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

Выполнил студент гр. 3530901/10001 _____Солодовник И.Н.
(подпись)

Принял старший преподаватель _____ Коренев Д.А.
(подпись)

“23” декабря 2022 г.

Санкт-Петербург
2022

Оглавление

Описание задачи:

Разработать на языке C функцию, реализующую загрузку последовательности «коротких» чисел. Числа кодируются обычным для EDSAC образом (в виде псевдоинструкций).

Поместить определение функции в отдельный исходный файл и оформить заголовочный файл. Разработать тестовую программу.

Собрать программу «по шагам». Проанализировать состав выходных файлов. Выделить разработанную функцию в статическую библиотеку и разработать make-файлы.

(Вариант 10)

Алгоритм загрузки «коротких» чисел:

Программа считывает первый символ, сравнивает и записывает его в 4 старших разряда результата и запоминает знак. Затем записываются цифры, сдвигаются влево на 1 разряд, а в младший разряд записывается символ S или L. После этого программа проверяет знак и переходит к следующему значению.

Функция:

Основной файл:

```c
#include "inst_reader.h"

void inst_reader(const char *input, int *output) {
    const char edsac_symbols[] = "PQWERTYUIOJ#SZK*.F@D!HNM&LXGABCV";
    while (*input != '\0') {
        int i = 0;
        while (edsac_symbols[i] != *input) i++;
        short sign = i >= 16 ? 1 : 0;
        *output = (i % 16) << 12;

        input++;
        int mid_res = 0;
        while (*input <= '9' && *input >= '0')
            mid_res = mid_res * 10 + *(input++) - '0';
        *output += mid_res << 1;

        if (*input == 'L') *output += 1;
        *(output++) -= 65536 * sign;
        if (*(++input) == ' ') input++;
    }
}
```

Тестовая программа:

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

int main(void) {
    const char input[] = "P0S .0S W43L M333L P0L *2047L";

    int counter = *input == '\0' ? 0 : 1;
    const char *a = input;

    while (*a != '\0')
        if (*(a++) == ' ') counter++;

    int output[counter];
    inst_reader(input, output);
    for (int i = 0; i < counter; i++) {
        printf( format: "%d ", output[i]);
    }
    return 0;
}
```

Заголовочный файл:

```c
#ifndef LAB4_INST_READER_H
#define LAB4_INST_READER_H
void inst_reader (const char *input, int *output);

#endif
```

Сборка программы:

Препроцессирование:

- Запишем результат препроцессирования main.c и inst_reader.c в файлы main.i и inst_reader.i . Используем команды:

riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 -E main.c -o main.i
riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 -E inst_reader.c -o inst_reader.i

-march = rv32i -mabi = ilp32 – целевым является процессор с базовой архитектурой системы команд RV32i.
-O1 – выполнить простые оптимизации генерируемого кода.
-E – прекратить сборку после препроцессирования.
-o – выходной файл.

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


void inst_reader (const char *input, int *output);
# 2 "inst_reader.c" 2

void inst_reader(const char *input, int *output) {
    const char edsac_symbols[] = "PQWERTYUIOJ#SZK*.F@D!HNM&LXGABCV";
    while (*input != '\0') {
        int i = 0;
        while (edsac_symbols[i] != *input) i++;
        short sign = i >= 16 ? 1 : 0;
        *output = (i % 16) << 12;

        input++;
        int mid_res = 0;
        while (*input <= '9' && *input >= '0')
            mid_res = mid_res * 10 + *(input++) - '0';
        *output += mid_res << 1;

        if (*input == 'L') *output += 1;
        *(output++) -= 65536 * sign;
        if (*(++input) == ' ') input++;
    }
}
```

Текст inst_reader.i

```
# 1 "main.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "main.c"
...
# 2 "main.c" 2
# 1 "inst_reader.h" 1

# 3 "inst_reader.h"
void inst_reader (const char *input, int *output);
# 3 "main.c" 2

int main(void) {
    const char input[] = "P0S .0S W43L M333L P0L *2047L";

    int counter = *input == '\0' ? 0 : 1;
    const char *a = input;

    while (*a != '\0')
        if (*(a++) == ' ') counter++;

    int output[counter];
    inst_reader(input, output);
    for (int i = 0; i < counter; i++) {
        printf("%d ", output[i]);
    }
    return 0;
}
```

Текст main.i

Результат препроцессирования отличается от исходных файлов наличием большого количества новый строк, так как используется стандартная библиотека языка C (#include <stdio.h>).
Появившиеся нестандартные директивы используются для передачи информации из препроцессора в компилятор.

