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

Отчёт по лабораторной работе № 4

Дисциплина: Низкоуровневое программирование

Тема: Раздельная компиляция

Вариант №12

Выполнил студент гр. 3530901/00002 _____ Ф.Д. Тарсуков
(подпись)

Преподаватель _____ Д.С. Степанов
(подпись)

“____” _____ 2021 г.

Санкт-Петербург

2021

1. Задание

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.

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.

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.

2. Описание работы

Вариант – 12: формирование последовательности чисел в коде Грея заданной разрядности

Алгоритм работы:

Первоначально задается некоторое число разрядов, по нему формируется счетчик выводимых чисел. Затем создается некоторая переменная, первоначально приравненная к нулю и с каждой итерацией увеличивается на единицу, пока не сравняется по значению с определенным ранее счетчиком. На каждой итерации из этой переменной формируется число в коде Грея методом сдвига на разряд вправо и логического сложения по модулю 2 с изначальным числом. Полученное число выводится в двоичном коде при помощи маски.

3. Текст программы

Основной файл grayCode.c:

```c
#include "grayCode.h"
#include "math.h"
#include "stdio.h"

int grayCode(int digits) {
    if (digits <= 0) return 0;
    int currentNumber = 0;
    int totalNumbers = (int)pow(2.0, (double)digits);
    for (int i = 0; totalNumbers > i; i++) {
        int auxiliaryNumber = currentNumber>>1;
        auxiliaryNumber = auxiliaryNumber ^ currentNumber;
        int correction = digits - 1;
        for (int j = totalNumbers>>1; j > 0; j = j>>1) {
            printf("%u", (auxiliaryNumber & j)>>correction);
            correction--;
        }
        printf("\n");
        currentNumber++;
    }
    return 0;
}
```

Заголовочный файл grayCode.h:

```c
#ifndef UNTITLED_GREYCODE_H
#define UNTITLED_GREYCODE_H

int grayCode(int digits);

#endif
```

Тестовая программа main.c:

```c
#include "grayCode.h"

int main() {
    int digits = 3;
    grayCode(digits);
    return 0;
}
```

4. Сборка программы “по шагам”

4.1. Препроцессирование

Первым шагом является препроцессирование файлов исходного текста (файлов “main.c” и “grayCode.c”), результаты записываются в файлы “main.i” и “grayCode.i”:

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -E C:\Users\elknu\CLionProjects\untitled\main.c -o C:\Users\elknu\CLionProjects\untitled\main.i -v -E >C:\Users\elknu\CLionProjects\untitled\log_main_pre.txt 2>&1

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -E C:\Users\elknu\CLionProjects\untitled\grayCode.c -o C:\Users\elknu\CLionProjects\untitled\grayCode.i -v -E >C:\Users\elknu\CLionProjects\untitled\log_statistics_pre.txt 2>&1

Параметры:

-march=rv64iac -mabi=lp64 – целевым является процессор с базовой архитектурой системы команд RV64I;

-O1 – выполнять простые оптимизации генерируемого кода (мы используем эту опцию в примерах, потому что обычно генерируемый код получается более простым);

-E – остановить процесс сборки после препроцессирования;

-o – путь к выходному файлу.

Выход препроцессора (фрагмент из файла main.i):

```
# 1 "C:\\Users\\elknu\\CLionProjects\\untitled\\main.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "C:\\Users\\elknu\\CLionProjects\\untitled\\main.c"
# 1 "C:\\Users\\elknu\\CLionProjects\\untitled\\grayCode.h" 1



int grayCode(int digits);
# 2 "C:\\Users\\elknu\\CLionProjects\\untitled\\main.c" 2

int main() {
    int digits = 3;
    grayCode(digits);
    return 0;
}
```

