Федеральное государственное автономное
образовательное учреждение высшего образования
«Научно-образовательная корпорация ИТМО»

Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.04 Программная инженерия

Отчёт по лабораторной работе №1

По дисциплине «Системы ввода-вывода» (семестр 6)

Студент:
Дениченко Александр P3312
Разинкин Александр P3307
Практик:
Табунщик Сергей Михайлович

Санкт-Петербург
2025 г.

Задачи

Лабораторная работа:

1. Реализовать функцию putchar вывода данных в консоль
2. Реализовать функцию getchar для получения данных из консоли
3. На базе реализованных функций putchar и getchar написать программу, позволяющую вызывать определенным вариантом функции OpenSBI посредством взаимодействия пользователя через меню
4. Запустить программу и выполнить вызов пунктов меню, получив результаты их работы
5. Оформить отчет по работе в электронном формате

Вариант - 1:

1. Get SBI specification version
2. Get number of counters
3. Get details of a counter (должно быть возможно задавать номер счетчика)
4. System Shutdown

Листинг 1: run.sh

```
set -xue

QEMU=qemu-system-riscv32

CC=/opt/homebrew/opt/llvm/bin/clang
CFLAGS="-std=c11 -O2 -g3 -Wall -Wextra --target=riscv32 -ffreestanding -nostdlib"

$CC $CFLAGS -Wl,-Tkernel.ld -Wl,-Map=kernel.map -o kernel.elf kernel.c

$QEMU -machine virt -bios default -nographic -serial mon:stdio --no-reboot -kernel
    kernel.elf
```

Листинг 2: kernel.h

```
#pragma once

struct sbiret {
    long error;
    long value;
};
```

Листинг 3: kernel.ld

```
ENTRY(boot)

SECTIONS {
    . = 0x80200000;

    .text :{
        KEEP(*(.text.boot));
        *(.text .text.*);
    }

    .rodata : ALIGN(4) {
        *(.rodata .rodata.*);
    }

    .data : ALIGN(4) {
        *(.data .data.*);
    }
```

```
18
19          .bss : ALIGN(4) {
20              __bss = .;
21              *(.bss .bss.* .sbss .sbss.*);
22              __bss_end = .;
23          }
24
25          . = ALIGN(4);
26          . += 128 * 1024; /* 128KB */
27          __stack_top = .;
28      }
```

