Санкт-Петербургский политехнический университет Петра Великого

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчёт по лабораторной работе № 4**

**Дисциплина**: Низкоуровневое программирование

**Тема**: Раздельная компиляция

Выполнила

студент гр.3530901/10005 Вилисова Д. Д.

Преподаватель Коренев Д. А.

“ ”

Санкт-Петербург

2022

**Содержание:**

1. **Формулировка задачи**
2. **Вариант задания**
3. **Текст программы**
4. **Сборка программ «по шагам», анализ промежуточных и результирующих файлов**
5. **Создание статической библиотеки**
6. **Вывод**
7. **Формулировка задачи**

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.

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.

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.

1. **Вариант задания**

Разработать программу, реализующую сортировку выбором массива чисел in-place.

1. **Текст программы**

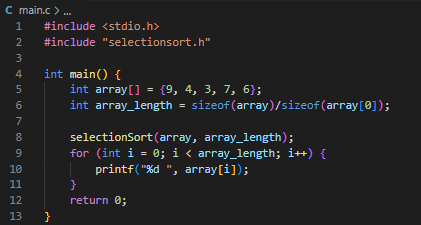


Рис. 1. Реализация тестовой программы main.c

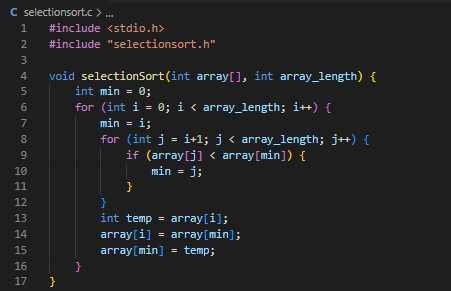


Рис. 2. Реализация функциональной части программы selectionsort.c

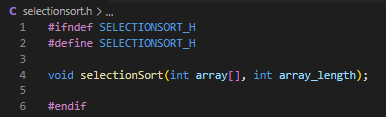


Рис. 3. Заголовочный файл selectionsort.h

В файле selectionsort.c реализована функция:

selectionSort(int array[], int array\_length), которая принимает два аргумента массив целых чисел и его длину.

В функции реализован алгоритм сортировки выбором массива чисел: за каждый проход по массиву выбирается минимальный элемент (для сортировки по возрастанию) и происходит обмен его местами с первым элементом в еще не отсортированном участке массива. Тем самым уменьшив длину этого участка на один, и так до тех пор, пока не будут отсортированы все элементы.

Заголовочный файл selectionsort.h содержит в себе определение функции selectionSort(). В дальнейшем для использования этой функции в другой программе, необходимо организовывать подключение этого заголовочного файла, а также компиляцию исходного файла selectionsort.c вместе с использующей ее программой.

1. **Сборка программ «по шагам», анализ промежуточных и результирующих файлов**

Начнем сборку созданных программ на языке C по шагам. Первым шагом является препроцессирование файлов исходного текста selectionsort.c и main.c в файлы main.i и selectionsort.i. Для препроцессирования применяются команды:

riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -O1 -E main.c -o main.i

riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -O1 -E selectionsort.c -o selectionsort.i

Параметры:

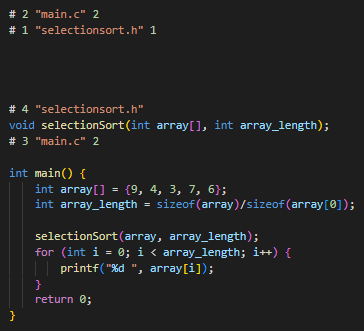
-march=rv32ic -mabi=ilp32 – целевым является процессор с базовой архитектурой системы команд RV32I;

-O1 – выполнять простые оптимизации генерируемого кода (мы используем эту опцию в примерах, потому что обычно генерируемый код получается более простым);

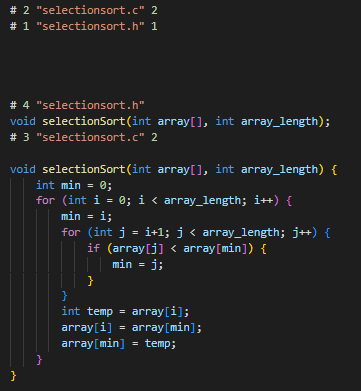
-E – остановить процесс сборки после препроцессирования;

-o – путь к выходному файлу.

Результаты работы препроцессора мало отличаются от исходных версий программ:



Результат препроцессирования main.i



Результат препроцессирования selectionsort.i

Появившиеся нестандартные директивы, начинающиеся с символа “#”, используются для передачи информации об исходном тексте из препроцессора в компилятор.