Выход препроцессора (фрагмент из файла grayCode.i):

```
# 1 "C:\\Users\\elknu\\CLionProjects\\untitled\\grayCode.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "C:\\Users\\elknu\\CLionProjects\\untitled\\grayCode.c"
# 1 "C:\\Users\\elknu\\CLionProjects\\untitled\\grayCode.h" 1



int grayCode(int digits);
# 2 "C:\\Users\\elknu\\CLionProjects\\untitled\\grayCode.c" 2

int grayCode(int digits) {
    if (digits <= 0) return 0;
    int currentNumber = 0;
    int totalNumbers = (int)pow(2.0, (double)digits);
    for (int i = 0; totalNumbers > i; i++) {
        int auxiliaryNumber = currentNumber>>1;
        auxiliaryNumber = auxiliaryNumber ^ currentNumber;
        int correction = digits - 1;
        for (int j = totalNumbers>>1; j > 0; j = j>>1) {
            printf("%u", (auxiliaryNumber & j)>>correction);
            correction--;
        }
        printf("\n");
        currentNumber++;
    }
    return 0;
}
```

Результат препроцессирования содержится в файле grayCode.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 также содержатся и исходные программы. Можно заметить, что препроцессор включил содержимое файла grayCode.h.

Появившиеся нестандартные директивы, начинающиеся с символа “#”, используются для передачи информации об исходном тексте из препроцессора в компилятор.

4.2. Компиляция

Далее необходимо выполнить компиляцию файлов “main.i” и “grayCode.i”, сохранив результат – сгенерированный код на языке ассемблера – в файлы “main.s” и “grayCode.s”.

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -S -fpreprocessed

C:\Users\elknu\CLionProjects\untitled\main.i -o
C:\Users\elknu\CLionProjects\untitled\main.s >C:\Users\elknu\CLionProjects\untitled\log_s_main.txt 2>&1

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -S -fpreprocessed
C:\Users\elknu\CLionProjects\untitled\grayCode.i -o
C:\Users\elknu\CLionProjects\untitled\grayCode.s >C:\Users\elknu\CLionProjects\untitled\log_s_statistics.txt 2>&1

Параметры:

-S – остановить процесс сборки после компиляции, не запуская ассемблер;

-fpreprocessed – выполняется компиляция файла, уже отработанного процесса;

Все остальные параметры из прошлого пункта означают то же самое.

Выход компилятора (файл main.s):

```
        .file   "main.c"
    .option nopic
    .attribute arch, "rv64i2p0_a2p0_c2p0"
    .attribute unaligned_access, 0
    .attribute stack_align, 16
    .text
    .align 1
    .globl main
    .type   main, @function
main:
    addi    sp,sp,-16
    sd ra,8(sp)
    li a0,3
    call    grayCode
    li a0,0
    ld ra,8(sp)
    addi    sp,sp,16
    jr ra
    .size   main, .-main
    .ident "GCC: (SiFive GCC 8.3.0-2020.04.1) 8.3.0"
```

Наибольший интерес представляет файл main.s, так как в нем можно заметить обращение к подпрограмме grayCode (значение регистра ra, содержащее адрес возврата из main, сохраняется на время вызова в стеке). Следует отметить, что символ grayCode используется в файле, но никак не определяется.

4.3. Ассемблирование

Как известно, содержательная часть объектного файла разбита на «разделы», называемые обычно секциями (section). Следующая команда обеспечивает отображение заголовков секций файлов “main.o” и “grayCode.o”:

riscv64-unknown-elf-objdump -h C:\Users\elknu\CLionProjects\untitled\main.o

riscv64-unknown-elf-objdump -h C:\Users\elknu\CLionProjects\untitled\grayCode.o

Заголовки секций файла main.o:

```
C:\Users\elknu\CLionProjects\untitled\main.o:     file format elf64-littleriscv

Sections:
Idx Name          Size      VMA               LMA               File off  Algn
  0 .text         00000016  0000000000000000  0000000000000000  00000040  2**1
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  0000000000000000  0000000000000000  00000056  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  0000000000000000  0000000000000000  00000056  2**0
                  ALLOC
  3 .comment      00000029  0000000000000000  0000000000000000  00000056  2**0
                  CONTENTS, READONLY
  4 .riscv.attributes 00000026  0000000000000000  0000000000000000  0000007f  2**0
                  CONTENTS, READONLY
```