Листинг 4: kernel.ld

```
1   #include "kernel.h"
2
3   extern char __bss[], __bss_end[], __stack_top[];
4
5   struct sbiret sbi_call(long arg0, long arg1, long arg2, long arg3, long arg4,
6       long arg5, long fid, long eid) {
7       register long a0 __asm__("a0") = arg0;
8       register long a1 __asm__("a1") = arg1;
9       register long a2 __asm__("a2") = arg2;
10      register long a3 __asm__("a3") = arg3;
11      register long a4 __asm__("a4") = arg4;
12      register long a5 __asm__("a5") = arg5;
13      register long a6 __asm__("a6") = fid;
14      register long a7 __asm__("a7") = eid;
15
16      __asm__ __volatile__("ecall"
17          : "=r"(a0), "=r"(a1)
18          : "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5),
19            "r"(a6), "r"(a7)
20          : "memory");
21      return (struct sbiret){.error = a0, .value = a1};
22  }
23
24  long sbi_call_long(long arg0, long arg1, long arg2, long arg3, long arg4,
25      long arg5, long fid, long eid) {
26      register long a0 __asm__("a0") = arg0;
27      register long a1 __asm__("a1") = arg1;
28      register long a2 __asm__("a2") = arg2;
29      register long a3 __asm__("a3") = arg3;
30      register long a4 __asm__("a4") = arg4;
31      register long a5 __asm__("a5") = arg5;
32      register long a6 __asm__("a6") = fid;
33      register long a7 __asm__("a7") = eid;
34
35      __asm__ __volatile__("ecall"
36          : "=r"(a0), "=r"(a1)
37          : "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5),
38            "r"(a6), "r"(a7)
39          : "memory");
40      return a0;
41  }
42
43  void putchar(char ch) {
```

```
44          sbi_call(ch, 0, 0, 0, 0, 0, 0, 1 /* Console Putchar */);
45      }
46
47      char getchar(void) {
48          for (;;) {
49              long ch = sbi_call_long(0, 0, 0, 0, 0, 0, 0, 2);
50              if (ch != -1 && ch != 0) {
51                  return ch;
52              }
53          }
54      }
55
56      long read_number(void) {
57          long number = 0;
58          int sign = 1;
59          char ch;
60
61          while ((ch = getchar()) != '\n') {
62              if (ch < '0' || ch > '9') {
63                  putchar('E');
64                  putchar('\n');
65                  return 0;
66              }
67
68              number = number * 10 + (ch - '0');
69          }
70
71          return number * sign;
72      }
73
74      void show_sbi_version(void) {
75          struct sbiret result = sbi_call(0, 0, 0, 0, 0, 0, 0, 0x10);
76
77          long number = result.value & 0xFFFFFF;
78          char high = (number >> 16) & 0xFF;
79          char mid = (number >> 8) & 0xFF;
80          char low = number & 0xFF;
81
82          putchar('0' + high + mid + low);
83
84          putchar('.');
85
86          putchar('0' + (result.value >> 24));
87      }
88
89      void show_num_counter(void) {
90          const char *s = "\n\nCount: ";
91          for (int i = 0; s[i] != '\0'; i++) {
92              putchar(s[i]);
93          }
94
95          struct sbiret result = sbi_call(0, 0, 0, 0, 0, 0, 0, 0x504D55);
96
97          long number = result.value;
98
99          if (number == 0) {
```

```
100             putchar('0');
101             return;
102         }
103
104         char digits[10];
105         int count = 0;
106
107         while (number > 0) {
108             digits[count] = '0' + (number % 10);
109             number /= 10;
110             count++;
111         }
112
113         for (int i = count - 1; i >= 0; i--) {
114             putchar(digits[i]);
115         }
116
117     }
118
119     void print_string(const char *s) {
120         for (int i = 0; s[i] != '\0'; i++) {
121         putchar(s[i]);
122         }
123     }
124
125     void print_number(long num) {
126         char buffer[32];
127         int i = 0;
128
129         if (num == 0) {
130         putchar('0');
131         return;
132         }
133
134         if (num < 0) {
135         putchar('-');
136         num = -num;
137         }
138
139         while (num > 0) {
140         buffer[i++] = '0' + (num % 10);
141         num /= 10;
142         }
143
144         while (--i >= 0) {
145         putchar(buffer[i]);
146         }
147     }
148
149     void show_counter(void) {
150
151         int counter_num = 0;
152         char digit = getchar();
153         putchar(digit);
154         if (digit >= '0' && digit <= '9') {
155         counter_num = digit - '0';
```

```
156         char next = getchar();
157         if (next >= '0' && next <= '9') {
158             putchar(next);
159             counter_num = counter_num * 10 + (next - '0');
160         }
161         }
162         print_string("\n");
163 
164         struct sbiret info = sbi_call(counter_num, 0, 0, 0, 0, 0, 1, 0x504D55);
165 
166         unsigned long counter_info = info.value;
167         unsigned long csr = counter_info & 0xFFF;           // bits [11:0]
168         unsigned long width = (counter_info >> 12) & 0x3F;  // bits [17:12]
169         unsigned long type = counter_info >> (sizeof(long) * 8 - 1); // highest bit
170 
171         print_string("Counter details:\n");
172         print_string("Type: ");
173         print_string(type ? "Firmware" : "Hardware");
174         print_string("\n");
175 
176         if (type == 0) {  // Only show CSR and width for hardware counters
177         print_string("CSR number: 0x");
178         print_number(csr);
179         print_string("\nWidth: ");
180         print_number(width + 1);  // Width is stored as (actual width - 1)
181         print_string(" bits\n");
182         }
183     }
184 
185     void system_shutdown(void) {
186         const char *s = "\n\nTurn off!\n";
187         for (int i = 0; s[i] != '\0'; i++) {
188             putchar(s[i]);
189         }
190 
191         sbi_call(0, 0, 0, 0, 0, 0, 0, 8);
192     }
193 
194     void kernel_main(void) {
195         for (;;) {
196             switch (getchar()) {
197                 case '1':
198                     show_sbi_version();
199                     putchar('\n');
200                     break;
201                 case '2':
202                     show_num_counter();
203                     putchar('\n');
204                     break;
205                 case '3':
206                     show_counter();
207                     break;
208                 case '4':
209                     system_shutdown();
210                     break;
211                 default:
```

