Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчёт по лабораторной работе**

**Дисциплина**: Низкоуровневое программирование

**Тема**: Раздельная компиляция

Вариант: 6

Выполнил студент гр. 5130901/20003 Гусев М.М.

(подпись)

Преподаватель Целищева А.А.

(подпись)

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Санкт-Петербург

2023

1. **Задание на работу.**
2. На языке C разработать функцию, реализующую определение медианы массива чисел. Поместить определение функции в отдельный исходный файл, оформить заголовочный файл. Разработать тестовую программу.
3. 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.
4. 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.
5. **Описание алгоритма**

На вход подаётся отсортированный по возрастанию массив целых чисел.

Проверяется длина.

ЕСЛИ длина массива нечётная, ТО она делится на 2 нацело и берётся элемент с полученным индексом, являющийся результатом работы программы.

ИНАЧЕ длина также делится нацело на 2, берётся элемент с полученным индексом, берётся предшествующий ему элемент и находится их среднее арифметическое, являющееся результатом работы программы.

1. **Программа на C**

Файл median.c:

#include "median.h"

double median(int array[], int array\_len) {

double res = 0.0;

if(array\_len % 2 != 0) {

res = (double) array[array\_len / 2];

} else {

res =((double) array[array\_len / 2 - 1] + array[array\_len / 2]) / 2.0;

}

return res;

}

Заголовочный файл median.h:

#ifndef LAB4\_CODE\_CLION\_MEDIAN\_H

#define LAB4\_CODE\_CLION\_MEDIAN\_H

double median( int array[], int array\_len);

#endif //LAB4\_CODE\_CLION\_MEDIAN\_H

Файл тестовой программы main.c:

#include <stdio.h>

#include "median.h"

int main() {

int a[] = {0,1, 2, 3, 4, 5, 6, 7, 8, 9};

int len\_a = sizeof(a) / sizeof(a[0]);

printf("Array: ");

for (int i = 0; i < len\_a; i++) {

printf("%d ", a[i]);

}

double result = median(a, len\_a);

if ((int)result == result) {

printf("Median: %d\n", (int)result);

} else {

printf("Median: %.1f\n", result);

}

return 0;

}

Результат выполнения программы в IDE:

Изображение выглядит как текст, снимок экрана, Шрифт, Мультимедийное программное обеспечение

Автоматически созданное описаниеРис. 1 Результат выполнения программы на C в IDE

1. **Сборка «по шагам»**
2. Препроцессирование:

Выполняется препроцессинг исходного файла «main.c» с умеренной оптимизацией. Для этого используется команда:

*riscv64-unknown-elf-gcc.exe -O1 -E main.c -o main.i.,*

Результат сохраняется в man.i. Этот файл содержит код после обработки препроцессором до фактической компиляции и сборки. Из-за подключения модуля <stdio.h> файл «main.i» содержит порядка 1300 строк кода, поэтому не будет приводиться полностью.

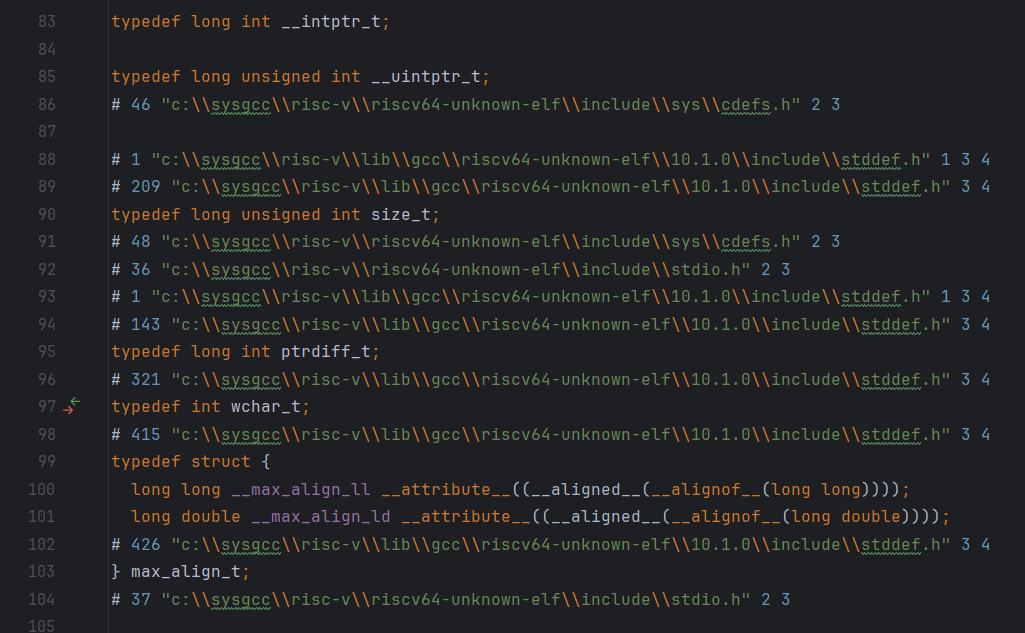


Рис. 2 Фрагмент файла «main.i»

Аналогичная операция производится с файлом «median.c». Содержимое файла «median.i» почти идентично оригинальному файлу .c.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описаниеРис. 3 Файл «median.i»

1. Компиляция

Вызывается компилятор для создания файла на языке ассемблера RISC-V из файла, созданного препроцессором. Для этого используется команда:

*riscv64-unknown-elf-gcc.exe -O1 -S main.i -o main.s,*

В файле «main.s» можно заметить вызов подпрограммы median.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рис. 4 Фрагмент файла «main.s»

Аналогичная команда выполняется для файла «median.s».