Компиляция:

- Запишем результат компилирования main.i и inst_reader.i в файлы main.s и inst_reader.s . Используем команды:

riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 -S main.i -o main.s
riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 -S inst_reader.i -o inst_reader.s

```
.file   "inst_reader.c"
.option nopic
.attribute arch, "rv32i2p0"
.attribute unaligned_access, 0
.attribute stack_align, 16
.text
.section    .rodata.str1.4,"aMS",@progbits,1
.align  2
.LC0:
.string "PQWERTYUIOJ#SZK*.F@D!HNM&LXGABCV"
.text
.align  2
.globl  inst_reader
.type   inst_reader, @function
        inst_reader:
addi    sp,sp,-64
sw  s0,60(sp)
lui a5,%hi(.LC0)
addi    a5,a5,%lo(.LC0)
lw  t4,0(a5)
lw  t3,4(a5)
lw  t1,8(a5)
lw  a7,12(a5)
lw  a6,16(a5)
lw  a2,20(a5)
lw  a3,24(a5)
lw  a4,28(a5)
sw  t4,12(sp)
sw  t3,16(sp)
sw  t1,20(sp)
sw  a7,24(sp)
sw  a6,28(sp)
sw  a2,32(sp)
sw  a3,36(sp)
sw  a4,40(sp)
lbu a5,32(a5)
sb  a5,44(sp)
lbu a3,0(a0)
```

```
39      beq a3,zero,.L1
40              lbu t2,12(sp)
41      addi    t3,sp,12
42      li  t1,1
43      sub t1,t1,t3
44      li  t4,0
45      li  a7,9
46      li  t0,76
47      li  t6,15
48      li  t5,32
49      j   .L3
50      .L14:
51      mv  a3,t4
52      j   .L5
53      .L7:
54      addi    a1,a1,4
55      sgt a6,a6,t6
56      slli    a6,a6,16
57      lw  a5,-4(a1)
58      sub a6,a5,a6
59      sw  a6,-4(a1)
60      lbu a5,1(a2)
61      sub a5,a5,t5
62      seqz    a5,a5
63      add a5,a2,a5
64      addi    a0,a5,1
65      lbu a3,1(a5)
66      beq a3,zero,.L1
67      .L3:
68      mv  a5,t3
69      mv  a6,t4
70      beq t2,a3,.L13
71      .L4:
72      add a6,t1,a5
73      addi    a5,a5,1
74      lbu a4,0(a5)
75      bne a4,a3,.L4
76      .L13:
```

```
77      srai    s0,a6,31
78      srli    s0,s0,28
79      add a5,a6,s0
80      andi    a5,a5,15
81      sub a5,a5,s0
82      slli    s0,a5,12
83      sw  s0,0(a1)
84      addi    a2,a0,1
85      lbu a0,1(a0)
86      addi    a5,a0,-48
87      andi    a5,a5,0xff
88      bgtu    a5,a7,.L14
89              mv  a3,t4
90      .L6:
91      addi    a2,a2,1
92      slli    a5,a3,2
93      add a5,a5,a3
94      slli    a5,a5,1
95      add a5,a5,a0
96      addi    a3,a5,-48
97      lbu a0,0(a2)
98      addi    a4,a0,-48
99      andi    a4,a4,0xff
100     bleu    a4,a7,.L6
101     .L5:
102     slli    a5,a3,1
103     add a5,a5,s0
104     sw  a5,0(a1)
105     lbu a4,0(a2)
106     bne a4,t0,.L7
107             addi    a5,a5,1
108     sw  a5,0(a1)
109     j   .L7
110     .L1:
111     lw  s0,60(sp)
112     addi    sp,sp,64
113     jr  ra
114     .size   inst_reader, .-inst_reader
115     .ident  "GCC: (SiFive GCC 10.1.0-2020.08.2) 10.1.0"
```

Текст inst_reader.s

```
.file   "main.c"
.option nopic
.attribute arch, "rv32i2p0"
.attribute unaligned_access, 0
.attribute stack_align, 16
.text
.section    .rodata.str1.4,"aMS",@progbits,1
.align  2
.LC1:
.string "%d "
.align  2
.LC0:
.string "P0S .0S W43L M333L P0L *2047L"
.text
.align  2
.globl  main
.type   main, @function
        main:
addi    sp,sp,-80
sw  ra,76(sp)
sw  s0,72(sp)
sw  s1,68(sp)
sw  s2,64(sp)
sw  s3,60(sp)
sw  s4,56(sp)
addi    s0,sp,80
lui a4,%hi(.LC0)
addi    a4,a4,%lo(.LC0)
lw  a5,0(a4)
lw  a7,4(a4)
lw  a6,8(a4)
lw  a0,12(a4)
lw  a1,16(a4)
lw  a2,20(a4)
lw  a3,24(a4)
sw  a5,-68(s0)
sw  a7,-64(s0)
sw  a6,-60(s0)
sw  a0,-56(s0)
```

```
40      sw  a1,-52(s0)
41      sw  a2,-48(s0)
42      sw  a3,-44(s0)
43      lhu a4,28(a4)
44      sh  a4,-40(s0)
45      andi    a5,a5,0xff
46      beq a5,zero,.L2
47              li  s2,1
48      addi    a4,s0,-68
49      li  a3,32
50      .L4:
51      addi    a4,a4,1
52      sub a5,a5,a3
53      seqz    a5,a5
54      add s2,s2,a5
55      lbu a5,0(a4)
56      bne a5,zero,.L4
57              slli    a5,s2,2
58      addi    a5,a5,15
59      andi    a5,a5,-16
60      sub sp,sp,a5
61      mv  s1,sp
62      mv  a1,s1
63      addi    a0,s0,-68
64      call    inst_reader
65      ble s2,zero,.L8
66      .L7:
67      li  s3,0
68      lui s4,%hi(.LC1)
69      .L6:
70      lw  a1,0(s1)
71      addi    a0,s4,%lo(.LC1)
72      call    printf
73      addi    s3,s3,1
74      addi    s1,s1,4
75      blt s3,s2,.L6
76      .L8:
77      li  a0,0
78      addi    sp,s0,-80
```

```
79      lw  ra,76(sp)
80      lw  s0,72(sp)
81      lw  s1,68(sp)
82      lw  s2,64(sp)
83      lw  s3,60(sp)
84      lw  s4,56(sp)
85      addi    sp,sp,80
86      jr  ra
87      .L2:
88      addi    a1,s0,-36
89      addi    a0,s0,-68
90      call    inst_reader
91      li  s2,1
92      addi    s1,s0,-36
93      j   .L7
94      .size   main, .-main
95      .ident  "GCC: (SiFive GCC 10.1.0-2020.08.2) 10.1.0"
```

