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.М. Ковалевский

(подпись)

Преподаватель А.О. Алексюк

(подпись)

“ ” 2021 г.

Санкт-Петербург

2021

**СОДЕРЖАНИЕ**

[1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 3](#_Toc351429464)

[2. МЕТОД РЕШЕНИЯ 4](#_Toc1687912337)

[2.1. Текст программ и описание алгоритма 4](#_Toc918429310)

[2.2. Сборка программ по шагам, анализ промежуточных и результирующих файлов 6](#_Toc1030487446)

[2.3. Формирование статической библиотеки, разработка make-файлов для сборки библиотеки. 29](#_Toc611663402)

[3. Выводы 32](#_Toc78793466)

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ

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 иисполняемом файле.

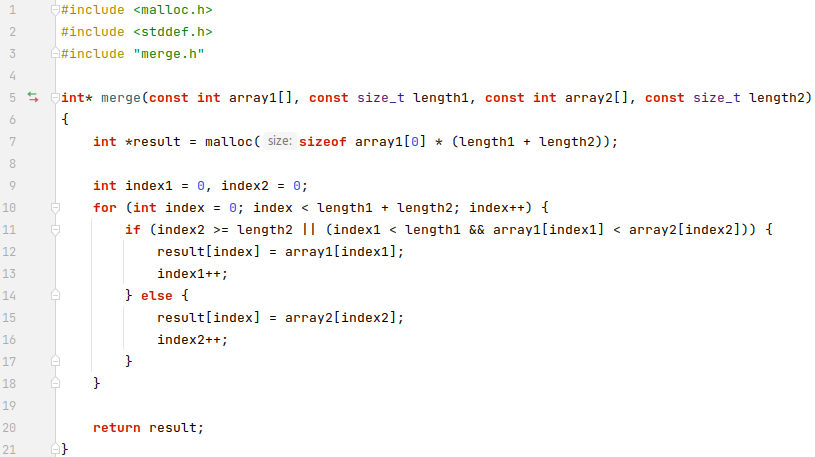
3) Выделить разработанную функцию в статическую библиотеку. Разработать make-файлы для сборки библиотеки и использующей ее тестовой программы. Проанализировать ход сборки библиотеки и программы, созданные файлы зависимостей.

**ВАРИАНТ:** Слияние двух отсортированных массивов.

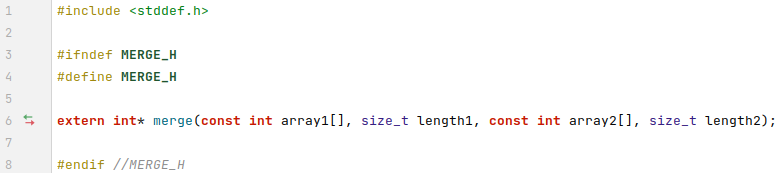
# МЕТОД РЕШЕНИЯ

## Текст программ и описание алгоритма

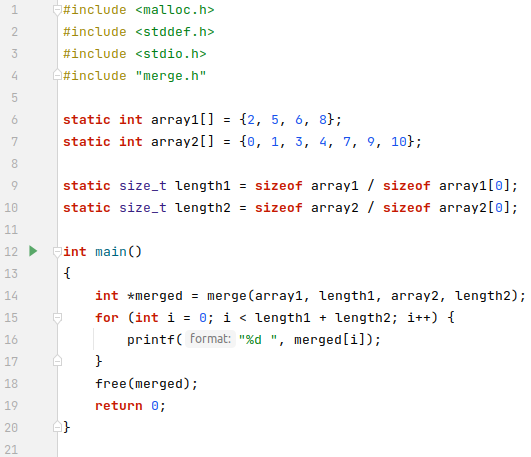
Листинг 1. Программа “merge.c”



Листинг 2. Программа “merge.h”



Листинг 3. Программа “main.c”



Суть алгоритма в следующем. Достаем очередные элементы двух массивов. Записываем в выходной массив меньший из них, увеличиваем индекс массива, из которого мы взяли элемент. Если же один из массивов кончился - добавляем элемент из другого массива.

В файле “merge.c” реализована функция **merge()**, в которую передаются массивы и их размеры (так как в С высчитать их размеры внутри функции нельзя). В начале в функции инициализируется выходной массив длины **length1 + length2** (длины 1 и 2 массивов).

Заголовочный файл “merge.h” содержит в себе определение функции **merge()**. Для использования этой функции в другой программе необходимо обеспечить подключение этого заголовочного файла и компиляцию исходного файла “merge.c” вместе с использующей ее программой.

В файле “test.c” реализована функция **main()**, с которой начинается исполнение. Мы вызываем функцию **merge()**, и сохраняем результат в массив **merged**, занятую им память следует освободить через функцию **free()** в конце программы.

## Сборка программ по шагам, анализ промежуточных и результирующих файлов

Осуществим сборку программ на С по шагам под систему команд RISC-V.

Сначала следует препроцессировать файлы исходного текста “main.c” и “merge.c”:



Драйвер компилятора gcc - *riscv64-unknown-elf-gcc* - запускается с параметрами командной строки *“-march=rv32imafdc -mabi=ilp32”*, указывающих что целевым является процессор с базовой архитектурой системы команд RV32I; O1 – указание выполнять простые оптимизации генерируемого кода, это позволяет получить более простой код; -E – указание остановить процесс сборки после препроцессирования.

Получим файлы со следующим содержимым:

Листинг 4. Файл “merge.i”

*# 1 "merge.c"*

*# 1 "<built-in>"*

*# 1 "<command-line>"*

*# 1 "merge.c"*

*int\* merge(const int array1[], const size\_t length1, const int array2[], const size\_t length2)*

*{*

*int \*result = malloc(sizeof array1[0] \* (length1 + length2));*

*int index1 = 0, index2 = 0;*

*for (int index = 0; index < length1 + length2; index++) {*

*if (index2 >= length2 || (index1 < length1 && array1[index1] < array2[index2])) {*

*result[index] = array1[index1];*

*index1++;*

*} else {*

*result[index] = array2[index2];*

*index2++;*

*}*

*}*

*return result;*

*}*

Листинг 5. Файл “main.i”

*# 1 "main.c"*

*# 1 "<built-in>"*

*# 1 "<command-line>"*

*# 1 "main.c"*

*# 1 "merge.h" 1*

*extern int\* merge(const int array1[], size\_t length1, const int array2[], size\_t length2);*

*# 2 "main.c" 2*

*static int array1[] = {2, 5, 6, 8};*

*static int array2[] = {0, 1, 3, 4, 7, 9, 10};*

*static size\_t length1 = sizeof array1 / sizeof array1[0];*

*static size\_t length2 = sizeof array2 / sizeof array2[0];*

*int main()*

*{*

*int \*merged = merge(array1, length1, array2, length2);*

*for (int i = 0; i < length1 + length2; i++) {*

*printf("%d ", merged[i]);*

*}*

*free(merged);*

*return 0;*

*}*

Появившиеся нестандартные директивы, начинающиеся с символа “#”, используются для передачи информации об исходном тексте из препроцессора в компилятор; например, последняя директива «# 1 “main.c”» в файле **main.i** информирует компилятор о том, что следующая строка является результатом обработки строки 1 исходного файла “main.c”. Аналогично директива «# 1 “merge.c”» в файле **merge.i** информирует компилятор о том, что следующая строка является результатом обработки строки 1 исходного файла “merge.c”.

