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

Лабораторная работа № 4

Раздельная компиляция

по дисциплине «Низкоуровневое программирование»

Выполнил
студент гр. 3530901/10002

_____ Котов К. А.
(подпись)

Руководитель

_____ Максименко С. Л
(подпись)

«___» _____ 2022 г.

Санкт-Петербург
2022

Задачи

Реализовать заданную вариантом программу на языке С, содержащую заголовочный файл, файл тестирования и исходный файл алгоритма. Пошагово собрать программу. Проанализировать выходы предпроцессора, компилятора, составы и содержимое секций таблиц символов и таблиц перемещений, отладочную информацию объектных и исполняемых файлов. Создать статическую библиотеку разработанной функции, разработав make-файлы. Проанализировать ход сборки библиотеки и программы, созданные файлы зависимостей.

Вариант

Реализовать алгоритм нахождения биномиальных коэффициентов по треугольнику Паскаля.

Решение программой на языке С

Далее представлено содержимое файлов тестирования, алгоритма и заголовочного файла.

Файл main.c – содержит тестирование.

```c
#include <stdlib.h>
#include <stdio.h>
#include "pascal.h"

int main() {
    int pokaz = -1;
    int *array = calloc(pokaz + 1, sizeof(int));
    pascal(pokaz, array);
    for(int i = 0; i <= pokaz; i++) {
        printf("%d\t", *(array + i));
    }

    printf("\n");

    pokaz = 0;
    int *array1 = calloc(pokaz + 1, sizeof(int));
    pascal(pokaz, array1);
    for (size_t i = 0; i <= pokaz; i++) {
```

```
        printf("%d\t", *(array1 + i));
    }

    printf("\n");

    pokaz = 1;
    int *array2 = calloc(pokaz + 1, sizeof(int));
    pascal(pokaz, array2);
    for (size_t i = 0; i <= pokaz; i++) {
        printf("%d\t", *(array2 + i));
    }

    printf("\n");

    pokaz = 10;
    int *array3 = calloc(pokaz + 1, sizeof(int));
    pascal(pokaz, array3);
    for (size_t i = 0; i <= pokaz; i++) {
        printf("%d\t", *(array3 + i));
    }

    return 0;
}
```

Файл pascal.h – заголовочный файл

```
#include <stddef.h>
int pascal(int pokaz, int *array);
```

Файл pascal.c – алгоритм

```
#include "pascal.h"
#include <stdio.h>

int pascal(int pokaz, int *array) {
    if (pokaz < 0) {
        printf("pokaz must be greater than zero");
        return -1;
    }
    *array = 1;
    int len = 2;
    int* ptr = array;
    while (pokaz > 0) {
        for (int i = len - 1; i >= 1; i--) {
            *(ptr + i) = *(ptr + i - 1) + *(ptr + i);
        }
        pokaz--;
        len++;
    }
    return 0;
}
```

Результат работы программы показан на рис. 1

```
pokaz must be greater than zero
1
1       1
1       10      45      120     210     252     210     120     45      10      1
```

Рис.1 – Результат работы программы

Раздельная компиляция

1. **Предпроцессирование.** Предпроцессирование выполняется следующими командами.

```
>riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -E main.c -o main.i
>riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -E pascal.c -o
pascal.i
```

Результаты работы команд находятся в файлах main.i и pascal.i соответственно.

Файл main.i:

```
...
# 2 "pascal.h"
int pascal(int pokaz, int *array);
# 4 "main.c" 2

int main() {
    int pokaz = -1;
    int *array = calloc(pokaz + 1, sizeof(int));
    pascal(pokaz, array);
    for(int i = 0; i <= pokaz; i++) {
        printf("%d\t", *(array + i));
    }

    printf("\n");

    pokaz = 0;
    int *array1 = calloc(pokaz + 1, sizeof(int));
    pascal(pokaz, array1);
    for (size_t i = 0; i <= pokaz; i++) {
        printf("%d\t", *(array1 + i));
    }

    printf("\n");

    pokaz = 1;
    int *array2 = calloc(pokaz + 1, sizeof(int));
    pascal(pokaz, array2);
```

```
    for (size_t i = 0; i <= pokaz; i++) {
        printf("%d\t", *(array2 + i));
    }

    printf("\n");

    pokaz = 10;
    int *array3 = calloc(pokaz + 1, sizeof(int));
    pascal(pokaz, array3);
    for (size_t i = 0; i <= pokaz; i++) {
        printf("%d\t", *(array3 + i));
    }

    return 0;
}
```

