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. 3530901/10003 Фаретдинов Р.А.

(подпись)

Преподаватель Коренев Д. А. (подпись)

“ ”

Санкт-Петербург

2022

Оглавление

1. ТЗ
2. Метод решения
3. Руководство программисту
4. Реализация программы
5. Работа программы
6. Реализация программы
7. Работа программы

**1. Техническое задание**

Найти минимальный элемент массива

**2. Метод решения**

Изначально минимальным элементом считается первый элемент массива, который в последствие сравнивается с остальными

**3. Руководство программисту**

Исходные данные: массив целых чисел без знака

Результат: минимальный элемент массива

Код представлен двумя программами тестовой (main) и основной (findMin). В main происходит инициализация данных и передача аргументов в функцию findMin. Среди аргументов ссылка на массив и длинна массива.

**4.Реализация программы**

**Текст программы на языке C:**

main.c

Изображение выглядит как текст

Автоматически созданное описание

findMin.c

Изображение выглядит как текст, монитор, экран, снимок экрана

Автоматически созданное описание

findMin.h

Изображение выглядит как текст

Автоматически созданное описание

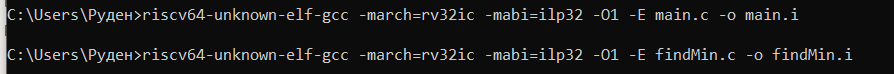
Соберем программу по шагам

**Этап процессирования:**

**riscv64-unknown-elf-gcc -march=rv32ic -mabi=ilp32 -O1 -E findMin.c -o findMin.i**

**riscv64-unknown-elf-gcc -march=rv32ic -mabi=ilp32 -O1 -E main.c -o main.i**

**Результат работы препроцессора**



В программе используются директивы #include<stdio.h> и #include<stddef.h>, поэтому в начале файла main.i содержится строки с инструкциями по присоединению stddef.h и stdio.h к проекту, а затем идет наш код на C. В файле findMin.c тоже есть строки по присоединению stddef.h

Main.i

Изображение выглядит как текст, снимок экрана, экран

Автоматически созданное описание

findMin.i

Изображение выглядит как текст, экран, снимок экрана

Автоматически созданное описание

Как видно, результат работы препроцессора не сильно различается с исходным кодом программы

**Этап компиляции**

При помощи следующих команд получены файлы, сформированные компилятором:

riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 –S main.i -o main.s

riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 –S findMin.i -o findMin.s

**Результат работы компилятора**

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Код из файла main.s

Изображение выглядит как текст

Автоматически созданное описание

Код из файла findMin.s

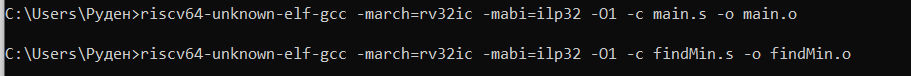
**Этап ассемблирования**

Команды для ассемблирования:

riscv64-unknown-elf-gcc -march=rv32ic -mabi=ilp32 -c main.s -o main.o

riscv64-unknown-elf-gcc -march=rv32ic -mabi=ilp32 -c findMin.s -o findMin.o

**Результат работы ассемблера**



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

Для вывода заголовков секций, таблиц символов, таблица перемещений:

riscv64-unknown-elf-objdump.exe -h main.o (См. рис.1)

riscv64-unknown-elf-objdump.exe -t main.o (См.рис.2)

riscv64-unknown-elf-objdump.exe -r main.o (См.рис.3 )

riscv64-unknown-elf-objdump.exe -h findMin.o (См.рис.4)

riscv64-unknown-elf-objdump.exe -t findMin.o (См.рис. 5)

riscv64-unknown-elf-objdump.exe -r findMin.o (См.рис.6)

Изображение выглядит как текст

Автоматически созданное описание

Рис.1 заголовки секций main.o

В файлах “main.o” и “findMin.o” имеются следующие секции:

.text – секция кода, в которой содержатся коды инструкций (название секции обусловлено историческими причинами);

.data – секция инициализированных данных;

