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

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

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. 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.

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

void main() {
  int n = 6; // array size
  int array[] = {12, 36, 110, 1, 6, 99}; // array
  int k = 4;
  int result = nth(array, n, k);
  printf("%i\n", result);
}
```

Листинг 1: main.c

```c
#ifndef NTH_H
#define NTH_H
int nth(int *array, unsigned int n, unsigned int k);
#endif
```

Листинг 2: nth.h

```c
#include "partition.h"

int nth(int *array, unsigned int n, unsigned int k) {
  k--;
  int left = 0;
  int right = n - 1;
  while (1) {
   int mid = partition(array, left, right);
```

```
    if (mid == k) {
     return array[mid];
    }
    if (k < mid) {
     right = mid;
    } else {
     left = mid + 1;
    }
   }
}
```

Листинг 3: nth.c

```
#ifndef PARTITION_H
#define PARTITION_H
int partition(int *array, unsigned int left, unsigned int right);
#endif
```

Листинг 4: partition.h

```
int partition(int *array, unsigned int left, unsigned int right) {
  int v = array[(left + right) / 2];
  int i = left;
  int j = right;
  while (i < j) {
    while (array[i] < v) {
      i++;
    }
    while (array[j] > v) {
      j--;
    }
    int t = array[i];
    array[i] = array[j];
    array[j] = t;
  }
  return j;
}
```

Листинг 5: partition.c

3. Проверка работы программы с использованием gcc

Соберем программу:

```
> gcc main.c nth.c partition.c
```

Исполним:

```
> ./a.out
36
```

4. Сборка программы

4.1. Препроцессинг

Команды:

```
> riscv64-unknown-elf-gcc --save-temps -march=rv32i -mabi=ilp32 -O1 -E
↪  main.c -o main.i
> riscv64-unknown-elf-gcc --save-temps -march=rv32i -mabi=ilp32 -O1 -E
↪  nth.c -o nth.i
> 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 "main.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "main.c"
# 2 "main.c" 2
# 1 "nth.h" 1

/* stdio.h */

# 3 "nth.h"
int nth(int *array, unsigned int n, unsigned int k);
# 3 "main.c" 2

void main() {
  int n = 6;
  int array[] = {12, 36, 110, 1, 6, 99};
  int k = 4;
  int result = nth(array, n, k);
  printf("%i\n", result);
}
```

Листинг 6: main.i

```
# 0 "nth.c"
# 0 "<built-in>"
# 0 "<command-line>"
# 1 "nth.c"
# 1 "partition.h" 1


int partition(int *array, unsigned int left, unsigned int right);
# 2 "nth.c" 2

int nth(int *array, unsigned int n, unsigned int k) {
  k--;
  int left = 0;
  int right = n - 1;
  while (1) {
```

```
    int mid = partition(array, left, right);
    if (mid == k) {
     return array[mid];
    }
    if (k < mid) {
     right = mid;
    } else {
     left = mid + 1;
    }
   }
}
```

Листинг 7: nth.i

```
# 0 "partition.c"
# 0 "<built-in>"
# 0 "<command-line>"
# 1 "partition.c"
int partition(int *array, unsigned int left, unsigned int right) {
  int v = array[(left + right) / 2];
  int i = left;
  int j = right;
  while (i < j) {
    while (array[i] < v) {
      i++;
    }
    while (array[j] > v) {
      j--;
    }
    int t = array[i];
    array[i] = array[j];
    array[j] = t;
  }
  return j;
}
```

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,

```
# 2 "nth.c" 2
```

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

4.2. Компиляция

Команды:

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

Эти команды генерируют код для ассемблера. Изучим их:

```
        .file   "main.c"
        .option nopic
        .attribute arch, "rv32i2p0"
        .attribute unaligned_access, 0
        .attribute stack_align, 16
        .text
        .section        .rodata.str1.4,"aMS",@progbits,1
        .align  2
.LC1:
        .string "%i\n"
        .text
        .align  2
        .globl  main
        .type   main, @function
main:
        addi    sp,sp,-48
        sw      ra,44(sp)
        lui     a5,%hi(.LANCHOR0)
        addi    a5,a5,%lo(.LANCHOR0)
        lw      a0,0(a5)
        lw      a1,4(a5)
        lw      a2,8(a5)
        lw      a3,12(a5)
        lw      a4,16(a5)
        lw      a5,20(a5)
        sw      a0,8(sp)
        sw      a1,12(sp)
        sw      a2,16(sp)
        sw      a3,20(sp)
        sw      a4,24(sp)
        sw      a5,28(sp)
        li      a2,4
        li      a1,6
        addi    a0,sp,8
        call    nth
        mv      a1,a0
        lui     a0,%hi(.LC1)
        addi    a0,a0,%lo(.LC1)
        call    printf
        lw      ra,44(sp)
        addi    sp,sp,48
        jr      ra
        .size   main, .-main
        .section        .rodata
        .align  2
        .set    .LANCHOR0,. + 0
```

```
.LC0:
        .word   12
        .word   36
        .word   110
        .word   1
        .word   6
        .word   99
        .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.

```
        .file   "nth.c"
        .option nopic
        .attribute arch, "rv32i2p0"
        .attribute unaligned_access, 0
        .attribute stack_align, 16
        .text
        .align  2
        .globl  nth
        .type   nth, @function
nth:
        addi    sp,sp,-32
        sw      ra,28(sp)
        sw      s0,24(sp)
        sw      s1,20(sp)
        sw      s2,16(sp)
        sw      s3,12(sp)
        mv      s2,a0
        addi    s0,a2,-1
        addi    s1,a1,-1
        li      s3,0
        j       .L2
.L5:
        mv      s1,a0
.L2:
        mv      a2,s1
        mv      a1,s3
        mv      a0,s2
        call    partition
        beq     a0,s0,.L7
        bgtu    a0,s0,.L5
        addi    s3,a0,1
        j       .L2
.L7:
        slli    s0,s0,2
        add     s2,s2,s0
        lw      a0,0(s2)
        lw      ra,28(sp)
        lw      s0,24(sp)
        lw      s1,20(sp)
        lw      s2,16(sp)
        lw      s3,12(sp)
        addi    sp,sp,32
        jr      ra
```

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

Листинг 10: nth.s

```
        .file   "partition.c"
        .option nopic
        .attribute arch, "rv32i2p0"
        .attribute unaligned_access, 0
        .attribute stack_align, 16
        .text
        .align  2
        .globl  partition
        .type   partition, @function
partition:
        mv      t1,a0
        mv      a0,a2
        add     a5,a1,a2
        srli    a5,a5,1
        slli    a5,a5,2
        add     a5,t1,a5
        lw      a2,0(a5)
        blt     a1,a0,.L2
        ret
.L4:
        addi    a1,a1,1
        mv      a3,a5
        addi    a5,a5,4
        lw      a4,-4(a5)
        blt     a4,a2,.L4
        mv      a6,a3
.L8:
        slli    a5,a0,2
        add     a7,t1,a5
        lw      a3,0(a7)
        bge     a2,a3,.L5
        addi    a5,a5,-4
        add     a5,t1,a5
.L6:
        addi    a0,a0,-1
        mv      a7,a5
        addi    a5,a5,-4
        lw      a3,4(a5)
        bgt     a3,a2,.L6
.L5:
        sw      a3,0(a6)
        sw      a4,0(a7)
        bge     a1,a0,.L1
.L2:
        slli    a5,a1,2
        add     a6,t1,a5
        lw      a4,0(a6)
        addi    a5,a5,4
        add     a5,t1,a5
        bgt     a2,a4,.L4
        j       .L8
.L1:
        ret
```

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

Листинг 11: partition.s

4.3. Сборка объектов

Команды:

