

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский политехнический университет Петра Великого»

Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Задание №4

Определение НОД всех элементов массива

Дисциплина: «Низкоуровневое программирование»

Выполнил студент гр. 3530901/90002

Сергиенко Н.И.

Преподаватель

Степанов Д.С.

Санкт-Петербург

2021

Оглавление

| | |
|--|----|
| Задание..... | 3 |
| 1. Программа на языке С..... | 4 |
| 2. Сборка программы «по шагам»..... | 6 |
| Препроцессирование..... | 6 |
| Компиляция | 7 |
| Ассемблирование | 9 |
| Компоновка..... | 12 |
| 3. Создание статической библиотеки и make-файлов | 14 |
| Вывод..... | 18 |

Задание

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, находящую НОД массива чисел. 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 исполнимом 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 на языке C

Листинг 1.1. Заголовочный файл GCD.h

```
#ifndef LAB4_GCD_H
#define LAB4_GCD_H
void GCD(int array[]);
#endif //LAB4_GCD_H
```

Листинг 1.2. Основной файл GCD.c

```
#include "GCD.h"
void GCD(int array[]) {
    if (sizeof(array) > 1) {
        int first = array[0];
        int second = 0;
        for (int i = 1; i < 5; i++) {
            second = array[i];
            while (first != 1 && first != second && second != 1) {
                if (first >= second) {
                    first = first - second;
                }
                else {
                    second = second - first;
                }
            }
            if (first == 1 || second == 1) {
                break;
            }
        }
        array[1] = first;
    }
}
```

Листинг 1.3. Тестовая программа main.c

```
1  #include <stdio.h>
2  #include "GCD.h"
3
4  int main() {
5      int array[5] = {35, 70, 105, 100, 45};
6      GCD(array);
7      printf( _Format: "GCD = %d", array[1]);
8      return 0;
9  }
```

2. Сборка программы «по шагам»

Препроцессирование

Препроцессирование выполняется следующими командами:

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

```
riscv64-unknown-elf-gcc.exe -march=rv64iac -mabi=lp64 -O1 -E GCD.c -o GCD.i
```

Результат препроцессирования содержится в файлах main.i и GCD.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 2.1. Файл main.i (фрагмент)

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

.....

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

# 3 "GCD.h"
void GCD(int array[]);
# 3 "main.c" 2

int main() {
    int array[5] = {35, 70, 105, 100, 45};
    GCD(array);
    printf("GCD = %d", array[1]);
    return 0;
}
```

Листинг 2.2. Файл GCD.i

```
# 1 "GCD.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "GCD.c"
void GCD(int array[]) {
    if (sizeof(array) > 1) {
        int first = array[0];
        int second = 0;
        for (int i = 1; i < 5; i++) {
            second = array[i];
            while (first != 1 && first != second && second != 1) {
```

```

    if (first >= second) {
        first = first - second;
    }
    else {
        second = second - first;
    }
}
if (first == 1 || second == 1) {
    break;
}
}
array[1] = first;
}
}

```

Компиляция

Компиляция осуществляется следующими командами:

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

```
riscv64-unknown-elf-gcc.exe -march=rv64iac -mabi=lp64 -O1 -S GCD.i -o GCD.s
```

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

Листинг 2.3. 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      "GCD = %d"
.text
.align 1
.globl main
.type main, @function
main:
    addi    sp,sp,-48
    sd      ra,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    a0,sp,8

```

```

        call    GCD
        lw      a1,12(sp)
        lui     a0,%hi(.LC1)
        addi    a0,a0,%lo(.LC1)
        call    printf
        li      a0,0
        ld      ra,40(sp)
        addi    sp,sp,48
        jr      ra
        .size   main, .-main
        .section .rodata
        .align 3
        .set    .LANCHOR0,. + 0
.LC0:
        .word   35
        .word   70
        .word   105
        .word   100
        .word   45
        .ident   "GCC: (SiFive GCC-Metal 10.2.0-2020.12.8) 10.2.0"

```

Листинг 2.4. Файл GCD.s

```

        .file   "GCD.c"
        .option nopic
        .attribute arch, "rv64i2p0_a2p0_c2p0"
        .attribute unaligned_access, 0
        .attribute stack_align, 16
        .text
        .align 1
        .globl GCD
        .type   GCD, @function
GCD:
        lw      a5,0(a0)
        addi    a2,a0,4
        addi    a1,a0,20
        li      a3,1
        j       .L8
.L5:
        subw    a4,a4,a5
.L6:
        beq     a5,a3,.L2
        beq     a5,a4,.L9
        beq     a4,a3,.L2
.L7:
        blt     a5,a4,.L5
        subw    a5,a5,a4
        j       .L6
.L9:
        mv      a5,a4
.L3:
        addi    a2,a2,4
        beq     a2,a1,.L2
.L8:
        lw      a4,0(a2)
        beq     a5,a3,.L2
        beq     a4,a5,.L3
        bne     a4,a3,.L7
.L2:
        sw      a5,4(a0)

