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](#_Toc967598031)

[2. МЕТОД РЕШЕНИЯ 4](#_Toc593021896)

[2.1. Текст программ и описание алгоритма 4](#_Toc245232414)

[2.2. Сборка программ по шагам, анализ промежуточных и результирующих файлов 6](#_Toc636232775)

[2.3. Формирование статической библиотеки, разработка make-файлов для сборки библиотеки. 32](#_Toc252232449)

[3. Выводы 35](#_Toc1666378896)

# 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.

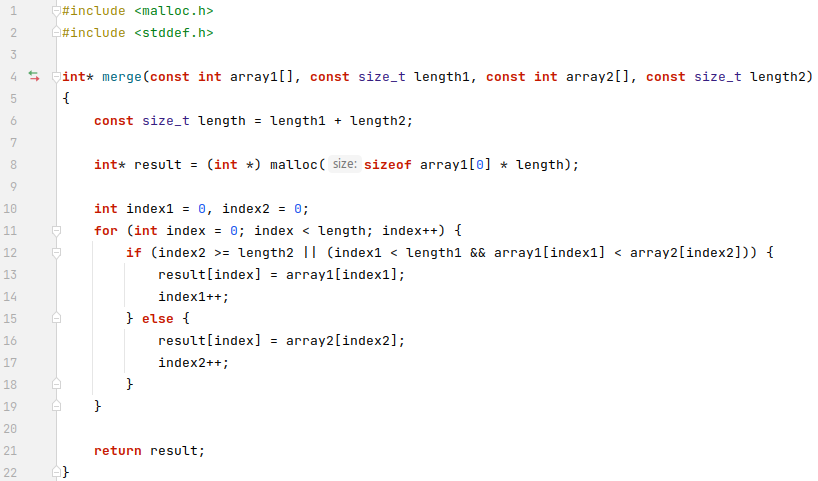
3) Выделить разработанную функцию в статическую библиотеку. Разработать make-файлы для сборки библиотеки и использующей ее тестовой программы. Проанализировать ход сборки библиотеки и программы, созданные файлы зависимостей.

**ВАРИАНТ:** Слияние двух отсортированных массивов.

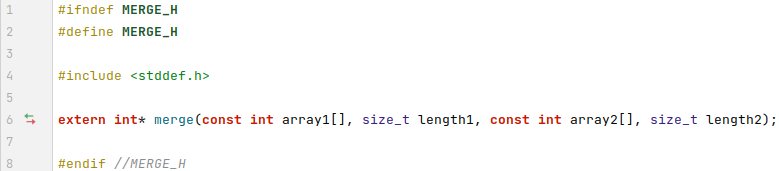
# МЕТОД РЕШЕНИЯ

## Текст программ и описание алгоритма

Листинг 1. Программа “merge.c”

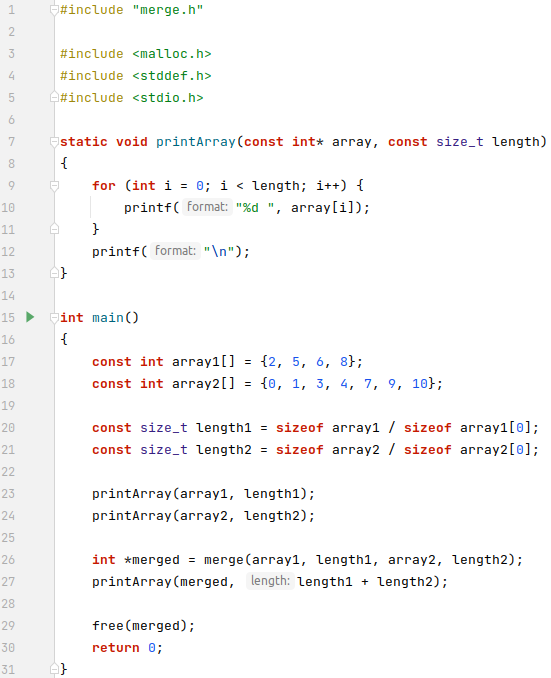


Листинг 2. Программа “merge.h”



./riscv64-unknown-elf-gcc --save-temps -march=rv32iafdc -mabi=ilp32 -O1 -v main.c merge.c >log 2>&1

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)*

*{*

*const size\_t length = length1 + length2;*

*int\* result = (int \*) malloc(sizeof array1[0] \* length);*

*int index1 = 0, index2 = 0;*

*for (int index = 0; index < length; 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*

*static void printArray(const int\* array, const size\_t length)*

*{*

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

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

*}*

*printf("\n");*

*}*

*int main()*

*{*

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

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

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

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

*printArray(array1, length1);*

*printArray(array2, length2);*

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

*printArray(merged, length1 + length2);*

*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\_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)*

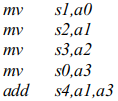
*addi sp,sp,32*

*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.



По этим строчкам можно понять, что в a1 и a3 лежат длины массивов (мысчитаем длину выходного массива через add s4,a1,a3), тогда в a0 и a2 лежат массивы.

Листинг 7. Файл “main.s”

*.file "main.c"*

*.option nopic*

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

*.attribute unaligned\_access, 0*

*.attribute stack\_align, 16*

*.text*

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

*.align 2*

*.LC2:*

*.string "%d "*

*.text*

*.align 1*

*.type printArray, @function*

*printArray:*

*addi sp,sp,-32*

*sw ra,28(sp)*

*sw s0,24(sp)*

*sw s1,20(sp)*

*sw s2,16(sp)*

*sw s3,12(sp)*

*beq a1,zero,.L2*

*mv s2,a1*

*mv s0,a0*

*li s1,0*

*lui s3,%hi(.LC2)*

*.L3:*

*lw a1,0(s0)*

*addi a0,s3,%lo(.LC2)*

*call printf*

*addi s1,s1,1*

*addi s0,s0,4*

*bne s1,s2,.L3*

*.L2:*

*li a0,10*

*call putchar*

*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 printArray, .-printArray*

*.align 1*

*.globl main*

*.type main, @function*

*main:*

*addi sp,sp,-64*

*sw ra,60(sp)*

*sw s0,56(sp)*

*lui a5,%hi(.LANCHOR0)*

*addi a5,a5,%lo(.LANCHOR0)*

*lw a1,0(a5)*

*lw a2,4(a5)*

*lw a3,8(a5)*

*lw a4,12(a5)*

*sw a1,32(sp)*

*sw a2,36(sp)*

*sw a3,40(sp)*

*sw a4,44(sp)*

*lw a6,16(a5)*

*lw a0,20(a5)*

*lw a1,24(a5)*

*lw a2,28(a5)*

*lw a3,32(a5)*

*lw a4,36(a5)*

*lw a5,40(a5)*

*sw a6,4(sp)*

*sw a0,8(sp)*

*sw a1,12(sp)*

*sw a2,16(sp)*

*sw a3,20(sp)*

*sw a4,24(sp)*

*sw a5,28(sp)*

*li a1,4*

*addi a0,sp,32*

*call printArray*

*li a1,7*

*addi a0,sp,4*

*call printArray*

*li a3,7*

*addi a2,sp,4*

*li a1,4*

*addi a0,sp,32*

*call merge*

*mv s0,a0*

*li a1,11*

*call printArray*

*mv a0,s0*

*call free*

*li a0,0*

*lw ra,60(sp)*

*lw s0,56(sp)*

*addi sp,sp,64*

*jr ra*

*.size main, .-main*

*.section .rodata*

*.align 2*

*.set .LANCHOR0,. + 0*

*.LC0:*

*.word 2*

*.word 5*

*.word 6*

*.word 8*

*.LC1:*

*.word 0*

*.word 1*

*.word 3*

*.word 4*

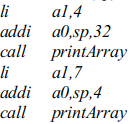
*.word 7*

*.word 9*

*.word 10*

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

По метке main начинается тестовая программа. Функция printArray была размещена по метке printArray здесь же. Массивы размещены по меткам .LC0 и .LC1 в конце файла.



