Федеральное государственное автономное образовательное
учреждение высшего образования
Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина *«Системы ввода-вывода»*

Отчет по лабораторной работе №1

«Принципы организации ввода/вывода без операционной системы»

Вариант: 1

Студент:
Барсуков Максим Андреевич,
поток 1.3, группа P3315

Преподаватель:
Табунщик Сергей Михайлович

г. Санкт-Петербург, 2025 г.

Оглавление

Введение

Цели работы

Познакомиться с принципами организации ввода/вывода без операционной системы на примере компьютерной системы на базе процессора с архитектурой RISC-V и интерфейсом OpenSBI с использованием эмулятора QEMU.

Задачи работы

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.

№ варианта	Пункты меню
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

Выполнение

Ссылка на репозиторий: GitHub.

Описание функций

`putchar`

```
#define SBI_ECALL_0_1_PUTCHAR 0x01

void
putchar(char ch)
{
  sbi_call(ch, 0, 0, 0, 0, 0, 0, SBI_ECALL_0_1_PUTCHAR);
}
```

Используем легаси расширение, которое выводит символ из аргумента. Является блокирующей функцией, т.е. до вывода текущего символа не будет выведен следующий.

`getchar`

```
#define SBI_ECALL_0_1_GETCHAR 0x02
#define SBI_ERR_FAILED -1

int
getchar(void)
{
  struct sbiret ret;

  do
  {
    ret = sbi_call(0, 0, 0, 0, 0, 0, 0, SBI_ECALL_0_1_GETCHAR);
  } while (ret.error == SBI_ERR_FAILED);

  return (int) ret.error;
}
```

Используем легаси расширение, который принимает символ, введённый пользователем. Функция будет ждать до тех пор (путём возврата -1), пока пользователь не введет символ через клавиатуру.

readline

```
void
readline(char *buf, int max_len)
{
  int i = 0;
  char c;

  while (i < max_len - 1) {
    c = getchar();

    if (c == 8 || c == 127) {
      if (i > 0) {
        i--;
        buf[i] = '\0';

        putchar(8);     // Перемещаем курсор назад
        putchar(' ');   // Стираем символ пробелом
        putchar(8);     // Снова перемещаем курсор назад
      }
      continue;
    }

    if (c == '\r' || c == '\n') {
      putchar('\n');
      break;
    }

    if (c >= 32 && c < 127) {
      putchar(c);
      buf[i++] = c;
    }
  }

  buf[i] = '\0';
}
```

Функция реализует ввод строки с консоли с поддержкой удаления символов (backspace).

puts

```
void
puts(const char *s)
{
  while (*s)
  {
    putchar(*s++);
  }
}
```

Простая утилитарная функция для посимвольного вывода строки с помощью `putchar`.