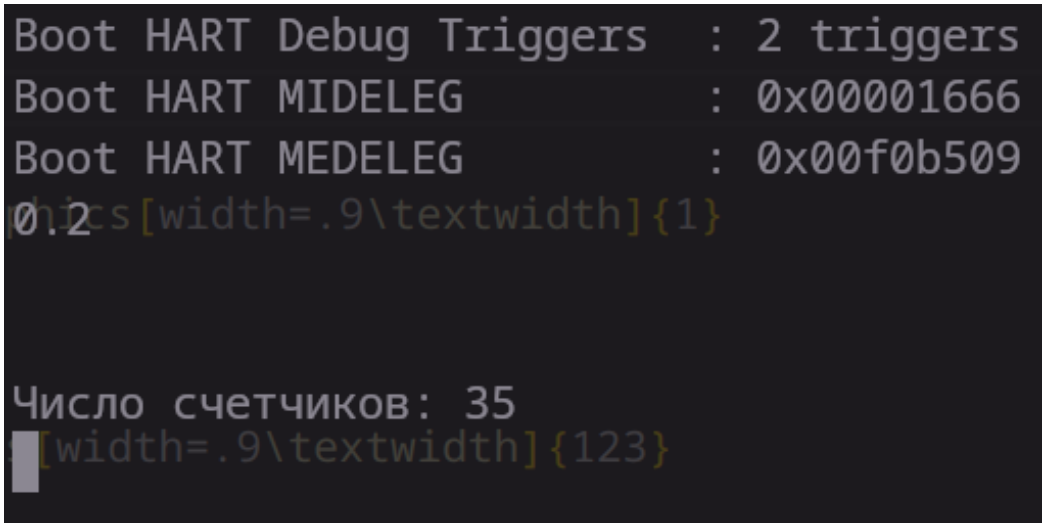
```
                    const char *s = "\n\nUnknown symbol";
                    for (int i = 0; s[i] != '\0'; i++) {
                        putchar(s[i]);
                    }
                    putchar('\n');
                }
            }
        }

        __attribute__((section(".text.boot")))
        __attribute__((naked))
        void boot(void) {
            __asm__ __volatile__(
                "mv sp, %[stack_top]\n"
                "j kernel_main\n"
                :
                : [stack_top] "r" (__stack_top)
            );
        }
```

1 Тестирование системы

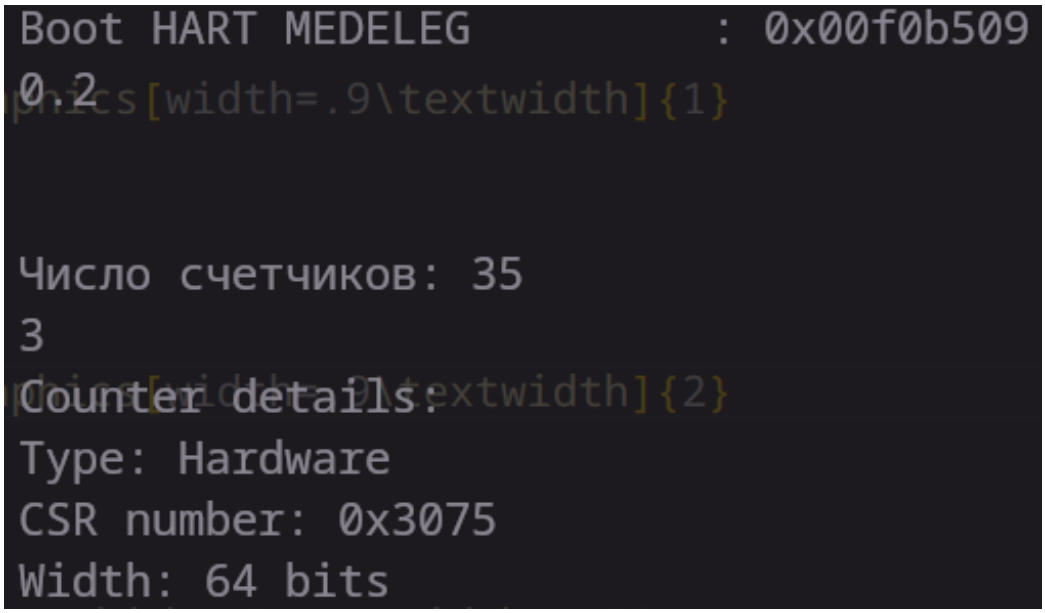
Ввод -> 1

```
Boot HART ID              : 0
Boot HART Domain          : root
Boot HART Priv Version    : v1.12
Boot HART Base ISA        : rv32imafdch
Boot HART ISA Extensions  : sstc,zicntr,zihpm,zicboz,zicbom,sdtrig,svadu
Boot HART PMP Count       : 16
Boot HART PMP Granularity : 2 bits
Boot HART PMP Address Bits: 32
Boot HART MHPM Info       : 16 (0x0007fff8)
Boot HART Debug Triggers  : 2 triggers
Boot HART MIDELEG         : 0x00001666
Boot HART MEDELEG         : 0x00f0b509
0.2
```

