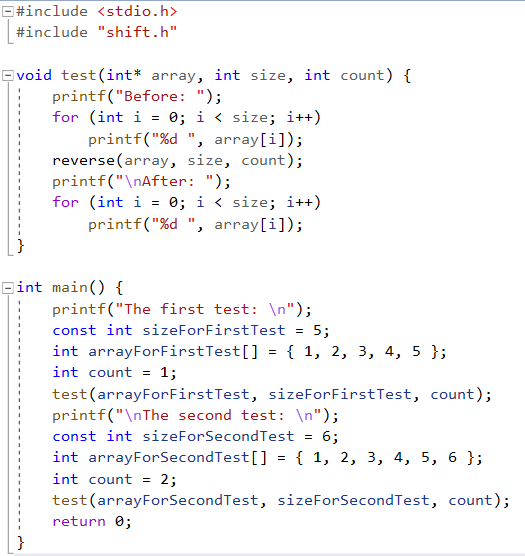
Листинг 2. Программа shift.c



Листинг 3. Заголовочный файл shift.h



Листинг 4. Программа test.c



4. Препроцессирование

Первым шагом сборки, написанных на Си программ, является препроцессирование файлов исходного текста shift.c и test.c в файлы shift.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 shift.i и test.i, параметр -E указывает, что надо остановить процесс сборки после препроцессирования.

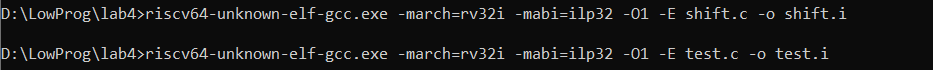
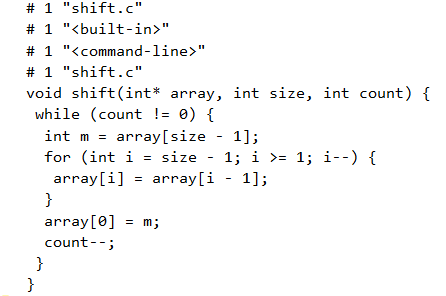


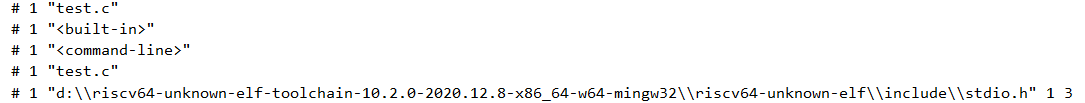
Рис. 1 Параметры запуска

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

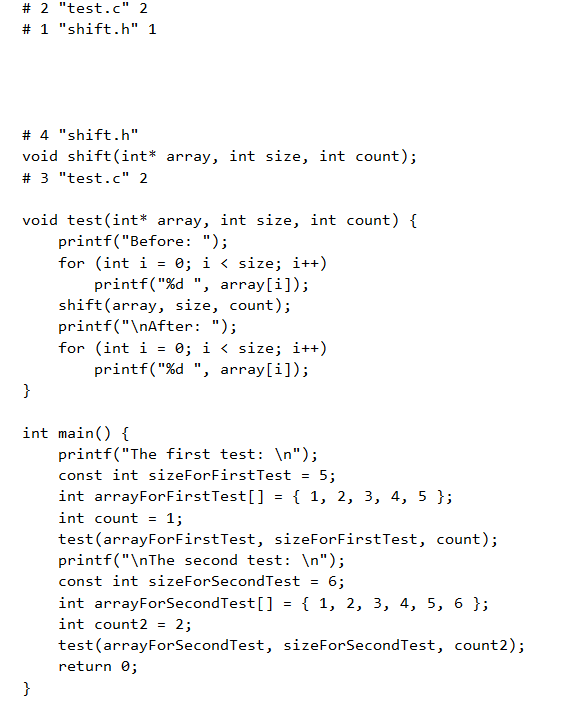
Листинг 5. shift.i



Листинг 6. test.i



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



Начинающиеся с символа “#” директивы, используются для передачи информации об исходном тексте из препроцессора в компилятор. В файле test.i четвертая директива «# 3 “test.c”» информирует компилятор о том, что следующая строка является результатом обработки строки 1 исходного файла “test.c”. В этой строке стояла команда #include “shift.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 shift из файла shift.i.

5. Компиляция файлов shift.i и test.i

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

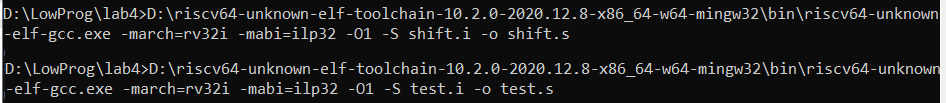
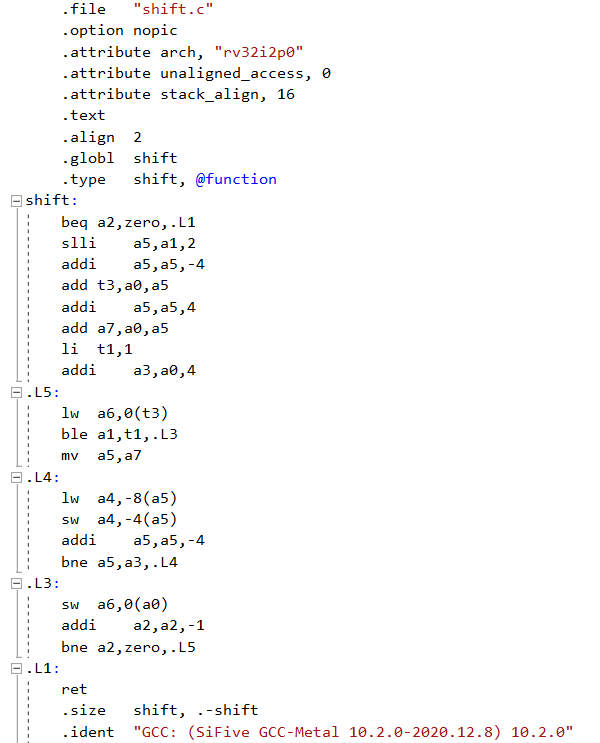


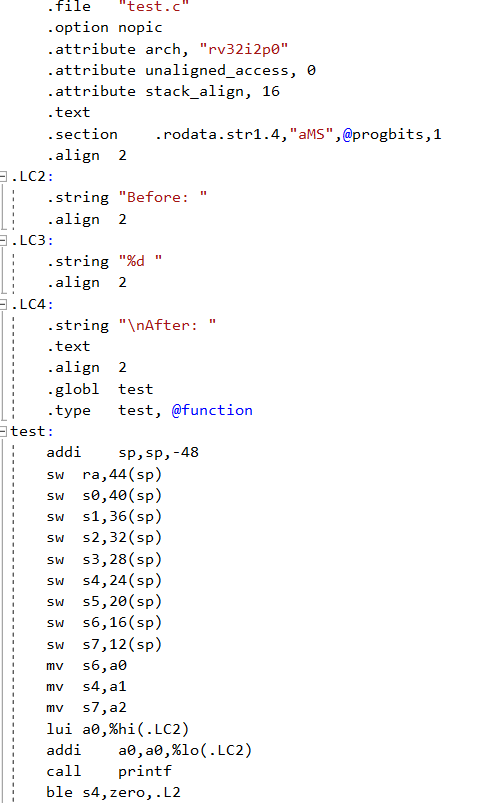
Рис. 2 Параметры запуска

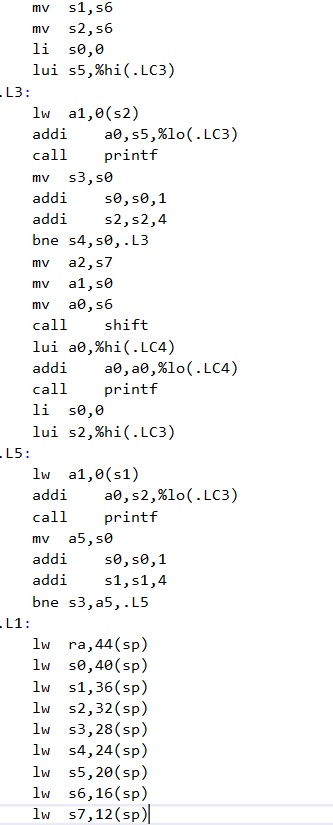
Ниже представлен код на языке ассемблера, листинг shift.s и test.s.

