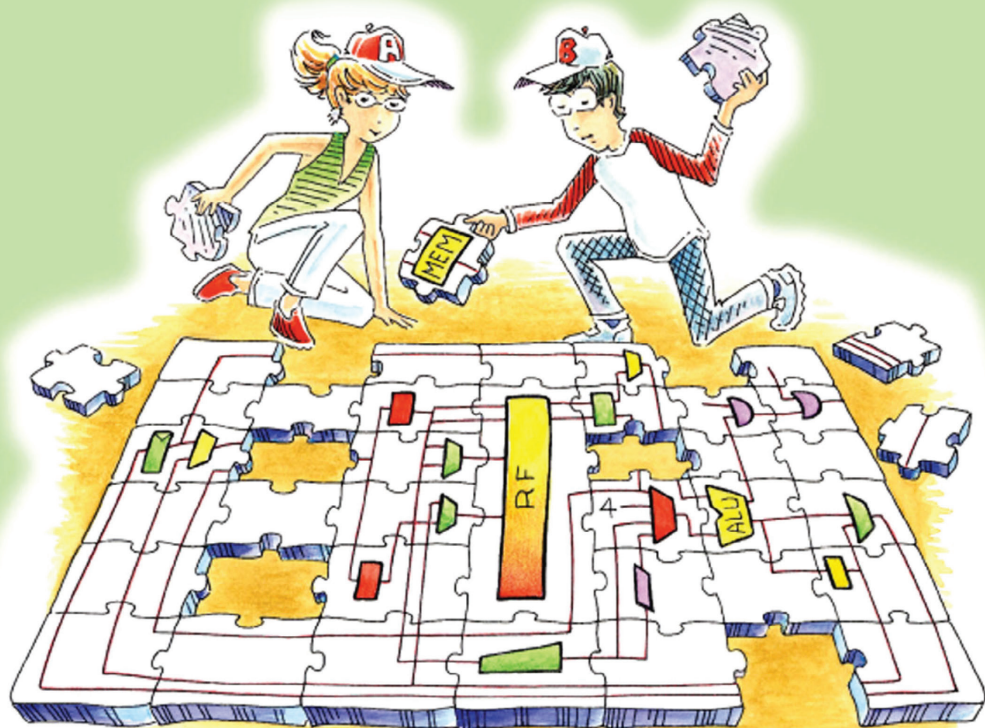


# Цифровая схемотехника и архитектура компьютера



Дэвид М. Харрис и Сара Л. Харрис

Цветное издание

УДК 004.2+744.4  
ББК 32.971.3  
Х21

Дэвид М. Харрис, Сара Л. Харрис  
Х21 Цифровая схемотехника и архитектура компьютера. / пер. с англ. Imagination Technologies. – М.: ДМК Пресс, 2018. – 792 с.: цв. ил.

**ISBN 978-5-97060-570-7**

В книге представлен уникальный и современный подход к разработке цифровых устройств. Авторы начинают с цифровых логических элементов, переходят к разработке комбинационных и последовательных схем, а затем используют эти базовые блоки как основу для самого сложного: проектирования настоящего процессора MIPS. По всему тексту приводятся примеры на языках SystemVerilog и VHDL, иллюстрирующие методы и способы проектирования схем с помощью САПР. Изучив эту книгу, читатели смогут разработать свой собственный микропроцессор и получат полное понимание того, как он работает. В книге объединен привлекательный и юмористический стиль изложения с развитым и практичным подходом к разработке цифровых устройств.

Во второе англоязычное издание вошли новые материалы о системах ввода/вывода применительно к процессорам общего назначения как для ПК, так и для микроконтроллеров. Приведены практические примеры интерфейсов периферийных устройств с применением RS-232, SPI, управления двигателями, прерываний, беспроводной связи и аналого-цифрового преобразования. Представлено высокоуровневое описание интерфейсов, включая USB, SDRAM, WiFi, PCI Express и другие.

Издание будет полезно студентам, инженерам, а также широкому кругу читателей, интересующихся современной схемотехникой.

This edition of «Digital Design and Computer Architecture by David Money Harris and Sarah L. Harris is published by arrangement with ELSEVIER INC., a Delaware corporation having its principal place of business at 360 Park Avenue South, New York, NY 10010, USA

Это издание книги Дэвида Мани Харриса и Сары Л. Харрис «Цифровая схемотехника и архитектура компьютера» публикуется по соглашению с ELSEVIER INC., Делавэрской корпорацией, которая осуществляет основную деятельность по адресу 360 Park Avenue South, New York, NY 10010, USA.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-0-12-394424-5 (анг.)

© 2013 Elsevier, Inc. All rights reserved.

© Перевод, Imagination Technologies, 2016

ISBN 978-5-97060-570-7 (рус.)

© Оформление, издание, ДМК Пресс, 2017

# Оглавление

<b>Похвальные отзывы на книгу</b>	
<b>«Цифровая схемотехника и архитектура компьютера»</b>	<b>15</b>
<b>Об авторах</b>	<b>17</b>
<b>Предисловие к изданию на русском языке</b>	<b>18</b>
Благодарности участникам проекта .....	22
<b>Предисловие</b>	<b>24</b>
Особенности книги .....	24
Материалы в Интернете .....	26
Как использовать программный инструментарий в учебном курсе .....	27
Опечатки .....	28
Признательность за поддержку .....	28
<b>Глава 1 От нуля до единицы</b>	<b>31</b>
1.1. План игры .....	31
1.2. Искусство управления сложностью .....	32
1.2.1. Абстракция .....	33
1.2.2. Конструкторская дисциплина .....	35
1.2.3. Три базовых принципа .....	36
1.3. Цифровая абстракция .....	38
1.4. Системы счисления .....	40
1.4.1. Десятичная система счисления .....	40
1.4.2. Двоичная система счисления .....	41
1.4.3. Шестнадцатеричная система счисления .....	43
1.4.4. Байт, полубайт и «весь этот джаз» .....	45
1.4.5. Сложение двоичных чисел .....	46
1.4.6. Знак двоичных чисел .....	47
1.5. Логические элементы .....	53
1.5.1. Логический вентиль НЕ .....	53
1.5.2. Буфер .....	54
1.5.3. Логический вентиль И .....	54
1.5.4. Логический вентиль ИЛИ .....	54
1.5.5. Другие логические элементы с двумя входными сигналами .....	55
1.5.6. Логические элементы с количеством входов больше двух .....	56
1.6. За пределами цифровой абстракции .....	57
1.6.1. Напряжение питания .....	57
1.6.2. Логические уровни .....	57
1.6.3. Допускаемые уровни шумов .....	58
1.6.4. Передаточная характеристика .....	59
1.6.5. Статическая дисциплина .....	60
1.7. КМОП-транзисторы .....	62
1.7.1. Полупроводники .....	63

1.7.2. Диоды .....	64
1.7.3. Конденсаторы .....	64
1.7.4. п-МОП- и р-МОП-транзисторы.....	65
1.7.5. Логический вентиль НЕ на КМОП-транзисторах.....	69
1.7.6. Другие логические вентили на КМОП-транзисторах .....	69
1.7.7. Передаточный логический вентиль .....	72
1.7.8. Псевдо п-МОП-логика .....	72
1.8. Потребляемая мощность.....	73
1.9. Краткий обзор главы 1 и того, что нас ждет впереди .....	75
Упражнения .....	77
Вопросы для собеседования .....	89