Директива «# 1 “merge.h” 1» заменила собой директиву «#include merge.h» в файле **main.i**, здесь препроцессор произвел вставку содержимого этого заголовочного файла. Соответственно далее идет объявление функции **merge()**. С директивы «# 2 "main.c" 2» начинается описание самого содержимого файла.

Исходные коды функций остались без изменений.

Следующим шагом является компиляция файлов “merge.i” и “main.i” в код на языке ассемблера RISC-V “merge.s” и “main.s”:



Листинг 6. Файл “merge.s”

*.file "merge.c"*

*.option nopic*

*.attribute arch, "rv32i2p0\_m2p0\_a2p0\_f2p0\_d2p0\_c2p0"*

*.attribute unaligned\_access, 0*

*.attribute stack\_align, 16*

*.text*

*.align 1*

*.globl merge*

*.type merge, @function*

*merge:*

*addi sp,sp,-32*

*sw ra,28(sp)*

*sw s0,24(sp)*

*sw s1,20(sp)*

*sw s2,16(sp)*

*sw s3,12(sp)*

*sw s4,8(sp)*

*mv s1,a0*

*mv s2,a1*

*mv s3,a2*

*mv s0,a3*

*add s4,a1,a3*

*slli a0,s4,2*

*call malloc*

*beq s4,zero,.L1*

*mv a6,a0*

*mv a3,s4*

*li a4,0*

*li a7,0*

*li t1,0*

*j .L6*

*.L3:*

*slli a5,t1,2*

*add a5,s1,a5*

*lw a5,0(a5)*

*sw a5,0(a6)*

*addi t1,t1,1*

*.L5:*

*addi a4,a4,1*

*addi a6,a6,4*

*beq a3,a4,.L1*

*.L6:*

*mv a5,a7*

*bgeu a7,s0,.L3*

*bgeu t1,s2,.L4*

*slli a1,t1,2*

*add a1,s1,a1*

*slli a2,a7,2*

*add a2,s3,a2*

*lw a1,0(a1)*

*lw a2,0(a2)*

*blt a1,a2,.L3*

*.L4:*

*slli a5,a5,2*

*add a5,s3,a5*

*lw a5,0(a5)*

*sw a5,0(a6)*

*addi a7,a7,1*

*j .L5*

*.L1:*

*lw ra,28(sp)*

*lw s0,24(sp)*

*lw s1,20(sp)*

*lw s2,16(sp)*

*lw s3,12(sp)*

*lw s4,8(sp)*

*addi sp,sp,32*

*jr ra*

*.size merge, .-merge*

*.ident "GCC: (SiFive GCC-Metal 10.2.0-2020.12.8) 10.2.0"*

По метке “merge” начинается тело функции.

Листинг 7. Файл “main.s”

*.file "main.c"*

*.option nopic*

*.attribute arch, "rv32i2p0\_m2p0\_a2p0\_f2p0\_d2p0\_c2p0"*

*.attribute unaligned\_access, 0*

*.attribute stack\_align, 16*

*.text*

*.section .rodata.str1.4,"aMS",@progbits,1*

*.align 2*

*.LC0:*

*.string "%d "*

*.text*

*.align 1*

*.globl main*

*.type main, @function*

*main:*

*addi sp,sp,-32*

*sw ra,28(sp)*

*sw s0,24(sp)*

*sw s1,20(sp)*

*sw s2,16(sp)*

*sw s3,12(sp)*

*lui a2,%hi(.LANCHOR0)*

*addi a0,a2,%lo(.LANCHOR0)*

*li a3,7*

*addi a2,a2,%lo(.LANCHOR0)*

*li a1,4*

*addi a0,a0,28*

*call merge*

*mv s3,a0*

*mv s0,a0*

*addi s2,a0,44*

*lui s1,%hi(.LC0)*

*.L2:*

*lw a1,0(s0)*

*addi a0,s1,%lo(.LC0)*

*call printf*

*addi s0,s0,4*

*bne s0,s2,.L2*

*mv a0,s3*

*call free*

*li a0,0*

*lw ra,28(sp)*

*lw s0,24(sp)*

*lw s1,20(sp)*

*lw s2,16(sp)*

*lw s3,12(sp)*

*addi sp,sp,32*

*jr ra*

*.size main, .-main*

*.data*

*.align 2*

*.set .LANCHOR0,. + 0*

*.type array2, @object*

*.size array2, 28*

*array2:*

*.word 0*

*.word 1*

*.word 3*

*.word 4*

*.word 7*

*.word 9*

*.word 10*

*.type array1, @object*

*.size array1, 16*

*array1:*

*.word 2*

*.word 5*

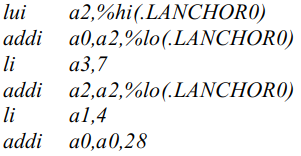
*.word 6*

*.word 8*

*.ident "GCC: (SiFive GCC-Metal 10.2.0-2020.12.8) 10.2.0"*

По метке main начинается тестовая программа. Стоит отметить, что компилятор вывел длины массивов 1 и 2. Это можно понять из директив «.size array2, 28» и «.size array1, 16»

Рассмотрим запись аргументов функции. Сначала в **a0** записывается адрес массива 2 (он идет сразу за .LANCHOR0), потом в **a3** записывается его размер (7). После этого в **a2**, как ни странно, также записывается адрес массива 2, потом в **a1** записывается длина массива 1, равная 4. И уже потом к регистру **a0** (где лежит адрес массива 1) мы прибавляем 28 - это длина массива 2 в байтах (7 слов по 4 байта), то есть по сути мы получили адрес массива 1.



Снова, аргументы идут в таком порядке (a0, a1, a2, a3): **адрес массива 2, длина массива 2, адрес массива 1, длина массива 1.**

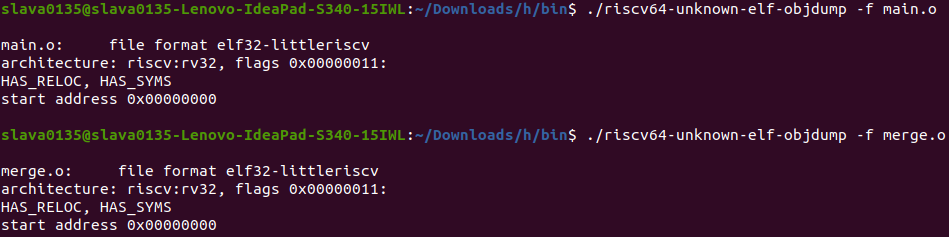
Следующим шагом является ассемблирование файлов “merge.s” и “main.s” в объектные файлы “merge.o” и “main.o”:





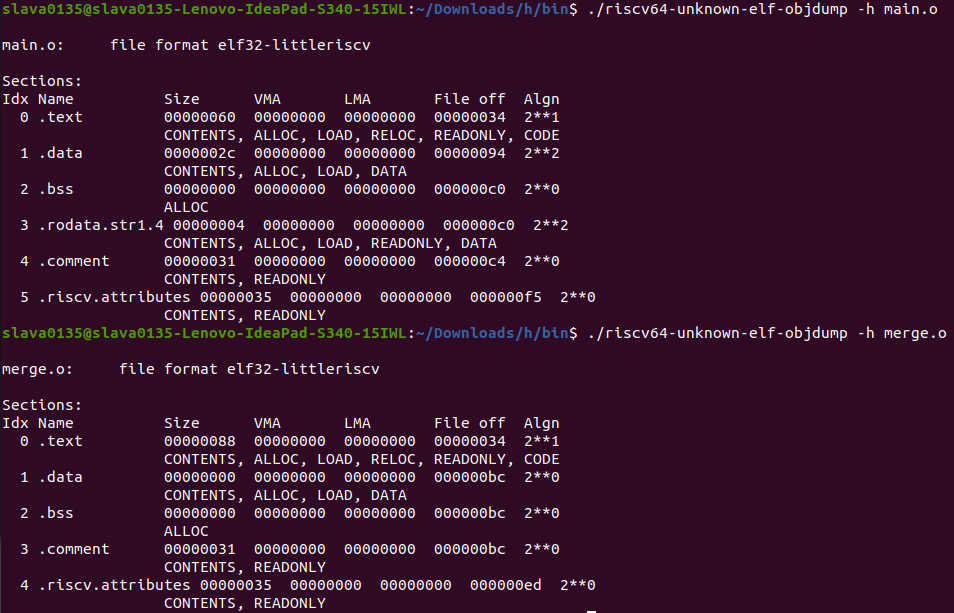
-C – указание остановить процесс сборки после ассемблирования.

