Санкт-Петербургский политехнический университет Петра Великого

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

Лабораторная работа № 4

Дисциплина: Низкоуровневое программирование

Тема: Раздельная компиляция

Выполнил студент гр. 3530901/90004 _____ Кривицкий В.В

Преподаватель _____ Алексюк А. О

“___”_____ 2021 г.

Санкт-Петербург
2021

Оглавление

Техническое 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 сортировки «пузырьком». 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 С**

Листинг 1.1. Заголовочный файл bSort.h

```
#ifndef LAB4_BSORT_H
#define LAB4_BSORT_H
void bSort(int *num, int size);
#endif
```

Листинг 1.2. Основной файл bSort.c

```
#include "bSort.h"
void bSort(int *num, int size) {
    for (int i = 0; i < size - 1; i++) {
        for (int j = (size - 1); j > i; j--) {
            if (num[j - 1] > num[j]) {
                int temp = num[j - 1];
                num[j - 1] = num[j];
                num[j] = temp;
            }
        }
    }
}
```

Листинг 1.3. Тестовая программа main.c

```
#include <stdio.h>
#include "bSort.h"
int main() {
    int a[10] = {100, 1, 4, 0, 1000, 52, 26, 89, -35, 389};
    bSort(a, 10);
    for (int i = 0; i < 10; i++) {
        printf("%d ", a[i]);
    }
    return 0;
}
```

```
C:\Users\Vadim\CLionProjects\lab4\cmake-build-debug\lab4.exe
-35 0 1 4 26 52 89 100 389 1000
Process finished with exit code 0
```

2. Сборка программы «по шагам»

Препроцессирование

Препроцессирование выполняется следующими командами:

```
riscv64-unknown-elf-gcc.exe -O1 -E main.c -o main.i

riscv64-unknown-elf-gcc.exe -O1 –E bSort.c -o bSort.i
```

Результат препроцессирования содержится в файлах main.i и bSort.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 "bSort.h" 1



# 3 "bSort.h"
void bSort(int *num, int size);
# 3 "main.c" 2

int main() {
        int array[10] = { 100, 1, 4, 0, 1000, 52, 26, 89, -35, 389};
        bSort(a, 10);
        for (int i = 0; i < 10; i++) {
                printf("%d ", a[i]);
        }
        return 0;
}
```

Листинг 2.2. Файл bSort.i

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

void bSort(int *num, int size);
# 2 "bSort.c" 2
```

```
void bSort(int *num, int size) {
    for (int i = 0; i < size - 1; i++) {
        for (int j = (size - 1); j > i; j--) {
            if (num[j - 1] > num[j]) {
                int temp = num[j - 1];
                num[j - 1] = num[j];
                num[j] = temp;
            }
        }
    }
}
```

Компиляция

Компиляция осуществляется следующими командами:

```
riscv64-unknown-elf-gcc.exe -O1 –S main.i -o main.s

riscv64-unknown-elf-gcc.exe -O1 -S bSort.i -o bSort.s
```

Наибольший интерес представляет файл main.s, так как в нем можно заметить обращение к подпрограмме bSort (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_m2p0_a2p0_f2p0_d2p0_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,-80
        sd  ra,72(sp)
        sd  s0,64(sp)
        sd  s1,56(sp)
        sd  s2,48(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    bSort
        addi    s0,sp,8
        addi    s2,sp,48
        lui     s1,%hi(.LC1)
.L2:
        lw      a1,0(s0)
        addi    a0,s1,%lo(.LC1)
        call    printf
        addi    s0,s0,4
        bne     s0,s2,.L2
        li      a0,0
        ld      ra,72(sp)
        ld      s0,64(sp)
        ld      s1,56(sp)
        ld      s2,48(sp)
        addi    sp,sp,80
        jr      ra
        .size   main, .-main
        .section        .rodata
        .align  3
        .set    .LANCHOR0,. + 0
.LC0:
        .word   100
        .word   1
        .word   4
        .word   0
        .word   1000
        .word   52
        .word   26
        .word   89
        .word   -35
        .word   389
        .ident  "GCC: (SiFive GCC-Metal 10.2.0-2020.12.8) 10.2.0"
```

Листинг 2.4. Файл bSort.s

