Санкт-Петербургский политехнический университет Петра Великого

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчёт по лабораторной работе № 4**

**Дисциплина**: Низкоуровневое программирование

**Тема**: Раздельная компиляция.

Варианта №1

Выполнила

студентка гр. 3530901/10003 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Р. Саушкина

(подпись)

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.А. Коренев

(подпись)

“ ” 2022 г.

Санкт-Петербург

**Оглавление**

[**Техническое задание** 3](#_Toc123164554)

[**Программа на языке C** 4](#_Toc123164555)

[**Сборка программ по шагам** 5](#_Toc123164556)

[**Формирование статической библиотеки, разработка make-файлов** 6](#_Toc123164557)

# **Техническое задание**

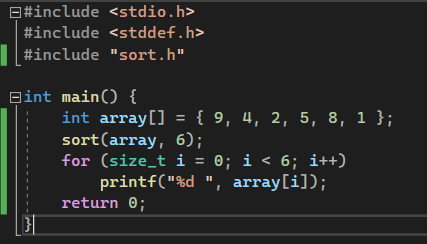
1. На языке C разработать функцию, реализующую сортировку массива выбором. Поместить определение функции в отдельный исходный файл, оформить заголовочный файл. Разработать тестовую программу на языке C.

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 и исполняемом файле.

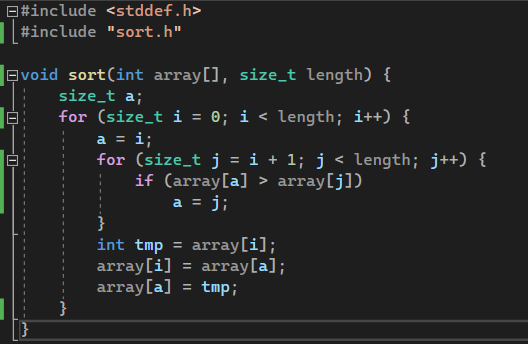
3. Выделить разработанную функцию в статическую библиотеку. Разработать make-файлы для сборки библиотеки и использующей ее тестовой программы. Проанализировать ход сборки библиотеки и программы, созданные файлы зависимостей.