## **Глава 2 Проектирование комбинационной логики 91**

2.1. Введение .....	91
2.2. Булевы уравнения.....	95
2.2.1. Терминология.....	95
2.2.2. Дизъюнктивная форма .....	96
2.2.3. Конъюнктивная форма .....	98
2.3. Булева алгебра.....	99
2.3.1. Аксиомы .....	100
2.3.2. Теоремы одной переменной .....	100
2.3.3. Теоремы с несколькими переменными .....	102
2.3.4. Правда обо всем этом .....	104
2.3.5. Упрощение уравнений.....	105
2.4. От логики к логическим элементам .....	106
2.5. Многоуровневая комбинационная логика .....	110
2.5.1. Минимизация аппаратуры.....	111
2.5.2. Перемещение инверсии.....	112
2.6. Что за X и Z?.....	115
2.6.1. Недопустимое значение: X .....	115
2.6.2. Третье состояние: Z.....	116
2.7. Карты Карно .....	118
2.7.1. Думайте об овалах.....	119
2.7.2. Логическая минимизация на картах Карно.....	120
2.7.3. Безразличные переменные .....	124
2.7.4. Подводя итоги .....	124
2.8. Базовые комбинационные блоки.....	125
2.8.1. Мультиплексоры .....	125
2.8.2. Дешифраторы.....	129
2.9. Временные характеристики .....	131
2.9.1. Задержка распространения и задержка реакции .....	131
2.9.2. Импульсные помехи .....	136
2.10. Резюме .....	139
Упражнения .....	140
Вопросы для собеседования .....	147

<b>Глава 3</b>	<b>Проектирование последовательной логики</b>	<b>149</b>
3.1.	Введение .....	149
3.2.	Защелки и триггеры .....	150
3.2.1.	RS-триггер .....	151
3.2.2.	D-защелка .....	154
3.2.3.	D-Триггер .....	155
3.2.4.	Регистр .....	156
3.2.5.	Триггер с функцией разрешения .....	156
3.2.6.	Триггер с функцией сброса .....	158
3.2.7.	Проектирование триггеров и защелок на транзисторном уровне .....	159
3.2.8.	Общий обзор .....	160
3.3.	Проектирование синхронных логических схем .....	161
3.3.1.	Некоторые проблемные схемы .....	161
3.3.2.	Синхронные последовательные схемы .....	163
3.3.3.	Синхронные и асинхронные схемы .....	166
3.4.	Конечные автоматы .....	166
3.4.1.	Пример проектирования конечного автомата .....	167
3.4.2.	Кодирование состояний .....	173
3.4.3.	Автоматы Мура и Мили .....	176
3.4.4.	Декомпозиция конечных автоматов .....	180
3.4.5.	Восстановление конечных автоматов по электрической схеме .....	182
3.4.6.	Обзор конечных автоматов .....	185
3.5.	Синхронизация последовательных схем .....	185
3.5.1.	Динамическая дисциплина .....	187
3.5.2.	Временные характеристики системы .....	188
3.5.3.	Расфазировка тактовых сигналов .....	194
3.5.4.	Метастабильность .....	197
3.5.5.	Синхронизаторы .....	199
3.5.6.	Вычисление времени разрешения .....	201
3.6.	Параллелизм .....	205
3.7.	Резюме .....	209
	Упражнения .....	210
	Вопросы для собеседования .....	218
<b>Глава 4</b>	<b>Языки описания аппаратуры</b>	<b>221</b>
4.1.	Введение .....	221
4.1.1.	Модули .....	222
4.1.2.	Происхождение языков SystemVerilog и VHDL .....	222
4.1.3.	Симуляция и Синтез .....	224
4.2.	Комбинационная логика .....	226
4.2.1.	Битовые операторы .....	227
4.2.2.	Комментарии и пробелы .....	229
4.2.3.	Операторы сокращения .....	230
4.2.4.	Условное присваивание .....	230
4.2.5.	Внутренние переменные .....	233
4.2.6.	Приоритет .....	235
4.2.7.	Числа .....	235

4.2.8. Z-состояние и X-состояние .....	237
4.2.9. Манипуляция битами .....	239
4.2.10. Задержки .....	239
4.3. Структурное моделирование .....	241
4.4. Последовательная логика .....	245
4.4.1. Регистры .....	245
4.4.2. Регистры со сбросом .....	245
4.4.3. Регистры с сигналом разрешения .....	248
4.4.4. Группы регистров .....	249
4.4.5. Защелки .....	250
4.5. И снова комбинационная логика .....	251
4.5.1. Операторы case .....	254
4.5.2. Операторы if .....	256
4.5.3. Таблицы истинности с незначимыми битами .....	259
4.5.4. Блокирующие и неблокирующие присваивания .....	260
4.6. Конечные автоматы .....	264
4.7. Типы данных .....	268
4.7.1. SystemVerilog .....	268
4.7.2. VHDL .....	269
4.8. Параметризованные модули .....	272
4.9. Среда тестирования .....	275
4.10. Резюме .....	280
Упражнения .....	281
Вопросы для собеседования .....	291

## **Глава 5 Цифровые функциональные узлы 293**

5.1. Введение .....	293
5.2. Арифметические схемы .....	294
5.2.1. Сложение .....	294
5.2.2. Вычитание .....	302
5.2.3. Компараторы .....	303
5.2.4. АЛУ .....	304
5.2.5. Схемы сдвига и циклического сдвига .....	306
5.2.6. Умножение .....	308
5.2.7. Деление .....	309
5.2.8. Дополнительная литература .....	311
5.3. Представление чисел .....	311
5.3.1. Числа с фиксированной точкой .....	311
5.3.2. Числа с плавающей точкой .....	312
5.4. Функциональные узлы последовательной логики .....	317
5.4.1. Счетчики .....	317
5.4.2. Сдвигающие регистры .....	318
5.5. Матрицы памяти .....	321
5.5.1. Обзор .....	321
5.5.2. Динамическое ОЗУ (DRAM) .....	324
5.5.3. Статическое ОЗУ (SRAM) .....	325

5.5.4. Площадь и задержки.....	326
5.5.5. Регистровые файлы .....	327
5.5.6. Постоянное запоминающее устройство.....	327
5.5.7. Реализация логических функций с использованием матриц памяти.....	330
5.5.8. Языки описания аппаратуры и память.....	331
5.6. Матрицы логических элементов .....	332
5.6.1. Программируемые логические матрицы .....	333
5.6.2. Программируемые пользователем вентильные матрицы .....	335
5.6.3. Схемотехника матриц.....	340
5.7. Резюме.....	342
Упражнения .....	343
Вопросы для собеседования .....	353

## **Глава 6    Архитектура** **355**

6.1. Предисловие .....	355
6.2. Язык ассемблера .....	357
6.2.1. Инструкции .....	358
6.2.2. Операнды: регистры, память и константы .....	360
6.3. Машинный язык .....	367
6.3.1. Инструкции типа <i>R</i> .....	368
6.3.2. Инструкции типа <i>I</i> .....	369
6.3.3. Инструкции типа <i>J</i> .....	371
6.3.4. Расшифровываем машинные коды.....	371
6.3.5. Могущество хранимой программы .....	372
6.4. Программирование .....	373
6.4.1. Арифметические / логические инструкции.....	374
6.4.2. Переходы.....	378
6.4.3. Условные операторы .....	381
6.4.4. Зацикливаемся .....	383
6.4.5. Массивы.....	385
6.4.6. Вызовы функций.....	390
6.5. Режимы адресации.....	400
6.6. Камера, мотор! Компилируем, ассемблируем и загружаем.....	404
6.6.1. Карта памяти .....	404
6.6.2. Трансляция и запуск программы .....	406
6.7. Добавочные сведения .....	409
6.7.1. Псевдокоманды .....	410
6.7.2. Исключения.....	411
6.7.3. Команды для чисел со знаком и без знака.....	413
6.7.4. Команды для работы с числами с плавающей точкой .....	415
6.8. Живой пример: архитектура x86.....	417
6.8.1. Регистры x86 .....	418
6.8.2. Операнды x86 .....	418
6.8.3. Флаги состояния .....	420
6.8.4. Команды x86.....	421
6.8.5. Кодировка команд x86 .....	423
6.8.6. Другие особенности x86.....	425

