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

# Тема: Раздельная компиляция

Вариант 6

Выполнил студент гр. 3530901/00002 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.А. Колупаев

(подпись)

Принял старший преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Д.С. Степанов

(подпись)

“ ” 2021 г.

Санкт-Петербург

2021

Цель работы:

1. Изучить методические материалы, опубликованные на сайте курса.

2. Установить пакет средств разработки “SiFive GNU Embedded Toolchain” для RISC-V.

3. 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.

4. 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.

5. 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 6:** Нахождение медианы in-place.

**1.Функция на C**

Сначала разработаем функцию на С, которая будет реализовывать поиск медианы. Напишем функцию в отдельном файле.

Изображение выглядит как текст

Автоматически созданное описание

Рис.1 Заголовочный файл

Изображение выглядит как текст

Автоматически созданное описание

Рис.2 Файл main.c

Изображение выглядит как текст

Автоматически созданное описание

Рис.3 Файл findMedian.c

Алгоритм программы:

1. Проходимся по всему массиву, сравнивая элементы друг с другом
2. В зависимости от сравнения увеличиваем/уменьшаем счётчик
3. Подсчитываем счётчик и, если он равен 0(для нечетных массивов)/-1 и 1 (для четных), выводим результат

Результат работы функции:

Изображение выглядит как текст

Автоматически созданное описание

Рис. 4 Результат работы программы

**2.Сборка простейшей программы «по шагам»**

**Препроцессирование**

Первым шагом является препроцессирование файлов с исходными текстами. Для этого используется пакет разработки «SiFive GNU Embedded Toolchain». Чтобы это выполнить, необходимо использовать команды:

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

*riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -E findMedian.c -o findMedian.i -v -E >log\_ findMedian\_pre.txt 2>&1*

*Изображение выглядит как текст

Автоматически созданное описание*

Рис.5. Фрагмент изначального кода в main.i.

Изображение выглядит как текст

Автоматически созданное описание

Рис.6. Фрагмент изначального кода в findMedian.i

Как видно на рис.5 и 6, в созданных файлах после препроцессирования содержится наш изначальный код.

**Компиляция**

Для выполнения компиляции используем следующие команды:

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

*riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -S -fpreprocessed findMedian.i -o findMedian.s >log\_s\_findMedian.txt 2>&1*

Получаем следующие файлы:

main.s:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

И findMedian.s:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Получаем инструкции на RISC-V.

**Ассемблирование**

Используем следующие команды:

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

*riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -v -c findMedian.s -o findMedian.o >log\_o.txt 2>&1*

После их выполнения получаем объектные файлы main.o и findMedian.o. Для их прочтения будем использовать следующие команды:

*riscv64-unknown-elf-objdump -h main.o*

*riscv64-unknown-elf-objdump -findMedian.o*

*Изображение выглядит как текст

Автоматически созданное описание*

Рис.7. Хедер файла main.o.

Изображение выглядит как текст

Автоматически созданное описание

Рис.8 Хедер файла findMedian.о

Вся информация размещается в секциях:

|  |  |
| --- | --- |
| Секция | Назначение |
| .text | секция кода, в которой содержатся коды инструкций |
| .data | секция инициализированных данных |
| .bss | секция данных, инициализированных нулями |
| .comment | секция данных о версиях размером 12 байт |
| .rodata | секция данных в формате read-only |

Далее вводим команду:

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

И можем более подробно рассмотреть секцию .text.

Изображение выглядит как текст

Автоматически созданное описание

Рис.9. Секция .text файла main.o.

Командой: *riscv64-unknown-elf-objdump -t findMedian.o main.o*

Получаем таблицу символов.

Изображение выглядит как текст, табличка

Автоматически созданное описание

Рис.10. Таблица символов

Изображение выглядит как текст

Автоматически созданное описание

Рис.11. Таблица перемещений.

Здесь можно найти записи типа R\_RISCV\_CALL – сообщения для компоновщика о возможных оптимизациях.

