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 7

Выполнил студент гр. 3530901/90002 _____Д.С. Ковалевский
(подпись)

Принял старший преподаватель _____Д.С. Степанов
(подпись)

“___” _____ 2021 г.

Санкт-Петербург
2021

Цель работы:

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 определенную вариантом задания функциональность. 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 исполняеммом 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.

Вариант 7: Определение k-й порядковой статистики in-place.

1.Функция на С

Сначала разработаем функцию на С, которая будет реализовывать поиск k-ой порядковой статистики. Напишем функцию в отдельном файле.

```c
#include "main.h"

unsigned statistics(unsigned array[], unsigned size, int k){
    unsigned temp;
    for(int i = 0; i < size - 1; i++){
        for(int j = 0; j < size - 1; j++){
            if (array[j] > array[i+1]){
                temp = array[j];
                array[j] = array[j + 1];
                array[j+1] = temp;
            }
        }
    }
    return array[k];
}
```

Рис.1. Функция файла statistics.c.

Так же напишем тестирующую функцию в main.c:

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

int main() {
    int k = 4;
    unsigned array[] = {0, 6, 2, 8, 6};
    size_t size = sizeof(array) / sizeof(array[0]);
    for (int i = 0; i < size; i++) {
        printf( _Format: "%u", array[i]);
    }
    printf( _Format: "\n");
    statistics(array, size, k);
    for (int i = 0; i < size; i++) {
        printf( _Format: "%u", array[i]);
    }
    printf( _Format: "\n%u", statistics(array, size, k));

    return 0;
}
```

Рис.2. Функция файла main.c.

Так же не забудем связать это все в main.h:

```
#ifndef FORRISC_MAIN_H
#define FORRISC_MAIN_H

unsigned statistics(unsigned array[], unsigned size, int k);

#endif //FORRISC_MAIN_H
```

Рис.3. Файл main.h.

Алгоритм:

1. Если i-ый элемент строго больше i+1, то меняем их местами
2. Идем дальше по массиву
3. Повторяем шаг 1 и 2 до конца массива N раз
4. Возвращаем k-ый элемент

Результат работы функции:

```
06286
02668
8
```

Рис.4. Результат работы функций.

На Рис.4 видим в первой строчки изначальный массив, дальше отсортированный массив и в третьей строчке искомая k-ая статистика (k = 4).

2.Сборка простейшей программы «по шагам»

Препроцессирование

Первым шагом является препроцессирование файлов с исходными текстами. Для этого используется пакет разработки «SiFive GNU Embedded Toolchain». Чтобы это выполнить, необходимо использовать команды:

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -E main.c -o main.i -v -E >log_main_pre.txt 2>&1

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -E statistics.c -o statistics.i -v -E >log_statistics_pre.txt 2>&1

```
# 9 "main.h"
unsigned statistics(unsigned array[], unsigned size, int k);
# 3 "main.c" 2

unsigned statistics(unsigned array[], unsigned size, int k);

int main() {
    int k = 4;
    unsigned array[] = {0, 6, 2, 8, 6};
    size_t size = sizeof(array) / sizeof(array[0]);
    for (int i = 0; i < size; i++) {
        printf("%u", array[i]);
    }
    printf("\n");
    statistics(array, size, k);
    for (int i = 0; i < size; i++) {
        printf("%u", array[i]);
    }
    printf("\n%u", statistics(array, size, k));

    return 0;
}
```

Рис.5. Фрагмент изначального кода в main.i.

```
# 1 "statistics.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "statistics.c"
# 1 "main.h" 1
# 9 "main.h"
unsigned statistics(unsigned array[], unsigned size, int k);
# 2 "statistics.c" 2