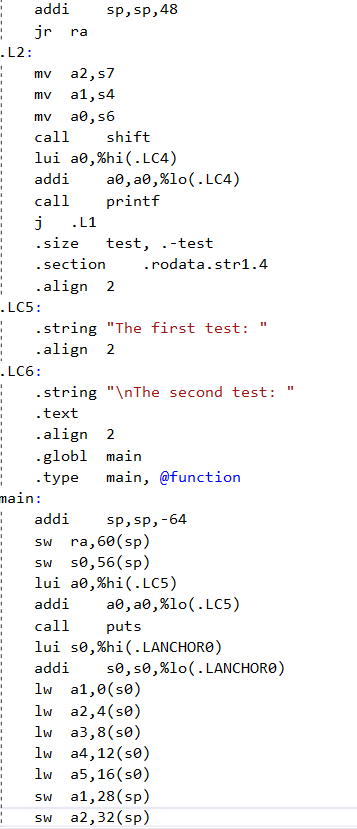
Листинг 7. shift.s

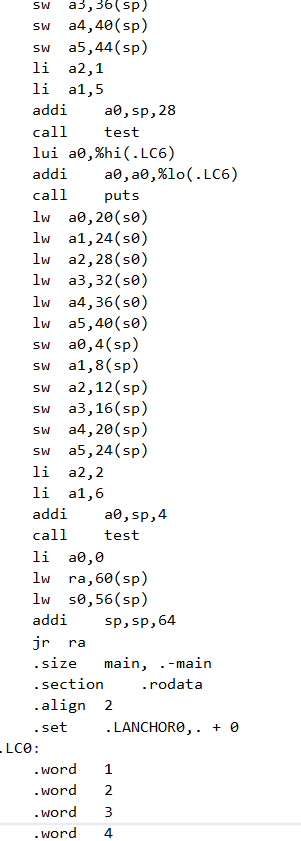


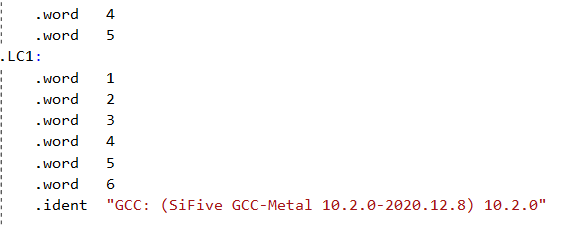
Листинг 8. test.s











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

**6. Ассемблирование файлов shift.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.

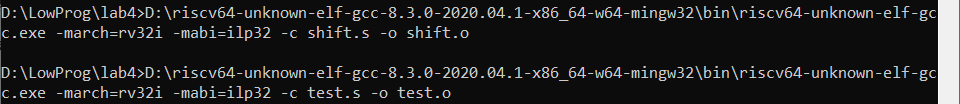


Рис. 3 Параметры запуска

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

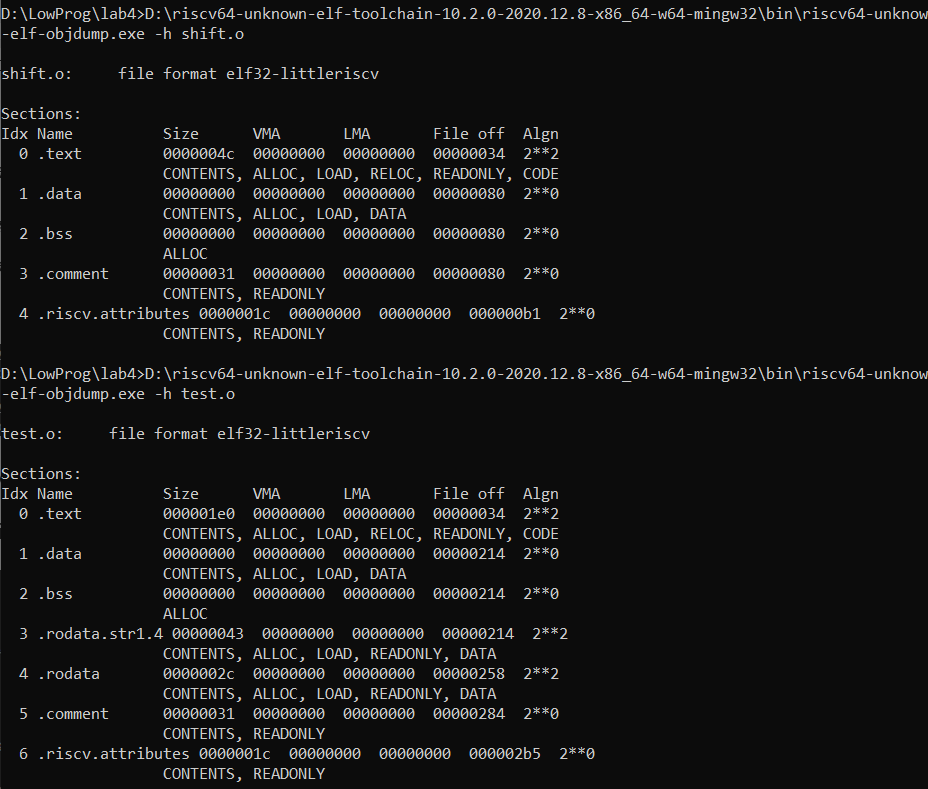


Рис. 4 Заголовки секций файлов shift.o и test.o

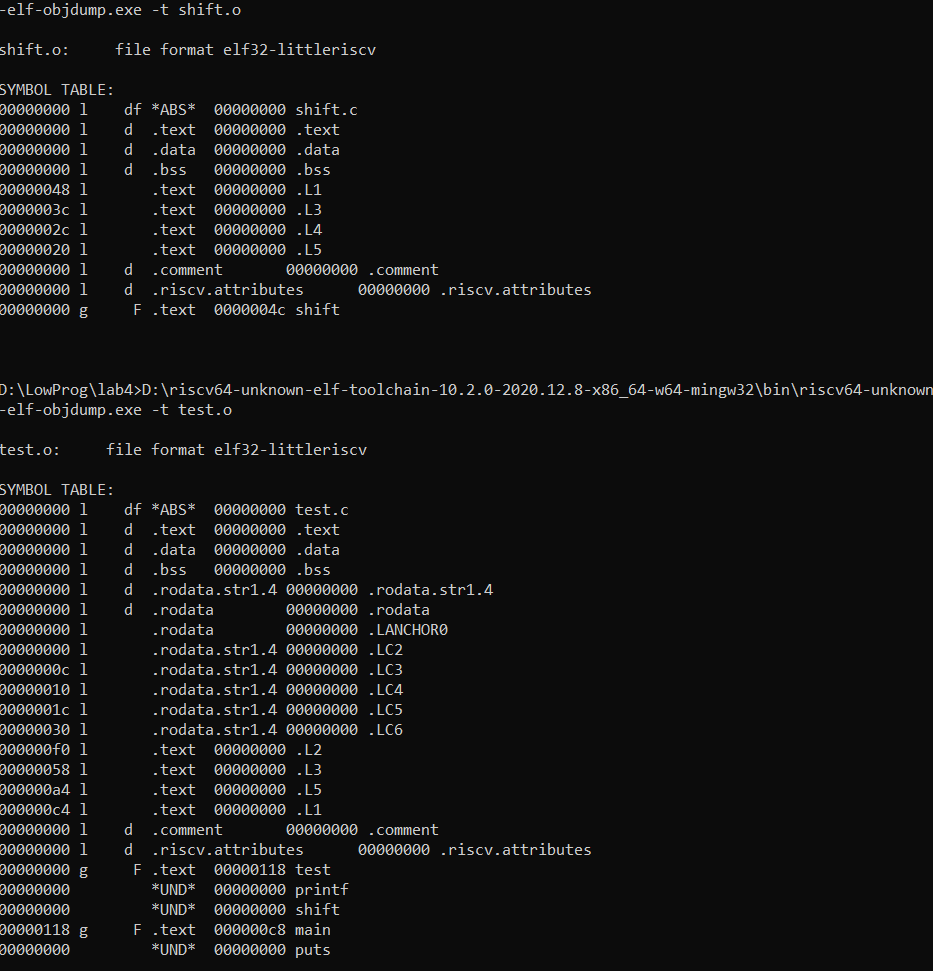


Рис. 5 Таблицы символов файлов shift.o и test.o

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

В таблице символов файла test.o содержатся два глобальных (g) символа типа функции (F) – main и test. Три символа типа \*UND\* (undefined) – shift и printf, puts. 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.