# **Программа на языке C**

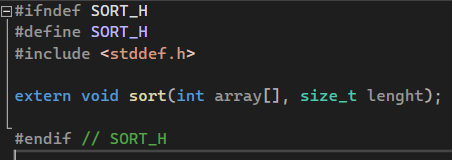
*Листинг 1. Программа «main.c»*



*Листинг 2. Программа «sort.c»*

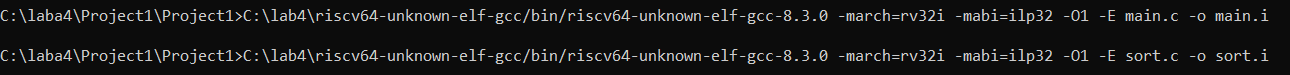


*Листинг 3. Заголовочный файл «sort.h»*

****

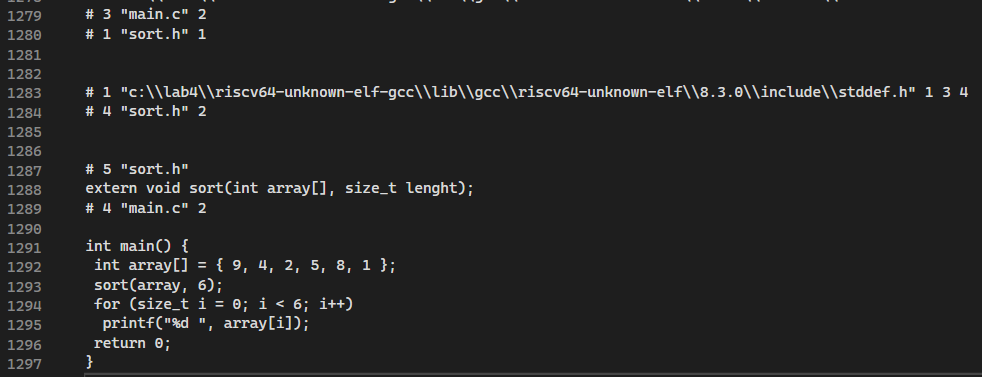
****

# **Сборка программ по шагам**

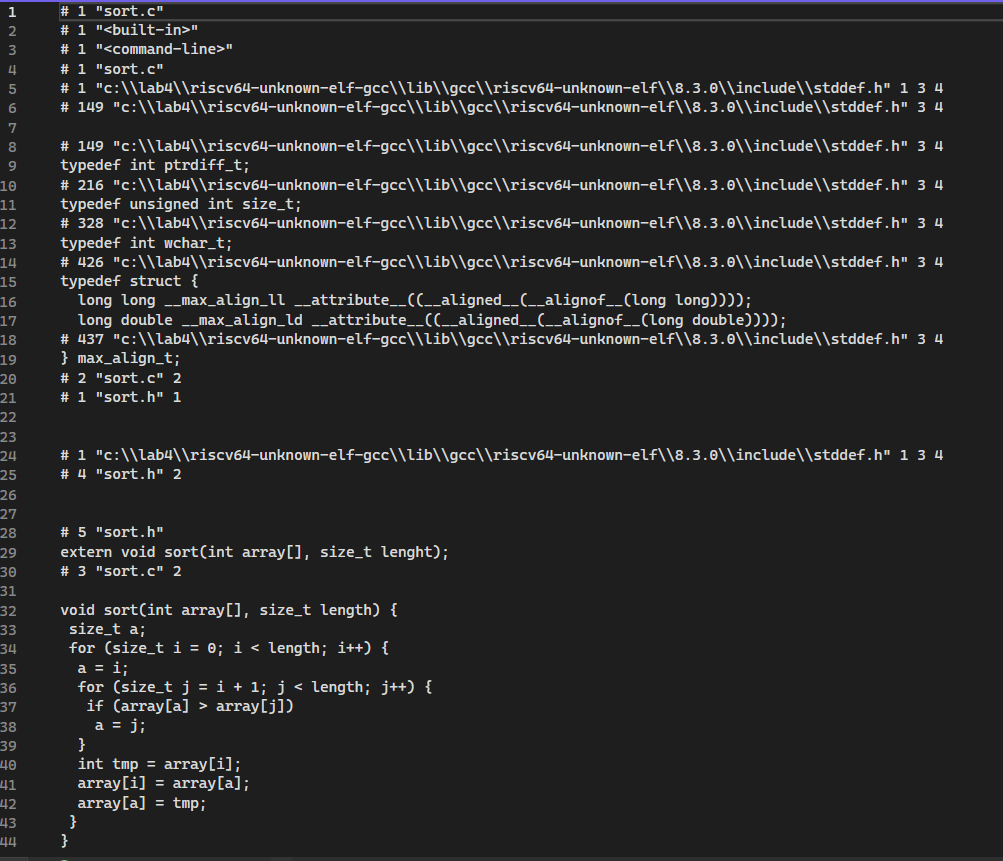
Произведем препроцессирование с помощью следующих команд:

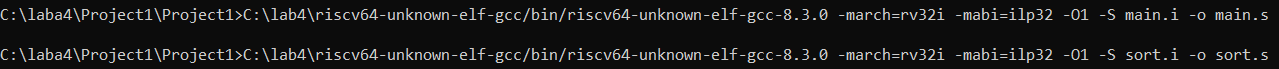
Результат содержится в файлах main.i и sort.i. Так как в main.c используется библиотека stdio.h, в main.i записано около 1300 добавочных строк.

*Листинг 4. Файл main.i*

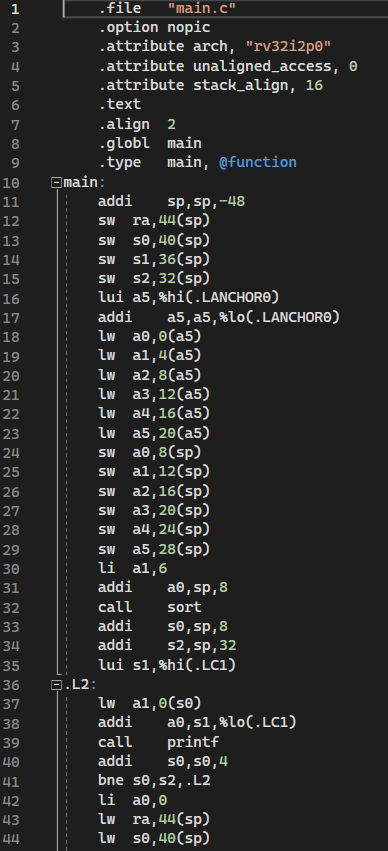
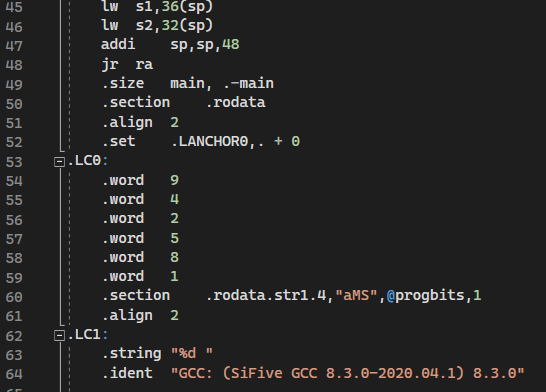