unsigned statistics(unsigned array[], unsigned size, int k){
    unsigned temp;
    for(int i = 0; i < size - 1; i++){
        for(int j = 0; j < size - 1; j++){
            if (array[j] > array[i+1]){
                temp = array[j];
                array[j] = array[j + 1];
                array[j+1] = temp;
            }
        }
    }
    return array[k];
}
```

Рис.6. Фрагмент изначального кода в statistics.i.

Как видно на рис.5 и 6, в созданных файлах после препроцессирования содержится наш изначальный код.

Компиляция

Для выполнения компиляции используем следующие команды:

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -S -fpreprocessed main.i -o main.s >log_s_main.txt 2>&1

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 -v -S -fpreprocessed statistics.i -o statistics.s >log_s_statistics.txt 2>&1

Получаем следующие файлы:

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    "%u"
   .align 3
.LC2:
   .string    "\n%u"
   .text
   .align 1
   .globl main
   .type  main, @function
main:
   addi   sp,sp,-80
   sd ra,72(sp)
   sd s0,64(sp)
   sd s1,56(sp)
   sd s2,48(sp)
   sd s3,40(sp)
   lui    a5,%hi(.LANCHOR0)
   addi   a5,a5,%lo(.LANCHOR0)
   ld a4,0(a5)
   sd a4,8(sp)
   ld a4,8(a5)
   sd a4,16(sp)
   lw a5,16(a5)
   sw a5,24(sp)
   addi   s0,sp,8
   addi   s2,sp,28
   mv s1,s0
   lui    s3,%hi(.LC1)
.L2:
   lw a1,0(s1)
   addi   a0,s3,%lo(.LC1)
   call   printf
   addi   s1,s1,4
   bne    s1,s2,.L2
   li a0,10
   call   putchar
   li a2,4
   li a1,5
   addi   a0,sp,8
   call   statistics
   lui    s1,%hi(.LC1)
.L3:
   lw a1,0(s0)
   addi   a0,s1,%lo(.LC1)
   call   printf
   addi   s0,s0,4
   bne    s0,s2,.L3
   li a2,4
   li a1,5
   addi   a0,sp,8
   call   statistics
   sext.w a1,a0
   lui    a0,%hi(.LC2)
   addi   a0,a0,%lo(.LC2)
   call   printf
   li a0,0
   ld ra,72(sp)
```

```
   ld s0,64(sp)
   ld s1,56(sp)
   ld s2,48(sp)
   ld s3,40(sp)
   addi   sp,sp,80
   jr ra
   .size  main, .-main
   .section   .rodata
   .align 3
   .set   .LANCHOR0,. + 0
.LC0:
   .word  0
   .word  6
   .word  2
   .word  8
   .word  6
   .ident "GCC: (SiFive GCC-Metal 10.2.0-2020.12.8) 10.2.0"
```

И statistics.s:

```
    .file  "statistics.c"
   .option nopic
   .attribute arch, "rv64i2p0_a2p0_c2p0"
   .attribute unaligned_access, 0
   .attribute stack_align, 16
   .text
   .align 1
   .globl statistics
   .type  statistics, @function
statistics:
   li a5,1
   beq    a1,a5,.L2
   addi   a6,a0,4
   addiw  a1,a1,-2
   slli   a5,a1,32
   srli   a1,a5,30
   addi   a7,a0,8
   add    a7,a7,a1
   add    a1,a6,a1
   j  .L3
.L4:
   addi   a5,a5,4
   beq    a5,a1,.L7
.L5:
   lw a4,0(a5)
   lw a3,0(a6)
   bgeu   a3,a4,.L4
   lw a3,4(a5)
   sw a3,0(a5)
   sw a4,4(a5)
   j  .L4
.L7:
   addi   a6,a6,4
   beq    a6,a7,.L2
.L3:
   mv a5,a0
   j  .L5
.L2:
   slli   a5,a2,2
   add    a0,a0,a5
   lw a0,0(a0)
```

```
    ret
    .size  statistics, .-statistics
    .ident "GCC: (SiFive GCC-Metal 10.2.0-2020.12.8) 10.2.0"
```

Получаем инструкции на RISC-V.

Ассемблирование

Используем следующие команды:

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -v -c main.s -o main.o >log_o.txt 2>&1

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -v -c statistics.s -o statistics.o >log_o.txt 2>&1

После их выполнения получаем объектные файлы main.o и statistics.o. Для их прочтения будем использовать следующие команды:

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

riscv64-unknown-elf-objdump -h statistics.o

```
C:\Users\User\CLionProjects\forRISC>riscv64-unknown-elf-objdump -h main.o

main.o:     file format elf64-littleriscv

Sections:
Idx Name          Size      VMA               LMA               File off  Algn
  0 .text         000000a2  0000000000000000  0000000000000000  00000040  2**1
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  0000000000000000  0000000000000000  000000e2  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  0000000000000000  0000000000000000  000000e2  2**0
                  ALLOC
  3 .rodata.str1.8 0000000c  0000000000000000  0000000000000000  000000e8  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .rodata       00000014  0000000000000000  0000000000000000  000000f8  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .comment      00000031  0000000000000000  0000000000000000  0000010c  2**0
                  CONTENTS, READONLY
  6 .riscv.attributes 00000026  0000000000000000  0000000000000000  0000013d  2**0
                  CONTENTS, READONLY
```

Рис.7. Хедер файла main.o.