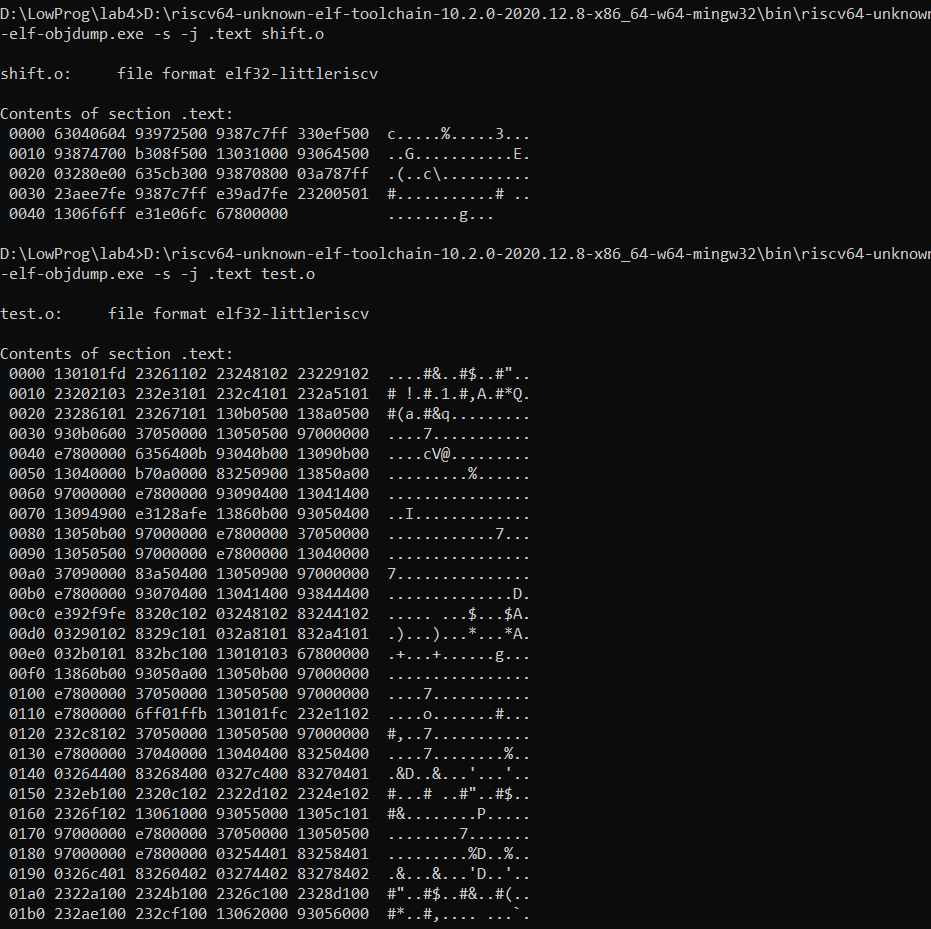


Рис. 6 Секции .text файлов shift.o и test.o

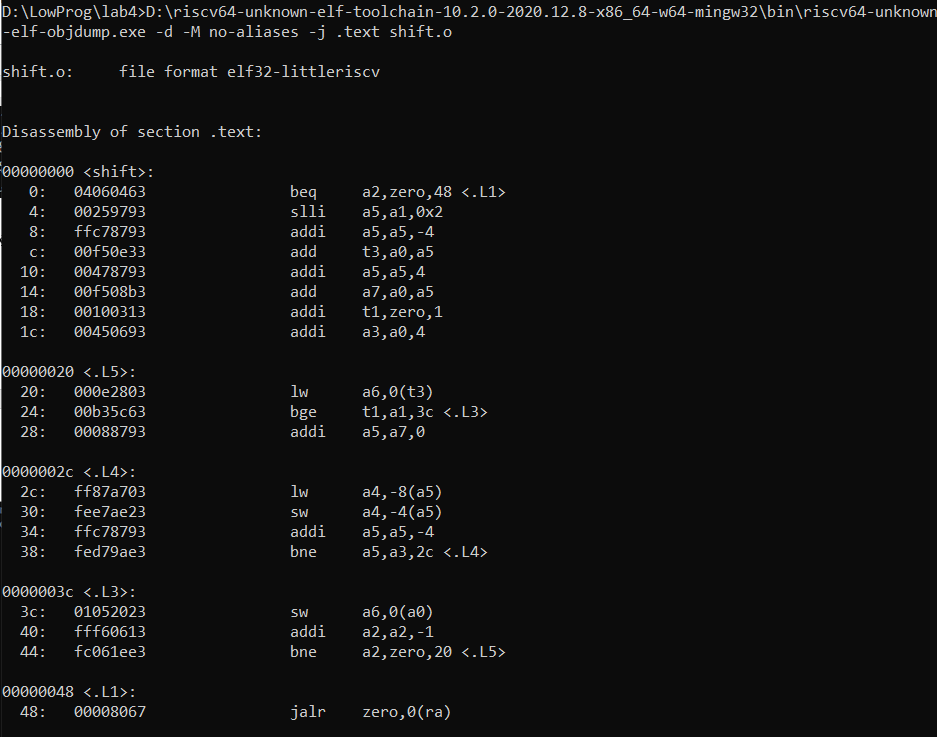
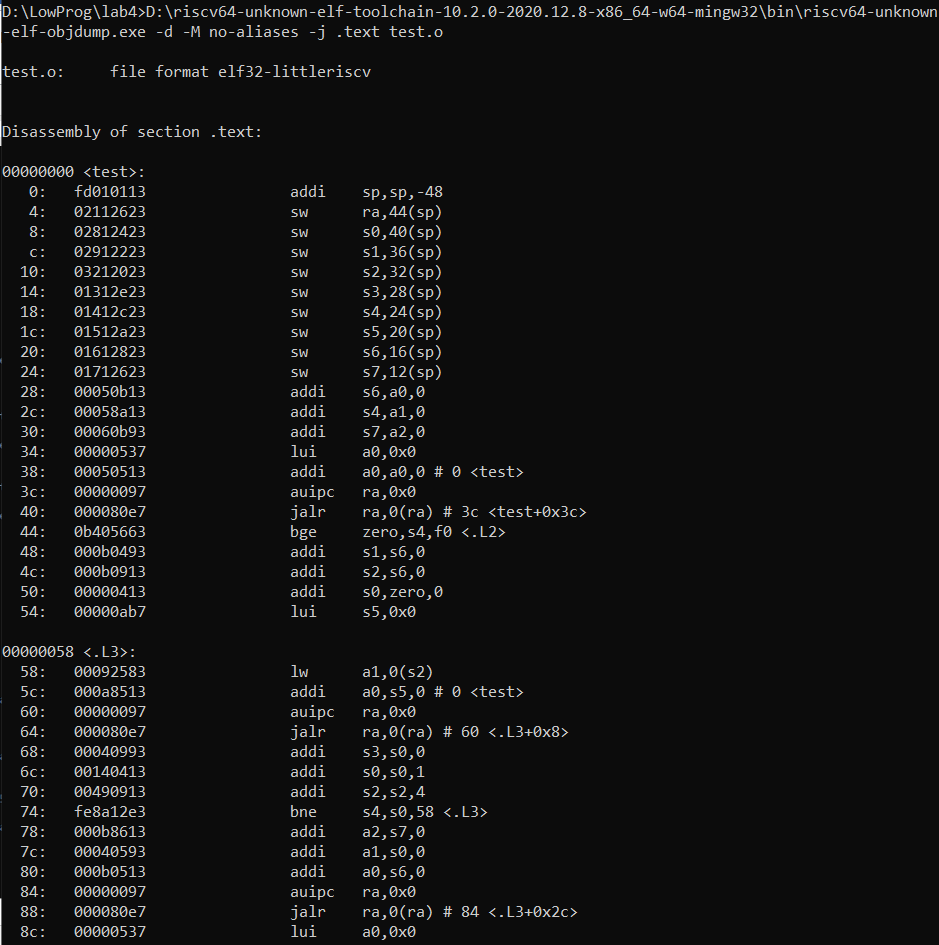
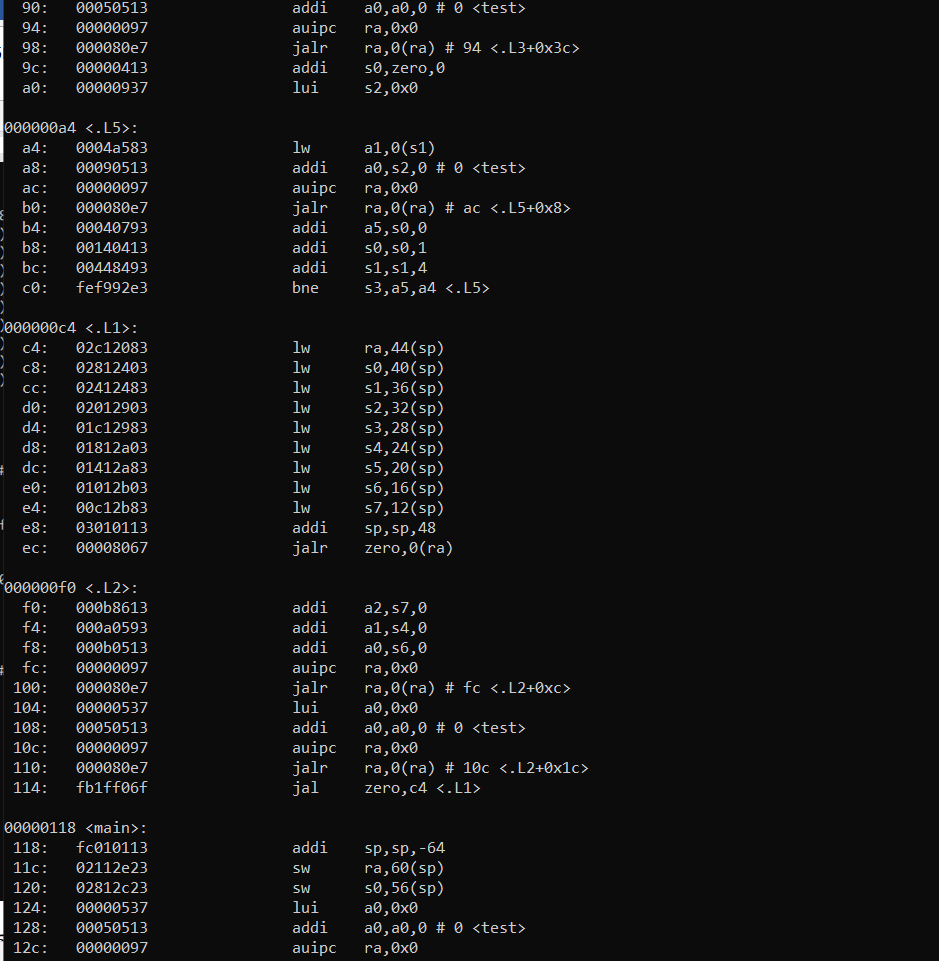


