**Лабораторная работа 1**

**По дисциплине “Системы ввода-вывода”**

“Принципы организации ввода/вывода без операционной системы”

Вариант 2

Выполнил:

Баянов Равиль Динарович

P3334

Санкт-Петербург

2025 г.

**Оглавление**

[**Цель** 3](#_Toc190439717)

[**Задачи** 4](#_Toc190439718)

[**Вариант** 5](#_Toc190439719)

[**Описание функций** 6](#_Toc190439720)

[**Вывод** 7](#_Toc190439721)

# **Цель**

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

# **Задачи**

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. Get SBI implementation version

2. Hart get status (должно быть возможно задавать номер ядра)

3. Hart stop

4. System Shutdown

# **Описание функций**

* *Putchar*

|  |
| --- |
| void putchar(char ch) {  sbi\_call(ch, 0, 0, 0, 0, 0, 0, 1 /\* Console Putchar \*/); } |

Вызываем sbi\_call и указываем 1 в EID, для записи символа в консоль.

* Getchar

|  |
| --- |
| char getchar() {  struct sbiret ret;  ret.error = -1;  while (ret.error == -1) {  ret = sbi\_call(0, 0, 0, 0, 0, 0, 0, 2);  }  return (char) ret.error; } |

Вызываем sbi\_call пока ret.error не станет отличным от -1, так как -1 это означает, что ничего не пришло.

* Get SBI implementation version

|  |
| --- |
| uint32\_t get\_sbi\_version() {  struct sbiret ret = sbi\_call(0, 0, 0, 0, 0, 0, 0x1, 0x10);  return (uint32\_t) ret.value; }  void print\_sbi\_version() {  uint32\_t version = get\_sbi\_version();   printf("SBI Implementation Version: %d\n", version); } |
|  |

Вызов sbi\_call с соответствующим EID, а именно 0x10 и указав правильный FID.

* Hart get status (должно быть возможно задавать номер ядра)

|  |
| --- |
| uint32\_t get\_hart\_status(uint32\_t hart\_id) {  struct sbiret ret = sbi\_call(hart\_id, 0, 0, 0, 0, 0, 0, 0x101);  return ret.value; }  void print\_hart\_status(uint32\_t hart\_id) {  uint32\_t status = get\_hart\_status(hart\_id);  printf("Hart %d Status: %d\n", hart\_id, status); } |

Вызов sbi\_call с соответствующим EID, а именно 0x101 и вывод на экран статуса.

* Hart stop

|  |
| --- |
| void stop\_hart(uint32\_t hart\_id) {  sbi\_call(hart\_id, 0, 0, 0, 0, 0, 0, 0x102); } |

Вызов sbi\_call с соответствующим EID, а именно 0x102, предварительно указав id нужного нам hart.

* System Shutdown

|  |
| --- |
| void shutdown() {  sbi\_call(0, 0, 0, 0, 0, 0, 0, 0x08); } |

Вызов sbi\_call с соответствующим EID, а именно 0x08.

# **Вывод**

Get SBI implementation version

Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Hart get status (должно быть возможно задавать номер ядра)

Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Hart stop

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

System Shutdown