printf

Для реализации `printf` я написал несколько других простых функций: `strlen`, `reverse`, `atoi`, `itoa` (см. [common.c]).

```
void
printf(const char *fmt, ...)
{
  va_list vargs;
  va_start(vargs, fmt);

  while (*fmt)
  {
    if (*fmt == '%')
    {
      fmt++;
      int width = 0;
      int zero_pad = 0;
      int left_align = 0;

      while (1)
      {
        if (*fmt == '0') zero_pad = 1;
        else if (*fmt == '-') left_align = 1;
        else break;
        fmt++;
      }

      while (*fmt >= '0' && *fmt <= '9')
      {
          width = width * 10 + (*fmt++ - '0');
      }

      switch (*fmt)
      {
        case 'c':
        {
          char c = (char) va_arg(vargs, int);
          putchar(c);
          break;
        }

        case 's':
        {
          const char *s = va_arg(vargs, const char *);
          if (!s) s = "(null)";
          int len = strlen(s);

          if (width > len && !left_align)
          {
              for (int i = len; i < width; i++)
                  putchar(zero_pad ? '0' : ' ');
          }

          puts(s);

          if (width > len && left_align)
          {
              for (int i = len; i < width; i++)
                  putchar(' ');
          }
```

```
          break;
        }

        case 'd':
        case 'i':
        {
          int num = va_arg(vargs, int);
          char buffer[32];
          itoa(num, buffer);

          int len = strlen(buffer);

          if (width > len && !left_align)
          {
              for (int i = len; i < width; i++)
                  putchar(zero_pad ? '0' : ' ');
          }

          puts(buffer);

          if (width > len && left_align)
          {
              for (int i = len; i < width; i++)
                  putchar(' ');
          }
          break;
        }

        case 'u':
        {
          unsigned num = va_arg(vargs, unsigned);
          char buffer[32];
          itoa((int)num, buffer);

          int len = strlen(buffer);

          if (width > len && !left_align)
          {
              for (int i = len; i < width; i++)
                  putchar(zero_pad ? '0' : ' ');
          }

          puts(buffer);

          if (width > len && left_align)
          {
              for (int i = len; i < width; i++)
                  putchar(' ');
          }
          break;
        }

        case 'x':
        case 'X':
        {
          unsigned num = va_arg(vargs, unsigned);
          char buffer[32];
          char *p = buffer;
          const char *digits = (*fmt == 'X') ? "0123456789ABCDEF" :
"0123456789abcdef";

          if (num == 0)
```

```
            {
                *p++ = '0';
            } else {
                while (num > 0)
                {
                    *p++ = digits[num & 0xF];
                    num >>= 4;
                }
            }

            *p = '\0';
            reverse(buffer);

            int len = strlen(buffer);
            if (width > len && !left_align)
            {
                for (int i = len; i < width; i++)
                    putchar(zero_pad ? '0' : ' ');
            }

            puts(buffer);

            if (width > len && left_align)
            {
                for (int i = len; i < width; i++)
                    putchar(' ');
            }
            break;
        }

        case '\0':
        {
          putchar('%');
          goto end;
        }

        case '%':
        {
          putchar('%');
          break;
        }

        default:
        {
            putchar('%');
            putchar(*fmt);
            break;
        }

      }
    }

    else
    {
      putchar(*fmt);
    }

    fmt++;
  }

end:
  va_end(vargs);
}
```

Функция реализует форматированный вывод с поддержкой спецификаторов:

- `%c` – символ
- `%s` – строка
- `%d`, `%i` – целое число со знаком
- `%u` – целое число без знака
- `%x`, `%X` – шестнадцатеричное число
- Поддержка ширины вывода и выравнивания

Описание интерфейса вызова функций OpenSBI, заданных вариантом задания

1. Get SBI specification version

```
#define SBI_EXT_BASE           0x10
#define SBI_EXT_VER            0x00

void
get_sbi_version()
{
  struct sbiret ret = sbi_call(0, 0, 0, 0, 0, 0, SBI_EXT_VER, SBI_EXT_BASE);
  long major = ret.value >> 24;
  long minor = ret.value & 0xFFFFFF;
  printf("\nВерсия SBI: %d.", major);
  if(minor < 10)
  {
    putchar('0');
  }
  if(minor < 100)
  {
    putchar('0');
  }
  printf("%d\n", minor);
}
```

Получает версию спецификации. Minor-версия закодирована в последних 24 битах численного значения, возвращаемого вызовом, а Major — в предстоящих 7 битах. Для этого в функции есть дополнительные битовые операции.

Расширение: 0x10 (Base Extension). Функция: 0x0.

4.1. Function: Get SBI specification version (FID #0)

```
struct sbiret sbi_get_spec_version(void);
```

Returns the current SBI specification version. This function must always succeed. The minor number of the SBI specification is encoded in the low 24 bits, with the major number encoded in the next 7 bits. Bit 31 must be 0 and is reserved for future expansion. When XLEN is greater than 32, bits 32 and above are also reserved and must be 0.

2. Get number of counters

```
#define SBI_EXT_PMU             0x504D55
#define SBI_EXT_CTR_NUM         0x00

void
get_num_counters()
{
  struct sbiret ret = sbi_call(0, 0, 0, 0, 0, 0, SBI_EXT_CTR_NUM,
SBI_EXT_PMU);
  printf("\nЧисло счётчиков: %d\n", ret.value);
}
```

Получает число счётчиков в ОС. Они доступны в read-only режиме.

Расширение: 0x544D (Timer Extension). Функция: 0x0.

11.6. Function: Get number of counters (FID #0)

```
struct sbiret sbi_pmu_num_counters()
```

Returns the number of counters (both hardware and firmware) in `sbiret.value` and always returns `SBI_SUCCESS` in sbiret.error.

