

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

Вариант 6.

Выполнил студент гр. 3530901/90002 _____ М.В. Дергачев
(подпись)

Принял преподаватель _____ Д.С. Степанов
(подпись)

“ ____ ” _____ 2021 г.

Санкт-Петербург

2021

Задача

1. 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.
2. 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.
3. 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 задания

По варианту номер 6 необходимо реализовать нахождение медианы массива in-place.

Решение

Медиана – элемент массива, который находится ровно посередине после сортировки. Для выполнения поставленной задачи программа проходит по массиву, сравнивая каждое число с каждым, прибавляя к счетчику 1, если второе число больше текущего; вычитая 1, если число меньше текущего и не делая ничего, если они равны. В конце каждого прохода проверяется значение счетчика, если он меньше 0 или больше 1 (количество чисел больше текущего меньше, чем количество чисел меньше текущего и наоборот, соответственно), то программа за место текущего числа берет следующее из массива и начинает проход заново, если значение счетчика равно 0 или 1 (0, когда в массиве нечетное количество элементов и 1, когда - четное), то медиана найдена и программа, записав ее, завершает работу.

1. Написание программы на языке C

Была написана программа, которая находит медиану массива. Функция помещена в отдельный файл, написан заголовочный файл.

```
1  ▾ #ifndef MEDIAN_H
2    #define MEDIAN_H
3
4    int median(unsigned *array, size_t size);
5
6    #endif
```

Рис.1 Заголовочный файл

```
1  #include <stddef.h>
2  #include <stdio.h>
3  #include "median.h"
4
5  static unsigned array[] = {2, 9, 10, 5, 6, 7, 3, 4, 1, 8};
6  static const size_t array_length = sizeof(array) / sizeof(array[0]);
7
8  int main( void )
9  {
10     printf("array: \n");
11     for (size_t i = 0; i < array_length; i++)
12     {
13         printf("%i, ", array[i]);
14     }
15     printf("\n");
16
17     int temp = median(array, array_length);
18     printf("median = %i",temp);
19 }
20
```

Рис.2 Текст программы

```

1  ∨ #include <stddef.h>
2    #include "median.h"
3
4  ∨ int median(unsigned *array, size_t array_length)
5  {
6      int count = 0;
7  ∨  for (size_t i = 0; i < array_length; i++)
8      {
9  ∨      for (size_t j = 0; j < array_length; j++)
10     {
11  ∨         if (array[i] < array[j])
12             {
13                 count++;
14             }
15  ∨         if (array[i] > array[j])
16             {
17                 count--;
18             }
19     }
20  ∨  if ((count == 0) | (count == 1))
21     {
22         return(array[i]);
23         break;
24     }
25     count = 0;
26 }
27     return -1;
28 }
29

```

Рис.3 Функция median

Произведем компиляцию программы:

```

array:
2, 9, 10, 5, 6, 7, 3, 4, 1, 8,
median = 5

```

Рис. 4 Результат исполнения программы

2. Сборка программы «по шагам»

Препроцессирование.

Выполним препроцессирование файлов с помощью пакета разработки «SiFive GNU Embedded Toolchain» для RISK-V. Для этого необходимо выполнить следующие команды:

```
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -E main.c -o main.i  
>log_main_prepr.txt 2>&1
```

```
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -E median.c -o  
median.i >log_median_prepr.txt 2>&1
```

-march=rv64iac -mabi=lp64 – объявляем, что целевым является процессор с базовой архитектурой системы команд RV64IAC

-O1 – выполняются простые оптимизации генерируемого кода

-v – печатаются (в стандартный поток ошибок) выполняемые драйвером команды, а также дополнительную информацию

> - печатать в файл

-E – обработка файлов будет выполняться только препроцессором

Рассмотрим результаты препроцессирования.

```
1  # 1 "main.c"
2  # 1 "<built-in>"
3  # 1 "<command-line>"
4  # 1 "main.c"

1276 # 4 "median.h"
1277 int median(unsigned *array, size_t size);
1278 # 4 "main.c" 2
1279
1280 static unsigned array[] = {2, 9, 10, 5, 6, 7, 3, 4, 1, 8};
1281 static const size_t array_length = sizeof(array) / sizeof(array[0]);
1282
1283 int main( void )
1284 {
1285     printf("array: \n");
1286     for (size_t i = 0; i < array_length; i++)
1287     {
1288         printf("%i, ", array[i]);
1289     }
1290     printf("\n");
1291
1292     int temp = median(array, array_length);
1293     printf("median = %i",temp);
1294 }
1295
```

