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

На языке С разработать функцию, реализующую определенную вариантом задания функциональность. Поместить определение функции в отдельный исходный файл, оформить заголовочный файл. Разработать тестовую программу на языке С.

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.

2. Вариант задания

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

3. Текст программы

```c
#include <stdio.h>
#include "selectionsort.h"

int main(void) {
    unsigned int array[] = {9, 4, 3, 7, 6};
    size_t array_length = sizeof(array)/sizeof(array[0]);

    selectionSort(array, array_length);
    for (size_t i = 0; i < array_length; i++) {
        printf("%d, ", array[i]);
    }
    return 0;
}
```

Рис. 1. Реализация тестовой программы main.c

```c
#include <stdio.h>
#include "selectionsort.h"

void selectionSort(unsigned int array[], size_t array_length) {
    size_t min = 0;
    for (size_t i = 0; i < array_length; i++) {
        min = i;
        for (size_t j = i+1; j < array_length; j++) {
            if (array[j] < array[min]) {
                min = j;
            }
        }
        unsigned int temp = array[i];
        array[i] = array[min];
        array[min] = temp;
    }
}
```

Рис. 2. Реализация функциональной части программы selectionsort.c

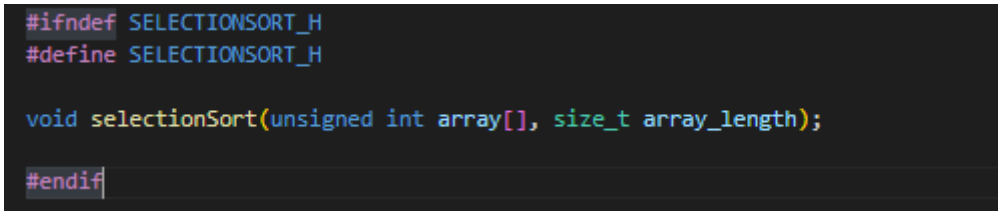


Рис. 3. Заголовочный файл selectionsort.h

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.

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

Начнем сборку созданных программ на языке С 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:

```
# 4 "selectionsort.h"
void selectionSort(unsigned int array[], size_t array_length);
# 3 "main.c" 2

int main(void) {
    unsigned int array[] = {9, 4, 3, 7, 6};
    size_t array_length = sizeof(array)/sizeof(array[0]);

    selectionSort(array, array_length);
    for (size_t i = 0; i < array_length; i++) {
        printf("%d, ", array[i]);
    }
    return 0;
}
```

Результат препроцессирования main.i

```
# 4 "selectionsort.h"
void selectionSort(unsigned int array[], size_t array_length);
# 3 "selectionsort.c" 2

void selectionSort(unsigned int array[], size_t array_length) {
    size_t min = 0;
    for (size_t i = 0; i < array_length; i++) {
        min = i;
        for (size_t j = i+1; j < array_length; j++) {
            if (array[j] < array[min]) {
                min = j;
            }
        }
        unsigned int temp = array[i];
        array[i] = array[min];
        array[min] = temp;
    }
}
```

Результат препроцессирования selectionsort.i

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"
```

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:
        beq     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)
        bgeu    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:

```
main.o:     file format elf32-littleriscv

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000094  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000c8  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000c8  2**0
                  ALLOC
  3 .rodata       00000014  00000000  00000000  000000c8  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .rodata.str1.4 00000005  00000000  00000000  000000dc  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .comment      00000029  00000000  00000000  000000e1  2**0
                  CONTENTS, READONLY
  6 .riscv.attributes 0000001c  00000000  00000000  0000010a  2**0
                  CONTENTS, READONLY
```

Результаты selectionsort.o:

```
selectionsort.o:     file format elf32-littleriscv

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000070  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000a4  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000a4  2**0
                  ALLOC
  3 .comment      00000029  00000000  00000000  000000a4  2**0
                  CONTENTS, READONLY
  4 .riscv.attributes 0000001c  00000000  00000000  000000cd  2**0
                  CONTENTS, READONLY
```

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:

```
main.o:     file format elf32-littleriscv

SYMBOL TABLE:
00000000 l    df *ABS*  00000000 main.c
00000000 l    d  .text  00000000 .text
00000000 l    d  .data  00000000 .data
00000000 l    d  .bss   00000000 .bss
00000000 l    d  .rodata        00000000 .rodata
00000000 l       .rodata        00000000 .LANCHOR0
00000000 l    d  .rodata.str1.4 00000000 .rodata.str1.4
00000000 l       .rodata.str1.4 00000000 .LC1
00000060 l       .text  00000000 .L2
00000000 l    d  .comment       00000000 .comment
00000000 l    d  .riscv.attributes      00000000 .riscv.attributes
00000000 g     F .text  00000094 main
00000000         *UND*  00000000 selectionSort
00000000         *UND*  00000000 printf
```

Результаты selectionsort.o:

```
selectionsort.o:     file format elf32-littleriscv

SYMBOL TABLE:
00000000 l    df *ABS*  00000000 selectionsort.c
00000000 l    d  .text  00000000 .text
00000000 l    d  .data  00000000 .data
00000000 l    d  .bss   00000000 .bss
0000006c l       .text  00000000 .L1
00000058 l       .text  00000000 .L6
00000038 l       .text  00000000 .L8
00000010 l       .text  00000000 .L3
0000001c l       .text  00000000 .L4
00000000 l    d  .comment       00000000 .comment
00000000 l    d  .riscv.attributes      00000000 .riscv.attributes
00000000 g     F .text  00000070 selectionSort
```

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

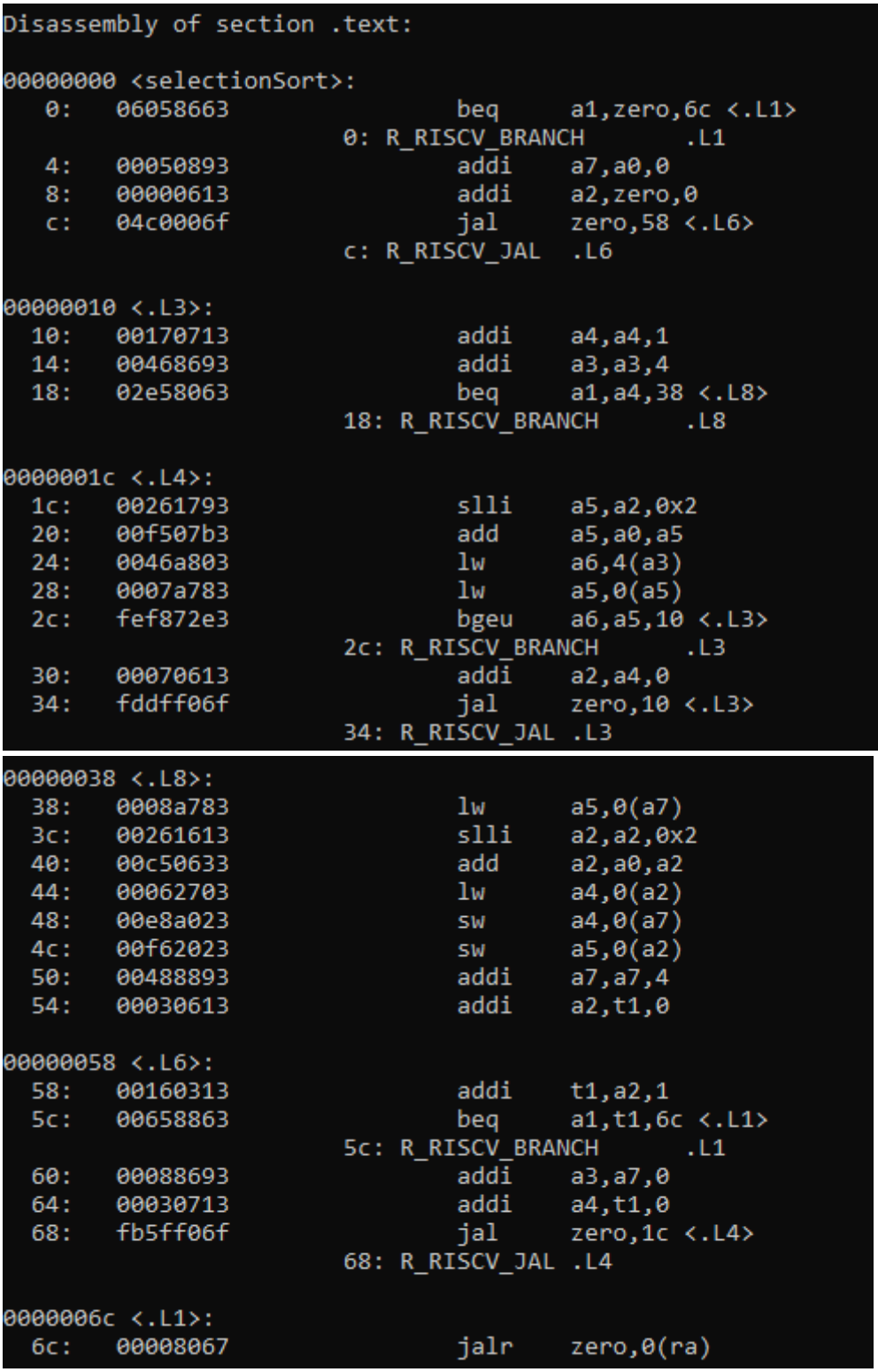
Результаты main.o:

```
Disassembly of section .text:

00000000 <main>:
   0:   fd010113                addi    sp,sp,-48
   4:   02112623                sw      ra,44(sp)
   8:   02812423                sw      s0,40(sp)
   c:   02912223                sw      s1,36(sp)
  10:   03212023                sw      s2,32(sp)
  14:   000007b7                lui     a5,0x0
                        14: R_RISCV_HI20        .LANCHOR0
                        14: R_RISCV_RELAX       *ABS*
  18:   00078793                addi    a5,a5,0 # 0 <main>
                        18: R_RISCV_LO12_I      .LANCHOR0
                        18: R_RISCV_RELAX       *ABS*
  1c:   0007a583                lw      a1,0(a5)
  20:   0047a603                lw      a2,4(a5)
  24:   0087a683                lw      a3,8(a5)
  28:   00c7a703                lw      a4,12(a5)
  2c:   0107a783                lw      a5,16(a5)
  30:   00b12623                sw      a1,12(sp)
  34:   00c12823                sw      a2,16(sp)
  38:   00d12a23                sw      a3,20(sp)
  3c:   00e12c23                sw      a4,24(sp)
  40:   00f12e23                sw      a5,28(sp)
  44:   00500593                addi    a1,zero,5
  48:   00c10513                addi    a0,sp,12
  4c:   00000097                auipc   ra,0x0
                        4c: R_RISCV_CALL        selectionSort
                        4c: R_RISCV_RELAX       *ABS*
  50:   000080e7                jalr    ra,0(ra) # 4c <main+0x4c>
  54:   00c10413                addi    s0,sp,12
  58:   02010913                addi    s2,sp,32
  5c:   000004b7                lui     s1,0x0
                        5c: R_RISCV_HI20        .LC1
                        5c: R_RISCV_RELAX       *ABS*
```

