

Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных
технологий

Низкоуровневое программирование

Лабораторная работа №4 "Раздельная компиляция"

Работу
выполнил:
В. А. Денисов
Группа:
3530901/10006
Преподаватель:
М. А. Петров

Санкт-Петербург
2022

Содержание

1. Формулировка задачи	3
2. Исходные файлы	3
3. Проверка работы программы с использованием gcc	4
4. Сборка программы	5
4.1. Препроцессинг	5
4.2. Компиляция	7
4.3. Сборка объектов	10
4.4. Компоновка	14
4.5. Выделение статических библиотек	16
5. Создание Makefile	17
5.1. Проверка	18
6. Вывод	18

1. Формулировка задачи

В соответствии с условием 7 варианта требуется написать программу на С осуществляющую определение k-й порядковой статистики in-place:

1. 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.
2. Провести пошаговую компиляцию данных программ с анализом получающихся файлов. Создать статическую библиотеку, состоящую из тестовой программы и программы с необходимым функционалом, проследив этапы её создания.

2. Исходные файлы

Для реализации используется модифицированный алгоритм быстрой сортировки. Итоговая программа состоит из пяти файлов кода - main.c (основная программа), nth.c (подпрограмма поиска k-ой статистики), partition.c (подпрограмма разбиения массива), а также двух заголовочных файлов (чтобы выделить подпрограммы в статические библиотеки) - nth.h и partition.h.

```
1  #include <stdio.h>
2  #include "nth.h"
3
4  void main() {
5      int n = 6; // array size
6      int array[] = {12, 36, 110, 1, 6, 99}; // array
7      int k = 4;
8      int result = nth(array, n, k);
9      printf("%i\n", result);
10 }
```

Листинг 1: main.c

```
1  #ifndef NTH_H
2  #define NTH_H
3  int nth(int *array, unsigned int n, unsigned int k);
4  #endif
```

Листинг 2: nth.h

```
1  #include "partition.h"
2
3  int nth(int *array, unsigned int n, unsigned int k) {
4      k--;
5      int left = 0;
6      int right = n - 1;
7      while (1) {
8          int mid = partition(array, left, right);
```

```

9     if (mid == k) {
10         return array[mid];
11     }
12     if (k < mid) {
13         right = mid;
14     } else {
15         left = mid + 1;
16     }
17 }
18 }

```

Листинг 3: nth.c

```

1  #ifndef PARTITION_H
2  #define PARTITION_H
3  int partition(int *array, unsigned int left, unsigned int right);
4  #endif

```

Листинг 4: partition.h

```

1  int partition(int *array, unsigned int left, unsigned int right) {
2      int v = array[(left + right) / 2];
3      int i = left;
4      int j = right;
5      while (i < j) {
6          while (array[i] < v) {
7              i++;
8          }
9          while (array[j] > v) {
10             j--;
11         }
12         int t = array[i];
13         array[i] = array[j];
14         array[j] = t;
15     }
16     return j;
17 }

```

Листинг 5: partition.c

3. Проверка работы программы с использованием gcc

Соберем программу:

```

1  > gcc main.c nth.c partition.c

```

Исполним:

```

1  > ./a.out
2  36

```

4. Сборка программы

4.1. Препроцессинг

Команды:

```
1 > riscv64-unknown-elf-gcc --save-temps -march=rv32i -mabi=ilp32 -O1 -E  
  ↳ main.c -o main.i  
2 > riscv64-unknown-elf-gcc --save-temps -march=rv32i -mabi=ilp32 -O1 -E  
  ↳ nth.c -o nth.i  
3 > riscv64-unknown-elf-gcc --save-temps -march=rv32i -mabi=ilp32 -O1 -E  
  ↳ partition.c -o partition.i
```

Рассмотрим выходы препроцессора для main.c, nth.c и partition.c

Файл main.i получился огромным, так как main.c включает в себя файл стандартной библиотеки stdio.h. С целью экономии места, файл приведен частично.

```
1 # 1 "main.c"  
2 # 1 "<built-in>"  
3 # 1 "<command-line>"  
4 # 1 "main.c"  
5 # 2 "main.c" 2  
6 # 1 "nth.h" 1  
7  
8 /* stdio.h */  
9  
10 # 3 "nth.h"  
11 int nth(int *array, unsigned int n, unsigned int k);  
12 # 3 "main.c" 2  
13  
14 void main() {  
15     int n = 6;  
16     int array[] = {12, 36, 110, 1, 6, 99};  
17     int k = 4;  
18     int result = nth(array, n, k);  
19     printf("%i\n", result);  
20 }
```