Текст main.i

Результат компилирования – это код представленный на языке ассемблера.

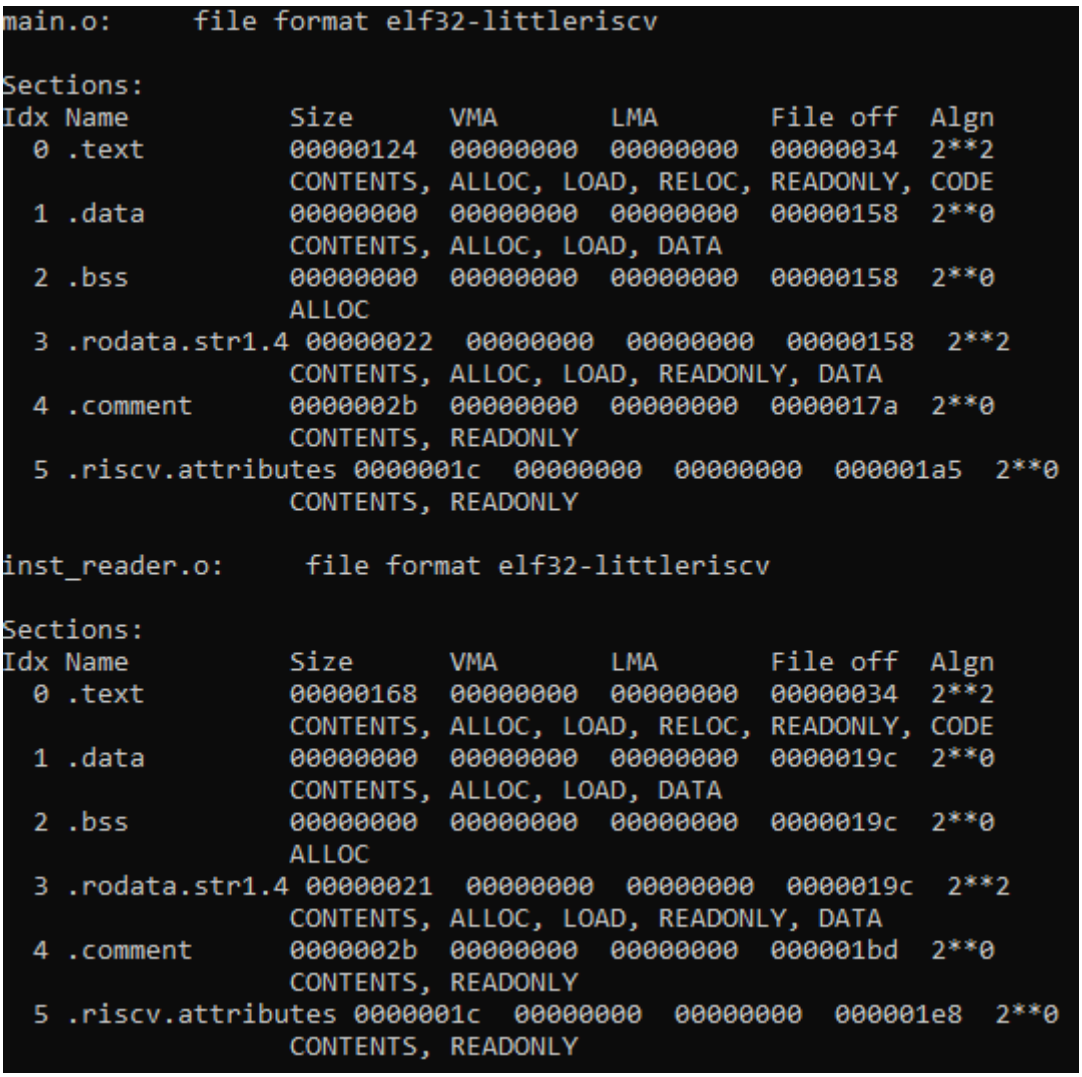
Ассемблирование:

- Запишем результат ассемблирования main.s и inst_reader.s в файлы main.o и inst_reader.o . Используем команды:

riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 -c main.s -o main.o
riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 -c inst_reader.s -o inst_reader.o

Секции:

Воспользуемся командой `riscv64-unknown-elf-objdump -h main.o inst_reader.o`, для вывода заголовков и информации о них.



```
main.o:     file format elf32-littleriscv

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000124  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000158  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000158  2**0
                  ALLOC
  3 .rodata.str1.4 00000022  00000000  00000000  00000158  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .comment      0000002b  00000000  00000000  0000017a  2**0
                  CONTENTS, READONLY
  5 .riscv.attributes 0000001c  00000000  00000000  000001a5  2**0
                  CONTENTS, READONLY

inst_reader.o:     file format elf32-littleriscv

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000168  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  0000019c  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  0000019c  2**0
                  ALLOC
  3 .rodata.str1.4 00000021  00000000  00000000  0000019c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .comment      0000002b  00000000  00000000  000001bd  2**0
                  CONTENTS, READONLY
  5 .riscv.attributes 0000001c  00000000  00000000  000001e8  2**0
                  CONTENTS, READONLY
```

Секции

.text – это секция кода, в которой хранятся коды инструкций.
.data – это секция инициализированных данных.
.rodata – секция инициализированных данных, предназначенных только для чтения.
.bss – это секция данных, инициализированных нулями.
.comment – секция данных содержащая информацию о версиях.
.riscv.attributes – секция с информацией о Risc-V.

Рассмотрим секции .text подробнее с использование команды riscv64-unknown-elf-objdump – d – M no-aliases – j .text inst_reader.o main.o.
Опция -d инициирует процесс дизассемблирования, а опция -M no-aliases выводит только инструкции системы команд, но не псевдоинструкций.

```
00000000 <inst_reader>:
   0:   fc010113                addi    sp,sp,-64
   4:   02812e23                sw      s0,60(sp)
   8:   000007b7                lui     a5,0x0
   c:   00078793                addi    a5,a5,0 # 0 <inst_reader>
  10:   0007ae83                lw      t4,0(a5)
  14:   0047ae03                lw      t3,4(a5)
  18:   0087a303                lw      t1,8(a5)
  1c:   00c7a883                lw      a7,12(a5)
  20:   0107a803                lw      a6,16(a5)
  24:   0147a603                lw      a2,20(a5)
  28:   0187a683                lw      a3,24(a5)
  2c:   01c7a703                lw      a4,28(a5)
  30:   01d12623                sw      t4,12(sp)
  34:   01c12823                sw      t3,16(sp)
  38:   00612a23                sw      t1,20(sp)
  3c:   01112c23                sw      a7,24(sp)
  40:   01012e23                sw      a6,28(sp)
  44:   02c12023                sw      a2,32(sp)
  48:   02d12223                sw      a3,36(sp)
  4c:   02e12423                sw      a4,40(sp)
  50:   0207c783                lbu     a5,32(a5)
  54:   02f10623                sb      a5,44(sp)
  58:   00054683                lbu     a3,0(a0)
  5c:   10068063                beq     a3,zero,15c <.L1>
  60:   00c14383                lbu     t2,12(sp)
  64:   00c10e13                addi    t3,sp,12
  68:   00100313                addi    t1,zero,1
  6c:   41c30333                sub     t1,t1,t3
  70:   00000e93                addi    t4,zero,0
  74:   00900893                addi    a7,zero,9
  78:   04c00293                addi    t0,zero,76
  7c:   00f00f93                addi    t6,zero,15
  80:   02000f13                addi    t5,zero,32
  84:   0400006f                jal     zero,c4 <.L3>

00000088 <.L14>:
  88:   000e8693                addi    a3,t4,0
  8c:   0b00006f                jal     zero,13c <.L5>

00000090 <.L7>:
  90:   00458593                addi    a1,a1,4
  94:   010fa833                slt     a6,t6,a6
  98:   01081813                slli    a6,a6,0x10
  9c:   ffc5a783                lw      a5,-4(a1)
  a0:   41078833                sub     a6,a5,a6
  a4:   ff05ae23                sw      a6,-4(a1)
  a8:   00164783                lbu     a5,1(a2)
  ac:   41e787b3                sub     a5,a5,t5
  b0:   0017b793                sltiu   a5,a5,1
  b4:   00f607b3                add     a5,a2,a5
  b8:   00178513                addi    a0,a5,1
  bc:   0017c683                lbu     a3,1(a5)
  c0:   08068e63                beq     a3,zero,15c <.L1>
```

```
000000c4 <.L3>:
  c4:   000e0793                addi    a5,t3,0
  c8:   000e8813                addi    a6,t4,0
  cc:   00d38a63                beq     t2,a3,e0 <.L13>

000000d0 <.L4>:
  d0:   00f30833                add     a6,t1,a5
  d4:   00178793                addi    a5,a5,1
  d8:   0007c703                lbu     a4,0(a5)
  dc:   fed71ae3                bne     a4,a3,d0 <.L4>

000000e0 <.L13>:
  e0:   41f85413                srai    s0,a6,0x1f
  e4:   01c45413                srli    s0,s0,0x1c
  e8:   008807b3                add     a5,a6,s0
  ec:   00f7f793                andi    a5,a5,15
  f0:   408787b3                sub     a5,a5,s0
  f4:   00c79413                slli    s0,a5,0xc
  f8:   0085a023                sw      s0,0(a1)
  fc:   00150613                addi    a2,a0,1
 100:   00154503                lbu     a0,1(a0)
 104:   fd050793                addi    a5,a0,-48
 108:   0ff7f793                andi    a5,a5,255
 10c:   f6f8eee3                bltu    a7,a5,88 <.L14>
 110:   000e8693                addi    a3,t4,0

00000114 <.L6>:
 114:   00160613                addi    a2,a2,1
 118:   00269793                slli    a5,a3,0x2
 11c:   00d787b3                add     a5,a5,a3
 120:   00179793                slli    a5,a5,0x1
 124:   00a787b3                add     a5,a5,a0
 128:   fd078693                addi    a3,a5,-48
 12c:   00064503                lbu     a0,0(a2)
 130:   fd050713                addi    a4,a0,-48
 134:   0ff77713                andi    a4,a4,255
 138:   fce8fee3                bgeu    a7,a4,114 <.L6>

0000013c <.L5>:
 13c:   00169793                slli    a5,a3,0x1
 140:   008787b3                add     a5,a5,s0
 144:   00f5a023                sw      a5,0(a1)
 148:   00064703                lbu     a4,0(a2)
 14c:   f45712e3                bne     a4,t0,90 <.L7>
 150:   00178793                addi    a5,a5,1
 154:   00f5a023                sw      a5,0(a1)
 158:   f39ff06f                jal     zero,90 <.L7>

0000015c <.L1>:
 15c:   03c12403                lw      s0,60(sp)
 160:   04010113                addi    sp,sp,64
 164:   00008067                jalr    zero,0(ra)
```