Рис. 5 Файл main.i

```
1  # 1 "median.c"
2  # 1 "<built-in>"
3  # 1 "<command-line>"
4  # 1 "median.c"

26 # 4 "median.h"
27 int median(unsigned *array, size_t size);
28 # 3 "median.c" 2
29
30 int median(unsigned *array, size_t array_length)
31 {
32     int count = 0;
33     for (size_t i = 0; i < array_length; i++)
34     {
35         for (size_t j = 0; j < array_length; j++)
36         {
37             if (array[i] < array[j])
38             {
39                 count++;
40             }
41             if (array[i] > array[j])
42             {
43                 count--;
44             }
45         }
46         if ((count == 0) | (count == 1))
47         {
48             return(array[i]);
49             break;
50         }
51         count = 0;
52     }
53     return -1;
54 }
55
```

Рис. 6 Файл median.i

Компиляция

Выполним компиляцию файлов с помощью пакета разработки «SiFive GNU Embedded Toolchain» для RISK-V. Для этого необходимо выполнить следующие команды:

```
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -S -fpreprocessed  
median.i -o median.s>log_median_comp.txt 2>&1
```

```
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -S -fpreprocessed  
main.i -o main.s>log_main_comp.txt 2>&1
```

```

main.s ✕
1      .file      "main.c"
2      .option    nopic
3      .attribute arch, "rv64i2p0_a2p0_c2p0"
4      .attribute    unaligned_access, 0
5      .attribute    stack_align, 16
6      .text
7      .section    .rodata.str1.8,"aMS",@progbits,1
8      .align      3
9      .LC0:
10     .string      "array: "
11     .align      3
12     .LC1:
13     .string      "%i, "
14     .align      3
15     .LC2:
16     .string      "median = %i"
17     .text
18     .align      1
19     .globl    main
20     .type      main, @function
21     .main:
22     addi      sp,sp,-32
23     sd        ra,24(sp)
24     sd        s0,16(sp)
25     sd        s1,8(sp)
26     sd        s2,0(sp)
27     lui       a0,%hi(.LC0)
28     addi      a0,a0,%lo(.LC0)
29     call      puts
30     lui       s0,%hi(.LANCHOR0)
31     addi      s0,s0,%lo(.LANCHOR0)
32     addi      s2,s0,40
33     lui       s1,%hi(.LC1)
34     .L2:
35     lw        a1,0(s0)
36     addi      a0,s1,%lo(.LC1)
37     call      printf
38     addi      s0,s0,4
39     bne       s0,s2,.L2
40     li        a0,10
41     call      putchar
42     li        a1,10
43     lui       a0,%hi(.LANCHOR0)
44     addi      a0,a0,%lo(.LANCHOR0)
45     call      median
46     mv        a1,a0
47     lui       a0,%hi(.LC2)
48     addi      a0,a0,%lo(.LC2)
49     call      printf
50     li        a0,0
51     ld        ra,24(sp)
52     ld        s0,16(sp)
53     ld        s1,8(sp)
54     ld        s2,0(sp)
55     addi      sp,sp,32
56     jr        ra
57     .size      main, .-main
58     .data
59     .align      3
60     .set       .LANCHOR0,. + 0
61     .type      array, @object
62     .size      array, 40
63     .array:
64     .word      2
65     .word      9
66     .word      10
67     .word      5
68     .word      6
69     .word      7
70     .word      3
71     .word      4
72     .word      1
73     .word      8
74     .ident     "GCC: (SiFive GCC-Metal 10.2.0-2020.12.8) 10.2.0"