*Листинг 5. Файл sort.i*

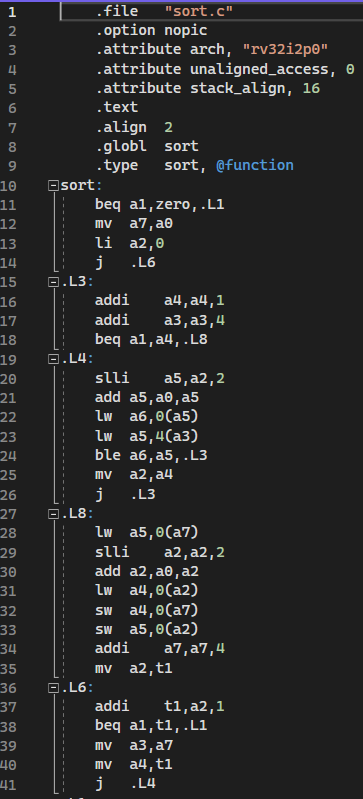


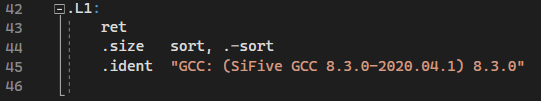
Произведем компиляцию с помощью следующих команд:

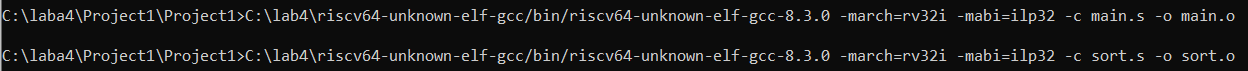
*Листинг 6. Файл main.s*



*Листинг 7. Файл sort.s*

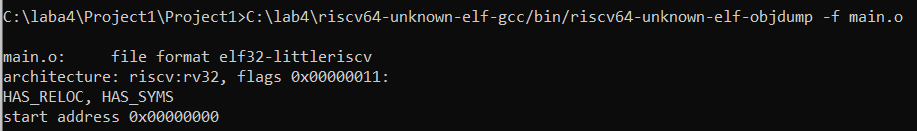
**

**

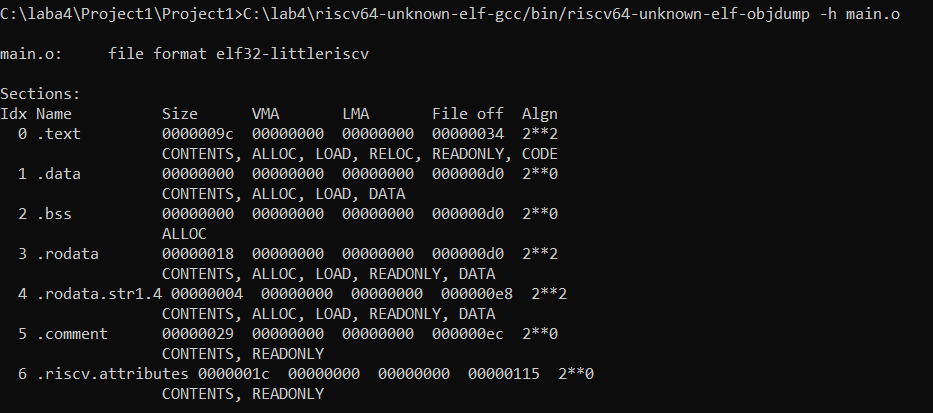
Произведем ассемблирование с помощью следующих команд:

Объектный файл не является текстовым, используем objdump для изучения файлов.

