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

Отчет по лабораторной работе №4

Тема: Раздельная компиляция

Вариант 16

Выполнил студент гр. 3530901/90002 _____ П. В. Рубинова
(подпись)

Руководитель _____ Д. С. Степанов
(подпись)

“_____” _____ 2021 г.

Санкт-Петербург

2021

Постановка задачи:

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. 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.
3. 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.

Формулировка задачи:

Определение максимального простого числа, не превышающего заданное число N, методом решета Эратосфена (используется рабочий массив длины не более N).

Теория:

Решето Эратосфена — алгоритм нахождения всех простых чисел до некоторого целого числа N, который приписывают древнегреческому математику Эратосфену Киренскому.

В данном случае решето подразумевает фильтрацию всех чисел за исключением простых. По мере обработки массива чисел нужные числа (простые) остаются, а ненужные (составные) исключаются.

Программа на языке С

Листинг 1. Тестовая программа main.c.

```
#include <stdio.h>
#include "lab4C.h"

int main() {
    int array[] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1};
    int n = 10;
    lab4C(array, n);
    printf("%d", array[0]);
    return 0;
}
```

Листинг 2. Основной файл lab4C.c.

```
void lab4C(int array[], int n) {
    for(int i = 1; i < n; i++){
        if(array[i] != 0){
            array[0] = i + 1;
            for(int j = i; j < n; j += i + 1){
                array[j] = 0;
            }
        }
    }
}
```

Листинг 3. Заголовочный файл lab4C.h.

```
#ifndef lab4C_H
#define lab4C_H
void lab4C(int array[], int n);
```

```
#endif
```

Тестирование

```c
#include <stdio.h>

void lab4C(int array[], int n) {
    for(int i = 1; i < n; i++){
        if(array[i] != 0){
            array[0] = i + 1;
            for(int j = i; j < n; j += i + 1){
                array[j] = 0;
            }
        }
    }
}

int main() {
    int array[] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1};
    int n = 10;
    lab4C(array, n);
    printf("%d", array[0]);

    return 0;
}
```

```
7

...Program finished with exit code 0
Press ENTER to exit console.
```

В результате программы в нулевой элемент массива записывается, а впоследствии и выводится на экран, максимальное простое число, не превышающее заданное число N.

Сборка программы «по шагам»

Препроцессирование

Препроцессирование выполняется следующими командами:

```
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -E main.c -o main.i
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -E lab4C.c -o lab4C.i
```

Результат препроцессирования содержится в файлах main.i и lab4C.i. По причине того, что main.c содержит заголовочный файл стандартной библиотеки языка C stdio.h, результат препроцессирования этого файла имеет достаточно много добавочных строк.

Листинг 4. Файл main.i (фрагмент)

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

…

# 2 "main.c" 2
# 1 "lab4C.h" 1




# 3 "lab4C.h"
void lab4C(int array[], int n);
# 3 "main.c" 2
```

```
int main() {
    int array[] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1};
    int n = 10;
    lab4C(array, n);
    printf("%d", array[0]);
    return 0;
}
```

Листинг 5. Файл lab4C.i

```
# 1 "lab4C.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "lab4C.c"
# 1 "lab4C.h" 1



void lab4C(int array[], int n);
# 2 "lab4C.c" 2

void lab4C(int array[], int n) {
    for(int i = 1; i < n; i++){
        if(array[i] != 0){
            array[0] = i + 1;
            for(int j = i; j < n; j += i + 1){
                array[j] = 0;
            }
        }
    }
}
```

Компиляция

Компиляция препроцессированных файлов осуществляется следующими командами:

```
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -S -fpreprocessed main.i -o main.s
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -S -fpreprocessed lab4C.i -o lab4C.s
```

Листинг 6. Файл main.s