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

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

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

Исследуем объектные файлы:

```

main.o:     file format elf32-littleriscv

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000074  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000a8  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000a8  2**0
                  ALLOC
  3 .rodata.str1.4 00000004  00000000  00000000  000000a8  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .rodata       00000018  00000000  00000000  000000ac  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .comment      00000022  00000000  00000000  000000c4  2**0
                  CONTENTS, READONLY
  6 .riscv.attributes 0000001c  00000000  00000000  000000e6  2**0
                  CONTENTS, READONLY
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.str1.4        00000000 .rodata.str1.4
00000000 l    d  .rodata        00000000 .rodata
00000000 l    d  .comment        00000000 .comment
00000000 l    d  .riscv.attributes        00000000 .riscv.attributes
00000000 g     F .text        00000074 main
00000000         *UND*        00000000 nth
```

```
00000000         *UND*        00000000 printf


Disassembly of section .text:

00000000 <main>:
   0:        fd010113                    addi        sp,sp,-48
   4:        02112623                    sw        ra,44(sp)
   8:        000007b7                    lui        a5,0×0
   c:        00078793                    mv        a5,a5
  10:        0007a503                    lw        a0,0(a5) # 0 <main>
  14:        0047a583                    lw        a1,4(a5)
  18:        0087a603                    lw        a2,8(a5)
  1c:        00c7a683                    lw        a3,12(a5)
  20:        0107a703                    lw        a4,16(a5)
  24:        0147a783                    lw        a5,20(a5)
  28:        00a12423                    sw        a0,8(sp)
  2c:        00b12623                    sw        a1,12(sp)
  30:        00c12823                    sw        a2,16(sp)
  34:        00d12a23                    sw        a3,20(sp)
  38:        00e12c23                    sw        a4,24(sp)
  3c:        00f12e23                    sw        a5,28(sp)
  40:        00400613                    li        a2,4
  44:        00600593                    li        a1,6
  48:        00810513                    addi        a0,sp,8
  4c:        00000097                    auipc        ra,0×0
  50:        000080e7                    jalr        ra # 4c <main+0×4c>
  54:        00050593                    mv        a1,a0
  58:        00000537                    lui        a0,0×0
  5c:        00050513                    mv        a0,a0
  60:        00000097                    auipc        ra,0×0
  64:        000080e7                    jalr        ra # 60 <main+0×60>
  68:        02c12083                    lw        ra,44(sp)
  6c:        03010113                    addi        sp,sp,48
  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.

```

nth.o:     file format elf32-littleriscv

SYMBOL TABLE:
00000000 l    df *ABS*        00000000 nth.c
00000000 l    d  .text        00000000 .text
00000000 l    d  .data        00000000 .data
00000000 l    d  .bss        00000000 .bss
00000000 l    d  .comment        00000000 .comment
00000000 l    d  .riscv.attributes        00000000 .riscv.attributes
```

```
00000000 g     F .text        0000007c nth
00000000         *UND*        00000000 partition



Disassembly of section .text:

00000000 <nth>:
   0:        fe010113                addi        sp,sp,-32
   4:        00112e23                sw        ra,28(sp)
   8:        00812c23                sw        s0,24(sp)
   c:        00912a23                sw        s1,20(sp)
  10:        01212823                sw        s2,16(sp)
  14:        01312623                sw        s3,12(sp)
  18:        00050913                mv        s2,a0
  1c:        fff60413                addi        s0,a2,-1
  20:        fff58493                addi        s1,a1,-1
  24:        00000993                li        s3,0
  28:        0080006f                j        30 <.L2>

0000002c <.L5>:
  2c:        00050493                mv        s1,a0

00000030 <.L2>:
  30:        00048613                mv        a2,s1
  34:        00098593                mv        a1,s3
  38:        00090513                mv        a0,s2
  3c:        00000097                auipc        ra,0x0
  40:        000080e7                jalr        ra # 3c <.L2+0xc>
  44:        00850863                beq        a0,s0,54 <.L7>
  48:        fea462e3                bltu        s0,a0,2c <.L5>
  4c:        00150993                addi        s3,a0,1
  50:        fe1ff06f                j        30 <.L2>

00000054 <.L7>:
  54:        00241413                slli        s0,s0,0x2
  58:        00890933                add        s2,s2,s0
  5c:        00092503                lw        a0,0(s2)
  60:        01c12083                lw        ra,28(sp)
  64:        01812403                lw        s0,24(sp)
  68:        01412483                lw        s1,20(sp)
  6c:        01012903                lw        s2,16(sp)
  70:        00c12983                lw        s3,12(sp)
  74:        02010113                addi        sp,sp,32
  78:        00008067                ret
```

Листинг 13: nth.o

