Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчёт по лабораторной работе № 4**

Дисциплина: Низкоуровневое программирование

Вариант 6.

Выполнил студент гр. 3530901/90002 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.В. Дергачев

(подпись)

Принял преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.С. Степанов

(подпись)

“ ” 2021 г.

Санкт-Петербург

2021

**Задача**

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 и исполняеммом 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 задания**

По варианту номер 6 необходимо реализовать нахождение медианы массива in-place.

**Решение**

Медиана – элемент массива, который находится ровно посередине после сортировки. Для выполнения поставленной задачи программа проходится по массиву, сравнивая каждое число с каждым, прибавляя к счетчику 1, если второе число больше текущего; вычитая 1, если число меньше текущего и не делая ничего, если они равны. В конце каждого прохода проверяется значение счетчика, если он меньше 0 или больше 1 (количество чисел больше текущего меньше, чем количество чисел меньше текущего и наоборот, соответственно), то программа за место текущего числа берет следующее из массива и начинает проход заново, если значение счетчика равно 0 или 1 (0, когда в массиве нечетное количество элементов и 1, когда - четное), то медиана найдена и программа, записав ее, завершает работу.

1. **Написание программы на языке С**

Была написана программа, которая находит медиану массива. Функция помещена в отдельный файл, написан заголовочный файл.

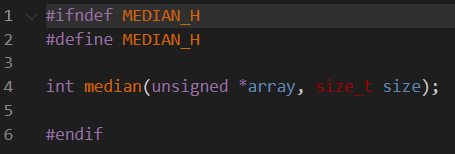


Рис.1 Заголовочный файл

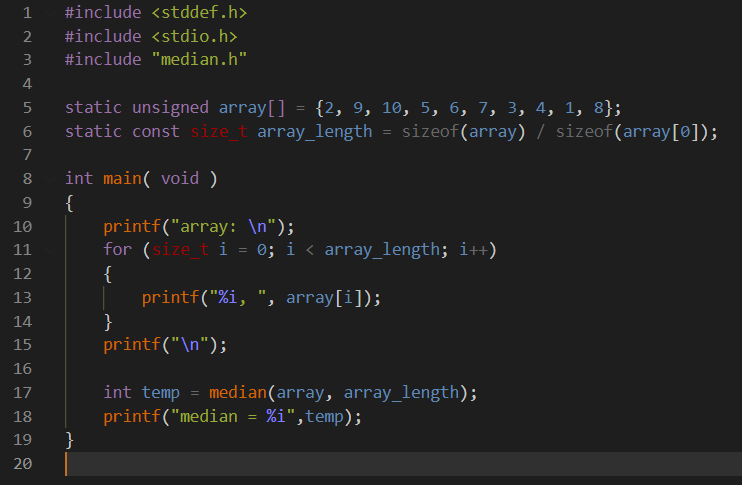


Рис.2 Текст программы

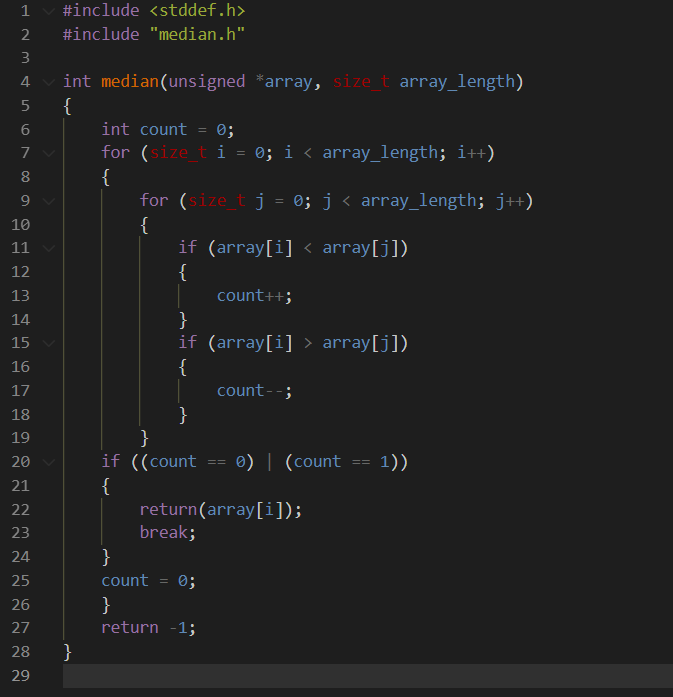


Рис.3 Функция median

Произведем компиляцию программы:



Рис. 4 Результат исполнения программы

1. **Сборка программы «по шагам»**

Препроцессирование.

Выполним препроцессирование файлов с помощью пакета разработки «SiFive GNU Embedded Toolchain» для RISK-V. Для этого необходимо выполнить следующие команды:

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -E main.c -o main.i >log\_main\_prepr.txt 2>&1

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -E median.c -o median.i >log\_median\_prepr.txt 2>&1

**-march=rv64iac -mabi=lp64** – объявляем, что целевым является процессор с базовой архитектурой системы команд RV64IAC

**-O1** – выполняются простые оптимизации генерируемого кода

**-v** – печатаются (в стандартный поток ошибок) выполняемые драйвером команды, а также дополнительную информацию

**>** - печатать в файл

**-E** – обработка файлов будет выполнятся только препроцессором

Рассмотрим результаты препроцессирования.

|  |  |
| --- | --- |
|  |  |
| Рис. 5 Файл main.i | Рис. 6 Файл median.i |

Компиляция

Выполним компиляцию файлов с помощью пакета разработки «SiFive GNU Embedded Toolchain» для RISK-V. Для этого необходимо выполнить следующие команды:

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -S -fpreprocessed median.i -o median.s>log\_median\_comp.txt 2>&1

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -S -fpreprocessed main.i -o main.s>log\_main\_comp.txt 2>&1

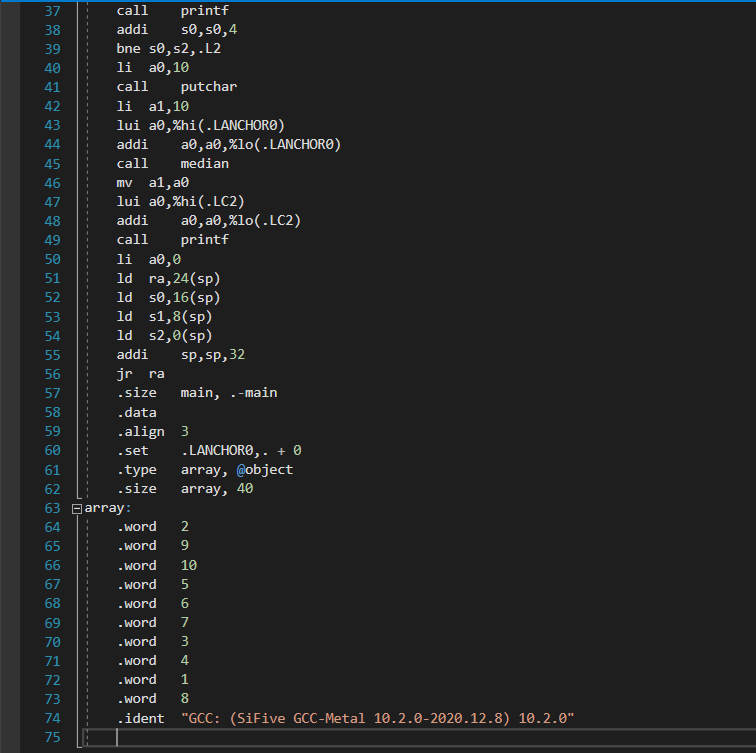
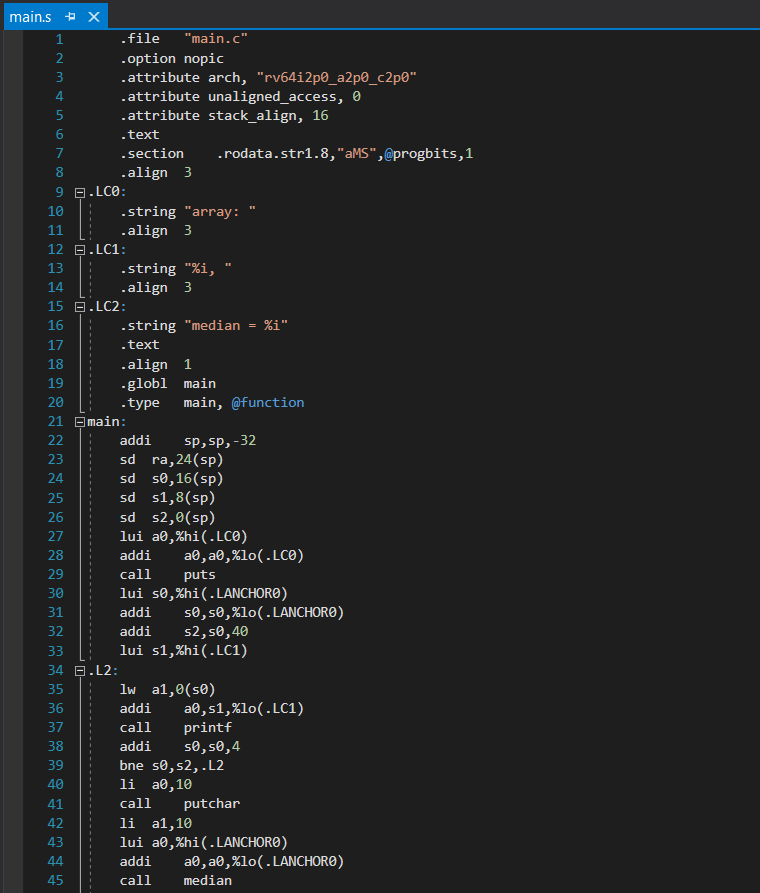