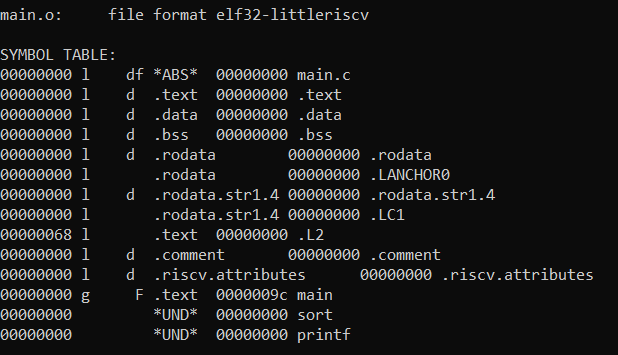
*Листинг 8. Информация о main.o*



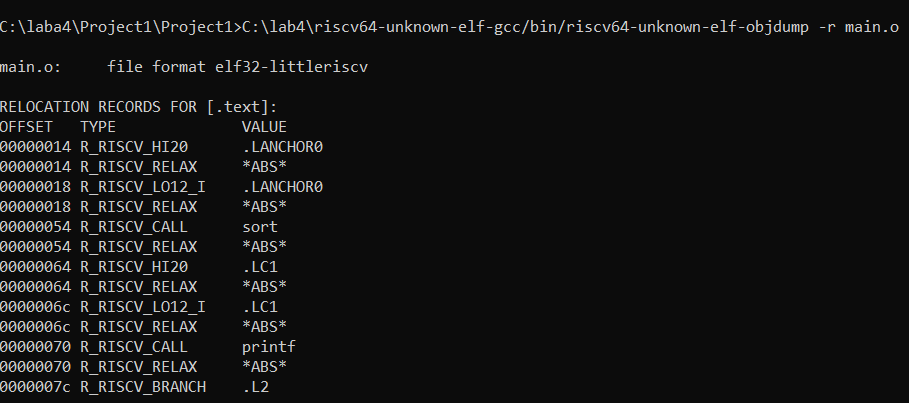
*Листинг 9. Секций main.o*



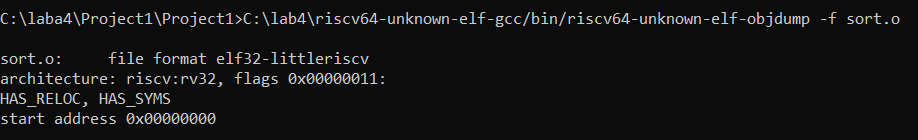
*Листинг 10. Таблица символов main.o*

**

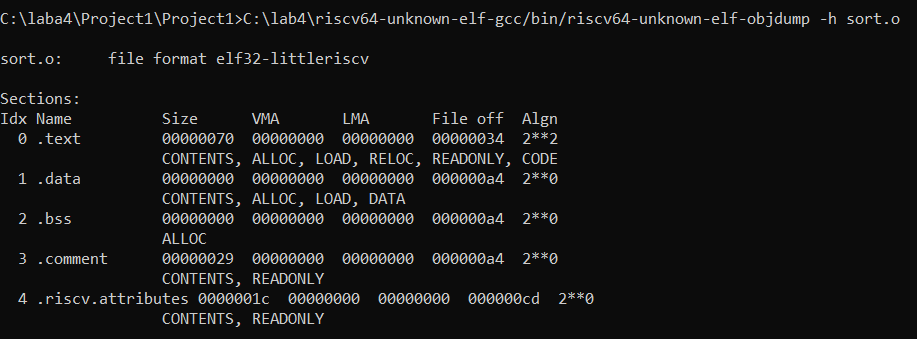
*Листинг 11. Таблица перемещений main.o*