Листинг 6: main.i

```
1 # 0 "nth.c"  
2 # 0 "<built-in>"  
3 # 0 "<command-line>"  
4 # 1 "nth.c"  
5 # 1 "partition.h" 1  
6  
7  
8 int partition(int *array, unsigned int left, unsigned int right);  
9 # 2 "nth.c" 2  
10  
11 int nth(int *array, unsigned int n, unsigned int k) {  
12     k--;  
13     int left = 0;  
14     int right = n - 1;  
15     while (1) {
```

```

16     int mid = partition(array, left, right);
17     if (mid == k) {
18         return array[mid];
19     }
20     if (k < mid) {
21         right = mid;
22     } else {
23         left = mid + 1;
24     }
25 }
26 }

```

Листинг 7: nth.i

```

1  # 0 "partition.c"
2  # 0 "<built-in>"
3  # 0 "<command-line>"
4  # 1 "partition.c"
5  int partition(int *array, unsigned int left, unsigned int right) {
6      int v = array[(left + right) / 2];
7      int i = left;
8      int j = right;
9      while (i < j) {
10         while (array[i] < v) {
11             i++;
12         }
13         while (array[j] > v) {
14             j--;
15         }
16         int t = array[i];
17         array[i] = array[j];
18         array[j] = t;
19     }
20     return j;
21 }

```

Листинг 8: partition.i

Можно заметить, что файлы после препроцессинга содержат код исходных файлов (в том числе код из заголовочных файлов) без комментариев, но с нестандартными директивами, начинающиеся с символа “`#`”, использующимися для передачи информации об исходном тексте из препроцессора в компилятор.

Например,

```

1  # 2 "nth.c" 2

```

обозначает, что далее идет код из файла nth.c со второй строчки.

4.2. Компиляция

Команды:

```
1 > riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 -S main.i -o  
  ↪ main.s  
2 > riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 -S nth.i -o nth.s  
3 > riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 -S partition.i -o  
  ↪ partition.s
```

Эти команды генерируют код для ассемблера. Изучим их:

```
1      .file          "main.c"  
2      .option nopic  
3      .attribute arch, "rv32i2p0"  
4      .attribute unaligned_access, 0  
5      .attribute stack_align, 16  
6      .text  
7      .section      .rodata.str1.4, "aMS", @progbits, 1  
8      .align        2  
9  
10     .LC1:  
11     .string        "%i\n"  
12     .text  
13     .align        2  
14     .globl         main  
15     .type          main, @function  
16  
17     main:  
18     addi           sp, sp, -48  
19     sw             ra, 44(sp)  
20     lui            a5, %hi(.LANCHOR0)  
21     addi           a5, a5, %lo(.LANCHOR0)  
22     lw             a0, 0(a5)  
23     lw             a1, 4(a5)  
24     lw             a2, 8(a5)  
25     lw             a3, 12(a5)  
26     lw             a4, 16(a5)  
27     lw             a5, 20(a5)  
28     sw             a0, 8(sp)  
29     sw             a1, 12(sp)  
30     sw             a2, 16(sp)  
31     sw             a3, 20(sp)  
32     sw             a4, 24(sp)  
33     sw             a5, 28(sp)  
34     li             a2, 4  
35     li             a1, 6  
36     addi           a0, sp, 8  
37     call           nth  
38     mv             a1, a0  
39     lui            a0, %hi(.LC1)  
40     addi           a0, a0, %lo(.LC1)  
41     call           printf  
42     lw             ra, 44(sp)  
43     addi           sp, sp, 48  
44     jr            ra  
45     .size          main, .-main  
46     .section      .rodata  
47     .align        2  
48     .set           .LANCHOR0, . + 0
```

```

47  .LC0:
48      .word      12
49      .word      36
50      .word      110
51      .word      1
52      .word      6
53      .word      99
54      .ident      "GCC: (g2ee5e430018-dirty) 12.2.0"

