Я благодарен своим родителям и родственникам
за их постоянную поддержку и мотивацию

Оглавление

От издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте **www.dmkpress.com**, зайдя на страницу книги и оставив комментарий в разделе «*Отзывы и рецензии*». Также можно послать письмо главному редактору по адресу **dmkpress@gmail.com**; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу **http://dmkpress.com/authors/publish_book/** или напишите в издательство по адресу **dmkpress@gmail.com**.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу **dmkpress@gmail.com**. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты **dmkpress@gmail.com**.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Об авторе

Андрей Ипполитов родился в Москве и с раннего возраста увлекся программированием. После получения образования в МГУПС он много экспериментировал с разными языками программирования. Автор живет в Подмосковье, помимо писательства, активно занимается созданием программы *kitasm*. Его жизненная философия вдохновляет читателей на программирование.

Сейчас автор работает над новыми проектами и продолжает радовать своих поклонников свежими сюжетами и идеями.

Пролог

Ночь в лаборатории была теплой и тихой – свет индикаторов на панелях мерцал, как звезды в искусственном небе. За столом, у старого терминала с потертым шрифтом на экране, сидел человек, для которого цифры и биты были больше, чем просто знаки: они были языком мышления, тайной, которую стоило расшифровать.

Он вспомнил свои первые шаги: как нерешительно вводил команды, наблюдая за реакцией машины, как одна простая инструкция могла изменить поток исполнения, как каждая метка и каждое смещение были крошечными рычагами управления вселенной микропроцессора. Ассемблер казался тогда заклинанием – строгой рифмовкой операторов и регистров, где ошибка в одном символе могла превратить систему в молчаливого монстра.

Эта книга – не о магии и не о забытом ремесле, а о языке, который позволяет взглянуть в сердце вычислений. Здесь нет волшебных трюков, только чистая логика и точность: как строятся инструкции, как передаются данные, как осуществляется контроль потока, как из простых операций вырастает сложная программа. Ассемблер – это алхимия простых операций, превращающая электрические импульсы в смысл.

Читатель встретит здесь не сухие описания и не абстрактные теории, а практику: примеры, отладочные приемы, шаблоны решения типовых задач и объяснения, почему тот или иной подход оказывается эффективнее. Мы пройдем путь от базовых команд к реальным программам, от простейших циклов до управления прерываниями, работы с памятью и оптимизации исполнения.

Если вы пришли из высокоуровневых языков – приготовьтесь к смене парадигмы. Ассемблер потребует точности мышления, понимания архитектуры и умения мыслить в терминах машинных состояний. Но взамен он даст необычную, почти интимную близость к машине: вы научитесь слышать ее ритм и предугадывать ее поведение.

Если же вы уже знакомы с ассемблером – возможно, вы найдете здесь новые взгляды и полезные приемы, способные сделать код компактнее, быстрее и яснее.

Перед началом: выключите лишние шумы, подготовьте отлаживающую среду и запаситесь терпением. Путь будет нелегким, но каждое пройденное испытание откроет новую грань понимания того, как устроен мир компьютеров – от стека вызовов до мельчайших переключений флагов.

Добро пожаловать в мир, где байты говорят правду.

Введение

Добро пожаловать в мир низкоуровневого программирования, где каждая инструкция имеет значение, а аппаратное обеспечение раскрывает свои самые сокровенные тайны! Эта книга – ваш проводник в увлекательное путешествие по архитектуре ARM64 и языку ассемблера, ориентированному на платформу macOS. Если вы когда-либо задумывались, как на самом деле работают программы, что скрывается за магией высокоуровневых языков, или стремитесь получить максимальную производительность от своего Mac, то эта книга для вас.

Почему ARM64 и macOS?

Современные устройства Apple, от iPhone и iPad до MacBook и Mac Pro, работают под управлением процессоров на базе архитектуры ARM64. Это означает, что понимание ARM64 – это ключ к пониманию сердца большинства устройств, которыми мы пользуемся каждый день. macOS как операционная система тесно интегрирована с этой архитектурой, предоставляя уникальные возможности и инструменты для разработки на ассемблере.

Что такое Assembler?

Assembler (или язык ассемблера) – язык программирования, который находится на очень низком уровне. Он представляет собой почти прямое отображение машинных инструкций, которые процессор выполняет напрямую. В отличие от высокоуровневых языков, таких как Python, Java или C++, где вы работаете с абстракциями и командами, ассемблер требует от вас понимания таких вещей, как:

- *регистры*: крошечные, сверхбыстрые ячейки памяти внутри процессора, используемые для временного хранения данных и адресов;
- *инструкции*: фундаментальные операции, которые процессор может выполнять, такие как сложение, вычитание, перемещение данных, условные переходы и вызовы функций;

- *память*: адресация ячеек основной памяти, где хранятся данные и код программы;
- *архитектура набора команд*: набор всех инструкций, которые может выполнять конкретный процессор.