3. Get details of a counter

```
#define SBI_EXT_PMU             0x504D55
#define SBI_EXT_CTR_DTLS        0x01

void
get_counter_details()
{
  printf("\nВведите номер счётчика: ");
  char input[32];
  readline(input, sizeof(input));
  long counter_num = atoi(input);

  struct sbiret ret = sbi_call(counter_num, 0, 0, 0, 0, 0, SBI_EXT_CTR_DTLS,
SBI_EXT_PMU);
  printf("\nСчётчик: %d\n", counter_num);
  printf("Детали: \n");
  if(ret.error)
  {
    printf("Ошибка: %d\n", ret.error);
    return;
  }
  unsigned long counter_info = ret.value;
  int type = (counter_info >> (__riscv_xlen - 1)) & 0x1;
  int width = (counter_info >> 12) & 0x3F;
  int csr = counter_info & 0xFFF;

  printf("%4s* Тип: %s", "", type ? "Прошивка\n" : "Аппаратура\n");

  if(!type)
  {
    printf("%4s* CSR: %d\n", "", csr);
    printf("%4s* Ширина: %d бит\n", "", width + 1);
  }
  else
  {
    printf("%4s* CSR и ширина не применимы для прошивки.\n", "");
  }
}
```

Получает специфичные детали одного счётчика. Пользователь вводит номер счётчика, который передается в вызов интерфейса. В итоге выводится такая информация, как CSR (Control and Status Register), ширина счётчика и тип (аппаратура или прошивка).

Расширение: 0x544D (Timer Extension). Функция: 0x1.

11.7. Function: Get details of a counter (FID #1)

```
struct sbiret sbi_pmu_counter_get_info(unsigned long counter_idx)
```

Get details about the specified counter such as underlying CSR number, width of the counter, type of counter hardware/firmware, etc.

The `counter_info` returned by this SBI call is encoded as follows:

```
counter_info[11:0] = CSR (12bit CSR number)
counter_info[17:12] = Width (One less than number of bits in CSR)
counter_info[XLEN-2:18] = Reserved for future use
counter_info[XLEN-1] = Type (0 = hardware and 1 = firmware)
```

If `counter_info.type == 1` then `counter_info.csr` and `counter_info.width` should be ignored.

Returns the `counter_info` described above in `sbiret.value`.

The possible error codes returned in `sbiret.error` are shown in the Table 36 below.

Table 36. PMU Counter Get Info Errors

Error code	Description
SBI_SUCCESS	`counter_info` read successfully.
SBI_ERR_INVALID_PARAM	`counter_idx` points to an invalid counter.

4. System Shutdown

```c
#define SBI_EXT_SRST            0x53525354
#define SBI_EXT_SHUTDOWN        0x00

void
system_shutdown()
{
  printf("Завершение работы системы...\n");
  printf("\nСпасибо за использование программы!\n");
  printf("     |\\      _,,,---,,_\nZZZzz /,`.-'`'    -.  ;-;;,_\n   |,4-  ) )-,_.
,\\ (  `'-'\n  '---''(_/--'  `-'\\_)\n\n"); // Это ASCII-арт котенка
  sbi_call(0, 0, 0, 0, 0, 0, SBI_EXT_SHUTDOWN, SBI_EXT_SRST);
  while(1);
}
```

Завершает работу системы. Расширение: 0x53525354 (SRST Extension). Функция: 0x0.

5.9. Extension: System Shutdown (EID #0x08)

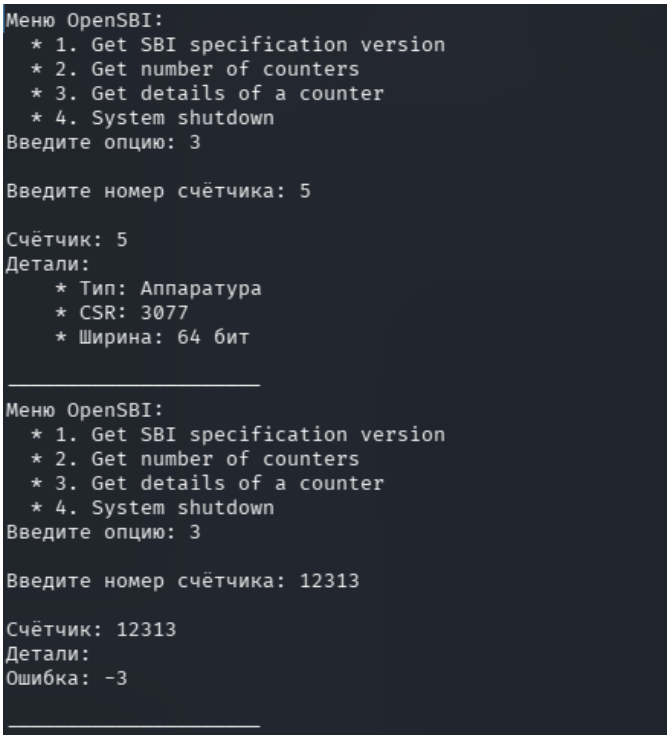
```
void sbi_shutdown(void)
```

Puts all the harts to shutdown state from supervisor point of view.