Отсюда можно понять, что компилятор знает о длине массивов - мы передаем их функции printArray.

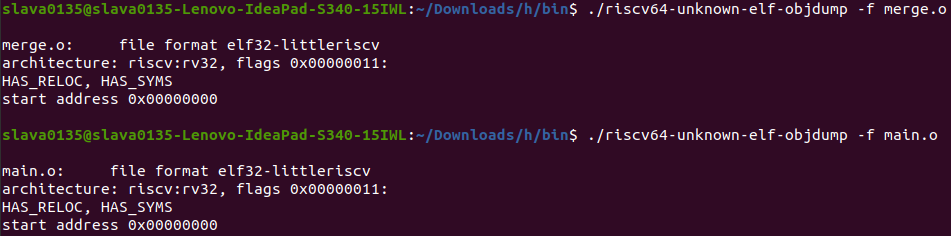
Следующим шагом является ассемблирование файлов “merge.s” и “main.s” в объектные файлы “merge.o” и “main.o”:





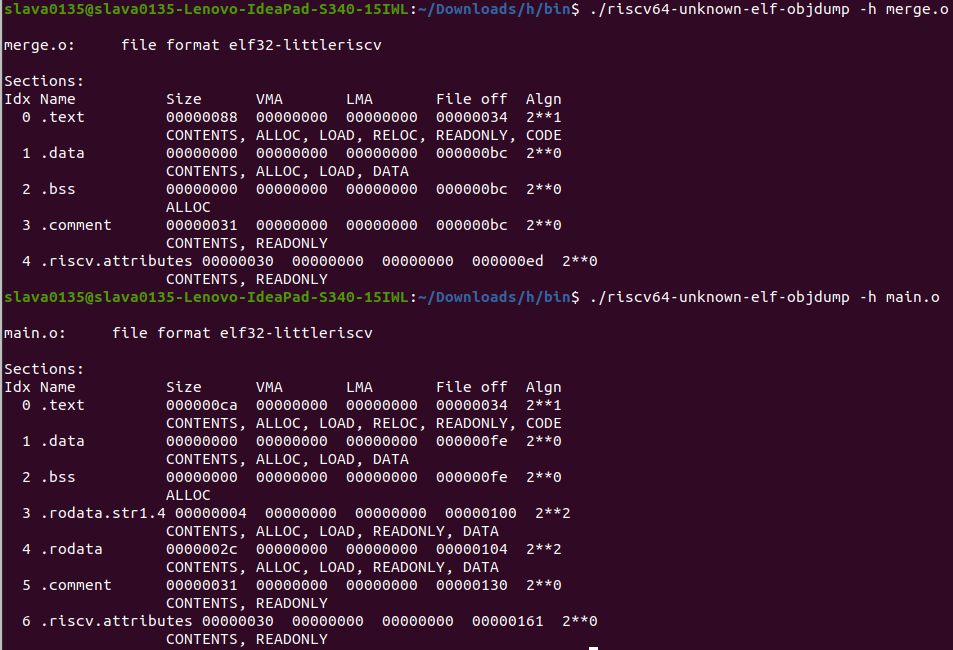
-C – указание остановить процесс сборки после ассемблирования.

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



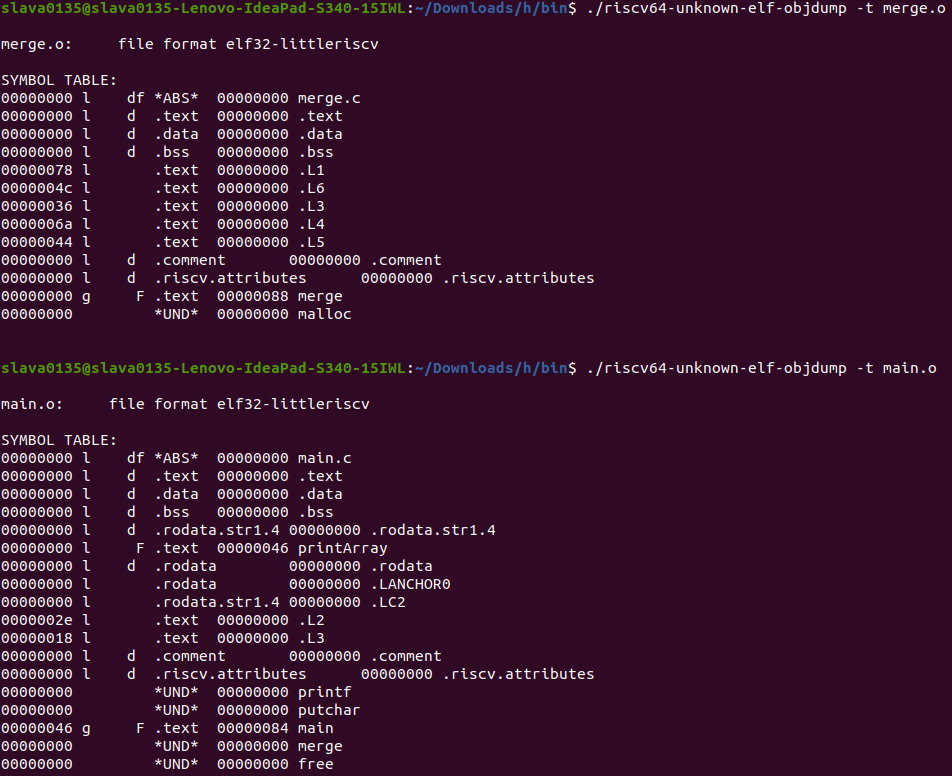
Оба файла содержат таблицу перемещений (значения флагов 00000011, всего два флага HAS\_RELOC, HAS\_SYMS - оба имеют значение 1) .