**Компиляция**

Далее необходимо выполнить компиляцию файлов main.i и selectionsort.i в код на языке ассемблера main.s и selectionsort.s. Для компилирования выполняются команды:

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

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

Параметры:

-S – остановить процесс сборки после компиляции, не запуская ассемблер;

-march=rv32ic -mabi=ilp32 – целевым является процессор с базовой архитектурой системы команд RV32I;

-O1 – выполнять простые оптимизации генерируемого кода (мы используем эту опцию в примерах, потому что обычно генерируемый код получается более простым);

-o – путь к выходному файлу.

**Результат компилирования main.s:**

.file "main.c"

.option nopic

.attribute arch, "rv32i2p0"

.attribute unaligned\_access, 0

.attribute stack\_align, 16

.text

.align 2

.globl main

.type main, @function

main:

addi sp,sp,-48

sw ra,44(sp)

sw s0,40(sp)

sw s1,36(sp)

sw s2,32(sp)

lui a5,%hi(.LANCHOR0)

addi a5,a5,%lo(.LANCHOR0)

lw a1,0(a5)

lw a2,4(a5)

lw a3,8(a5)

lw a4,12(a5)

lw a5,16(a5)

sw a1,12(sp)

sw a2,16(sp)

sw a3,20(sp)

sw a4,24(sp)

sw a5,28(sp)

li a1,5

addi a0,sp,12

call selectionSort

addi s0,sp,12

addi s2,sp,32

lui s1,%hi(.LC1)

.L2:

lw a1,0(s0)

addi a0,s1,%lo(.LC1)

call printf

addi s0,s0,4

bne s0,s2,.L2

li a0,0

lw ra,44(sp)

lw s0,40(sp)

lw s1,36(sp)

lw s2,32(sp)

addi sp,sp,48

jr ra

.size main, .-main

.section .rodata

.align 2

.set .LANCHOR0,. + 0

.LC0:

.word 9

.word 4

.word 3

.word 7

.word 6

.section .rodata.str1.4,"aMS",@progbits,1

.align 2

.LC1:

.string "%d "

.ident "GCC: (SiFive GCC 8.3.0-2020.04.1) 8.3.0"

По метке main начинается тело самой функции. Также можно заметить обращение к подпрограмме selectionSort.

**Результат компилирования selectionsort.s:**

.file "selectionsort.c"

.option nopic

.attribute arch, "rv32i2p0"

.attribute unaligned\_access, 0

.attribute stack\_align, 16

.text

.align 2

.globl selectionSort

.type selectionSort, @function

selectionSort:

ble a1,zero,.L1

mv a7,a0

li a2,0

j .L6

.L3:

addi a4,a4,1

addi a3,a3,4

beq a1,a4,.L8

.L4:

slli a5,a2,2

add a5,a0,a5

lw a6,4(a3)

lw a5,0(a5)

bge a6,a5,.L3

mv a2,a4

j .L3

.L8:

lw a5,0(a7)

slli a2,a2,2

add a2,a0,a2

lw a4,0(a2)

sw a4,0(a7)

sw a5,0(a2)

addi a7,a7,4

mv a2,t1

.L6:

addi t1,a2,1

beq a1,t1,.L1

mv a3,a7

mv a4,t1

j .L4

.L1:

ret

.size selectionSort, .-selectionSort

.ident "GCC: (SiFive GCC 8.3.0-2020.04.1) 8.3.0"

**Ассемблирование**

Ассемблирование файлов “main.s” и “adder.s” выполняется по следующим командам:

riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -c main.s -o main.o

riscv64-unknown-elf-gcc.exe -march=rv32i -mabi=ilp32 -c selectionsort.s -o selectionsort.o

Параметры:

-c – остановить процесс сборки после ассемблирования.

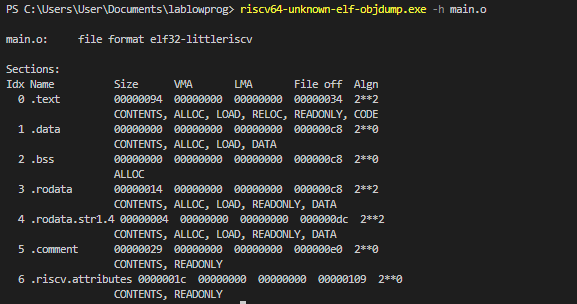
Остальные параметры такие же, как и в прошлых пунктах.