**Компоновка**

Выполняем компоновку командой:

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

Мы получили файл main.out, для рассмотрения его секции, используем:

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

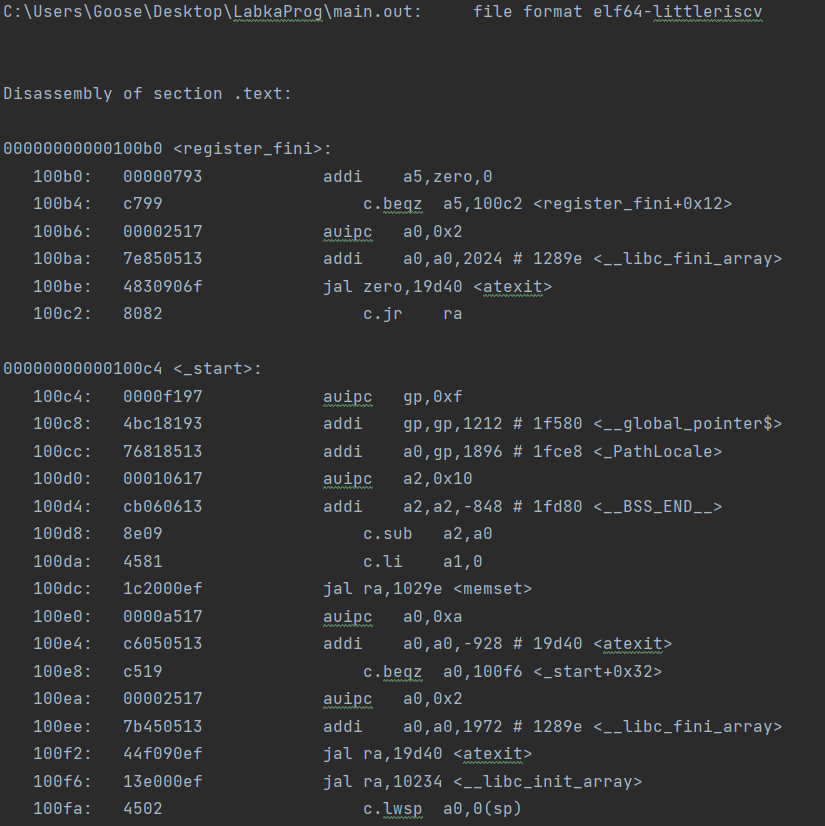


Рис.12. Исполняемый файл.

Изображение выглядит как текст

Автоматически созданное описание

Рис.13. Исполняемый файл.

По рисункам 12 и 13 видим, что адресация изменилась на абсолютную.

**3.Создание статической 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.

Создаем объектный файл findMedian.o и собираем в библиотеку командами:

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

*riscv64-unknown-elf-ar -rsc libStatistics.a findMedian.o*

И можем получить список символовlibStatistics.a:

*riscv64-unknown-elf-nm libStatistics.a*

*Изображение выглядит как стол

Автоматически созданное описание*

Рис.14. Список символов файла libStatistics.a.

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

Сделаем исполняемый файл тестовой программы:

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

Для прочтения используем:

*riscv64-unknown-elf-objdump -t main.out >main.ds*

*Изображение выглядит как текст

Автоматически созданное описание*

Рис.15. Таблица символов main.out.

Видим, что в эту таблицу входит содержание объектного файла findMedian.o

**Создание make-файлов**

Используя примеры с сайта курса, было написано 2 make-файла.

Изображение выглядит как текст

Автоматически созданное описание

Рис 16. Содержание файла Makelib

Изображение выглядит как текст

Автоматически созданное описание

Рис 17. Содержание файла Makefile

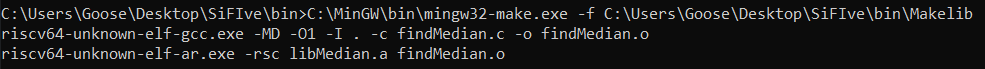


Рис.18. Выполнение файла Makelib

**Вывод**

В ходе выполнения лабораторной работы, была сделана функция и тестирующая ее функция на языке С для нахождения медианны массива. Далее была выполнена сборка по шагам для RISC-V. Была создана библиотека libMedian.a, а также make-файлы для её сборки и сборки тестовой программы с использованием библиотеки.