

А. П. Жмакин

$$y := \frac{x^2 + 72x - 6400}{-168}$$

$$y := (x - 11)^2 - 1$$

ОР-IP

АРХИТЕКТУРА ЭВМ



■ *Функциональная организация ЭВМ*

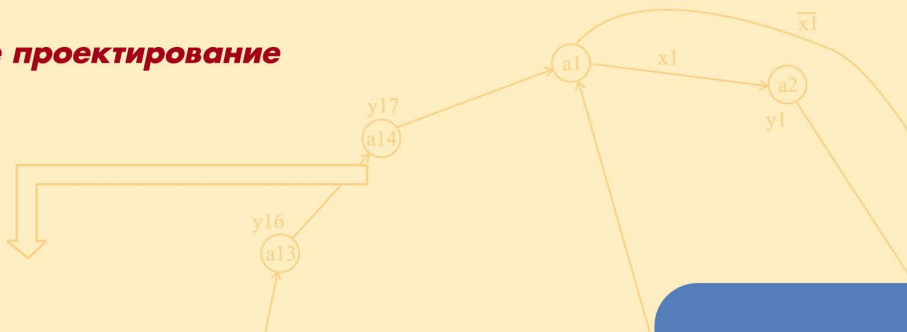
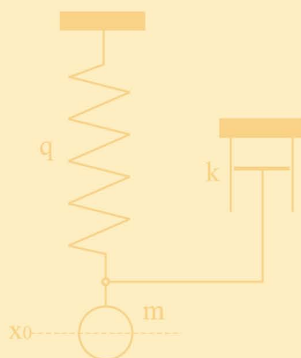
■ *Машинная арифметика и синтез устройств*

■ *Архитектура микропроцессорных систем*

■ *Программная модель учебной ЭВМ*

■ *Лабораторный практикум*

■ *Курсовое проектирование*



УЧЕБНОЕ ПОСОБИЕ



УДК 681.3(075.8)
ББК 32.973-02я73
Ж77

Жмакин А. П.

Ж77 Архитектура ЭВМ. — СПб.: БХВ-Петербург, 2006. — 320 с.: ил.
ISBN 5-94157-719-2

Пособие объединяет в одном издании теоретическую часть одноименной дисциплины и лабораторный практикум. Рассмотрены базовые вопросы организации ЭВМ: функциональная организация ЭВМ, системы команд и командный цикл. Большое внимание уделено арифметическим основам ЭВМ, принципам построения различных устройств и их взаимодействию. Обсуждаются вопросы построения микропроцессорных систем. Лабораторный практикум проводится на программной модели ЭВМ, представленной на прилагаемом компакт-диске. Также пособие содержит материалы для выполнения курсового проектирования.

Для студентов и преподавателей технических вузов

УДК 681.3(075.8)
ББК 32.973-02я73

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Людмила Еремеевская</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Анна Кузьмина</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн серии	<i>Игоря Цырульниковца</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Рецензенты:

Терехов А. Н., д. ф.-м. н., профессор,
заведующий кафедрой системного программирования
Санкт-Петербургского государственного университета
Костин В. А., к. ф.-м. н., доцент кафедры информатики
Санкт-Петербургского государственного университета

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 12.12.05.

Формат 70×100¹/₁₆. Печать офсетная. Усл. печ. л. 25,8.

Тираж 3000 экз. Заказ №

"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-719-2

© Жмакин А. П., 2006
© Оформление, издательство "БХВ-Петербург", 2006

Оглавление

Предисловие	9
ЧАСТЬ I. ПРИНЦИПЫ ОРГАНИЗАЦИИ ЭВМ	11
Глава 1. Начальные сведения об ЭВМ	13
1.1. История развития вычислительной техники.....	13
1.2. Цифровые и аналоговые вычислительные машины.....	15
1.3. Варианты классификации ЭВМ.....	16
1.4. Классическая архитектура ЭВМ.....	20
1.5. Иерархическое описание ЭВМ	21
Глава 2. Функциональная организация ЭВМ.....	25
2.1. Командный цикл процессора.....	25
2.2. Система команд процессора.....	27
2.2.1. Форматы команд.....	27
2.2.2. Способы адресации	28
2.2.3. Система операций.....	30
Глава 3. Арифметические основы ЭВМ	33
3.1. Системы счисления.....	34
3.2. Представление чисел в различных системах счисления.....	37
3.2.1. Перевод целых чисел из одной системы счисления в другую	37
Преобразование $Z_p \rightarrow Z_1 \rightarrow Z_q$	37
Преобразование $Z_p \rightarrow Z_w \rightarrow Z_q$	38
3.2.2. Перевод дробных чисел из одной системы счисления в другую	41
3.2.3. Перевод чисел между системами счисления $2 \leftrightarrow 8 \leftrightarrow 16$	43
3.2.4. Понятие экономичности системы счисления.....	45
3.3. Представление информации в ЭВМ. Прямой код.....	47