```

Рис. 7 файл main.s


```

median.s  + X
1  .file  "median.c"
2  .option nopie
3  .attribute arch, "rv64i2p0_a2p0_c2p0"
4  .attribute unaligned_access, 0
5  .attribute stack_align, 16
6  .text
7  .align 1
8  .globl median
9  .type median, @function
10 median:
11     beq a1,zero,.L8
12     slli a6,a1,2
13     add a6,a6,a0
14     mv t1,a0
15     li a7,0
16     li t4,0
17     li t3,1
18     j .L3
19 .L5:
20     addi a5,a5,4
21     beq a5,a6,.L11
22 .L6:
23     lw a4,0(a5)
24     bgeu a2,a4,.L4
25     addiw a3,a3,1
26 .L4:
27     bleu a2,a4,.L5
28     addiw a3,a3,-1
29     j .L5
30 .L11:
31     sext.w a3,a3
32     bleu a3,t3,.L12
33     addi a7,a7,1
34     addi t1,t1,4
35     beq a1,a7,.L9
36 .L3:
37     lw a2,0(t1)
38     mv a5,a0
39     mv a3,t4
40     j .L6
41 .L12:
42     slli a7,a7,2
43     add a7,a0,a7
44     lw a0,0(a7)
45     ret
46 .L8:
47     li a0,-1
48     ret
49 .L9:
50     li a0,-1
51     ret
52 .size median,.-median
53 .ident "GCC: (SiFive GCC-Metal 10.2.0-2020.12.8) 10.2.0"

```

Рис. 8 файл median.s

Объектный файл

Выполним ассемблирование для получения объектных файлов программы.

Для этого исполним следующие команды:

```
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -v -c main.s -o main.o
```

```
>log_obj main.txt 2>&1
```

```
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -v -c median.s -o median.o
```

```
>log_obj median.txt 2>&1
```

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

main.o:      file format elf64-littleriscv

Sections:
Idx Name          Size      VMA           LMA           File off  Algn
  0 .text          0000007a  0000000000000000  0000000000000000  00000040  2**1
CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data          00000028  0000000000000000  0000000000000000  000000c0  2**3
CONTENTS, ALLOC, LOAD, DATA
  2 .bss           00000000  0000000000000000  0000000000000000  000000e8  2**0
ALLOC
  3 .rodata.str1.8 0000001c  0000000000000000  0000000000000000  000000e8  2**3
CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .comment       00000031  0000000000000000  0000000000000000  00000104  2**0
CONTENTS, READONLY
  5 .riscv.attributes 00000026  0000000000000000  0000000000000000  00000135  2**0
CONTENTS, READONLY
```

Рис. 9 Хэдер файла main.o, полученный по команде в первой строке.

```
riscv64-unknown-elf-objdump -h median.o

median.o:    file format elf64-littleriscv

Sections:
Idx Name          Size      VMA           LMA           File off  Algn
  0 .text          00000052  0000000000000000  0000000000000000  00000040  2**1
CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data          00000000  0000000000000000  0000000000000000  00000092  2**0
CONTENTS, ALLOC, LOAD, DATA
  2 .bss           00000000  0000000000000000  0000000000000000  00000092  2**0
ALLOC
  3 .comment       00000031  0000000000000000  0000000000000000  00000092  2**0
CONTENTS, READONLY
  4 .riscv.attributes 00000026  0000000000000000  0000000000000000  000000c3  2**0
CONTENTS, READONLY
```

Рис. 10 Хэдер файла median.o, полученный по команде в первой строке.

.text – секция кода, содержащая коды инструкций

.data – секция инициализированных данных

.bss – секция данных, инициализированных нулями

.comment – секция данных о версиях

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

Опция “-d” иницирует процесс дизассемблирования (disassemble),

опция “-M no-aliases” требует использовать в выводе только инструкции системы команд (но не псевдоинструкции ассемблера).

Вывод утилиты:

```
main.o:      file format elf64-littleriscv

Disassembly of section .text:

0000000000000000 <main>:
 0: 1101          c.addi    sp,-32
 2: ec06          c.sdsp    ra,24(sp)
 4: e822          c.sdsp    s0,16(sp)
 6: e426          c.sdsp    s1,8(sp)
 8: e04a          c.sdsp    s2,0(sp)
 a: 00000537      lui      a0,0x0
 e: 00050513      addi     a0,a0,0 # 0 <main>
12: 00000097      auipc    ra,0x0
16: 000080e7      jalr     ra,0(ra) # 12 <main+0x12>
1a: 00000437      lui      s0,0x0
1e: 00040413      addi     s0,s0,0 # 0 <main>
22: 02840913      addi     s2,s0,40
26: 000004b7      lui      s1,0x0