Заголовки секций файла grayCode.o:

```
C:\Users\elknu\CLionProjects\untitled\grayCode.o:     file format elf64-littleriscv

Sections:
Idx Name          Size      VMA               LMA               File off  Algn
  0 .text         000000b4  0000000000000000  0000000000000000  00000040  2**1
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  0000000000000000  0000000000000000  000000f4  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  0000000000000000  0000000000000000  000000f4  2**0
                  ALLOC
  3 .rodata.str1.8 00000003  0000000000000000  0000000000000000  000000f8  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .srodata.cst8 00000008  0000000000000000  0000000000000000  00000100  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .comment      00000029  0000000000000000  0000000000000000  00000108  2**0
                  CONTENTS, READONLY
  6 .riscv.attributes 00000026  0000000000000000  0000000000000000  00000131  2**0
                  CONTENTS, READONLY
```

В файлах “main.o” и “grayCode.o” имеются следующие секции:

.text – секция кода, в которой содержатся коды инструкций (название секции обусловлено историческими причинами);

.data – секция инициализированных данных;

.bss – секция неинициализированных статических переменных (название секции также обусловлено историческими причинами);

.rodata – аналог .data для неизменяемых данных

.comment – секция данных о версиях размером 12 байт

.riscv.attributes – информация про RISC-V

Изучим содержимое таблиц символов объектных файлов “main.o” и “grayCode.o”:

riscv64-unknown-elf-objdump -t statistics.o C:\Users\elknu\CLionProjects\untitled\main.o

riscv64-unknown-elf-objdump -t statistics.o C:\Users\elknu\CLionProjects\untitled\grayCode.o

Таблица символов файла main.o:

```
C:\Users\elknu\CLionProjects\untitled\main.o:     file format elf64-littleriscv

SYMBOL TABLE:
0000000000000000 l    df *ABS*  0000000000000000 main.c
0000000000000000 l    d  .text  0000000000000000 .text
0000000000000000 l    d  .data  0000000000000000 .data
0000000000000000 l    d  .bss   0000000000000000 .bss
0000000000000000 l    d  .comment       0000000000000000 .comment
0000000000000000 l    d  .riscv.attributes      0000000000000000 .riscv.attributes
0000000000000000 g     F .text  0000000000000016 main
0000000000000000         *UND*  0000000000000000 grayCode



C:\Users\elknu>
```

Таблица символов файла grayCode.o:

```
C:\Users\elknu\CLionProjects\untitled\grayCode.o:     file format elf64-littleriscv

SYMBOL TABLE:
0000000000000000 l    df *ABS*  0000000000000000 grayCode.c
0000000000000000 l    d  .text  0000000000000000 .text
0000000000000000 l    d  .data  0000000000000000 .data
0000000000000000 l    d  .bss   0000000000000000 .bss
0000000000000000 l    d  .rodata.str1.8 0000000000000000 .rodata.str1.8
0000000000000000 l    d  .srodata.cst8  0000000000000000 .srodata.cst8
0000000000000000 l       .srodata.cst8  0000000000000000 .LC0
0000000000000000 l       .rodata.str1.8 0000000000000000 .LC1
00000000000000b0 l       .text  0000000000000000 .L9
0000000000000096 l       .text  0000000000000000 .L2
0000000000000066 l       .text  0000000000000000 .L5
0000000000000056 l       .text  0000000000000000 .L3
0000000000000076 l       .text  0000000000000000 .L4
0000000000000000 l    d  .comment       0000000000000000 .comment
0000000000000000 l    d  .riscv.attributes      0000000000000000 .riscv.attributes
0000000000000000         *UND*  0000000000000000 __floatsidf
0000000000000000         *UND*  0000000000000000 __fixdfsi
0000000000000000 g     F .text  00000000000000b4 grayCode
0000000000000000         *UND*  0000000000000000 pow
0000000000000000         *UND*  0000000000000000 putchar
0000000000000000         *UND*  0000000000000000 printf
```

В таблице символов main.o имеется запись: символ “grayCode” типа *UND*. Эта запись означает, что символ “grayCode” 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.