Файл pascal.i:

```
...
# 2 "pascal.h"
int pascal(int pokaz, int *array);
# 3 "pascal.c" 2


int pascal(int pokaz, int *array) {
    if (pokaz < 0) {
        printf("pokaz must be greater than zero");
        return -1;
    }
    *array = 1;
    int len = 2;
    int* ptr = array;
    while (pokaz > 0) {
        for (int i = len - 1; i >= 1; i--) {
            *(ptr + i) = *(ptr + i - 1) + *(ptr + i);
        }
        pokaz--;
        len++;
    }
    return 0;
}
```

В прикрепленных листингах мы видим, что при предпроцессировании в обработанных файлах появилось определение функции pascal, которое появилось из заголовочного файла pascal.h. Кроме этого полученные файлы содержат много строк кода, которые появились в связи со включением в программу заголовочных файлов stdio.h и stdlib.h, которые позволяют нам

использовать дополнительные функции, в нашем случае printf() и calloc().

2. Компиляция

Компиляция осуществляется следующими командами:

```
>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 pascal.i -o pascal.s
```

Коды соответствующих выходных файлов представлены далее.

Файл main.s:

```
    .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
.LC0:
    .string "%d\t"
    .text
    .align  2
    .globl  main
    .type   main, @function
main:
    addi    sp,sp,-16
    sw  ra,12(sp)
    sw  s0,8(sp)
    sw  s1,4(sp)
    sw  s2,0(sp)
    li  a1,4
    li  a0,0
    call    calloc
    mv  a1,a0
    li  a0,-1
    call    pascal
    li  a0,10
    call    putchar
    li  a1,4
    li  a0,1
    call    calloc
    mv  s0,a0
    mv  a1,a0
    li  a0,0
    call    pascal
    lw  a1,0(s0)
    lui s1,%hi(.LC0)
    addi    a0,s1,%lo(.LC0)
```

```
    call    printf
    li  a0,10
    call    putchar
    li  a1,4
    li  a0,2
    call    calloc
    mv  s0,a0
    mv  a1,a0
    li  a0,1
    call    pascal
    lw  a1,0(s0)
    addi    a0,s1,%lo(.LC0)
    call    printf
    lw  a1,4(s0)
    addi    a0,s1,%lo(.LC0)
    call    printf
    li  a0,10
    call    putchar
    li  a1,4
    li  a0,11
    call    calloc
    mv  s1,a0
    mv  a1,a0
    li  a0,10
    call    pascal
    mv  s0,s1
    addi    s1,s1,44
    lui s2,%hi(.LC0)
.L2:
    lw  a1,0(s0)
    addi    a0,s2,%lo(.LC0)
    call    printf
    addi    s0,s0,4
    bne s1,s0,.L2
    li  a0,0
    lw  ra,12(sp)
    lw  s0,8(sp)
    lw  s1,4(sp)
    lw  s2,0(sp)
    addi    sp,sp,16
    jr  ra
    .size   main, .-main
    .ident  "GCC: (SiFive GCC 10.1.0-2020.08.2) 10.1.0"
```

Файл pascal.s:

```
    .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
.LC0:
    .string "%d\t"
    .text
    .align  2
    .globl  main
    .type   main, @function
main:
    addi    sp,sp,-16
    sw  ra,12(sp)
    sw  s0,8(sp)
    sw  s1,4(sp)
    sw  s2,0(sp)
    li  a1,4
    li  a0,0
    call    calloc
    mv  a1,a0
    li  a0,-1
    call    pascal
    li  a0,10
    call    putchar
    li  a1,4
    li  a0,1
    call    calloc
    mv  s0,a0
    mv  a1,a0
    li  a0,0
    call    pascal
    lw  a1,0(s0)
    lui s1,%hi(.LC0)
    addi    a0,s1,%lo(.LC0)
    call    printf
    li  a0,10
    call    putchar
    li  a1,4
    li  a0,2
    call    calloc
    mv  s0,a0
    mv  a1,a0
    li  a0,1
    call    pascal
    lw  a1,0(s0)
    addi    a0,s1,%lo(.LC0)
    call    printf
    lw  a1,4(s0)
    addi    a0,s1,%lo(.LC0)
    call    printf
    li  a0,10
    call    putchar
```

```
    li  a1,4
    li  a0,11
    call    calloc
    mv  s1,a0
    mv  a1,a0
    li  a0,10
    call    pascal
    mv  s0,s1
    addi    s1,s1,44
    lui s2,%hi(.LC0)
.L2:
    lw  a1,0(s0)
    addi    a0,s2,%lo(.LC0)
    call    printf
    addi    s0,s0,4
    bne s1,s0,.L2
    li  a0,0
    lw  ra,12(sp)
    lw  s0,8(sp)
    lw  s1,4(sp)
    lw  s2,0(sp)
    addi    sp,sp,16
    jr  ra
    .size   main, .-main
    .ident  "GCC: (SiFive GCC 10.1.0-2020.08.2) 10.1.0"
```