**

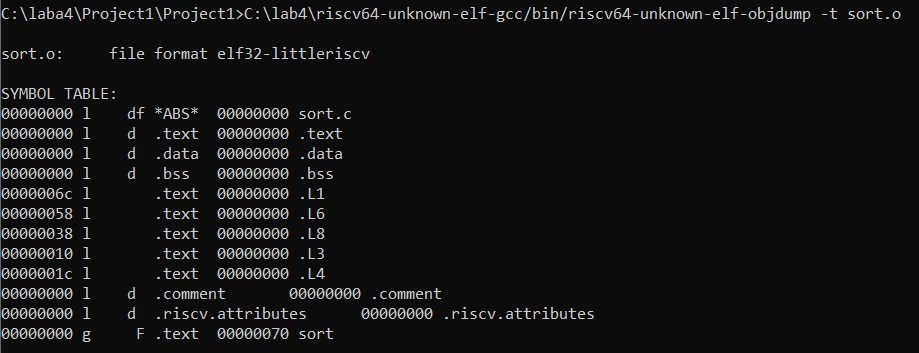
*Листинг 12. Информация sort.o*

**

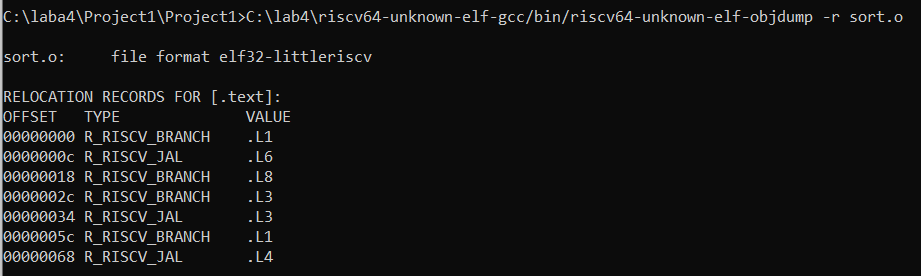
*Листинг 13. Секция sort.o*