```
00000060 <.L2>:
  60:   00042583                lw      a1,0(s0)
  64:   00048513                addi    a0,s1,0 # 0 <main>
                        64: R_RISCV_LO12_I      .LC1
                        64: R_RISCV_RELAX       *ABS*
  68:   00000097                auipc   ra,0x0
                        68: R_RISCV_CALL        printf
                        68: R_RISCV_RELAX       *ABS*
  6c:   000080e7                jalr    ra,0(ra) # 68 <.L2+0x8>
  70:   00440413                addi    s0,s0,4
  74:   ff2416e3                bne     s0,s2,60 <.L2>
                        74: R_RISCV_BRANCH      .L2
  78:   00000513                addi    a0,zero,0
  7c:   02c12083                lw      ra,44(sp)
  80:   02812403                lw      s0,40(sp)
  84:   02412483                lw      s1,36(sp)
  88:   02012903                lw      s2,32(sp)
  8c:   03010113                addi    sp,sp,48
  90:   00008067                jalr    zero,0(ra)
```

Результаты selectionsort.o:



```
Disassembly of section .text:

00000000 <selectionSort>:
   0:   06058663                beq     a1,zero,6c <.L1>
                        0: R_RISCV_BRANCH       .L1
   4:   00050893                addi    a7,a0,0
   8:   00000613                addi    a2,zero,0
   c:   04c0006f                jal     zero,58 <.L6>
                        c: R_RISCV_JAL  .L6

00000010 <.L3>:
  10:   00170713                addi    a4,a4,1
  14:   00468693                addi    a3,a3,4
  18:   02e58063                beq     a1,a4,38 <.L8>
                        18: R_RISCV_BRANCH      .L8

0000001c <.L4>:
  1c:   00261793                slli    a5,a2,0x2
  20:   00f507b3                add     a5,a0,a5
  24:   0046a803                lw      a6,4(a3)
  28:   0007a783                lw      a5,0(a5)
  2c:   fef872e3                bgeu    a6,a5,10 <.L3>
                        2c: R_RISCV_BRANCH      .L3
  30:   00070613                addi    a2,a4,0
  34:   fddff06f                jal     zero,10 <.L3>
                        34: R_RISCV_JAL .L3

00000038 <.L8>:
  38:   0008a783                lw      a5,0(a7)
  3c:   00261613                slli    a2,a2,0x2
  40:   00c50633                add     a2,a0,a2
  44:   00062703                lw      a4,0(a2)
  48:   00e8a023                sw      a4,0(a7)
  4c:   00f62023                sw      a5,0(a2)
  50:   00488893                addi    a7,a7,4
  54:   00030613                addi    a2,t1,0

00000058 <.L6>:
  58:   00160313                addi    t1,a2,1
  5c:   00658863                beq     a1,t1,6c <.L1>
                        5c: R_RISCV_BRANCH      .L1
  60:   00088693                addi    a3,a7,0
  64:   00030713                addi    a4,t1,0
  68:   fb5ff06f                jal     zero,1c <.L4>
                        68: R_RISCV_JAL .L4

0000006c <.L1>:
  6c:   00008067                jalr    zero,0(ra)
```

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

```
00010144 <main>:
   10144:   fd010113            addi  sp,sp,-48
   10148:   02112623            sw ra,44(sp)
   1014c:   02812423            sw s0,40(sp)
   10150:   02912223            sw s1,36(sp)
   10154:   03212023            sw s2,32(sp)
   10158:   000257b7            lui   a5,0x25
   1015c:   96078793            addi  a5,a5,-1696 # 24960 <__clzsi2+0x50>
   10160:   0007a583            lw a1,0(a5)
   10164:   0047a603            lw a2,4(a5)
   10168:   0087a683            lw a3,8(a5)
   1016c:   00c7a703            lw a4,12(a5)
   10170:   0107a783            lw a5,16(a5)
   10174:   00b12623            sw a1,12(sp)
   10178:   00c12823            sw a2,16(sp)
   1017c:   00d12a23            sw a3,20(sp)
   10180:   00e12c23            sw a4,24(sp)
   10184:   00f12e23            sw a5,28(sp)
   10188:   00500593            addi  a1,zero,5
   1018c:   00c10513            addi  a0,sp,12
   10190:   040000ef            jal   ra,101d0 <selectionSort>
   10194:   00c10413            addi  s0,sp,12
   10198:   02010913            addi  s2,sp,32
   1019c:   000254b7            lui   s1,0x25
   101a0:   00042583            lw a1,0(s0)
   101a4:   97448513            addi  a0,s1,-1676 # 24974 <__clzsi2+0x64>
   101a8:   2f4000ef            jal   ra,1049c <printf>
   101ac:   00440413            addi  s0,s0,4
   101b0:   ff2418e3            bne   s0,s2,101a0 <main+0x5c>
   101b4:   00000513            addi  a0,zero,0
   101b8:   02c12083            lw ra,44(sp)
   101bc:   02812403            lw s0,40(sp)
   101c0:   02412483            lw s1,36(sp)
   101c4:   02012903            lw s2,32(sp)
   101c8:   03010113            addi  sp,sp,48
   101cc:   00008067            jalr  zero,0(ra)
```