3. Ассемблирование

Ассемблирование выполняется следующими командами:

```
>riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -c main.s -o main.o
>riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -c pascal.s -o pascal.o
```

Файлы main.o и pascal.o содержат в себе машинный код. Для того, чтобы его анализировать вызовем заголовки секций, таблицу символов и таблицу перемещений для main.o.

Используются соответствующие команды:

```
riscv64-unknown-elf-objdump -t pascal.o main.o
riscv64-unknown-elf-objdump –d –M no-aliases –j .text pascal.o main.o
riscv64-unknown-elf-objdump -r pascal.o main.o
```

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

SYMBOL TABLE:
00000000 l    df *ABS*  00000000 pascal.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       .rodata.str1.4 00000000 .LC0
00000024 l       .text  00000000 .L14
00000084 l       .text  00000000 .L7
00000050 l       .text  00000000 .L6
00000070 l       .text  00000000 .L4
00000058 l       .text  00000000 .L5
0000004c l       .text  00000000 .L8
00000000 l    d  .comment       00000000 .comment
00000000 l    d  .riscv.attributes      00000000 .riscv.attributes
00000000 g     F .text  0000008c pascal
00000000         *UND*  00000000 printf


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.str1.4 00000000 .rodata.str1.4
00000000 l       .rodata.str1.4 00000000 .LC0
00000104 l       .text  00000000 .L2
00000000 l    d  .comment       00000000 .comment
00000000 l    d  .riscv.attributes      00000000 .riscv.attributes
00000000 g     F .text  00000138 main
00000000         *UND*  00000000 calloc
00000000         *UND*  00000000 pascal
00000000         *UND*  00000000 putchar
00000000         *UND*  00000000 printf
```

Рис.2 – таблицы символов файлов pascal.o и main.o соответственно

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


Disassembly of section .text:

00000000 <pascal>:
   0:   02054263                blt     a0,zero,24 <.L14>
   4:   00050893                addi    a7,a0,0
   8:   00100793                addi    a5,zero,1
   c:   00f5a023                sw      a5,0(a1)
  10:   06a05a63                bge     zero,a0,84 <.L7>
  14:   00858813                addi    a6,a1,8
  18:   00100513                addi    a0,zero,1
  1c:   00458613                addi    a2,a1,4
  20:   0300006f                jal     zero,50 <.L6>

00000024 <.L14>:
  24:   ff010113                addi    sp,sp,-16
  28:   00112623                sw      ra,12(sp)
  2c:   00000537                lui     a0,0x0
  30:   00050513                addi    a0,a0,0 # 0 <pascal>
  34:   00000097                auipc   ra,0x0
  38:   000080e7                jalr    ra,0(ra) # 34 <.L14+0x10>
  3c:   fff00513                addi    a0,zero,-1
  40:   00c12083                lw      ra,12(sp)
  44:   01010113                addi    sp,sp,16
  48:   00008067                jalr    zero,0(ra)

0000004c <.L8>:
  4c:   00078513                addi    a0,a5,0

00000050 <.L6>:
  50:   02a05063                bge     zero,a0,70 <.L4>
  54:   00080793                addi    a5,a6,0

00000058 <.L5>:
  58:   ffc7a703                lw      a4,-4(a5)
  5c:   ff87a683                lw      a3,-8(a5)
  60:   00d70733                add     a4,a4,a3
  64:   fee7ae23                sw      a4,-4(a5)
  68:   ffc78793                addi    a5,a5,-4
  6c:   fec796e3                bne     a5,a2,58 <.L5>
```