.bss – секция неинициализированных статических переменных (название секции также обусловлено историческими причинами);

.rodata – аналог .data для неизменяемых данных

.comment – секция данных о версиях размером 12 байт

.riscv.attributes – информация про RISC-V

Изображение выглядит как текст

Автоматически созданное описание

Рису.2 таблица символов main.o

В таблицах находится по одному глобальному символу (флаг g) и по одному символу типа функция (F)

В таблице символов файла main.o есть два неопределенных символа (\*UND\*)

Изображение выглядит как текст

Автоматически созданное описание

Рис.3 таблица перемещений main.o

В файле “main.o” имеется записи,например, относящиеся к адресу 24 (по этому адресу в “main.o” находится первая инструкция пары auipc+jalr).

Изображение выглядит как текст

Автоматически созданное описание

Рис.4 заголовки секций findMin.o

Изображение выглядит как текст

Автоматически созданное описание

Рис.5 таблица символов findMin.o

Изображение выглядит как текст

Автоматически созданное описание

Рис.6 таблица перемещений findMin.o

Дизассемблирование и вывод таблицы перемещений можно совместить:

**riscv64-unknown-elf-objdump.exe -d -M no-aliases -r main.o**

Изображение выглядит как текст

Автоматически созданное описание

Таблица перемещений содержит информацию о переходах (R\_RISCV\_JUMP), ветвлениях (R\_RISCV\_BRANCH), вызовах (R\_RISCV\_CALL) и оптимизации (R\_RISCV\_RELAX)

Для того чтобы внести необходимые исправления, требуется знать, что исправить, как исправить и какой символ следует использовать, именно эта информация и содержится в записях о перемещениях. Так, в таблице перемещений указано, что по адресу 24 следует исправить пару инструкций (тип перемещения “R\_RISCV\_CALL”) так, чтобы результат соответствовал вызову подпрограммы “findMin ”. Типы перемещений специфичны для каждой архитектуры системы команд и обычно определены в ABI (Application Binary Interface). Следующая запись таблицы перемещений специфична для средств разработки RISC-V. 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.

**Выполнение компоновки:**

Компоновка выполняется при помощи команды:

**riscv64-unknown-elf-gcc -march=rv32ic -mabi=ilp32 main.o findMin.o -o main**

Изучим содержимое секции “.text” полученного в результате компоновки программы исполняемого файла:

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

Изображение выглядит как текст, табличка, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Видно, что секций стало больше, по сравнению с объектным файлом, а также увеличились их размеры

Прежде всего можно видеть, что в результат компоновки попало содержимое обоих объектных файлов – “main.o” и “findMin.o”. Инструкции подпрограммы “findMin” начинаются с адреса 1014c.

**статическая библиотека**

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.

Поместим findMin.o в статическую библиотеку lib:

**riscv64-unknown-elf-ar -rsc lib.a findMin.o**

Параметры:

-r – заменить старые файлы с такими названиями (selection.o), если они уже есть в архиве;

-s – записать «index» в архив. Index – это список всех символов, объявленных во включенных в архив объектных файлах, и его присутствие ускоряет линковку;

-с – создать архив, если его еще не было.

При помощи библиотеки соберем исполняемый файл:

**riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -O1 --save-temps main.c lib.a**

Таблица символов исполняемого файла (фрагменты): Команда:

**riscv64-unknown-elf-objdump.exe -t a.out**







Видно, что объектные файлы вошли в состав программы.

Проверим символьные таблицы:

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



Таблица main.o

**riscv64-unknown-elf-objdump -t filling.o**



Так как имеются неразрешенные символы, то компоновщик будет искать их (сам) в подключенной библиотеке.

Изображение выглядит как текст

Автоматически созданное описание

Список символов библиотеки lib

**Вывод**

В ходе выполнения лабораторной работы были получены навыки программирования на языке C, получены навыки пошаговой сборки программ и навыки работы с препроцессором, компилятором, ассемблером и компоновщиком. Был изучен каждый этап пошаговой сборки программ. Также были изучены инструменты, позволяющие выделить разработанные программы в статическую библиотеку.