Изучим содержимое секции “.text” объектных файлов “main.o” и “grayCode.o”:

riscv64-unknown-elf-objdump –s –j .comment C:\Users\elknu\CLionProjects\untitled\main.o

```
C:\Users\elknu>riscv64-unknown-elf-objdump -s -j .comment C:\Users\elknu\CLionProjects\untitled\main.o

C:\Users\elknu\CLionProjects\untitled\main.o:     file format elf64-littleriscv

Contents of section .comment:
 0000 00474343 3a202853 69466976 65204743  .GCC: (SiFive GC
 0010 4320382e 332e302d 32303230 2e30342e  C 8.3.0-2020.04.
 0020 31292038 2e332e30 00                 1) 8.3.0.
```

Процедура декодирования кодов инструкций является «механической», следовательно, разумно поручить ее выполнение ЭВМ:

riscv64-unknown-elf-objdump -d -M no-aliases -j .text C:\Users\elknu\CLionProjects\untitled\main.o

riscv64-unknown-elf-objdump -d -M no-aliases -j .text statistics.o C:\Users\elknu\CLionProjects\untitled\grayCode.o

Опция “-d” инициирует процесс дизассемблирования (disassemble), опция “-M no-aliases” требует использовать в выводе только инструкции системы команд (но не псевдоинструкции ассемблера).

Вывод утилиты (фрагмент из grayCode.o и main.o):

```
C:\Users\elknu\CLionProjects\untitled\main.o:     file format elf64-littleriscv


Disassembly of section .text:

0000000000000000 <main>:
   0:   1141                    c.addi  sp,-16
   2:   e406                    c.sdsp  ra,8(sp)
   4:   450d                    c.li    a0,3
   6:   00000097                auipc   ra,0x0
   a:   000080e7                jalr    ra,0(ra) # 6 <main+0x6>
   e:   4501                    c.li    a0,0
  10:   60a2                    c.ldsp  ra,8(sp)
  12:   0141                    c.addi  sp,16
  14:   8082                    c.jr    ra
```

Результат дизассемблирования “grayCode.o” интереса не представляет, в отличие от результата дизассемблирования “main.o” в сравнении его с “main.s”.

5. Создание статической библиотеки

Статическая библиотека является, по сути, архивом объектных файлов.

Поместим grayCode.o в статическую библиотеку lib:

riscv64-unknown-elf-ar -rsc C:\Users\elknu\CLionProjects\untitled\libStatistics.a C:\Users\elknu\CLionProjects\untitled\grayCode.o

Параметры:

-r – заменить старые файлы с такими названиями (grayCode.o), если они уже есть в архиве;

-s – записать «index» в архив. Index – это список всех символов, объявленных во включенных в архив объектных файлах, и его присутствие ускоряет линковку;

-c – создать архив, если его еще не было.

Содержимое библиотеки:

```
C:\Users\elknu>riscv64-unknown-elf-nm C:\Users\elknu\CLionProjects\untitled\libStatistics.a

grayCode.o:
0000000000000096 t .L2
0000000000000056 t .L3
0000000000000076 t .L4
0000000000000066 t .L5
00000000000000b0 t .L9
0000000000000000 r .LC0
0000000000000000 r .LC1
                 U __fixdfsi
                 U __floatsidf
0000000000000000 T grayCode
                 U pow
                 U printf
                 U putchar
```

6. Вывод

Были разработаны функция и заголовочный файл на языке C, выполнена раздельная компиляция и изучены принципы создания статической библиотеки.