6.8.7. Оглядываясь назад .....	425
6.9. Резюме .....	426
Упражнения .....	427
Вопросы для собеседования .....	437

## **Глава 7 Микроархитектура 439**

7.1. Введение .....	439
7.1.1. Архитектурное состояние и система команд .....	440
7.1.2. Процесс разработки .....	441
7.1.3. Микроархитектуры MIPS .....	443
7.2. Анализ производительности .....	444
7.3. Однотактный процессор .....	446
7.3.1. Однотактный тракт данных .....	446
7.3.2. Однотактное устройство управления .....	452
7.3.3. Дополнительные команды .....	456
7.3.4. Анализ производительности .....	458
7.4. Многотактный процессор .....	460
7.4.1. Многотактный тракт данных .....	460
7.4.2. Многотактное устройство управления .....	467
7.4.3. Дополнительные команды .....	474
7.4.4. Анализ производительности .....	478
7.5. Конвейерный процессор .....	479
7.5.1. Конвейерный тракт данных .....	482
7.5.2. Конвейерное устройство управления .....	484
7.5.3. Конфликты .....	484
7.5.4. Дополнительные команды .....	496
7.5.5. Анализ производительности .....	496
7.6. Пишем процессор на HDL .....	498
7.6.1. Однотактный процессор .....	499
7.6.2. Универсальные строительные блоки .....	504
7.6.3. Тестовое окружение .....	506
7.7. Исключения .....	510
7.8. Улучшенные микроархитектуры .....	513
7.8.1. Длинные конвейеры .....	514
7.8.2. Предсказание условных переходов .....	516
7.8.3. Суперскалярный процессор .....	518
7.8.4. Процессор с внеочередным выполнением команд .....	521
7.8.5. Переименование регистров .....	524
7.8.6. SIMD .....	525
7.8.7. Многопоточность .....	526
7.8.8. Симметричные мультипроцессоры .....	528
7.8.9. Гетерогенные мультипроцессоры .....	529
7.9. Живой пример: микроархитектура x86 .....	532
7.10. Резюме .....	539
Упражнения .....	541
Вопросы для собеседования .....	546



<b>Глава 8 Иерархия памяти и подсистема ввода-вывода</b>	<b>549</b>
8.1. Введение .....	549
8.2. Анализ производительности систем памяти .....	554
8.3. Кэш-память .....	556
8.3.1. Какие данные хранятся в кэш-памяти? .....	557
8.3.2. Как найти данные в кэш-памяти? .....	558
8.3.3. Какие данные заместить в кэш-памяти? .....	567
8.3.4. Улучшенная кэш-память .....	569
8.3.5. Эволюция кэш-памяти процессоров MIPS .....	573
8.4. Виртуальная память .....	573
8.4.1. Трансляция адресов .....	576
8.4.2. Таблица страниц .....	578
8.4.3. Буфер ассоциативной трансляции .....	580
8.4.4. Защита памяти .....	582
8.4.5. Стратегии замещения страниц .....	583
8.4.6. Многоуровневые таблицы страниц .....	584
8.5. Системы ввода-вывода .....	586
8.6. Ввод-вывод во встроенных системах .....	588
8.6.1. Микроконтроллер PIC32MX675F512H .....	589
8.6.2. Цифровой ввод-вывод общего назначения .....	594
8.6.3. Последовательный ввод-вывод .....	596
8.6.4. Таймеры .....	610
8.6.5. Прерывания .....	612
8.6.6. Аналоговый ввод-вывод .....	614
8.6.7. Другие внешние устройства микроконтроллера .....	621
8.7. Интерфейсы ввода-вывода персональных компьютеров .....	644
8.7.1. USB .....	646
8.7.2. PCI и PCI Express .....	647
8.7.3. Память DDR3 .....	648
8.7.4. Сеть .....	648
8.7.5. SATA .....	649
8.7.6. Подключения к ПК .....	649
8.8. Живой пример: системы памяти и ввода-вывода семейства x86 .....	652
8.8.1. Системы кэш-памяти процессоров семейства x86 .....	652
8.8.2. Виртуальная память x86 .....	655
8.8.3. Программируемый ввод-вывод x86 .....	656
8.9. Резюме .....	656
Эпилог .....	657
Упражнения .....	658
Вопросы для собеседования .....	665
<b>Приложение А Реализация цифровых систем</b>	<b>667</b>
А.1. Введение .....	667
А.2. Логические микросхемы серии 74xx .....	668
А.2.1. Логические элементы .....	668
А.2.2. Другие логические функции .....	669

А.3. Программируемая логика .....	671
А.3.1. PROM .....	672
А.3.2. Блоки PLA .....	673
А.3.3. FPGA .....	673
А.4. Заказные специализированные интегральные схемы .....	676
А.5. Работа с документацией .....	677
А.6. Семейства логических элементов .....	682
А.7. Корпуса и монтаж интегральных схем .....	685
А.8. Линии передачи .....	690
А.8.1. Согласованная нагрузка .....	691
А.8.2. Нагрузка холостого хода .....	693
А.8.3. Нагрузка короткого замыкания .....	694
А.8.4. Рассогласованная нагрузка .....	695
А.8.5. Когда нужно применять модели линии передачи .....	697
А.8.6. Правильное подключение нагрузки к линии передачи .....	698
А.8.7. Вывод формулы для $Z_0$ .....	700
А.8.8. Вывод формулы для коэффициента отражения .....	701
А.8.9. Подводя итог .....	702
А.9. Экономика .....	704

## **Приложение В Инструкции архитектуры MIPS**

**707**

## **Приложение С Программирование на языке Си**

**713**

С.1. Введение .....	713
С.2. Добро пожаловать в язык Си .....	716
С.2.1. Структура программы на языке Си .....	716
С.2.2. Запуск Си-программы .....	717
С.3. Компиляция .....	718
С.3.1. Комментарии .....	719
С.3.2. <code>#define</code> .....	719
С.3.3. <code>#include</code> .....	720
С.4. Переменные .....	721
С.4.1. Базовые типы данных .....	722
С.4.2. Глобальные и локальные переменные .....	724
С.4.3. Инициализация переменных .....	725
С.5. Операции .....	726
С.6. Вызовы функций .....	729
С.7. Управление последовательностью выполнения действий .....	731
С.7.1. Условные операторы .....	731
С.7.2. Циклы .....	733
С.8. Другие типы данных .....	736
С.8.1. Указатели .....	736
С.8.2. Массивы .....	738
С.8.3. Символы .....	743
С.8.4. Строки символов .....	744
С.8.5. Структуры .....	745