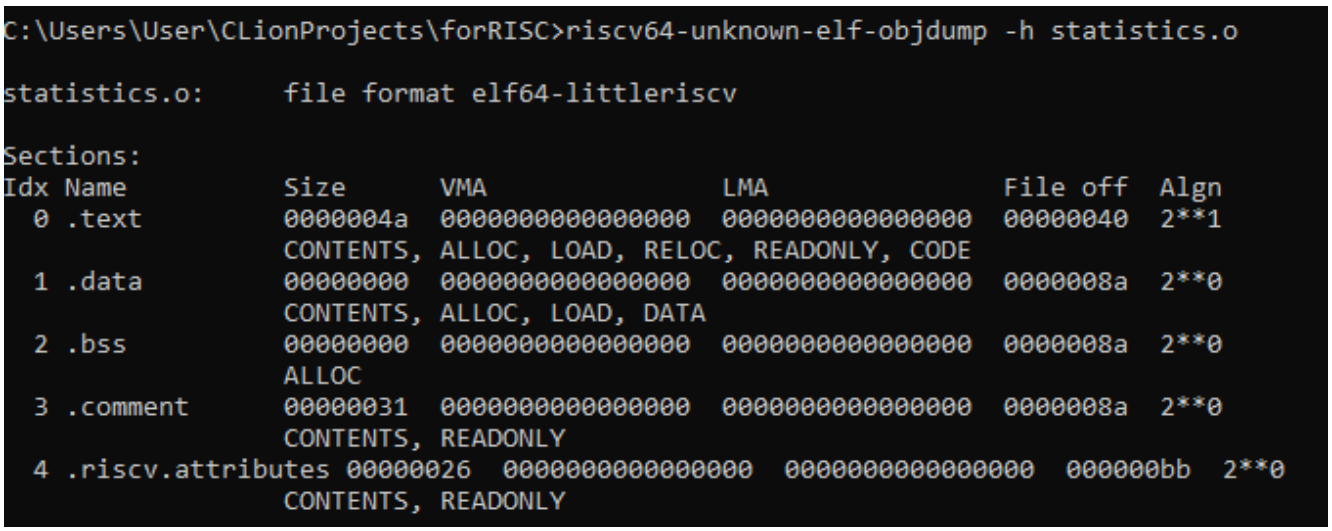

Рис.8. Хедер файла statistics.o.

Вся информация размещается в секциях:

Секция	Назначение
.text	секция кода, в которой содержатся коды инструкций
.data	секция инициализированных данных
.bss	секция данных, инициализированных нулями
.comment	секция данных о версиях размером 12 байт
.rodata	секция данных в формате read-only

Далее вводим команду:

riscv64-unknown-elf-objdump -d -M no-aliases -j .text main.o

И можем более подробно рассмотреть секцию .text.

```
Disassembly of section .text:

0000000000000000 <main>:
   0:   715d                    c.addi16sp      sp,-80
   2:   e486                    c.sdsp  ra,72(sp)
   4:   e0a2                    c.sdsp  s0,64(sp)
   6:   fc26                    c.sdsp  s1,56(sp)
   8:   f84a                    c.sdsp  s2,48(sp)
   a:   f44e                    c.sdsp  s3,40(sp)
   c:   000007b7                lui     a5,0x0
  10:   00078793                addi    a5,a5,0 # 0 <main>
  14:   6398                    c.ld    a4,0(a5)
  16:   e43a                    c.sdsp  a4,8(sp)
  18:   6798                    c.ld    a4,8(a5)
  1a:   e83a                    c.sdsp  a4,16(sp)
  1c:   4b9c                    c.lw    a5,16(a5)
  1e:   cc3e                    c.swsp  a5,24(sp)
  20:   0020                    c.addi4spn      s0,sp,8
  22:   01c10913                addi    s2,sp,28
  26:   84a2                    c.mv    s1,s0
  28:   000009b7                lui     s3,0x0

000000000000002c <.L2>:
  2c:   408c                    c.lw    a1,0(s1)
  2e:   00098513                addi    a0,s3,0 # 0 <main>
  32:   00000097                auipc   ra,0x0
  36:   000080e7                jalr    ra,0(ra) # 32 <.L2+0x6>
  3a:   0491                    c.addi  s1,4
  3c:   ff2498e3                bne     s1,s2,2c <.L2>
  40:   4529                    c.li    a0,10
  42:   00000097                auipc   ra,0x0
  46:   000080e7                jalr    ra,0(ra) # 42 <.L2+0x16>
  4a:   4611                    c.li    a2,4
  4c:   4595                    c.li    a1,5
  4e:   0028                    c.addi4spn      a0,sp,8
  50:   00000097                auipc   ra,0x0
  54:   000080e7                jalr    ra,0(ra) # 50 <.L2+0x24>
  58:   000004b7                lui     s1,0x0

000000000000005c <.L3>:
  5c:   400c                    c.lw    a1,0(s0)
  5e:   00048513                addi    a0,s1,0 # 0 <main>
  62:   00000097                auipc   ra,0x0
  66:   000080e7                jalr    ra,0(ra) # 62 <.L3+0x6>
  6a:   0411                    c.addi  s0,4
  6c:   ff2418e3                bne     s0,s2,5c <.L3>
  70:   4611                    c.li    a2,4
  72:   4595                    c.li    a1,5
  74:   0028                    c.addi4spn      a0,sp,8
  76:   00000097                auipc   ra,0x0
  7a:   000080e7                jalr    ra,0(ra) # 76 <.L3+0x1a>
  7e:   0005059b                addiw   a1,a0,0
  82:   00000537                lui     a0,0x0
  86:   00050513                addi    a0,a0,0 # 0 <main>
  8a:   00000097                auipc   ra,0x0
  8e:   000080e7                jalr    ra,0(ra) # 8a <.L3+0x2e>
  92:   4501                    c.li    a0,0
  94:   60a6                    c.ldsp  ra,72(sp)
  96:   6406                    c.ldsp  s0,64(sp)
  98:   74e2                    c.ldsp  s1,56(sp)
  9a:   7942                    c.ldsp  s2,48(sp)
  9c:   79a2                    c.ldsp  s3,40(sp)
  9e:   6161                    c.addi16sp      sp,80
  a0:   8082                    c.jr    ra
```

Рис.9. Секция .text файла main.o.

Командой:

riscv64-unknown-elf-objdump -t statistics.o main.o

Получаем таблицу символов.

```
C:\Users\User\CLionProjects\forRISC>riscv64-unknown-elf-objdump -t statistics.o main.o

statistics.o:     file format elf64-littleriscv

SYMBOL TABLE:
0000000000000000 l    df *ABS*  0000000000000000 statistics.c
0000000000000000 l    d  .text  0000000000000000 .text
0000000000000000 l    d  .data  0000000000000000 .data
0000000000000000 l    d  .bss   0000000000000000 .bss
0000000000000040 l       .text  0000000000000000 .L2
000000000000003c l       .text  0000000000000000 .L3
0000000000000036 l       .text  0000000000000000 .L7
000000000000001e l       .text  0000000000000000 .L4
0000000000000024 l       .text  0000000000000000 .L5
0000000000000000 l    d  .comment       0000000000000000 .comment
0000000000000000 l    d  .riscv.attributes      0000000000000000 .riscv.attributes
0000000000000000 g     F .text  000000000000004a statistics



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
0000000000000008 l       .rodata.str1.8 0000000000000000 .LC2
000000000000002c l       .text  0000000000000000 .L2
000000000000005c l       .text  0000000000000000 .L3
0000000000000000 l    d  .comment       0000000000000000 .comment
0000000000000000 l    d  .riscv.attributes      0000000000000000 .riscv.attributes
0000000000000000 g     F .text  00000000000000a2 main
0000000000000000         *UND*  0000000000000000 printf
0000000000000000         *UND*  0000000000000000 putchar
0000000000000000         *UND*  0000000000000000 statistics
```

Рис.10. Таблица символов

А так же таблица перемещений:

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

```
C:\Users\User\CLionProjects\forRISC>riscv64-unknown-elf-objdump -r statistics.o main.o

statistics.o:     file format elf64-littleriscv

RELOCATION RECORDS FOR [.text]:
OFFSET           TYPE              VALUE
0000000000000002 R_RISCV_BRANCH    .L2
000000000000001c R_RISCV_RVC_JUMP  .L3
0000000000000020 R_RISCV_BRANCH    .L7
000000000000002a R_RISCV_BRANCH    .L4
0000000000000034 R_RISCV_RVC_JUMP  .L4
0000000000000038 R_RISCV_BRANCH    .L2
000000000000003e R_RISCV_RVC_JUMP  .L5



main.o:     file format elf64-littleriscv

RELOCATION RECORDS FOR [.text]:
OFFSET           TYPE              VALUE
000000000000000c R_RISCV_HI20      .LANCHOR0
000000000000000c R_RISCV_RELAX     *ABS*
0000000000000010 R_RISCV_LO12_I    .LANCHOR0
0000000000000010 R_RISCV_RELAX     *ABS*
0000000000000028 R_RISCV_HI20      .LC1
0000000000000028 R_RISCV_RELAX     *ABS*
000000000000002e R_RISCV_LO12_I    .LC1
000000000000002e R_RISCV_RELAX     *ABS*
0000000000000032 R_RISCV_CALL      printf
0000000000000032 R_RISCV_RELAX     *ABS*
0000000000000042 R_RISCV_CALL      putchar
0000000000000042 R_RISCV_RELAX     *ABS*
0000000000000050 R_RISCV_CALL      statistics
0000000000000050 R_RISCV_RELAX     *ABS*
0000000000000058 R_RISCV_HI20      .LC1
0000000000000058 R_RISCV_RELAX     *ABS*
000000000000005e R_RISCV_LO12_I    .LC1
000000000000005e R_RISCV_RELAX     *ABS*
0000000000000062 R_RISCV_CALL      printf
0000000000000062 R_RISCV_RELAX     *ABS*
0000000000000076 R_RISCV_CALL      statistics
0000000000000076 R_RISCV_RELAX     *ABS*
0000000000000082 R_RISCV_HI20      .LC2
0000000000000082 R_RISCV_RELAX     *ABS*
0000000000000086 R_RISCV_LO12_I    .LC2
0000000000000086 R_RISCV_RELAX     *ABS*
000000000000008a R_RISCV_CALL      printf
000000000000008a R_RISCV_RELAX     *ABS*
000000000000003c R_RISCV_BRANCH    .L2
000000000000006c R_RISCV_BRANCH    .L3
```

Рис.11. Таблица перемещений.

Здесь можно найти записи типа R_RISCV_CALL – сообщения для компоновщика о возможных оптимизациях.

Компоновка

Выполняем компоновку командой:

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -v main.o statistics.o -o main.out >log_out.txt 2>&1

Мы получили файл main.out, для рассмотрения его секции, используем:

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

```
0000000000010156 <main>:
   10156:   715d                    c.addi16sp  sp,-80
   10158:   e486                    c.sdsp  ra,72(sp)
   1015a:   e0a2                    c.sdsp  s0,64(sp)
   1015c:   fc26                    c.sdsp  s1,56(sp)
   1015e:   f84a                    c.sdsp  s2,48(sp)
   10160:   f44e                    c.sdsp  s3,40(sp)
   10162:   67f5                    c.lui   a5,0x1d
   10164:   c5078793            addi    a5,a5,-944 # 1cc50 <__clzdi2+0x42>
   10168:   6398                    c.ld    a4,0(a5)
   1016a:   e43a                    c.sdsp  a4,8(sp)
   1016c:   6798                    c.ld    a4,8(a5)
   1016e:   e83a                    c.sdsp  a4,16(sp)
   10170:   4b9c                    c.lw    a5,16(a5)
   10172:   cc3e                    c.swsp  a5,24(sp)
   10174:   0020                    c.addi4spn  s0,sp,8
   10176:   01c10913            addi    s2,sp,28
   1017a:   84a2                    c.mv    s1,s0
   1017c:   69f5                    c.lui   s3,0x1d
   1017e:   408c                    c.lw    a1,0(s1)
   10180:   c4098513            addi    a0,s3,-960 # 1cc40 <__clzdi2+0x32>
   10184:   1f2000ef            jal ra,10376 <printf>
   10188:   0491                    c.addi  s1,4
   1018a:   ff249ae3            bne s1,s2,1017e <main+0x28>
   1018e:   4529                    c.li    a0,10
   10190:   216000ef            jal ra,103a6 <putchar>
   10194:   4611                    c.li    a2,4
   10196:   4595                    c.li    a1,5
   10198:   0028                    c.addi4spn  a0,sp,8
   1019a:   03e000ef            jal ra,101d8 <statistics>
   1019e:   64f5                    c.lui   s1,0x1d
   101a0:   400c                    c.lw    a1,0(s0)
   101a2:   c4048513            addi    a0,s1,-960 # 1cc40 <__clzdi2+0x32>
   101a6:   1d0000ef            jal ra,10376 <printf>
   101aa:   0411                    c.addi  s0,4
   101ac:   ff241ae3            bne s0,s2,101a0 <main+0x4a>
   101b0:   4611                    c.li    a2,4
   101b2:   4595                    c.li    a1,5
   101b4:   0028                    c.addi4spn  a0,sp,8
   101b6:   022000ef            jal ra,101d8 <statistics>
   101ba:   0005059b            addiw   a1,a0,0
   101be:   6575                    c.lui   a0,0x1d
   101c0:   c4850513            addi    a0,a0,-952 # 1cc48 <__clzdi2+0x3a>
   101c4:   1b2000ef            jal ra,10376 <printf>
   101c8:   4501                    c.li    a0,0
   101ca:   60a6                    c.ldsp  ra,72(sp)
   101cc:   6406                    c.ldsp  s0,64(sp)
   101ce:   74e2                    c.ldsp  s1,56(sp)
   101d0:   7942                    c.ldsp  s2,48(sp)
   101d2:   79a2                    c.ldsp  s3,40(sp)
   101d4:   6161                    c.addi16sp  sp,80
   101d6:   8082                    c.jr    ra
```

Рис.12. Исполняемый файл.

```
00000000000101d8 <statistics>:
   101d8:   4785                    c.li    a5,1
   101da:   02f58f63            beq a1,a5,10218 <statistics+0x40>
   101de:   00450813            addi    a6,a0,4
   101e2:   35f9                    c.addiw a1,-2
   101e4:   02059793            slli    a5,a1,0x20
   101e8:   01e7d593            srli    a1,a5,0x1e
   101ec:   00850893            addi    a7,a0,8
   101f0:   98ae                    c.add   a7,a1
   101f2:   95c2                    c.add   a1,a6
   101f4:   a005                    c.j 10214 <statistics+0x3c>
   101f6:   0791                    c.addi  a5,4
   101f8:   00b78b63            beq a5,a1,1020e <statistics+0x36>
   101fc:   4398                    c.lw    a4,0(a5)
   101fe:   00082683            lw  a3,0(a6)
   10202:   fee6fae3            bgeu    a3,a4,101f6 <statistics+0x1e>
   10206:   43d4                    c.lw    a3,4(a5)
   10208:   c394                    c.sw    a3,0(a5)
   1020a:   c3d8                    c.sw    a4,4(a5)
   1020c:   b7ed                    c.j 101f6 <statistics+0x1e>
   1020e:   0811                    c.addi  a6,4
   10210:   01180463            beq a6,a7,10218 <statistics+0x40>
   10214:   87aa                    c.mv    a5,a0
   10216:   b7dd                    c.j 101fc <statistics+0x24>
   10218:   00261793            slli    a5,a2,0x2
   1021c:   953e                    c.add   a0,a5
   1021e:   4108                    c.lw    a0,0(a0)
   10220:   8082                    c.jr    ra
```

Рис.13. Исполняемый файл.

По рисункам 12 и 13 видим, что адресация изменилась на абсолютную.

3.Создание статической библиотеки

Статическая библиотека (static library) является, по сути, архивом (набором, коллекцией) объектных файлов, среди которых компоновщик выбирает «полезные» для данной программы: объектный файл считается «полезным», если в нем определяется еще не разрешенный компоновщиком символ.

Создаем объектный файл statistics.o и собираем в библиотеку командами:

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

riscv64-unknown-elf-ar -rsc libStatistics.a statistics.o

И можем получить список символов libStatistics.a:

riscv64-unknown-elf-nm libStatistics.a

```
C:\Users\User\CLionProjects\forRISC>riscv64-unknown-elf-nm libStatistics.a

statistics.o:
0000000000000040 t .L2
000000000000003c t .L3
000000000000001e t .L4
0000000000000024 t .L5
0000000000000036 t .L7
0000000000000000 T statistics
```

Рис.14. Список символов файла libStatistics.a.

