Машинно-зависимые языки программирования

Лабораторная работа №12

“Программирование ARM-процессоров”

Справочная информация

Общие сведения об архитектуре ARMv8

Архитектура ARMv8 включает в себя:

- 31 64-битный регистр R0..R30. Доступ осуществляется по алиасам X0..X30 и по алиасам W0..W30 для младших 32-битных половин. При записи значений в регистры W0..W30 старшие половины регистров зануляются, в отличие от x86;
- PC (program counter) - программный счётчик, аналог IP в x86;
- LR (алиас для X30) - ссылочный регистр. Хранит в себе последний адрес возврата;
- SP - указатель стека;
- псевдорегистр PSTATE, который прочитать напрямую нельзя, и связанный регистр флагов NZCV;
- V0..V31 - 128-битные SIMD-регистры расширения NEON (аналог SSE в x86). Доступ осуществляется по алиасам Q0..Q31 для всех 128 бит и по другим алиасам для младших частей.

Базовые команды:

- LDR/STR - для загрузки значений из памяти в регистр/сохранения в память;
- MOV - для загрузки непосредственных операндов в регистр, а также пересылки в память и обратно;
- add, sub - трёхоперандные команды сложения/вычитания (a = b + c, a = b - c);
- and, orr - трёхоперандные команды побитовых операций;
- b - переход на метку (ветвление, branching). аналог команды безусловного перехода JMP в x86;
- bl - переход на метку с сохранением адреса возврата в LR;
- br - аналог b, но осуществляет переход по адресу из регистра;
- blr - аналог bl, но осуществляет переход по адресу из регистра;
- cmp - команда сравнения;
- beq, bne, blt, ble, bgt, bge и т.д. - команды условного ветвления.

Команды NEON - команды для выполнения поточных операций, описание доступно на сайте:
https://developer.arm.com/documentation/102474/0100/Fundamentals-of-Armv8-Neon-technology
https://developer.arm.com/documentation/102159/0400/Overview?lang=en
https://developer.arm.com/documentation/ihi0073/h/?lang=en

Ссылки для самостоятельного изучения:
https://developer.arm.com/documentation/ddi0487/latest/
http://www.mka.ru/categories/91/681/
https://habr.com/ru/articles/353994/
https://habr.com/ru/articles/548698/

Эмулятор qemu

QEMU — opensource ПО для эмуляции аппаратного обеспечения различных платформ.

1. Установить qemu, скачать следующие файлы:
   https://drive.google.com/file/d/1RySqPQcegOohRYewlPTk1aASMvOMurkW/view?usp=drive_link
   https://drive.google.com/file/d/13MnylhtvbklbhtI2BDUKkR4H8KH7McO_/view?usp=drive_link
   https://drive.google.com/file/d/1d6-1UY4Rp1ahITERgENRKCtun1oMOm3x/view?usp=drive_link
2. Запуск

```
qemu-system-aarch64 -machine virt -cpu cortex-a72 -nographic -smp
1 -m 512 -kernel vmlinuz-run -initrd initrd-run.img -append
"root=/dev/sda2 console=ttyAMA0" -global
virtio-blk-device.scsi=off -device virtio-scsi-device,id=scsi
-drive file=debian11-arm64.img,id=rootimg,cache=unsafe,if=none
-device scsi-hd,drive=rootimg -virtfs
local,path=/home/user/path-to-mount,mount_tag=host0,security_model=
none,id=host0
```

Вместо **`/home/user/path-to-mount`** нужно указать существующий в основной ОС каталог, который затем будет примонтирован в гостевую ОС.

3. Пароль пользователей iu7 и root: 1234.

4. Монтирование хост-каталога (в каталог asm в домашней папке пользователя iu7, должно выполняться из-под root):

```
mount -t 9p -o trans=virtio host0 asm -oversion=9p2000.L
```

(Подобный образ любой ОС под aarch64 можно получить по шагам, аналогичным описанным в инструкции: https://gist.github.com/philipz/04a9a165f8ce561f7ddd)

В случае хостовой ОС Windows virtfs может быть недостпна, тогда потребуется подключать диск с использованием samba либо копировать/редактировать исходный код через nano.

Практическое задание

Для компьютеров с процессорами x86-64 установить эмулятор QEMU, для ARM-процессоров задание можно выполнить без использования эмулятора.

1. С использованием ассемблерной вставки определить длину строки.
2. С использованием расширения NEON реализовать какую-либо операцию поточной обработки данных: сложение массивов, вычисление скалярного произведения и т.д.