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

Лабораторная работа №4

Дисциплина: «Низкоуровневое программирование»

Выполнил студент гр. 3530901/90002 _____ С.В. Пименов
(подпись)

Принял преподаватель _____ Д.С. Степанов
(подпись)

“____” _____ 2021 г.

Санкт-Петербург
2021

Задача:

1. Изучить методические материалы, опубликованные на сайте курса;

2. Установить пакет средств разработки “SiFive GNU Embedded Toolchain” для RISC-V;

3. На языке C разработать функцию, реализующую вывод n-й строки треугольника Паскаля. Поместить определение функции в отдельный исходный файл, оформить заголовочный файл. Разработать тестовую программу на языке C;

4. 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;

5. 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.

1. Программа, реализующая вывод n-й строки треугольника Паскаля.

```c
#ifndef LABA4_PASCAL_H
#define LABA4_PASCAL_H

extern void pascal(unsigned int pascalIndex);

#endif //LABA4_PASCAL_H
```

Листинг 1.1. – Заголовочный файл pascal.h.

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

void pascal(unsigned int pascalIndex) {
    unsigned int number = 1;
    for (int i = 1; i<=pascalIndex; i++) {
        printf("%u", number);
        number = number * (pascalIndex - i) / i;
        printf(" ");
    }
}
```

Листинг 1.2. – Основной файл pascal.c.

```c
#include "pascal.h"

int main() {
    unsigned int pascalIndex = 5;
    pascal(pascalIndex);
    return 0;
}
```

Листинг 1.3. – Тестовая программа main.c.

2. Сборка программы «по шагам».

2.1. Препроцессирование.

Препроцессирование выполнялось следующими командами:

```
riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 -E main.c -o main.i
riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 -E pascal.c -o pascal.i
```

Результат препроцессирования содержится в файле pascal.i и main.i. По причине того, что исходные файлы содержат заголовочные файлы нескольких стандартных библиотек C, результат препроцессирования отличается от исходных файлов и имеет достаточно много добавочных строк, среди которых и исходные программы. Также можно заметить, что препроцессор включил содержимое файла pascal.h.

```
# 1 "pascal.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "pascal.c"
# 1 "pascal.h" 1

extern void pascal(unsigned int pascalIndex);
# 2 "pascal.c" 2
# 1 "c:\\users\\DNS\\downloads\\riscv64-unknown-elf-toolchain-
10.2.0-2020.12.8-x86_64-w64-mingw32\\riscv64-unknown-
elf\\include\\stdio.h" 1 3
# 29 "c:\\users\\DNS\\downloads\\riscv64-unknown-elf-
toolchain-10.2.0-2020.12.8-x86_64-w64-mingw32\\riscv64-unknown-
elf\\include\\stdio.h" 3
…
# 3 "pascal.c" 2


# 4 "pascal.c"
void pascal(unsigned int pascalIndex) {
    unsigned int number = 1;
    for (int i = 1; i<=pascalIndex; i++) {
        printf("%u", number);
        number = number * (pascalIndex - i) / i;
        printf(" ");
    }
}
```

Листинг 2.1.1. Выход препроцессора (фрагменты), файл pascal.i.

```
# 1 "main.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "main.c"
# 1 "pascal.h" 1

extern void pascal(unsigned int pascalIndex);
# 2 "main.c" 2

int main() {
    unsigned int pascalIndex = 5;
```

```
    pascal(pascalIndex);
    return 0;
}
```

Листинг 2.1.2. Выход препроцессора (фрагменты), файл main.i.

2.2. Компиляция.

Компиляция выхода препроцессора осуществляется командой:

```
riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 –S
pascal.i -o pascal.s
riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 –S main.i -
o main.s
```

Наибольший интерес представляет файл main.s, так как в нём можно заметить обращение к подпрограмме pascal (значение регистра содержащее адрес возврата из main, сохраняется на время вызова в стеке).

Следует отметить, что символ pascal используется в файле, но никак не определяется.