В выводе утилиты “nm” кодом “T” обозначаются символы, определенные в соответствующем объектном файле.

Сделаем исполняемый файл тестовой программы:

riscv64-unknown-elf-gcc -march=rv64iac -mabi=lp64 -O1 main.c libStatistics.a -o main.out

Для прочтения используем:

riscv64-unknown-elf-objdump -t main.out >main.ds

```
224 000000000001cbb4 g     F .text  000000000000000a .hidden __umoddi3
225 000000000001b62c g     F .text  0000000000000026 _isatty
226 000000000001fd00 g     O .sdata 0000000000000008 _global_impure_ptr
227 0000000000019f3a g     F .text  000000000000040a _realloc_r
228 0000000000010240 g     F .text  000000000000006a __libc_init_array
229 000000000001cb88 g     F .text  000000000000002c .hidden __udivdi3
230 00000000000101d8 g     F .text  000000000000004a statistics
231 0000000000019904 g     F .text  0000000000000022 _fputwc_r
232 000000000001d3d0 g     O .rodata    0000000000000028 __mprec_bigtens
233 000000000001599a g     F .text  00000000000000ee __s2b
```

Рис.15. Таблица символов main.out.

Видим, что в эту таблицу входит содержание объектного файла statistics.o.

Создание make-файлов

Используя примеры с сайта курса, было написано 2 make-файла.

```
1  CC=riscv64-unknown-elf-gcc
2  AR=riscv64-unknown-elf-ar
3  CFLAGS=-march=rv64iac -mabi=lp64
4
5  all: lib
6
7  lib: statistics.o
8          $(AR) -rsc libStatistics.a statistics.o
9          del -f *.o
10 statistics.o: statistics.c
11         $(CC) $(CFLAGS) -c statistics.c -o statistics.o
```

Рис.16. Make-файл для создания библиотеки make_lib.

```
TARGET=main
CC=riscv64-unknown-elf-gcc
CFLAGS=-march=rv64iac -mabi=lp64

all:
	make -f make_lib
	$(CC) $(CFLAGS) main.c libStatistics.a -o $(TARGET)
	del -f *.o *.a
```

Рис.17. Make-файл для создания приложения make_app.

```
26.04.2021  19:12    <DIR>          .
26.04.2021  19:12    <DIR>          ..
25.04.2021  22:41               485 main.c
25.04.2021  20:28               186 main.h
26.04.2021  00:40           235 534 make.exe
26.04.2021  01:12               176 make_app
26.04.2021  01:12               256 make_lib
25.04.2021  22:42               398 statistics.c
               6 файлов        237 035 байт
               2 папок  50 992 173 056 байт свободно
```

Рис.18. До исполнения make-файлов.

```
26.04.2021  19:13    <DIR>          .
26.04.2021  19:13    <DIR>          ..
26.04.2021  19:13             1 724 libStatistics.a
25.04.2021  22:41               485 main.c
25.04.2021  20:28               186 main.h
26.04.2021  00:40           235 534 make.exe
26.04.2021  01:12               176 make_app
26.04.2021  01:12               256 make_lib
25.04.2021  22:42               398 statistics.c
               7 файлов        238 759 байт
               2 папок  50 991 104 000 байт свободно
```

Рис.19. После исполнения make_lib.

```
26.04.2021  19:13    <DIR>          .
26.04.2021  19:13    <DIR>          ..
26.04.2021  19:13           143 400 main
25.04.2021  22:41               485 main.c
25.04.2021  20:28               186 main.h
26.04.2021  00:40           235 534 make.exe
26.04.2021  01:12               176 make_app
26.04.2021  01:12               256 make_lib
25.04.2021  22:42               398 statistics.c
               7 файлов        380 435 байт
               2 папок  50 990 419 968 байт свободно
```

Рис.20. После исполнения make_app.

Вывод

В ходе выполнения лабораторной работы, был сделана функция и тестирующая ее функция на языке C для поиска k-ой порядковой статистики. Далее была выполнена сборка по шагам для RISC-V. Была создана библиотека libStatistics.a, а также make-файлы для её сборки и сборки тестовой программы с использованием библиотеки.