Выведем все заголовки секций объектных файлов:



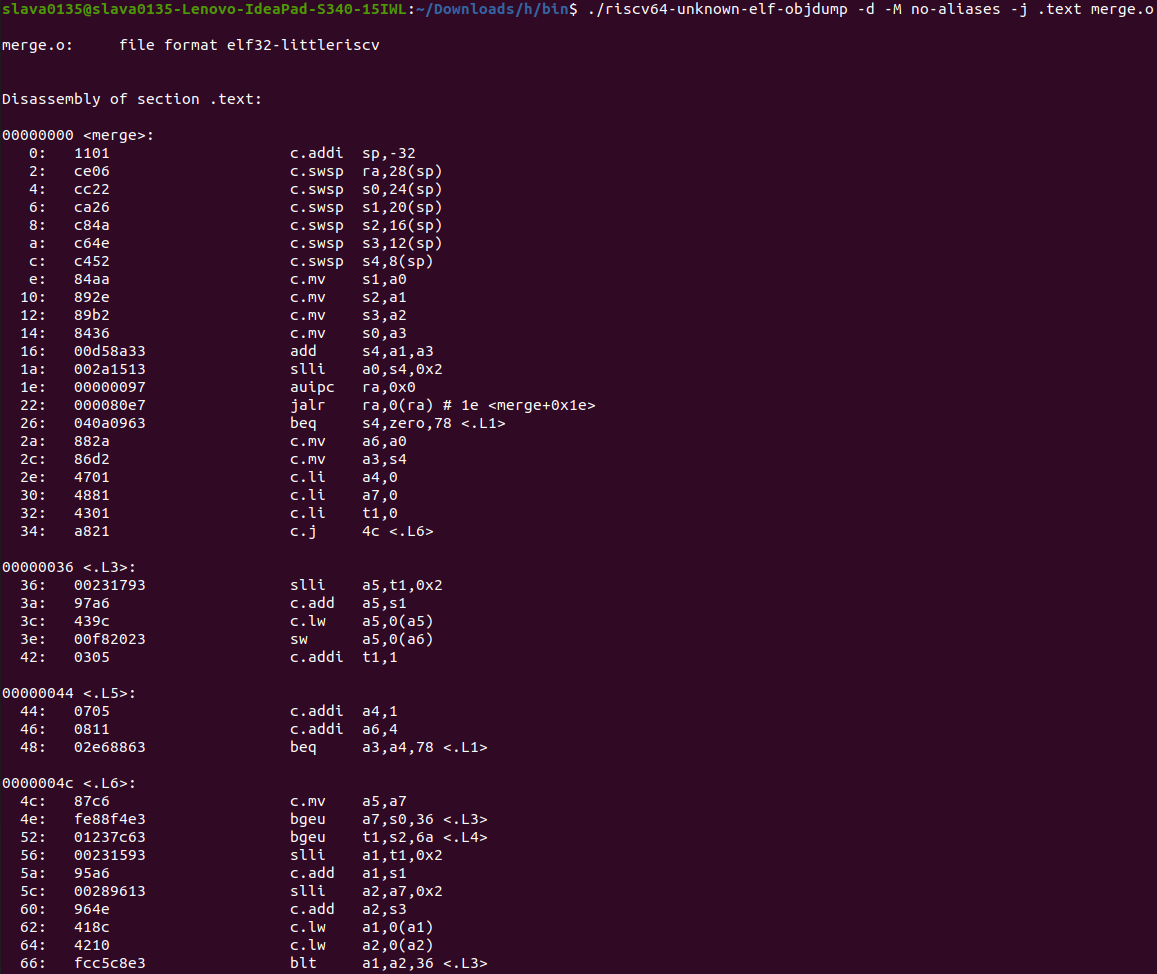
Можно обратить внимание на формат - в RISC-V используется система little-endion. (Сначала идут старшие разряды, потом младшие)

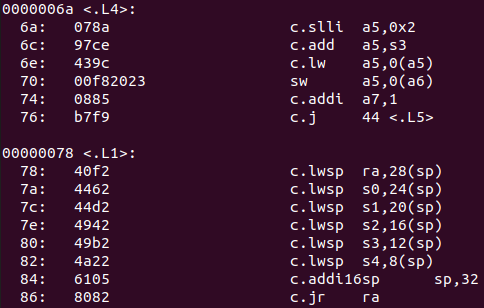
Теперь просмотрим таблицы символов файлов main.o и merge.o



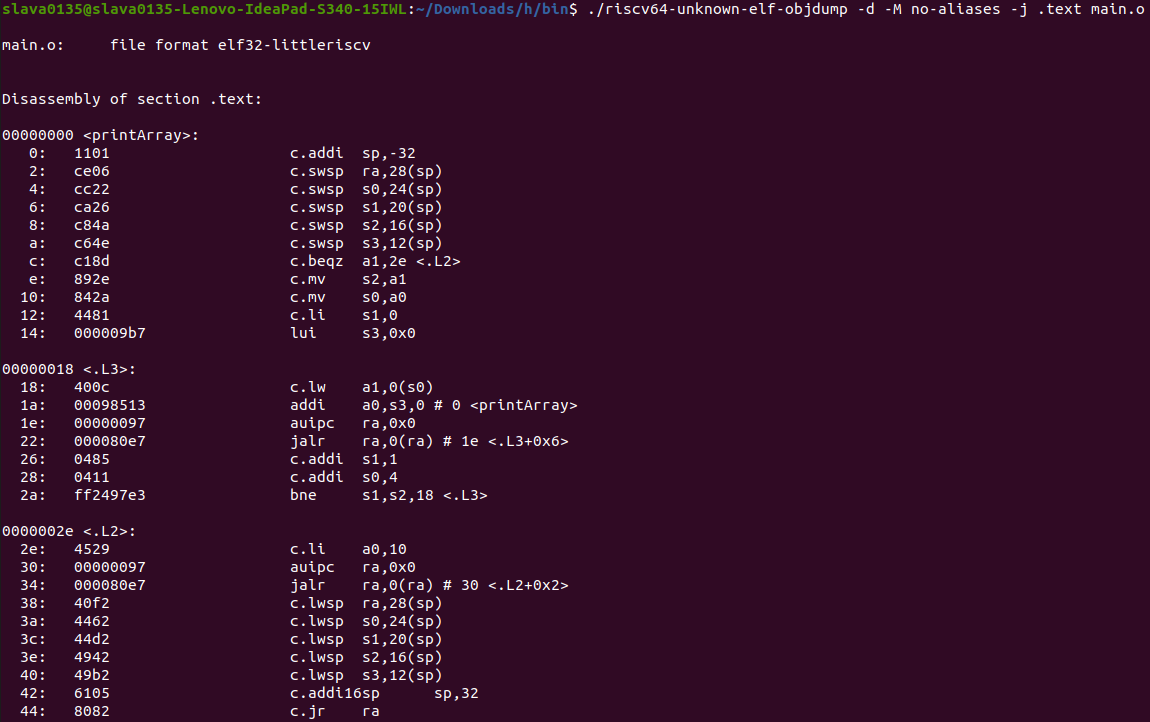
В каждой таблице по одному глобальному символу (флаг “g”) типа функция (“F”) - это функции main() и merge(). А вот функция printArray является локальным символом (флаг “l”), но тоже отмечена как “F”.