```



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

Ассемблирование

Ассемблирование осуществляется следующими командами:

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

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

Листинг 2.5. Заголовки секций файла main.o

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

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

Sections:
Idx Name          Size      VMA           LMA           File off  Algn
 0 .text          0000003c 0000000000000000 0000000000000000 00000040 2**1
                CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data           00000000 0000000000000000 0000000000000000 0000007c 2**0
                CONTENTS, ALLOC, LOAD, DATA
 2 .bss            00000000 0000000000000000 0000000000000000 0000007c 2**0
                ALLOC
 3 .rodata.str1.8 00000009 0000000000000000 0000000000000000 00000080 2**3
                CONTENTS, ALLOC, LOAD, READONLY, DATA
 4 .rodata         00000014 0000000000000000 0000000000000000 00000090 2**3
                CONTENTS, ALLOC, LOAD, READONLY, DATA
 5 .comment        00000031 0000000000000000 0000000000000000 000000a4 2**0
                CONTENTS, READONLY
 6 .riscv.attributes 00000026 0000000000000000 0000000000000000 000000d5 2**0
                CONTENTS, READONLY
```

Листинг 2.6. Таблица символов файла main.o

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

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

SYMBOL TABLE:
0000000000000000 1  df *ABS* 0000000000000000 main.c
0000000000000000 1  d .text 0000000000000000 .text
0000000000000000 1  d .data 0000000000000000 .data
0000000000000000 1  d .bss 0000000000000000 .bss
```

| | | | | | |
|----------------------|---|---|-------------------|----------------------|-------------------|
| 00000000000000000000 | 1 | d | .rodata.str1.8 | 00000000000000000000 | .rodata.str1.8 |
| 00000000000000000000 | 1 | d | .rodata | 00000000000000000000 | .rodata |
| 00000000000000000000 | 1 | | .rodata | 00000000000000000000 | .LANCHOR0 |
| 00000000000000000000 | 1 | | .rodata.str1.8 | 00000000000000000000 | .LC1 |
| 00000000000000000000 | 1 | d | .comment | 00000000000000000000 | .comment |
| 00000000000000000000 | 1 | d | .riscv.attributes | 00000000000000000000 | .riscv.attributes |
| 00000000000000000000 | g | F | .text | 0000000000000003c | main |
| 00000000000000000000 | | | *UND* | 00000000000000000000 | GCD |
| 00000000000000000000 | | | *UND* | 00000000000000000000 | printf |

В таблице символов main.o имеется запись: символ “GCD” типа *UND*. Эта запись означает, что символ “GCD” 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 и к символу “printf”.

Листинг 2.7. Таблица перемещений файла main.o

```
riscv64-unknown-elf-objdump.exe -d -M no-aliases -r main.o
```

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

Disassembly of section .text:

```
000000000000000000 <main>:
```

```

0: 7179          c.addi16sp    sp,-48
2: f406          c.sdsp    ra,40(sp)
4: 000007b7      lui    a5,0x0
               4: R_RISCV_HI20 .LANCHOR0
               4: R_RISCV_RELAX    *ABS*
8: 00078793      addi    a5,a5,0 # 0 <main>
               8: R_RISCV_LO12_I    .LANCHOR0
               8: R_RISCV_RELAX    *ABS*

c: 6398          c.ld    a4,0(a5)
e: e43a          c.sdsp    a4,8(sp)
10: 6798         c.ld    a4,8(a5)
12: e83a         c.sdsp    a4,16(sp)
14: 4b9c         c.lw    a5,16(a5)
16: cc3e         c.swsp    a5,24(sp)
18: 0028         c.addi4spn    a0,sp,8
1a: 00000097     auipc    ra,0x0
               1a: R_RISCV_CALL    GCD
               1a: R_RISCV_RELAX    *ABS*

```

```

1e: 000080e7      jalr  ra,0(ra) # 1a <main+0x1a>
22: 45b2          c.lwsp a1,12(sp)
24: 00000537      lui   a0,0x0
                24: R_RISCV_HI20      .LC1
                24: R_RISCV_RELAX      *ABS*
28: 00050513      addi  a0,a0,0 # 0 <main>
                28: R_RISCV_LO12_I   .LC1
                28: R_RISCV_RELAX      *ABS*
2c: 00000097      auipc ra,0x0
                2c: R_RISCV_CALL      printf
                2c: R_RISCV_RELAX      *ABS*
30: 000080e7      jalr  ra,0(ra) # 2c <main+0x2c>
34: 4501          c.li   a0,0
36: 70a2          c.ldsp ra,40(sp)
38: 6145          c.addi16sp sp,48
3a: 8082          c.jr   ra