Почему стоит изучать Assembler на macOS ARM64?

Хотя ассемблер может показаться сложным и трудоемким, его изучение открывает двери к:

- *глубокому пониманию работы компьютера*: вы начнете видеть, как программы взаимодействуют с аппаратным обеспечением на самом фундаментальном уровне;
- *оптимизации производительности*: для критически важных участков кода, где важна каждая наносекунда, ассемблер позволяет добиться максимальной скорости выполнения;
- *разработке операционных систем и драйверов*: эти области часто требуют прямого взаимодействия с аппаратным обеспечением;
- *реверс-инжинирингу и анализу вредоносного ПО*: понимание ассемблера – неотъемлемая часть изучения того, как программы работают «изнутри», что важно для безопасности;
- *созданию кросс-платформенных решений*: знание ARM64 пригодится вам при работе с различными устройствами, использующими эту архитектуру;
- *отладке сложных проблем*: иногда только понимание ассемблерного кода может помочь выявить и исправить трудноуловимые ошибки.

Что вы узнаете из этой книги

В этой книге мы последовательно разберем:

- *основы архитектуры ARM64*: понимание регистровой модели, режимов выполнения и общих концепций;
- *базовые инструкции ARM64*: изучение основных операций для работы с данными, арифметики, логики и управления потоком выполнения;

- *системные вызовы macOS*: как ваша программа взаимодействует с операционной системой для выполнения таких задач, как ввод/вывод, выделение памяти и работа с файлами;
- *процесс сборки и отладки*: использование инструментов, доступных на macOS, для компиляции и отладки ассемблерного кода;
- *практические примеры*: от простых программ до более сложных задач, демонстрирующих применение полученных знаний;
- *особенности ARM64 на macOS*: специфические моменты, связанные с работой на этой платформе.

Кому предназначена эта книга:

- *начинающим программистам*: если вы хотите копнуть глубже, чем просто написание скриптов;
- *опытным разработчикам*: желающим расширить свои знания и освоить новые парадигмы программирования;
- *системным администраторам и DevOps-специалистам*: интересующимся внутренним устройством систем;
- *студентам технических специальностей*: изучающим информатику, компьютерную инженерию или смежные области.

Предварительные знания

Для успешного освоения материала желательно иметь базовое представление о принципах работы компьютеров и программирования. Знакомство с каким-либо высокоуровневым языком программирования также будет плюсом, так как вы сможете проводить параллели и лучше понимать абстракции.

Путь к мастерству

Изучение ассемблера – это марафон, а не спринт. Не ожидайте, что вы станете экспертом за одну ночь. Главное – это терпение, практика и последовательность. Я предлагаю вам вместе шаг за шагом разбирать каждую тему, решать практические задачи и экспериментировать.

Готовы ли вы к этому захватывающему путешествию? Откройте для себя мир, где код оживает и где вы контролируете каждое действие процессора.

Далее, по умолчанию, все относится к ассемблеру GAS для macOS на чипе ARM64 (M chip).

Прежде чем начать

Прежде чем писать код, нам нужно убедиться, что у нас есть все необходимое. На macOS для разработки на ассемблере нам потребуются: `Xcode Command Line Tools` – это набор утилит командной строки, который включает в себя компилятор (ассемблер) `as` и компоновщик `ld`, а также отладчик `lldb`. Если они еще не установлены, вы можете сделать это, выполнив в *Терминале*:

```
$ xcode-select –install
```

Затем установим IDE. IDE расшифровывается как интегрированная среда разработки (Integrated Development Environment). Это программное обеспечение, которое предоставляет разработчикам следующие функции:

- *редактор кода*: удобный инструмент для написания и редактирования программного кода;
- *компилятор или интерпретатор*: преобразует написанный код в исполняемый файл;
- *отладчик*: помогает находить и устранять ошибки в коде.

IDE значительно упрощает процесс разработки, собирая все необходимые инструменты в одном приложении. Мы скачаем и установим программу `kitasm`.