```

Листинг 9: main.s

В целом, все остальные файлы по структуре напоминают файл main.s, а остальной код очень похож на код файлов из Лабораторной работы №3, так что не вижу смысла в детальном пояснении.

```

1      .file      "nth.c"
2      .option nopic
3      .attribute arch, "rv32i2p0"
4      .attribute unaligned_access, 0
5      .attribute stack_align, 16
6      .text
7      .align      2
8      .globl      nth
9      .type      nth, @function
10 nth:
11      addi      sp,sp,-32
12      sw        ra,28(sp)
13      sw        s0,24(sp)
14      sw        s1,20(sp)
15      sw        s2,16(sp)
16      sw        s3,12(sp)
17      mv        s2,a0
18      addi      s0,a2,-1
19      addi      s1,a1,-1
20      li        s3,0
21      j         .L2
22  .L5:
23      mv        s1,a0
24  .L2:
25      mv        a2,s1
26      mv        a1,s3
27      mv        a0,s2
28      call      partition
29      beq       a0,s0,.L7
30      bgtu      a0,s0,.L5
31      addi      s3,a0,1
32      j         .L2
33  .L7:
34      slli      s0,s0,2
35      add       s2,s2,s0
36      lw        a0,0(s2)
37      lw        ra,28(sp)
38      lw        s0,24(sp)
39      lw        s1,20(sp)
40      lw        s2,16(sp)
41      lw        s3,12(sp)
42      addi      sp,sp,32
43      jr        ra

```



```

44     .size      nth, .-nth
45     .ident     "GCC: (g2ee5e430018-dirty) 12.2.0"

```

Листинг 10: nth.s

```

1      .file      "partition.c"
2      .option nopie
3      .attribute arch, "rv32i2p0"
4      .attribute unaligned_access, 0
5      .attribute stack_align, 16
6      .text
7      .align     2
8      .globl     partition
9      .type      partition, @function
10     partition:
11         mv      t1,a0
12         mv      a0,a2
13         add     a5,a1,a2
14         srli    a5,a5,1
15         slli    a5,a5,2
16         add     a5,t1,a5
17         lw      a2,0(a5)
18         blt     a1,a0,.L2
19         ret
20     .L4:
21         addi    a1,a1,1
22         mv      a3,a5
23         addi    a5,a5,4
24         lw      a4,-4(a5)
25         blt     a4,a2,.L4
26         mv      a6,a3
27     .L8:
28         slli    a5,a0,2
29         add     a7,t1,a5
30         lw      a3,0(a7)
31         bge     a2,a3,.L5
32         addi    a5,a5,-4
33         add     a5,t1,a5
34     .L6:
35         addi    a0,a0,-1
36         mv      a7,a5
37         addi    a5,a5,-4
38         lw      a3,4(a5)
39         bgt     a3,a2,.L6
40     .L5:
41         sw      a3,0(a6)
42         sw      a4,0(a7)
43         bge     a1,a0,.L1
44     .L2:
45         slli    a5,a1,2
46         add     a6,t1,a5
47         lw      a4,0(a6)
48         addi    a5,a5,4
49         add     a5,t1,a5
50         bgt     a2,a4,.L4
51         j       .L8
52     .L1:
53         ret

```

```

54     .size      partition, .-partition
55     .ident     "GCC: (g2ee5e430018-dirty) 12.2.0"

```

Листинг 11: partition.s

4.3. Сборка объектов

Команды:

```

1  > riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 -c main.s -o
   ↪ main.o
2  > riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 -c nth.s -o nth.o
3  > riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 -c partition.i -o
   ↪ partition.o

```

Команды для вывода заголовков секций, таблиц символов, таблицы перемещений:

```

1  > riscv64-unknown-elf-objdump -d -t -h -S main.o > ../tmp/main_obj.txt
2  > riscv64-unknown-elf-objdump -d -t -S nth.o > ../tmp/nth_obj.txt
3  > riscv64-unknown-elf-objdump -d -t -S partition.o >
   ↪ ../tmp/partition_obj.txt