```

Листинг 2.8. Заголовки секций файла GCD.o

```

GCD.o:  file format elf64-littleriscv
Sections:
Idx Name          Size      VMA           LMA           File off  Algn
 0 .text          0000003e 0000000000000000 0000000000000000 00000040 2**1
                CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data          00000000 0000000000000000 0000000000000000 0000007e 2**0
                CONTENTS, ALLOC, LOAD, DATA
 2 .bss           00000000 0000000000000000 0000000000000000 0000007e 2**0
                ALLOC
 3 .comment       00000031 0000000000000000 0000000000000000 0000007e 2**0
                CONTENTS, READONLY
 4 .riscv.attributes 00000026 0000000000000000 0000000000000000 000000af 2**0
                CONTENTS, READONLY

```

Листинг 2.9. Таблица символов файла GCD.o

```

GCD.o:  file format elf64-littleriscv

SYMBOL TABLE:
0000000000000000 1  df *ABS* 0000000000000000 GCD.c
0000000000000000 1  d .text 0000000000000000 .text
0000000000000000 1  d .data 0000000000000000 .data
0000000000000000 1  d .bss 0000000000000000 .bss
0000000000000000 2c 1  .text 0000000000000000 .L8

```

```

0000000000000003a 1 .text 0000000000000000 .L2
00000000000000024 1 .text 0000000000000000 .L9
0000000000000000e 1 .text 0000000000000000 .L5
00000000000000010 1 .text 0000000000000000 .L6
00000000000000026 1 .text 0000000000000000 .L3
0000000000000001c 1 .text 0000000000000000 .L7
0000000000000000 1 d .comment 0000000000000000 .comment
0000000000000000 1 d .riscv.attributes 0000000000000000 .riscv.attributes
0000000000000000 g F .text 000000000000003e GCD

```

Компоновка

Компоновка осуществляется следующей командой:

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

Листинг 2.10. Исполняемый файл a.out (фрагмент)

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

a.out: file format elf64-littleriscvDisassembly of section .text:

0000000000010156 <main>:

```

10156: 7179      c.addi16sp      sp,-48
10158: f406      c.sdsp         ra,40(sp)
1015a: 67f5      c.lui          a5,0x1d
1015c: b2078793  addi          a5,a5,-1248 # 1cb20 <__clzdi2+0x4e>
10160: 6398      c.ld           a4,0(a5)
10162: e43a      c.sdsp         a4,8(sp)
10164: 6798      c.ld           a4,8(a5)
10166: e83a      c.sdsp         a4,16(sp)
10168: 4b9c      c.lw           a5,16(a5)
1016a: cc3e      c.swsp         a5,24(sp)
1016c: 0028      c.addi4spn     a0,sp,8
1016e: 018000ef  jal           ra,10186 <GCD>
10172: 45b2      c.lwsp        a1,12(sp)
10174: 6575      c.lui          a0,0x1d
10176: b1050513  addi          a0,a0,-1264 # 1cb10 <__clzdi2+0x3e>
1017a: 19e000ef  jal           ra,10318 <printf>
1017e: 4501      c.li           a0,0
10180: 70a2      c.ldsp         ra,40(sp)
10182: 6145      c.addi16sp     sp,48
10184: 8082      c.jr           ra

```

0000000000010186 <GCD>:

| | | | |
|--------|----------|--------|------------------------|
| 10186: | 411c | c.lw | a5,0(a0) |
| 10188: | 00450613 | addi | a2,a0,4 |
| 1018c: | 01450593 | addi | a1,a0,20 |
| 10190: | 4685 | c.li | a3,1 |
| 10192: | a005 | c.j | 101b2 <GCD+0x2c> |
| 10194: | 9f1d | c.subw | a4,a5 |
| 10196: | 02d78563 | beq | a5,a3,101c0 <GCD+0x3a> |
| 1019a: | 00e78863 | beq | a5,a4,101aa <GCD+0x24> |
| 1019e: | 02d70163 | beq | a4,a3,101c0 <GCD+0x3a> |
| 101a2: | fee7c9e3 | blt | a5,a4,10194 <GCD+0xe> |
| 101a6: | 9f99 | c.subw | a5,a4 |
| 101a8: | b7fd | c.j | 10196 <GCD+0x10> |
| 101aa: | 87ba | c.mv | a5,a4 |
| 101ac: | 0611 | c.addi | a2,4 |
| 101ae: | 00b60963 | beq | a2,a1,101c0 <GCD+0x3a> |
| 101b2: | 4218 | c.lw | a4,0(a2) |
| 101b4: | 00d78663 | beq | a5,a3,101c0 <GCD+0x3a> |
| 101b8: | fef70ae3 | beq | a4,a5,101ac <GCD+0x26> |
| 101bc: | fed713e3 | bne | a4,a3,101a2 <GCD+0x1c> |
| 101c0: | c15c | c.sw | a5,4(a0) |
| 101c2: | 8082 | c.jr | ra... |

3. Создание статической библиотеки и make-файлов

Выделим из программы GCD.c функцию вычитания из большего числа меньшее в отдельную программу Hmm.c. Объединим GCD.c и Hmm.c в статическую библиотеку GCDlib, тестовую программу main.c оставим без изменений.