Рис. 7 Дизассемблированный код shift.o





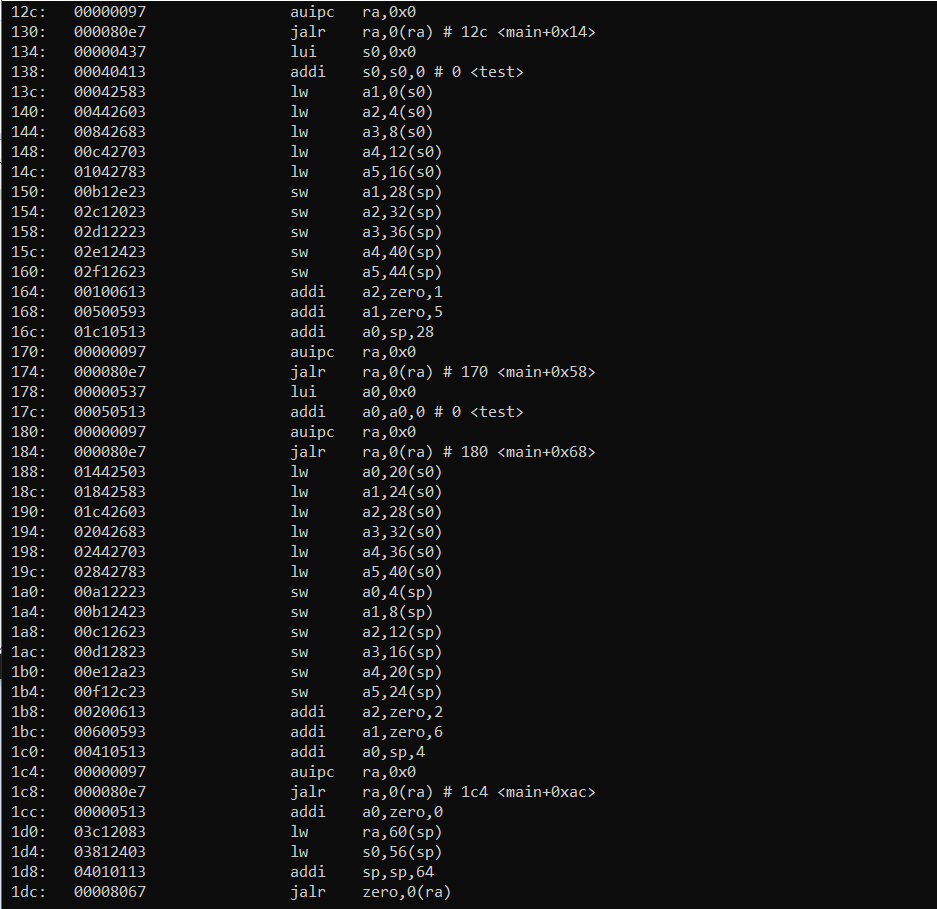


Рис. 8 Дизассемблированный код test.o

Из рисунков 7 и 8 видно, что дизассемблированный код практически идентичен сгенерированном.