```
        .file   "main.c"
        .option nopic
        .attribute arch, "rv64i2p0_a2p0_c2p0"
        .attribute unaligned_access, 0
        .attribute stack_align, 16
        .text
        .section .rodata.str1.8,"aMS",@progbits,1
        .align  3
.LC1:
        .string "%d"
        .text
        .align  1
        .globl  main
        .type   main, @function
main:
        addi    sp,sp,-64
        sd      ra,56(sp)
        lui     a5,%hi(.LANCHOR0)
        addi    a5,a5,%lo(.LANCHOR0)
        ld      a1,0(a5)
```

```
        ld      a2,8(a5)
        ld      a3,16(a5)
        ld      a4,24(a5)
        ld      a5,32(a5)
        sd      a1,8(sp)
        sd      a2,16(sp)
        sd      a3,24(sp)
        sd      a4,32(sp)
        sd      a5,40(sp)
        li      a1,10
        addi    a0,sp,8
        call    lab4C
        lw      a1,8(sp)
        lui     a0,%hi(.LC1)
        addi    a0,a0,%lo(.LC1)
        call    printf
        li      a0,0
        ld      ra,56(sp)
        addi    sp,sp,64
        jr      ra
        .size   main, .-main
        .section .rodata
        .align  3
        .set    .LANCHOR0,. + 0
.LC0:
        .word   1
        .word   1
        .word   1
        .word   1
        .word   1
```

```
        .word   1
        .word   1
        .word   1
        .word   1
        .word   1
        .ident  "GCC: (SiFive GCC-Metal 10.2.0-2020.12.8) 10.2.0"
```

Листинг 7. Файл lab4C.s

```
        .file   "lab4C.c"
        .option nopic
        .attribute arch, "rv64i2p0_a2p0_c2p0"
        .attribute unaligned_access, 0
        .attribute stack_align, 16
        .text
        .align  1
        .globl  lab4C
        .type   lab4C, @function
lab4C:
        li      a5,1
        ble     a1,a5,.L1
        addi    a2,a0,4
        li      a3,1
        li      t1,4
        sub     t1,t1,a0
        j       .L5
.L3:
        addiw   a3,a3,1
        addi    a2,a2,4
        beq     a1,a3,.L1
.L5:
```

```
        lw      a5,0(a2)
        beq     a5,zero,.L3
        addiw   a5,a3,1
        sext.w  a6,a5
        sw      a5,0(a0)
        ble     a1,a3,.L3
        add     a7,t1,a2
        mv      a4,a2
        mv      a5,a3
.L4:
        sw      zero,0(a4)
        addw    a5,a5,a6
        add     a4,a4,a7
        bgt     a1,a5,.L4
        j       .L3
.L1:
        ret
        .size   lab4C, .-lab4C
        .ident  "GCC: (SiFive GCC-Metal 10.2.0-2020.12.8) 10.2.0"
```

Ассемблирование

Ассемблирование для получения объектных файлов программы осуществляется следующими командами:

```
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -v -c main.s -o main.o
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -v -c lab4C.s -o lab4C.o
```

Отображение заголовков секций файла “main.o” осуществляется командой:

```
riscv64-unknown-elf-objdump -h main.o
```

Листинг 8. Заголовки секций файла main.o

```
main.o:     file format elf64-littleriscv

Sections:
Idx Name          Size      VMA               LMA               File off  Algn
  0 .text         00000046  0000000000000000  0000000000000000  00000040  2**1
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  0000000000000000  0000000000000000  00000086  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  0000000000000000  0000000000000000  00000086  2**0
                  ALLOC
  3 .rodata.str1.8 00000003  0000000000000000  0000000000000000  00000088
2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .rodata       00000028  0000000000000000  0000000000000000  00000090
2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .comment      00000031  0000000000000000  0000000000000000  000000b8
2**0
                  CONTENTS, READONLY
```

```
 6 .riscv.attributes 00000035  0000000000000000  0000000000000000  000000e9
2**0
                  CONTENTS, READONLY
```

Отображение таблицы символов файла “main.o” осуществляется командой:

```
riscv64-unknown-elf-objdump –t main.o
```

Листинг 9. Таблица символов файла main.o

```
      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  .rodata.str1.8 0000000000000000 .rodata.str1.8
      0000000000000000 l    d  .rodata        0000000000000000 .rodata
      0000000000000000 l       .rodata        0000000000000000 .LANCHOR0
      0000000000000000 l       .rodata.str1.8 0000000000000000 .LC1
      0000000000000000 l    d  .comment       0000000000000000 .comment
      0000000000000000 l    d  .riscv.attributes      0000000000000000
.riscv.attributes
      0000000000000000 g     F .text  0000000000000046 main
      0000000000000000         *UND*  0000000000000000 lab4C
0000000000000000         *UND*  0000000000000000 printf
```

Отображение таблицы перемещений файла “main.o” осуществляется командой:

```
riscv64-unknown-elf-objdump -r main.o
```

Листинг 10. Таблица перемещений файла main.o

```
main.o:     file format elf64-littleriscv

RELOCATION RECORDS FOR [.text]:
OFFSET           TYPE              VALUE
0000000000000004 R_RISCV_HI20      .LANCHOR0
0000000000000004 R_RISCV_RELAX     *ABS*
0000000000000008 R_RISCV_LO12_I    .LANCHOR0
0000000000000008 R_RISCV_RELAX     *ABS*
0000000000000024 R_RISCV_CALL      lab4C
0000000000000024 R_RISCV_RELAX     *ABS*
000000000000002e R_RISCV_HI20      .LC1
000000000000002e R_RISCV_RELAX     *ABS*
0000000000000032 R_RISCV_LO12_I    .LC1
0000000000000032 R_RISCV_RELAX     *ABS*
0000000000000036 R_RISCV_CALL      printf
0000000000000036 R_RISCV_RELAX     *ABS*
```

Аналогичными командами отобразим заголовки секций, таблицу символов и таблицу перемещений файла lab4C.c.

Листинг 11. Заголовки секций файла lab4C.o

```
lab4C.o:     file format elf64-littleriscv

Sections:
Idx Name          Size      VMA               LMA               File off  Algn
  0 .text         00000048  0000000000000000  0000000000000000  00000040  2**1
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  0000000000000000  0000000000000000  00000088  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  0000000000000000  0000000000000000  00000088  2**0
```

```
                ALLOC
 3 .comment      00000031  0000000000000000  0000000000000000  00000088
2**0
                CONTENTS, READONLY
 4 .riscv.attributes 00000035  0000000000000000  0000000000000000  000000b9
2**0
                CONTENTS, READONLY
```

Листинг 12. Таблица символов файла lab4C.o

```
lab4C.o:     file format elf64-littleriscv

SYMBOL TABLE:
0000000000000000 l    df *ABS*  0000000000000000 lab4C.c
0000000000000000 l    d  .text  0000000000000000 .text
0000000000000000 l    d  .data  0000000000000000 .data
0000000000000000 l    d  .bss   0000000000000000 .bss
0000000000000046 l       .text  0000000000000000 .L1
000000000000001c l       .text  0000000000000000 .L5
0000000000000014 l       .text  0000000000000000 .L3
0000000000000036 l       .text  0000000000000000 .L4
0000000000000000 l    d  .comment       0000000000000000 .comment
0000000000000000 l    d  .riscv.attributes      0000000000000000 .riscv.attributes
0000000000000000 g     F .text  0000000000000048 lab4C
```

Листинг 13. Таблица перемещений файла lab4C.o