Для создания статической библиотеки получим объектные файлы всех используемых программ: GCD.o и Hmm.o.

```
riscv64-unknown-elf-gcc.exe -march=rv64iac -mabi=lp64 -O1 -c GCD.c -o GCD.o  
riscv64-unknown-elf-gcc.exe -march=rv64iac -mabi=lp64 -O1 -c Hmm.c -o Hmm.o
```

Объединим получившиеся файлы в одну библиотеку следующей командой:

```
riscv64-unknown-elf-ar.exe -rsc GCDlib.a GCD.o Hmm.o
```

Используя получившуюся библиотеку, соберем исполняемый файл программы следующей командой:

```
riscv64-unknown-elf-gcc.exe -march=rv64iac -mabi=lp64 -O1 --save-temps main.c  
GCDlib.a
```

Листинг 3.1. Таблица символов исполняемого файла (фрагмент)

```
riscv64-unknown-elf-objdump.exe -t a.out
```

```
a.out: file format elf64-littleriscv  
  
SYMBOL TABLE:  
000000000000100b0 1 d .text 0000000000000000 .text  
...  
0000000000000000 1 df *ABS* 0000000000000000 main.c  
0000000000000000 1 df *ABS* 0000000000000000 GCD.c  
0000000000000000 1 df *ABS* 0000000000000000 Hmm.c  
...  
000000000000101b6 g F .text 0000000000000030 Hmm  
...  
00000000000010156 g F .text 0000000000000030 main  
...  
00000000000010186 g F .text 0000000000000030 GCD  
...
```

```
0000000000001566e g F .text 00000000000000012 _Bfree
```

Можно заметить, что в состав программы вошло содержимое объектных файлов GCD.o и Hmm.o.

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

Листинг 3.2. Makefile для создания статической библиотеки

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

# Исходные файлы, необходимые для сборки библиотеки
OBJS= GCD.c \
      Hmm.c

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

# Файл библиотеки
MYLIBNAME = GCDlib.a

# Параметры компиляции
CFLAGS= -march=rv64iac -mabi=lp64 -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)

# $^ = (GCD.o, Hmm.o)
$(MYLIBNAME): GCD.o Hmm.o
```

```
$(AR) -rsc $@ $^
```

Листинг 3.3. Makefile для сборки исполняемого файла

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

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

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

# Параметры компиляции
CFLAGS= -march=rv64iac -mabi=lp64 -O1 --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
```

Для запуска Makefile воспользуемся GNU Make

Листинг 3.4. Запуск Makefile

```
C:\Users\bdoubleo\Desktop\lab4\lib>make
```

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

Листинг 3.5. Таблица символов исполняемого
файла, созданного с помощью Makefile (фрагмент)

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

SYMBOL TABLE:
000000000000100b0 1  d .text 0000000000000000 .text
...
0000000000000000 1  df *ABS* 0000000000000000 main.c
0000000000000000 1  df *ABS* 0000000000000000 GCD.c
0000000000000000 1  df *ABS* 0000000000000000 Hmm.c
...
000000000000101b6 g  F .text 0000000000000030 Hmm
...
00000000000010156 g  F .text 0000000000000030 main
...
00000000000010186 g  F .text 0000000000000030 GCD
...
0000000000001566e g  F .text 0000000000000012 _Bfree
```

Видим, созданный исполняемый файл аналогичен тому, что был создан через терминал.

Попробуем собрать нашу программу с помощью обычного *gcc*:

```
C:\Users\bdoubleo\Desktop\lab4\app>gcc main.c Hmm.c GCD.c
GCD.c: In function 'GCD':
GCD.c:7:15: warning: 'sizeof' on array function parameter 'array' will return size of 'int *' [-Wsizeof-array-argument]
   7 |     if (sizeof(array) > 1) {
     |           ^
GCD.c:6:14: note: declared here
   6 | void GCD(int array[]) {
     |           ~~~~~^~~~~~
GCD.c:12:21: warning: implicit declaration of function 'Hmm' [-Wimplicit-function-declaration]
   12 |         first = Hmm(first, second);
     |                   ^~~
C:\Users\bdoubleo\Desktop\lab4\app>a.exe
GCD = 45
```

Как можно заметить, программа выводит верный результат.

Вывод

В ходе лабораторной работы изучена пошаговая компиляция программы на языке C. Также была создана статическая библиотека и произведена сборка программы с помощью Makefile.