```

Исследуем объектные файлы:

```

1  main.o:      file format elf32-littleriscv
2
3
4  Sections:
5  Idx Name          Size      VMA      LMA      File off  Algn
6    0 .text          00000074  00000000  00000000  00000034  2**2
7      CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
8    1 .data           00000000  00000000  00000000  000000a8  2**0
9      CONTENTS, ALLOC, LOAD, DATA
10   2 .bss            00000000  00000000  00000000  000000a8  2**0
11      ALLOC
12   3 .rodata.str1.4  00000004  00000000  00000000  000000a8  2**2
13      CONTENTS, ALLOC, LOAD, READONLY, DATA
14   4 .rodata          00000018  00000000  00000000  000000ac  2**2
15      CONTENTS, ALLOC, LOAD, READONLY, DATA
16   5 .comment         00000022  00000000  00000000  000000c4  2**0
17      CONTENTS, READONLY
18   6 .riscv.attributes 0000001c  00000000  00000000  000000e6  2**0
19      CONTENTS, READONLY
20  SYMBOL TABLE:
21  00000000 l      df *ABS*          00000000 main.c
22  00000000 l      d  .text          00000000 .text
23  00000000 l      d  .data          00000000 .data
24  00000000 l      d  .bss           00000000 .bss
25  00000000 l      d  .rodata.str1.4 00000000 .rodata.str1.4
26  00000000 l      d  .rodata        00000000 .rodata
27  00000000 l      d  .comment       00000000 .comment
28  00000000 l      d  .riscv.attributes 00000000 .riscv.attributes
29  00000000 g      F  .text          00000074 main
30  00000000      *UND*          00000000 nth

```

```

31 00000000      *UND*      00000000 printf
32
33
34
35 Disassembly of section .text:
36
37 00000000 <main>:
38 0:      fd010113          addi      sp,sp,-48
39 4:      02112623          sw        ra,44(sp)
40 8:      000007b7          lui       a5,0x0
41 c:      00078793          mv        a5,a5
42 10:     0007a503          lw        a0,0(a5) # 0 <main>
43 14:     0047a583          lw        a1,4(a5)
44 18:     0087a603          lw        a2,8(a5)
45 1c:     00c7a683          lw        a3,12(a5)
46 20:     0107a703          lw        a4,16(a5)
47 24:     0147a783          lw        a5,20(a5)
48 28:     00a12423          sw        a0,8(sp)
49 2c:     00b12623          sw        a1,12(sp)
50 30:     00c12823          sw        a2,16(sp)
51 34:     00d12a23          sw        a3,20(sp)
52 38:     00e12c23          sw        a4,24(sp)
53 3c:     00f12e23          sw        a5,28(sp)
54 40:     00400613          li        a2,4
55 44:     00600593          li        a1,6
56 48:     00810513          addi      a0,sp,8
57 4c:     00000097          auipc     ra,0x0
58 50:     000080e7          jalr      ra # 4c <main+0x4c>
59 54:     00050593          mv        a1,a0
60 58:     00000537          lui       a0,0x0
61 5c:     00050513          mv        a0,a0
62 60:     00000097          auipc     ra,0x0
63 64:     000080e7          jalr      ra # 60 <main+0x60>
64 68:     02c12083          lw        ra,44(sp)
65 6c:     03010113          addi      sp,sp,48
66 70:     00008067          ret

```

Листинг 12: main.o

В таблице перемещений содержится информация о необходимых заменах адресов, которая будет передана компоновщику. Видно, например, что в файле main.o есть информация о необходимости замены для всех адресов внешних функций.

Файлы имеют исполняемый формат (elf), несколько разделов (с данными, с значениями инициализируемыми нулями, с кодом и meta-информацией).

Отметим, что функции nth, и printf помечены как неопределённые. Это нужно для того, чтобы компоновщик при помощи таблицы перемещений подставил вместо них нужные адреса вызовов.

```

1 nth.o:      file format elf32-littleriscv
2
3
4 SYMBOL TABLE:
5 00000000 l    df *ABS*      00000000 nth.c
6 00000000 l    d  .text      00000000 .text
7 00000000 l    d  .data      00000000 .data
8 00000000 l    d  .bss      00000000 .bss
9 00000000 l    d  .comment  00000000 .comment
10 00000000 l    d  .riscv.attributes 00000000 .riscv.attributes