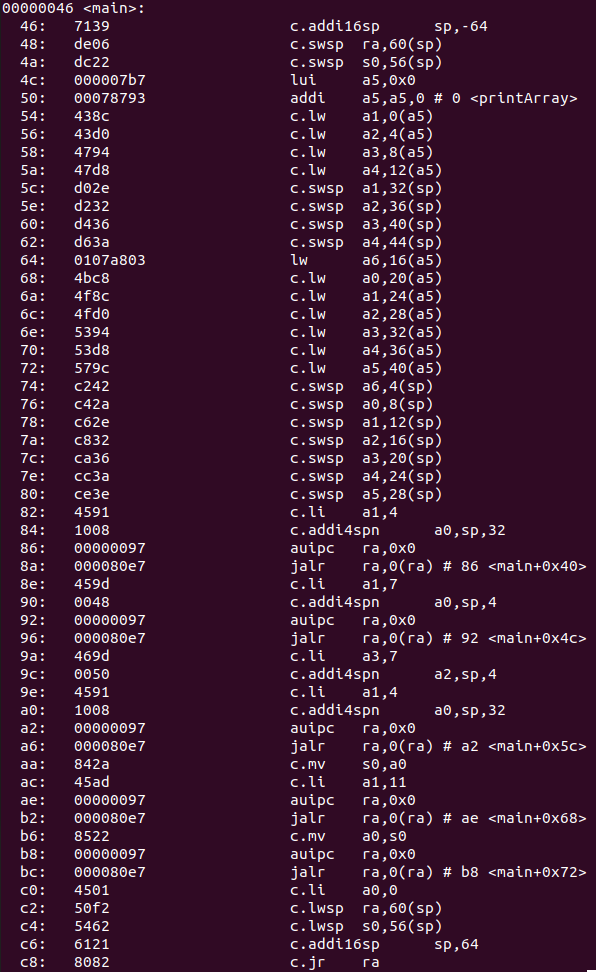
Проанализируем секции .text объектных файлов – секций кода, в которых содержатся коды инструкций:





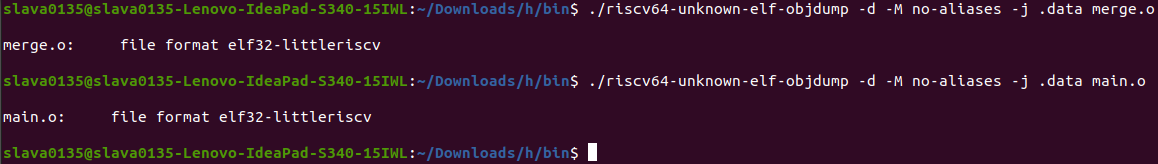
Дизассемблированный код идентичен сгенерированному, за исключением того, что псевдоинструкции были заменены на инструкции системы команд, так как опция “-M no-aliases” требует использовать в выводе только инструкции системы команд (но не псевдоинструкции ассемблера). Например, все псевдоинструкции **call**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





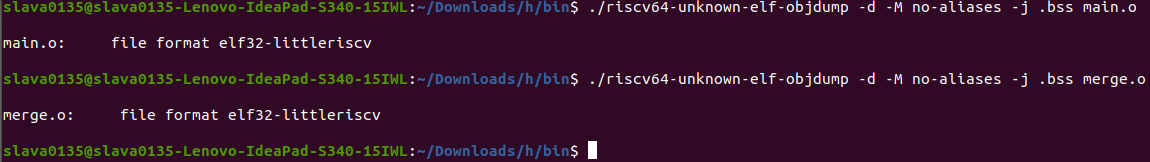
Здесь можно сделать такое же заключение насчет идентичности.

Теперь просмотрим блоки .data.



Эти блоки пусты в обоих файлах.

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



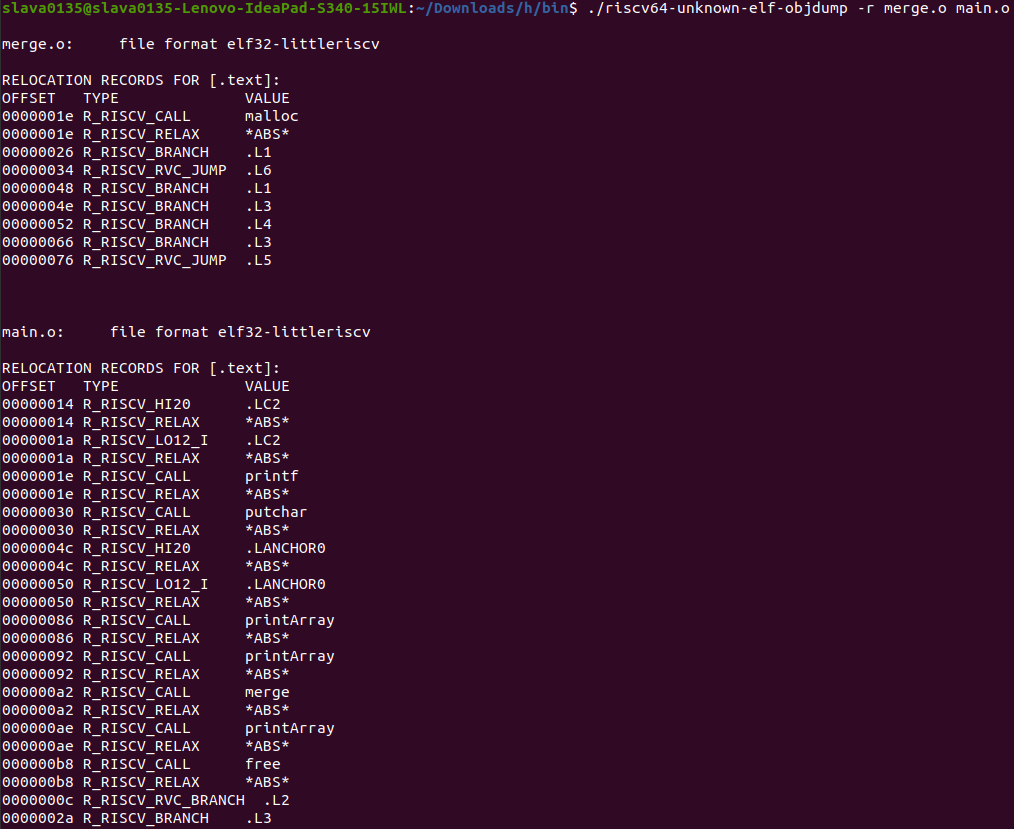
Секция .comment – секция данных о версиях – и для одного и для другого файла содержит одни и те же значения – сведения о GCC версии 10.2.0 от SiFive:



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



Теперь проанализируем таблицу перемещений объектных файлов:

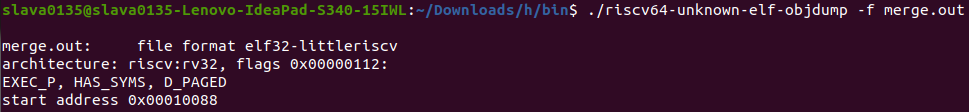


В таблицах перемещения maxPrime.o содержится информация о переходах (R\_RISCV\_JAL) и ветвлениях (R\_RISCV\_BRANCH). В таблицах перемещения main.o, есть R\_RISCV\_CALL, чтобы информация о переходах соответствовала merge. Записи типа R\_RISCV\_RELAX заносятся в таблицу перемещений в дополнение к записям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

Последним шагом является компановка и формирование исполняемых файлов программ:

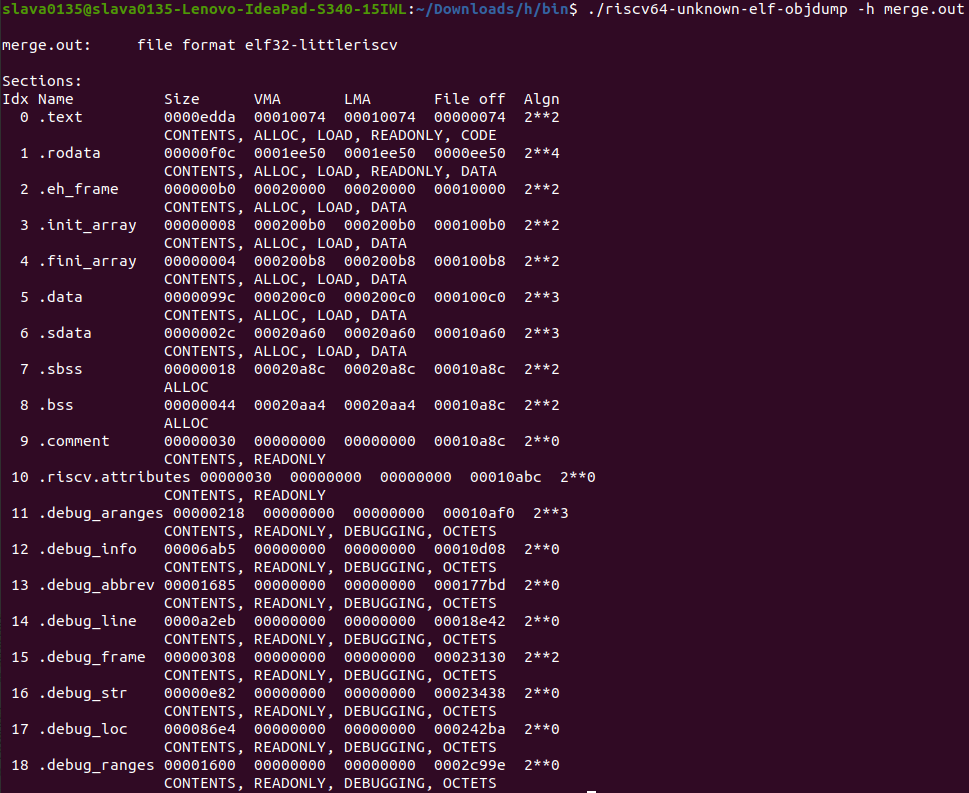


Сформированный компановщиком файл “merge.out” также является бинарным файлом - нужно воспользоваться утилитой objdump.



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

Перечислим секции исполняемого файла:

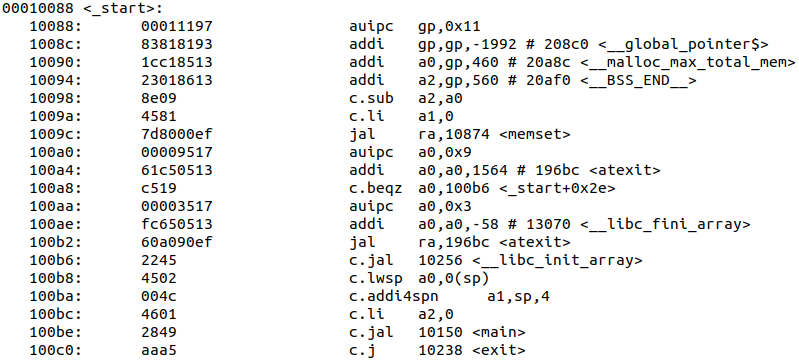


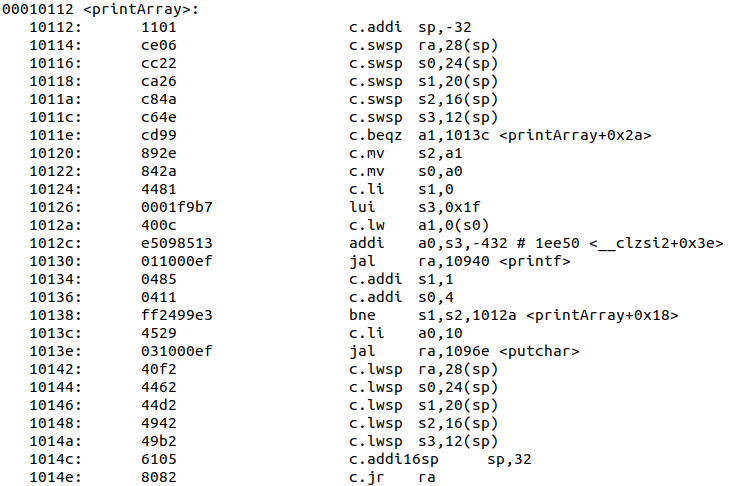
В исполняемом файле действительно производится слияние содержания секций обоих объектных файлов, а также значительное расширение списка секций новыми блоками, например для дебага.

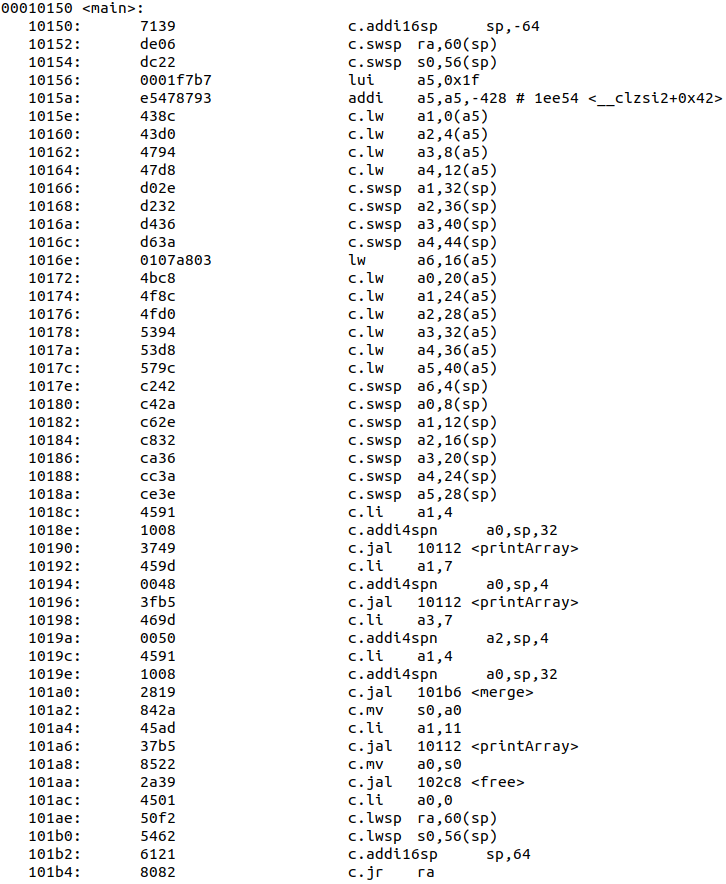
Проанализируем содержимое секции .text исполняемого файла:

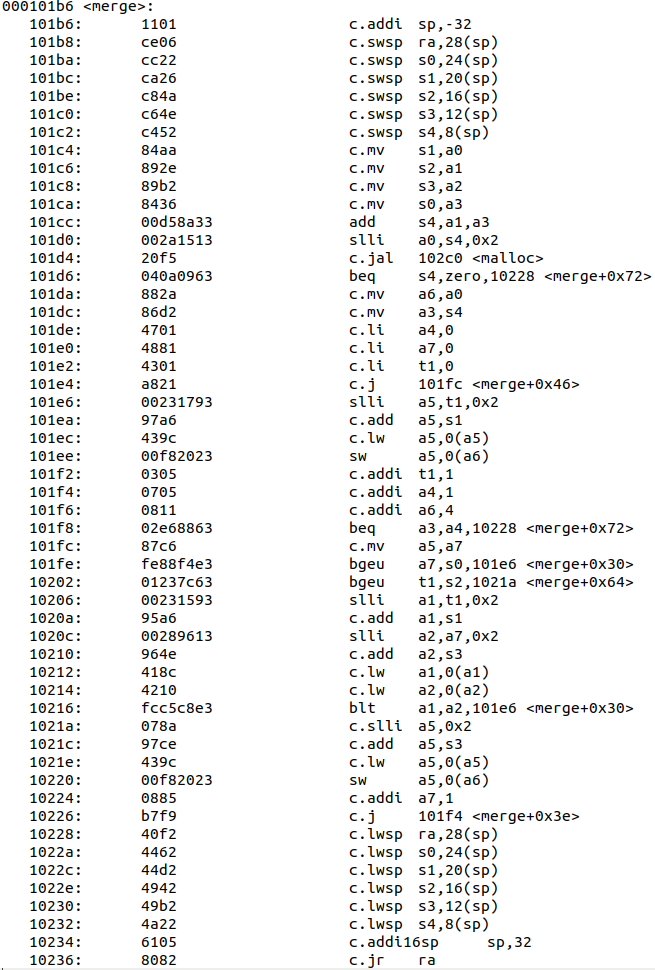


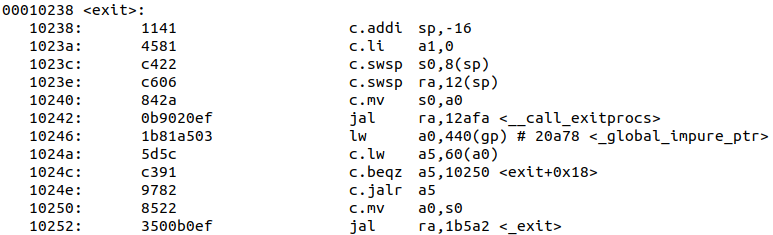
Мне пришлось перенаправить вывод в файл, так как секция кода многократно увеличилась, я приведу только некоторую часть.

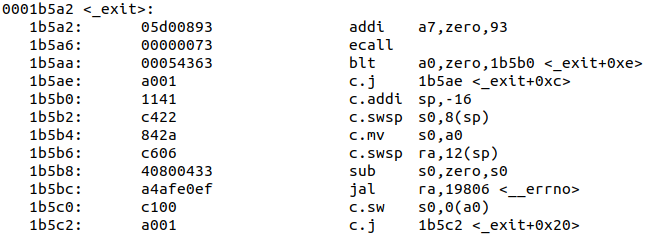




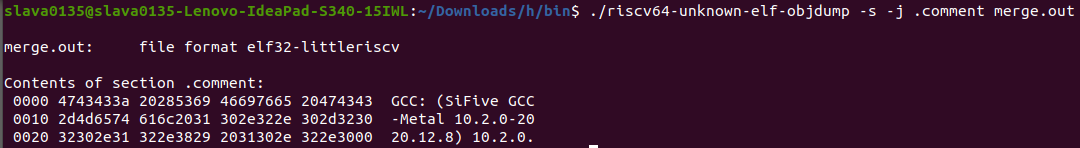








Секция .comment:



Секция .riscv.attributes:

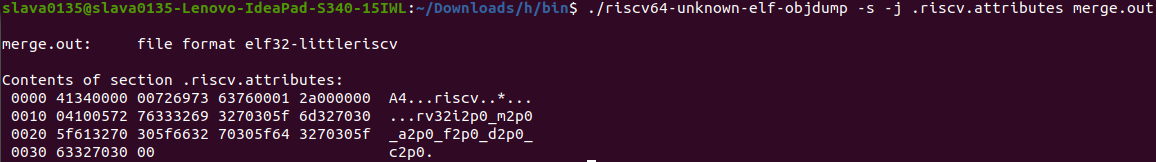


Таблица символов:merge.out: file format elf32-littleriscv

SYMBOL TABLE:

00010074 l d .text 00000000 .text

0001ee50 l d .rodata 00000000 .rodata

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

000200b0 l d .init\_array 00000000 .init\_array

000200b8 l d .fini\_array 00000000 .fini\_array

000200c0 l d .data 00000000 .data

00020a60 l d .sdata 00000000 .sdata

00020a8c l d .sbss 00000000 .sbss

00020aa4 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

00020aa4 l O .bss 00000001 completed.1

000200b8 l O .fini\_array 00000000 \_\_do\_global\_dtors\_aux\_fini\_array\_entry