Рис 3.1 – содержимое секции .text файла pascal.o

```
00000070 <.L4>:
  70:   00150793                addi    a5,a0,1
  74:   00480813                addi    a6,a6,4
  78:   fd151ae3                bne     a0,a7,4c <.L8>
  7c:   00000513                addi    a0,zero,0
  80:   00008067                jalr    zero,0(ra)

00000084 <.L7>:
  84:   00000513                addi    a0,zero,0
  88:   00008067                jalr    zero,0(ra)
```

Рис. 3.2 – содержимое секции .text файла pascal.o

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


Disassembly of section .text:

00000000 <main>:
   0:   ff010113                addi    sp,sp,-16
   4:   00112623                sw      ra,12(sp)
   8:   00812423                sw      s0,8(sp)
   c:   00912223                sw      s1,4(sp)
  10:   01212023                sw      s2,0(sp)
  14:   00400593                addi    a1,zero,4
  18:   00000513                addi    a0,zero,0
  1c:   00000097                auipc   ra,0x0
  20:   000080e7                jalr    ra,0(ra) # 1c <main+0x1c>
  24:   00050593                addi    a1,a0,0
  28:   fff00513                addi    a0,zero,-1
  2c:   00000097                auipc   ra,0x0
  30:   000080e7                jalr    ra,0(ra) # 2c <main+0x2c>
  34:   00a00513                addi    a0,zero,10
  38:   00000097                auipc   ra,0x0
  3c:   000080e7                jalr    ra,0(ra) # 38 <main+0x38>
  40:   00400593                addi    a1,zero,4
  44:   00100513                addi    a0,zero,1
  48:   00000097                auipc   ra,0x0
  4c:   000080e7                jalr    ra,0(ra) # 48 <main+0x48>
  50:   00050413                addi    s0,a0,0
  54:   00050593                addi    a1,a0,0
  58:   00000513                addi    a0,zero,0
  5c:   00000097                auipc   ra,0x0
  60:   000080e7                jalr    ra,0(ra) # 5c <main+0x5c>
  64:   00042583                lw      a1,0(s0)
  68:   000004b7                lui     s1,0x0
  6c:   00048513                addi    a0,s1,0 # 0 <main>
  70:   00000097                auipc   ra,0x0
  74:   000080e7                jalr    ra,0(ra) # 70 <main+0x70>
  78:   00a00513                addi    a0,zero,10
  7c:   00000097                auipc   ra,0x0
  80:   000080e7                jalr    ra,0(ra) # 7c <main+0x7c>
  84:   00400593                addi    a1,zero,4
  88:   00200513                addi    a0,zero,2
  8c:   00000097                auipc   ra,0x0
  90:   000080e7                jalr    ra,0(ra) # 8c <main+0x8c>
```

Рис 4.1 – содержимое секции .text файла main.o

```
  94:   00050413                addi    s0,a0,0
  98:   00050593                addi    a1,a0,0
  9c:   00100513                addi    a0,zero,1
  a0:   00000097                auipc   ra,0x0
  a4:   000080e7                jalr    ra,0(ra) # a0 <main+0xa0>
  a8:   00042583                lw      a1,0(s0)
  ac:   00048513                addi    a0,s1,0
  b0:   00000097                auipc   ra,0x0
  b4:   000080e7                jalr    ra,0(ra) # b0 <main+0xb0>
  b8:   00442583                lw      a1,4(s0)
  bc:   00048513                addi    a0,s1,0
  c0:   00000097                auipc   ra,0x0
  c4:   000080e7                jalr    ra,0(ra) # c0 <main+0xc0>
  c8:   00a00513                addi    a0,zero,10
  cc:   00000097                auipc   ra,0x0
  d0:   000080e7                jalr    ra,0(ra) # cc <main+0xcc>
  d4:   00400593                addi    a1,zero,4
  d8:   00b00513                addi    a0,zero,11
  dc:   00000097                auipc   ra,0x0
  e0:   000080e7                jalr    ra,0(ra) # dc <main+0xdc>
  e4:   00050493                addi    s1,a0,0
  e8:   00050593                addi    a1,a0,0
  ec:   00a00513                addi    a0,zero,10
  f0:   00000097                auipc   ra,0x0
  f4:   000080e7                jalr    ra,0(ra) # f0 <main+0xf0>
  f8:   00048413                addi    s0,s1,0
  fc:   02c48493                addi    s1,s1,44
 100:   00000937                lui     s2,0x0

