

Оглавление

Об авторе.....	13
1. Введение.....	14
Безграничные возможности.....	15
Начинаем экспериментировать	16
Компилятор GNU C.....	16
Учимся на примерах.....	17
Что вы узнаете?.....	17
Совместимость четвертого издания книги	18
ОС Raspberry Pi.....	19
А что насчет 64-разрядной системы?.....	20
Клавиатурные вычисления.....	20
Значимость ARM	20
Raspberry Pi сквозь века	21
Вычислительные модули	23
Используемые обозначения	24
Центр истории вычислительной техники	24
Веб-сайт и бесплатные книги	25
Благодарности.....	26
2. Начало	27
Числа со смыслом.....	27
Команды ARM	28
Процесс преобразования.....	29
А зачем вообще машинный код?	30
Языковые уровни.....	30
На орбиту!	31
RISC и наборы команд	32
Структура ассемблера	32
Ошибки на пути	33
Кросс-компиляторы.....	33
Чипы Raspberry Pi ARM.....	33
3. Проба пера	35
Командная строка3. Проба пера.....	35
Создание исходного файла	36
Написанное — исполнить!.....	40
Ошибки ассемблера.....	40
Компоненты	41
А если нет метки <code>_start</code> ?	44
Связывание файлов.....	44
Приблиаемся.....	46

Пара слов о комментариях.....	47
Редактор Geany Programmer's Editor.....	48
4. О битах в RISC-машинах.....	49
Преобразование двоичных чисел в десятичные.....	50
Преобразование десятичных чисел в двоичные.....	51
Преобразование двоичного числа в шестнадцатеричное.....	51
Преобразуем шестнадцатеричные числа в десятичные и обратно.....	53
Двоичное сложение.....	53
Вычитание.....	54
Дополнительный код.....	56
Когда двоичные числа не складываются.....	57
Стандартный калькулятор.....	58
5. Соглашения ARM.....	59
Длина слов.....	59
Доступ к памяти по байтам и словам.....	60
Регистры.....	61
Регистр <i>R15</i> : программный счетчик.....	63
Регистр состояния текущей программы.....	63
Биты и флаги.....	64
Установка флагов.....	64
Суффикс <i>S</i>	65
<i>R14</i> : регистр ссылок.....	66
<i>R13</i> : указатель стека.....	66
6. Обработка данных.....	67
Команды сложения.....	68
Вычитание.....	71
Умножение.....	72
Теперь о делении.....	74
Команды перемещения.....	75
Команды сравнения.....	76
Сортировка чисел.....	77
7. Входы и выходы.....	78
Команды <i>SWI</i> и <i>SVC</i>	78
Вывод на экран.....	79
Чтение с клавиатуры.....	81
Регистр <i>eax</i> и прочие.....	83
Программа Make.....	83
8. Логические операции.....	86
Логическое И (<i>AND</i>).....	86
Логическое ИЛИ (<i>OR</i>).....	87
Исключающее ИЛИ (<i>EOR</i>).....	87
Команды логических операций.....	88
Команда <i>ORR</i> для преобразования регистра символов.....	89
Очистка бита командой <i>BIC</i>	90
Проверка флагов.....	91
Регистры системных вызовов.....	94

9. Условное выполнение	95
Коды состояния с одним флагом	97
<i>EQ</i> : равно	97
<i>NE</i> : не равно	97
<i>VS</i> : переполнение	98
<i>VC</i> : нет переполнения	98
<i>MI</i> : знак «минус»	98
<i>PL</i> : знак «плюс»	98
<i>CS</i> : имеется перенос (<i>HS</i> : беззнаковое больше или равно)	99
<i>CC</i> : нет переноса (<i>LO</i> : беззнаковое меньше)	99
<i>AL</i> : безусловное исполнение	100
<i>NV</i> : безусловное неисполнение	100
Коды, проверяющие несколько флагов	100
<i>HI</i> : беззнаковое больше	100
<i>LS</i> : беззнаковое меньше или равно	101
<i>GE</i> : знаковое больше или равно	101
<i>LT</i> : знаковое меньше	101
<i>GT</i> : знаковое больше	101
<i>LE</i> : знаковое меньше или равно	102
Добавление суффикса <i>S</i>	102
 10. Ветви и сравнения	 103
Команды ветвления	103
Регистр ссылок	104
Использование команд сравнения	104
Применяем дальновидное мышление	105
Эффективное использование условных операторов	106
Обмен ветвей	107
 11. Сдвиги и вращения	 108
Логические сдвиги	108
Логический сдвиг вправо	110
Арифметический сдвиг вправо	110
Вращение	111
Расширенное вращение	112
Использование сдвигов и вращений	112
Прямой постоянный диапазон	113
Движение вверх	115
 12. Умные числа	 116
Длинное умножение	116
Умножение с накоплением	118
Деление и остаток	119
Умное умножение	120
Это только начало	121
 13. Программный счетчик <i>R15</i>	 122
Конвейерная обработка	123
Расчет ветвей	124

14. Отладка с использованием GDB	126
Когда все зависло.....	126
Сборка с GDB	127
Дизассемблер	130
Точки останова	132
Дамп памяти.....	136
Сокращения.....	137
Параметры сборки GDB.....	137
15. Передача данных	139
Директива <i>ADR</i>	139
Косвенная адресация	141
Команды <i>ADR</i> и <i>LDR</i>	143
Предварительно индексированная адресация	143
Доступ к байтам памяти	144
Обратная запись адреса	146
Постиндексированная адресация	146
Байтовые условия	148
Относительная адресация через регистр <i>PC</i>	148
16. Передача блока	150
Обратная запись	152
Процедура копирования блока	153
17. Стеки	155
Тянитолкай ;-)	155
Рост стека	157
Применение стеков	159
Работа в фрейме	160
Указатель фрейма	160
18. Директивы и макросы.....	162
Директивы хранения данных	162
Выравнивание данных.....	164
Макросы	165
Включение макросов	168
19. Работа с файлами	171
Права доступа к файлам	176
20. Использование библиотеки <i>libc</i>	179
Использование функций языка C в ассемблере	179
Структура файла исходного кода	180
Исследование исполняемого файла	182
Ввод чисел с помощью функции <i>scanf</i>	184
Вывод информации	186
21. Пишем функции	188
Стандарты функций.....	188
Использование регистров	190