Объектный файл не является текстовым, для изучения его содержимого используем утилиту objdump:



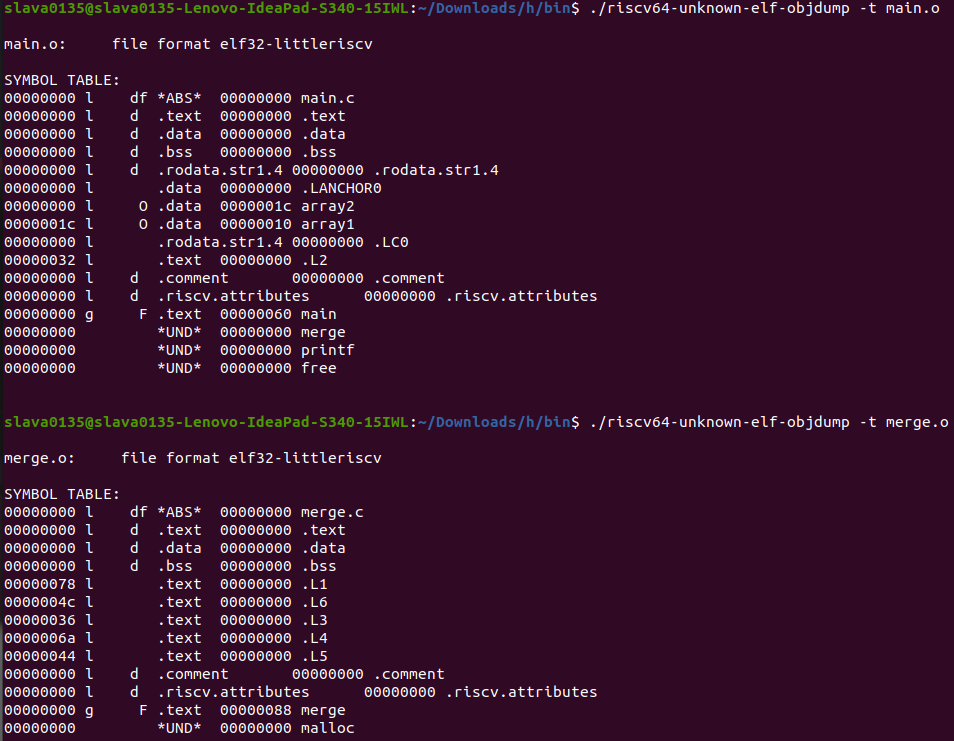
Оба файла содержат таблицу перемещений (значения флагов 00000011, всего два флага HAS\_RELOC, HAS\_SYMS - оба подняты) .

Выведем все заголовки секций объектных файлов:



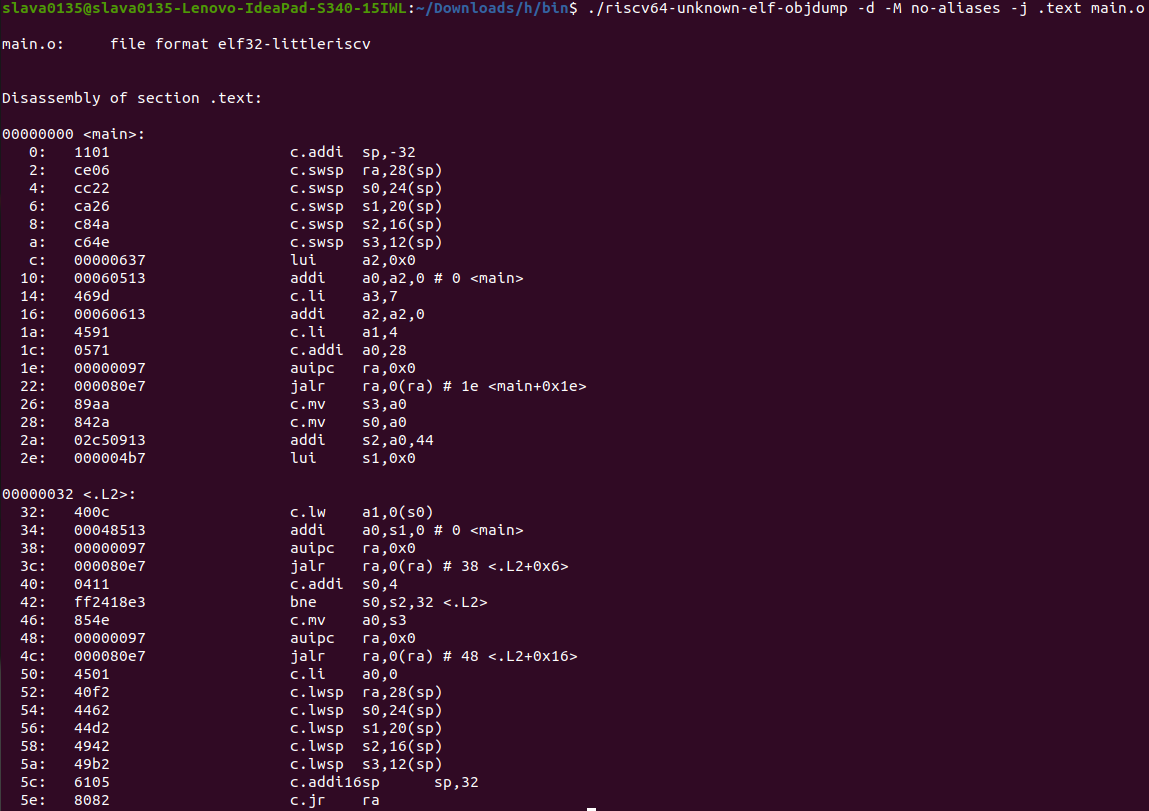
Можно обратить внимание на формат - в RISC-V используется система little-endion. (Сначала идут старшие разряды, потом младшие)