**

*Листинг 14. Таблица символов sort.o*

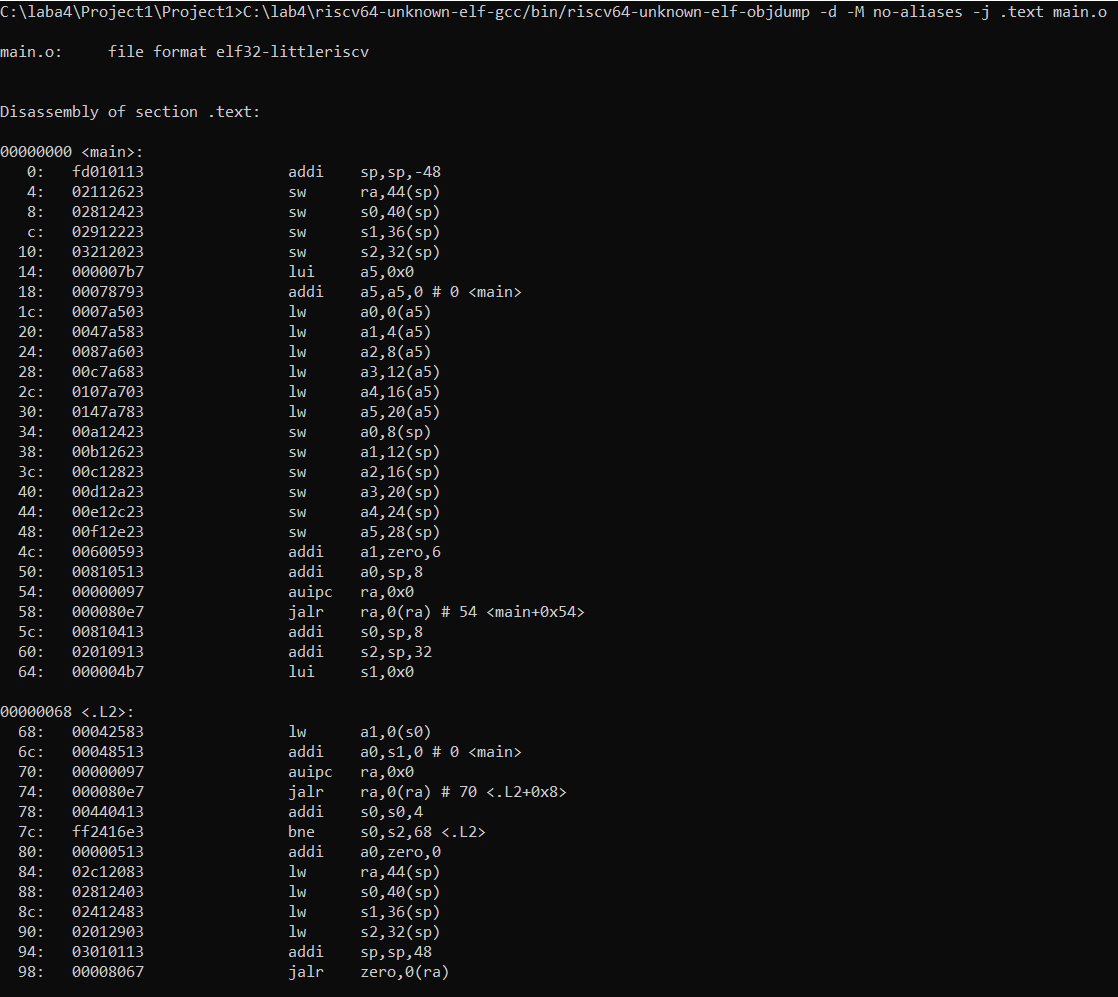


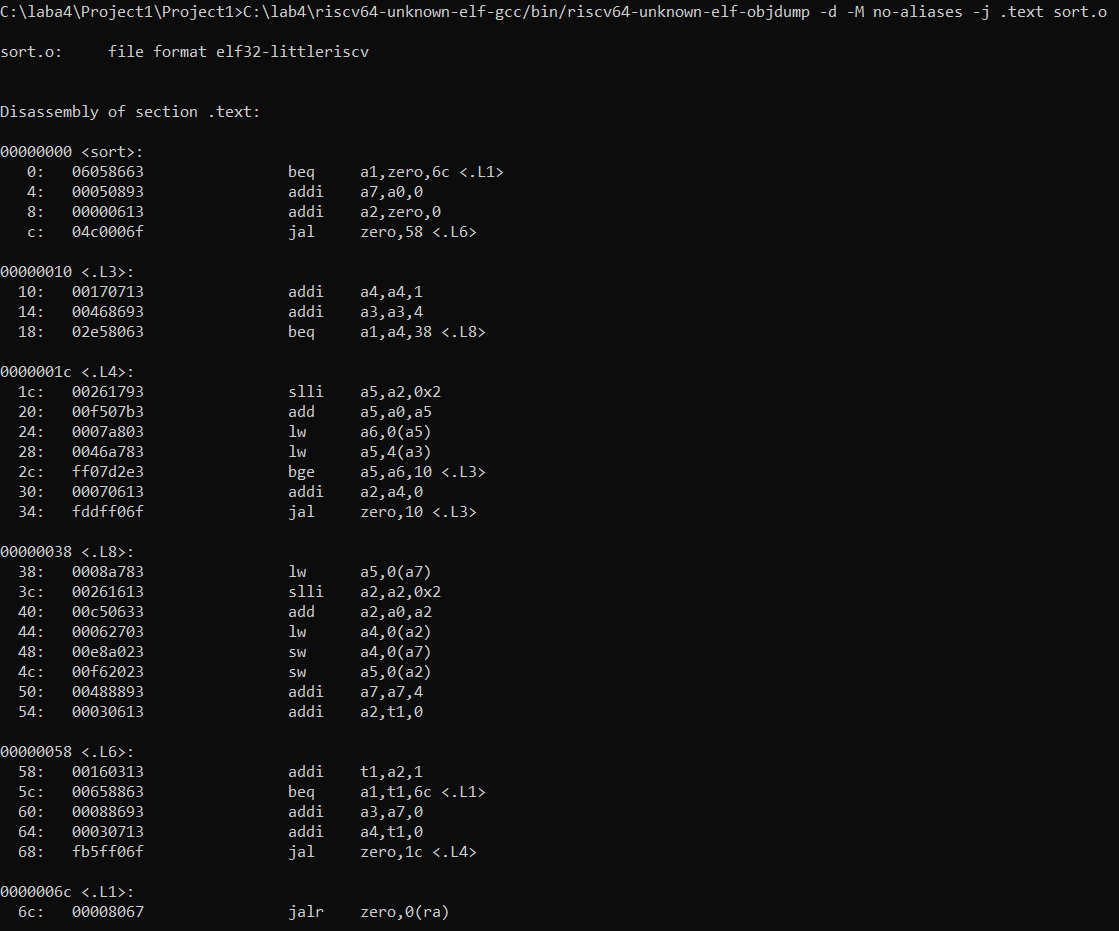
*Листинг 15. Таблица перемещений sort.o*

