Нашим семьям

Оглавление

Похвальные отзывы на книгу «Цифровая схемотехника и архитектура компьютера». Дополнение по архитектуре ARM

Харрис и Харрис проделали замечательную похвальную работу по созданию действительно стоящего учебника, ясно показывающего их любовь и страсть к преподаванию и образованию. Студенты, прочитавшие эту книгу, будут благодарны Харрису и Харрис многие годы после окончания обучения. Стиль изложения, ясность, подробные диаграммы, поток информации, постепенное повышение сложности предмета, великолепные примеры по всем главам, упражнения в конце глав, краткие, но понятные объяснения, полезные примеры из реального мира, покрытие всех аспектов каждой темы – все эти вещи проделаны очень хорошо. Если вы студент, пользующийся этой книгой для подготовки к своему курсу, приготовьтесь получать удовольствие, поражаться, а также многому обучаться!

Мехди Хатамиан, старший вице-президент Broadcom

Харрис и Харрис проделали превосходную работу по созданию ARM версии своей популярной книги «Цифровая схемотехника и архитектура компьютера». Переориентация на ARM – это сложная задача, но авторы успешно справились с ней, при этом оставив свой ясный и тщательный стиль изложения, а также выдающееся качество включенной в текст документации. Я полагаю, что это новое издание будет очень хорошо принято как студентами, так и профессионалами.

Дональд Хунг, государственный университет Сан-Хосе

Из всех учебников, что я рецензировал и рекомендовал за 10 лет профессорства, «Цифровая схемотехника и архитектура компьютера» является одним из всего лишь двух, которые безусловно стоит купить (другой такой учебник – «Архитектура компьютера и проектирование компьютерных систем»). Изложение ясное и краткое, диаграммы просты для понимания, а процессор, который авторы используют в качестве рабочего примера, достаточно сложен, чтобы быть реалистичным, но достаточно прост, чтобы быть полностью понятным для моих студентов.

Захари Курмас, государственный университет Гранд Вэлли

Книга дает свежий взгляд на старую дисциплину. Многие учебники напоминают неухоженные заросли кустарника, но авторы данного учебника сумели отстричь засохшие ветви, сохранив основы и представив их в современном контексте. Эта книга поможет студентам справиться с техническими испытаниями завтрашнего дня.

Джим Френзел, Университет Айдахо

Книга написана в информативном, приятном для чтения стиле. Материал представлен на хорошем уровне для введения в проектирование компьютеров и содержит множество полезных диаграмм. Комбинационные схемы, микроархитектура и системы памяти изложены особенно хорошо.

Джеймс Пинтер-Люк, Колледж им. Дональда Маккенны, Клермонт

Харрис и Харрис написали очень ясную и легкую для понимания книгу. Упражнения хорошо разработаны, а примеры из реальной практики являются замечательным дополнением. Длинные и вводящие в заблуждение объяснения, часто встречающиеся в подобных книгах, здесь отсутствуют. Очевидно, что авторы посвятили много времени и усилий созданию доступного текста. Я настоятельно рекомендую книгу.

Пейи Чжао, Университет Чепмена

Предисловие

Эта книга уникальна тем, что описывает процесс проектирования цифровых систем с точки зрения компьютерной архитектуры, начиная от единиц и нулей и заканчивая разработкой микропроцессора.

Мы считаем, что построение микропроцессора – это особый обряд посвящения для студентов инженерных и компьютерных специальностей. Внутренняя работа процессора кажется почти волшебной для непосвященных, но после подробного объяснения оказывается простой для понимания. Проектирование цифровых систем – само по себе мощный и захватывающий предмет. Программирование на языке ассемблера позволяет увидеть внутренний язык, на котором говорит процессор. Микроархитектура является тем самым звеном, которое связывает эти части воедино.

Первые два издания этой все более набирающей популярность книги описывали архитектуру MIPS, следуя традиции широко распространенных книг по архитектуре Паттерсона и Хеннесси. Будучи одной из первых архитектур с сокращенным набором команд (RISC), MIPS опрятна и исключительно проста для понимания и разработки. MIPS остается важной архитектурой и получила приток свежих сил, после того как Imagination Technologies приобрела ее в 2013 году.