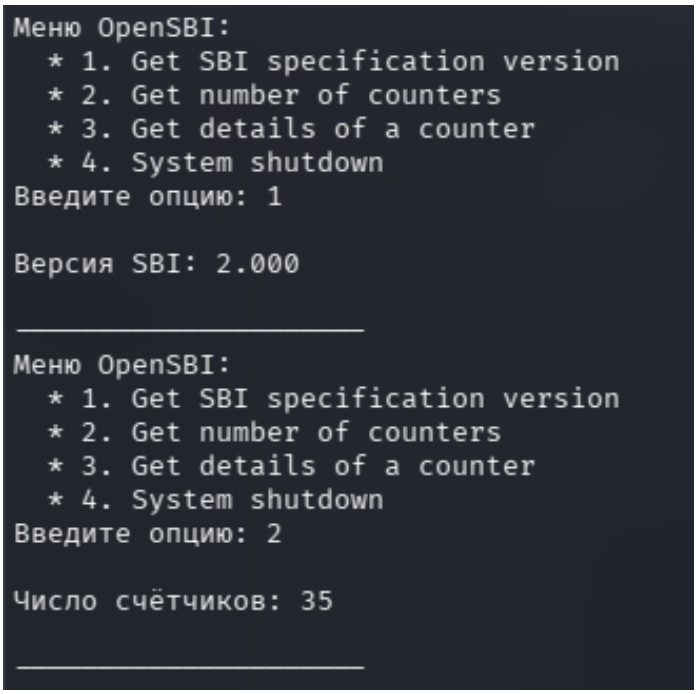
This SBI call doesn't return irrespective whether it succeeds or fails.