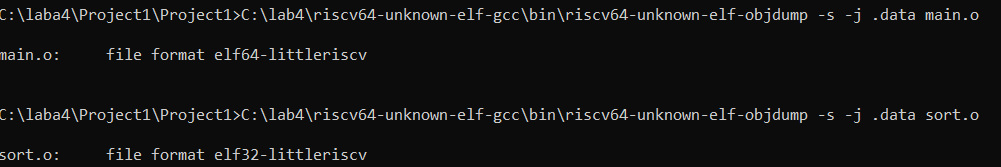
**

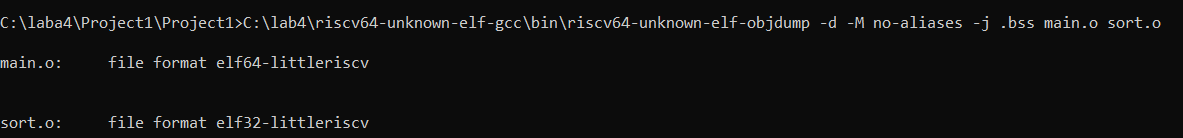
Изучим содержимое секции .text

*Листинг 16. Секция .text*

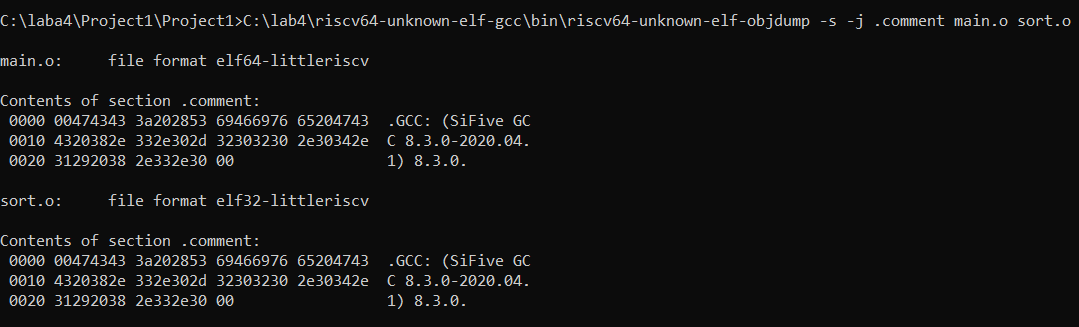




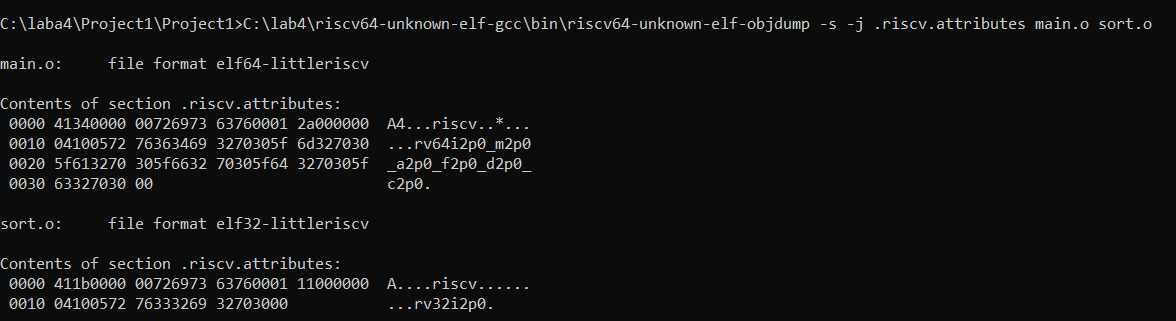
Секции .data объектных файлов – секции инициализированных данных равны нулю.

Секции .bss объектных файлов – секции данных, инициализированных нулями – не содержат данных.

Секция .comment – секция данных о версиях – и для одного и для другого файла содержит одни и те же значения – сведения о GCC версии 8.3.0 от SiFive.