За последние два десятилетия архитектура ARM испытала взрыв популярности, причина которого – в ее эффективности и богатой экосистеме. Было произведено более 50 миллиардов процессоров ARM, и более 75% людей на планете пользуются продуктами с процессорами ARM. На момент написания данного текста почти каждый проданный сотовый телефон и планшет содержал один или несколько процессоров ARM. По прогнозам десятки миллиардов ARM систем вскоре будут контролировать интернет вещей (Internet of Things). Многие компании разрабатывают высокопроизводительные ARM-системы, чтобы бросить вызов Intel на рынке серверов. По причине такой коммерческой важности и интереса студентов мы написали данное ARM издание книги.

С педагогической точки зрения цели изданий MIPS и ARM одни и те же. Архитектура ARM имеет ряд особенностей, таких как режимы адресации и условное выполнение, которые вносят ощутимый вклад в эффективность, но при этом добавляют совсем немного сложности. К тому же эти микроархитектуры очень похожи, а условное выполнение и счетчик команд являются их самыми большими различиями. Глава о вводе-выводе содержит множество примеров, использующих Raspberry Pi – популярный одноплатный компьютер на основе ARM с Linux.

Мы рассчитываем предоставлять как MIPS-, так и ARM-издания до тех пор, пока рынок требует этого.

Особенности книги

Эта книга содержит ряд особенностей. В книге ссылки на главы основной книги «Цифровая схемотехника и архитектура компьютера» помечены.

Одновременное использование языков SystemVerilog и VHDL

Языки описания аппаратуры (hardware description languages, HDL) находятся в центре современных методов проектирования сложных цифровых систем. К сожалению, разработчики делятся на две примерно равные группы, использующие два разных языка – SystemVerilog и VHDL. Языки описания аппаратуры рассматриваются в **главе 4** (книга 1), сразу после глав, посвященных проектированию комбинационных и последовательных логических схем. Затем языки HDL используются в **главах 5 и 7** (книга 1) для разработки цифровых блоков большего размера и процессора целиком. Тем не менее **главу 4** (книга 1) можно безболезненно пропустить, если изучение языков HDL не входит в программу.

Эта книга уникальна тем, что использует одновременно и SystemVerilog, и VHDL, что позволяет читателю освоить проектирование цифровых систем сразу на двух языках. В **главе 4** (книга 1) сначала описываются общие принципы, применимые к обоим языкам, а затем вводится синтаксис и приводятся примеры использования этих языков. Этот двуязычный подход облегчает преподавателю выбор языка HDL, а читателю позволит перейти с одного языка на другой как во время учебы, так и в профессиональной деятельности.

Архитектура и микроархитектура классического процессора ARM

Главы 1 и 2 содержат первый всесторонний обзор архитектуры и микроархитектуры ARM. ARM – идеально подходящая для изучения архитектура, поскольку она является реальной архитектурой, поставляемой в составе миллионов продуктов ежегодно, но, несмотря на это, она рациональна и проста в освоении. Более того, ввиду популярности в коммерческом и любительском мирах существует немало средств разработки и эмуляции для архитектуры ARM. Все материалы, связанные с технологией ARM®, воспроизводятся с разрешения ARM Limited.

Живые примеры

В дополнение к живым примерам, обсуждаемым в связи с архитектурой ARM, в **главе 1** в качестве стороннего примера рассматривается архитектура процессоров Intel x86. **Глава 4** (доступная также в качестве онлайн-дополнения) описывает периферийные устройства в контексте одноплатного компьютера Raspberry Pi – весьма популярной платформы на базе ARM. Эти живые примеры показывают, как описанные в данных главах концепции применяются в реальных микросхемах, которые широко используются в персональных компьютерах и бытовой электронике.

Доступное описание высокопроизводительных архитектур

Глава 2 содержит краткий обзор современных высокопроизводительных микроархитектур: с предсказанием переходов, суперскалярной, с внеочередным выполнением команд, многопоточной и многоядерной. Материал изложен в доступной для первокурсников форме и показывает, как можно расширить микроархитектуры, описанные в книге, чтобы получить современный процессор.