С.8.6. Оператор typedef.....	747
С.8.7. Динамическое распределение памяти.....	748
С.8.8. Связные списки.....	749
С.9. Стандартная библиотека языка Си.....	752
С.9.1. stdio.....	753
С.9.2. stdlib.....	757
С.9.3. math.....	759
С.9.4. string.....	760
С.10. Компилятор и опции командной строки.....	760
С.10.1. Компиляция нескольких исходных Си-файлов.....	761
С.10.2. Опции компилятора.....	761
С.10.3. Аргументы командной строки.....	762
С.11. Типичные ошибки.....	762










<b>Литература для дальнейшего изучения</b>	<b>769</b>
--	------------

<b>Дополнительная информация</b>	<b>771</b>
----------------------------------	------------

<b>Предметный указатель</b>	<b>772</b>
-----------------------------	------------

## От нуля до единицы

- 1.1. План игры
- 1.2. Искусство управления сложностью
- 1.3. Цифровая абстракция
- 1.4. Системы счисления
- 1.5. Логические элементы
- 1.6. За пределами цифровой абстракции
- 1.7. КМОП-транзисторы
- 1.8. Потребляемая мощность
- 1.9. Краткий обзор главы 1 и того, что нас ждет впереди
- Упражнения
- Вопросы для собеседования

Прикладное ПО	
Операционные системы	
Архитектура	
Микро-архитектура	
Логика	
Цифровые схемы	
Аналоговые схемы	
Пп приборы	
Физика	

### 1.1. План игры

За последние тридцать лет микропроцессоры буквально изменили наш мир до неузнаваемости. Сегодняшний ноутбук обладает большей вычислительной мощностью, чем большой компьютер недавнего прошлого, занимавший целую комнату. Внутри современного автомобиля представительского класса можно обнаружить около пятидесяти микропроцессоров. Именно прогресс в области микропроцессорной техники сделал возможным появление сотовых телефонов и Интернета, значительно продвинул вперед медицину и радикально изменил тактику и стратегию современной войны. Объем продаж мировой полупроводниковой промышленности вырос с 21 миллиарда долларов в 1985 году до 300 миллиардов долларов в 2011 году, причем микропроцессоры составили львиную долю этих продаж. И мы убеждены, что микропроцессоры важны не только с технической, экономической и социальной точек зрения, но и стали одним из самых увлекательных изобретений в истории челове-

ства. Когда вы закончите чтение этой книги, вы будете знать, как спроектировать и построить ваш собственный микропроцессор, а навыки, полученные на этом пути, пригодятся вам для разработки и многих других цифровых систем.

Мы предполагаем, что у вас уже есть базовые знания по теории электричества, некоторый опыт программирования и искреннее желание понять, что происходит под капотом компьютера. В этой книге основное внимание уделяется разработке цифровых систем, то есть систем, которые используют для своей работы два уровня напряжения, представляющих единицу и ноль. Мы начнем с простейших цифровых логических элементов — вентилях (digital logic gates), которые принимают определенную комбинацию единиц и нулей на входе и трансформируют ее в другую комбинацию единиц и нулей на выходе. После этого мы с вами научимся объединять эти простейшие логические элементы в более сложные модули, такие как сумматоры и блоки памяти. Затем мы перейдем к программированию на языке ассемблера — родном языке микропроцессора. И в завершение, из кирпичиков логических элементов мы с вами соберем полноценный микропроцессор, способный выполнять ваши программы, написанные на языке ассемблера.

Огромным преимуществом цифровых систем над аналоговыми является то, что необходимые для их построения блоки чрезвычайно просты, поскольку оперируют не непрерывными сигналами, а единицами и нулями.

Построение цифровой системы не требует запутанных математических расчетов или глубоких знаний в области физики. Вместо этого, задача, стоящая перед разработчиком цифровых устройств, заключается в том, чтобы собрать сложную работающую систему из этих простых блоков. Возможно, микропроцессор станет первой спроектированной вами системой, настолько сложной, что ее невозможно целиком удержать в голове. Именно поэтому одной из тем, проходящих красной нитью через эту книгу, является искусство управления сложностью системы.

## 1.2. Искусство управления сложностью

Одной из характеристик, отличающих профессионального инженера-электронщика или программиста от дилетанта, является систематический подход к управлению сложностью многоуровневой системы. Современные цифровые системы построены из миллионов и миллиардов транзисторов. Человеческий мозг не в состоянии предсказать поведение подобных систем путем составления уравнений, описывающих движение каждого электрона в каждом транзисторе системы, и последующего решения этой системы уравнений. Для того, чтобы разработать удачный микропроцес-

сор и не утонуть при этом в море избыточной информации, необходимо научиться управлять сложностью разрабатываемой системы.

### 1.2.1. Абстракция

Критически важный принцип управления сложностью системы — *абстракция*, подразумевающая исключение из рассмотрения тех элементов, которые в данном конкретном случае несущественны для понимания работы этой системы. Любую систему можно рассматривать с различных уровней абстракции. Политику, участвующему в выборах, например, нет нужды учитывать все детали окружающего его мира, ему достаточно абстрактной иерархической модели страны, состоящей из населенных пунктов, областей и федеральных округов. В области может быть несколько населенных пунктов, а федеральный округ включает в себя разные области. Если политик борется за пост президента, то его, скорее всего, интересует то, как проголосует федеральный округ в целом, при этом ему не обязательно знать, какое количество голосов он наберет в каждом конкретном населенном пункте этого округа. Для политика федеральный округ — это его уровень абстракции. С другой стороны, бюро переписи населения обязано знать количество жителей в каждом городе или поселке страны и потому должно оперировать на самом низком уровне абстракции данной системы — на уровне населенных пунктов.

На **Рис. 1.1** показаны уровни абстракции, типичные для любой электронной компьютерной системы вместе со строительными блоками, характерными для каждого уровня абстракции этой системы. На самом низком уровне абстракции находится физика, изучающая движение электронов. Поведение электронов описывается квантовой механикой и системой уравнений Максвелла.

Рассматриваемая нами современная электронная система состоит из полупроводниковых устройств (devices), таких как транзисторы (а когда-то это были электронные лампы). Каждое такое устройство имеет четко определенные точки соединения с другими подобными устройствами. Эти точки мы будем называть *контактами* (в англоязычной литературе используется термин *terminal*). Любое электронное устройство может быть представлено абстрактной математической моделью, описывающей изменяющуюся во времени взаимозависимость тока и напряжения. Такие же изменения тока и напряжения можно наблюдать на экране осциллографа, если подключить осциллограф к контактам реального устройства. Данный под-

Прикладное ПО		Прикладные программы (ПО)
Операционные системы		Драйверы устройств
Архитектура		Регистры команд (инструкций)
Микро-архитектура		Управление потоками
Логика		Сумматоры Память
Цифровые схемы		Элементы И Элементы НЕ
Аналоговые схемы		Усилители Фильтры
Пп приборы		Транзисторы Диоды
Физика		Электроны

**Рис. 1.1** Уровни абстракции электронной вычислительной системы

Каждая глава этой книги начинается с иконок (см. Рис. 1.1), символически изображающих уровни абстракции электронной системы, которые мы перечислили выше. Иконка темно-синего цвета указывает на тот уровень абстракции, которому уделяется главное внимание в этой конкретной главе. Иконки более светлого оттенка синего указывают на другие уровни абстракции, также затронутые в этой главе.

ход означает, что, если рассматривать систему на уровне устройств, функции которых однозначно определены, то можно не учитывать поведение электронов внутри отдельных устройств этой системы.