Секция .text для inst_reader.o

```
00000000 <main>:
   0:   fb010113                addi    sp,sp,-80
   4:   04112623                sw      ra,76(sp)
   8:   04812423                sw      s0,72(sp)
   c:   04912223                sw      s1,68(sp)
  10:   05212023                sw      s2,64(sp)
  14:   03312e23                sw      s3,60(sp)
  18:   03412c23                sw      s4,56(sp)
  1c:   05010413                addi    s0,sp,80
  20:   00000737                lui     a4,0x0
  24:   00070713                addi    a4,a4,0 # 0 <main>
  28:   00072783                lw      a5,0(a4)
  2c:   00472883                lw      a7,4(a4)
  30:   00872803                lw      a6,8(a4)
  34:   00c72503                lw      a0,12(a4)
  38:   01072583                lw      a1,16(a4)
  3c:   01472603                lw      a2,20(a4)
  40:   01872683                lw      a3,24(a4)
  44:   faf42e23                sw      a5,-68(s0)
  48:   fd142023                sw      a7,-64(s0)
  4c:   fd042223                sw      a6,-60(s0)
  50:   fca42423                sw      a0,-56(s0)
  54:   fcb42623                sw      a1,-52(s0)
  58:   fcc42823                sw      a2,-48(s0)
  5c:   fcd42a23                sw      a3,-44(s0)
  60:   01c75703                lhu     a4,28(a4)
  64:   fce41c23                sh      a4,-40(s0)
  68:   0ff7f793                andi    a5,a5,255
  6c:   08078e63                beq     a5,zero,108 <.L2>
  70:   00100913                addi    s2,zero,1
  74:   fbc40713                addi    a4,s0,-68
  78:   02000693                addi    a3,zero,32

0000007c <.L4>:
  7c:   00170713                addi    a4,a4,1
  80:   40d787b3                sub     a5,a5,a3
  84:   0017b793                sltiu   a5,a5,1
  88:   00f90933                add     s2,s2,a5
  8c:   00074783                lbu     a5,0(a4)
  90:   fe0796e3                bne     a5,zero,7c <.L4>
  94:   00291793                slli    a5,s2,0x2
  98:   00f78793                addi    a5,a5,15
  9c:   ff07f793                andi    a5,a5,-16
  a0:   40f10133                sub     sp,sp,a5
  a4:   00010493                addi    s1,sp,0
  a8:   00048593                addi    a1,s1,0
  ac:   fbc40513                addi    a0,s0,-68
  b0:   00000097                auipc   ra,0x0
  b4:   000080e7                jalr    ra,0(ra) # b0 <.L4+0x34>
  b8:   03205463                bge     zero,s2,e0 <.L8>

000000bc <.L7>:
  bc:   00000993                addi    s3,zero,0
  c0:   00000a37                lui     s4,0x0
```

```
000000c4 <.L6>:
  c4:   0004a583                lw      a1,0(s1)
  c8:   000a0513                addi    a0,s4,0 # 0 <main>
  cc:   00000097                auipc   ra,0x0
  d0:   000080e7                jalr    ra,0(ra) # cc <.L6+0x8>
  d4:   00198993                addi    s3,s3,1
  d8:   00448493                addi    s1,s1,4
  dc:   ff29c4e3                blt     s3,s2,c4 <.L6>

000000e0 <.L8>:
  e0:   00000513                addi    a0,zero,0
  e4:   fb040113                addi    sp,s0,-80
  e8:   04c12083                lw      ra,76(sp)
  ec:   04812403                lw      s0,72(sp)
  f0:   04412483                lw      s1,68(sp)
  f4:   04012903                lw      s2,64(sp)
  f8:   03c12983                lw      s3,60(sp)
  fc:   03812a03                lw      s4,56(sp)
 100:   05010113                addi    sp,sp,80
 104:   00008067                jalr    zero,0(ra)

00000108 <.L2>:
 108:   fdc40593                addi    a1,s0,-36
 10c:   fbc40513                addi    a0,s0,-68
 110:   00000097                auipc   ra,0x0
 114:   000080e7                jalr    ra,0(ra) # 110 <.L2+0x8>
 118:   00100913                addi    s2,zero,1
 11c:   fdc40493                addi    s1,s0,-36
 120:   f9dff06f                jal     zero,bc <.L7>
```