00000104 <.L2>:
 104:   00042583                lw      a1,0(s0)
 108:   00090513                addi    a0,s2,0 # 0 <main>
 10c:   00000097                auipc   ra,0x0
 110:   000080e7                jalr    ra,0(ra) # 10c <.L2+0x8>
 114:   00440413                addi    s0,s0,4
 118:   fe8496e3                bne     s1,s0,104 <.L2>
 11c:   00000513                addi    a0,zero,0
 120:   00c12083                lw      ra,12(sp)
 124:   00812403                lw      s0,8(sp)
 128:   00412483                lw      s1,4(sp)
 12c:   00012903                lw      s2,0(sp)
 130:   01010113                addi    sp,sp,16
 134:   00008067                jalr    zero,0(ra)
```

Рис 4.2 – содержимое секции .text файла main.o

```
RELOCATION RECORDS FOR [.text]:
OFFSET   TYPE              VALUE
0000001c R_RISCV_CALL      calloc
0000001c R_RISCV_RELAX     *ABS*
0000002c R_RISCV_CALL      pascal
0000002c R_RISCV_RELAX     *ABS*
00000038 R_RISCV_CALL      putchar
00000038 R_RISCV_RELAX     *ABS*
00000048 R_RISCV_CALL      calloc
00000048 R_RISCV_RELAX     *ABS*
0000005c R_RISCV_CALL      pascal
0000005c R_RISCV_RELAX     *ABS*
00000068 R_RISCV_HI20      .LC0
00000068 R_RISCV_RELAX     *ABS*
0000006c R_RISCV_LO12_I    .LC0
0000006c R_RISCV_RELAX     *ABS*
00000070 R_RISCV_CALL      printf
00000070 R_RISCV_RELAX     *ABS*
0000007c R_RISCV_CALL      putchar
0000007c R_RISCV_RELAX     *ABS*
0000008c R_RISCV_CALL      calloc
0000008c R_RISCV_RELAX     *ABS*
000000a0 R_RISCV_CALL      pascal
000000a0 R_RISCV_RELAX     *ABS*
000000ac R_RISCV_LO12_I    .LC0
000000ac R_RISCV_RELAX     *ABS*
000000b0 R_RISCV_CALL      printf
000000b0 R_RISCV_RELAX     *ABS*
000000bc R_RISCV_LO12_I    .LC0
000000bc R_RISCV_RELAX     *ABS*
000000c0 R_RISCV_CALL      printf
000000c0 R_RISCV_RELAX     *ABS*
000000cc R_RISCV_CALL      putchar
000000cc R_RISCV_RELAX     *ABS*
000000dc R_RISCV_CALL      calloc
000000dc R_RISCV_RELAX     *ABS*
000000f0 R_RISCV_CALL      pascal
000000f0 R_RISCV_RELAX     *ABS*
00000100 R_RISCV_HI20      .LC0
00000100 R_RISCV_RELAX     *ABS*
00000108 R_RISCV_LO12_I    .LC0
00000108 R_RISCV_RELAX     *ABS*
0000010c R_RISCV_CALL      printf
0000010c R_RISCV_RELAX     *ABS*
00000118 R_RISCV_BRANCH    .L2
```

Рис.5 – таблица перемещений для файла main.o

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

RELOCATION RECORDS FOR [.text]:
OFFSET   TYPE              VALUE
0000002c R_RISCV_HI20      .LC0
0000002c R_RISCV_RELAX     *ABS*
00000030 R_RISCV_LO12_I    .LC0
00000030 R_RISCV_RELAX     *ABS*
00000034 R_RISCV_CALL      printf
00000034 R_RISCV_RELAX     *ABS*
00000000 R_RISCV_BRANCH    .L14
00000010 R_RISCV_BRANCH    .L7
00000020 R_RISCV_JAL       .L6
00000050 R_RISCV_BRANCH    .L4
0000006c R_RISCV_BRANCH    .L5
00000078 R_RISCV_BRANCH    .L8
```

Рис. 6 – таблица перемещений для файла pascal.o

4. Компоновка.