Больше трех.....	190
Сохранение ссылок и флагов.....	192
Надежные процедуры вывода.....	193
Пузырьковая сортировка.....	195
22. Дизассемблирование программ на C.....	198
GCC — он как швейцарский нож.....	198
Простой фреймворк C.....	199
Создание файла ассемблера.....	200
Строительные блоки.....	202
Пример функции <i>printf</i>	204
Переменные указателя фрейма.....	205
Дизассемблирование системных вызовов.....	206
23. Функции GPIO.....	207
Отображение памяти.....	207
Контроллер GPIO.....	209
Вводы и выходы GPIO.....	211
Сборка кода.....	217
Другие функции GPIO.....	222
Описание контактов GPIO.....	222
24. Числа с плавающей точкой.....	225
Архитектура VFP.....	225
Регистровый файл.....	227
Управление и вывод на экран.....	229
Сборка и отладка на VFP с помощью GDB.....	231
Загрузка, хранение и перемещение.....	233
Преобразование точности.....	235
Векторная арифметика.....	236
25. Регистр управления VFP.....	237
Условное исполнение.....	238
Скалярные и векторные операции.....	240
Какой тип оператора?.....	242
Параметры <i>LEN</i> и <i>STRIDE</i>	243
26. Сопроцессор Neon.....	247
Ассемблер Neon.....	249
Команды и типы данных Neon.....	251
Режимы адресации.....	253
Параметр <i>Stride</i> команд <i>VLD</i> и <i>VST</i>	253
Загрузка в прочих форматах.....	256
Neon Intrinsic.....	256
Массивы Neon.....	257
Правильный порядок.....	261
Матричная математика.....	262
Матричное умножение.....	265
Пример использования макроса.....	270

27. Код Thumb	272
Различия	272
Пишем на Thumb	274
Доступ к старшим регистрам.....	278
Операторы стека	279
Одно- и многорегистровые команды	279
Функции в Thumb	279
Команды ARMv7 Thumb.....	280
28. Единый язык	281
Изменения Thumb	282
Новые команды A32	283
Сравнение по нулю.....	284
Сборка UAL.....	284
29. Обработка исключений.....	287
Режимы работы	287
Векторы	288
Настройка регистров	290
Обработка исключений	292
Команды <i>MRS</i> и <i>MSR</i>	293
Что происходит при возникновении прерывания?	294
Решения о прерываниях	295
Возврат из прерываний	295
Пишем процедуры прерывания	296
30. System-on-Chip	297
Микросхема и набор команд ARM	298
Сопроцессоры	299
Конвейер.....	299
Память и кэши.....	300
GPU	301
Обзор ARMv8.....	301
64-разрядная ОС Raspberry Pi.....	302
А что в итоге?.....	302
Принцип Архимеда.....	302
Приложение 1. Коды символов ASCII	304
Приложение 2. Набор команд ARM.....	306
Команды сравнения и проверки	307
Команды ветвления	307
Арифметические команды	308
Логические команды	308
Команды перемещения данных	309
Приложение 3. Системные вызовы ROS.....	310
Приложение 4. Описание электронного архива.....	316
Предметный указатель	317

СПИСОК ПРОГРАММ

Программа 3.1. Простой исходный файл	39
Программа 3.2. Часть 1 исходного файла	44
Программа 3.3. Часть 2 исходного файла	45
Программа 6.1. Простое 32-битное сложение	69
Программа 6.2. 64-битное сложение	70
Программа 6.3. 32-битное умножение	73
Программа 6.4. Использование умножения с накоплением MLA	73
Программа 6.5. Деление командой SDIV	74
Программа 7.1. Системный вызов 4 для вывода строки на экран	80
Программа 7.2. Системный вызов 3 и чтение с клавиатуры	81
Программа 7.3. Автоматизация сборки и привязки с помощью Make	84
Программа 8.1. Преобразование регистра символов	89
Программа 8.2. Вывод двоичного числа	91
Программа 10.1. Условное выполнение позволяет сократить программу	107
Программа 12.1. Долгое и нудное умножение	117
Программа 12.2. Долгое нудное деление	119
Программа 14.1. Гибкий make-файл для отладки кода	138
Программа 15.1. Использование директивы ADR	140
Программа 15.2. Использование предварительно индексированной косвенной адресации	145
Программа 15.3. Использование постиндексированной адресации	147
Программа 16.1. Перемещение блоков памяти	153
Программа 18.1. Использование директив <code>.byte</code> и <code>.equ</code>	163
Программа 18.2. Реализация простого макроса	166
Программа 18.3. Многократный вызов макроса	167
Программа 18.4. Файл макроса <code>AddMult</code>	168
Программа 18.5. Проверка подключения макроса	169
Программа 19.1. Создание файлов и доступ к ним	172
Программа 20.1. Структура исходного файла GCC	181
Программа 20.2. Передача параметров в <code>printf</code>	183
Программа 20.3. Чтение и преобразование числа с помощью функции <code>scanf</code>	185
Программа 20.4. Объединение <code>scanf</code> и <code>printf</code>	187
Программа 21.1. Передача значений функции через стек	191
Программа 21.2. Вывод вектора слов	193
Программа 21.3. Проверка <code>printw</code>	194
Программа 21.4. Процедура пузырьковой сортировки	195
Программа 21.5. Тест пузырьковой сортировки	196
Программа 22.1. Простая программа на языке C для вывода символа	200
Программа 22.2. Окончательный вариант кода	203
Программа 22.3. Код программы <code>hello world</code> на C	204
Программа 22.4. Код для работы с функцией <code>scanf</code>	205
Программа 23.1. Файловый доступ к памяти GPIO	213
Программа 24.1. Вывод значения с плавающей точкой с помощью функции <code>printf</code>	229
Программа 24.2. Вывод двух или более значений с плавающей точкой	230
Программа 25.1. Условное выполнение в VFP	240
Программа 25.2. Использование битов <code>LEN</code> и <code>STRIDE</code> для сложения векторов	245
Программа 26.1. Простая проверка Neon	249
Программа 26.2. Поворот 2D-матрицы на 90 градусов (по часовой стрелке)	258
Программа 26.3. Сложение двух матриц размером 4×4	263
Программа 26.4. Матричное умножение чисел с одинарной точностью	266
Программа 26.5. Умножение матриц, но с макросами	270
Программа 27.1. Вызов режима <code>Thumb</code> и запуск кода <code>Thumb</code>	275
Программа 27.2. Использование внешних функций путем взаимодействия кода	277
Программа 28.1. Единый язык ассемблера	285