Секция .text для main.o

При сравнении .s и .o файлов, можно заметить что ассемблер транслирует псевдоинструкции call в пары инструкций auipc и jalr.

```
  b0:   00000097                auipc   ra,0x0
  b4:   000080e7                jalr    ra,0(ra) # b0 <.L4+0x34>
```

```
  cc:   00000097                auipc   ra,0x0
  d0:   000080e7                jalr    ra,0(ra) # cc <.L6+0x8>
```

Ассемблер не имел достаточной информации для определения целевого адреса перехода, поэтому сформировал не корректную инструкцию, которая приводит к зацикливанию. Это должно быть исправлено компоновщиком при помощи таблицы перемещений.

Таблицы символов:

С помощью команды `riscv64-unknown-elf-objdump -t main.o inst_reader.o` исследуем таблицы символов файлов main.o и inst_reader.o .

```
main.o:     file format elf32-littleriscv

SYMBOL TABLE:
00000000 l    df *ABS*  00000000 main.c
00000000 l    d  .text  00000000 .text
00000000 l    d  .data  00000000 .data
00000000 l    d  .bss   00000000 .bss
00000000 l    d  .rodata.str1.4 00000000 .rodata.str1.4
00000004 l       .rodata.str1.4 00000000 .LC0
00000000 l       .rodata.str1.4 00000000 .LC1
00000108 l       .text  00000000 .L2
0000007c l       .text  00000000 .L4
000000e0 l       .text  00000000 .L8
000000c4 l       .text  00000000 .L6
000000bc l       .text  00000000 .L7
00000000 l    d  .comment       00000000 .comment
00000000 l    d  .riscv.attributes      00000000 .riscv.attributes
00000000 g     F .text  00000124 main
00000000         *UND*  00000000 inst_reader
00000000         *UND*  00000000 printf



inst_reader.o:     file format elf32-littleriscv

SYMBOL TABLE:
00000000 l    df *ABS*  00000000 inst_reader.c
00000000 l    d  .text  00000000 .text
00000000 l    d  .data  00000000 .data
00000000 l    d  .bss   00000000 .bss
00000000 l    d  .rodata.str1.4 00000000 .rodata.str1.4
00000000 l       .rodata.str1.4 00000000 .LC0
0000015c l       .text  00000000 .L1
000000c4 l       .text  00000000 .L3
0000013c l       .text  00000000 .L5
000000e0 l       .text  00000000 .L13
000000d0 l       .text  00000000 .L4
00000088 l       .text  00000000 .L14
00000114 l       .text  00000000 .L6
00000090 l       .text  00000000 .L7
00000000 l    d  .comment       00000000 .comment
00000000 l    d  .riscv.attributes      00000000 .riscv.attributes
00000000 g     F .text  00000168 inst_reader
```

Таблицы символов

В таблицах имеются типы *UND* (undefined). Это означает, что символы inst_reader и printf использовались в коде ассемблера, из которого были получены данные объектные файлы, но не были определены. Ассемблер сделал вывод о том, что символы были определены где-то ещё и отобразил это.