000100f6 l F .text 00000000 frame\_dummy

00020aa8 l O .bss 00000018 object.0

000200b4 l O .init\_array 00000000 \_\_frame\_dummy\_init\_array\_entry

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

00010112 l F .text 0000003e printArray

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

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

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

000200c0 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 putchar.c

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

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

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

0001287a l F .text 00000092 \_\_sbprintf

0001f030 l O .rodata 00000010 blanks.1

0001f040 l O .rodata 00000010 zeroes.0

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

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

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

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

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

00012dd8 l F .text 00000004 \_\_fp\_unlock

00012ddc 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

00013472 l F .text 00000048 eshdn1

000134ba l F .text 0000004e eshup1

00013508 l F .text 0000009c m16m

000135a4 l F .text 0000008c emovo.constprop.0

00013630 l F .text 0000010c enormlz

0001373c l F .text 0000017a eshift.part.0

000138b6 l F .text 0000008e emovi

00013944 l F .text 000000e0 ecmp

00013a24 l F .text 000002ea emdnorm

00013d0e l F .text 000000ee eiremain

00013dfc l F .text 00000376 emul

00014172 l F .text 0000042a ediv

0001459c l F .text 0000010c e113toe.isra.0

0001f178 l O .rodata 00000014 ezero

0001f18c l O .rodata 00000014 eone

0001f1a0 l O .rodata 00000104 etens

0001f2a4 l O .rodata 00000104 emtens

0001f3a8 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

0001f498 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

0001f72c l O .rodata 00000010 blanks.1

0001f73c l O .rodata 00000010 zeroes.0

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

0001878e l F .text 00000098 \_\_sprint\_r.part.0

00019584 l F .text 00000094 \_\_sbprintf

0001f8b8 l O .rodata 00000010 blanks.1

0001f8c8 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

0001980c 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

0001fb88 l O .rodata 00000010 blanks.1

0001fb98 l O .rodata 00000010 zeroes.0

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

00020aa0 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

000200ac l O .eh\_frame 00000000 \_\_FRAME\_END\_\_

00000000 l df \*ABS\* 00000000

000200bc l .fini\_array 00000000 \_\_fini\_array\_end

000200b8 l .fini\_array 00000000 \_\_fini\_array\_start

000200b8 l .init\_array 00000000 \_\_init\_array\_end

000200b0 l .init\_array 00000000 \_\_preinit\_array\_end

000200b0 l .init\_array 00000000 \_\_init\_array\_start

000200b0 l .init\_array 00000000 \_\_preinit\_array\_start

0001096e g F .text 0000000c putchar

00016394 g F .text 0000006e \_mprec\_log10

00016448 g F .text 00000048 \_\_any\_on

00019de0 g F .text 00000030 \_isatty\_r

0001f570 g O .rodata 00000028 \_\_mprec\_tinytens

0001689a g F .text 00000078 strcpy

00016490 g F .text 00000060 cleanup\_glue

000101b6 g F .text 00000082 merge

0001b5fe g F .text 00000004 \_getpid

00019e10 g F .text 00000036 \_lseek\_r

0001ca9a g F .text 000000f4 .hidden \_\_getf2

0001b542 g F .text 00000032 \_kill\_r

0001c9f4 g F .text 000000a6 .hidden \_\_eqtf2

00010940 g F .text 0000002a printf

000208c0 g \*ABS\* 00000000 \_\_global\_pointer$

0001b0e6 g F .text 00000044 \_wcrtomb\_r

0001b230 g F .text 00000064 \_signal\_r

00016732 g F .text 00000042 \_\_sseek

00013044 g F .text 00000008 \_\_sinit

0001290c g F .text 000000f0 \_\_swbuf\_r

00015506 g F .text 00000058 \_setlocale\_r

00012f10 g F .text 0000004c \_\_sfmoreglue

0001091e g F .text 00000002 \_\_malloc\_unlock

0001e926 g F .text 000000d0 .hidden \_\_floatsitf

00019f2e g F .text 000000d0 memmove

00013036 g F .text 0000000e \_cleanup

00015828 g F .text 00000066 \_Balloc

00019806 g F .text 00000006 \_\_errno

0001b70e g F .text 0000007c \_conv\_stat

000154f4 g F .text 00000006 \_\_localeconv\_l

00019a82 g F .text 00000034 \_fstat\_r

00020a9c g O .sbss 00000004 errno

000166da g F .text 00000004 \_\_seofread

00020a60 g .sdata 00000000 \_\_SDATA\_BEGIN\_\_

0001096a g F .text 00000004 \_putchar\_r

00019e46 g F .text 000000e8 memcpy

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

000169ee g F .text 00001da0 \_svfprintf\_r

00016338 g F .text 0000005c \_\_ratio

000102c0 g F .text 00000008 malloc

00010920 g F .text 00000020 \_printf\_r

0001cb8e g F .text 000000f4 .hidden \_\_letf2

0001ed9c g F .text 0000002c .hidden \_\_udivsi3

0001b294 g F .text 0000007a \_raise\_r

0001b5c4 g F .text 0000003a \_fstat

00019674 g F .text 0000003e \_\_assert\_func

0001f4a8 g O .rodata 000000c8 \_\_mprec\_tens

00020a94 g O .sbss 00000004 \_\_malloc\_top\_pad

0001b574 g F .text 00000002 \_getpid\_r

00020a7c g O .sdata 00000000 .hidden \_\_dso\_handle

000154fa g F .text 00000006 \_localeconv\_r

00019ab6 g F .text 0000032a \_\_sfvwrite\_r

00015b3a g F .text 0000006c \_\_i2b

0001b30e g F .text 00000088 \_\_sigtramp\_r

00010a42 g F .text 00000030 \_sbrk\_r

00019ffe g F .text 00000036 \_read\_r

0001988c g F .text 0000003e \_fclose\_r

00012d80 g F .text 0000004a fflush

00020a90 g O .sbss 00000004 \_\_malloc\_max\_sbrked\_mem

00016180 g F .text 000000c8 \_\_b2d

0001bbd2 g F .text 000003ac .hidden \_\_umoddi3

0001b602 g F .text 00000024 \_isatty

00020a78 g O .sdata 00000004 \_global\_impure\_ptr

0001a034 g F .text 000003e6 \_realloc\_r

00010256 g F .text 0000006a \_\_libc\_init\_array

0001b78a g F .text 00000448 .hidden \_\_udivdi3

00019a1a g F .text 0000001e \_fputwc\_r

0001f598 g O .rodata 00000028 \_\_mprec\_bigtens

0001599e g F .text 000000ca \_\_s2b

0001b17c g F .text 0000000c abort

0001b692 g F .text 00000050 \_sbrk

00015f64 g F .text 00000040 \_\_mcmp

00013054 g F .text 0000000e \_\_fp\_lock\_all

00013070 g F .text 00000038 \_\_libc\_fini\_array

000164f0 g F .text 000000ae \_reclaim\_reent