В результате получили два объектных файла main.o и selectionsort.o, которые не являются текстовыми. Объектные файлы содержат коды инструкций, таблицу символов и таблицу перемещений. Для изучения содержимого используем утилиту objdump, отображающую содержимое бинарных файлов в текстовом виде. Содержательная часть объектных файлов разбита на разделы – секции (section), используем команды для отображения заголовков секций:

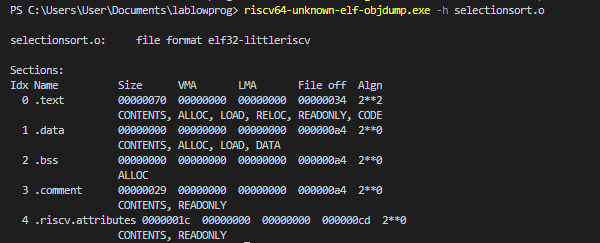
riscv64-unknown-elf-objdump.exe -h main.o

riscv64-unknown-elf-objdump.exe -h selectionsort.o

**Результаты main.o:**

****

**Результаты selectionsort.o:**

****

Обозначение секций: .text – секция кода, в которой содержатся коды инструкций;

.data - секция инициализированных данных;

.bss – секция неинициализированных статических переменных;

.rodata – аналог .data для неизменяемых данных;

.comment – секция данных о версиях размером 12 байт;

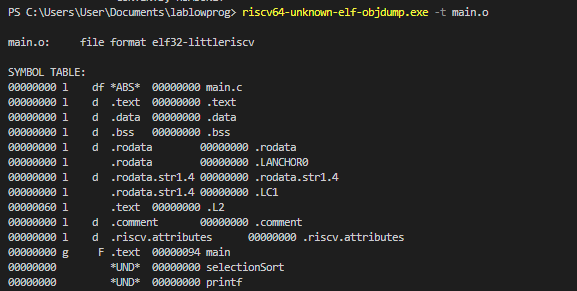
.riscv.attributes – информация про RISC-V

Для получения таблицы символов используем команды:

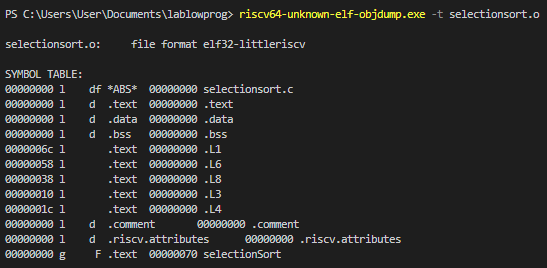
riscv64-unknown-elf-objdump.exe -t main.o

riscv64-unknown-elf-objdump.exe -t selectionsort.o

**Результаты main.o:**

****

**Результаты selectionsort.o:**

****

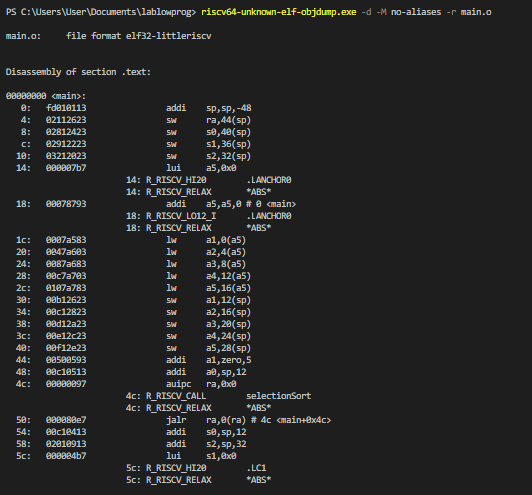
В каждой таблице только один глобальный (флаг “g”) символ типа «функция» (“F”) – “main” и “selectionSort” соответственно.

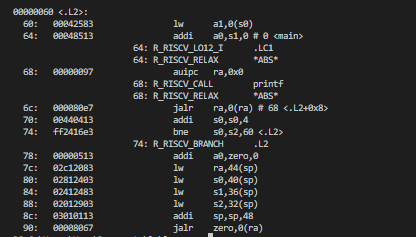
Для получения таблицы перемещений используем команды:

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

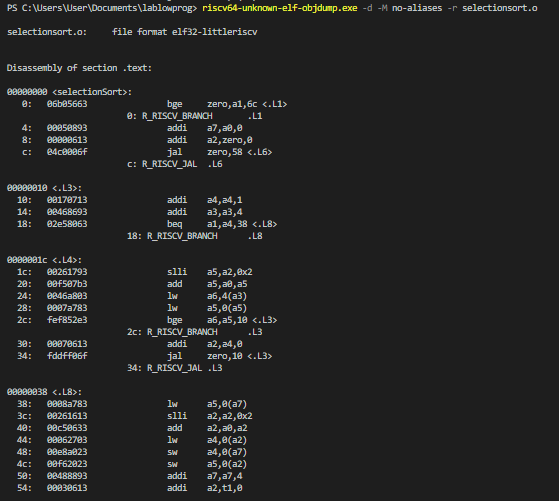
riscv64-unknown-elf-objdump.exe -d -M no-aliases -r selectionsort.o

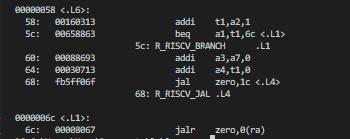
**Результаты main.o:**

****

****

**Результаты selectionsort.o:**

****

****

Дизассемблированный код практически идентичен сгенерированному (за исключением псевдоинструкций).

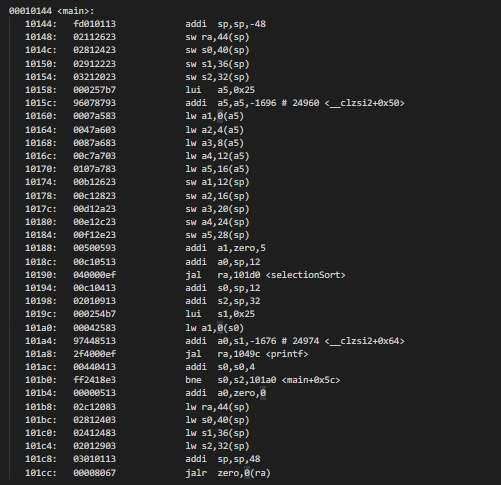
**Компоновка**

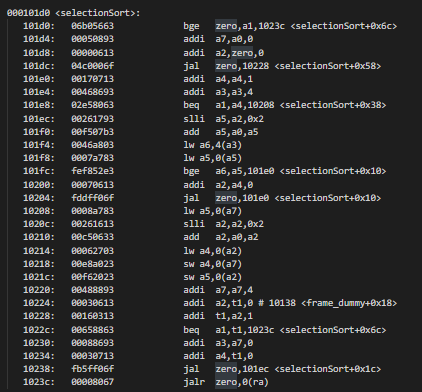
Компоновка программы выполняется командой:

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

Результатом является исполняемый файл a.out (бинарный). Изучим содержимое секции .text полученного в результате компоновки исполняемого файла, используя команду:

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

****

****

Итогом сборки программ на языке C по шагам является исполняемый на процессорах архитектуры RISC-V файл, решающий задачу сортировки выбором массива чисел и проверяющий корректность решения этой задачи на тестовом примере.

1. **Создание статической библиотеки**

Статическая библиотека является, по сути, архивом объектных файлов. Разработанная функция сортировки выбором массива чисел содержится в единственном исходном файле на языке C. Выделим этот файл selectionsort.o в статическую библиотеку sortlib.a:

riscv64-unknown-elf-ar -rsc sortlib.a selectionsort.o

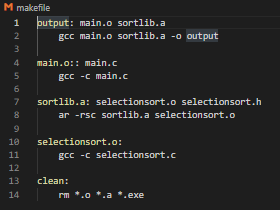
Параметры:

-r – заменить старые файлы с такими названиями (adder.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.

Используем статическую библиотеку для сборки программ. Напишем makefile – набор инструкций для программы make, которая помогает собирать программу из файлов в один вызов make:



Что происходит в Makefile:

1. Создаём объектный файл main.o из исходного main.c

2. Создаём объектный файл selectionsort.o из исходного selectionsort.c

3. Архивируем объектный файл selectionsort.o (создаём статическую библиотеку sortlib.a)

4. Компонуем статическую библиотеку sortlib.a с объектным файлом main.o и получаем исполняемый файл output

Запуск makefile и процесс сборки:



Результат работы exe файла для массива {9, 4, 3, 7, 6}:



Массив отсортирован.

1. **Вывод**

В ходе выполнения данной лабораторной работы была разработана функция на языке C, реализующая заданную вариантом задания функциональность: сортировка выбором массива чисел. Была проделана сборка программы: препроцессирование, компиляция, ассемблирование и компоновка. Была создана статическая библиотека и автоматизирована сборка программы с помощью makefile.