Упражнения в конце глав и вопросы для собеседования

Лучшим способом изучения цифровой схемотехники является разработка устройств. В конце каждой главы приведены многочисленные упражнения. За упражнениями следует набор вопросов для собеседования, которые наши коллеги обычно задают студентам, претендующим на работу в отрасли. Эти вопросы предлагают читателю взглянуть на задачи, с которыми соискателям придется столкнуться в ходе собеседования при трудоустройстве. Решения упражнений доступны через веб-сайт книги и специальный веб-сайт для преподавателей.

Материалы в Интернете

Дополнительные материалы для этой книги доступны на веб-сайте по адресу http://booksite.elsevier.com/9780128000564. Этот веб-сайт доступен всем читателям и содержит:

- Решения нечетных упражнений.
- Ссылки на профессиональные средства автоматизированного проектирования (САПР) компании Altera®.
- Ссылку на Kiel ARM Microcontroller Development Kit (MDK-ARM) – инструменты для компиляции, ассемблирования и эмуляции Си и ассемблерного кода для процессоров ARM.
- HDL-код процессора ARM.
- Полезные советы по использованию САПР Altera Quartus II.
- Слайды лекций в формате PowerPoint.
- Образцы учебных и лабораторных материалов для курса.
- Список опечаток.

Также существует специальный веб-сайт для преподавателей, зарегистрировавшихся на http://booksite.elsevier.com/9780128000564, который содержит:

- Решения всех упражнений.
- Ссылки на профессиональные средства автоматизированного проектирования (САПР) компании Altera®.
- Рисунки из текста в форматах JPG и PPT.

Также на данном веб-сайте приведена дополнительная информация по использованию инструментов Altera, Raspberry Pi и MDK-ARM в вашем курсе. Там же находится информация о материалах для лабораторных работ.

Как использовать программный инструментарий в учебном курсе

Altera Quartus II

Quartus II Web Edition является бесплатной версией профессиональной САПР Quartus™ II, предназначенной для разработки на ПЛИС (FPGA). Она позволяет сту-

дентам проектировать цифровые устройства в виде принципиальных схем или на языках SystemVerilog и VHDL. После создания схемы или кода устройства студенты могут симулировать их поведение с использованием САПР ModelSim™ – Altera Starter Edition, которая доступна вместе с Altera Quartus II Web Edition. Quartus II Web Edition также включает в себя встроенный логический синтезатор, поддерживающий как SystemVerilog, так и VHDL.

Разница между Web Edition и Subscription Edition заключается в том, что Web Edition поддерживает только подмножество наиболее распространенных ПЛИС производства Altera. Разница между ModelSim – Altera Starter Edition и коммерческими версиями ModelSim заключается в том, что Starter Edition искусственно снижает производительность симуляции для проектов, содержащих больше 10 тысяч строк HDL-кода.

Kiel ARM Microcontroller Development Kit (MDK-ARM)

Kiel MDK-ARM – это инструмент для разработки кода для процессора ARM. Он доступен для скачивания онлайн бесплатно. MDK-ARM включает в себя коммерческий компилятор Си для ARM и эмулятор, который позволяет студентам писать программы на языке Си и ассемблере, компилировать их и затем эмулировать.

Лабораторные работы

Веб-сайт книги содержит ссылки на ряд лабораторных работ, которые охватывают все темы, начиная от проектирования цифровых систем и заканчивая архитектурой компьютера. Из лабораторных работ студенты узнают, как использовать САПР Quartus II для описания своих проектов, их симулирования, синтеза и реализации. Лабораторные работы также включают темы по программированию на языке Си и языке ассемблера с использованием средств разработки MDK-ARM и Raspberry Pi.

После синтеза студенты могут реализовать свои проекты, используя обучающие платы Altera DE2 (или DE2-115). Эта мощная и относительно недорогая плата доступна для заказа на веб-сайте www.altera.com. Плата содержит микросхему ПЛИС (FPGA), которую можно сконфигурировать для реализации студенческих проектов. Мы предоставляем лабораторные работы, которые описывают, как реализовать различные блоки на плате DE2 с использованием Quartus II Web Edition.