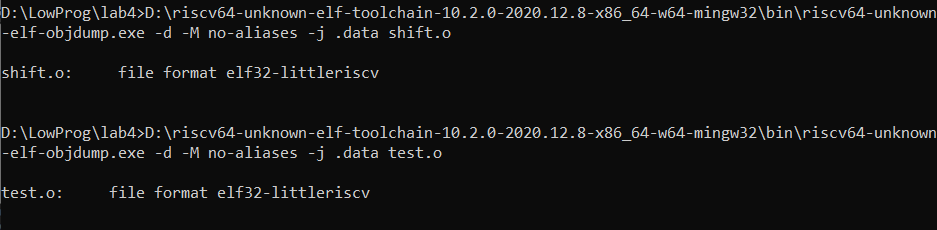


Рис. 9 Секция .data файлов shift.o и test.o

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

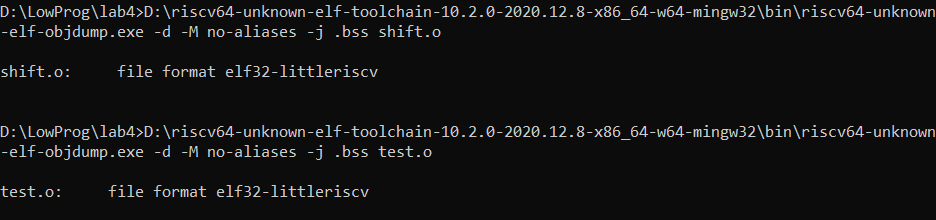


Рис. 10 Секция .bss файлов shift.o и test.o

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

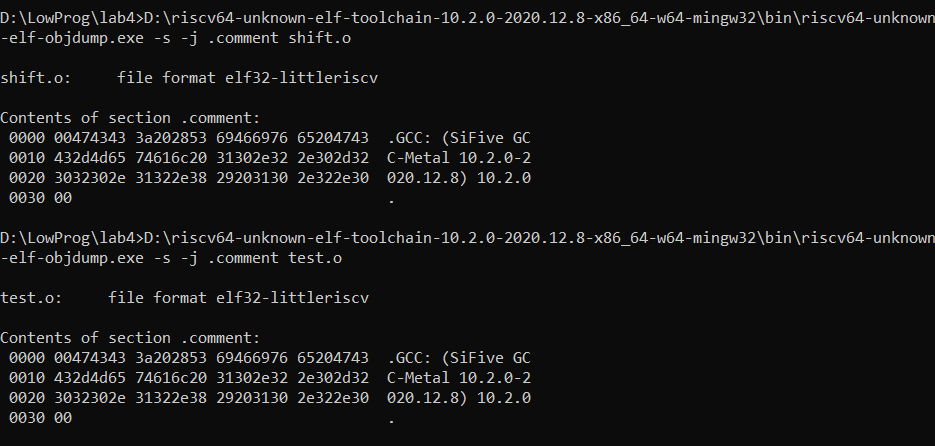


Рис. 11 Секция .comment файлов shift.o и test.o

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

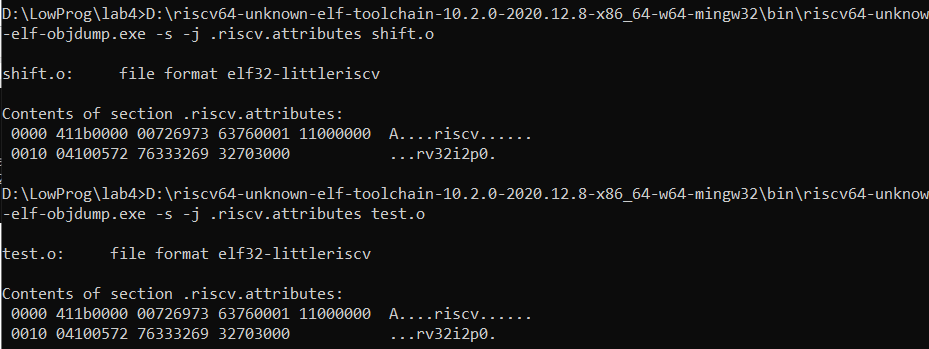
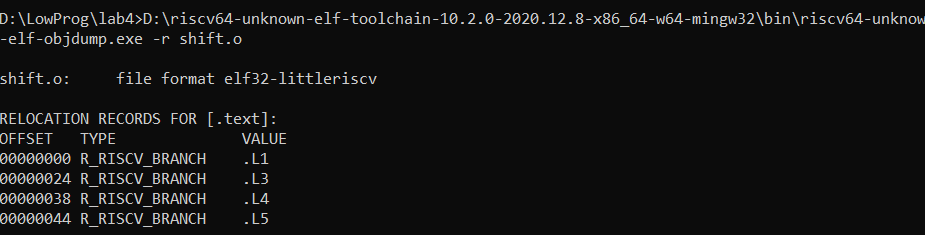
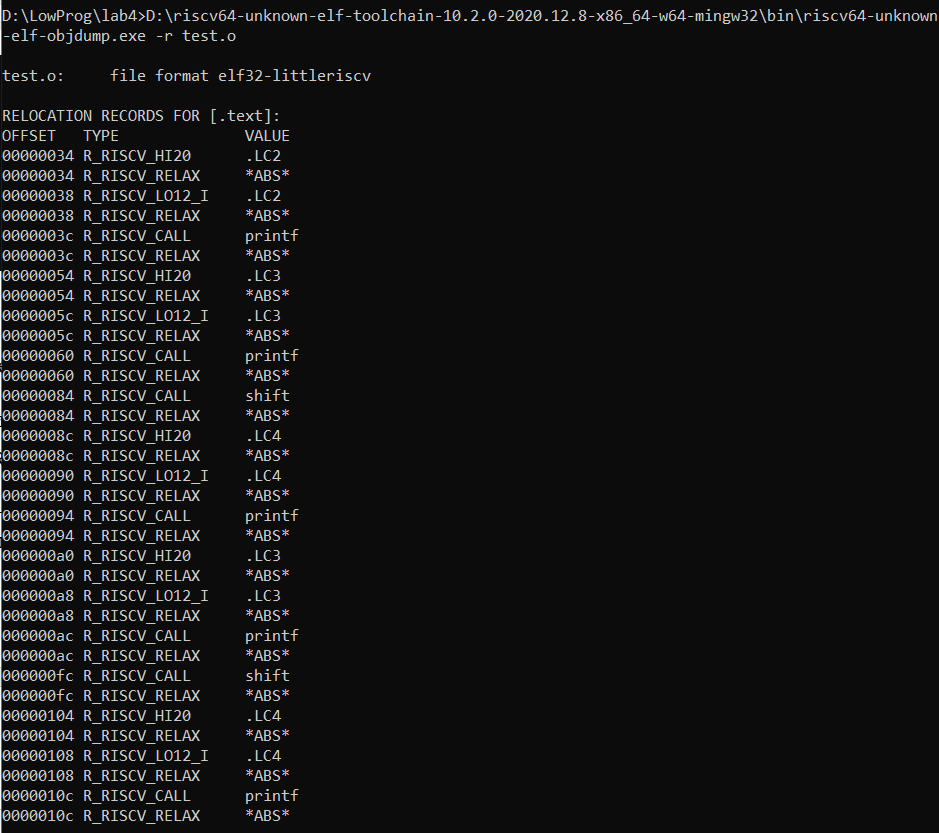


Рис. 12 Секция .riscv.attributes файлов shift.o и test.o

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





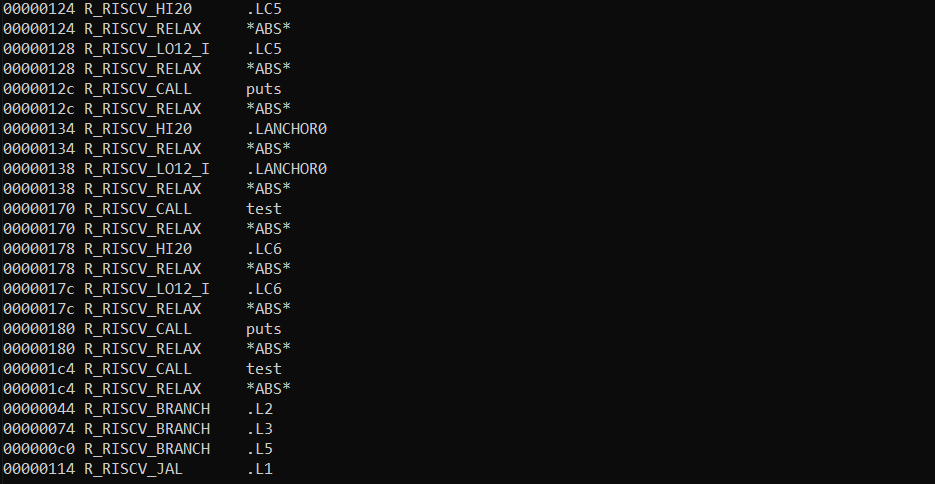


Рис. 13 Таблицы перемещения файлов shift.o и test.o

Из рисунка 13 видно, что в таблице перемещения shift.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 shift.o test.o -o shift.out* компоновщиком был сформирован файл shift.out.

Для просмотра содержимого файлов shift.out используется objdump.

