Лабораторная №3. Использование кросс-компилятора.

Цель работы

Научиться собирать программы для RISC-V на x86_64 машине.

Подготовительный материал

На момент выполнения лабораторной №3 читатель находится в следующих обстоятельствах:

- есть плата с архитектурой RISCV, под которую нужно писать программный код;
- есть ПК на архитектуре x86_64, на котором удобно писать программный код;

Возникает довольно обоснованный вопрос: "Как организуют работу с кодом люди, которые постоянно разрабатывают программный код, поставляемый на процессорные архитектуры отличные от x86_64".

Общий принцип всегда один:

- пишется код
- компилируется кросс-компилятором
- так или иначе переносится на целевую платформу ИЛИ целевая платформа эмулируется
- код запускается и отлаживается

Опустим процесс написания кода и рассмотрим остальные этапы этого принципа.

Кросс-компилляция

Для компиляции программы на хостовой машине под целевую платформу необходимо использовать специальный**тулчейн**, который может быть установлен вместе с пакетом *gcc-riscv64-linux-gnu*.

Тулчейн (Toolchain) — это набор инструментов (утилит, компиляторов, библиотек), которые используются для разработки и сборки программного обеспечения под определенную платформу или архитектуру.

Главное, что оказывается необходимым — **кросс-компилятор**.

Кросс-компилятор — это специальный компилятор, который позволяет компилировать код для платформы, отличной от той, на которой он выполняется.

Где можно найти кросс-компилятор? Например в репозитории для разработчиков Альт Линукс!

Настройка репозитория пакетов

Классически в дистрибутивах Alt все пакеты устанавляиваются из репозиториев, которые поддерживаются сообществом ALT(Alt Linux Team).

Узнать, о том, из каких репозиториев пакетный менеджер apt выкачивает программные компаненты можно с помощью следующего вызова:

```
apt-repo
```

Зачастую многие инструменты необходимые для разработки находятся в специальном репозитории под названием Sisyphus. Для переключения на него(если ваша машина еще не базируется на нем) нужно выполнить от root:

```
apt-repo rm all

apt-repo set sisyphus

apt-get update
```

Установка кросс-компилятора

Настроив работу с репозиторием на машине, специальный тулчейн с кросс-компилятором можно установить одной программой:

```
# apt-get install gcc-riscv64-linux-gnu
```

Итак. Кросс-компилятор — есть.

Среда для написания кода

Нужно решить, где писать программный код.

Классический опыт написания кода связан с разработкой с помощью IDE. Как показывает практика, в сфере embedded-разработки этот подход является менее популярным. По сути IDE является всего лишь надстройкой(пускай зачастую вполне удобной и полезной) над консольными командами, которые выполняют обращения к компиляторам, интерпретаторам и отладчикам с определенными параметрами в зависимости от настроек, установленных пользователем.

Зачастую многим разработчикам привычно пользоваться популярными редакторами такими как vim, neovim, nano и emacs.

В рамках курса вероятно студентом будет затронуто оба варианта написания кода.

В рамках данной лабораторной будет достаточно воспользоваться любым из перечисленных редакторов.

Пример компиляции и тестовый запуск

Скомпилировать исполняемый файл на хостовой машине можно так:

```
riscv64-linux-gnu-gcc -o  название_исполняемого_файла файл_исходник.c
[библиотеки]
```

Кроме того, альтернативным вариантом может быть компиляция непосредственно на целевой плате или использование технологии qemu-user-static-binfmt, которая позволяет запускать на хостовой системе бинарники для других архитектур.

Достаточно установить пакет из репозитория Sisyphus:

```
# apt-get install qemu-user-static-binfmt-riscv
```

И попробовать выполнить скомпилированный ранее файл.

Задание

Ознакомившись с подготовительным материалом к лабораторной №3 выполнить следующие подзадачи:

- Убедиться в наличии или установить кросс-компилятор и подходящий вам редактор на хостовой машине
- Написать приложение, которое с помощью библиотеки math высчитывает 5 значений любых элементарных математических функций для аргумента передаваемого приложению через CLI(см. пример работы ниже)
- Скомпилировать программу кросс-компилятором
- Попробовать запустить программу с помощью qemu-user-static-binfmt
- Передать исполняемый файл на плату и выполнить его
- **Продемонстрировать работу преподавателю**
- Сформировать отчет о выполнении поставленных задач .doc и **выслать на почту преподавателя до обозначенного срока**

```
[taranev@taranev work_repos]$ ./test_cli 2
Math calculations for x = 2.0000:
================================
sqrt(x)   = 1.414214
x^2       = 4.000000
x^3       = 8.000000
sin(x)    = 0.909297
cos(x)    = -0.416147
```

Пример работы приложения