Рис. 7 файл main.s

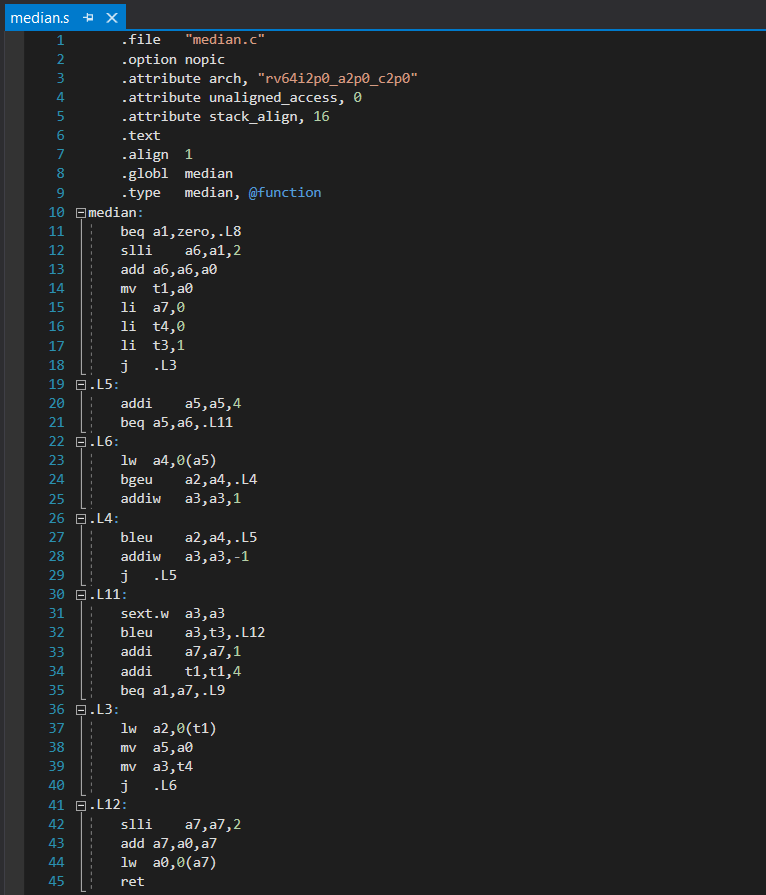
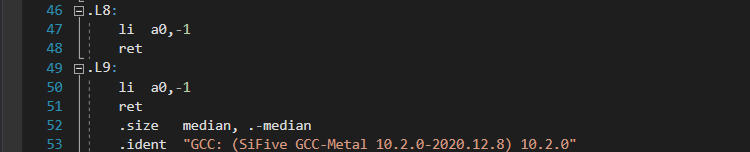
 