*Посвящается всем медицинским работникам:
медсестрам, врачам и другому персоналу,
осуществляющим уход за пациентами во всем мире.*

*Тем, кто выложился по полной в эти годы пандемии.
Память о вас останется в наших сердцах навсегда.*

Спасибо!

Об авторе



Брюс Смит приобрел свой первый компьютер Acorn Atom в 1980 году. Увлечение компьютерами мгновенно затянуло его, и он стал постоянным автором крупных тематических изданий, включая *Computing Today* и *Personal Computer World*. С появлением компьютера BBC Micro его работа перешла от журналов к книгам, и его книга 1982 года «*Interfacing Projects for BBC Micro*» стала классикой своего времени. В ней он рассказал простым пользователям домашних компьютеров, как подключиться к внешнему миру. Он был одним из первых, кто писал о чипе ARM, когда они были пред-

ставлены на Acorn Archimedes в 1987 году.

Брюс рассказывал обо всех аспектах использования компьютера. Его дружелюбный и понятный стиль письма вдохновил одного рецензента на вот такой отзыв: «Это первая книга о компьютерах, которую я читал для удовольствия, лежа в постели, а не ради того, чтобы поскорее заснуть!» Книги Брюса переведены на многие языки и продаются по всему миру. Его издавали в BBC Books, Collins, Virgin Books и Rough Guides.

Брюс родился в лондонском Ист-Энде, а сейчас живет в Сиднее, Новый Южный Уэльс.

Его сайт в Интернете: www.brucesmith.info.

Из отзывов с пятью звездами на Amazon на предыдущие издания этой книги:

«Отличная книга. Настоятельно рекомендую ее всем, кто хочет работать с ARM-асемблером».

«Эта книга подойдет не только начинающим, но будет также полезна программистам среднего уровня, и в ней затронуты некоторые более сложные темы. На мой взгляд, эта книга заслуживает оценки 10/10 и обязательна к прочтению всем, кто планирует начать или возвращается к программированию на асемблере на процессоре ARM v6 (Raspberry Pi). Книга сэкономила мне много времени, поскольку вся необходимая информация теперь собрана в одном месте. Спасибо Брюсу за такое сокровище!»

«Это отличная книга для начинающих программистов, которые хотят глубже изучить Raspberry Pi и по-настоящему понять, как работает компьютер. Приведены отличные и 00подробные примеры».

«Смит не только излагает все нужные материалы и инструкции на языке асемблера для процессора ARM Raspberry Pi, но также дает четкие практические указания о том, как заставить все это работать... Не важно, хотите ли вы просто побаловаться с Raspberry Pi, или рассматриваете Pi как ступеньку к чему-то большему — книга Брюса Смита определенно должна стоять на вашей полке».

1. Введение

Я ничуть не удивился, когда в начале 2013 года появилась новость о том, что продажи контроллеров Raspberry Pi (рис. 1.1) достигли 1 млн штук. А само это число заставило меня вернуться на несколько лет (даже десятилетий) назад и вспомнить аналогичную новость о платах BBC Micro. Обе платформы производятся на предприятии, расположенном в Кембридже в Великобритании, и общего у них гораздо больше, чем вы можете себе представить.