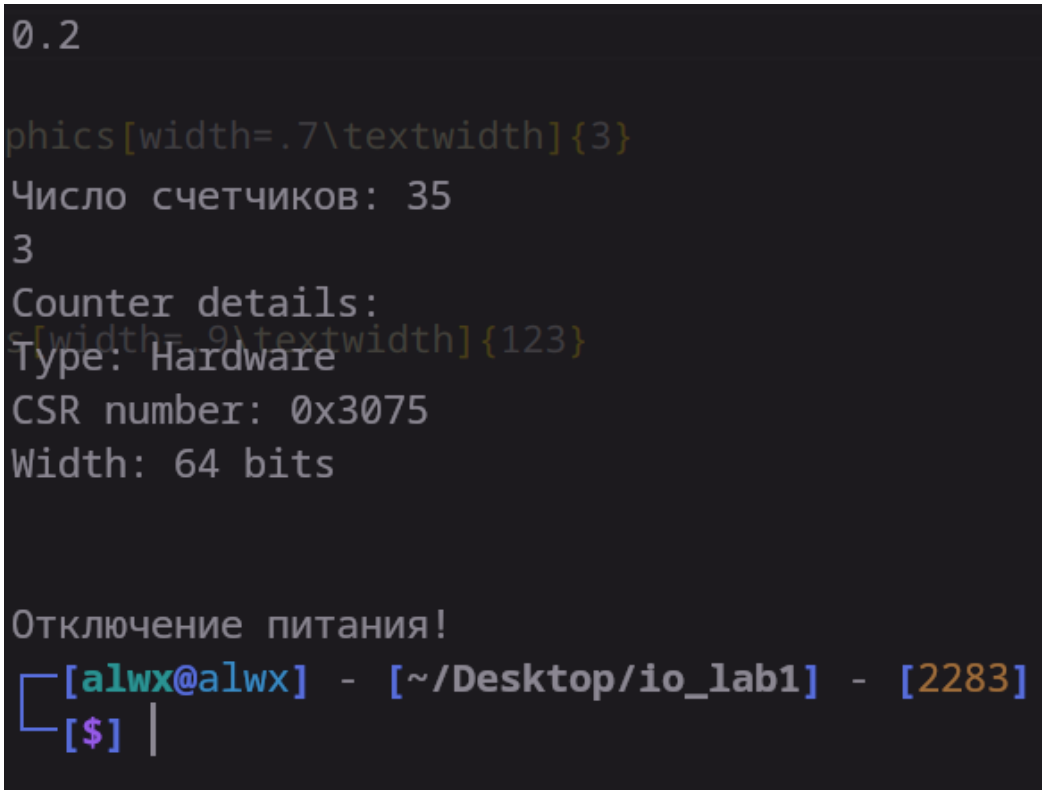
Ввод -> 2



Ввод -> 3 3



Ввод -> 4



(в отчёте вставлен код со всем выводом на en локали, функционал не менялся)