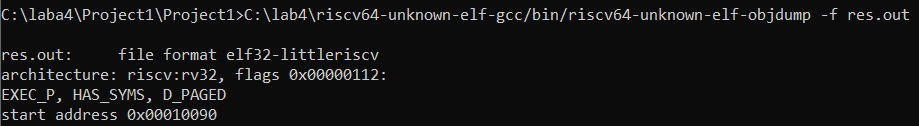
Секция .riscv.attributesобоих объектных файлов содержит одну и ту же информацию об используемой архитектуре команд RV32I:



Произведем компоновку с помощью следующей команды:

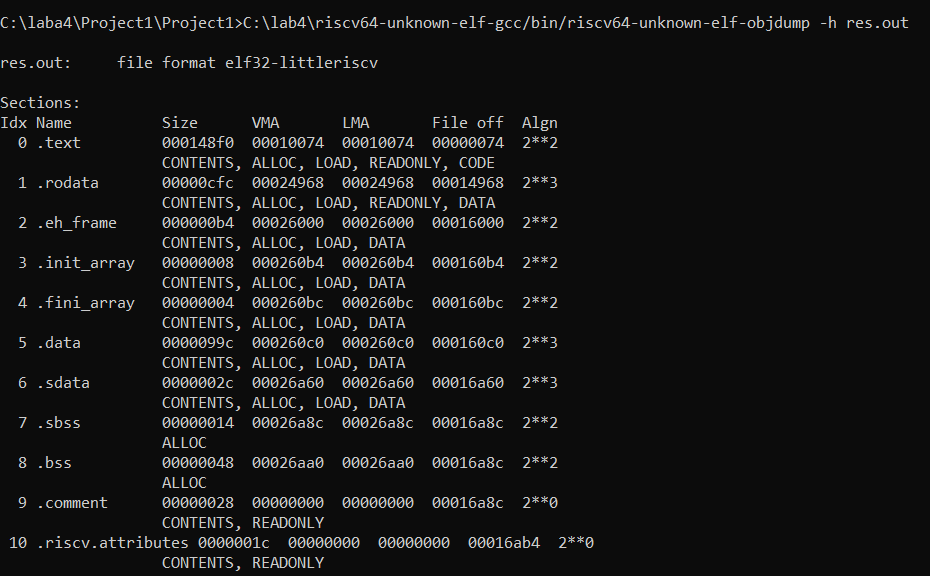
Результатом будет являться исполняемый файл res.out. Он является бинарным, поэтому его изучения будем использовать objdump.

*Листинг 17. Информация о res.out*

**

Флаг EXEC\_P указывает, что файл действительно является исполняемым, после загрузки его выполнение должно начаться с адреса 0x00010090.

*Листинг 18. Секции res.out*

**

В исполняемом файле действительно производится слияние содержания секций обоих объектных файлов.

*Листинг 19. Таблица перемещений res.out*

**

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 сортировка массива выбором.

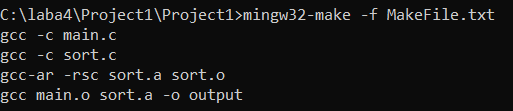
# **Формирование статической библиотеки, разработка make-файлов**

Статическая библиотека 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.

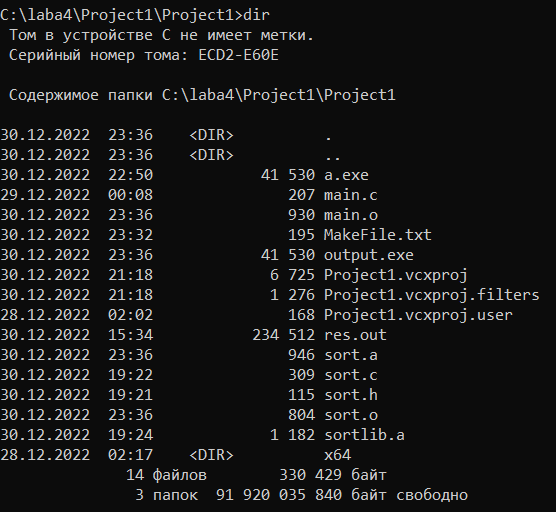
Поместим sort.o в статическую библиотеку:

Создадим make-файл, который будет создавать библиотеку и собирать программу.

*Листинг 20. Запуск MakeFile*



*Листинг 21. Полученная директория*



*Листинг 22. Результат работы output.exe*

****

# **Выводы**

В ходе выполнения лабораторной работы были закреплены знания языка C, ассемблера RISC-V, получены навыки работы спрепроцессором, компилятором, ассемблером и компоновщиком пакета GCCи 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 – «сортировка выбором», а затем проведена проверка правильности перевода программы решения этой задачи в набор инструкций, выполняемых процессором.