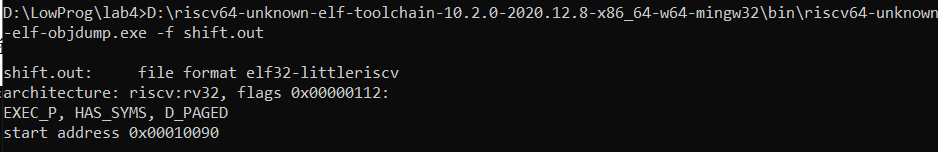


Рис. 14 Файл shift.out

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

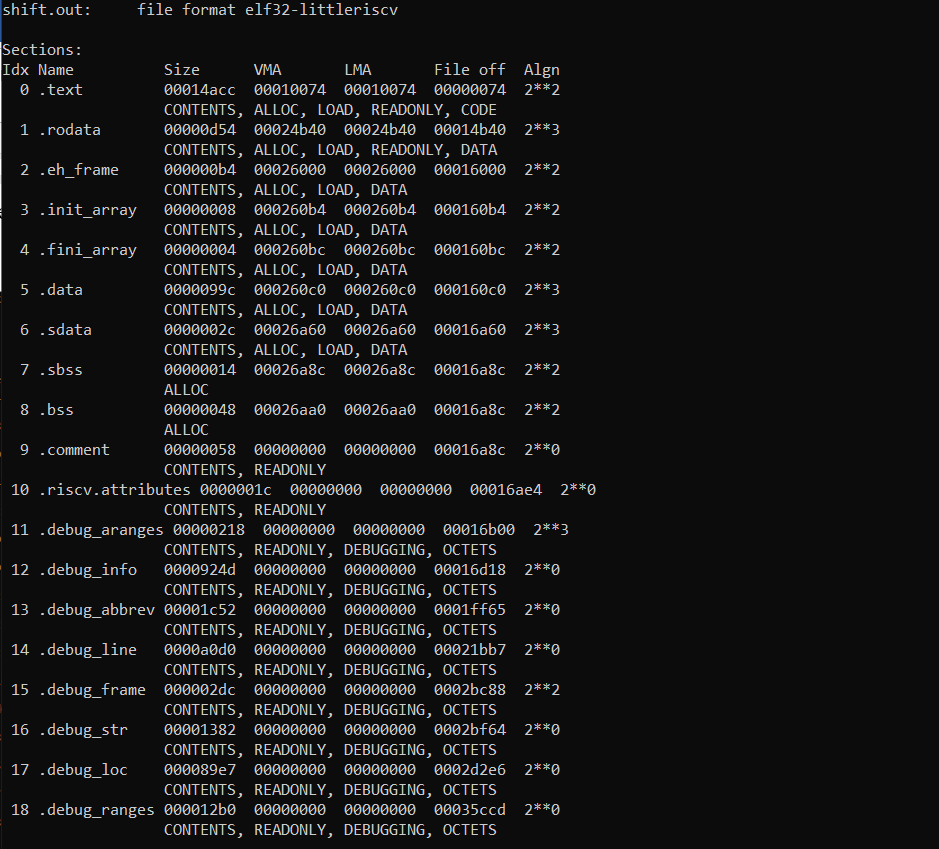


Рис. 15 Секции файла shift.out

Из рисунка 15 видно, что в файле shift.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

shift.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: 30c50513 addi a0,a0,780 # 1038c <\_\_libc\_fini\_array>

10088: 2bc0006f jal zero,10344 <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 <\_PathLocale>

1009c: 22818613 addi a2,gp,552 # 26ae8 <\_\_BSS\_END\_\_>

100a0: 40a60633 sub a2,a2,a0

100a4: 00000593 addi a1,zero,0

100a8: 3d8000ef jal ra,10480 <memset>

100ac: 00000517 auipc a0,0x0

100b0: 2e050513 addi a0,a0,736 # 1038c <\_\_libc\_fini\_array>

100b4: 290000ef jal ra,10344 <atexit>

100b8: 334000ef jal ra,103ec <\_\_libc\_init\_array>

100bc: 00012503 lw a0,0(sp)

100c0: 00410593 addi a1,sp,4

100c4: 00000613 addi a2,zero,0

100c8: 1c4000ef jal ra,1028c <main>

00010144 <shift>:

10144: 04060463 beq a2,zero,1018c <shift+0x48>

10148: 00259793 slli a5,a1,0x2

1014c: ffc78793 addi a5,a5,-4

10150: 00f50e33 add t3,a0,a5

10154: 00478793 addi a5,a5,4

10158: 00f508b3 add a7,a0,a5

1015c: 00100313 addi t1,zero,1

10160: 00450693 addi a3,a0,4

10164: 000e2803 lw a6,0(t3)

10168: 00b35c63 bge t1,a1,10180 <shift+0x3c>

1016c: 00088793 addi a5,a7,0

10170: ff87a703 lw a4,-8(a5)

10174: fee7ae23 sw a4,-4(a5)

10178: ffc78793 addi a5,a5,-4

1017c: fed79ae3 bne a5,a3,10170 <shift+0x2c>

10180: 01052023 sw a6,0(a0)

10184: fff60613 addi a2,a2,-1

10188: fc061ee3 bne a2,zero,10164 <shift+0x20>

0001028c <main>:

1028c: fc010113 addi sp,sp,-64

10290: 02112e23 sw ra,60(sp)

10294: 02812c23 sw s0,56(sp)

10298: 00025537 lui a0,0x25

1029c: b5c50513 addi a0,a0,-1188 # 24b5c <\_\_clzsi2+0x68>

102a0: 424000ef jal ra,106c4 <puts>

102a4: 00025437 lui s0,0x25

102a8: b8440413 addi s0,s0,-1148 # 24b84 <\_\_clzsi2+0x90>

102ac: 00042583 lw a1,0(s0)

102b0: 00442603 lw a2,4(s0)

102b4: 00842683 lw a3,8(s0)

102b8: 00c42703 lw a4,12(s0)

102bc: 01042783 lw a5,16(s0)

102c0: 00b12e23 sw a1,28(sp)

102c4: 02c12023 sw a2,32(sp)

102c8: 02d12223 sw a3,36(sp)

102cc: 02e12423 sw a4,40(sp)

102d0: 02f12623 sw a5,44(sp)

102d4: 00100613 addi a2,zero,1

102d8: 00500593 addi a1,zero,5

102dc: 01c10513 addi a0,sp,28

102e0: eb1ff0ef jal ra,10190 <test>

102e4: 00025537 lui a0,0x25

102e8: b7050513 addi a0,a0,-1168 # 24b70 <\_\_clzsi2+0x7c>

102ec: 3d8000ef jal ra,106c4 <puts>

102f0: 01442503 lw a0,20(s0)

102f4: 01842583 lw a1,24(s0)

102f8: 01c42603 lw a2,28(s0)

102fc: 02042683 lw a3,32(s0)

10300: 02442703 lw a4,36(s0)

10304: 02842783 lw a5,40(s0)

10308: 00a12223 sw a0,4(sp)

1030c: 00b12423 sw a1,8(sp)

10310: 00c12623 sw a2,12(sp)

10314: 00d12823 sw a3,16(sp)

10318: 00e12a23 sw a4,20(sp)

1031c: 00f12c23 sw a5,24(sp)

10320: 00200613 addi a2,zero,2

10324: 00600593 addi a1,zero,6

10328: 00410513 addi a0,sp,4

1032c: e65ff0ef jal ra,10190 <test>

10330: 00000513 addi a0,zero,0

10334: 03c12083 lw ra,60(sp)

10338: 03812403 lw s0,56(sp)

1033c: 04010113 addi sp,sp,64

10340: 00008067 jalr zero,0(ra)

00010358 <exit>:

10358: ff010113 addi sp,sp,-16

1035c: 00000593 addi a1,zero,0

10360: 00812423 sw s0,8(sp)

10364: 00112623 sw ra,12(sp)

10368: 00050413 addi s0,a0,0

1036c: 1bc030ef jal ra,13528 <\_\_call\_exitprocs>

10370: 1b818793 addi a5,gp,440 # 26a78 <\_global\_impure\_ptr>

10374: 0007a503 lw a0,0(a5)

10378: 03c52783 lw a5,60(a0)

1037c: 00078463 beq a5,zero,10384 <exit+0x2c>

10380: 000780e7 jalr ra,0(a5)