Теперь просмотрим таблицы символов файлов main.o и merge.o

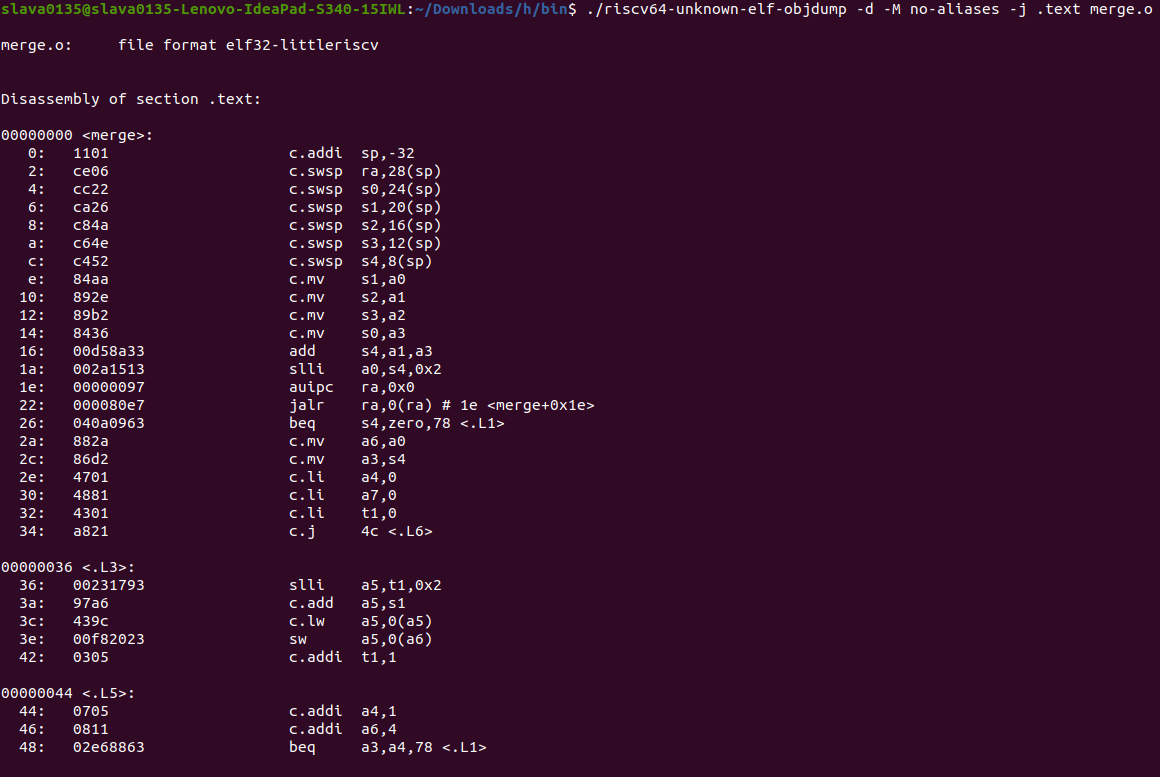


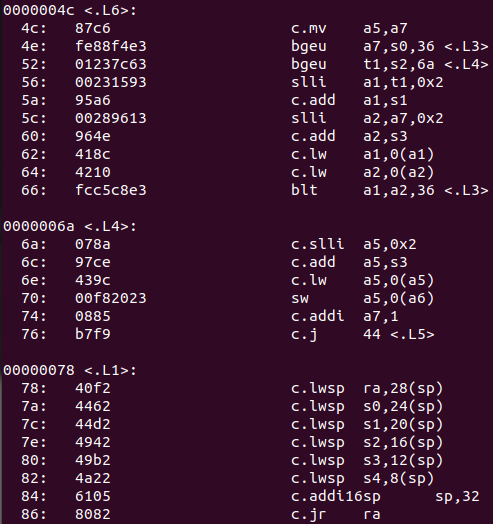
В каждой таблице по одному глобальному символу (флаг “g”) типа функция (“F”) - это функции main() и merge().

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



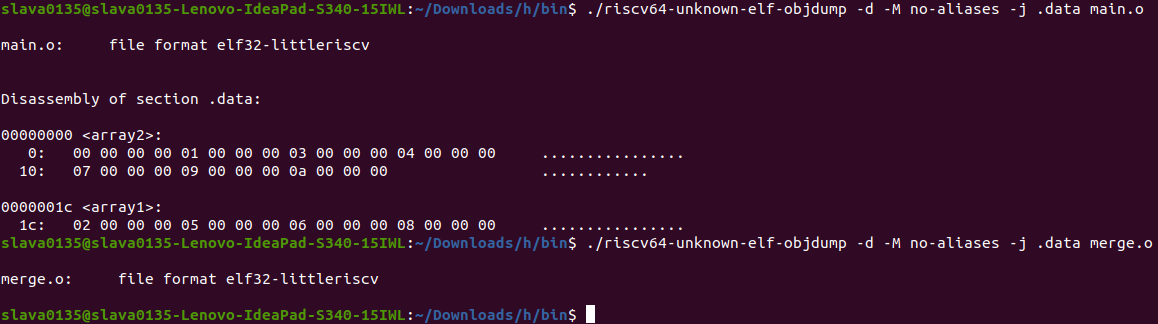
Дизассемблированный код идентичен сгенерированному, за исключением того, что псевдоинструкции были заменены на инструкции системы команд, так как опция “-M no-aliases” требует использовать в выводе только инструкции системы команд (но не псевдоинструкции ассемблера). Например, все псевдоинструкции **call** заменены на связки **auipc + jalr** с комментариями о перемещении.





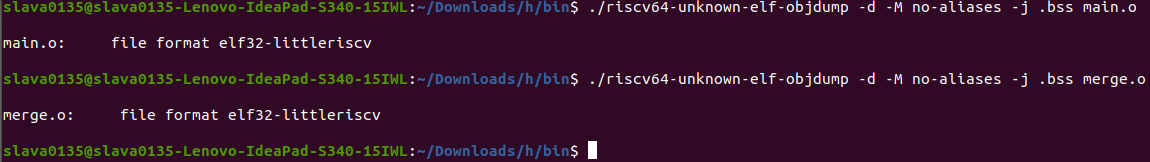
Здесь можно сделать такое же заключение насчет идентичности.

Теперь просмотрим блоки .data.



Как видно в файле main.o в блоке .data определены 2 массива с данными. Можно заметить, что каждый элемент занимает 4 байта. В файле merge.o этот блок пуст.

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



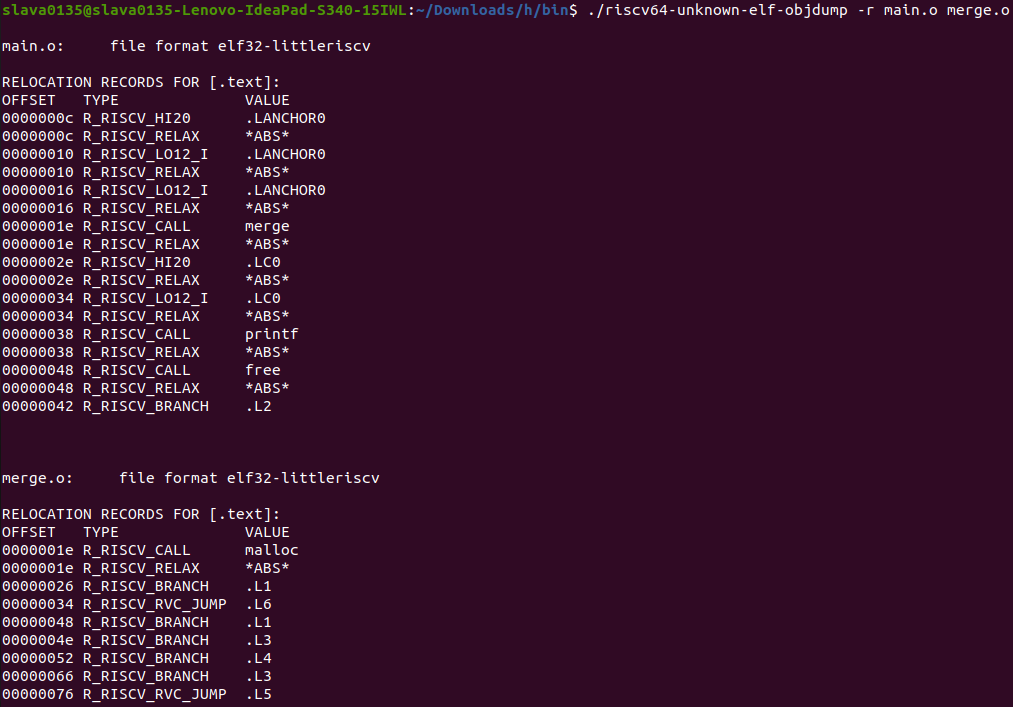
Секция .comment – секция данных о версиях – и для одного и для другого файла содержит одни и те же значения – сведения о GCC версии 10.2.0 от SiFive:



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



Теперь проанализируем таблицу перемещений объектных файлов:

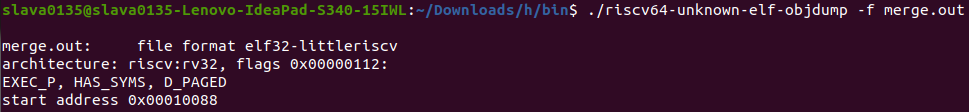


В таблицах перемещения maxPrime.o содержится информация о переходах (R\_RISCV\_JAL) и ветвлениях (R\_RISCV\_BRANCH). В таблицах перемещения main.o, есть R\_RISCV\_CALL, чтобы информация о переходах соответствовала merge. Записи типа R\_RISCV\_RELAX заносятся в таблицу перемещений в дополнение к записям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

Последним шагом является компановка и формирование исполняемых файлов программ:

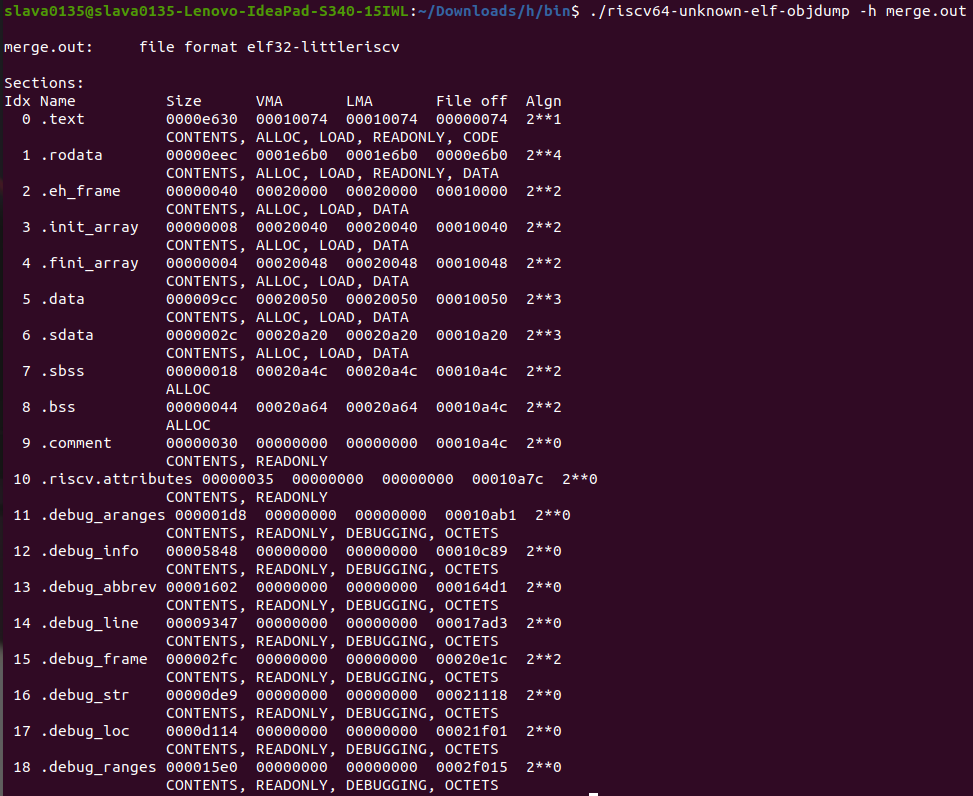


Сформированный компановщиком файл “merge.out” также является бинарным файлом - нужно воспользоваться утилитой objdump.



Флаг EXEC\_P указывает, что файл действительно является исполняемым, после загрузки его выполнение должно начаться с адреса 0x00010088 (entrypoint).

Перечислим секции исполняемого файла:

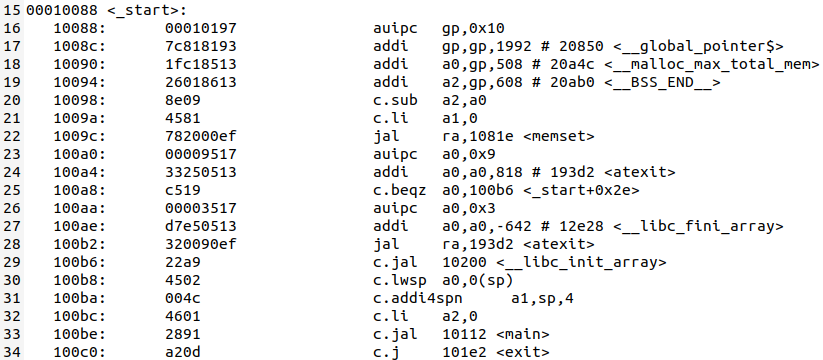


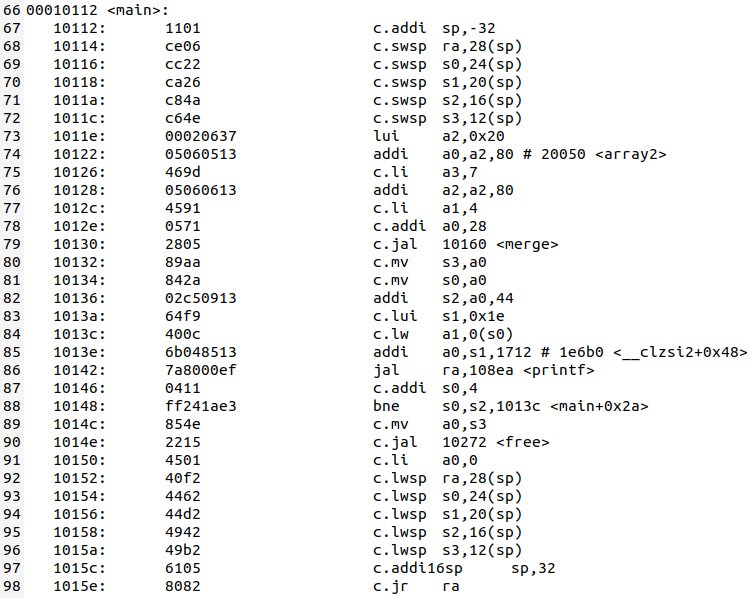
В исполняемом файле действительно производится слияние содержания секций обоих объектных файлов, а также значительное расширение списка секций новыми блоками, например для дебага.