Следующий уровень абстракции — это *аналоговые схемы* (analog circuits), в которых полупроводниковые устройства соединены таким образом, чтобы они образовывали функциональные компоненты, такие как усилители, например. Напряжение на входе и на выходе аналоговой цепи изменяется в непрерывном диапазоне.

В отличие от аналоговых цепей, *цифровые схемы* (digital circuits), такие как логические вентили, используют два строго ограниченных дискретных уровня напряжения. Один из этих дискретных уровней — это логический нуль, другой — логическая единица. В разделах этой книги, посвященных разработке цифровых схем и устройств, мы будем использовать простейшие цифровые схемы для построения сложных цифровых модулей, таких как сумматоры и блоки памяти.

Микроархитектурный уровень абстракции, или просто *микроархитектура* (microarchitecture), связывает логический и архитектурный уровни абстракции. Архитектурный уровень абстракции, или *архитектура* (architecture), описывает компьютер с точки зрения программиста. Например, архитектура Intel x86, используемая микропроцессорами большинства персональных компьютеров (ПК), определяется набором инструкций и регистров (памяти для временного хранения переменных), доступным для использования программистом. Микроархитектура — это соединение простейших цифровых элементов в логические блоки, предназначенные для выполнения команд, определенных какой-то конкретной архитектурой. Отдельно взятая архитектура может быть реализована с использованием различных вариантов микроархитектур с разным соотношением цены, производительности и потребляемой энергии, и такое соотношение зачастую выбирается как баланс между этими тремя факторами. Процессоры Intel Core i7, Intel 80486 и AMD Athlon, например, используют одну и ту же архитектуру x86, но реализованную с использованием трех разных микроархитектурных решений.

Теперь мы перемещаемся в область программного обеспечения. *Операционная система* (operating system) управляет операциями нижнего уровня, такими как доступ к жесткому диску или управление памятью. И, наконец, программное обеспечение использует ресурсы операционной системы для решения конкретных задач пользователя.

Именно принцип *абстрагирования от маловажных деталей* позволяет вашей бабушке общаться с внуками в Интернете, не задумываясь о квантовых колебаниях электронов или организации памяти компьютера.

Предмет этой книги — уровни абстракции от цифровых схем до компьютерной архитектуры. Работая на каком-либо из этих уровней абстракции, полезно знать кое-что и об уровнях абстракции, непосредственно сопряженных с тем уровнем, где вы находитесь. Программист, например, не сможет полностью оптимизировать код без понимания архитектуры процессора, который будет выполнять эту программу. Инженер-электронщик, разрабатывающий какой-либо блок микросхемы, не сможет найти компромисс между быстродействием и уровнем потребления энергии транзисторами, ничего не зная о той цифровой схеме, где этот блок будет использоваться. Мы надеемся, что к тому времени, когда вы закончите чтение этой книги, вы сможете выбрать уровень абстракции, необходимый для успешного выполнения любой стоящей перед вами задачи, и оценить влияние ваших инженерных решений на другие уровни абстракции в разрабатываемой вами системе.

### 1.2.2. Конструкторская дисциплина

*Конструкторская дисциплина* — это преднамеренное ограничение самим конструктором выбора возможных вариантов разработки, что позволяет работать продуктивнее на более высоком уровне абстракции. Использование взаимозаменяемых частей — это, вероятно, самый хорошо знакомый всем нам пример практического применения конструкторской дисциплины. Одним из первых примеров использования взаимозаменяемых деталей и узлов стала унификация при производстве кремневых ружей. До начала 19-го века такие ружья производились вручную и в штучном порядке. Высококвалифицированный оружейный мастер тщательно подтачивал и подгонял комплектующие, произведенные несколькими не связанными друг с другом ремесленниками. Конструкторская дисциплина для обеспечения взаимозаменяемости деталей и узлов произвела революцию в оружейной промышленности. Ограничение ассортимента комплектующих деталей до стандартного набора с жестко установленными допусками для каждой детали позволило собирать и ремонтировать ружья гораздо быстрее и использовать при этом менее квалифицированный персонал. Оружейный мастер перестал тратить свое время на разрешение проблем, связанных с нижними уровнями абстракции, такими как доводка какого-то конкретного ствола или исправление формы отдельного взятого приклада.

В контексте данной книги соблюдение конструкторской дисциплины в виде максимального использования цифровых схем играет очень важную роль. В цифровых схемах используются дискретные значения напряжения, в то время как в аналоговых схемах напряжение изменяется непрерывно. Таким образом, цифровые схемы, которые можно рассматривать как подмножество аналоговых цепей, в некотором смысле уступают по своим характеристикам более широкому классу аналоговых



цепей. Однако цифровые цепи гораздо проще проектировать. Ограничивая использование аналоговых схем и по возможности заменяя их цифровыми, мы можем легко объединять отдельные компоненты в сложные системы, которые, в конечном итоге, для большинства приложений превзойдут по своим параметрам системы, построенные на аналоговых цепях. Примером тому могут служить цифровые телевизоры, компакт-диски (CD) и мобильные телефоны, которые уже практически полностью вытеснили своих аналоговых предшественников.

Капитан Мериуззер Льюис — один из руководителей знаменитой экспедиции Льюиса и Кларка на северо-запад США, был, пожалуй, одним из самых ранних сторонников взаимозаменяемости. В 1806 году в своем дневнике, касаясь унификации деталей кремневых ружей того времени, он написал следующее:

«Ружья Дрюера и сержанта Прайора одновременно вышли из строя. На ружье Дрюера сломался ударно-спусковой механизм, и мы заменили его на новый. У ружья сержанта Прайора был сломан курковый винт, вместо которого мы поставили запасной курковый винт, заранее изготовленный специально для ударно-спускового механизма этого ружья на мануфактуре Харперс Фейри, где это оружие и было произведено. Если бы не предусмотрительность, заключающаяся в том, что мы заранее позаботились о запасных частях для ружей, и не мастерство Джона Шилдса, выполнившего всю работу, то большинство ружей нашей экспедиции к этому времени было бы полностью непригодно для какого-либо использования. И я имею полное право записать в своем дневнике, что, к счастью для нас, все наше оружие находится в прекрасном состоянии».