Демонстрация

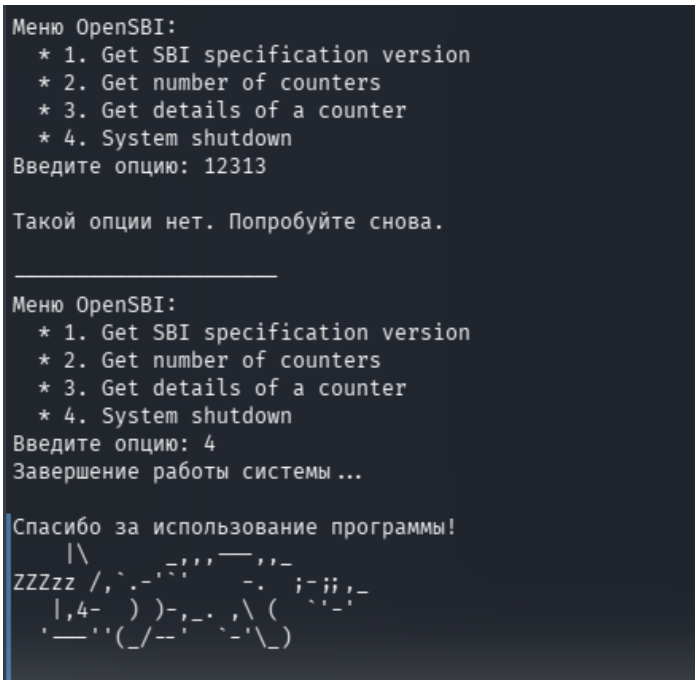
```
> ./run.sh
+ QEMU=qemu-system-riscv32
+ CC=clang
+ CFLAGS='-std=c11 -O2 -g3 -Wall -Wextra --target=riscv32 -ffreestanding -nostdlib'
+ clang -std=c11 -O2 -g3 -Wall -Wextra --target=riscv32 -ffreestanding -nostdlib -Wl,-Tkernel.ld -Wl
,-Map=kernel.map -o kernel.elf kernel.c io.c common.c
+ qemu-system-riscv32 -machine virt -bios default -nographic -serial mon:stdio --no-reboot -kernel k
ernel.elf

OpenSBI v1.5.1
   ___                  ____ ___ ____
  / __ \               / ___|  _ \_   _|
 | |  | |_ __   ___ _ _| |___| |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |___) | |_) || |_
  \____/| .__/ \___|_| |_|____/|____/____|
        | |
        |_|

Platform Name             : riscv-virtio,qemu
Platform Features         : medeleg
Platform HART Count       : 1
Platform IPI Device       : aclint-mswi
Platform Timer Device     : aclint-mtimer @ 10000000Hz
Platform Console Device   : uart8250
Platform HSM Device       : ---
Platform PMU Device       : ---
Platform Reboot Device    : syscon-reboot
Platform Shutdown Device  : syscon-poweroff
Platform Suspend Device   : ---
Platform CPPC Device      : ---
Firmware Base             : 0x80000000
Firmware Size             : 319 KB
Firmware RW Offset        : 0x40000
Firmware RW Size          : 63 KB
Firmware Heap Offset      : 0x47000
Firmware Heap Size        : 35 KB (total), 2 KB (reserved), 10 KB (used), 22 KB (free)
Firmware Scratch Size     : 4096 B (total), 244 B (used), 3852 B (free)
Runtime SBI Version       : 2.0

Domain0 Name              : root
Domain0 Boot HART         : 0
Domain0 HARTs             : 0*
Domain0 Region00          : 0x00100000-0x00100fff M: (I,R,W) S/U: (R,W)
Domain0 Region01          : 0x10000000-0x10000fff M: (I,R,W) S/U: (R,W)
Domain0 Region02          : 0x02000000-0x0200ffff M: (I,R,W) S/U: ()
Domain0 Region03          : 0x80040000-0x8004ffff M: (R,W) S/U: ()
Domain0 Region04          : 0x80000000-0x8003ffff M: (R,X) S/U: ()
Domain0 Region05          : 0x0c400000-0x0c5fffff M: (I,R,W) S/U: (R,W)
Domain0 Region06          : 0x0c000000-0x0c3fffff M: (I,R,W) S/U: (R,W)
Domain0 Region07          : 0x00000000-0xffffffff M: () S/U: (R,W,X)
Domain0 Next Address      : 0x80200000
Domain0 Next Arg1         : 0x87e00000
Domain0 Next Mode         : S-mode
```



```
Меню OpenSBI:
  * 1. Get SBI specification version
  * 2. Get number of counters
  * 3. Get details of a counter
  * 4. System shutdown
Введите опцию: 3

Введите номер счётчика: 5

Счётчик: 5
Детали:
    * Тип: Аппаратура
    * CSR: 3077
    * Ширина: 64 бит

____________________
Меню OpenSBI:
  * 1. Get SBI specification version
  * 2. Get number of counters
  * 3. Get details of a counter
  * 4. System shutdown
Введите опцию: 3

Введите номер счётчика: 12313

Счётчик: 12313
Детали:
Ошибка: -3

____________________
```



```
Меню OpenSBI:
  * 1. Get SBI specification version
  * 2. Get number of counters
  * 3. Get details of a counter
  * 4. System shutdown
Введите опцию: 1

Версия SBI: 2.000

____________________
Меню OpenSBI:
  * 1. Get SBI specification version
  * 2. Get number of counters
  * 3. Get details of a counter
  * 4. System shutdown
Введите опцию: 2

Число счётчиков: 35

____________________
```



```
Меню OpenSBI:
  * 1. Get SBI specification version
  * 2. Get number of counters
  * 3. Get details of a counter
  * 4. System shutdown
Введите опцию: 12313

Такой опции нет. Попробуйте снова.

____________________
Меню OpenSBI:
  * 1. Get SBI specification version
  * 2. Get number of counters
  * 3. Get details of a counter
  * 4. System shutdown
Введите опцию: 4
Завершение работы системы...

Спасибо за использование программы!
    |\      _,,,---,,_
ZZZzz /,`.-'`'    -.  ;-;;,_
   |,4-  ) )-,_. ,\ (  `'-'
  '---''(_/--'  `-'\_)
```

Вывод

В ходе выполнения лабораторной работы я познакомился с принципами организации ввода/вывода без операционной системы на примере компьютерной системы на базе процессора с архитектурой RISC-V и интерфейсом OpenSBI с использованием эмулятора QEMU.