Таблица перемещений:

Таблица перемещений хранит информацию для компоновщика. С помощью неё компоновщик при необходимости редактирует код полученный от ассемблера. Для отображения этих таблиц используем команду `riscv64-unknown-elf-objdump -r inst_reader.o main.o` .

```
inst_reader.o:     file format elf32-littleriscv

RELOCATION RECORDS FOR [.text]:
OFFSET   TYPE              VALUE
00000008 R_RISCV_HI20      .LC0
00000008 R_RISCV_RELAX     *ABS*
0000000c R_RISCV_LO12_I    .LC0
0000000c R_RISCV_RELAX     *ABS*
0000005c R_RISCV_BRANCH    .L1
00000084 R_RISCV_JAL       .L3
0000008c R_RISCV_JAL       .L5
000000c0 R_RISCV_BRANCH    .L1
000000cc R_RISCV_BRANCH    .L13
000000dc R_RISCV_BRANCH    .L4
0000010c R_RISCV_BRANCH    .L14
00000138 R_RISCV_BRANCH    .L6
0000014c R_RISCV_BRANCH    .L7
00000158 R_RISCV_JAL       .L7



main.o:     file format elf32-littleriscv

RELOCATION RECORDS FOR [.text]:
OFFSET   TYPE              VALUE
00000020 R_RISCV_HI20      .LC0
00000020 R_RISCV_RELAX     *ABS*
00000024 R_RISCV_LO12_I    .LC0
00000024 R_RISCV_RELAX     *ABS*
000000b0 R_RISCV_CALL      inst_reader
000000b0 R_RISCV_RELAX     *ABS*
000000c0 R_RISCV_HI20      .LC1
000000c0 R_RISCV_RELAX     *ABS*
000000c8 R_RISCV_LO12_I    .LC1
000000c8 R_RISCV_RELAX     *ABS*
000000cc R_RISCV_CALL      printf
000000cc R_RISCV_RELAX     *ABS*
00000110 R_RISCV_CALL      inst_reader
00000110 R_RISCV_RELAX     *ABS*
0000006c R_RISCV_BRANCH    .L2
00000090 R_RISCV_BRANCH    .L4
000000b8 R_RISCV_BRANCH    .L8
000000dc R_RISCV_BRANCH    .L6
00000120 R_RISCV_JAL       .L7
```

Таблицы перемещений

Для того, чтобы компоновщику внести изменения в код, необходимо знать что и как исправлять. В таблице хранится информация об адресе и способе исправления. Ранее мы заметили, что по адресу 000000b0 необходимо исправить пару инструкций (R_RISCV_CALL) таким образом, чтобы результат вызывал подпрограмму inst_reader. R_RISCV_RELAX сообщает компоновщику, что пара инструкций может быть оптимизирована.

Полученные объектные файлы main.o и inst_reader.o содержат секции с кодами инструкций и таблицы символов и перемещений. Эти файлы не являются текстовыми, поэтому для их просмотра мы используем -objdump, для отображения данных в текстовом виде.

Компоновка:

- Запишем результат компоновки main.o и inst_reader.o в файлы main. Используем команду:

riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 main.o inst_reader.o -o main

Теперь рассмотрим .text: **riscv64-unknown-elf-objdump –j .text –d –M no-aliases main >main.ds**

Файл main.ds содержит более 22000 строк, поэтому рассмотрим только небольшую часть.

```
00010140 <main>:
10140:  fb010113            addi    sp,sp,-80
10144:  04112623            sw  ra,76(sp)
10148:  04812423            sw  s0,72(sp)
1014c:  04912223            sw  s1,68(sp)
10150:  05212023            sw  s2,64(sp)
10154:  03312e23            sw  s3,60(sp)
10158:  03412c23            sw  s4,56(sp)
1015c:  05010413            addi    s0,sp,80
10160:  00025737            lui a4,0x25
10164:  49470713            addi    a4,a4,1172 # 25494 <__clzsi2+0x50>
10168:  00072783            lw  a5,0(a4)
1016c:  00472883            lw  a7,4(a4)
10170:  00872803            lw  a6,8(a4)
10174:  00c72503            lw  a0,12(a4)
10178:  01072583            lw  a1,16(a4)
1017c:  01472603            lw  a2,20(a4)
10180:  01872683            lw  a3,24(a4)
10184:  faf42e23            sw  a5,-68(s0)
10188:  fd142023            sw  a7,-64(s0)
1018c:  fd042223            sw  a6,-60(s0)
10190:  fca42423            sw  a0,-56(s0)
10194:  fcb42623            sw  a1,-52(s0)
10198:  fcc42823            sw  a2,-48(s0)
1019c:  fcd42a23            sw  a3,-44(s0)
```

```
1019c:  fcd42a23            sw  a3,-44(s0)
101a0:  01c75703            lhu a4,28(a4)
101a4:  fce41c23            sh  a4,-40(s0)
101a8:  0ff7f793            andi    a5,a5,255
101ac:  08078a63            beq a5,zero,10240 <main+0x100>
101b0:  00100913            addi    s2,zero,1
101b4:  fbc40713            addi    a4,s0,-68
101b8:  02000693            addi    a3,zero,32
101bc:  00170713            addi    a4,a4,1
101c0:  40d787b3            sub a5,a5,a3
101c4:  0017b793            sltiu   a5,a5,1
101c8:  00f90933            add s2,s2,a5
101cc:  00074783            lbu a5,0(a4)
101d0:  fe0796e3            bne a5,zero,101bc <main+0x7c>
101d4:  00291793            slli    a5,s2,0x2
101d8:  00f78793            addi    a5,a5,15
101dc:  ff07f793            andi    a5,a5,-16
101e0:  40f10133            sub sp,sp,a5
101e4:  00010493            addi    s1,sp,0
101e8:  00048593            addi    a1,s1,0
101ec:  fbc40513            addi    a0,s0,-68
101f0:  068000ef            jal ra,10258 <inst_reader>
101f4:  03205263            bge zero,s2,10218 <main+0xd8>
101f8:  00000993            addi    s3,zero,0
101fc:  00025a37            lui s4,0x25
10200:  0004a583            lw  a1,0(s1)
10204:  490a0513            addi    a0,s4,1168 # 25490 <__clzsi2+0x4c>
10208:  398000ef            jal ra,105a0 <printf>
1020c:  00198993            addi    s3,s3,1
10210:  00448493            addi    s1,s1,4
10214:  ff29c6e3            blt s3,s2,10200 <main+0xc0>
10218:  00000513            addi    a0,zero,0
1021c:  fb040113            addi    sp,s0,-80
10220:  04c12083            lw  ra,76(sp)
10224:  04812403            lw  s0,72(sp)
10228:  04412483            lw  s1,68(sp)
1022c:  04012903            lw  s2,64(sp)
10230:  03c12983            lw  s3,60(sp)
10234:  03812a03            lw  s4,56(sp)
```

```
63    10234:  03812a03            lw  s4,56(sp)
64    10238:  05010113            addi    sp,sp,80
65    1023c:  00008067            jalr    zero,0(ra)
66    10240:  fdc40593            addi    a1,s0,-36
67    10244:  fbc40513            addi    a0,s0,-68
68    10248:  010000ef            jal ra,10258 <inst_reader>
69    1024c:  00100913            addi    s2,zero,1
70    10250:  fdc40493            addi    s1,s0,-36
71    10254:  fa5ff06f            jal zero,101f8 <main+0xb8>
```