3.4. Алгебраическое сложение/вычитание в прямом коде	48
3.5. Обратный код и выполнение алгебраического сложения в нем	50
3.5.1. Алгебраическое сложение в обратном коде	51
3.6. Дополнительный код и арифметические операции в нем	56
3.6.1. Алгебраическое сложение в дополнительном коде	57
3.6.2. Модифицированные обратный и дополнительный коды	61
3.7. Алгоритмы алгебраического сложения в обратном и дополнительном коде	62
3.8. Алгоритмы умножения	64
3.8.1. Умножение в дополнительном коде	66
3.8.2. Методы ускорения умножения	66
3.9. Алгоритмы деления	70
3.9.1. Деление без восстановления остатка	71
3.10. Арифметические операции с числами, представленными в формате с плавающей запятой	72
3.10.1. Сложение и вычитание	74
3.10.2. Умножение и деление	77
3.11. Арифметические операции над десятичными числами	78
3.11.1. Кодирование десятичных чисел	78
3.11.2. Арифметические операции над десятичными числами	79
3.12. Машинная арифметика в остаточных классах	83
3.12.1. Представление чисел в системе остаточных классов	83
3.12.2. Арифметические операции с положительными числами	84
3.12.3. Арифметические операции с отрицательными числами	87
Глава 4. Организация устройств ЭВМ	89
4.1. Принцип микропрограммного управления	89
4.2. Концепция операционного и управляющего автоматов	90
4.3. Операционный автомат	91
4.3.1. Пример проектирования операционного автомата АЛУ	92
Определение форматов данных	92
Разработка алгоритма деления	93
Разработка структуры операционного автомата	95
4.4. Управляющий автомат	99
4.4.1. Управляющий автомат с "жесткой" логикой	99
Пример проектирования УАЖЛ	100
4.4.2. Управляющий автомат с программируемой логикой	107
Принципы организации	107
Адресация микрокоманд	109
Кодирование микроопераций	114
Пример проектирования УАПЛ	117
Глава 5. Организация памяти в ЭВМ	125
5.1. Концепция многоуровневой памяти	125
5.2. Сверхоперативная память	127
5.2.1. СОЗУ с прямым доступом	128
5.2.2. СОЗУ с ассоциативным доступом	128

5.3. Виртуальная память	136
5.3.1. Алгоритмы замещения	137
5.3.2. Сегментная организация памяти.....	139

ЧАСТЬ II. АРХИТЕКТУРА МИКРОПРОЦЕССОРНЫХ СИСТЕМ.....141

Глава 6. Базовая архитектура микропроцессорной системы147

6.1. Процессорный модуль	148
6.1.1. Внутренняя структура микропроцессора.....	148
6.1.2. Командный и машинный циклы микропроцессора.....	150
6.1.3. Реализация процессорных модулей и состав линий системного интерфейса	152
6.2. Машина пользователя и система команд.....	154
6.2.1. Распределение адресного пространства.....	155
6.2.2. Система команд i8086	156
6.3. Функционирование основных подсистем МПС	158
6.3.1. Оперативная память	160
Диспетчер памяти	160
6.3.2. Ввод/вывод	161
Параллельный обмен	161
Последовательный обмен.....	166
6.3.3. Прерывания	168
Обнаружение изменения состояния внешней среды	170
Идентификация источника прерывания.....	170
Приоритет запросов.....	171
Приоритет программ	171
Обработка прерывания.....	172
6.3.4. Прямой доступ в память	175

Глава 7. Эволюция архитектур микропроцессоров и микроЭВМ.....177

7.1. Защищенный режим и организация памяти.....	178
7.1.1. Сегментная организация памяти.....	178
7.1.2. Страничная организация памяти.....	183
7.1.3. Защита памяти.....	186
Защита памяти на уровне сегментов	187
Защита доступа к данным	189
Защита сегментов кода.....	189
Защита памяти на уровне страниц.....	191
7.2. Мультизадачность.....	192
7.2.1. Сегмент состояния задачи	193
7.2.2. Переключение задачи.....	196
7.3. Прерывания и особые случаи.....	198
7.3.1. Deskрипторная таблица прерываний.....	202
7.3.2. Учет уровня привилегий	204