Для выполнения лабораторных работ студенты должны будут загрузить и установить САПР Altera Quartus II Web Edition и либо MDK-ARM, либо инструменты Raspberry Pi. Преподаватели могут также установить эти САПР в учебных лабораториях. Лабораторные работы включают инструкции по разработке проектов на плате DE2. Этап практической реализации проекта на плате можно пропустить, однако мы считаем, что он имеет большое значение для получения практических навыков.

Мы протестировали лабораторные работы на ОС Windows, но инструменты доступны и для ОС Linux.

Опечатки

Все опытные программисты знают, что любая сложная программа непременно содержит ошибки. Так же происходит и с книгами. Мы старались выявить и исправить все ошибки и опечатки в этой книге. Тем не менее некоторые ошибки могли остаться. Список найденных ошибок будет опубликован на веб-сайте книги.

Пожалуйста, присылайте найденные ошибки по адресу ddcabugs@gmail.com[1]. Первый человек, который сообщит об ошибке и предоставит исправление, которое мы используем в будущем издании, будет вознагражден премией в $1!

Признательность за поддержку

Мы ценим тяжелую работу Нэйта МакФаддена (Nate McFadden), Джо Хэйтона (Joe Hayton), Пунитавати Говиндараджана (Punithavathy Govindaradjane) и остальных членов команды издательства Morgan Kaufmann, которые сделали возможным появление этой книги. Нам нравится художественная работа Дуэйна Бибби (Duane Bibby), чьи иллюстрации вдохнули жизнь в главы.

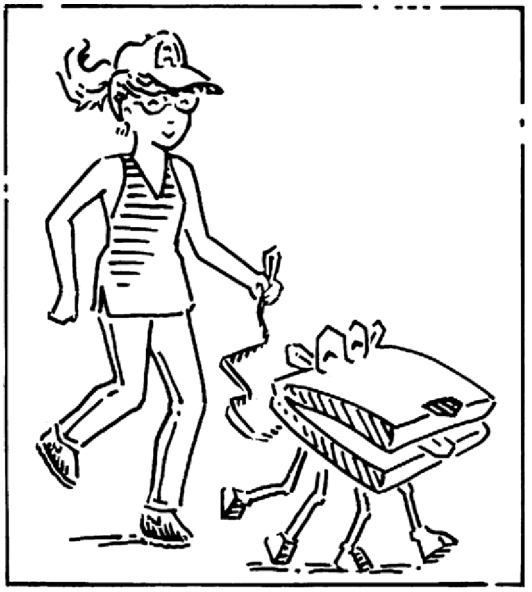
Мы хотели бы поблагодарить Мэтью Уоткинса (Matthew Watkins), который помог написать раздел о гетерогенных многопроцессорных системах в **главе 2**. Мы очень ценим работу Джошуа Васкеза (Joshua Vasquez), разработавшего код для Raspberry Pi из **главы 4**. Мы также благодарны Джозефу Спьюту (Josef Spjut) и Руйе Вангу (Ruye Wang), протестировавшим материал в классе.