Для установки пройдите на сайт *www.kitasm.site* и загрузите последнюю версию. Программа платная, но можно скачать демоверсию, она тоже подойдет. Затем разархивируйте пакет с программой, войдите в терминал и введите команду

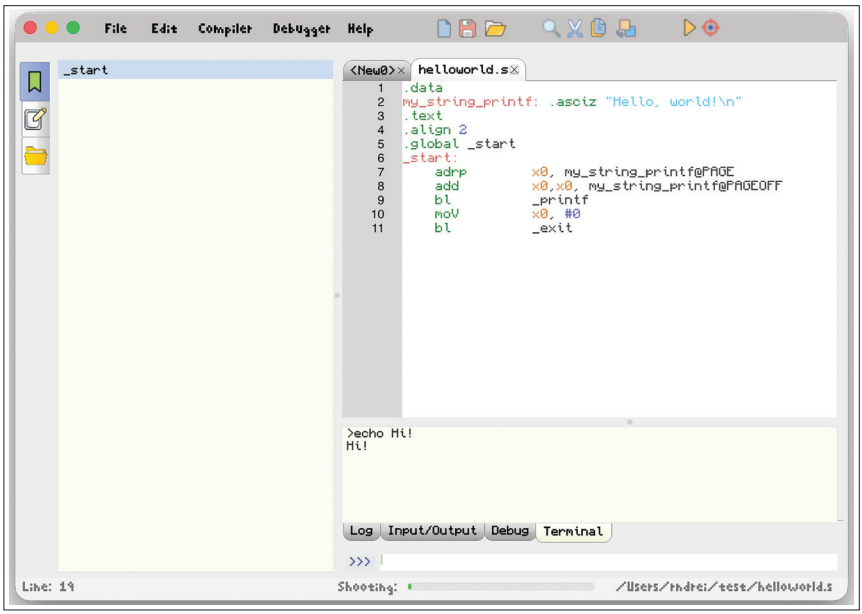
```
$ sudo xattr -r -d com.apple.quarantine kitasm.app
```

Эта команда разрешит запуск программы, загруженной из интернета. Теперь запустим `kitasm`.

Ниже на рисунке можно видеть, как выглядит запущенная программа `kitasm`.

- Слева панель для отображения меток и переменных и для вывода регистров дебаггером.

- Посередине простой редактор кода с подсвечиваемым синтаксисом.
- Внизу окно для вывода логов.



Для увеличения или уменьшения размера шрифта используйте клавиши **Ctrl+**, **Ctri–**.

Глава 1

Первая программа

Создание первой программы с использованием ассемблера – увлекательный процесс, который позволяет увидеть, как работает низкоуровневое программирование. Давайте создадим простую программу, которая выводит текст на экран.

Hello world

Откроем `kitasm` и напишем следующий код:

```
.global _start
.text
_start:
    ; Системный вызов для записи (write)
    ; x0 = файловый дескриптор (1 - stdout)
    ; x1 = указатель на буфер с сообщением
    ; x2 = длина сообщения
    mov x0, 1               ; stdout
    adr x1, message         ; Указатель на строку message
    mov x2, #13             ; Длина строки "Hello, World!\n"
    mov x16, 4              ; Номер системного вызова SYS_write
    svc #0                  ; Вызов ядра

    ; Системный вызов для завершения программы (exit)
    ; x0 = код возврата (0 - успешно)
    mov x0, 0               ; Код возврата 0
    mov x16, 1              ; Номер системного вызова SYS_exit
    svc #0                  ; Вызов ядра

message:
    .ascii "Hello, World!\n"
```

Теперь сохраним в «*Hello, World.s*» и запустим его на выполнение. Для этого нажимаем правую кнопку мыши и выбираем пункт «**run**». Если все прошло успешно, внизу программы *kitasm*, в окне **input/output** вы увидите вывод фразы «Hello world!». Также там будет показан выход из программы с параметром 0. Это означает, что программа завершилась без ошибок.

```
Hello, World!
End program. Exitcode: 0
```

В папке, куда вы сохранили, появится исполняемый файл «*helloworld*». Его можно запустить, написав в терминале команду:

```
$ ./helloworld
```

> *Комментарии в ассемблере обозначаются символом «;» или «//».*

Поздравляю! Вы только что написали и запустили свою первую программу на ассемблере. Этот простой пример иллюстрирует базовые принципы взаимодействия с операционной системой и работу с системными вызовами на низком уровне. Дальнейшее изучение ассемблера открывает множество возможностей для оптимизации кода и понимания работы компьютеров на более глубоком уровне.

Разберем код построчно.

- `.global start`: эта директива сообщает компоновщику, что метка `_start` является глобальной. В macOS точка входа в программу (начало исполнения) обычно называется `_start`.
- `.text`: эта директива указывает, что далее следует секция кода, где будут располагаться исполняемые инструкции.
- `_start`: это метка. В ассемблере метки используются для обозначения адресов инструкций или данных. К ним можно обращаться для переходов или вызовов.
- `mov x0, 1`: инструкция `mov` (`move`) перемещает значение 1 в регистр x0. Регистры x0–x30 используются для передачи аргументов системным вызовам и функциям. Для си-