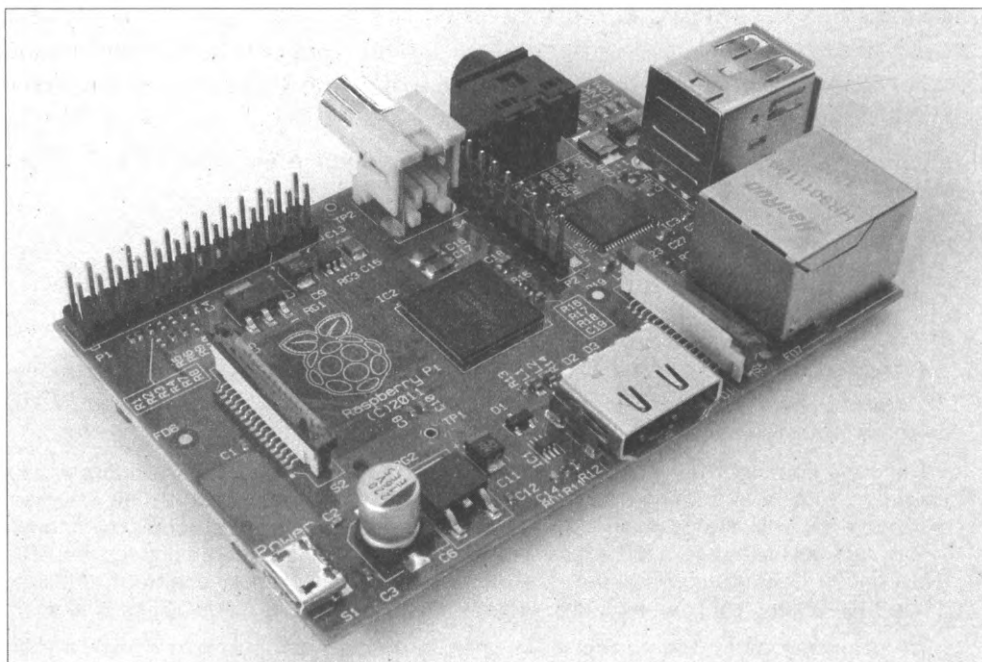


Рис. 1.1. Плата Raspberry Pi Model B

Обе системы покорили мое сердце и сердца миллионов других людей. На первую я в свое время спускал все свои сбережения, а вот Raspberry Pi совершенно не ударила по моему кошельку. Но огромная разница в цене была почти единственным различием, а прочие черты оказались весьма схожи. Одна из самых важных черт —

возможность запускать на них широкий спектр программного обеспечения и образовательных инструментов, а также легкость в подключении внешних устройств и управлении внешним миром. Я подозреваю, что у большинства владельцев Raspberry Pi лежит не один, а как минимум парочка контроллеров.

Впервые после BBC Micro на рынке появилась дешевая и невероятно богатая возможностями система, которую может использовать практически каждый. В то же время на рынке доминировали платформы PC и Mac и появились другие устройства, более ориентированные на игры. Но ничего из этого не сказалось на тех, чьим домашним хобби была работа с техникой. У профессионалов имелось множество платформ для разработки, с которыми можно было экспериментировать, но у домашних любителей компьютеров оставались проблемы с доступностью таких платформ. А вот Raspberry Pi со своей ценой, соизмеримой с парочкой DVD, изменила все.

Запуск Raspberry Pi Zero в ноябре 2015 года вывел экономические преимущества платформы на новый уровень. Полноценный компьютер на базе ARM можно получить всего за 5 долларов. Это ж с ума сойти! И, чтобы окончательно добить фанатов, к каждому экземпляру MagPi (официального журнала Raspberry Pi) за декабрь 2015 года прилагался контроллер в подарок. Нельзя ли теперь считать Raspberry Pi первым в мире полноценным компактным компьютером?

На момент подготовки этой книги (декабрь 2020 г.) продажи плат Raspberry превысили 30 млн штук. Эта отметка, как сообщается, была достигнута к началу 2020 года, когда многие компании начали использовать Raspberry Pi в своей инфраструктуре и разработках. Компания Oracle назвала кластер из 1060 плат Raspberry Pi самым большим в мире суперкомпьютером Pi с 4240 ядрами. Jet Propulsion Laboratory (JPL) используют эти платы в своей марсианской миссии.

Безграничные возможности

Одна из причин появления Raspberry Pi заключалась в том, что у людей имелась необходимость в дешевой системе программирования, пределы возможностей которой ограничивались бы только воображением пользователя. Я надеюсь, что эта книга поможет многим из читателей осуществить свою мечту. А это вполне реально. Весьма вероятно, что чтение этих страниц станет вашим первым шагом к полезному и познавательному времяпрепровождению. И, кто знает, может быть, вы станете частью нового поколения компьютерных программистов, творящих на пределе возможного.

Цель этой книги — дать читателю лучшее понимание того, как устроена работа компьютера на уровне ниже высокоуровневых языков программирования, таких как C или Python. Глубже разобравшись с работой компьютера, вы сможете гораздо более продуктивно писать программное обеспечение на языках более высокого уровня. Именно обучение программированию на ассемблере поможет вам достичь этой цели.

Эта книга представляет собой начальное руководство по написанию кода на ассемблере для Raspberry Pi и, в частности, с использованием ОС Raspberry Pi

(операционной системы, ранее называвшейся Raspbian). Язык ассемблера генерирует машинный код, который можно запускать прямо на вашем компьютере.

Сам я сначала учился программировать на ассемблере на машинах, созданных еще на ранних этапах разработки Acorn, а позднее стал свидетелем развития чипов ARM в системах Archimedes и BBC Micro Second-Processor. В конечном итоге именно тот ассемблер стал предшественником того, что сегодня используется в Raspberry Pi. Этим я хочу лишь подчеркнуть, что все, чему вы здесь научитесь, будет полезно и актуально в течение долгих лет. Однозначно, стоящая инвестиция.

Начинаем экспериментировать

Эта книга адресована не совсем уж полным новичкам, однако вам не потребуется иметь опыт работы с языком ассемблера или машинным кодом, чтобы понять ее и начать экспериментировать. Тем не менее вам пригодится любой опыт в программировании, и любой известный вам структурированный язык поможет понять базовые концепции ассемблера.

Поскольку это в первую очередь практическое руководство, мы с вами напишем множество программ. Чтобы научиться программировать, нужно экспериментировать, совершать ошибки, а затем учиться на них. Вне всяческих сомнений, лучший способ понять происходящее — экспериментировать со значениями и данными, и такое экспериментирование следует поощрять. Все изложенные в книге программы можно будет скачать с сайта книги. Книга одинаково применима ко всем версиям Raspberry Pi. Но подробнее об этом чуть позже.