```

partition.o:     file format elf32-littleriscv

SYMBOL TABLE:
00000000 l    df *ABS*        00000000 partition.c
00000000 l    d  .text        00000000 .text
00000000 l    d  .data        00000000 .data
00000000 l    d  .bss        00000000 .bss
00000000 l    d  .comment        00000000 .comment
00000000 l    d  .riscv.attributes        00000000 .riscv.attributes
```

```
00000000 g     F .text        00000094 partition


Disassembly of section .text:

00000000 <partition>:
   0:        00050313                mv        t1,a0
   4:        00060513                mv        a0,a2
   8:        00c587b3                add        a5,a1,a2
   c:        0017d793                srli        a5,a5,0×1
  10:        00279793                slli        a5,a5,0×2
  14:        00f307b3                add        a5,t1,a5
  18:        0007a603                lw        a2,0(a5)
  1c:        04a5cc63                blt        a1,a0,74 <.L2>
  20:        00008067                ret

00000024 <.L4>:
  24:        00158593                addi        a1,a1,1
  28:        00078693                mv        a3,a5
  2c:        00478793                addi        a5,a5,4
  30:        ffc7a703                lw        a4,-4(a5)
  34:        fec748e3                blt        a4,a2,24 <.L4>
  38:        00068813                mv        a6,a3

0000003c <.L8>:
  3c:        00251793                slli        a5,a0,0×2
  40:        00f308b3                add        a7,t1,a5
  44:        0008a683                lw        a3,0(a7)
  48:        02d65063                bge        a2,a3,68 <.L5>
  4c:        ffc78793                addi        a5,a5,-4
  50:        00f307b3                add        a5,t1,a5

00000054 <.L6>:
  54:        fff50513                addi        a0,a0,-1
  58:        00078893                mv        a7,a5
  5c:        ffc78793                addi        a5,a5,-4
  60:        0047a683                lw        a3,4(a5)
  64:        fed648e3                blt        a2,a3,54 <.L6>

00000068 <.L5>:
  68:        00d82023                sw        a3,0(a6)
  6c:        00e8a023                sw        a4,0(a7)
  70:        02a5d063                bge        a1,a0,90 <.L1>

00000074 <.L2>:
  74:        00259793                slli        a5,a1,0×2
  78:        00f30833                add        a6,t1,a5
  7c:        00082703                lw        a4,0(a6)
  80:        00478793                addi        a5,a5,4
```

Листинг 14: partition.o

4.4. Компоновка

Команда:

```
> riscv64-unknown-elf-gcc --save-temps -march=rv32i -mabi=ilp32 -O1 -E
↪  main.o nth.o partition.o -o main.out