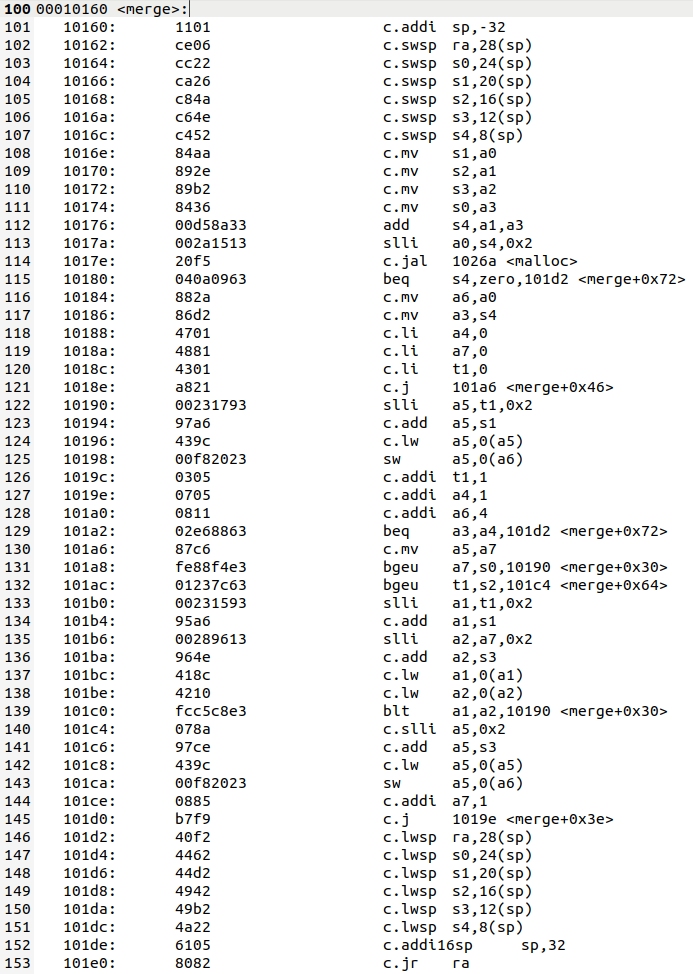
Проанализируем содержимое секции .text исполняемого файла:

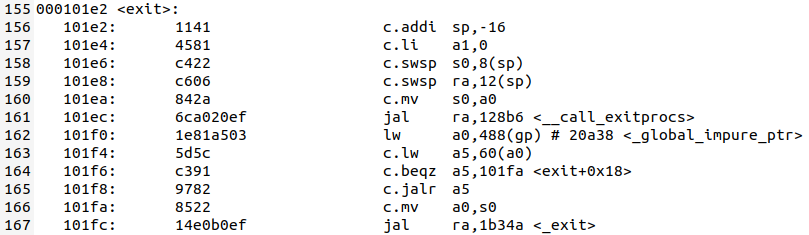


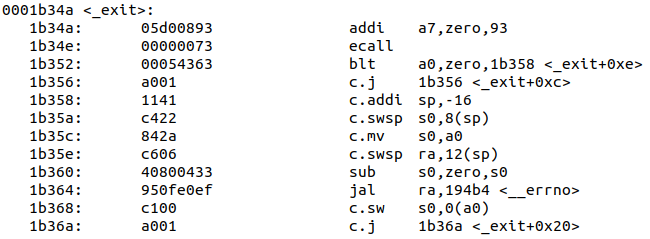
Мне пришлось перенаправить вывод в файл, так как секция кода многократно увеличилась, я приведу только некоторую часть.



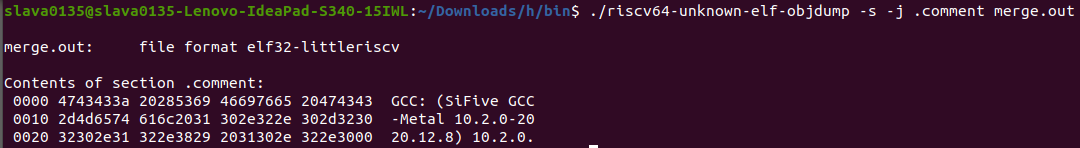








Секция .comment:



Секция .riscv.attributes:

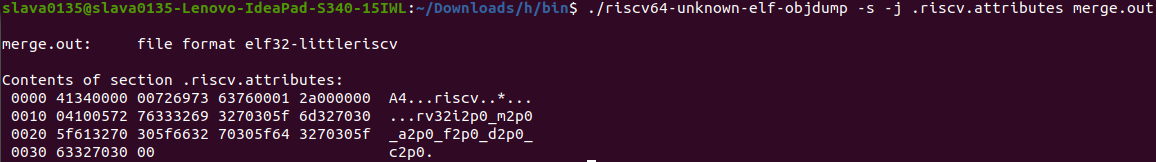


Таблица символов:merge.out: file format elf32-littleriscv

SYMBOL TABLE:

00010074 l d .text 00000000 .text

0001e6b0 l d .rodata 00000000 .rodata

00020000 l d .eh\_frame 00000000 .eh\_frame

00020040 l d .init\_array 00000000 .init\_array

00020048 l d .fini\_array 00000000 .fini\_array

00020050 l d .data 00000000 .data

00020a20 l d .sdata 00000000 .sdata

00020a4c l d .sbss 00000000 .sbss

00020a64 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 00000014 register\_fini

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

00020000 l O .eh\_frame 00000000 \_\_EH\_FRAME\_BEGIN\_\_

000100c2 l F .text 00000000 \_\_do\_global\_dtors\_aux

00020a64 l O .bss 00000001 completed.1

00020048 l O .fini\_array 00000000 \_\_do\_global\_dtors\_aux\_fini\_array\_entry

000100f6 l F .text 00000000 frame\_dummy

00020a68 l O .bss 00000018 object.0

00020044 l O .init\_array 00000000 \_\_frame\_dummy\_init\_array\_entry

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

00020050 l O .data 0000001c array2

0002006c l O .data 00000010 array1

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

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

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

00020080 l O .data 00000428 impure\_data

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

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

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

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

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

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

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

00012730 l F .text 00000092 \_\_sbprintf

0001e864 l O .rodata 00000010 blanks.1

0001e874 l O .rodata 00000010 zeroes.0

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

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

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

00012b86 l F .text 00000004 \_\_fp\_lock

00012b94 l F .text 00000004 \_\_fp\_unlock

00012b98 l F .text 00000134 \_\_sinit.part.0

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

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

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

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

0001322a l F .text 00000048 eshdn1

00013272 l F .text 0000004e eshup1

000132c0 l F .text 00000074 m16m

00013334 l F .text 0000008c emovo.constprop.0

000133c0 l F .text 0000010c enormlz

000134cc l F .text 0000017a eshift.part.0

00013646 l F .text 0000008e emovi

000136d4 l F .text 000000e0 ecmp

000137b4 l F .text 000002ea emdnorm

00013a9e l F .text 000000ee eiremain

00013b8c l F .text 00000376 emul

00013f02 l F .text 00000426 ediv

00014328 l F .text 0000010c e113toe.isra.0

0001e9b8 l O .rodata 00000014 ezero

0001e9cc l O .rodata 00000014 eone

0001e9e0 l O .rodata 00000104 etens

0001eae4 l O .rodata 00000104 emtens

0001ebe8 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 mbtowc\_r.c

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

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

0001ecd8 l O .rodata 0000000c p05.0

00000000 l df \*ABS\* 00000000 reent.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

0001ef6c l O .rodata 00000010 blanks.1

0001ef7c l O .rodata 00000010 zeroes.0

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

000184a4 l F .text 00000098 \_\_sprint\_r.part.0

0001929a l F .text 00000094 \_\_sbprintf

0001f0f8 l O .rodata 00000010 blanks.1

0001f108 l O .rodata 00000010 zeroes.0

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

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

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

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

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

000194ba l F .text 00000080 \_fclose\_r.part.0

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

0001f3c8 l O .rodata 00000010 blanks.1

0001f3d8 l O .rodata 00000010 zeroes.0

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

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

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

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

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

00000000 l df \*ABS\* 00000000 signalr.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\_getpid.c