Компилятор GNU C

Существует несколько операционных систем (сред программирования), которые вы можете скачать для работы с Raspberry Pi. Как следует из ее названия, в этой книге мы будем работать с ОС Raspberry Pi (ранее называвшейся Raspbian). В ней есть все, что вам потребуется для написания и запуска ваших программ.

Также для работы нам потребуется GCC — коллекция компиляторов GNU. Первоначальным автором компилятора GNU C (GCC) был Ричард Столлман, основатель проекта GNU. Проект GNU был запущен в 1984 году с целью создания полноценной свободно доступной операционной системы, которая дала бы свободу сотрудничеству между пользователями компьютеров и программистами. Любой операционной системе нужен компилятор C, и, поскольку в те времена свободно доступных компиляторов не существовало, проекту GNU пришлось писать свои с нуля.

Работа финансировалась за счет пожертвований физических лиц и компаний в Фонд свободного программного обеспечения — некоммерческую организацию, созданную для поддержки проекта GNU. Первая версия GCC вышла в 1987 году. С тех пор GCC стал одним из важнейших инструментов в разработке бесплатного программного обеспечения и работает практически на всех существующих ОС.

GCC — бесплатное программное обеспечение, распространяемое под Стандартной публичной лицензией GNU (GNU General Public License). Это означает, что вы

можете использовать и изменять GCC, как и все программное обеспечение GNU. Если вам нужна поддержка нового ЦП, нового языка или новой функции, вы можете добавить ее самостоятельно или попросить кого-нибудь сделать это за вас.

Вы, наверное, знаете, что C — чрезвычайно популярный язык программирования, а еще он очень тесно связан с микропроцессором Advanced RISC Machine (ARM), который используется в ядре Raspberry Pi. Но вам не обязательно знать C, чтобы писать программы на ассемблере, поэтому этим не озабочивайтесь.

GCC — очень умная программа, и использовать ее можно по-разному. Одним из ее ключевых компонентов является ассемблер, который нас, собственно говоря, и интересует в первую очередь.

Учимся на примерах

Программы, приведенные в этой книге, призваны проиллюстрировать вам концепции, которые я буду объяснять простым языком и, где это возможно, с практической точки зрения. Я не буду грузить вас длинными и сложными листингами программ, да и нет в этом необходимости. Я буду давать простые примеры и информацию и объединять их в группы по два, три, четыре и более, чтобы программа в итоге получала полезный результат, а вы научились бы многому в процессе ее создания.

Иногда возникают вопросы вроде «курицы и яйца», но я старался свести их к минимуму. Мы будем вводить новые понятия в порядке, соответствующем полученным знаниям. Однако это не всегда возможно, и на этих моментах я буду останавливаться отдельно. В таких случаях вам придется просто поверить, что это работает, и однажды станет ясно, почему.

Программирование — это и впрямь весело. Я написал множество книг на эту тему, и значительная часть из них была посвящена домашним компьютерам и тому, как их программировать и использовать. Меня никто не учил работе с компьютерами. А если я могу сделать что-то, значит, сможет и любой другой человек. Но в этом есть и доля огорчения! Нет на свете ни одного программиста, будь то новичок или эксперт, который не тратил бы кучу времени на решение некоторой задачи программирования с тем, чтобы позже понять, что решение было прямо у него под носом. Настоящее удовлетворение приходит тогда, когда вы сами решаете задачу.

Дам совет: если вы не можете решить проблему, отвлекитесь и займитесь чем-нибудь еще. Очень часто решение приходит к вам именно тогда, когда вы занимаетесь чем-то другим. Это работает не только в программировании, но и в жизни!

Что вы узнаете?

Если говорить коротко, вы станете опытным программистом на ассемблере. К концу этой книги, если вы проработаете и примените на практике приведенные в ней примеры программ, вы сможете разрабатывать, писать и создавать программы с машинным кодом для выполнения огромного множества задач. Вы также получи-

те основу, позволяющую углубиться в другие темы, касающиеся микросхем ARM и системного программирования.

Вы также познакомитесь с различными способами использования компилятора GCC, включая написание кода для операционной системы Raspberry Pi и объединение собранных программ с библиотеками автономных функций.

Вы узнаете, как интерпретировать и управлять возможностями платы Raspberry Pi на фундаментальном уровне. Вы будете писать инструкции непосредственно для чипа ARM.

Вам также необходимо научиться решать проблемы. Когда программа с машинным кодом начинает работать неправильно, причиной этого часто является простая логическая ошибка. У GCC есть встроенные инструменты отладки, и мы увидим, как их эффективно использовать. Я также дам несколько полезных советов о том, как сузить область поиска и в конечном итоге найти источник проблемы.

Совместимость четвертого издания книги

В принципе, эта книга совместима со всей линейкой компьютеров Raspberry Pi. Один из основных принципов разработчиков аппаратного и программного обеспечения этой системы — обеспечить ее обратную совместимость (насколько это возможно). По сути, это означает, что программа, которая работает на одной модели Raspberry Pi, будет работать и на другой, если она написана правильно (с точки зрения программного обеспечения). Стороннее оборудование тоже должно работать, за исключением имеющего некоторые аппаратные изменения. Например, порт HDMI работать будет, но подключаться к нему придется через другой кабель (к примеру, не стандартный, а с микроразъемом).