> 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:

```
<..>
00010190 <main>:
   10190:        fd010113                addi        sp,sp,-48
   10194:        02112623                sw        ra,44(sp)
   10198:        000257b7                lui        a5,0×25
   1019c:        4b478793                addi        a5,a5,1204 # 254b4
   ↪  <__clzsi2+0×54>
   101a0:        0007a503                lw        a0,0(a5)
   101a4:        0047a583                lw        a1,4(a5)
   101a8:        0087a603                lw        a2,8(a5)
   101ac:        00c7a683                lw        a3,12(a5)
   101b0:        0107a703                lw        a4,16(a5)
   101b4:        0147a783                lw        a5,20(a5)
   101b8:        00a12423                sw        a0,8(sp)
   101bc:        00b12623                sw        a1,12(sp)
   101c0:        00c12823                sw        a2,16(sp)
   101c4:        00d12a23                sw        a3,20(sp)
   101c8:        00e12c23                sw        a4,24(sp)
   101cc:        00f12e23                sw        a5,28(sp)
   101d0:        00400613                li        a2,4
   101d4:        00600593                li        a1,6
   101d8:        00810513                addi        a0,sp,8
   101dc:        020000ef                jal        ra,101fc <nth>
   101e0:        00050593                mv        a1,a0
   101e4:        00025537                lui        a0,0×25
   101e8:        4b050513                addi        a0,a0,1200 # 254b0
   ↪  <__clzsi2+0×50>
   101ec:        2cc000ef                jal        ra,104b8 <printf>
   101f0:        02c12083                lw        ra,44(sp)
   101f4:        03010113                addi        sp,sp,48
   101f8:        00008067                ret

000101fc <nth>:
   101fc:        fe010113                addi        sp,sp,-32
   10200:        00112e23                sw        ra,28(sp)
   10204:        00812c23                sw        s0,24(sp)
   10208:        00912a23                sw        s1,20(sp)
   1020c:        01212823                sw        s2,16(sp)
   10210:        01312623                sw        s3,12(sp)
   10214:        00050913                mv        s2,a0
   10218:        fff60413                addi        s0,a2,-1
   1021c:        fff58493                addi        s1,a1,-1
   10220:        00000993                li        s3,0
```

```
   10224:        0080006f                j        1022c <nth+0×30>
   10228:        00050493                mv        s1,a0
   1022c:        00048613                mv        a2,s1
   10230:        00098593                mv        a1,s3
   10234:        00090513                mv        a0,s2
   10238:        03c000ef                jal        ra,10274
    ↪ <partition>
   1023c:        00850863                beq        a0,s0,1024c
    ↪ <nth+0×50>
   10240:        fea464e3                bltu        s0,a0,10228
    ↪ <nth+0×2c>
   10244:        00150993                addi        s3,a0,1
   10248:        fe5ff06f                j        1022c <nth+0×30>
   1024c:        00241413                slli        s0,s0,0×2
   10250:        00890933                add        s2,s2,s0
   10254:        00092503                lw        a0,0(s2)
   10258:        01c12083                lw        ra,28(sp)
   1025c:        01812403                lw        s0,24(sp)
   10260:        01412483                lw        s1,20(sp)
   10264:        01012903                lw        s2,16(sp)
   10268:        00c12983                lw        s3,12(sp)
   1026c:        02010113                addi        sp,sp,32
   10270:        00008067                ret

00010274 <partition>:
   10274:        00050313                mv        t1,a0
   10278:        00060513                mv        a0,a2
   1027c:        00c587b3                add        a5,a1,a2
   10280:        0017d793                srli        a5,a5,0×1
   10284:        00279793                slli        a5,a5,0×2
   10288:        00f307b3                add        a5,t1,a5
   1028c:        0007a603                lw        a2,0(a5)
   10290:        04a5cc63                blt        a1,a0,102e8
    ↪ <partition+0×74>
   10294:        00008067                ret
   10298:        00158593                addi        a1,a1,1
   1029c:        00078693                mv        a3,a5
   102a0:        00478793                addi        a5,a5,4
   102a4:        ffc7a703                lw        a4,-4(a5)
   102a8:        fec748e3                blt        a4,a2,10298
    ↪ <partition+0×24>
   102ac:        00068813                mv        a6,a3
   102b0:        00251793                slli        a5,a0,0×2
   102b4:        00f308b3                add        a7,t1,a5
   102b8:        0008a683                lw        a3,0(a7)
   102bc:        02d65063                bge        a2,a3,102dc
    ↪ <partition+0×68>
   102c0:        ffc78793                addi        a5,a5,-4
   102c4:        00f307b3                add        a5,t1,a5
   102c8:        fff50513                addi        a0,a0,-1
   102cc:        00078893                mv        a7,a5
   102d0:        ffc78793                addi        a5,a5,-4
   102d4:        0047a683                lw        a3,4(a5)
   102d8:        fed648e3                blt        a2,a3,102c8
    ↪ <partition+0×54>
   102dc:        00d82023                sw        a3,0(a6)
   102e0:        00e8a023                sw        a4,0(a7)
   102e4:        02a5d063                bge        a1,a0,10304
    ↪ <partition+0×90>
   102e8:        00259793                slli        a5,a1,0×2
```

```
   102ec:        00f30833                add        a6,t1,a5
   102f0:        00082703                lw        a4,0(a6)
   102f4:        00478793                addi        a5,a5,4
   102f8:        00f307b3                add        a5,t1,a5
   102fc:        f8c74ee3                blt        a4,a2,10298
    ↪  <partition+0×24>
   10300:        fb1ff06f                j        102b0
    ↪  <partition+0×3c>
   10304:        00008067                ret
<..>
```