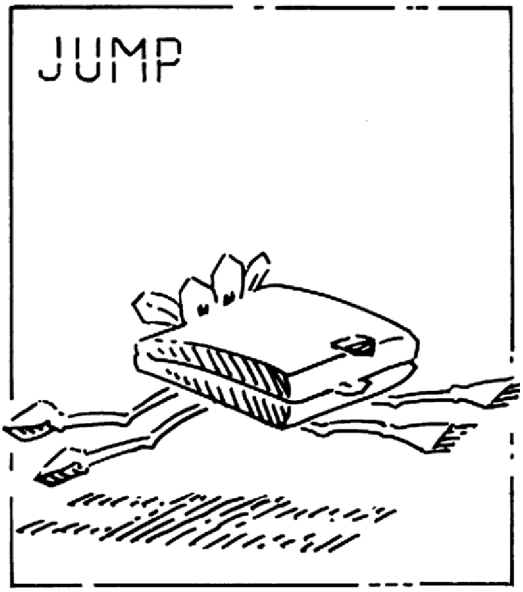
Огромный вклад в улучшение качества книги внесли многочисленные рецензенты, среди которых Бойанг Ванг (Boyang Wang), Джон Барр (John Barr), Джэк Брайнер (Jack V. Briner), Эндрю Браун (Andrew C. Brown), Карл Баумгартнер (Carl Baumgaertner), Утку Дирил (A. Utku Diril), Джим Френцель (Jim Frenzel), Джаэха Ким (Jaeha Kim), Филлип Кинг (Phillip King), Джеймс Пинтер-Лаки (James Pinter-Lucke), Амир Рот (Amir Roth), Джерри Ши (Z. Jerry Shi), Джеймс Стайн (James E. Stine), Люк Тэсье (Luke Teyssier), Пейуй Чжао (Peiyi Zhao), Зак Доддс (Zach Dodds), Натаниэл Гай (Nathaniel Guy), Эшвин Кришна (Aswin Krishna), Волней Педрони (Volnei Pedroni), Карл Ванг (Karl Wang), Рикардо Яасински (Ricardo Jasinski), Джозеф Спьют (Josef Spjut), Йорген Лиен (Jörgen Lien), Самеер Шарма (Sameer Sharma), Джон Нестор (John Nestor), Сайев Манзоор (Syed Manzoor), Джеймс Хо (James Hoe), Сриниваза Вемуру (Srinivasa Vemuru), Джозеф Хасс (K. Joseph Hass), Джафанта Херат (Jayantha Herath), Роберт Муллинс (Robert Mullins), Бруно Куоитин (Bruno Quoitin), Субраманьям Ганеша (Subramaniam Ganesan), Браден Филлипс (Braden Phillips), Джон Оливер (John Oliver), Яхсвант Малайа (Yahswant K. Malaiya), Мохаммад Аведх (Mohammad Awedh), Захари Курмас (Zachary Kurmas), Дональд Хунг (Donald Hung) и анонимный рецензент. Мы очень признательны Кхаледу Бенкриду (Khaled Benkrid) и его коллегам из ARM за тщательное рецензирование материалов, связанных с ARM.

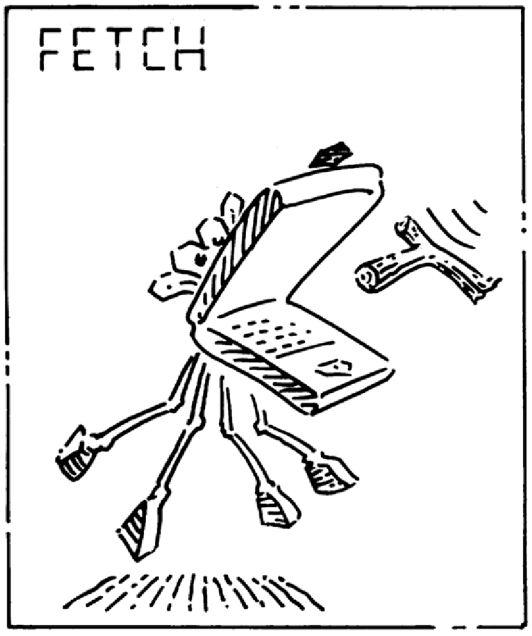
[1] Это относится лишь к ошибкам в исходном англоязычном издании. Ошибки в переводе присылайте на адрес dmkpress@gmail.com (хотя в этом случае издательство доллар не выдает). – *Прим. перевод.*