Физически разница между моделями и версиями достаточно очевидна (рис. 1.2). Для целей этой книги я буду исходить из предположения, что у вас есть плата

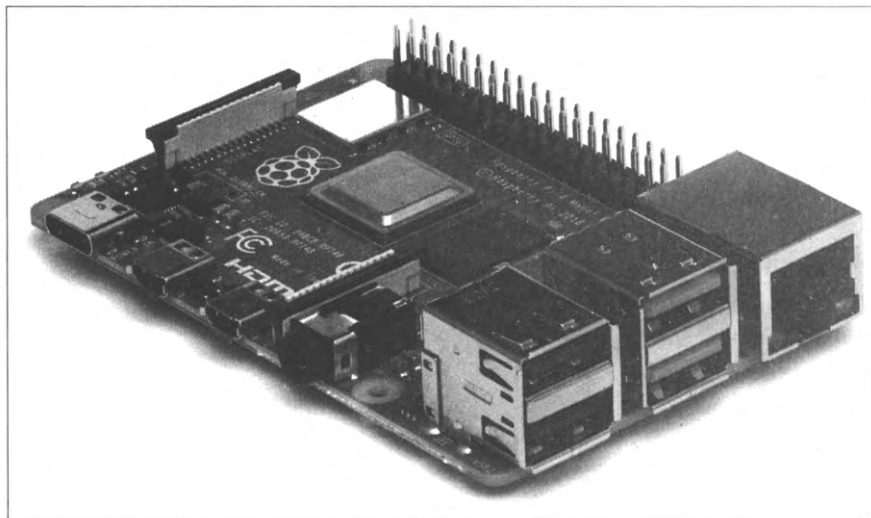


Рис. 1.2. Raspberry Pi 4 Model B

с портом GPIO в стандартной комплектации, а именно здесь с точки зрения этой книги проявляется значительная часть изменений. Вам станет понятно, что я имею в виду, когда вы станете рассматривать некоторые из примеров, в которых показано, как использовать код для подключения к устройствам. О необходимых изменениях я также буду предупреждать.

К сегодняшнему моменту плата Raspberry Pi сменила несколько различных версий чипа ARM, и это может продолжаться и в будущем. В настоящее время просто имейте в виду, что различия могут быть, но они станут видны вам по мере роста ваших знаний. Поскольку книга эта, по сути, адресована новичкам, имеющиеся различия не повлияют на основные понятия программирования. Эта книга — введение в язык ассемблера ARM, независимо от модели Raspberry Pi, которую вы используете.

ОС Raspberry Pi

Когда появилась новость о выпуске Raspberry Pi 4 с объемом памяти 8 Гбайт (май 2020 г.), компания Raspberry Pi Foundation сообщила, что меняет название своей официальной операционной системы с Raspbian на Raspberry Pi OS. Целью такого изменения явилось создание аналогичной и стабильной операционной системы и среды в стиле Windows для всех платформ, а также по максимуму, насколько возможно, обеспечение постоянной обратной совместимости с программным обеспечением и приложениями.

Имеются две основные категории процессоров: 32-разрядные и 64-разрядные. Платформа поддерживает как 32-разрядные, так и 64-разрядные операционные системы, и внешний вид ОС не меняется, т. к. сборка этих версий происходит отдельно. Попросту говоря, 64-разрядный процессор более эффективен, чем 32-разрядный, т. к. он может обрабатывать больше данных одновременно. 64-битный процессор способен хранить больше значений, включая адреса памяти, и имеет в распоряжении в 4 млрд раз больше адресов, чем физическая память 32-разрядного процессора. Это очень круто!

С точки зрения пользователя, использующего стандартную 32-разрядную операционную систему, функциональной разницы в том, что описано в этой книге, не будет. Читайте Raspberry Pi OS вместо Raspbian и наоборот. Что еще более важно, ОС остается обратно совместимой, так что ОС Raspberry Pi будет работать на всех версиях Raspberry Pi. Вы можете обновить текущую операционную систему до последней версии Raspberry Pi OS в любой момент.

Проект Raspbian не сворачивается и, несомненно, ляжет в основу при создании 32-разрядной версии ОС Raspberry Pi. Учитывая, что несколько моделей Raspberry Pi, включая очень популярную Raspberry Pi Zero, никогда не будут работать с 64-разрядной ОС, 32-разрядная платформа еще несколько лет останется востребованной, если не доминирующей.

А что насчет 64-разрядной системы?

Raspberry Pi 2B (v1.2) была первой платой с 64-разрядным процессором ARM, но для него не было выпущено официальной 64-разрядной ОС. Дело в том, что Raspberry Pi Foundation сосредоточилась в основном на том, чтобы 32-разрядная ОС Raspbian работала на всех поколениях RPi. Начиная с упомянутой Raspberry Pi 2B (v1.2), все версии RPi работают на 64-разрядных процессорах, которые также могут работать в 32-разрядном режиме.

32-разрядный режим микросхемы ARM называется AArch32 (или A32), а 64-разрядный — AArch64 (или A64). Это означает, что 32-разрядное программное обеспечение может работать на 64-разрядном чипе ARM, а вот наоборот — нет.

На момент подготовки книги уже вышла бета-версия 64-разрядной ОС Raspberry Pi, и вы можете скачать и попробовать ее. После отработки бета-версии она будет выпущена для всех 64-разрядных плат (см. главу 30). Обратите внимание, что 32-разрядная версия (A32) языка ассемблера ARM отличается от 64-разрядной версии (A64), поэтому код, написанный на A32, не будет просто так работать на A64 без определенных изменений.

Клавиатурные вычисления

В ноябре 2020 года произошло неизбежное: наконец-то появились настольные компьютеры на Raspberry Pi, в которых плата была модернизирована и упакована в красно-белую клавиатуру для запуска Raspberry Pi 400 (рис. 1.3).



Рис. 1.3. Raspberry Pi 400 — идеальный компьютер для клавиатурных вычислений

Значимость ARM

Все микропроцессоры основаны на определенной архитектуре набора команд (ISA), и наиболее значимой из них в последние годы была архитектура x86, которая доминировала на рынке настольных компьютеров и ноутбуков (ПК и Apple). В последнее время также используется 64-разрядная версия архитектуры под названием x86-64, или AMD64.