000000000000002a <.L2>:
2a: 400c          c.lw      a1,0(s0)
2c: 00048513      addi     a0,s1,0 # 0 <main>
30: 00000097      auipc    ra,0x0
34: 000080e7      jalr     ra,0(ra) # 30 <.L2+0x6>
38: 0411          c.addi    s0,4
3a: ff2418e3      bne      s0,s2,2a <.L2>
3e: 4529          c.li      a0,10
40: 00000097      auipc    ra,0x0
44: 000080e7      jalr     ra,0(ra) # 40 <.L2+0x16>
48: 45a9          c.li      a1,10
4a: 00000537      lui      a0,0x0
4e: 00050513      addi     a0,a0,0 # 0 <main>
52: 00000097      auipc    ra,0x0
56: 000080e7      jalr     ra,0(ra) # 52 <.L2+0x28>
5a: 85aa          c.mv      a1,a0
5c: 00000537      lui      a0,0x0
60: 00050513      addi     a0,a0,0 # 0 <main>
64: 00000097      auipc    ra,0x0
68: 000080e7      jalr     ra,0(ra) # 64 <.L2+0x3a>
6c: 4501          c.li      a0,0
6e: 60e2          c.ldsp    ra,24(sp)
70: 6442          c.ldsp    s0,16(sp)
72: 64a2          c.ldsp    s1,8(sp)
74: 6902          c.ldsp    s2,0(sp)
76: 6105          c.addi16sp sp,32
78: 8082          c.jr      ra
```

Рис. 11 дизассемблированный файл main.o

Рассмотрим таблицу символов, выполнив команду:

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

```
median.o:      file format elf64-littleriscv

SYMBOL TABLE:
0000000000000000 1      df *ABS*  0000000000000000 median.c
0000000000000000 1      d  .text  0000000000000000 .text
0000000000000000 1      d  .data  0000000000000000 .data
0000000000000000 1      d  .bss   0000000000000000 .bss
000000000000004a 1      .text  0000000000000000 .L8
0000000000000036 1      .text  0000000000000000 .L3
0000000000000028 1      .text  0000000000000000 .L11
0000000000000020 1      .text  0000000000000000 .L4
0000000000000012 1      .text  0000000000000000 .L5
0000000000000040 1      .text  0000000000000000 .L12
000000000000004e 1      .text  0000000000000000 .L9
0000000000000018 1      .text  0000000000000000 .L6
0000000000000000 1      d  .comment 0000000000000000 .comment
0000000000000000 1      d  .riscv.attributes 0000000000000000 .riscv.attributes
0000000000000000 g      F  .text  0000000000000052 median

main.o:      file format elf64-littleriscv

SYMBOL TABLE:
0000000000000000 1      df *ABS*  0000000000000000 main.c
0000000000000000 1      d  .text  0000000000000000 .text
0000000000000000 1      d  .data  0000000000000000 .data
0000000000000000 1      d  .bss   0000000000000000 .bss
0000000000000000 1      d  .rodata.str1.8 0000000000000000 .rodata.str1.8
0000000000000000 1      .data  0000000000000000 .ANCHOR0
0000000000000000 1      0  .data  0000000000000028 array
0000000000000000 1      .rodata.str1.8 0000000000000000 .LC0
0000000000000008 1      .rodata.str1.8 0000000000000000 .LC1
0000000000000010 1      .rodata.str1.8 0000000000000000 .LC2
000000000000002a 1      .text  0000000000000000 .L2
0000000000000000 1      d  .comment 0000000000000000 .comment
0000000000000000 1      d  .riscv.attributes 0000000000000000 .riscv.attributes
0000000000000000 g      F  .text  000000000000007a main
0000000000000000      *UND*  0000000000000000 puts
0000000000000000      *UND*  0000000000000000 printf
0000000000000000      *UND*  0000000000000000 putchar
0000000000000000      *UND*  0000000000000000 median
```

Рис. 12 Таблица символов.

В таблице символов “main.o” имеется интересная запись: символ “zero” типа “*UND*” (undefined – не определен). Эта запись означает, что символ “zero” использовался в ассемблерном коде, из которого был получен данный объектный файл, но не был определен; ассемблер сделал вывод о том, что символ должен быть определен где-то еще, и отразил это в таблице символов. Информация обо всех «неоконченных» инструкциях передается ассемблером компоновщику посредством таблицы перемещений, получить которую можно по команде:

riscv64-unknown-elf-objdump -r median.o main.o