10384: 00040513 addi a0,s0,0

10388: 5e80f0ef jal ra,1f970 <\_exit>

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

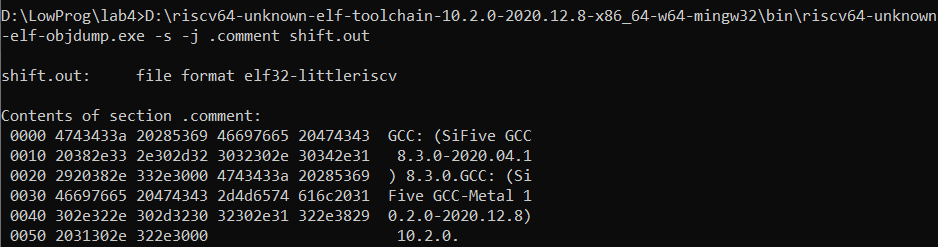


Рис. 16 Секция .comment файла shift.out

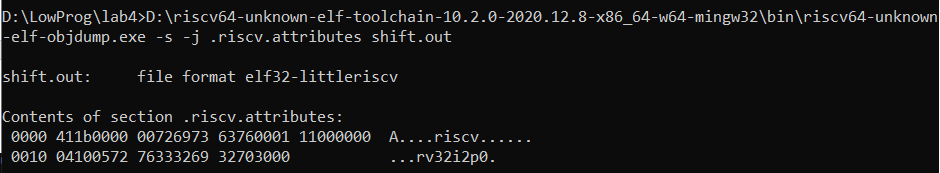


Рис. 17 Секция .riscv.attributes файла shift.out

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

Листинг 9. Таблица символов файла shift.out

D:\LowProg\lab4>D:\riscv64-unknown-elf-toolchain-10.2.0-2020.12.8-x86\_64-w64-mingw32\bin\riscv64-unknown-elf-objdump.exe -t shift.out

shift.out: file format elf32-littleriscv

SYMBOL TABLE:

00010074 l d .text 00000000 .text

00024b40 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 shift.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 puts.c