```
000101d0 <selectionSort>:
   101d0:   06058663            beq   a1,zero,1023c <selectionSort+0x6c>
   101d4:   00050893            addi  a7,a0,0
   101d8:   00000613            addi  a2,zero,0
   101dc:   04c0006f            jal   zero,10228 <selectionSort+0x58>
   101e0:   00170713            addi  a4,a4,1
   101e4:   00468693            addi  a3,a3,4
   101e8:   02e58063            beq   a1,a4,10208 <selectionSort+0x38>
   101ec:   00261793            slli  a5,a2,0x2
   101f0:   00f507b3            add   a5,a0,a5
   101f4:   0046a803            lw a6,4(a3)
   101f8:   0007a783            lw a5,0(a5)
   101fc:   fef872e3            bgeu  a6,a5,101e0 <selectionSort+0x10>
   10200:   00070613            addi  a2,a4,0
   10204:   fddff06f            jal   zero,101e0 <selectionSort+0x10>
   10208:   0008a783            lw a5,0(a7)
   1020c:   00261613            slli  a2,a2,0x2
   10210:   00c50633            add   a2,a0,a2
   10214:   00062703            lw a4,0(a2)
   10218:   00e8a023            sw a4,0(a7)
   1021c:   00f62023            sw a5,0(a2)
   10220:   00488893            addi  a7,a7,4
   10224:   00030613            addi  a2,t1,0 # 10138 <frame_dummy+0x18>
   10228:   00160313            addi  t1,a2,1
   1022c:   00658863            beq   a1,t1,1023c <selectionSort+0x6c>
   10230:   00088693            addi  a3,a7,0
   10234:   00030713            addi  a4,t1,0
   10238:   fb5ff06f            jal   zero,101ec <selectionSort+0x1c>
   1023c:   00008067            jalr  zero,0(ra)
```

Итогом сборки программ на языке С 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.

5. Создание статической библиотеки

Разработанная функция сортировки выбором массива чисел содержится в единственном исходном файле на языке С. 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;

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

Напишем makefile для автоматической сборки проекта:

```
M makefile
1   output: main.o sortlib.a
2       gcc main.o sortlib.a -o output
3
4   main.o:: main.c
5       gcc -c main.c
6
7   sortlib.a: selectionsort.o selectionsort.h
8       ar -rsc sortlib.a selectionsort.o
9
10  selectionsort.o:
11      gcc -c selectionsort.c
12
13  clean:
14      rm *.o *.a *.exe
```

Запуск makefile и процесс сборки:

```
PS C:\Users\User\Desktop\lab4> mingw32-make.exe -f makefile
gcc -c main.c
gcc -c selectionsort.c
ar -rsc sortlib.a selectionsort.o
gcc main.o sortlib.a -o output
```

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

```
PS C:\Users\User\Desktop\lab4> .\output.exe
3, 4, 6, 7, 9,
```

Массив отсортирован.

6. 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.