Рис. 8 файл median.s

Объектный файл

Выполним ассемблирование для получения объектных файлов программы. Для этого исполним следующие команды:

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -v -c main.s -o main.o >log\_obj\_main.txt 2>&1

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -v -c median.s -o median.o >log\_obj\_median.txt 2>&1

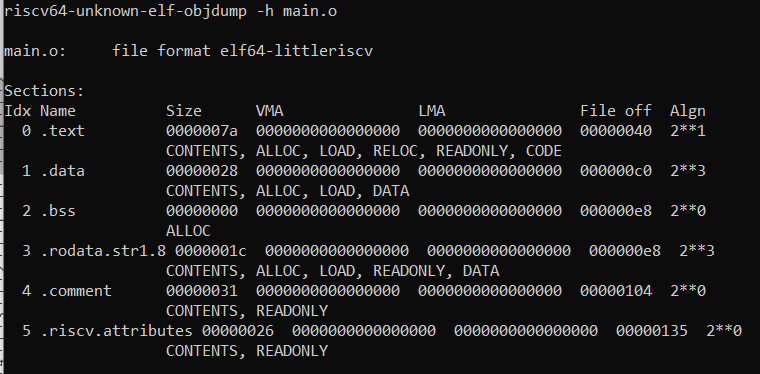


Рис. 9 Хэдер файла main.o, полученный по команде в первой строке.

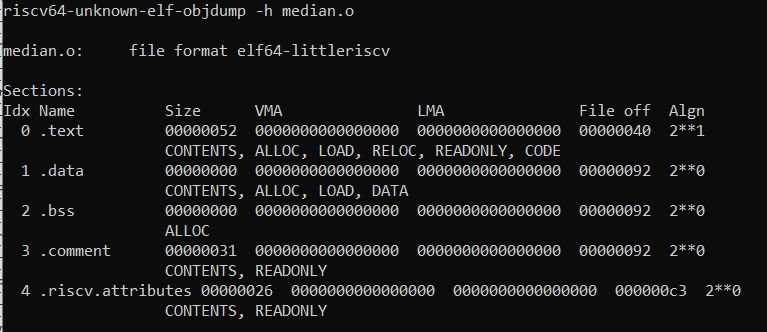


Рис. 10 Хэдер файла median.o, полученный по команде в первой строке.

.text – секция кода, содержащая коды инструкций

.data – секция инициализированных данных

.bss – секция данных, инициализированных нулями

.comment – секция данных о версиях

riscv64-unknown-elf-objdump -d -M no-aliases -j .text main.o

Опция “-d” инициирует процесс дизассемблирования (disassemble),

опция “-M no-aliases” требует использовать в выводе только инструкции системы команд (но не псевдоинструкции ассемблера).