```
        .file   "bSort.c"
        .option nopic
        .attribute arch, "rv64i2p0_m2p0_a2p0_f2p0_d2p0_c2p0"
        .attribute unaligned_access, 0
        .attribute stack_align, 16
        .text
        .align  1
        .globl  bSort
        .type   bSort, @function
bSort:
        addiw   a7,a1,-1
        ble     a7,zero,.L1
        mv      t1,a7
        slli    t3,a7,2
        add     t3,a0,t3
        slli    a6,a1,2
```

```
        add     a6,a0,a6
        li      a0,0
        addi    a6,a6,-8
        addiw   a1,a1,-2
        j       .L3
.L4:
        addi    a5,a5,-4
        beq     a5,a2,.L7
.L5:
        lw      a4,-4(a5)
        lw      a3,0(a5)
        ble     a4,a3,.L4
        sw      a3,-4(a5)
        sw      a4,0(a5)
        j       .L4
.L7:
        addiw   a0,a0,1
        beq     a0,t1,.L1
.L3:
        ble     a7,a0,.L7
        subw    a2,a1,a0
        slli    a5,a2,32
        srli    a2,a5,30
        sub     a2,a6,a2
        mv      a5,t3
        j       .L5
.L1:
        ret
        .size   bSort, .-bSort
        .ident  "GCC: (SiFive GCC-Metal 10.2.0-2020.12.8) 10.2.0"
```

Ассемблирование

Ассемблирование осуществляется следующими командами:

```
riscv64-unknown-elf-gcc.exe -v -c main.s -o main.o

riscv64-unknown-elf-gcc.exe -v -c bSort.s -o bSort.o
```

Листинг 2.5. Заголовки секций файла main.o

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

```
C:\Users\Vadim\CLionProjects\lab4>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         0000005e  0000000000000000  0000000000000000  00000040  2**1
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  0000000000000000  0000000000000000  0000009e  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  0000000000000000  0000000000000000  0000009e  2**0
                  ALLOC
  3 .rodata.str1.8 00000004  0000000000000000  0000000000000000  000000a0  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .rodata       00000028  0000000000000000  0000000000000000  000000a8  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .comment      00000031  0000000000000000  0000000000000000  000000d0  2**0
                  CONTENTS, READONLY
  6 .riscv.attributes 00000035  0000000000000000  0000000000000000  00000101  2**0
                  CONTENTS, READONLY
```

Листинг 2.6. Таблица символов файла main.o

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

```
C:\Users\Vadim\CLionProjects\lab4>riscv64-unknown-elf-objdump.exe -t 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
000000000000003c l       .text  0000000000000000 .L2
0000000000000000 l    d  .comment       0000000000000000 .comment
0000000000000000 l    d  .riscv.attributes      0000000000000000 .riscv.attributes
0000000000000000 g     F .text  000000000000005e main
0000000000000000         *UND*  0000000000000000 bSort
0000000000000000         *UND*  0000000000000000 printf
```

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

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:   000007b7                lui     a5,0x0
                        a: R_RISCV_HI20 .LANCHOR0
                        a: R_RISCV_RELAX        *ABS*
   e:   00078793                addi    a5,a5,0 # 0 <main>
                        e: R_RISCV_LO12_I       .LANCHOR0
                        e: R_RISCV_RELAX        *ABS*
  12:   638c                    c.ld    a1,0(a5)
  14:   6790                    c.ld    a2,8(a5)
  16:   6b94                    c.ld    a3,16(a5)
  18:   6f98                    c.ld    a4,24(a5)
  1a:   739c                    c.ld    a5,32(a5)
  1c:   e42e                    c.sdsp  a1,8(sp)
  1e:   e832                    c.sdsp  a2,16(sp)
  20:   ec36                    c.sdsp  a3,24(sp)
  22:   f03a                    c.sdsp  a4,32(sp)
  24:   f43e                    c.sdsp  a5,40(sp)
  26:   45a9                    c.li    a1,10
  28:   0028                    c.addi4spn      a0,sp,8
  2a:   00000097                auipc   ra,0x0
                        2a: R_RISCV_CALL        bSort
                        2a: R_RISCV_RELAX       *ABS*
  2e:   000080e7                jalr    ra,0(ra) # 2a <main+0x2a>
  32:   0020                    c.addi4spn      s0,sp,8
  34:   03010913                addi    s2,sp,48
  38:   000004b7                lui     s1,0x0
                        38: R_RISCV_HI20        .LC1
                        38: R_RISCV_RELAX       *ABS*

000000000000003c <.L2>:
  3c:   400c                    c.lw    a1,0(s0)
  3e:   00048513                addi    a0,s1,0 # 0 <main>
                        3e: R_RISCV_LO12_I      .LC1
                        3e: R_RISCV_RELAX       *ABS*
  42:   00000097                auipc   ra,0x0
                        42: R_RISCV_CALL        printf
                        42: R_RISCV_RELAX       *ABS*
  46:   000080e7                jalr    ra,0(ra) # 42 <.L2+0x6>
  4a:   0411                    c.addi  s0,4
  4c:   ff2418e3                bne     s0,s2,3c <.L2>
                        4c: R_RISCV_BRANCH      .L2
  50:   4501                    c.li    a0,0
  52:   60a6                    c.ldsp  ra,72(sp)
  54:   6406                    c.ldsp  s0,64(sp)
  56:   74e2                    c.ldsp  s1,56(sp)
  58:   7942                    c.ldsp  s2,48(sp)
  5a:   6161                    c.addi16sp      sp,80
  5c:   8082                    c.jr    ra
```