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

00000000 l df \*ABS\* 00000000 sys\_kill.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

00020a60 l O .sbss 00000004 heap\_end.0

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

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

0002003c l O .eh\_frame 00000000 \_\_FRAME\_END\_\_

00000000 l df \*ABS\* 00000000

0002004c l .fini\_array 00000000 \_\_fini\_array\_end

00020048 l .fini\_array 00000000 \_\_fini\_array\_start

00020048 l .init\_array 00000000 \_\_init\_array\_end

00020040 l .init\_array 00000000 \_\_preinit\_array\_end

00020040 l .init\_array 00000000 \_\_init\_array\_start

00020040 l .init\_array 00000000 \_\_preinit\_array\_start

000160c2 g F .text 0000006e \_mprec\_log10

00016176 g F .text 00000048 \_\_any\_on

00019a86 g F .text 00000030 \_isatty\_r

0001edb0 g O .rodata 00000028 \_\_mprec\_tinytens

000165c8 g F .text 00000078 strcpy

000161be g F .text 00000060 cleanup\_glue

00010160 g F .text 00000082 merge

0001b3a6 g F .text 00000004 \_getpid

00019ab6 g F .text 00000036 \_lseek\_r

0001c5fa g F .text 000000f4 .hidden \_\_getf2

0001b2ea g F .text 00000032 \_kill\_r

0001c554 g F .text 000000a6 .hidden \_\_eqtf2

000108ea g F .text 0000002a printf

00020850 g \*ABS\* 00000000 \_\_global\_pointer$

0001ae8e g F .text 00000044 \_wcrtomb\_r

0001afd8 g F .text 00000064 \_signal\_r

00016460 g F .text 00000042 \_\_sseek

00012dfc g F .text 00000008 \_\_sinit

0001ad8c g F .text 000000f8 \_\_swbuf\_r

00015292 g F .text 00000058 \_setlocale\_r

00012ccc g F .text 00000048 \_\_sfmoreglue

000108c8 g F .text 00000002 \_\_malloc\_unlock

0001e20c g F .text 000000d0 .hidden \_\_floatsitf

00019bd4 g F .text 000000d0 memmove

00012dee g F .text 0000000e \_cleanup

000155b4 g F .text 00000066 \_Balloc

000194b4 g F .text 00000006 \_\_errno

0001b4b6 g F .text 0000007c \_conv\_stat

00015280 g F .text 00000006 \_\_localeconv\_l

00019730 g F .text 00000034 \_fstat\_r

00020a5c g O .sbss 00000004 errno

00016408 g F .text 00000004 \_\_seofread

00020a20 g .sdata 00000000 \_\_SDATA\_BEGIN\_\_

00019aec g F .text 000000e8 memcpy

00012b8a g F .text 0000000a \_cleanup\_r

0001671c g F .text 00001d88 \_svfprintf\_r

00016066 g F .text 0000005c \_\_ratio

0001026a g F .text 00000008 malloc

000108ca g F .text 00000020 \_printf\_r

0001c6ee g F .text 000000f4 .hidden \_\_letf2

0001b03c g F .text 0000007a \_raise\_r

0001b36c g F .text 0000003a \_fstat

0001938a g F .text 0000003e \_\_assert\_func

0001ece8 g O .rodata 000000c8 \_\_mprec\_tens

00020a54 g O .sbss 00000004 \_\_malloc\_top\_pad

0001b31c g F .text 00000002 \_getpid\_r

00020a3c g O .sdata 00000000 .hidden \_\_dso\_handle

00015286 g F .text 00000006 \_localeconv\_r

00019764 g F .text 00000322 \_\_sfvwrite\_r

000158a6 g F .text 0000006c \_\_i2b

0001b0b6 g F .text 00000088 \_\_sigtramp\_r

00010914 g F .text 00000030 \_sbrk\_r

00019ca4 g F .text 00000036 \_read\_r

0001953a g F .text 0000003e \_fclose\_r

00012b3c g F .text 0000004a fflush

00020a50 g O .sbss 00000004 \_\_malloc\_max\_sbrked\_mem

00015eb0 g F .text 000000c8 \_\_b2d

0001b892 g F .text 00000340 .hidden \_\_umoddi3

0001b3aa g F .text 00000024 \_isatty

00020a38 g O .sdata 00000004 \_global\_impure\_ptr

00019cda g F .text 000003e6 \_realloc\_r

00010200 g F .text 0000006a \_\_libc\_init\_array

0001b532 g F .text 00000360 .hidden \_\_udivdi3

000196c8 g F .text 0000001e \_fputwc\_r

0001edd8 g O .rodata 00000028 \_\_mprec\_bigtens

00015712 g F .text 000000c2 \_\_s2b

0001af24 g F .text 0000000c abort

0001b43a g F .text 00000050 \_sbrk

00015c94 g F .text 00000040 \_\_mcmp

00012e0c g F .text 0000000e \_\_fp\_lock\_all

00012e28 g F .text 00000038 \_\_libc\_fini\_array

0001621e g F .text 000000ae \_reclaim\_reent

000157d4 g F .text 00000048 \_\_hi0bits

0001e150 g F .text 000000bc .hidden \_\_fixtfsi

00015aae g F .text 000000d6 \_\_pow5mult

0001e668 g F .text 0000003c .hidden \_\_clzsi2

00012e04 g F .text 00000002 \_\_sfp\_lock\_acquire

00015520 g F .text 00000094 memchr

00016348 g F .text 00000040 \_sprintf\_r

00012f2a g F .text 00000220 \_free\_r

000152ea g F .text 00000006 \_\_locale\_mb\_cur\_max

000128b6 g F .text 000000be \_\_call\_exitprocs

00020a44 g O .sdata 00000004 \_\_malloc\_sbrk\_base

00010088 g F .text 0000003a \_start

0001b3e2 g F .text 0000002c \_lseek

00015b84 g F .text 00000110 \_\_lshift

0001c6ee g F .text 000000f4 .hidden \_\_lttf2

0001b1b8 g F .text 0000006c signal

0001c554 g F .text 000000a6 .hidden \_\_netf2

0001a0c0 g F .text 0000012c \_\_ssprint\_r

0001af30 g F .text 0000006c \_\_register\_exitproc

00015236 g F .text 0000004a \_ldcheck

00015912 g F .text 0000019c \_\_multiply

000166a2 g F .text 0000007a strncpy

00020a80 g O .bss 00000028 \_\_malloc\_current\_mallinfo

00015f78 g F .text 000000ee \_\_d2b

0001e2dc g F .text 00000150 .hidden \_\_extenddftf2

00019484 g F .text 00000030 \_close\_r

000127c2 g F .text 000000f4 \_\_swsetup\_r

0001bbd2 g F .text 00000540 .hidden \_\_divdf3

00012d14 g F .text 000000da \_\_sfp

00016130 g F .text 00000046 \_\_copybits

00020ab0 g .bss 00000000 \_\_BSS\_END\_\_

000204a8 g O .data 00000408 \_\_malloc\_av\_

00012e0a g F .text 00000002 \_\_sinit\_lock\_release

0001c112 g F .text 00000442 .hidden \_\_muldf3

000163d0 g F .text 00000038 \_\_sread

000108c6 g F .text 00000002 \_\_malloc\_lock

00012b08 g F .text 00000034 \_fflush\_r