Вывод утилиты:

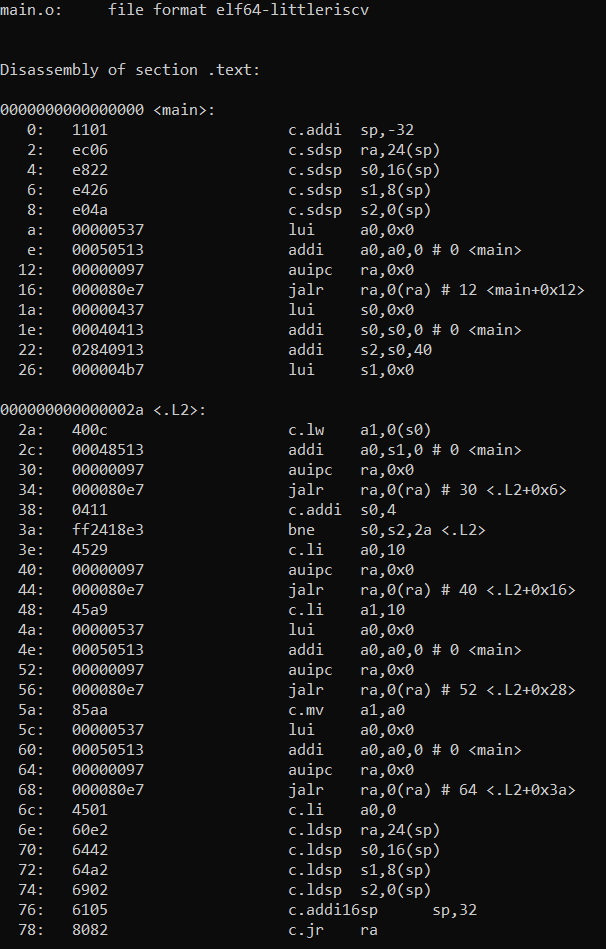


Рис. 11 дизассемблированный файл main.o

Рассмотрим таблицу символов, выполнив команду:

riscv64-unknown-elf-objdump -t median.o main.o

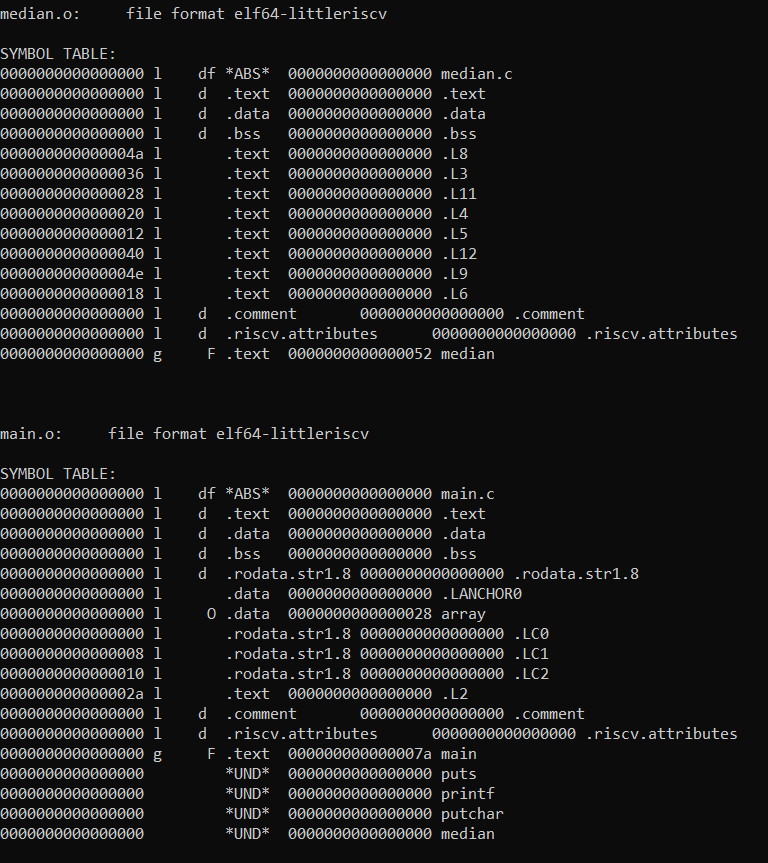


Рис. 12 Таблица символов.