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рис. 5 Фрагмент файла «median.s»

1. Ассемблирование

Производится ассемблирование файла. На этом этапе исходный код на языке ассемблера транслируется в машинный код. Команда, с помощью которой выполняется ассемблирование:

*riscv64-unknown-elf-gcc.exe -v -c main.s -o main.o,*

Открыть полученный файлы в текстовом редакторе, как предыдущие, не получится. Для анализа используем следующие команды:

* *riscv64-unknown-elf-objdump.exe -h main.o* **–** заголовки секций

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описаниеРис. 6 Заголовки секций файла «main.o»

* *riscv64-unknown-elf-objdump.exe -t main.o* – таблица символов

Изображение выглядит как текст, снимок экрана, Шрифт, меню

Автоматически созданное описание

Рис. 7 Таблица символов файла «main.o»

Таблица символов файла main.o содержит информацию о символах, присутствующих в объектном файле. Например, имеется запись: символ «median» типа \*UND\*. Она обозначает, что символ «median» использовался в ассемблерном коде, но не был определён ассемблер сделал вывод о том, что символ должен быть определен где-то еще, и отразил это в таблице символов. То же самое относится и к символу «printf».

* *riscv64-unknown-elf-objdump.exe -r main.o* – таблица перемещений

Изображение выглядит как снимок экрана, текст

Автоматически созданное описание

Рис. 8 Таблица перемещений файла «main.o»

Аналогичные действия производятся с файлом «median.o»

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 9 Заголовки секций файла «median.o»

Из рис. 9 видно, что секция .text хранящая исполняемый код программ занимает 4а16= 74 байта, а секции .data и .bss не имеют информации.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 10 Таблица символов файла «median.o»

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 11 Таблица перемещений файла «median.o»

Таблица перемещений содержит информацию о том, какие адреса в исполняемом коде должны быть скорректированы компоновщиком в процессе связывания. Релокации указывают на конкретные места в коде, где необходимо вставить абсолютные адреса или относительные смещения, т.к. они могут быть известны только на этапе связывания.

1. Компоновка

Команда для компоновки файлов:

*riscv64-unknown-elf-gcc.exe -v main.o median.o -o programm.*

Компонует перечисленные объектные файлы, в один исполняемый, по умолчанию называющийся a.out.

С помощью команды

r*iscv64-unknown-elf-objdump.exe -j .text -d programm > programmCode.ds*

файл «programm» дизассемблируется в файл «programmCode.ds».

Этот файл содержит около 18 тысяч строк, поэтому приводится лишь его фрагмент.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рис. 12 Фрагмент файла «programmCode.ds»

1. **Создание статической библиотеки и make-файлов**

Статическая библиотека создаётся следующей командой

*riscv64-unknown-elf-ar.exe -rsc medainLib.a median.o.*

Исполняемый файл программы собирается из получившейся библиотеки «medianLib.a» и файла «main.c» командой

*riscv64-unknown-elf-gcc.exe -O1 --save-temps main.c medianLib.a -o runningFile.*

С помощью уже знакомой команды вызывается таблица символов полученного файла.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 13 Фрагмент таблицы символов файла «runningFile»

Видны знакомые названия файлов («main.c» и «median.c»).

Процесс выполнения всех этих команд можно заменить make-файлом, который произведёт создание библиотеки и сборку программы.

Для этого создаётся файл «MakeFile» (без расширения). Заполняется следующим образом:

output: main.o medianLib.a

gcc main.o medianLib.a -o output

main.o: main.c

gcc -c main.c

medianLib.a: median.o median.h

ar -rsc medianLib.a median.o

median.o:

gcc -c median.c

clean:

del \*.o \*.a output

Далее в терминале выполняется команда «make». Создался файл «output.exe». Проверка что всё работает как надо (в терминале):

*.\output.exe*

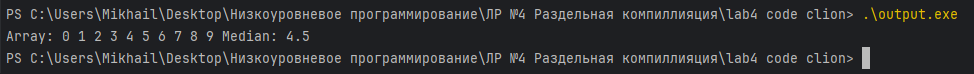


Рис. 14 Результат выполнения файла «output.exe»

1. **Вывод**

В ходе работы была рассмотрена пошаговая компиляция программы на языке C, отдельная компиляция библиотеки, проанализированы все шаги компиляции. Дополнительно, были рассмотрены Make-файлы для автоматической сборки и компиляции кода библиотеки и основной программы.