Фрагмент main.ds

Инструкции были должным образом откорректированы. В данном примере точка вызова подпрограммы и сама программа находятся близко, поэтому auipc и jalr оптимизируются до jal (до этого была пометка R_RISCV_RELAX).

Статическая библиотека:

Статическая библиотека – это архив, набор объектных файлов. Поместим inst_reader.o в libds.a используя команду:

riscv64-unknown-elf-ar - rsc libds.a inst_reader.o

-r – перезаписывать файлы с одинаковыми названиями.

-s – записать список всех символов, объявленных в объектных файлах, в архив.

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

Проверка:

Проверим содержимое архива:

riscv64-unknown-elf-ar -t libds.a

```
E:\lab4\step>riscv64-unknown-elf-ar -t libds.a
inst_reader.o
```

В результате мы видим, что действительно записали только inst_reader.o .
Теперь соберем программу, используя созданную статическую библиотеку.

riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -O1 --save-temps main.c libds.a -o main1

При изучении таблицы символов мы убеждаемся что функция из нашей библиотеки попала туда.

```
00013b10 g     F .text  00000130 _malloc_trim_r
00010258 g     F .text  00000168 inst_reader
00018fa4 g     F .text  0000017c strcmp
```

Выведем таблицу символов библиотеки libds.a:

riscv64-unknown-elf-nm libds.a

```
inst_reader.o:
0000015c t .L1
000000e0 t .L13
00000088 t .L14
000000c4 t .L3
000000d0 t .L4
0000013c t .L5
00000114 t .L6
00000090 t .L7
00000000 r .LC0
00000000 T inst_reader
```

Таблица символов библиотеки

Make-файлы:

Makefile – это набор инструкций для программы make, которая помогает собирать программу из файлов в один вызов make.

Реализация:

В makefile создается объектный файл inst_reader.o на остнове файла inst_reader.c, а затем упаковывается в библиотеку makeLib.a:

```
CC=riscv64-unknown-elf-gcc
AR=riscv64-unknown-elf-ar
CFLAGS=-march=rv32i -mabi=ilp32 -O1

all: makeLib

staticLib: inst_reader.o
$(AR) -rsc makeLib.a inst_reader.o

inst_reader.o: inst_reader.c
$(CC) $(CFLAGS) -c inst_reader.c -o inst_reader.o

clean:
rm -f *.o *.a
```

Затем в makefile, для сборки программы на основании библиотеки, собирается main, используя main.c и makeLib.a:

```
TARGET=main
CC=riscv64-unknown-elf-gcc
CFLAGS=-march=rv64iac -mabi=lp64 -O1

main:
$(CC) $(CFLAGS) main.c makeLib.a -o $(TARGET)

clean:
rm -f *.o *.a $(TARGET)
```

Вывод:

В ходе выполнения лабораторной работы была разработана функция на языке C, реализующая загрузку коротких чисел, кодирующихся привычным для EDSAC образом.

Была проведена сборка программы «по шагам». Проанализирован выход препроцессора и компилятора. Также проанализированы содержимое секций, таблицы символов, таблицы перемещений содержащиеся в объектных и исполняемом файле.

Разработанная функция была выделена в статическую библиотеку и были разработаны make-файлы для сборки библиотеки и её тестовой программы.