```
lab4C.o:     file format elf64-littleriscv

RELOCATION RECORDS FOR [.text]:
OFFSET           TYPE              VALUE
0000000000000002 R_RISCV_BRANCH    .L1
```

```
0000000000000012 R_RISCV_RVC_JUMP  .L5
0000000000000018 R_RISCV_BRANCH    .L1
000000000000001e R_RISCV_RVC_BRANCH  .L3
000000000000002a R_RISCV_BRANCH    .L3
0000000000000040 R_RISCV_BRANCH    .L4
0000000000000044 R_RISCV_RVC_JUMP  .L3
```

Компоновка

Компоновка осуществляется следующей командой:

```
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -v main.o lab4C.o
```

Для того чтобы рассмотреть секцию кода созданного исполняемого бинарного файла a.out воспользуемся следующей командой:

```
riscv64-unknown-elf-objdump –j .text –d –M no-aliases a.out >a.ds
```

Листинг 14. Исполняемый файл a.out (фрагмент)

```
a.out:     file format elf64-littleriscv


Disassembly of section .text:

...

0000000000010156 <main>:
   10156:       7139            c.addi16sp      sp,-64
   10158:       fc06            c.sdsp  ra,56(sp)
   1015a:       67f5            c.lui   a5,0x1d
   1015c:       83878793        addi    a5,a5,-1992 # 1c838 <__clzdi2+0x42>
   10160:       638c            c.ld    a1,0(a5)
   10162:       6790            c.ld    a2,8(a5)
   10164:       6b94            c.ld    a3,16(a5)
   10166:       6f98            c.ld    a4,24(a5)
   10168:       739c            c.ld    a5,32(a5)
   1016a:       e42e            c.sdsp  a1,8(sp)
   1016c:       e832            c.sdsp  a2,16(sp)
```

```
  1016e:        ec36            c.sdsp  a3,24(sp)
  10170:        f03a            c.sdsp  a4,32(sp)
  10172:        f43e            c.sdsp  a5,40(sp)
  10174:        45a9            c.li    a1,10
  10176:        0028            c.addi4spn      a0,sp,8
  10178:        018000ef        jal     ra,10190 <lab4C>
  1017c:        45a2            c.lwsp  a1,8(sp)
  1017e:        6575            c.lui   a0,0x1d
  10180:        83050513        addi    a0,a0,-2000 # 1c830 <__clzdi2+0x3a>
  10184:        1a8000ef        jal     ra,1032c <printf>
  10188:        4501            c.li    a0,0
  1018a:        70e2            c.ldsp  ra,56(sp)
  1018c:        6121            c.addi16sp      sp,64
  1018e:        8082            c.jr    ra

0000000000010190 <lab4C>:
  10190:        4785            c.li    a5,1
  10192:        04b7d263        bge     a5,a1,101d6 <lab4C+0x46>
  10196:        00450613        addi    a2,a0,4
  1019a:        4685            c.li    a3,1
  1019c:        4311            c.li    t1,4
  1019e:        40a30333        sub     t1,t1,a0
  101a2:        a029            c.j     101ac <lab4C+0x1c>
  101a4:        2685            c.addiw a3,1
  101a6:        0611            c.addi  a2,4
  101a8:        02d58763        beq     a1,a3,101d6 <lab4C+0x46>
  101ac:        421c            c.lw    a5,0(a2)
  101ae:        dbfd            c.beqz  a5,101a4 <lab4C+0x14>
  101b0:        0016879b        addiw   a5,a3,1
  101b4:        0007881b        addiw   a6,a5,0
```

```
  101b8:        c11c            c.sw    a5,0(a0)
  101ba:        feb6d5e3        bge     a3,a1,101a4 <lab4C+0x14>
  101be:        00c308b3        add     a7,t1,a2
  101c2:        8732            c.mv    a4,a2
  101c4:        87b6            c.mv    a5,a3
  101c6:        00072023        sw      zero,0(a4)
  101ca:        010787bb        addw    a5,a5,a6
  101ce:        9746            c.add   a4,a7
  101d0:        feb7cbe3        blt     a5,a1,101c6 <lab4C+0x36>
  101d4:        bfc1            c.j     101a4 <lab4C+0x14>
  101d6:        8082            c.jr    ra