Листинг 2.8. Заголовки секций файла bSort.o

```
C:\Users\Vadim\CLionProjects\lab4>riscv64-unknown-elf-objdump.exe -h bSort.o

bSort.o:     file format elf64-littleriscv

Sections:
Idx Name          Size      VMA               LMA               File off  Algn
  0 .text         00000056  0000000000000000  0000000000000000  00000040  2**1
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  0000000000000000  0000000000000000  00000096  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  0000000000000000  0000000000000000  00000096  2**0
                  ALLOC
  3 .comment      00000031  0000000000000000  0000000000000000  00000096  2**0
                  CONTENTS, READONLY
  4 .riscv.attributes 00000035  0000000000000000  0000000000000000  000000c7  2**0
                  CONTENTS, READONLY
```

Листинг 2.9. Таблица символов файла bSort.o

```
C:\Users\Vadim\CLionProjects\lab4>riscv64-unknown-elf-objdump.exe -t bSort.o

bSort.o:     file format elf64-littleriscv

SYMBOL TABLE:
0000000000000000 l    df *ABS*  0000000000000000 bSort.c
0000000000000000 l    d  .text  0000000000000000 .text
0000000000000000 l    d  .data  0000000000000000 .data
0000000000000000 l    d  .bss   0000000000000000 .bss
0000000000000054 l       .text  0000000000000000 .L1
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  0000000000000056 bSort
```

Компоновка

Компоновка осуществляется следующей командой:

```
riscv64-unknown-elf-gcc.exe -v main.o bSort.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-littleriscv
```

```
Disassembly of section .text:

…

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: 67f5                c.lui   a5,0x1d
   10162: 85878793            addi    a5,a5,-1960 # 1c858 <__clzdi2+0x3c>
   10166: 638c                c.ld    a1,0(a5)
   10168: 6790                c.ld    a2,8(a5)
   1016a: 6b94                c.ld    a3,16(a5)
   1016c: 6f98                c.ld    a4,24(a5)
   1016e: 739c                c.ld    a5,32(a5)
   10170: e42e                c.sdsp  a1,8(sp)
   10172: e832                c.sdsp  a2,16(sp)
   10174: ec36                c.sdsp  a3,24(sp)
   10176: f03a                c.sdsp  a4,32(sp)
   10178: f43e                c.sdsp  a5,40(sp)
   1017a: 45a9                c.li    a1,10
   1017c: 0028                c.addi4spn      a0,sp,8
   1017e: 02a000ef            jal     ra,101a8 <bSort>
   10182: 0020                c.addi4spn      s0,sp,8
   10184: 03010913            addi    s2,sp,48
   10188: 64f5                c.lui   s1,0x1d
   1018a: 400c                c.lw    a1,0(s0)
   1018c: 85048513            addi    a0,s1,-1968 # 1c850 <__clzdi2+0x34>
   10190: 1c2000ef            jal     ra,10352 <printf>
   10194: 0411                c.addi  s0,4
   10196: ff241ae3            bne     s0,s2,1018a <main+0x34>
   1019a: 4501                c.li    a0,0
   1019c: 60a6                c.ldsp  ra,72(sp)
   1019e: 6406                c.ldsp  s0,64(sp)
   101a0: 74e2                c.ldsp  s1,56(sp)
   101a2: 7942                c.ldsp  s2,48(sp)
   101a4: 6161                c.addi16sp      sp,80
   101a6: 8082                c.jr    ra


…


00000000000101a8 <bSort>:
   101a8: fff5889b            addiw   a7,a1,-1
   101ac: 05105863            bge     zero,a7,101fc <bSort+0x54>
   101b0: 8346                c.mv    t1,a7
   101b2: 00289e13            slli    t3,a7,0x2
   101b6: 9e2a                c.add   t3,a0
```

```
101b8: 00259813          slli    a6,a1,0x2
101bc: 982a              c.add   a6,a0
101be: 4501              c.li    a0,0
101c0: 1861              c.addi  a6,-8
101c2: 35f9              c.addiw a1,-2
101c4: a005              c.j     101e4 <bSort+0x3c>
101c6: 17f1              c.addi  a5,-4
101c8: 00c78b63          beq     a5,a2,101de <bSort+0x36>
101cc: ffc7a703          lw      a4,-4(a5)
101d0: 4394              c.lw    a3,0(a5)
101d2: fee6dae3          bge     a3,a4,101c6 <bSort+0x1e>
101d6: fed7ae23          sw      a3,-4(a5)
101da: c398              c.sw    a4,0(a5)
101dc: b7ed              c.j     101c6 <bSort+0x1e>
101de: 2505              c.addiw a0,1
101e0: 00650e63          beq     a0,t1,101fc <bSort+0x54>
101e4: ff155de3          bge     a0,a7,101de <bSort+0x36>
101e8: 40a5863b          subw    a2,a1,a0
101ec: 02061793          slli    a5,a2,0x20
101f0: 01e7d613          srli    a2,a5,0x1e
101f4: 40c80633          sub     a2,a6,a2
101f8: 87f2              c.mv    a5,t3
101fa: bfc9              c.j     101cc <bSort+0x24>
101fc: 8082              c.jr    ra
```