00015a68 g F .text 00000048 \_\_hi0bits

0001097a g F .text 0000005a \_putc\_r

0001e86a g F .text 000000bc .hidden \_\_fixtfsi

00015d7e g F .text 000000d6 \_\_pow5mult

0001edc8 g F .text 00000008 .hidden \_\_umodsi3

0001ee12 g F .text 0000003c .hidden \_\_clzsi2

0001304c g F .text 00000002 \_\_sfp\_lock\_acquire

00015794 g F .text 00000094 memchr

0001661a g F .text 00000040 \_sprintf\_r

00013172 g F .text 00000220 \_free\_r

0001555e g F .text 00000006 \_\_locale\_mb\_cur\_max

00012afa g F .text 000000be \_\_call\_exitprocs

00020a84 g O .sdata 00000004 \_\_malloc\_sbrk\_base

00010088 g F .text 0000003a \_start

0001b63a g F .text 0000002c \_lseek

00015e54 g F .text 00000110 \_\_lshift

0001cb8e g F .text 000000f4 .hidden \_\_lttf2

0001b410 g F .text 0000006c signal

0001c9f4 g F .text 000000a6 .hidden \_\_netf2

0001a41a g F .text 0000012c \_\_ssprint\_r

0001b188 g F .text 0000006c \_\_register\_exitproc

000154aa g F .text 0000004a \_ldcheck

00015ba6 g F .text 000001d8 \_\_multiply

0001ed7c g F .text 00000014 .hidden \_\_mulsi3

00016974 g F .text 0000007a strncpy

00020ac0 g O .bss 00000028 \_\_malloc\_current\_mallinfo

00016248 g F .text 000000f0 \_\_d2b

0001e9f6 g F .text 0000014c .hidden \_\_extenddftf2

000197d6 g F .text 00000030 \_close\_r

00012a06 g F .text 000000f4 \_\_swsetup\_r

0001bf7e g F .text 000005ca .hidden \_\_divdf3

00012f5c g F .text 000000da \_\_sfp

00016402 g F .text 00000046 \_\_copybits

00020af0 g .bss 00000000 \_\_BSS\_END\_\_

000204e8 g O .data 00000408 \_\_malloc\_av\_

00013052 g F .text 00000002 \_\_sinit\_lock\_release

0001c548 g F .text 000004ac .hidden \_\_muldf3

000166a2 g F .text 00000038 \_\_sread

0001091c g F .text 00000002 \_\_malloc\_lock

00012d4c g F .text 00000034 \_fflush\_r

000196c8 g F .text 0000010e \_calloc\_r

00020a8c g .sbss 00000000 \_\_bss\_start

00010874 g F .text 000000a8 memset

00010150 g F .text 00000066 main

00020a8c g O .sbss 00000004 \_\_malloc\_max\_total\_mem

000129fc g F .text 0000000a \_\_swbuf

00016774 g F .text 00000008 \_\_sclose

000198ca g F .text 00000042 fclose

000102d2 g F .text 000005a2 \_malloc\_r

0001961e g F .text 00000020 \_\_ascii\_wctomb

0001990c g F .text 00000020 \_fiprintf\_r

0001b47c g F .text 0000003e \_init\_signal

00013392 g F .text 0000006c \_fwalk

0001574e g F .text 00000006 \_mbtowc\_r

000109d4 g F .text 0000006e putc

0001ed94 g F .text 00000058 .hidden \_\_divsi3

000130a8 g F .text 000000ca \_malloc\_trim\_r

0001677c g F .text 0000011e strcmp

00019576 g F .text 0000000e vfiprintf

0001cc82 g F .text 00000d62 .hidden \_\_multf3

0001665a g F .text 00000048 sprintf

0001fc5c g O .rodata 00000100 .hidden \_\_clz\_tab

00020a98 g O .sbss 00000004 \_PathLocale

000196bc g F .text 0000000c atexit

0001963e g F .text 00000036 \_write\_r

00015564 g F .text 00000058 setlocale

00020a80 g O .sdata 00000004 \_impure\_ptr

00012bb8 g F .text 00000194 \_\_sflush\_r

0001ca9a g F .text 000000f4 .hidden \_\_gttf2

0001a546 g F .text 00000ba0 \_svfiprintf\_r

00015754 g F .text 00000040 \_\_ascii\_mbtowc

0001d9e4 g F .text 00000e86 .hidden \_\_subtf3

0001613a g F .text 00000046 \_\_ulp

00013062 g F .text 0000000e \_\_fp\_unlock\_all

00015500 g F .text 00000006 localeconv

000156cc g F .text 00000082 \_\_swhatbuf\_r

000200c0 g .data 00000000 \_\_DATA\_BEGIN\_\_

0001b6e2 g F .text 0000002c \_write

00020a8c g .sdata 00000000 \_edata

00020af0 g .bss 00000000 \_end

00019956 g F .text 000000c4 \_\_fputwc

000166de g F .text 00000054 \_\_swrite

00020a88 g O .sdata 00000004 \_\_malloc\_trim\_threshold

00010238 g F .text 0000001e exit

0001992c g F .text 0000002a fiprintf

00018834 g F .text 00000d42 \_vfiprintf\_r

000133fe g F .text 00000074 \_fwalk\_reent

00015fa4 g F .text 00000196 \_\_mdiff

0001edec g F .text 00000024 .hidden \_\_modsi3

0001b626 g F .text 00000014 \_kill

0001304e g F .text 00000002 \_\_sfp\_lock\_release

000146a8 g F .text 00000e02 \_ldtoa\_r

0001f918 g O .rodata 00000101 \_ctype\_

0001b666 g F .text 0000002c \_read

0001b5a2 g F .text 00000022 \_exit

000155bc g F .text 00000110 \_\_smakebuf\_r

00016912 g F .text 00000062 strlen

0001b1f4 g F .text 0000003c \_init\_signal\_r

000196b2 g F .text 0000000a \_\_assert

00018826 g F .text 0000000e \_\_sprint\_r

00019618 g F .text 00000006 \_wctomb\_r

00010a72 g F .text 00001dfa \_vfprintf\_r

00015ab0 g F .text 0000008a \_\_lo0bits

0001b4ba g F .text 00000088 \_\_sigtramp

0001b12a g F .text 00000052 wcrtomb

0001659e g F .text 0000007c frexp

000208f0 g O .data 0000016c \_\_global\_locale

0001286c g F .text 0000000e vfprintf

0001eb42 g F .text 00000238 .hidden \_\_trunctfdf2

00019a38 g F .text 0000004a fputwc

0001b396 g F .text 0000007a raise

0001b576 g F .text 0000002c \_close

000102c8 g F .text 0000000a free

00013050 g F .text 00000002 \_\_sinit\_lock\_acquire

000158a0 g F .text 000000fe \_\_multadd

0001588e 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.

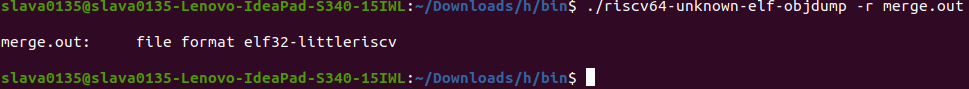


Таблица перемещений оказывается пуста, все необходимые релокации, оптимизации и замены инструкций были успешно проведены компоновщиком.

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:

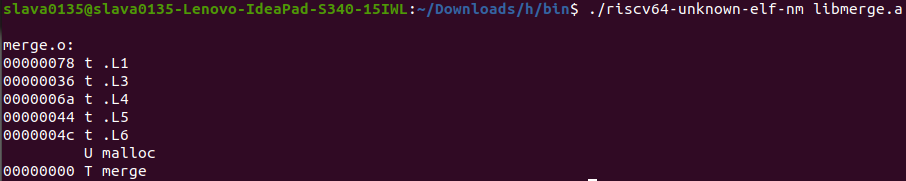


Просмотрим, какие файлы туда вошли:



Туда вошел только объектный файл с нашей функцией.

Просмотрим список символов библиотеки:



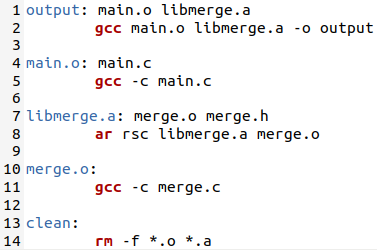
В выводе утилиты “nm” кодом “T” обозначаются символы, определенные в соответствующем объектном файле, кодом “U” - внешние символы.

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

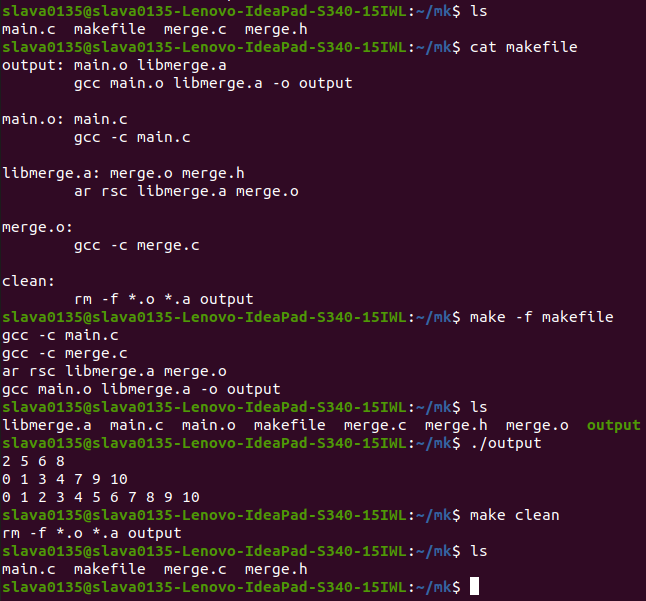
Теперь осуществим сборку тестовой программы используя эту библиотеку:



Теперь напишем makefile для сборки программы:



И осуществим сборку программы через gcc:



* ls => список файлов до сборки.
* cat makefile => просмотр make файла.
* make -f makefile => осуществляем сборку
* ls => список файлов после сборки
* ./output => запускаем программу, печатаются элементы массивов, сначала первый, потом второй и выходной.
* 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.