000193de g F .text 000000a6 \_calloc\_r

00020a4c g .sbss 00000000 \_\_bss\_start

0001081e g F .text 000000a8 memset

00010112 g F .text 0000004e main

00020a4c g O .sbss 00000004 \_\_malloc\_max\_total\_mem

0001ae84 g F .text 0000000a \_\_swbuf

000164a2 g F .text 00000008 \_\_sclose

00019578 g F .text 00000042 fclose

0001027c g F .text 000005a2 \_malloc\_r

00019334 g F .text 00000020 \_\_ascii\_wctomb

000195ba g F .text 00000020 \_fiprintf\_r

0001b224 g F .text 0000003e \_init\_signal

0001314a g F .text 0000006c \_fwalk

000154da g F .text 00000006 \_mbtowc\_r

00012e60 g F .text 000000ca \_malloc\_trim\_r

000164aa g F .text 0000011e strcmp

0001928c g F .text 0000000e vfiprintf

0001c7e2 g F .text 00000b10 .hidden \_\_multf3

00016388 g F .text 00000048 sprintf

0001f49c g O .rodata 00000100 .hidden \_\_clz\_tab

00020a58 g O .sbss 00000004 \_PathLocale

000193d2 g F .text 0000000c atexit

00019354 g F .text 00000036 \_write\_r

000152f0 g F .text 00000058 setlocale

00020a40 g O .sdata 00000004 \_impure\_ptr

00012974 g F .text 00000194 \_\_sflush\_r

0001c5fa g F .text 000000f4 .hidden \_\_gttf2

0001a1ec g F .text 00000ba0 \_svfiprintf\_r

000154e0 g F .text 00000040 \_\_ascii\_mbtowc

0001d2f2 g F .text 00000e5e .hidden \_\_subtf3

00015e6a g F .text 00000046 \_\_ulp

00012e1a g F .text 0000000e \_\_fp\_unlock\_all

0001528c g F .text 00000006 localeconv

00015458 g F .text 00000082 \_\_swhatbuf\_r

00020050 g .data 00000000 \_\_DATA\_BEGIN\_\_

0001b48a g F .text 0000002c \_write

00020a4c g .sdata 00000000 \_edata

00020ab0 g .bss 00000000 \_end

00019604 g F .text 000000c4 \_\_fputwc

0001640c g F .text 00000054 \_\_swrite

00020a48 g O .sdata 00000004 \_\_malloc\_trim\_threshold

000101e2 g F .text 0000001e exit

000195da g F .text 0000002a fiprintf

0001854a g F .text 00000d42 \_vfiprintf\_r

000131b6 g F .text 00000074 \_fwalk\_reent

00015cd4 g F .text 00000196 \_\_mdiff

0001b3ce g F .text 00000014 \_kill

00012e06 g F .text 00000002 \_\_sfp\_lock\_release

00014434 g F .text 00000e02 \_ldtoa\_r

0001f158 g O .rodata 00000101 \_ctype\_

0001b40e g F .text 0000002c \_read

0001b34a g F .text 00000022 \_exit

00015348 g F .text 00000110 \_\_smakebuf\_r

00016640 g F .text 00000062 strlen

0001af9c g F .text 0000003c \_init\_signal\_r

000193c8 g F .text 0000000a \_\_assert

0001853c g F .text 0000000e \_\_sprint\_r

0001932e g F .text 00000006 \_wctomb\_r

00010944 g F .text 00001dde \_vfprintf\_r

0001581c g F .text 0000008a \_\_lo0bits

0001b262 g F .text 00000088 \_\_sigtramp

0001aed2 g F .text 00000052 wcrtomb

000162cc g F .text 0000007c frexp

000208b0 g O .data 0000016c \_\_global\_locale

00012722 g F .text 0000000e vfprintf

0001e42c g F .text 0000023c .hidden \_\_trunctfdf2

000196e6 g F .text 0000004a fputwc

0001b13e g F .text 0000007a raise

0001b31e g F .text 0000002c \_close

00010272 g F .text 0000000a free

00012e08 g F .text 00000002 \_\_sinit\_lock\_acquire

0001562c g F .text 000000e6 \_\_multadd

0001561a g F .text 00000012 \_Bfree

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

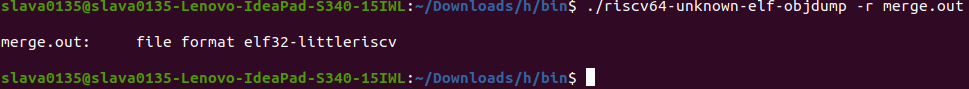


Таблица перемещений оказывается пуста, все необходимые релокации, оптимизации и замены инструкций были успешно проведены компоновщиком.

Итогом сборки программ на языке C по шагам является исполняемый на процессорах архитектуры RISC-V файл, решающий задачу cлияния двух массивов.

## **Формирование статической библиотеки, разработка make-файлов для сборки библиотеки.**

Статическая библиотека (static library) является, по сути, архивом (набором, коллекцией) 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 слияния двух массивов содержится в единственном исходном файле на языке C **merge.c** . Выделим этот файл в статическую библиотеку:

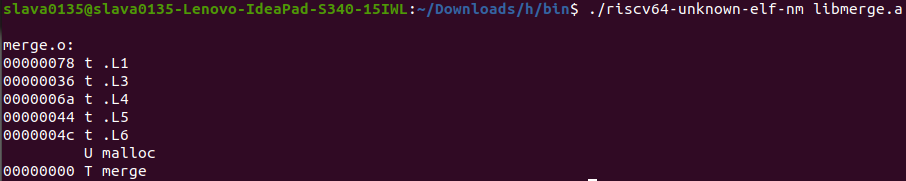


Просмотрим, какие файлы туда вошли:



Туда вошел только объектный файл с нашей функцией.

Просмотрим список символов библиотеки:



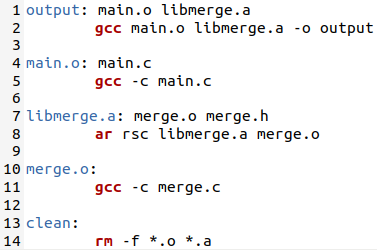
В выводе утилиты “nm” кодом “T” обозначаются символы, определенные в соответствующем объектном файле, кодом “U” - внешние символы.

Единственный внешний символ - **malloc** - библиотека для выделения памяти под массивы. Символ функции **merge** является основным символом, определяемым в этом объектном файле.

Теперь осуществим сборку тестовой программы используя эту библиотеку:



Теперь напишем makefile для сборки программы:



И осуществим сборку программы через gcc:



* ls => список файлов до сборки.
* make -f makefile => осуществляем сборку
* ls => список файлов после сборки
* ./output => запускаем программу, получаем ответ: 0 1 2 3 4 5 6 7 8 9 10 (здесь следовало бы еще переходить на новую строку, иначе получится как получилось)
* make clean => очищаем папку
* ls => остались те же файлы, что были и до сборки

# Выводы

В ходе выполнения лабораторной работы были закреплены знания языка C, ассемблера RISC-V, получены навыки работы с препроцессором, компилятором, ассемблером и компоновщиком пакета GCC и драйвером компилятора riscv64-unknown-elf-gcc. Были изучены особенности каждого этапа пошаговой сборки набора программ, а также инструменты, позволяющие выделить разработанные программы в статическую библиотеку и автоматизировать сборку этой библиотеки.

Была реализована поставленная задача – слияние двух отсортированных массивов, а затем проведена проверка правильности перевода программы решения этой задачи в набор инструкций, выполняемых процессором.