```
     .file     "main.c"
     .option nopic
     .attribute arch, "rv32i2p0"
     .attribute unaligned_access, 0
     .attribute stack_align, 16
     .text
     .align    2
     .globl    main
     .type     main, @function
main:
     addi sp,sp,-16
     sw   ra,12(sp)
     li     a0,5 call
     pascalli
          a0,0

     lw   ra,12(sp)
     addi sp,sp,16
     jr   ra
     .size     main, .-main
     .ident    "GCC: (SiFive GCC-Metal 10.2.0-2020.12.8) 10.2.0"
```

Листинг 2.2.1 Выход компилятора main.s.

```
     .file     "pascal.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   "%u"
     .globl    __mulsi3
     .globl    __udivsi3
     .text
     .align    2
     .globl    pascal
     .type     pascal, @function
pascal:
     beq  a0,zero,.L6
     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
     li   s1,1
     li   s0,1
     lui  s3,%hi(.LC0)
.L3:
     mv   a1,s0
     addi a0,s3,%lo(.LC0)
     call printf
     mv   a1,s0
     sub  a0,s2,s1
     call__mulsi3
     mv   a1,s1
     call__udivsi3
     mv   s0,a0
     li   a0,32
     call putchar
     addi s1,s1,1
     bgeu s2,s1,.L3
     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
.L6:
     ret
     .size     pascal, .-pascal
     .ident    "GCC: (SiFive GCC-Metal 10.2.0-2020.12.8) 10.2.0"
```

Листинг 2.2.2 Выход компилятора pascal.s.

2.3. Выход ассемблера – объектный файл.

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

```
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000024  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000058  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000058  2**0
                  ALLOC
  3 .comment      00000031  00000000  00000000  00000058  2**0
                  CONTENTS, READONLY
  4 .riscv.attributes 0000001c 00000000 00000000 00000089
2**0
```

CONTENTS, READONLY

Листинг 2.3.1. Заголовки секций файла main.o.

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

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000090  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000c4  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000c4  2**0
                  ALLOC
  3 .rodata.str1.4 00000003  00000000  00000000  000000c4  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .comment      00000031  00000000  00000000  000000c7  2**0
                  CONTENTS, READONLY
  5 .riscv.attributes 0000001c 00000000 00000000 000000f8
2**0
                  CONTENTS, READONLY
```

Листинг 2.3.1. Заголовки секций файла pascal.o.

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

SYMBOL TABLE:
00000000 l    df *ABS*  00000000 main.c
00000000 l    d  .text  00000000 .text
00000000 l    d  .data  00000000 .data
00000000 l    d  .bss   00000000 .bss
00000000 l    d  .comment       00000000 .comment
00000000 l    d  .riscv.attributes      00000000
.riscv.attributes
00000000 g     F .text  00000024 main
00000000         *UND*  00000000 pascal
```

Листинг 2.3.3. Таблица символов файла 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
0000008c l       .text  00000000 .L6
0000002c l       .text  00000000 .L3
00000000 l    d  .comment       00000000 .comment
00000000 l    d  .riscv.attributes      00000000
.riscv.attributes
00000000         *UND*  00000000___mulsi3
00000000         *UND*  00000000___udivsi3
00000000 g     F .text  00000090 pascal
00000000         *UND*  00000000 printf
00000000         *UND*  00000000 putchar
```

Листинг 2.3.4. Таблица символов файла pascal.o.

В таблице символов main.o имеется запись: символ “pascal” типа *UND*. Эта запись означает, что символ “pascal” 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. То же самое относится к символам “printf” и “putchar”.

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


RELOCATION RECORDS FOR [.text]:
OFFSET   TYPE              VALUE
0000000c R_RISCV_CALL      pascal
0000000c R_RISCV_RELAX     *ABS*
```

Листинг 2.3.5. Таблица перемещений.

2.4. Результат компоновки.

Рассмотрим результат такой оптимизации в нашей программе.