```

```

11 00000000 g      F .text      0000007c nth
12 00000000      *UND*      00000000 partition
13
14
15
16 Disassembly of section .text:
17
18 00000000 <nth>:
19     0:      fe010113          addi      sp,sp,-32
20     4:      00112e23          sw        ra,28(sp)
21     8:      00812c23          sw        s0,24(sp)
22    c:      00912a23          sw        s1,20(sp)
23    10:      01212823          sw        s2,16(sp)
24    14:      01312623          sw        s3,12(sp)
25    18:      00050913          mv        s2,a0
26    1c:      fff60413          addi      s0,a2,-1
27    20:      fff58493          addi      s1,a1,-1
28    24:      00000993          li        s3,0
29    28:      0080006f          j        30 <.L2>
30
31 0000002c <.L5>:
32    2c:      00050493          mv        s1,a0
33
34 00000030 <.L2>:
35    30:      00048613          mv        a2,s1
36    34:      00098593          mv        a1,s3
37    38:      00090513          mv        a0,s2
38    3c:      00000097          auipc     ra,0x0
39    40:      000080e7          jalr      ra # 3c <.L2+0xc>
40    44:      00850863          beq       a0,s0,54 <.L7>
41    48:      fea462e3          bltu      s0,a0,2c <.L5>
42    4c:      00150993          addi      s3,a0,1
43    50:      fe1ff06f          j        30 <.L2>
44
45 00000054 <.L7>:
46    54:      00241413          slli      s0,s0,0x2
47    58:      00890933          add       s2,s2,s0
48    5c:      00092503          lw        a0,0(s2)
49    60:      01c12083          lw        ra,28(sp)
50    64:      01812403          lw        s0,24(sp)
51    68:      01412483          lw        s1,20(sp)
52    6c:      01012903          lw        s2,16(sp)
53    70:      00c12983          lw        s3,12(sp)
54    74:      02010113          addi      sp,sp,32
55    78:      00008067          ret

```

Листинг 13: nth.o

```

1
2 partition.o:      file format elf32-littleriscv
3
4 SYMBOL TABLE:
5 00000000 l      df *ABS*      00000000 partition.c
6 00000000 l      d  .text      00000000 .text
7 00000000 l      d  .data      00000000 .data
8 00000000 l      d  .bss       00000000 .bss
9 00000000 l      d  .comment    00000000 .comment
10 00000000 l      d  .riscv.attributes 00000000 .riscv.attributes

```

```

11 00000000 g      F .text      00000094 partition
12
13
14
15 Disassembly of section .text:
16
17 00000000 <partition>:
18     0:      00050313          mv      t1,a0
19     4:      00060513          mv      a0,a2
20     8:      00c587b3          add      a5,a1,a2
21    c:      0017d793          srli     a5,a5,0x1
22   10:      00279793          slli     a5,a5,0x2
23   14:      00f307b3          add      a5,t1,a5
24   18:      0007a603          lw      a2,0(a5)
25   1c:      04a5cc63          blt     a1,a0,74 <.L2>
26   20:      00008067          ret
27
28 00000024 <.L4>:
29   24:      00158593          addi     a1,a1,1
30   28:      00078693          mv      a3,a5
31   2c:      00478793          addi     a5,a5,4
32   30:      ffc7a703          lw      a4,-4(a5)
33   34:      fec748e3          blt     a4,a2,24 <.L4>
34   38:      00068813          mv      a6,a3
35
36 0000003c <.L8>:
37   3c:      00251793          slli     a5,a0,0x2
38   40:      00f308b3          add      a7,t1,a5
39   44:      0008a683          lw      a3,0(a7)
40   48:      02d65063          bge     a2,a3,68 <.L5>
41   4c:      ffc78793          addi     a5,a5,-4
42   50:      00f307b3          add      a5,t1,a5
43
44 00000054 <.L6>:
45   54:      fff50513          addi     a0,a0,-1
46   58:      00078893          mv      a7,a5
47   5c:      ffc78793          addi     a5,a5,-4
48   60:      0047a683          lw      a3,4(a5)
49   64:      fed648e3          blt     a2,a3,54 <.L6>
50
51 00000068 <.L5>:
52   68:      00d82023          sw      a3,0(a6)
53   6c:      00e8a023          sw      a4,0(a7)
54   70:      02a5d063          bge     a1,a0,90 <.L1>
55
56 00000074 <.L2>:
57   74:      00259793          slli     a5,a1,0x2
58   78:      00f30833          add      a6,t1,a5
59   7c:      00082703          lw      a4,0(a6)
60   80:      00478793          addi     a5,a5,4