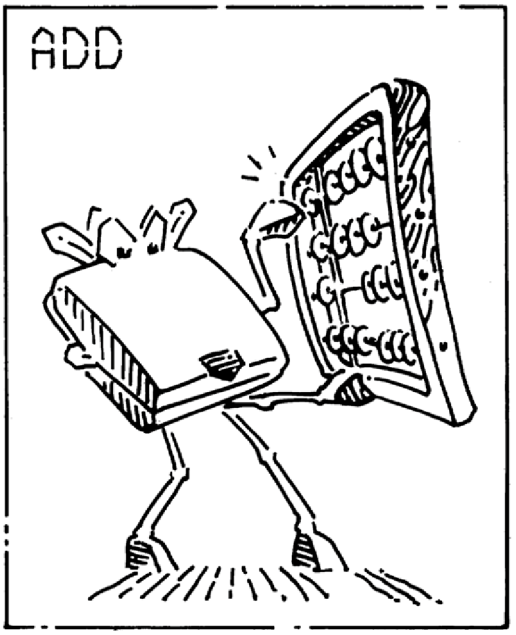
Мы также признательны нашим студентам из колледжа Harvey Mudd и UNLV, которые дали полезные отзывы на черновики этого учебника. Отдельного упоминания заслуживают Клинтон Барнс (Clinton Barnes), Мэтт Вайнер (Matt Weiner), Карл Уолш (Carl Walsh), Эндрю Картер (Andrew Carter), Кейси Шиллинг (Casey Schilling), Элис Клифтон(Alice Clifton), Крис Эйкон (Chris Acon) и Стивен Браунер (Stephen Brawner).

И, конечно же, мы благодарим наши семьи за их любовь и поддержку.



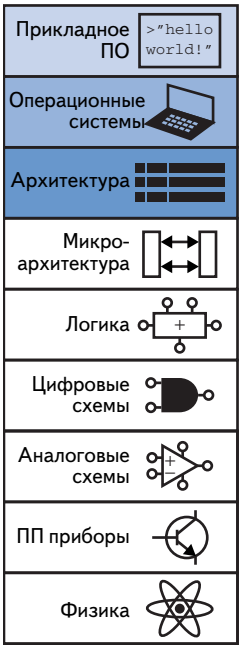

JUMP


FETCH


ADD

Глава 1

Архитектура



1.1. Введение

В предыдущих главах мы познакомились с принципами разработки цифровых устройств и основными строительными блоками. В этой главе мы поднимемся на несколько уровней абстракции и определим *архитектуру* компьютера. Архитектура – это то, как видит компьютер программист. Она определяется набором команд (языком) и местом нахождения операндов (регистры или память). Существует множество разных архитектур, например: x86, MIPS, SPARC и PowerPC.

Чтобы понять архитектуру любого компьютера, нужно в первую очередь выучить его язык. Слова в языке компьютера называются *командами*, а словарный запас компьютера – *набором*, или *системой, команд*.

Даже сложные приложения – редакторы текста и электронные таблицы – в конечном итоге состоят из последовательности таких простых команд, как сложение, вычитание и переход. Команда компьютера определяет операцию, которую нужно исполнить, и ее операнды. Операнды могут находиться в памяти, в регистрах или внутри самой команды.

Говоря об «архитектуре ARM», мы имеем в виду ARM версии 4 (ARMv4) и составляющий ее базовый набор команд. В разделе 1.7 дан краткий обзор возможностей, появившихся в версиях 5–8 этой архитектуры. В сети можно найти «Справочное руководство по архитектуре ARM» (ARM Architecture Reference Manual (ARM)), в котором содержится полное определение архитектуры.

Аппаратное обеспечение компьютера «понимает» только нули и единицы, поэтому команды закодированы двоичными числами в формате, который называется *машинным языком*. Так же как мы используем буквы и прочие символы на письме для представления речи, компьютеры используют двоичные числа, чтобы кодировать машинный язык. В архитектуре ARM каждая команда представлена 32-разрядным словом. Микропроцессоры – это цифровые системы, которые читают и выполняют команды машинного языка. Но для людей чтение компьютерных программ на машинном языке представляется нудным и утомительным занятием, поэтому мы предпочитаем представлять команды в символическом формате, который называется *языком ассемблера*.

Наборы команд в различных архитектурах можно сравнить с диалектами естественных языков. Почти во всех архитектурах определены такие базовые команды, как сложение, вычитание и переход, работающие с ячейками памяти или регистрами. Изучив один набор команд, понять другие уже довольно легко.