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

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

00013264 l F .text 000000c0 \_\_sbprintf

00024d20 l O .rodata 00000010 blanks.4504

00024d30 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

00013928 l F .text 00000008 \_\_fp\_lock

0001393c l F .text 00000184 \_\_sinit.part.0

00013ac0 l F .text 00000008 \_\_fp\_unlock

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

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

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

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

000146ec l F .text 00000068 eshdn1

00014754 l F .text 00000070 eshup1

000147c4 l F .text 000000e0 m16m

000148a4 l F .text 00000024 eisnan.part.0

000148c8 l F .text 0000004c eneg

00014914 l F .text 00000040 eisneg

00014954 l F .text 000000e8 emovi

00014a3c l F .text 0000011c ecmp

00014b58 l F .text 0000001c eisinf.part.1

00014b74 l F .text 000001ac eshift.part.3

00014d20 l F .text 0000017c enormlz

00014e9c l F .text 00000408 emdnorm

000152a4 l F .text 00000164 eiremain

00015408 l F .text 000000b0 emovo.isra.6

000154b8 l F .text 00000408 emul

000158c0 l F .text 00000504 ediv

00015dc4 l F .text 00000144 e113toe.isra.8

00024d84 l O .rodata 00000014 ezero

00024d98 l O .rodata 00000014 eone

00024dac l O .rodata 00000104 etens

00024eb0 l O .rodata 00000104 emtens

00024fb4 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 memcpy.c

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

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

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

00025010 l O .rodata 0000000c p05.3298

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

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 strncpy.c

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

000252a4 l O .rodata 00000010 blanks.4489

000252b4 l O .rodata 00000010 zeroes.4490

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

0001c7b8 l F .text 000000f0 \_\_sprint\_r.part.0

0001dba8 l F .text 000000c0 \_\_sbprintf

00025430 l O .rodata 00000010 blanks.4480

00025440 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 isattyr.c

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

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

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

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

000256c0 l O .rodata 00000010 blanks.4466

000256d0 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

00018f94 g F .text 0000009c \_mprec\_log10

00019098 g F .text 00000078 \_\_any\_on

0001e178 g F .text 00000054 \_isatty\_r

000250e8 g O .rodata 00000028 \_\_mprec\_tinytens

00019ba4 g F .text 000000b0 strcpy

0001e28c g F .text 00000040 cleanup\_glue

000105f4 g F .text 000000d0 \_puts\_r

0001e1cc g F .text 00000060 \_lseek\_r

00021674 g F .text 00000144 .hidden \_\_getf2

000215a8 g F .text 000000cc .hidden \_\_eqtf2

000105a0 g F .text 00000054 printf

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

0001f810 g F .text 00000078 \_wcrtomb\_r

000199b8 g F .text 00000068 \_\_sseek

00013c58 g F .text 00000010 \_\_sinit

0001f678 g F .text 00000184 \_\_swbuf\_r

00017278 g F .text 0000007c \_setlocale\_r

00013ac8 g F .text 00000078 \_\_sfmoreglue

0001805c g F .text 00000004 \_\_malloc\_unlock

00024358 g F .text 00000188 .hidden \_\_floatsitf

00010144 g F .text 0000004c shift

00017f38 g F .text 00000120 memmove

00013c44 g F .text 00000014 \_cleanup

00018060 g F .text 000000a8 \_Balloc

0001fc98 g F .text 0000000c \_\_errno

0001fbf4 g F .text 000000a4 \_conv\_stat

00017260 g F .text 00000008 \_\_localeconv\_l

0001e11c g F .text 0000005c \_fstat\_r

00026ae4 g O .bss 00000004 errno

00019928 g F .text 00000008 \_\_seofread

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

00017e1c g F .text 0000011c memcpy

00013930 g F .text 0000000c \_cleanup\_r

00019d20 g F .text 00002a98 \_svfprintf\_r

000106c4 g F .text 00000010 puts

00018ef0 g F .text 000000a4 \_\_ratio

0001055c g F .text 00000044 \_printf\_r

000217b8 g F .text 00000144 .hidden \_\_letf2

00024a48 g F .text 00000048 .hidden \_\_udivsi3

0001f9b4 g F .text 00000070 \_fstat

00025020 g O .rodata 000000c8 \_\_mprec\_tens

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

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

00017268 g F .text 00000008 \_localeconv\_r

000184c0 g F .text 00000034 \_\_i2b

000140c0 g F .text 000004bc \_\_sfvwrite\_r

000196e0 g F .text 00000054 \_sbrk\_r

0001e22c g F .text 00000060 \_read\_r

0001de20 g F .text 00000110 \_fclose\_r

00013900 g F .text 00000028 fflush

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

00018c94 g F .text 00000118 \_\_b2d

00020278 g F .text 000004f0 .hidden \_\_umoddi3

0001fa24 g F .text 00000040 \_isatty

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

00019110 g F .text 000005d0 \_realloc\_r

000103ec g F .text 00000094 \_\_libc\_init\_array

0001fca4 g F .text 000005d4 .hidden \_\_udivdi3

0001e06c g F .text 0000002c \_fputwc\_r

00025110 g O .rodata 00000028 \_\_mprec\_bigtens

00018284 g F .text 00000110 \_\_s2b

0001fb04 g F .text 000000a0 \_sbrk

000189f4 g F .text 0000005c \_\_mcmp

00013c78 g F .text 00000014 \_\_fp\_lock\_all

0001038c g F .text 00000060 \_\_libc\_fini\_array

0001e2cc g F .text 00000110 \_reclaim\_reent

00018394 g F .text 00000074 \_\_hi0bits

00024208 g F .text 00000150 .hidden \_\_fixtfsi

00018760 g F .text 00000148 \_\_pow5mult

00024a90 g F .text 00000010 .hidden \_\_umodsi3

00024af4 g F .text 0000004c .hidden \_\_clzsi2

00013c68 g F .text 00000004 \_\_sfp\_lock\_acquire

00017d40 g F .text 000000dc memchr

000197e4 g F .text 0000006c \_sprintf\_r

00013dc8 g F .text 000002f8 \_free\_r

000172f4 g F .text 00000008 \_\_locale\_mb\_cur\_max

00013528 g F .text 0000011c \_\_call\_exitprocs

00010190 g F .text 000000fc test

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

00010090 g F .text 00000040 \_start

0001fa64 g F .text 00000050 \_lseek

000188a8 g F .text 0000014c \_\_lshift

000217b8 g F .text 00000144 .hidden \_\_lttf2

000215a8 g F .text 000000cc .hidden \_\_netf2

0001e3dc g F .text 000001bc \_\_ssprint\_r

0001348c g F .text 0000009c \_\_register\_exitproc

000171fc g F .text 00000064 \_ldcheck

000184f4 g F .text 0000026c \_\_multiply

00024a1c g F .text 00000024 .hidden \_\_mulsi3

00019c54 g F .text 000000cc strncpy

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

00018dac g F .text 00000144 \_\_d2b

000244e0 g F .text 00000208 .hidden \_\_extenddftf2

0001ddcc g F .text 00000054 \_close\_r

00013324 g F .text 00000168 \_\_swsetup\_r

00020768 g F .text 000007d0 .hidden \_\_divdf3

00013b40 g F .text 00000104 \_\_sfp

00019030 g F .text 00000068 \_\_copybits

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

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

00013c74 g F .text 00000004 \_\_sinit\_lock\_release

00020f38 g F .text 00000670 .hidden \_\_muldf3

000198cc g F .text 0000005c \_\_sread

00018058 g F .text 00000004 \_\_malloc\_lock

000138a0 g F .text 00000060 \_fflush\_r

0001dd04 g F .text 000000c8 \_calloc\_r

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

00010480 g F .text 000000dc memset

0001028c g F .text 000000b8 main

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

0001f7fc g F .text 00000014 \_\_swbuf

00019a20 g F .text 00000008 \_\_sclose

0001df30 g F .text 00000010 fclose

00017500 g F .text 000007cc \_malloc\_r

0001dc74 g F .text 00000030 \_\_ascii\_wctomb

0001457c g F .text 000000b0 \_fwalk

00017ccc g F .text 0000000c \_mbtowc\_r

00024a40 g F .text 00000084 .hidden \_\_divsi3

00013ca0 g F .text 00000128 \_malloc\_trim\_r

00019a28 g F .text 0000017c strcmp

0001db90 g F .text 00000018 vfiprintf

000218fc g F .text 0000136c .hidden \_\_multf3

00019850 g F .text 0000007c sprintf

00025794 g O .rodata 00000100 .hidden \_\_clz\_tab

00026a8c g O .sbss 00000004 \_PathLocale

00010344 g F .text 00000014 atexit

0001dca4 g F .text 00000060 \_write\_r

000172fc g F .text 00000014 setlocale

00026a80 g O .sdata 00000004 \_impure\_ptr

00013644 g F .text 0000025c \_\_sflush\_r

00021674 g F .text 00000144 .hidden \_\_gttf2

0001e598 g F .text 000010e0 \_svfiprintf\_r

00017cd8 g F .text 00000068 \_\_ascii\_mbtowc

00022c68 g F .text 000015a0 .hidden \_\_subtf3

00018c34 g F .text 00000060 \_\_ulp

00013c8c g F .text 00000014 \_\_fp\_unlock\_all

00017270 g F .text 00000008 localeconv

00017310 g F .text 000000d0 \_\_swhatbuf\_r

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

0001fba4 g F .text 00000050 \_write

00026a8c g .sdata 00000000 \_edata

00026ae8 g .bss 00000000 \_end

0001df40 g F .text 0000012c \_\_fputwc

00019930 g F .text 00000088 \_\_swrite

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

00010358 g F .text 00000034 exit

0001c8c0 g F .text 000012d0 \_vfiprintf\_r

0001462c g F .text 000000c0 \_fwalk\_reent

00018a50 g F .text 000001e4 \_\_mdiff

00024ac4 g F .text 00000030 .hidden \_\_modsi3

00013c6c g F .text 00000004 \_\_sfp\_lock\_release

00015f08 g F .text 000012f4 \_ldtoa\_r

00025450 g O .rodata 00000101 \_ctype\_

0001fab4 g F .text 00000050 \_read

0001f970 g F .text 00000044 \_exit

000173e0 g F .text 00000120 \_\_smakebuf\_r

000106d4 g F .text 0000008c strlen

0001c8a8 g F .text 00000018 \_\_sprint\_r

0001dc68 g F .text 0000000c \_wctomb\_r

00010760 g F .text 00002aec \_vfprintf\_r

00018408 g F .text 000000b8 \_\_lo0bits

0001f888 g F .text 00000090 wcrtomb

00019734 g F .text 000000b0 frexp

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

0001324c g F .text 00000018 vfprintf

000246e8 g F .text 00000334 .hidden \_\_trunctfdf2

0001e098 g F .text 00000084 fputwc

0001f918 g F .text 00000058 \_close

00013c70 g F .text 00000004 \_\_sinit\_lock\_acquire

0001812c g F .text 00000158 \_\_multadd

00018108 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 shift, 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.

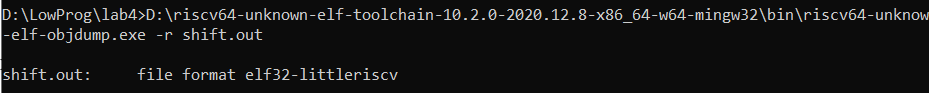


Рис. 18 Таблица перемещений файла shift.out

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.

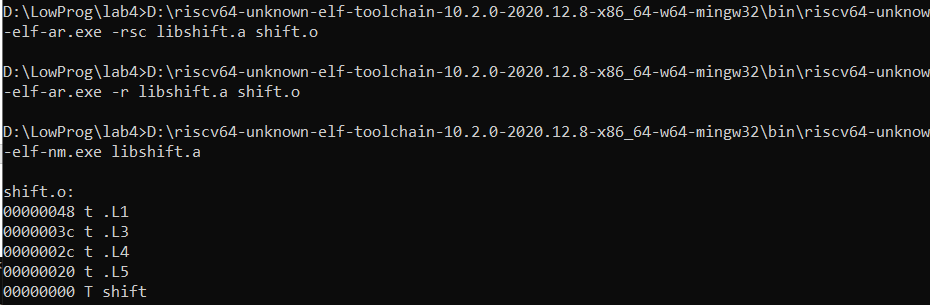


Рис. 19 Создание библиотеки libfib.a

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



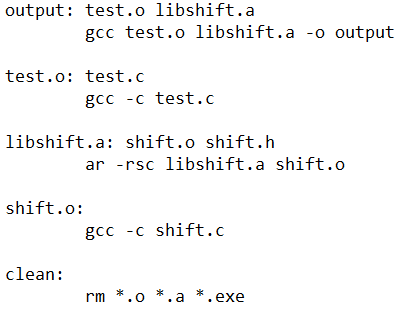
Рис. 20 Поиск библиотеки и сборка программы

После поиска библиотеки и сборки программы получается файл shift.

**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



Что происходит в makefile:

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

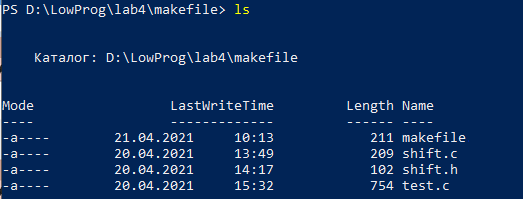
1

Рис. 21 Директория с makefile

Из рисунка 21 видно, что в директории 4 файла.

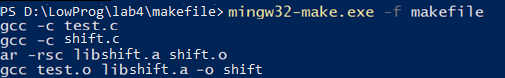


Рис. 22 Запуск makefile и процесс сборки

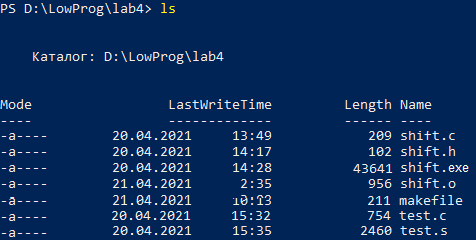


Рис. 23 Состояние директории после запуска makefile

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

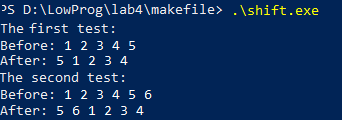


Рис. 24 Запуск shift.exe

**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.