Лабораторная работа 1

1. Цель

Получить общее представление о системе команд RISC-V путём решения несложной прикладной задачи.

2. Задание

1. Создать аккаунт на github (если еще нет) и создать репозиторий с именем lab-riscv-asm.
2. Разработать алгоритм решения вашего варианта задачи.
3. Реализовать алгоритм на языке ассемблера RISCV ISA. Исходный код программы сохранить в репозитории.
4. Реализовать алгоритм на языке Си. Исходный код программы сохранить в репозитории.
5. Скомпилировать Си-код используя riscv-компилятор. Получить дамп-файл из скомпилированного исполняемого файла вашей программы. Сохранить дамп-файл в репозитории.
6. Оформить отчет о проделанной работе и результатах в виде README.md файла вашего репозитоия.
7. Продемонстрировать результаты работы в симуляторе Venus https://www.kvakil.me/venus/

Каждое задание представляет собой простую задачу на обработку двумерного массива. Непосредственную обработку массива необходимо реализовать в виде отдельной функции с необходимыми аргументами. При написании программы на языке ассемблера следует учесть то, что в явном виде подпрограммы/процедуры/функции в RISC-подобных ассемблерах отсутствуют. Вместо этого существует специальный набор соглашений(ABI), устанавливающий формат передачи аргументов, специальную роль для каждого регистра, формат вызовов и так далее. В работе предлагается реализовать передачу аргументов используя один из следующих методов:

- по значению - для передачи переменной используется её непосредственное значение

- по ссылке - для передачи переменной используется указатель
- через стек - значение переменное сохраняется в стеке, вызываемая функция извлекает это значение с вершины стека

Для реализации этих методов необходимо использовать соответствующие регистры, описанные в стандарте RISC-V ISA. Передавать параметры необходимо только процедуре, осуществляющей непосредственную обработку массива, остальное - по желанию.

3. Варианты заданий

Во всех заданиях используется двумерный массив размера N на M, где N - размер строки, а M - размер столбца.

Номер варианта	**ФИО**	**Задание**	**Передача параметров**
1	Баранец Максим	Найти значение минимального элемента массива. (N = 5, M = 4)	Через стек
2	Криворотова Полина	Найти индекс максимального элемента массива. (N = 4, M = 3)	Через стек
3	Стручинский Артем	Найти сумму положительных элементов массива. (N = 4, M = 2)	по значению
4	Бондаренко Михаил	Определить количество положительных и отрицательных элементов массива. (N = 5, M = 6)	Через указатель
5	Смирнов Александр	Найти сумму элементов строки с заданным номером. (N = 4, M = 4)	по значению
6	Мирошниченко Владислав	Найти сумму элементов массива (N = 4, M = 4)	по значению
7	Тищук Богдана	Индивидуальное задание	-----

4. Пример выполнения

Пример выполнения доступен в репозитории https://github.com/sc-itmo-socdes-sp20/lab-riscv-asm

5. Дополнительная информация

5.1. Окружение

Очень настоятельно рекомендуем использовать Linux Mint/Ubuntu

В случае, если у вас установлена windows, то рекомендуем установить виртуальную машину: https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html

На виртуальную машину установить Linux: https://linuxmint.com/download.php или https://ubuntu.com/download/desktop

5.2. Git

Полная документация по работе с Git на русском языке https://git-scm.com/book/ru/v2

Пример работы с гитом (создание репо и загрузка его на сервер):

```
sudo apt-get install git
git config --global user.name "FIRST_NAME LAST_NAME"
git config --global user.email "EMAIL"

cd work_dir
git init
git add .
git commit -am "my first commit"
git remote add origin http....<link for repo>
git push origin master
```

5.3. RISC-v

Спецификация The RISC-V Instruction Set Manual (Unprivileged), доступную по ссылке: https://github.com/riscv/riscv-isa-manual/releases/download/Ratified-IMAFDQC/riscv-spec-20191213.pdf

Руководство по ассемблеру RISC-V Assembly Programmer's Manual находится по ссылке: https://github.com/riscv/riscv-asm-manual/blob/master/riscv-asm.md

5.4. Симулятор Venus

Для запуска и отладки тестовой программы необходимо использовать симулятор: https://www.kvakil.me/venus/

Очень рекомендуется использовать Visual Studio Code с расширением Name: RISC-V Venus Simulator

Симулятор поддерживает следующие директивы:

Directive	Effects
.data	Store subsequent items in the [[static segment
.text	Store subsequent instructions in the [[text segment
.byte	Store listed values as 8-bit bytes.
.asciiz	Store subsequent string in the data segment and add null-terminator.
.word	Store listed values as unaligned 32-bit words.
.globl	Makes the given label global.
.float	Reserved.
.double	Reserved.
.align	Reserved.

Симулятор поддерживает обработку следующих системных вызовов:

ID	Name	Description
1	print_int	prints integer in a1
4	print_string	prints the null-terminated string whose address is
9	sbrk	allocates a1 bytes on the heap, returns pointer to start in a0

ID	Name	Description
10	exit	ends the program
11	print_character	prints ASCII character in a1
17	exit2	ends the program with return code in a1

Полный User Guide по работе симулятора: https://github.com/kvakil/venus/wiki

5.5. Компилятор C

Для компиляции исходного кода, написанного на СИ, вам необходимо использовать специальный RISCV-совместимый компилятор. Вы можете собрать этот компилятор

- из исходников, следуя инструкциям из официального репозитория https://github.com/riscv/riscv-gcc
- или же использовать заранее собранный https://drive.google.com/file/d/16bmrM-W7LEGVLUZhkgr60LhGSM8esl6m/view?usp=sharing

Ручная сборка займет довольно продолжительное время, поэтому **рекомендуется** использовать второй вариант.

5.6. Настройка для Linux

1. Необходимо распаковать скачанный архив в одну из доступных вашему пользователю директорий(например, в /home/{ИМЯ_ПОЛЬЗОВАТЕЛЯ}/riscv-tools/})
2. Необходимо добавить путь к директории bin в переменную окружения $PATH. Например, выполнив в консоли: export PATH=/home/{ИМЯ_ПОЛЬЗОВАТЕЛЯ}/riscv-tools/{ИМЯ_РАЗАРХИВИРОВАННОЙ_ДИРЕКТОРИИ}/bin:$PATH

5.7. Компиляция

После выполненной процедуры настройки, в окружении той рабочей консоли, в которой выполнялась настройка, вы можете запустить процедуру компиляции. Компилировать необходимо с флагами -march=rv32i -mabi=ilp32. Например:

```
$ riscv64-unknown-elf-gcc -march=rv32i -
mabi=ilp32 {ПУТЬ_К_ВАШЕМУ_ИСХОДНОМУ_ФАЛУ} -o
 {ИМЯ_РЕЗУЛЬТИРУЮЩЕГО_ИСПОЛНЯЕМОГО_ФАЛА}.elf
```

5.8. Получение dump файла

Для получения дамп-файла необходимо использовать утилиту objdump. Например:

```
$ riscv64-unknown-elf-objdump -D {ПУТЬ_К_ИСПОЛНЯЕМОМУ_ФАЙЛУ} >
 {ИМЯ_РЕЗУЛЬТИРУЮЩЕГО_ФАЙЛА}.dump
```