Архитектура компьютера не определяет структуру аппаратного обеспечения, которое ее реализует. Зачастую существуют разные аппаратные реализации одной и той же архитектуры. Например, компании Intel и Advanced Micro Devices (AMD) производят разные микропроцессоры, построенные на базе архитектуры x86. Все они могут выполнять одни и те же программы, но в их основе лежит разное аппаратное обеспечение, поэтому эти процессоры характеризуются различным соотношением производительности, цены и энергопотребления. Одни микропроцессоры оптимизированы для работы в высокопроизводительных серверах, другие рассчитаны на продление срока службы батареи в ноутбуках. Конкретное сочетание регистров, памяти, АЛУ и других строительных блоков, из которых состоит микропроцессор, называют *микроархитектурой*, она будет рассмотрена в **главе 2**. Нередко для одной и той же архитектуры существует несколько разных микроархитектур.

В этой книге мы представим архитектуру ARM. Впервые она была разработана в 1980-х годах компанией Acorn Computer Group, от которой затем отпочковалась компания Advanced RISC Machines Ltd., известная ныне под названием ARM. Ежегодно продается свыше 10 млрд процессоров ARM. Почти все сотовые телефоны и планшеты оснащены несколькими процессорами ARM. Эта архитектура встречается повсеместно: в автоматах для игры в пинбол, в фотокамерах, в роботах, в серверах, смонтированных в стойке. Компания ARM необычна тем, что продает не сами процессоры, а лицензии, разрешающие другим компаниям самостоятельно производить процессоры, которые зачастую являются составной частью более крупной системы на кристалле. Например,

процессоры ARM изготавливают компании Samsung, Altera, Apple и Qualcomm – они построены на базе либо микроархитектуры, приобретенной у ARM, либо собственной микроархитектуры, разработанной по лицензии ARM. Мы остановились на архитектуре ARM, потому что она занимает лидирующие коммерческие позиции и в то же время является чистой и почти свободной от странностей. Начнем с описания команд языка ассемблера, мест нахождения операндов и таких общеупотребительных программных конструкций, как ветвления, циклы, операции с массивами и вызовы функций. Затем опишем, как язык ассемблера транслируется в машинный язык, и продемонстрируем, как программа загружается в память и выполняется.

В этой главе мы покажем, как архитектура ARM формировалась на основе четырех принципов, сформулированных Дэвидом Паттерсоном и Джоном Хеннесси в книге «Computer Organization and Design»:

1) единообразие способствует простоте;
2) типичный сценарий должен быть быстрым;
3) чем меньше, тем быстрее;
4) хороший проект требует хороших компромиссов.

1.2. Язык ассемблера

Язык ассемблера – это удобное для восприятия человеком представление внутреннего языка компьютера. Каждая команда языка ассемблера задает операцию, которую необходимо выполнить, а также операнды, над которыми производится эта операция. Далее мы познакомимся с простыми арифметическими командами и покажем, как они записываются на языке ассемблера. Затем определим операнды для команд ARM: регистры, ячейки памяти и константы.

В этой главе предполагается знакомство с каким-нибудь высокоуровневым языком программирования, например C, C++ или Java (эти языки практически равнозначны для большинства примеров в данной главе, но мы будем использовать C). В **приложении C** (книга 1) приведено введение в язык C для тех, у кого мало или совсем нет опыта программирования.

В этой главе для компиляции, ассемблирования и эмуляции ассемблерного кода мы пользуемся комплектом средств разработки ARM Microcontroller Development Kit (MDK-ARM) компании Keil. MDK-ARM – свободный инструмент разработки, в состав которого входит полный компилятор для ARM. В лабораторных работах на сопроводительном сайте для этой книги (см. **предисловие**) описано, как установить и использовать этот инструмент для написания, компиляции, эмуляции и отладки программ на C и ассемблере.

1.2.1. Команды

Наиболее частая операция, выполняемая компьютером, – сложение. В **примере кода 1.1** показан код, который складывает переменные `b` и `c` и записывает результат в переменную `a`. Слева показан вариант на языке высокого уровня (C, C++ или Java), а справа – на языке ассемблера

ARM. Обратите внимание, что предложения языка C оканчиваются точкой с запятой.