```
0000000000010156 <main>:
   10156: 1141                    c.addi    sp,-16
   10158: e406                    c.sdsp    ra,8(sp)
   1015a: 4515                    c.li a0,5
   1015c: 00c000ef           jal  ra,10168 <pascal>
   10160: 4501                    c.li a0,0
   10162: 60a2                    c.ldsp    ra,8(sp)
   10164: 0141                    c.addi    sp,16
   10166: 8082                    c.jr ra

0000000000010168 <pascal>:
   10168: c939                    c.beqz    a0,101be
<pascal+0x56>
   1016a: 7179                    c.addi16sp     sp,-48
   1016c: f406                    c.sdsp    ra,40(sp)
   1016e: f022                    c.sdsp    s0,32(sp)
   10170: ec26                    c.sdsp    s1,24(sp)
   10172: e84a                    c.sdsp    s2,16(sp)
   10174: e44e                    c.sdsp    s3,8(sp)
   10176: e052                    c.sdsp    s4,0(sp)
   10178: 89aa                    c.mv s3,a0
   1017a: 4485                    c.li s1,1
   1017c: 4405                    c.li s0,1
   1017e: 6a75                    c.lui     s4,0x1d
   10180: 0004891b           addiw     s2,s1,0
   10184: 85a2                    c.mv a1,s0
   10186: 8f0a0513           addi a0,s4,-1808       #       1c8f0
<__clzdi2+0x36>
   1018a: 18a000ef           jal  ra,10314 <printf>
   1018e: 412987bb           subw a5,s3,s2
   10192: 0287843b           mulw s0,a5,s0
   10196: 0324543b           divuw     s0,s0,s2
   1019a: 02000513           addi a0,zero,32
```

```
   1019e: 1a6000ef           jal  ra,10344 <putchar>
   101a2: 0014879b           addiw     a5,s1,1
   101a6: 0007849b           addiw     s1,a5,0
   101aa: fc99fbe3           bgeu s3,s1,10180 <pascal+0x18>
   101ae: 70a2                    c.ldsp    ra,40(sp)
   101b0: 7402                    c.ldsp    s0,32(sp)
   101b2: 64e2                    c.ldsp    s1,24(sp)
   101b4: 6942                    c.ldsp    s2,16(sp)
   101b6: 69a2                    c.ldsp    s3,8(sp)
   101b8: 6a02                    c.ldsp    s4,0(sp)
   101ba: 6145                    c.addi16sp     sp,48
   101bc: 8082                    c.jr ra
   101be: 8082                    c.jr ra
```

Листинг 2.4. Результат компоновки a.out.

Можно видеть, что подпрограмма main не стала короче. Компоновщик посчитал, что изначальная программа уже была оптимальной.

3. Создание статической библиотеки.

Выделим из программы pascal.c функцию расчета коэффициента на текущем шаге в отдельную программу и объединим эти программы в статическую библиотеку pascal, тестовую программу main оставим без изменений.

```c
#ifndef LABA4_PASCAL_H
#define LABA4_PASCAL_H

extern void pascal(unsigned int pascalIndex);

#endif //LABA4_PASCAL_H
```

Листинг 3.1. – Заголовочный файл pascal.h.

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

void pascal(unsigned int pascalIndex) {
    unsigned int number = 1;
```

```
    for (int i = 1; i<=pascalIndex; i++) {
        printf("%u", number);
        number = number * computation(pascalIndex, i) / i;
        printf(" ");
    }
}
```

Листинг 3.2. – Заголовочный файл pascal.c.

```
#ifndef LABA4_COMPUTATION_H
#define LABA4_COMPUTATION_H

extern unsigned int comp (unsigned int pascalIndex, inti);

#endif //LABA4_COMPUTATION_H
```

Листинг 3.3. – Заголовочный файл comp.h.

```
unsigned int comp (unsigned int pascalIndex, int i) {return
    pascalIndex - i;
}
```

Листинг 3.4. – Заголовочный файл comp.c.