```

Листинг 14: partition.o

4.4. Компоновка

Команда:

```
1 > riscv64-unknown-elf-gcc --save-temps -march=rv32i -mabi=ilp32 -O1 -E  
  ↪ main.o nth.o partition.o -o main.out  
2  
3  
4 > riscv64-unknown-elf-objdump -j .text -d ./ main.out > ../tmp/a.ds
```

В результате мы получаем исполняемый файл main.out, objdump которого содержит абсолютно все символы, которые необходимы для запуска на машине, которая поддерживает формат исполняемого файла elf32lriscv (32-bit Little RISC-V).

Ниже приведен фрагмент objdump:

```
1 <..>  
2 00010190 <main>:  
3 10190: fd010113 addi sp,sp,-48  
4 10194: 02112623 sw ra,44(sp)  
5 10198: 000257b7 lui a5,0x25  
6 1019c: 4b478793 addi a5,a5,1204 # 254b4  
  ↪ <__clzsi2+0x54>  
7 101a0: 0007a503 lw a0,0(a5)  
8 101a4: 0047a583 lw a1,4(a5)  
9 101a8: 0087a603 lw a2,8(a5)  
10 101ac: 00c7a683 lw a3,12(a5)  
11 101b0: 0107a703 lw a4,16(a5)  
12 101b4: 0147a783 lw a5,20(a5)  
13 101b8: 00a12423 sw a0,8(sp)  
14 101bc: 00b12623 sw a1,12(sp)  
15 101c0: 00c12823 sw a2,16(sp)  
16 101c4: 00d12a23 sw a3,20(sp)  
17 101c8: 00e12c23 sw a4,24(sp)  
18 101cc: 00f12e23 sw a5,28(sp)  
19 101d0: 00400613 li a2,4  
20 101d4: 00600593 li a1,6  
21 101d8: 00810513 addi a0,sp,8  
22 101dc: 020000ef jal ra,101fc <nth>  
23 101e0: 00050593 mv a1,a0  
24 101e4: 00025537 lui a0,0x25  
25 101e8: 4b050513 addi a0,a0,1200 # 254b0  
  ↪ <__clzsi2+0x50>  
26 101ec: 2cc000ef jal ra,104b8 <printf>  
27 101f0: 02c12083 lw ra,44(sp)  
28 101f4: 03010113 addi sp,sp,48  
29 101f8: 00008067 ret  
30  
31 000101fc <nth>:  
32 101fc: fe010113 addi sp,sp,-32  
33 10200: 00112e23 sw ra,28(sp)  
34 10204: 00812c23 sw s0,24(sp)  
35 10208: 00912a23 sw s1,20(sp)  
36 1020c: 01212823 sw s2,16(sp)  
37 10210: 01312623 sw s3,12(sp)  
38 10214: 00050913 mv s2,a0  
39 10218: fff60413 addi s0,a2,-1  
40 1021c: fff58493 addi s1,a1,-1  
41 10220: 00000993 li s3,0
```