Листинг 15: objdump main.out

4.5. Выделение статических библиотек

Команды:

```
> ar -rcs libnth.a partition.o nth.o
```

Данная команда генерирует файл libnth.a

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

```
In archive libnth.a:

nth.o:     file format elf32-littleriscv

SYMBOL TABLE:
00000000 l    df *ABS*        00000000 nth.c
00000000 l    d  .text        00000000 .text
00000000 l    d  .data        00000000 .data
00000000 l    d  .bss        00000000 .bss
00000000 l    d  .comment        00000000 .comment
00000000 l    d  .riscv.attributes        00000000 .riscv.attributes
00000000 g     F .text        0000004e nth
00000000         *UND*        00000000 partition



Disassembly of section .text:

00000000 <nth>:
   0:        1101                      addi        sp,sp,-32
   2:        ce06                      sw        ra,28(sp)
   4:        cc22                      sw        s0,24(sp)
   6:        ca26                      sw        s1,20(sp)
   8:        c84a                      sw        s2,16(sp)
   a:        c64e                      sw        s3,12(sp)
   c:        892a                      mv        s2,a0
   e:        fff60413                addi        s0,a2,-1
  12:        fff58493                addi        s1,a1,-1
  16:        4981                      li        s3,0
  18:        a011                      j        1c <.L2>
```

```
0000001a <.L5>:
  1a:        84aa                      mv        s1,a0

0000001c <.L2>:
  1c:        8626                      mv        a2,s1
  1e:        85ce                      mv        a1,s3
  20:        854a                      mv        a0,s2
  22:        00000097                auipc        ra,0x0
  26:        000080e7                jalr        ra # 22 <.L2+0x6>
  2a:        00850763                beq        a0,s0,38 <.L7>
  2e:        fea466e3                bltu        s0,a0,1a <.L5>
  32:        00150993                addi        s3,a0,1
  36:        b7dd                      j        1c <.L2>

00000038 <.L7>:
  38:        040a                      slli        s0,s0,0x2
  3a:        9922                      add        s2,s2,s0
  3c:        00092503                lw        a0,0(s2)
  40:        40f2                      lw        ra,28(sp)
  42:        4462                      lw        s0,24(sp)
  44:        44d2                      lw        s1,20(sp)
  46:        4942                      lw        s2,16(sp)
  48:        49b2                      lw        s3,12(sp)
  4a:        6105                      addi        sp,sp,32
  4c:        8082                      ret
```

Листинг 16: objdump libnth.a

Это позволяет нам при компоновке использовать сгенерированные файлы библиотек:

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

Аргумент -L. здесь указывает в какой папке искать статические библиотеки.

Аргументы вида -lnth означают, что надо искать библиотеку libnth.a (По принятому соглашению)

5. Создание Makefile

```
CC := gcc # default compiler for arm64
TARGET := arm64 # arm64 or rv32i are supported (default: arm64)
TARGET_ARCH_RISCV_ABI := ilp32 # default ABI for rv32i
TARGET_ARCH_ARM := arm64-apple-darwin22.2.0 # default target for arm64

CPREFIX := # prefix for compiler
CFLAGS := -O1 -save-temps -Wall --target=$(TARGET_ARCH_ARM) # default
↪  compiler options

ifeq ($(TARGET), rv32i)
CPREFIX := riscv64-unknown-elf-
CFLAGS := -save-temps -march=$(TARGET) -mabi=$(TARGET_ARCH_RISCV_ABI) -O1
↪  # compiler options for rv32i
endif
```

```
13
14  CC := $(CPREFIX)$(CC)
15  AR := $(CPREFIX)ar
16
17  all: clean build
18
19  build: main.o libnth.a
20          $(CC) $(CFLAGS) main.o -L. -lnth -o main
21
22  main.o: main.c
23          $(CC) $(CFLAGS) -c main.c
24
25  nth.o: nth.c
26          $(CC) $(CFLAGS) -c nth.c
27
28  partition.o: partition.c
29          $(CC) $(CFLAGS) -c partition.c
30
31  libnth.a: partition.o nth.o
32          $(AR) rcs $@ $^
33
34  clean:
35          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
gcc  -O1 -save-temps -Wall --target=arm64-apple-darwin22.2.0   -c nth.c
ar rcs libnth.a partition.o nth.o
gcc  -O1 -save-temps -Wall --target=arm64-apple-darwin22.2.0   main.o -L. -lnth -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.