...
```

Создание статической библиотеки и make-файлов

Выделим функцию lab4C.c в статическую библиотеку lab4Clib. Тестовую программу main.c оставим без изменений.

Для создания статической библиотеки получим объектный файл используемой программы lab4C.o, воспользовавшись следующей программой:

```
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -c lab4C.c -o lab4C.o
```

Соберем библиотеку следующей командой

```
riscv64-unknown-elf-ar -rsc lab4Clib.a lab4C.o
```

Рассмотрим список символов библиотеки, воспользовавшись следующей командой:

```
riscv64-unknown-elf-nm lab4Clib.a
```

Листинг 15. Таблица символов lab4Clib.a

```
        lab4C.o:
        0000000000000046 t .L1
        0000000000000014 t .L3
        0000000000000036 t .L4
        000000000000001c t .L5
0000000000000000 T lab4C
```

Используя получившуюся библиотеку, соберем исполняемый файл программы следующей командой:

```
riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 main.c lab4Clib.a
```

Листинг 16. Таблица символов исполняемого файла (фрагмент)

```
a.out:     file format elf64-littleriscv

SYMBOL TABLE:
00000000000100b0 l    d  .text  0000000000000000 .text

…

0000000000000000 l    df *ABS*          0000000000000000 main.c
0000000000000000 l    df *ABS*          0000000000000000 lab4C.c
0000000000000000 l    df *ABS*          0000000000000000 exit.c
0000000000000000 l    df *ABS*          0000000000000000 impure.c
000000000001eab0 l     O .data  0000000000000748 impure_data

…
```

Можно заметить, что в состав программы вошло содержимое объектного файла lab4C.o.

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

Листинг 16. MakeLib для создания статической библиотеки

```
.PHONY: all clean

# Исходные файлы, необходимые для сборки библиотеки

#Вызываемые приложения
AR = riscv64-unknown-elf-ar
CC = riscv64-unknown-elf-gcc
```

```
# Файл библиотеки
MYLIBNAME = lab4Clib.a

# Параметры компиляции
CFLAGS= -O1

# Включаемые файлы следует искать в текущем каталоге
INCLUDES+= -I .

# Make должна искать файлы *.h и *.c в текущей директории
vpath %.h .
vpath %.c .

# Построение объектного файла из исходного текста
# $< = %.c
# $@ = %.o
%.o: %.c
	$(CC) -MD $(CFLAGS) $(INCLUDES) -c $< -o $@

# Чтобы достичь цели "all", требуется построить библиотеку
all: $(MYLIBNAME)

# $^ = (lab4C.o)
$(MYLIBNAME): lab4C.o
	$(AR) -rsc $@ $^
```

Листинг 17. MakeApp для сборки исполняемого файла

```
# "Фиктивные" цели
.PHONY: all clean
```

```
# Файлы для сборки исполнимого файла
OBJS= main.c \
	lab4Clib.a

#Вызываемые приложения
CC = riscv64-unknown-elf-gcc.exe

# Параметры компиляции
CFLAGS= --save-temps

# Включаемые файлы следует искать в текущем каталоге
INCLUDES+= -I .

# Make должна искать файлы *.c и *.a в текущей директории
vpath %.c .
vpath %.a .

# Чтобы достичь цели "all", требуется собрать исполнимый файл
all: a.out

# Сборка исполнимого файла и удаление мусора
a.out: $(OBJS)
	$(CC) $(CFLAGS) $(INCLUDES) $^
	del *.o *.i *.s *.d
```

Сначала мы запускаем MakeLib со сборкой библиотеки, а затем MakeLib со сборкой исполняемого файла.

Вывод

В ходе данной лабораторной работы я изучила раздельную компиляцию на примере определения максимального простого числа, не превышающего заданное число N, методом решета Эратосфена.