42	10224:	0080006f	j	1022c <nth+0x30>
43	10228:	00050493	mv	s1,a0
44	1022c:	00048613	mv	a2,s1
45	10230:	00098593	mv	a1,s3
46	10234:	00090513	mv	a0,s2
47	10238:	03c000ef	jal	ra,10274
	↪	<partition>		
48	1023c:	00850863	beq	a0,s0,1024c
	↪	<nth+0x50>		
49	10240:	fea464e3	bltu	s0,a0,10228
	↪	<nth+0x2c>		
50	10244:	00150993	addi	s3,a0,1
51	10248:	fe5ff06f	j	1022c <nth+0x30>
52	1024c:	00241413	slli	s0,s0,0x2
53	10250:	00890933	add	s2,s2,s0
54	10254:	00092503	lw	a0,0(s2)
55	10258:	01c12083	lw	ra,28(sp)
56	1025c:	01812403	lw	s0,24(sp)
57	10260:	01412483	lw	s1,20(sp)
58	10264:	01012903	lw	s2,16(sp)
59	10268:	00c12983	lw	s3,12(sp)
60	1026c:	02010113	addi	sp,sp,32
61	10270:	00008067	ret	
62				
63	00010274	<partition>:		
64	10274:	00050313	mv	t1,a0
65	10278:	00060513	mv	a0,a2
66	1027c:	00c587b3	add	a5,a1,a2
67	10280:	0017d793	srli	a5,a5,0x1
68	10284:	00279793	slli	a5,a5,0x2
69	10288:	00f307b3	add	a5,t1,a5
70	1028c:	0007a603	lw	a2,0(a5)
71	10290:	04a5cc63	blt	a1,a0,102e8
	↪	<partition+0x74>		
72	10294:	00008067	ret	
73	10298:	00158593	addi	a1,a1,1
74	1029c:	00078693	mv	a3,a5
75	102a0:	00478793	addi	a5,a5,4
76	102a4:	ffc7a703	lw	a4,-4(a5)
77	102a8:	fec748e3	blt	a4,a2,10298
	↪	<partition+0x24>		
78	102ac:	00068813	mv	a6,a3
79	102b0:	00251793	slli	a5,a0,0x2
80	102b4:	00f308b3	add	a7,t1,a5
81	102b8:	0008a683	lw	a3,0(a7)
82	102bc:	02d65063	bge	a2,a3,102dc
	↪	<partition+0x68>		
83	102c0:	ffc78793	addi	a5,a5,-4
84	102c4:	00f307b3	add	a5,t1,a5
85	102c8:	fff50513	addi	a0,a0,-1
86	102cc:	00078893	mv	a7,a5
87	102d0:	ffc78793	addi	a5,a5,-4
88	102d4:	0047a683	lw	a3,4(a5)
89	102d8:	fed648e3	blt	a2,a3,102c8
	↪	<partition+0x54>		
90	102dc:	00d82023	sw	a3,0(a6)
91	102e0:	00e8a023	sw	a4,0(a7)
92	102e4:	02a5d063	bge	a1,a0,10304
	↪	<partition+0x90>		
93	102e8:	00259793	slli	a5,a1,0x2

```

94      102ec:      00f30833      add      a6,t1,a5
95      102f0:      00082703      lw       a4,0(a6)
96      102f4:      00478793      addi     a5,a5,4
97      102f8:      00f307b3      add      a5,t1,a5
98      102fc:      f8c74ee3      blt      a4,a2,10298
    ↪ <partition+0x24>
99      10300:      fb1ff06f      j        102b0
    ↪ <partition+0x3c>
100     10304:      00008067      ret
101 <..>

```

Листинг 15: objdump main.out

4.5. Выделение статических библиотек

Команды:

```

1 > riscv64-unknown-elf-ar -rcs libnth.a nth.o
2 > riscv64-unknown-elf-ar -rcs libpartition.a partition.o

```

Данные команды генерируют файлы libnth.a и libpartition.a

```

1 > riscv64-unknown-elf-objdump -d -t -S libnth.a

```

```

1 In archive libnth.a:
2
3 nth.o:      file format elf32-littleriscv
4
5 SYMBOL TABLE:
6 00000000 l    df *ABS*      00000000 nth.c
7 00000000 l    d  .text      00000000 .text
8 00000000 l    d  .data      00000000 .data
9 00000000 l    d  .bss      00000000 .bss
10 00000000 l    d  .comment  00000000 .comment
11 00000000 l    d  .riscv.attributes 00000000 .riscv.attributes
12 00000000 g    F  .text      0000004e nth
13 00000000      *UND*      00000000 partition
14
15
16
17 Disassembly of section .text:
18
19 00000000 <nth>:
20 0:      1101      addi     sp,sp,-32
21 2:      ce06      sw       ra,28(sp)
22 4:      cc22      sw       s0,24(sp)
23 6:      ca26      sw       s1,20(sp)
24 8:      c84a      sw       s2,16(sp)
25 a:      c64e      sw       s3,12(sp)
26 c:      892a      mv       s2,a0
27 e:      fff60413  addi     s0,a2,-1
28 12:     fff58493  addi     s1,a1,-1
29 16:     4981      li       s3,0