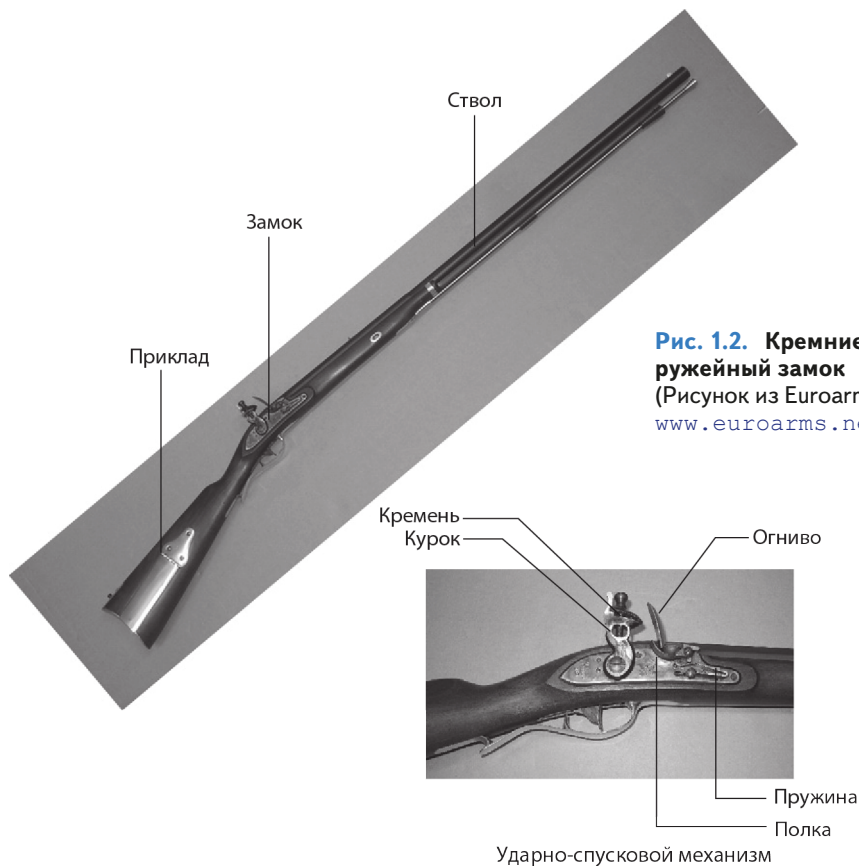
См. «История экспедиции Льюиса и Кларка» в четырех томах под редакцией Элиота Куэса. Первое издание: Харпер, Нью-Йорк, 1893; Переиздание: Довер, Нью-Йорк (3 тома), 3:817.

### 1.2.3. Три базовых принципа

В дополнение к абстрагированию от несущественных деталей и конструкторской дисциплине разработчики электронных систем используют еще три базовых принципа для управления сложностью системы: иерархичность, модульность конструкции и регулярность. Эти принципы применимы как к программному обеспечению, так и к аппаратной части компьютерных систем.

- ▶ *Иерархичность* — принцип иерархичности предполагает разделение системы на отдельные модули, а затем последующее разделение каждого такого модуля на фрагменты до уровня, позволяющего легко понять поведение каждого конкретного фрагмента.
- ▶ *Модульность* — принцип модульности требует, чтобы каждый модуль в системе имел четко определенную функциональность и набор интерфейсов и мог быть легко и без непредвиденных побочных эффектов соединен с другими модулями системы.
- ▶ *Регулярность* — принцип регулярности требует соблюдения единообразия при проектировании отдельных модулей системы. Стандартные модули общего назначения, например, такие как блоки питания, могут использоваться многократно, во много раз снижая количество модулей, необходимых для разработки новой системы.

Для иллюстрации трех базовых принципов вновь воспользуемся аналогией из оружейного производства. Нарезное кремневое ружье было одним из самых сложных устройств массового применения в начале 19-го века. Используя принцип иерархичности, мы можем разделить его на три главных модуля, как показано на **Рис. 1.2**: ствол, ударно-спусковой механизм и приклад с цевьем.



**Рис. 1.2. Кремниевый ружейный замок**  
(Рисунок из Euroarms Italia  
[www.euroarms.net](http://www.euroarms.net) © 2006 г.)

Ствол — это длинная металлическая труба, через которую при выстреле выбрасывается пуля. Ударно-спусковой механизм производит выстрел. Деревянные приклад и цевье соединяют воедино остальные части ружья и обеспечивают стрелку надежное удержание оружия при выстреле. В свою очередь, ударно-спусковой механизм включает в себя спусковой крючок, курок, кремень, огниво и пороховую полку. Каждый из этих компонентов также может рассматриваться как следующий иерархический уровень и может быть разделен на более мелкие детали.

Принцип модульности требует, чтобы каждый компонент выполнял четко определенную функцию и имел интерфейс. Функция приклада и цевья — служить базой для установки ствола и ударно-спускового механизма. Интерфейс для приклада и цевья — это их длина и расположение крепежных элементов, таких как винты или шурупы. Ствол ружья, изготовленного с соблюдением принципа модульности конструкции, может быть установлен на приклады и цевья от разных производителей, если все соединяемые части имеют правильную длину и подходящие

крепежные элементы. Функция ствола — разогнать пулю до необходимой скорости и придать ей вращение, чтобы увеличить точность стрельбы.<sup>1</sup> Принцип модульности требует также, чтобы при соединении модулей не возникало никаких побочных эффектов: конструкция приклада и цевья не должна препятствовать функционированию ствола.

Принцип регулярности учит тому, что взаимозаменяемые детали — это хорошая идея. При соблюдении принципа регулярности поврежденный ствол может быть с легкостью заменен на идентичный. Стволы могут изготавливаться на поточной линии с гораздо большей экономической эффективностью, чем в случае штучного производства.

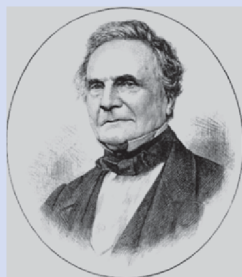
В данной книге мы будем постоянно возвращаться к этим трем базовым принципам: иерархичности, модульности и регулярности.

### 1.3. Цифровая абстракция

Большинство физических величин изменяется непрерывно. Например, напряжение в электрическом проводе, частота колебаний или распределение массы — все это параметры, изменяющиеся непрерывно. Цифровые системы, с другой стороны, представляют информацию в виде дискретно меняющихся переменных с конечным числом строго определенных значений.

Одной из наиболее ранних цифровых систем стала Аналитическая машина Чарльза Бэббиджа, которая использовала переменные с десятью дискретными значениями. Начиная с 1834 года и до 1871 года<sup>2</sup> Бэббидж разрабатывал и пытался построить этот механический компьютер. Шестеренки Аналитической машины могли находиться в одном из десяти фиксированных положений, а каждое такое положение было промаркировано от 0 до 9 подобно механическому счетчику пробега автомобиля. **Рис. 1.3** показывает, как выглядел прототип Аналитической машины. Каждый ряд шестеренок такой машины обрабатывал одну цифру. В своем механическом компьютере Бэббидж использовал 25 рядов шестеренок таким образом, чтобы машина обеспечивала вычисления с точностью до 25-го знака.

В отличие от машины Бэббиджа большинство электронных компьютеров использует двоичный (бинарный) код. В случае двоичного кода высокое напряжение — это единица, а низкое напряжение — нуль, поскольку гораздо легче оперировать двумя уровнями напряжения, чем десятью.

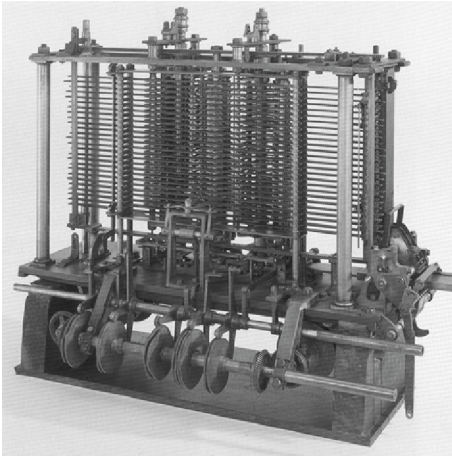


**Чарльз Бэббидж  
1791–1871**

Чарльз Бэббидж родился в 1791 году. Закончил Кембриджский университет и женился на Джорджиане Витмур. Он изобрел Аналитическую Машину — первый в мире механический компьютер. Чарльз Бэббидж также изобрел предохранительную решетку для локомотивов, спидометр и универсальный почтовый тариф. Ученый также очень интересовался отмычками для замков и почему-то ненавидел уличных музыкантов. (Портрет любезно предоставлен Fourmilab Швейцария, [www.fourmilab.ch](http://www.fourmilab.ch)).