Компоновка осуществляется следующей командой:

```
>riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 main.o pascal.o -o main.out
```

Выведем заголовки секций и таблицу переходов получившегося файла main.out.

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

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00015488  00010074  00010074  00000074  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .rodata       00000e4c  00025500  00025500  00015500  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  2 .eh_frame     000000b4  0002734c  0002734c  0001634c  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  3 .init_array   00000008  00027400  00027400  00016400  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  4 .fini_array   00000004  00027408  00027408  00016408  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  5 .data         0000099c  00027410  00027410  00016410  2**3
                  CONTENTS, ALLOC, LOAD, DATA
  6 .sdata        0000002c  00027db0  00027db0  00016db0  2**3
                  CONTENTS, ALLOC, LOAD, DATA
  7 .sbss         00000018  00027ddc  00027ddc  00016ddc  2**2
                  ALLOC
  8 .bss          00000044  00027df4  00027df4  00016ddc  2**2
                  ALLOC
  9 .comment      0000002a  00000000  00000000  00016ddc  2**0
                  CONTENTS, READONLY
 10 .riscv.attributes 0000001c  00000000  00000000  00016e06  2**0
                  CONTENTS, READONLY
 11 .debug_aranges 00000218  00000000  00000000  00016e28  2**3
                  CONTENTS, READONLY, DEBUGGING, OCTETS
 12 .debug_info   00006ab1  00000000  00000000  00017040  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
 13 .debug_abbrev 00001671  00000000  00000000  0001daf1  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
 14 .debug_line   0000a45e  00000000  00000000  0001f162  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
 15 .debug_frame  00000308  00000000  00000000  000295c0  2**2
                  CONTENTS, READONLY, DEBUGGING, OCTETS
 16 .debug_str    00000e78  00000000  00000000  000298c8  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
 17 .debug_loc    00008812  00000000  00000000  0002a740  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
 18 .debug_ranges 00001630  00000000  00000000  00032f52  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
```

Рис 7 – заголовки секций файла main.out

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

Рис. 8 – таблица перемещений файла main.out

По полученным результатам мы можем заметить, что компоновщик корректно обработал входные объектные файлы, он провел необходимые замены, то есть подключил неопределенные в объектных файлах функции, также он провел необходимые релокации, это мы видим из таблицы перемещений, и оптимизации.

Создание статической библиотеки.

Создадим статическую библиотеку с функцией pascal.

```
>riscv64-unknown-elf-ar -rsc lib.a pascal.o
```

Рассмотрим список символов полученной библиотеки

```
>riscv64-unknown-elf-nm lib.a
```

```
pascal.o:
00000024 t .L14
00000070 t .L4
00000058 t .L5
00000050 t .L6
00000084 t .L7
0000004c t .L8
00000000 r .LC0
00000000 T pascal
         U printf
```

Рис. 9 – таблица символов библиотеки

Как мы можем видеть по символу T функция pascal определена в этой библиотеке, в отличие от printf.

Создание make файла.

В процессе выполнения работы мы можем заметить как много команд нужно вводить, чтобы перекомпилировать проект при внесении в него изменений. Для этого были созданы make-файлы, чтобы компилировать проект одной командой.

```
# Переменные, чтобы не переписывать одно и то же много раз.
COMPILER=riscv64-unknown-elf-gcc
LIB=riscv64-unknown-elf-ar
ARCH=-march=rv32i -mabi=ilp32
#Команды записываются в формате [цель]: [зависимость]
#                                \t [команда]
# all – цель по умолчанию, перед исполнением команды цели, утилита
проходит по зависимостям и выполняет их команды.
all: pascal-risc

pascal-risc: main.o lib.a
    $(COMPILER) $(ARCH) main.o lib.a -o pascal-risc

main.o: main.c
    $(COMPILER) $(ARCH) -c main.c -o main.o

lib.a: pascal.o
```

```
    $(LIB) -rsc lib.a pascal.o

pascal.o: pascal.c
    $(COMPILER) $(ARCH) -c pascal.c -o pascal.o
```

При вызове команды «`make -f MakeFile`» строки из make-файла транслируются в командную строку, где исполняются. Это освобождает нас от необходимости каждый раз компилировать программу вручную.

Вывод.

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