Однако большинство планшетов и смартфонов Apple и Android работают на процессорах ARM, несовместимых с архитектурой x86. Чип ARM на этих устройствах

используется из-за его низкого энергопотребления и, как следствие, увеличения времени автономной работы. Но эта несовместимость означает, что программное обеспечение, скомпилированное для настольных компьютеров и ноутбуков, нельзя напрямую запустить на мобильных устройствах под управлением ARM.

Вычислительные процессоры с сокращенным набором команд (или процессоры RISC), применяемые в устройствах ARM, используют для выполнения задачи множество простых инструкций. Процессоры x86 работают на CISC (сложном наборе команд) и для выполнения аналогичной задачи задействуют большее число сложных инструкций.

За счет этого в чипах ARM и достигается более низкое энергопотребление, поскольку в их структуре ядра меньше транзисторов, чем у чипов на основе CISC. Это также означает, что, с точки зрения программиста, вам, как программисту ARM, придется учить меньше инструкций!

Другое очень важное отличие состоит в том, что компания ARM Holdings не является производителем микросхем. Она разрабатывает микросхемы, а затем продает лицензии на их производство, чтобы другие производители могли включить их в свои собственные разработки.

В середине 2020 года компания Apple Computers объявила, что полностью перейдет на собственные чипы ARM, которые будут устанавливаться во все новые компьютеры, что обеспечит работоспособность ее приложений на всех выпускаемых ею устройствах. Microsoft Office 365, Adobe Creative Cloud и прочие программы, вероятно, тоже станут запускаться на новых компьютерах Mac ARM. Не требуется большого воображения, чтобы представить, как это будет работать на любой машине на базе ARM.

Ноутбуки и настольные компьютеры на базе x86 еще какое-то время продолжат доминировать на рынке, по крайней мере с точки зрения количества выпускаемых компьютеров, но машины Apple Mac на базе ARM должны вывести на основной рынок ноутбуки/настольные компьютеры на базе ARM и могут соблазнить этим других игроков.

А это — еще один повод изучать программирование на ассемблере ARM.

Raspberry Pi сквозь века

Платформа Raspberry Pi развивается с момента первого выпуска в 2012 году. В каждом новом поколении часто внедряются значительные улучшения типа центрального процессора, емкости памяти, поддержки сети и периферийных устройств.

Некоторые из основных спецификаций для каждой из моделей Raspberry Pi приведены в табл. 1.1. Как правило, Raspberry Pi бывают трех разных моделей:

- ♦ Model B — это «полноразмерные» платы с портами Ethernet и USB. В версиях B+ часто добавляются улучшения или особые различия;
- ♦ Model A — это платы квадратной формы. Их можно считать «более легкой» версией Raspberry Pi, обычно с более низкими характеристиками, чем у флаг-

манской модели B, с меньшим количеством портов USB и часто без Ethernet. Поэтому они дешевле;

- ◆ Zero — самый маленький доступный вариант Raspberry Pi. У этих плат меньше вычислительной мощности, чем у модели B, но они также потребляют меньше энергии. Ни USB, ни Ethernet – все простенько и со вкусом!

Объем памяти, доступный на каждой Raspberry Pi, для этой книги значения не имеет, но имейте в виду, что адреса в памяти, которые будут отображаться в вашей программе, могут отличаться от приведенных в книге примеров. Однако это не должно мешать работе программы.

Таблица 1.1. Модели Raspberry Pi и их основные характеристики

Модель RPI	Дата выпуска	SOC	ЦП	FPU	GPU	ARM IS
400	Ноябрь 2020 г.	BCM 2711C0	4 × Cortex-A72 с частотой 1,8 ГГц	VFPv4 + NEON	Broadcom VideoCore VI	ARMv8-A (64/32-бит)
4 B	Июнь 2019 г.; май 2020 г. (8 Гбайт)	BCM2711B0	4 × Cortex-A72 с частотой 1,5 ГГц	VFPv4 + NEON	Broadcom VideoCore VI	ARMv8-A (64/32-бит)
3 B +	Май 2018 г.	BCM2837B0	4 × Cortex-A53 с частотой 1,4 ГГц	VFPv4 + NEON	Broadcom VideoCore IV	ARMv8-A (64/32-бит)
3 B	Февраль 2016 г.	BC2837	4 × Cortex-A53 с частотой 1,2 ГГц	VFPv4 + NEON	Broadcom Videocore IV	ARMv8-A (64/32-бит)
3 A +	Ноябрь 2018 г.	BCM2837B0	4 × Cortex-A53 с частотой 1,4 ГГц	VFPv4 + NEON	Broadcom Videocore IV	ARMv8-A (64/32-бит)
2 B v1.2	Октябрь 2016 г.	BCM2837	4 × Cortex-A53 с частотой 900 МГц	VFPv4 + NEON	Broadcom Videocore IV	ARMv8-A (64/32-бит)
2 B	Февраль 2015 г.	BCM2836	4 × Cortex-A7 с частотой 900 МГц	VFPv3 + NEON	Broadcom Videocore IV	ARMv7-A (32-бит)
1 B +	Июль 2014 г.	BCM2835	1 × ARM 1176JZF-S с частотой 700 МГц	VFPv2	Broadcom Videocore IV	ARMv6Z (32-бит)
1B	Май 2012 г.	BCM2835	1 × ARM 1176JZF-S с частотой 700 МГц	VFPv2	Broadcom Videocore IV	ARMv6Z (32-бит)
1 A +	Ноябрь 2014 г.	BCM2835	1 × ARM 1176JZF-S с частотой 700 МГц	VFPv2	Broadcom Videocore IV	ARMv6Z (32-бит)
1 A	Февраль 2013 г.	BCM2835	1 × ARM1176JZF-S с частотой 700 МГц	VFPv2	Broadcom Videocore IV	ARMv6Z (32-бит)
Zero WWH	Февраль 2017 г.	BCM2835	1 × ARM 1176JZF-S с частотой 1 ГГц	VFPv2	Broadcom VideoCore IV	ARMv6Z (32-бит)
Zero v1.3	Май 2016 г.	BCM2835	1 × ARM1176JZF-S с частотой 1 ГГц	VFPv2	Broadcom VideoCore IV	ARMv6Z (32-бит)
Zero v.1.2	Ноябрь 2015 г.	BCM2835	1 × ARM1176JZF-S с частотой 1 ГГц	VFPv2	Broadcom VideoCore IV	ARMv6Z (32-бит)