<sup>1</sup> Кремневые ружья не были нарезными и использовали круглые пули. — *Прим. перевод.*

<sup>2</sup> А большинству из нас кажется, что обучение в университете — это так долго!



**Рис. 1.3.** Аналитическая машина Беббиджа в год его смерти (1871)

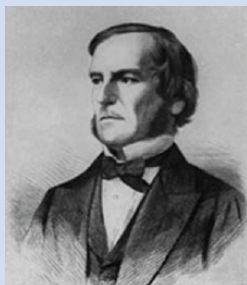
(Изображение любезно предоставлено музеем Науки и общества)

Объем информации  $D$ , передаваемый одной дискретной переменной, которая может находиться в  $N$  различных состояниях, измеряется в единицах, называемых *битами*, и вычисляется по следующей формуле:

$$D = \log_2 N \text{ бит} \quad (1.1)$$

Двоичная переменная передает  $\log_2 2 = 1$  — один бит информации. Теперь вам, вероятно, понятно, почему единица информации называется битом. Бит (bit) — это сокращение от английского *binary digit*, что дословно переводится как *двоичный разряд*. Каждая шестеренка в машине Беббиджа содержит  $\log_2 10 = 3.322$  бит информации, поскольку она может находиться в одном из  $2^{3.322} = 10$  уникальных положений. Теоретически непрерывный сигнал может передавать бесконечное количество информации, поскольку может принимать неограниченное число значений. На практике, однако, шум и ошибки измерения ограничивают информацию, передаваемую большинством непрерывных сигналов, диапазоном от 10 бит до 16 бит. Если же измерение уровня сигнала должно быть произведено очень быстро, то объем передаваемой информации будет еще ниже (в случае 10 бит, например, это будет только 8 бит).

Предмет этой книги — цифровые схемы, использующие двоичные переменные ноль и единицу. Джордж Буль разработал систему логики, использующую двоичные переменные, и эту систему сегодня называют его именем — *булева логика*. Булевы переменные могут принимать значения *ИСТИНА* (TRUE) или *ЛОЖЬ* (FALSE). В электронных компьютерах положительное напряжение обычно представляет единицу, а нулевое напряжение представляет ноль. В этой книге мы будем использовать понятия единица (1), ИСТИНА (TRUE) и ВЫСОКОЕ (HIGH) как синонимы. Аналогичным образом мы будем использовать ноль (0), ЛОЖЬ (FALSE) и НИЗКОЕ (LOW) как взаимозаменяемые термины.



**Джордж Буль**  
**1815–1864**

Джордж Буль родился в семье небогатого ремесленника. Родители Джорджа не могли оплатить его формального образования, поэтому он осваивал математику самоучкой. Несмотря на это, Булю удалось стать преподавателем Королевского колледжа в Ирландии. В 1854 году Джордж Буль написал свою работу Исследование законов мышления, которая впервые ввела в научный оборот двоичные переменные, а также три основных логических оператора И, ИЛИ, НЕ (AND, OR, NOT). (Портрет любезно предоставлен Американским физическим институтом).

Преимущества *цифровой абстракции* заключаются в том, что разработчик цифровой системы может сосредоточиться исключительно на единицах и нулях, полностью игнорируя, каким образом булевы переменные представлены на физическом уровне. Разработчика не волнует, представлены ли нули и единицы определенными значениями напряжения, вращающимися шестернями или уровнем гидравлической жидкости. Программист может продуктивно работать, не располагая детальной информацией об аппаратном обеспечении компьютера. Однако, понимание того, как работает это аппаратное обеспечение, позволяет программисту гораздо лучше оптимизировать программу для конкретного компьютера.

Как вы могли видеть выше, один-единственный бит не может передать большого количества информации. Поэтому в следующем разделе мы рассмотрим вопрос о том, каким образом набор битов можно использовать для представления десятичных чисел. В последующих главах мы также покажем, как группы битов могут представлять буквы и даже целую программу.

## 1.4. Системы счисления

Мы все привыкли работать с десятичными числами. Однако, в цифровых системах, построенных на единицах и нулях, использование двоичных или шестнадцатеричных чисел зачастую более удобно. В данном разделе мы рассмотрим системы счисления, использованные в этой книге.

### 1.4.1. Десятичная система счисления

Еще в начальной школе нас всех научили считать и выполнять различные арифметические операции в *десятичной* (decimal) системе счисления. Такая система использует десять арабских цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 – столько же, сколько у нас пальцев на руках. Числа больше 9 записываются в виде строки цифр. Причем, цифра, находящаяся в каждой последующей позиции такой строки, начиная с крайней правой цифры, имеет «вес», в десять раз превышающий «вес» цифры, находящейся в предыдущей позиции. Именно поэтому десятичную систему счисления называют *системой по основанию* (base) 10. Справа налево «вес» каждой позиции увеличивается следующим образом: 1, 10, 100, 1000 и т. д. Позицию, которую цифра занимает в строке десятичного числа, называют разрядом или декадой.

Чтобы избежать недоразумений при одновременной работе с более чем одной системой счисления, основание системы обычно указывается путем добавления цифры позади и чуть ниже основного числа:  $9742_{10}$ . **Рис. 1.4** показывает, для примера, как десятичное число  $9742_{10}$  может быть записано в виде суммы цифр, составляющих это число, умноженных на «вес» разряда, соответствующего каждой конкретной цифре.

$$\begin{array}{cccc}
 \text{Колонка тысяч} & \text{Колонка сотен} & \text{Колонка десятков} & \text{Колонка единиц} \\
 9 & 7 & 4 & 2 \\
 \times 10^3 & \times 10^2 & \times 10^1 & \times 10^0 \\
 \hline
 9 \times 10^3 & + 7 \times 10^2 & + 4 \times 10^1 & + 2 \times 10^0
 \end{array}$$

Девять тысяч      Семь сотен      Четыре десятки      Две единицы

**Рис. 1.4.** Представление десятичного числа

$N$ -разрядное десятичное число может представлять одну из  $10^N$  цифровых комбинаций: 0, 1, 2, 3, ...  $10^N - 1$ . Это называется диапазоном  $N$ -разрядного числа. Десятичное число, состоящее из трех цифр (разрядов), например, представляет одну из 1000 возможных цифровых комбинаций в диапазоне от 0 до 999.

## 1.4.2. Двоичная система счисления

Одиночный бит может принимать одно из двух значений, 0 или 1. Несколько битов, соединенных в одной строке, образуют *двоичное* (binary) число. Каждая последующая позиция в двоичной строке имеет вдвое больший «вес», чем предыдущая позиция, так что двоичная система счисления – это система по основанию 2. В двоичном числе «вес» каждой позиции увеличивается (так же, как и в десятичном – справа налево) следующим образом: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536 и т.д. Работая с двоичными числами, очень полезно для сохранения времени запомнить значения степеней двойки до  $2^{16}$ .

Произвольное  $N$ -разрядное двоичное число может представлять одну из  $2^N$  цифровых комбинаций: 0, 1, 2, 3, ...  $2^N - 1$ . В **Табл. 1.1** собраны 1-битные, 2-битные, 3-битные и 4-битные двоичные числа и их десятичные эквиваленты.