В таблице символов“main.o” имеется интересная запись: символ “zero” типа “\*UND\*” (undefined – не определен).Эта запись означает, что символ “zero” использовался в ассемблерном коде, из которого был получен данный объектный файл, но не был определен; ассемблер сделал вывод о том, что символ должен быть определен где-то еще, и отразил это в таблице символов. Информация обо всех «неоконченных» инструкциях передается ассемблером компоновщику посредством таблицы перемещений, получить которую можно по команде:

riscv64-unknown-elf-objdump -r median.o main.o

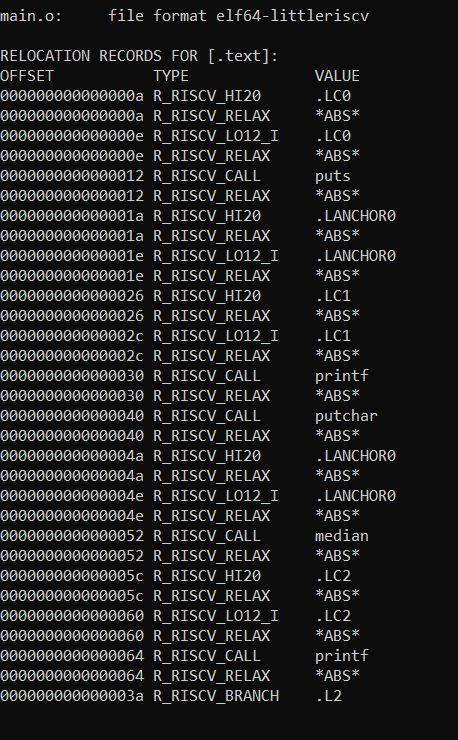
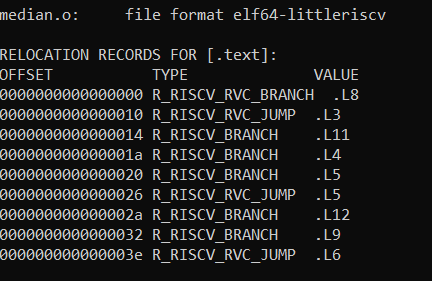


Рис. 13 Таблица перемещений

В таблице перемещений для main.o наблюдаем вызов метода median. Записи типа “R\_RISCV\_RELAX” заносятся в таблицу перемещений в дополнение к записям типа “R\_RISCV\_CALL” (и некоторым другим) и сообщают компоновщику, что пара инструкций, обеспечивающих вызов подпрограммы, может быть оптимизирована.