Для создания статической библиотеки получим объектные файлы всех используемых программ:

```
riscv64-unknown-elf-gcc -O1 -c pascal.c -o pascal.o
riscv64-unknown-elf-gcc -O1 -c computation.c -o computation.o
```

Объединим их в одну библиотеку pascal:

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

Используем получившуюся библиотеку для сборки программ:

```
main:     file format elf64-littleriscv
SYMBOL TABLE:
00000000000100b0 l    d  .text  0000000000000000 .text
…
0000000000010168 g           F .text   0000000000000054 pascal

000000000001c5cc g     F .text  00000000000000c2 .hidden
__extenddftf2
0000000000019424 g     F .text  0000000000000030 _close_r
000000000001224e g     F .text 00000000000000f8_swsetup_r
000000000001279e g     F .text  00000000000000da__sfp
0000000000016372 g     F .text  0000000000000048___copybits
000000000001f9d0 g       .bss   0000000000000000___BSS_END
000000000001f0d0 g     O .data  0000000000000810___malloc_av_
```

```
0000000000012894 g     F .text  0000000000000002
__sinit_lock_release
000000000001653e g     F .text  0000000000000038 __sread
000000000001573e g     F .text  0000000000000002 __malloc_lock
000000000001259a g     F .text  0000000000000036 _fflush_r
0000000000019378 g     F .text  00000000000000ac _calloc_r
000000000001f938 g       .sbss  0000000000000000 __bss_start
0000000000010248 g     F .text  00000000000000aa memset
0000000000010156 g     F .text  0000000000000012 main
000000000001f940 g     O .sbss  0000000000000008
__malloc_max_total_mem
0000000000012244 g     F .text  000000000000000a __swbuf
0000000000016610 g     F .text  0000000000000008 __sclose
0000000000019518 g     F .text  0000000000000042 fclose
000000000001503c g     F .text  0000000000000626 _malloc_r
00000000000192c6 g     F .text  0000000000000024 __ascii_wctomb
000000000001955a g     F .text  0000000000000020 _fiprintf_r
000000000001b1dc g     F .text  000000000000003e _init_signal
0000000000012c10 g     F .text  0000000000000082 _fwalk
0000000000015662 g     F .text  0000000000000006 _mbtowc_r
00000000000103ac g     F .text  0000000000000070 putc
00000000000128ea g     F .text  00000000000000d6 _malloc_trim_r
0000000000016618 g     F .text  00000000000000ea strcmp
000000000001921e g     F .text  000000000000000e vfiprintf
000000000001b6b2 g     F .text  0000000000000606 .hidden
__multf3
00000000000101bc g     F .text  0000000000000004 computation
```

Листинг 3.5. Таблица символов исполняемого файла.

Легко заметить, что в состав программы main вошло содержимое объектных файлов pascal.o и computation.o.

Процесс выполнения команд выше можно заменить make-файлами, которые произведут создание библиотеки и сборку программы.

```
CC=riscv64-unknown-elf-gcc
AR=riscv64-unknown-elf-ar
CFLAGS=-march=rv32i -mabi=ilp32 -O1
all: lib
lib: computation.o pascal.o
$(AR) -rsc lib.a computation.o pascal.o
computation.o: computation.c
$(CC) $(CFLAGS) -c computation.c -o computation.o
pascal.o: pascal.c
$(CC) $(CFLAGS) -c pascal.c -o pascal.o
clean:
rm -f *.o *.a
```

Листинг 3.6. make-файл для создания библиотеки.

```
TARGET=main
CC=riscv64-unknown-elf-gcc
CFLAGS=-march=rv32i -mabi=ilp32
-O1main:
$(CC) $(CFLAGS) main.c lib.a -o
$(TARGET)clean:
rm -f *.o *.a $(TARGET)
```

Листинг 3.7. make-файл для сборки.

Выводы

В ходе выполнения лабораторной работы я исследовал раздельную компиляцию (C -> Risc-V). Мной была разработана ф-ция на языке C, выводящая n-ую строку треугольника Паскаля, разработана тестовая программа и выделена подпрограмма, написан заголовочный файл.

Программа собрана по шагам, рассмотрен состав и содержимое секций, таблицы символов, таблицы перемещений и отладочная информация, содержащаяся в объектных файлах и исполняемом файле.

Выделена статическая библиотека, созданы make-файлы для создания статической библиотеки.