### Пример 1.1. ПРЕОБРАЗОВАНИЕ ЧИСЕЛ ИЗ ДВОИЧНОЙ СИСТЕМЫ СЧИСЛЕНИЯ В ДЕСЯТИЧНУЮ

Преобразовать двоичное число  $10110_2$  в десятичное.

**Решение:** Нужные преобразования представлены на **Рис. 1.5**.



Рис. 1.5.  
Преобразование  
двоичного числа  
в десятичное число

Колонка единиц  
Колонка двоек  
Колонка четверок  
Колонка восьмерок  
Колонка шестнадцати

$$10110_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 22_{10}$$

Одна шестнадцать    Нет восемь    Одна четыре    Одна двойка    Нет единиц

Табл. 1.1. Таблица двоичных чисел и их десятичный эквивалент

1-битные двоичные числа	2-битные двоичные числа	3-битные двоичные числа	4-битные двоичные числа	Десятичные эквиваленты
0	00	000	0000	0
1	01	001	0001	1
	10	010	0010	2
	11	011	0011	3
		100	0100	4
		101	0101	5
		110	0110	6
		111	0111	7
			1000	8
			1001	9
			1010	10
			1011	11
			1100	12
			1101	13
			1110	14
			1111	15

Пример 1.2. ПРЕОБРАЗОВАНИЕ ЧИСЕЛ ИЗ ДЕСЯТИЧНОЙ СИСТЕМЫ СЧИСЛЕНИЯ В ДВОИЧНУЮ

Преобразовать десятичное число  $84_{10}$  в двоичное.

**Решение:** Определите, что должно стоять в каждой позиции двоичного результата: 1 или 0. Вы можете делать это, начиная с левой или правой позиции.

Если начать слева, найдите наибольшую степень 2, меньше или равную заданному числу (в примере такая степень – это 64).  $84 > 64$ , поэтому ставим 1 в позиции, соответствующей 64. Остается  $84 - 64 = 20$ ,  $20 < 32$ , так что в позиции 32 надо поставить 0,  $20 > 16$ , поэтому в позиции 16 ставим 1. Остается  $20 - 16 = 4$ .  $4 < 8$ , поэтому 0 в позиции 8.  $4 \geq 4$  – ставим 1 в позицию 4.  $4 - 4 = 0$ , поэтому будут 0 в позициях 2 и 1. Сбрав все вместе, получаем  $84_{10} = 1010100_2$ .

Если начать справа, будем последовательно делить исходное число на 2. Остаток идет в очередную позицию.  $84/2 = 42$ , поэтому 0 в самой правой позиции.  $42/2 = 21$ , 0 во вторую позицию.  $21/2 = 10$ , остаток 1 идет в позицию, со-

ответствующую 4.  $10/2 = 5$ , поэтому 0 в позицию, соответствующую 8.  $5/2 = 2$ , остаток 1 в позицию 16.  $2/2 = 1$ , 0 в 32 позицию. Наконец,  $1/2 = 0$  с остатком 1, который идет в позицию 64. Снова,  $84_{10} = 1010100_2$

### 1.4.3. Шестнадцатеричная система счисления

Использование длинных двоичных чисел для записи и выполнения математических расчетов на бумаге утомительно и чревато ошибками. Однако длинное двоичное число можно разбить на группы по четыре бита, каждая из которых представляет одну из  $2^4 = 16$  цифровых комбинаций. Именно поэтому зачастую бывает удобнее использовать для работы систему счисления по основанию 16, называемую *шестнадцатеричной* (hexadecimal). Для записи шестнадцатеричных чисел используются цифры от 0 до 9 и буквы от A до F, как показано в Табл. 1.2 В шестнадцатеричном числе «вес» каждой позиции меняется следующим образом: 1,  $16$ ,  $16^2$  (или 256),  $16^3$  (или 4096) и т. д.

Интересно, что термин *hexadecimal* (шестнадцатеричный) введен в научный обиход корпорацией IBM в 1963 году и является комбинацией греческого слова *hexi* (шесть) и латинского *deset* (десять). Правильнее было бы использовать латинское же слово *sexa* (шесть), но термин *sexadecimal* воспринимался бы несколько неоднозначно.

Табл. 1.2. Шестнадцатеричная система счисления

Шестнадцатеричная цифра	Десятичный эквивалент	Двоичный эквивалент
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

#### Пример 1.3. ПРЕОБРАЗОВАНИЕ ШЕСТИНАДЦАТЕРИЧНОГО ЧИСЛА В ДВОИЧНОЕ И ДЕСЯТИЧНОЕ

Преобразовать шестнадцатеричное число  $2ED_{16}$  в двоичное и десятичное.

**Решение:** Преобразование шестнадцатеричного числа в двоичное и обратно — очень простое, так как каждая шестнадцатеричная цифра прямо соответ-



ствует 4-разрядному двоичному числу.  $2_{16} = 0010_2$ ,  $E_{16} = 1110_2$  и  $D_{16} = 1101_2$ , так что  $2ED_{16} = 001011101101_2$ . Преобразование в десятичную систему счисления требует арифметики, показанной на **Рис. 1.6**.

$$2ED_{16} = 2 \times 16^2 + E \times 16^1 + D \times 16^0 = 749_{10}$$

Две Четырнадцать Тринадцать  
двести шестнадцать единиц  
пятьдесят  
шесть

Колонка единицы  
 Колонка шестнадцать  
 Колонка двести пятьдесят шесть

**Рис. 1.6.** Преобразование шестнадцатеричного числа в десятичное число

#### Пример 1.4. ПРЕОБРАЗОВАНИЕ ДВОИЧНОГО ЧИСЛА В ШЕСТНАДЦАТЕРИЧНОЕ

Преобразовать двоичное число  $1111010_2$  в шестнадцатеричное.

**Решение:** Повторим еще раз, это просто. Начинаем справа. 4 наименее значимых бита  $1010_2 = A_{16}$ . Следующие биты  $111_2 = 7_{16}$ . Отсюда  $1111010_2 = 7A_{16}$ .

#### Пример 1.5. ПРЕОБРАЗОВАНИЕ ДЕСЯТИЧНОГО ЧИСЛА В ШЕСТНАДЦАТЕРИЧНОЕ И ДВОИЧНОЕ

Преобразовать десятичное число  $333_{10}$  в шестнадцатеричное и двоичное.

**Решение:** Как и в случае преобразования десятичного числа в двоичное, можно начать как слева, так и справа.

Если начать слева, найдите наибольшую степень шестнадцати, меньшую или равную заданному числу (в нашем случае это  $16^2 = 256$ ). Число 256 содержится в числе 333 только один раз, поэтому в позицию с «весом» 256 мы записываем единицу. Остается число  $333 - 256 = 77$ . Число 16 содержится в числе 77 четыре раза, поэтому в позицию с «весом» 16 записываем четверку. Остается  $77 - 16 \times 4 = 13$ .  $13_{10} = D_{16}$ , поэтому в позицию с «весом» 1 записываем цифру D. Итак,  $333_{10} = 14D_{16}$ , это число легко преобразовать в двоичное, как мы показали в примере 1.3:  $14D_{16} = 101001101_2$ .

Если начинать справа, будем повторять деление на 16. Каждый раз остаток идет в очередную колонку.  $333/16 = 20$  с остатком  $13_{10} = D_{16}$ , который идет в самую правую позицию.  $20/16 = 1$  с остатком 4, который идет в позицию с «весом» 16.  $1/16 = 0$  с остатком 1, который идет в позицию с «весом» 256. В результате опять получаем  $14D_{16}$ .