Пример кода 1.1. СЛОЖЕНИЕ

Код на языке высокого уровня	Код на языке ассемблера ARM
`a = b + c;`	`ADD a, b, c`

Слово «мнемоника» происходит от греческого слова *μνημονικός*. Мнемоники языка ассемблера запомнить проще, чем наборы нулей и единиц машинного языка, представляющих ту же операцию.

Первая часть команды ассемблера, `ADD`, называется *мнемоникой* и определяет, какую операцию нужно выполнить. Операция осуществляется над *операндами-источниками* `b` и `c`, а результат записывается в *операнд-приемник* `a`.

Пример кода 1.2. ВЫЧИТАНИЕ

Код на языке высокого уровня	Код на языке ассемблера ARM
`a = b - c;`	`SUB a, b, c`

В **примере кода 1.2** демонстрируется, что вычитание похоже на сложение. Формат команды такой же, как у команды `ADD`, только операция называется `SUB`. Единообразный формат команд — пример применения первого принципа хорошего проектирования:

> **Первый принцип хорошего проектирования:** единообразие способствует простоте.

Команды с одинаковым количеством операндов — в данном случае с двумя операндами-источниками и одним операндом-приемником — проще закодировать и выполнить на аппаратном уровне. Более сложный высокоуровневый код транслируется в несколько команд ARM, как показано в **примере кода 1.3**.

Пример кода 1.3. БОЛЕЕ СЛОЖНЫЙ КОД

Код на языке высокого уровня	Код на языке ассемблера ARM

`a = b + c - d; // однострочный комментарий`
`/* многострочный`
`комментарий */`
	`ADD t, b, c ; t = b + c`
	`SUB a, t, d ; a = t - d`

В примерах на языках высокого уровня однострочные комментарии начинаются с символов `//` и продолжаются до конца строки. Много-

строчные комментарии начинаются с `/*` и завершаются `*/`. В языке ассемблера ARM допустимы только однострочные комментарии. Они начинаются знаком `;` и продолжаются до конца строки. В программе на языке ассемблера в **примере 1.3** используется временная переменная `t` для хранения промежуточного результата. Использование нескольких команд ассемблера для выполнения более сложных операций – иллюстрация второго принципа хорошего проектирования компьютерной архитектуры:

> **Второй принцип хорошего проектирования:** типичный сценарий должен быть быстрым.

При использовании системы команд ARM типичный сценарий оказывается быстрым, потому что включает только простые, часто используемые команды. Количество команд специально оставляют небольшим, чтобы аппаратура для их поддержки была простой и быстрой. Более сложные операции, используемые реже, представляются в виде последовательности нескольких простых команд. По этой причине ARM относится к компьютерным архитектурам с *сокращенным набором команд* (reduced instruction set computer, RISC). Архитектуры с большим количеством сложных инструкций, такие как архитектура x86 корпорации Intel, называются *компьютерами со сложным набором команд* (complex instruction set computer, CISC). Например, в x86 определена команда «перемещение строки», которая копирует строку (последовательность символов) из одного участка памяти в другой. Такая операция требует большого количества, иногда до нескольких сотен, простых команд на RISC-машине. С другой стороны, реализация сложных команд в архитектуре CISC требует дополнительного оборудования и увеличивает накладные расходы, что приводит к замедлению простых команд.

В архитектуре RISC используется небольшое множество различных команд, что уменьшает сложность аппаратного обеспечения и размер команды. Например, код операции в системе, состоящей из 64 простых команд, требует $\log_2 64 = 6$ бит, а в системе из 256 сложных команд потребуется уже $\log_2 256 = 8$ бит. В CISC-машинах наличие сложных команд, даже очень редко используемых, увеличивает накладные расходы на выполнение всех команд, включая и самые простые.

1.2.2. Операнды: регистры, память и константы

Команда работает с операндами. В **примере кода 1.1** переменные `a`, `b` и `c` являются операндами. Но компьютеры оперируют нулями и единицами, а не именами переменных. Команда должна знать, откуда она сможет брать двоичные данные. Операнды могут находиться в регистрах или в памяти, а также могут быть *константами*, записанными в теле самой