```



```

30      18:          a011                j          1c <.L2>
31
32      0000001a <.L5>:
33      1a:          84aa                mv          s1,a0
34
35      0000001c <.L2>:
36      1c:          8626                mv          a2,s1
37      1e:          85ce                mv          a1,s3
38      20:          854a                mv          a0,s2
39      22:          00000097            auipc       ra,0x0
40      26:          000080e7            jalr        ra # 22 <.L2+0x6>
41      2a:          00850763            beq         a0,s0,38 <.L7>
42      2e:          fea466e3            bltu        s0,a0,1a <.L5>
43      32:          00150993            addi        s3,a0,1
44      36:          b7dd                j          1c <.L2>
45
46      00000038 <.L7>:
47      38:          040a                slli        s0,s0,0x2
48      3a:          9922                add         s2,s2,s0
49      3c:          00092503            lw          a0,0(s2)
50      40:          40f2                lw          ra,28(sp)
51      42:          4462                lw          s0,24(sp)
52      44:          44d2                lw          s1,20(sp)
53      46:          4942                lw          s2,16(sp)
54      48:          49b2                lw          s3,12(sp)
55      4a:          6105                addi        sp,sp,32
56      4c:          8082                ret

```

Листинг 16: objdump libnth.a

Это позволяет нам при компоновке использовать сгенерированные файлы библиотек:

```

1  > riscv64-unknown-elf-gcc --save-temps -march=rv32i -mabi=ilp32 -O1 -E
   ↪ main.o -L. -lnth -lpartition -o main.out

```

Аргумент -L. здесь указывает в какой папке искать статические библиотеки. Аргументы вида -lnth означают, что надо искать библиотеку libnth.a

5. Создание Makefile

```

1  CC := gcc # default compiler for arm64
2  TARGET := arm64 # arm64 or rv32i are supported (default: arm64)
3  TARGET_ARCH_RISCV_ABI := ilp32 # default ABI for rv32i
4  TARGET_ARCH_ARM := arm64-apple-darwin22.2.0 # default target for arm64
5
6  CPREFIX := # prefix for compiler
7  CFLAGS := -O1 -save-temps -Wall --target=$(TARGET_ARCH_ARM) # default
   ↪ compiler options
8
9  ifeq ($(TARGET), rv32i)
10 CPREFIX := riscv64-unknown-elf-
11 CFLAGS := -save-temps -march=$(TARGET) -mabi=$(TARGET_ARCH_RISCV_ABI) -O1
   ↪ # compiler options for rv32i
12 endif
13

```

```

14 CC := $(CPREFIX)$(CC)
15 AR := $(CPREFIX)ar
16
17 all: clean build
18
19 build: main.o libpartition.a libnth.a
20     $(CC) $(CFLAGS) main.o -L. -lnth -lpartition -o main
21
22 main.o:
23     $(CC) $(CFLAGS) -c main.c
24
25 nth.o:
26     $(CC) $(CFLAGS) -c nth.c
27
28 libnth.a: nth.o
29     $(AR) rcs $@ $^
30
31 partition.o:
32     $(CC) $(CFLAGS) -c partition.c
33
34 libpartition.a: partition.o
35     $(AR) rcs $@ $^
36
37 clean:
38     rm -rf *.o *.s *.bc *.i *.a main

```

Листинг 17: Makefile

Тогда команды для сборки для разных платформ будут выглядеть так:

```

1 > make # for arm64
2 > make TARGET=rv32i # for riscv

```

5.1. Проверка

Соберем и запустим программу под arm64 (Тестовое устройство на Apple M1):

```

> make
rm -rf *.o *.s *.bc *.i *.a main
gcc -O1 -save-temps -Wall --target=arm64-apple-darwin22.2.0 -c main.c
main.c:4:1: warning: return type of 'main' is not 'int' [-Wmain-return-type]
void main() {
^
main.c:4:1: note: change return type to 'int'
void main() {
^~~~~~
int
1 warning generated.
gcc -O1 -save-temps -Wall --target=arm64-apple-darwin22.2.0 -c partition.c
ar rcs libpartition.a partition.o
gcc -O1 -save-temps -Wall --target=arm64-apple-darwin22.2.0 -c nth.c
ar rcs libnth.a nth.o
gcc -O1 -save-temps -Wall --target=arm64-apple-darwin22.2.0 main.o -L. -lnth -lpartition -o main
> ./main
36

```

6. Вывод

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