```
median.o:      file format elf64-littleriscv
```

```
RELOCATION RECORDS FOR [.text]:
```

OFFSET	TYPE	VALUE
0000000000000000	R_RISCV_RVC_BRANCH	.L8
0000000000000010	R_RISCV_RVC_JUMP	.L3
0000000000000014	R_RISCV_BRANCH	.L11
000000000000001a	R_RISCV_BRANCH	.L4
0000000000000020	R_RISCV_BRANCH	.L5
0000000000000026	R_RISCV_RVC_JUMP	.L5
000000000000002a	R_RISCV_BRANCH	.L12
0000000000000032	R_RISCV_BRANCH	.L9
000000000000003e	R_RISCV_RVC_JUMP	.L6

```
main.o:      file format elf64-littleriscv
```

```
RELOCATION RECORDS FOR [.text]:
```

OFFSET	TYPE	VALUE
000000000000000a	R_RISCV_HI20	.LC0
000000000000000a	R_RISCV_RELAX	*ABS*
000000000000000e	R_RISCV_LO12_I	.LC0
000000000000000e	R_RISCV_RELAX	*ABS*
0000000000000012	R_RISCV_CALL	puts
0000000000000012	R_RISCV_RELAX	*ABS*
000000000000001a	R_RISCV_HI20	.LANCHOR0
000000000000001a	R_RISCV_RELAX	*ABS*
000000000000001e	R_RISCV_LO12_I	.LANCHOR0
000000000000001e	R_RISCV_RELAX	*ABS*
0000000000000026	R_RISCV_HI20	.LC1
0000000000000026	R_RISCV_RELAX	*ABS*
000000000000002c	R_RISCV_LO12_I	.LC1
000000000000002c	R_RISCV_RELAX	*ABS*
0000000000000030	R_RISCV_CALL	printf
0000000000000030	R_RISCV_RELAX	*ABS*
0000000000000040	R_RISCV_CALL	putchar
0000000000000040	R_RISCV_RELAX	*ABS*
000000000000004a	R_RISCV_HI20	.LANCHOR0
000000000000004a	R_RISCV_RELAX	*ABS*
000000000000004e	R_RISCV_LO12_I	.LANCHOR0
000000000000004e	R_RISCV_RELAX	*ABS*
0000000000000052	R_RISCV_CALL	median
0000000000000052	R_RISCV_RELAX	*ABS*
000000000000005c	R_RISCV_HI20	.LC2
000000000000005c	R_RISCV_RELAX	*ABS*
0000000000000060	R_RISCV_LO12_I	.LC2
0000000000000060	R_RISCV_RELAX	*ABS*
0000000000000064	R_RISCV_CALL	printf
0000000000000064	R_RISCV_RELAX	*ABS*
000000000000003a	R_RISCV_BRANCH	.L2

Рис. 13 Таблица перемещений

В таблице перемещений для main.o наблюдаем вызов метода median. 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.

Компоновка

Выполним компоновку командой

```
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -v main.o median.o -o  
main.out >log_out.txt 2>&1
```

В результате выполнения этой команды был получен файл main.out – исполняемый бинарный файл. Рассмотрим его секцию кода с помощью команды:

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