3. Создание статической библиотеки и make-файлов

Выделим из программы bSort.c функцию смены местами в отдельную программу swap.c. Объединим bSort.c и swap.c в статическую библиотеку bSortlib, тестовую программу main.c оставим без изменений.

Для создания статической библиотеки получим объектные файлы всех используемых программ: bSort.o и swap.o.

```
riscv64-unknown-elf-gcc.exe -O1 -c bSort.c -o bSort.o

riscv64-unknown-elf-gcc.exe -O1 -c swap.c -o swap.o
```

Объединим получившиеся файлы в одну библиотеку следующей командой:

```
riscv64-unknown-elf-ar.exe -rsc bSortlib.a bSort.o swap.o
```

Используя получившуюся библиотеку, соберем исполняемый файл программы следующей командой:

```
riscv64-unknown-elf-gcc.exe -O1 --save-temps main.c bSortlib.a
```

Листинг 3.1. Таблица символов исполняемого файла (фрагмент)

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

```
a.out:     file format elf32-littleriscv

SYMBOL TABLE:
0000000000000000 l    df *ABS*      0000000000000000 main.c
0000000000000000 l    df *ABS*      0000000000000000 bSort.c
0000000000000000 l    df *ABS*      0000000000000000 swap.c

00000000000101a8 g     F .text      0000000000000086 bSort

000000000001022e g     F .text      000000000000000a swap

0000000000010156 g     F .text      0000000000000052 main

000000000001563e g     F .text      0000000000000012 _Bfree
```

Можно заметить, что в состав программы вошло содержимое объектных файлов bSort.o и swap.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= bSort.c  \
      swap.c

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

# Файл библиотеки
MYLIBNAME = bSortlib.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)

# $^ = (bSort.o, swap.o)
$(MYLIBNAME): bSort.o swap.o
	$(AR) -rsc $@ $^
```

Листинг 3.3. Makefile для сборки исполняемого файла

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

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

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

# Параметры компиляции
CFLAGS= -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 программой mingw32-make.exe.

Листинг 3.4. Запуск Makefile

```
make
```

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

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

SYMBOL TABLE:
0000000000000000 l    df *ABS*  0000000000000000 main.c
0000000000000000 l    df *ABS*  0000000000000000 bSort.c
0000000000000000 l    df *ABS*  0000000000000000 swap.c

00000000000101a8 g     F .text  0000000000000086 bSort

0000000000000000 l    df *ABS*  0000000000000000 swap.c

0000000000010156 g     F .text  0000000000000052 main

000000000001563e g     F .text  0000000000000012 _Bfree
```

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

Вывод

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