В табл. 1.1 используется много сокращений, которые могут быть вам знакомы, а могут быть и новыми. Мы кратко рассмотрим каждое из них и при необходимости повторно вернемся к ним позже в книге.

- ◆ **SOC, System-on-Chip** (Система на кристалле) — системный чип. Один из ключевых элементов дизайна Raspberry Pi и, как правило, самый большой чип на плате, который содержит все важные компоненты в одном корпусе. В него входят процессор ARM и графические процессоры. Обратите внимание, что каждый чип SoC сопровождается числом — например, на плате Raspberry Pi 4 B установлен чип BCM2771.
- ◆ **ЦП** — это процессор ARM или используемые процессоры, интегрированные в SoC. В таблице указан конкретный чип ARM, а также количество его ядер и скорость работы. К примеру, на оригинальном Raspberry Pi B установлен процессор ARM 1176JZF-S, работающий на частоте 700 МГц. В новой Raspberry Pi 4 B используются четыре ядра ARM Cortex-A72, работающие на частоте 1,5 ГГц.
- ◆ **FPU** — модуль с плавающей точкой, или математический сопроцессор, который реализует сложные математические функции. Он тоже входит в SoC, а в более поздних версиях SoC установлен сопроцессор Neon, позволяющий FPU выполнять несколько операций параллельно. **VFP** — это сокращение от **Vector floating-point**, вектор с плавающей точкой.
- ◆ **GPU** — графический процессор. Он входит в SoC Broadcom, производится компанией Broadcom и обычно представляет собой VideoCore IV, а в Raspberry Pi 4 B — версию VideoCore VI. По сути, это мультимедийный процессор, который может декодировать различные графические и звуковые форматы (кодеки) для обеспечения отличного качества изображения на мониторе при сохранении низкого энергопотребления.
- ◆ В последнем столбце таблицы указана реализованная архитектура набора команд и отмечено, является она 32-разрядной или 64-разрядной. Как мы уже говорили ранее, 64-битный набор инструкций доступен на Raspberry Pi, начиная с RPi 2 B v1.2, т. е. фактически с октября 2016 года.

Вычислительные модули

Вычислительный модуль (Computer Module) — это упрощенная версия Raspberry Pi, предназначенная для промышленных или коммерческих приложений. Суть в том, что их может использовать в своих проектах кто угодно (вы приобретаете плату и дорабатываете ее самостоятельно под свою задачу). Таким образом, вычислительные модули не имеют возможностей подключения, которые вы можете найти на эквивалентной плате Raspberry Pi, но при этом они не столь надежны. Поэтому, хоть они и работают на процессорах ARM и на Raspberry Pi, они здесь не рассматриваются.

Но затем появился модуль Computer Module 4. В отличие от предыдущих вычислительных модулей, у него имеется обновленная плата ввода/вывода, простой доступ

ко всем стандартным разъемам интерфейсов, и, стало быть, это уже готовая платформа разработки и отправная точка для ваших собственных проектов.

Используемые обозначения

В этой книге применяются стандартные обозначения. Числовые типы и определенные операции с числами часто используются в книгах по программированию, подобных этой, и важно различать их. В табл. 1.2 приведен их краткий список. Точное значение обозначений будет ясно, когда мы встретим их в тексте книги.

Таблица 1.2. Стандартные обозначения

Обозначение	Пояснение
% или 0b	Обозначает, что идущее за этим обозначением число записано в двоичной системе счисления (с основанием 2). Например: %11110000 или 0b11110000
0x	Обозначает, что следующее за ним число является шестнадцатеричным. Например: 0xCAFE
< >	Угловые скобки часто используются для обозначения слова, которое не следует понимать буквально, а надо расценивать как объект, используемый с командой. Например, запись <Регистр> означает, что в угловых скобках следует использовать имя регистра — например: R0, а не само слово «Регистр»
Dest	Сокращение от destination
Операнд1	Комментарии в тексте часто будут ссылаться на <i>операнд1</i> и его использование. Вы же мысленно подставляете вместо него те или иные значения. Обычно <i>операнд1</i> — это регистр
Операнд2	Комментарии в тексте часто будут ссылаться на <i>операнд2</i> и его использование. Вы же мысленно подставляете вместо него те или иные значения. Обычно <i>операнд1</i> — это регистр
Op1	Сокращение от <i>операнд1</i>
Op2	Сокращение от <i>операнд2</i>
{ }	Скобки показывают, что указанный в них элемент является необязательным и может быть опущен, если он не нужен. Например, запись ADD(S) означает, что значения S может и не быть

Центр истории вычислительной техники

Центр истории вычислительной техники (CCH, Center of Computing History) — новаторская образовательная благотворительная организация, открывшаяся на своем нынешнем месте в Кембридже в августе 2013 года. CCH (www.computinghistory.org.uk) раскрывает историю информационной эпохи на основе изучения исторического, социального и культурного воздействия разработок в области персональных компьютеров на человеческую цивилизацию. В этом центре имеется огромная кол-