```

67 0000000000010156 <main>:
68 10156: 1101 c.addi sp,-32
69 10158: ec06 c.sdsp ra,24(sp)
70 1015a: e822 c.sdsp s0,16(sp)
71 1015c: e426 c.sdsp s1,8(sp)
72 1015e: e04a c.sdsp s2,0(sp)
73 10160: 6575 c.lui a0,0x1d
74 10162: cc050513 addi a0,a0,-832 # 1ccc0 <__clzdi2+0x3e>
75 10166: 37e000ef jal ra,104e4 <puts>
76 1016a: 0001f437 lui s0,0x1f
77 1016e: c5040413 addi s0,s0,-944 # 1ec50 <array>
78 10172: 02840913 addi s2,s0,40
79 10176: 64f5 c.lui s1,0x1d
80 10178: 400c c.lw a1,0(s0)
81 1017a: cc848513 addi a0,s1,-824 # 1ccc8 <__clzdi2+0x46>
82 1017e: 1de000ef jal ra,1035c <printf>
83 10182: 0411 c.addi s0,4
84 10184: ff241ae3 bne s0,s2,10178 <main+0x22>
85 10188: 4529 c.li a0,10
86 1018a: 202000ef jal ra,1038c <putchar>
87 1018e: 45a9 c.li a1,10
88 10190: 0001f537 lui a0,0x1f
89 10194: c5050513 addi a0,a0,-944 # 1ec50 <array>
90 10198: 01e000ef jal ra,101b6 <median>
91 1019c: 85aa c.mv a1,a0
92 1019e: 6575 c.lui a0,0x1d
93 101a0: cd050513 addi a0,a0,-816 # 1ccd0 <__clzdi2+0x4e>
94 101a4: 1b8000ef jal ra,1035c <printf>
95 101a8: 4501 c.li a0,0
96 101aa: 60e2 c.ldsp ra,24(sp)
97 101ac: 6442 c.ldsp s0,16(sp)
98 101ae: 64a2 c.ldsp s1,8(sp)
99 101b0: 6902 c.ldsp s2,0(sp)
100 101b2: 6105 c.addi16sp sp,32
101 101b4: 8082 c.jr ra
102
103 00000000000101b6 <median>:
104 101b6: c5a9 c.beqz a1,10200 <median+0x4a>
105 101b8: 00259813 slli a6,a1,0x2
106 101bc: 982a c.add a6,a0
107 101be: 832a c.mv t1,a0
108 101c0: 4881 c.li a7,0
109 101c2: 4e81 c.li t4,0
110 101c4: 4e05 c.li t3,1
111 101c6: a01d c.j 101ec <median+0x36>
112 101c8: 0791 c.addi a5,4
113 101ca: 01078a63 beq a5,a6,101de <median+0x28>
114 101ce: 4398 c.lw a4,0(a5)
115 101d0: 00e67363 bgeu a2,a4,101d6 <median+0x20>
116 101d4: 2685 c.addiw a3,1
117 101d6: fec779e3 bgeu a4,a2,101c8 <median+0x12>
118 101da: 36fd c.addiw a3,-1
119 101dc: b7f5 c.j 101c8 <median+0x12>
120 101de: 2681 c.addiw a3,0
121 101e0: 00de7b63 bgeu t3,a3,101f6 <median+0x40>
122 101e4: 0885 c.addi a7,1
123 101e6: 0311 c.addi t1,4
124 101e8: 01158e63 beq a1,a7,10204 <median+0x4e>
125 101ec: 00032603 lw a2,0(t1) # 1014c <frame_dummy+0x12>
126 101f0: 87aa c.mv a5,a0
127 101f2: 86f6 c.mv a3,t4
128 101f4: bfe9 c.j 101ce <median+0x18>
129 101f6: 088a c.slli a7,0x2
130 101f8: 98aa c.add a7,a0
131 101fa: 0008a503 lw a0,0(a7)
132 101fe: 8082 c.jr ra
133 10200: 557d c.li a0,-1
134 10202: 8082 c.jr ra
135 10204: 557d c.li a0,-1
136 10206: 8082 c.jr ra
137

```

Рис. 14 содержимое файла a.ds

Адресация для вызовов функций изменилась на абсолютную

3. Создание статической библиотеки

Выделим функцию `median` в отдельную статическую библиотеку. Для этого надо получить объектный файл `median.o` и собрать библиотеку.

```
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -c median.c -o median.o
```

```
riscv64-unknown-elf-ar -rsc libMedian.a median.o
```

Рассмотрим список символов `libMedian.a` с помощью команды

```
riscv64-unknown-elf-nm libMedian.a
```

```
median.o:
0000000000000028 t .L11
0000000000000040 t .L12
0000000000000036 t .L3
0000000000000020 t .L4
0000000000000012 t .L5
0000000000000018 t .L6
000000000000004a t .L8
000000000000004e t .L9
0000000000000000 T median
```

Рис. 15 список символов

В выводе утилиты “nm” кодом “T” обозначаются символы, определенные в соответствующем объектном файле. Используя собранную библиотеку, произведём исполняемый файл тестовой программы с помощью команды:

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

Посмотрим содержимое таблицы символов исполняемого файла с помощью команды:

```
riscv64-unknown-elf-objdump -t main.out >main.ds и убедимся, что так есть функция median
```

34	0000000000000000	1	df *ABS*	0000000000000000	main.c
35	0000000000001ec50	1	O .data	0000000000000028	array
36	0000000000000000	1	df *ABS*	0000000000000000	median.c
37	0000000000000000	1	df *ABS*	0000000000000000	exit.c
38	0000000000000000	1	df *ABS*	0000000000000000	impure.c

Рис. 16 таблица символов исполняемого файла

Создание make-файлов

Чтобы автоматизировать процесс сборки библиотеки и приложения напишем make-файлы. Используя пример с сайта курса, были написаны следующие файлы:

```
1 CC=riscv64-unknown-elf-gcc
2 AR=riscv64-unknown-elf-ar
3 CFLAGS=-march=rv64iac -mabi=lp64
4
5 all: lib
6
7 lib: median.o
8     $(AR) -rsc libmedian.a median.o
9     $(RM) -f *.o *.s
10 median.o: median.c
11     $(CC) $(CFLAGS) -c median.c -o median.o
```

Рис. 17 содержание файла make_lib

```
1 TARGET=main
2 CC=riscv64-unknown-elf-gcc
3 AR=riscv64-unknown-elf-ar
4 CFLAGS=-march=rv64iac -mabi=lp64
5
6 all:
7     make -f make_lib
8     riscv64-unknown-elf-gcc $(CFLAGS) main.c libMedian.a -o main
9     $(RM) -f *.a *.o
```

Рис. 18 содержание файла make_app

Для создания библиотеки необходимо выполнить make_lib, а для приложения - make_app.

```

C:\Users\skwit\Desktop\Study\4sem\Прога\lab4>C:\cygwin64\bin\make.exe -f make_lib
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -c median.c -o median.o
riscv64-unknown-elf-ar -rsc libmedian.a median.o
rm -f -f *.o *.s

C:\Users\skwit\Desktop\Study\4sem\Прога\lab4>dir
Том в устройстве C не имеет метки.
Серийный номер тома: 2C83-BF4D

Содержимое папки C:\Users\skwit\Desktop\Study\4sem\Прога\lab4

21.04.2021  02:22    <DIR>          .
21.04.2021  02:22    <DIR>          ..
21.04.2021  02:22             1 888 libmedian.a
17.04.2021  06:49             445 main.c
21.04.2021  12:08             213 make_app
21.04.2021  12:07             231 make_lib
17.04.2021  06:51             606 median.c
17.04.2021  06:36              89 median.h
21.04.2021  11:16           853 899 Дергачев_отчет4.docx
              7 файлов             857 371 байт
              2 папок   77 590 798 336 байт свободно

```

Рис. 19 выполнение файла make_lib

```

C:\Users\skwit\Desktop\Study\4sem\Прога\lab4>C:\cygwin64\bin\make.exe -f make_app
make -f make_lib
make[1]: Entering directory '/cygdrive/c/Users/skwit/Desktop/Study/4sem/Прога/lab4'
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -c median.c -o median.o
riscv64-unknown-elf-ar -rsc libmedian.a median.o
rm -f -f *.o *.s
make[1]: Leaving directory '/cygdrive/c/Users/skwit/Desktop/Study/4sem/Прога/lab4'
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 main.c libMedian.a -o main
rm -f-f *.a *.o

C:\Users\skwit\Desktop\Study\4sem\Прога\lab4>dir
Том в устройстве C не имеет метки.
Серийный номер тома: 2C83-BF4D

Содержимое папки C:\Users\skwit\Desktop\Study\4sem\Прога\lab4

21.04.2021  02:22    <DIR>          .
21.04.2021  02:22    <DIR>          ..
21.04.2021  02:22           143 656 main
17.04.2021  06:49             445 main.c
21.04.2021  12:08             213 make_app
21.04.2021  12:07             231 make_lib
17.04.2021  06:51             606 median.c
17.04.2021  06:36              89 median.h
21.04.2021  11:16           853 899 Дергачев_отчет4.docx
              7 файлов             999 139 байт
              2 папок   77 590 646 784 байт свободно

```

Рис. 20 выполнение файла make_app

Вывод:

В ходе выполнения работы была написана программа на языке C, была выполнена сборка этой программы по шагам для архитектуры RISC-V, была создана статическая библиотека. Также были написаны два make-файла для автоматизированной сборки библиотеки и тестовой программы.