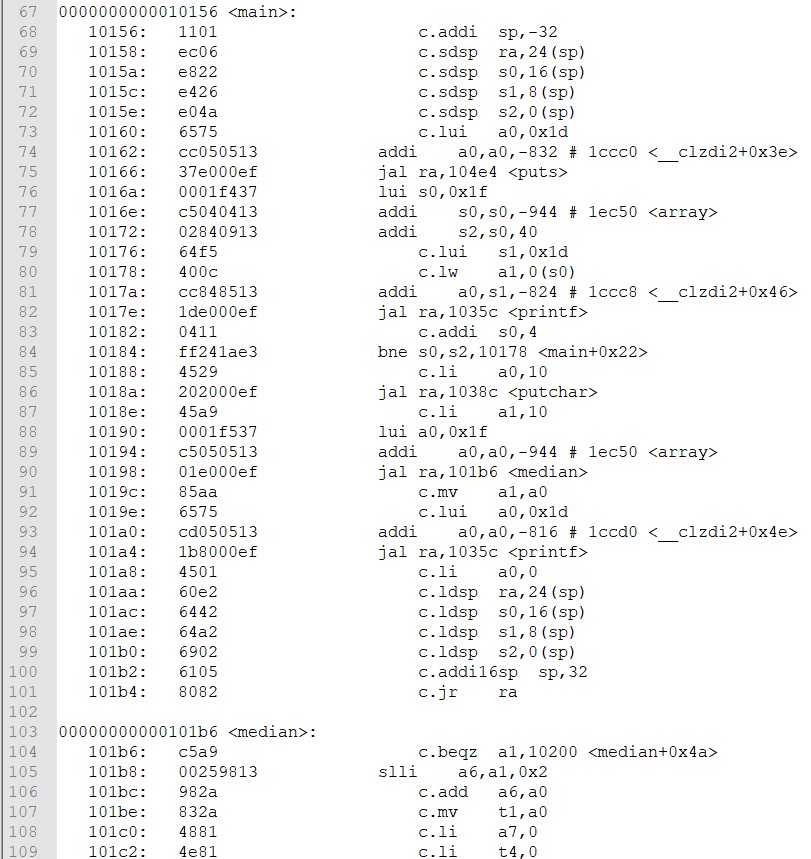
Компоновка

Выполним компоновку командой

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -v main.o median.o -o main.out >log\_out.txt 2>&1

В результате выполнения этой команды был получен файл main.out – исполняемый бинарный файл. Рассмотрим его секцию кода с помощью команды:

riscv64-unknown-elf-objdump –j .text –d –M no-aliases main.out >a.ds



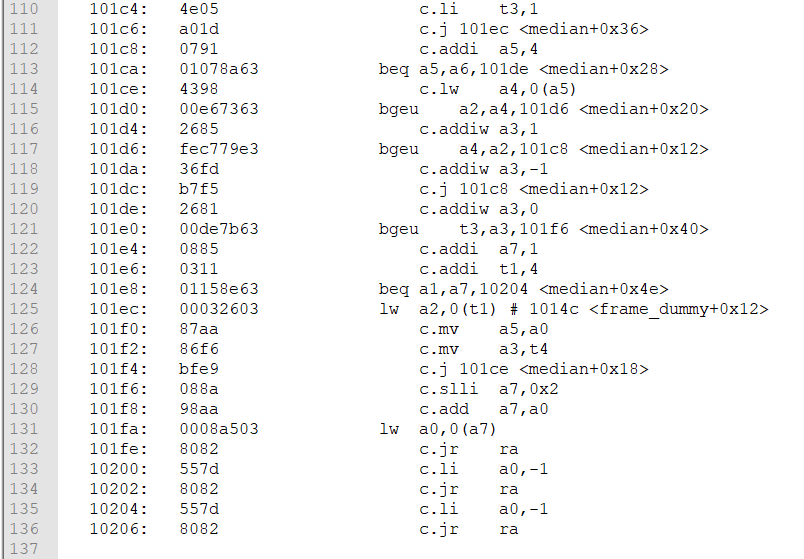


Рис. 14 содержимое файла a.ds

Адресация для вызовов функций изменилась на абсолютную

1. **Создание статической библиотеки**

Выделим функцию median в отдельную статическую библиотеку. Для этого надо получить объектный файл median.o и собрать библиотеку.

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -c median.c -o median.o

riscv64-unknown-elf-ar -rsc libMedian.a median.o

Рассмотрим список символов libMedian.a с помощью команды

riscv64-unknown-elf-nm libMedian.a

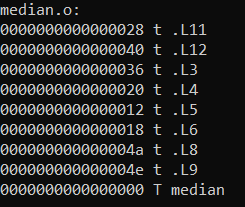


Рис. 15 список символов

В выводе утилиты “nm” кодом “T” обозначаются символы, определенные в соответствующем объектном файле. Используя собранную библиотеку, произведём исполняемый файл тестовой программы с помощью команды:

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 main.c libMedian.a -o main.out

Посмотрим содержимое таблицы символов исполняемого файла с помощью команды:

riscv64-unknown-elf-objdump -t main.out >main.ds и убедимся, что так есть функция median

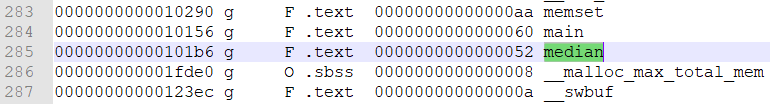


Рис. 16 таблица символов исполняемого файла

Создание make-файлов

Чтобы автоматизировать процесс сборки библиотеки и приложения напишем make-файлы. Используя пример с сайта курса, были написаны следующие файлы:

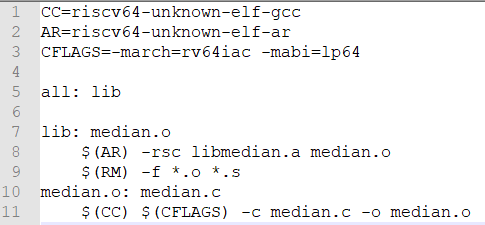


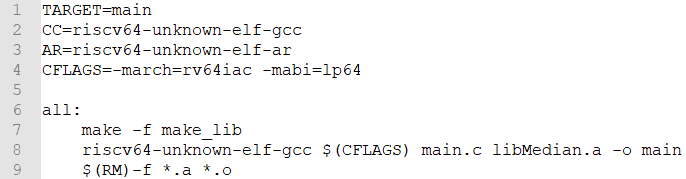
Рис. 17 содержание файла make\_lib

Рис. 18 содержание файла make\_app

Для создания библиотеки необходимо выполнить make\_lib, а для приложения - make\_app.

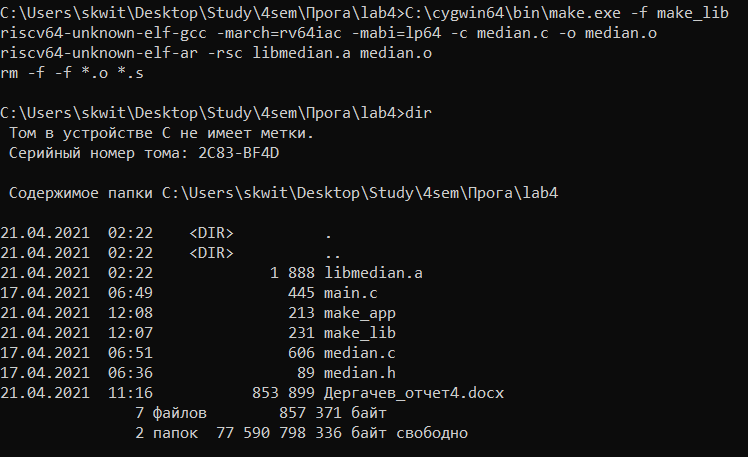


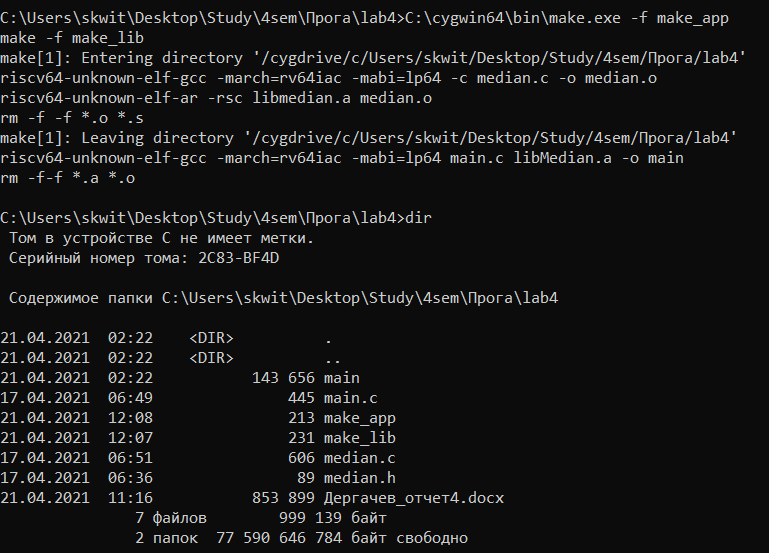
Рис. 19 выполнение файла make\_lib

Рис. 20 выполнение файла make\_app

**Вывод:**

В ходе выполнения работы была написана программа на языке С, была выполнена сборка этой программы по шагам для архитектуры RISC-V, была создана статическая библиотека. Также были написаны два make-файла для автоматизированной сборки библиотеки и тестовой программы.