7.3.3. Код ошибки	204
7.3.4. Описание особых случаев.....	205
7.4. Средства отладки	209
7.4.1. Регистры отладки.....	211
Регистрация нескольких особых случаев	215
7.5. Увеличение быстродействия процессора.....	215
7.5.1. Конвейеры	216
7.5.2. Динамический параллелизм	219
7.5.3. VLIW-архитектура.....	223
Выводы	225
7.6. Однокристалльные микроЭВМ	227

ЧАСТЬ III. ЛАБОРАТОРНЫЙ ПРАКТИКУМ И КУРСОВОЕ ПРОЕКТИРОВАНИЕ.....233

Глава 8. Описание архитектуры учебной ЭВМ235

8.1. Структура ЭВМ.....	235
8.2. Представление данных в модели	238
8.3. Система команд.....	238
8.3.1. Форматы команд.....	238
8.3.2. Способы адресации	239
8.3.3. Система операций.....	240
8.4. Состояния и режимы работы ЭВМ.....	240
8.5. Интерфейс пользователя	241
8.5.1. Окна основных обозревателей системы.....	242
Окно <i>Процессор</i>	242
Окно <i>Память</i>	244
Окно <i>Текст программы</i>	245
Окно <i>Программа</i>	246
Окно <i>Микрокомандный уровень</i>	248
Окно <i>Кэш-память</i>	248
8.6. Внешние устройства	248
8.6.1. Контроллер клавиатуры	250
8.6.2. Дисплей.....	253
8.6.3. Блок таймеров	255
8.6.4. Тоногенератор	257
8.7. Подсистема прерываний.....	257
8.8. Программная модель кэш-памяти	259
8.9. Вспомогательные таблицы.....	262

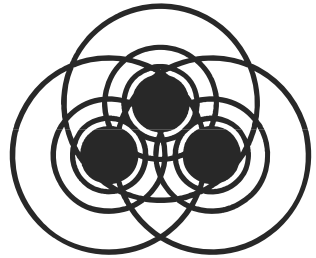
Глава 9. Лабораторные работы.....267

9.1. Лабораторная работа № 1. Архитектура ЭВМ и система команд.....	267
9.1.1. Общие положения.....	267
9.1.2. Пример 1	268

9.1.3. Задание 1	269
9.1.4. Содержание отчета	270
9.1.5. Контрольные вопросы.....	270
9.2. Лабораторная работа № 2. Программирование разветвляющегося процесса	271
9.2.1. Пример 2	271
9.2.2. Задание 2.....	273
9.2.3. Содержание отчета	275
9.2.4. Контрольные вопросы.....	275
9.3. Лабораторная работа № 3. Программирование цикла с переадресацией	275
9.3.1. Пример 3	275
9.3.2. Задание 3.....	277
9.3.3. Содержание отчета	278
9.3.4. Контрольные вопросы.....	278
9.4. Лабораторная работа № 4. Подпрограммы и стек	279
9.4.1. Пример 4	280
9.4.2. Задание 4.....	282
9.4.3. Содержание отчета	283
9.4.4. Контрольные вопросы.....	283
9.5. Лабораторная работа № 5. Командный цикл процессора.....	283
9.5.1. Задание 5.1.....	284
9.5.2. Задание 5.2.....	284
9.5.3. Контрольные вопросы.....	284
9.6. Лабораторная работа № 6. Программирование внешних устройств	286
9.6.1. Задание 6.....	286
9.6.2. Задания повышенной сложности	288
9.6.3. Порядок выполнения работы	289
9.6.4. Содержание отчета	289
9.6.5. Контрольные вопросы.....	289
9.7. Лабораторная работа № 7. Принципы работы кэш-памяти.....	290
9.7.1. Задание 7	290
9.7.2. Порядок выполнения работы	291
9.7.3. Содержание отчета	292
9.7.4. Контрольные вопросы.....	292
9.8. Лабораторная работа № 8. Алгоритмы замещения строк кэш-памяти.....	292
9.8.1. Задание 8.....	293
9.8.2. Порядок выполнения работы	293
9.8.3. Содержание отчета	294
9.8.4. Контрольные вопросы.....	294
Глава 10. Курсовая работа	295
10.1. Цель и содержание работы.....	295
10.2. Задания.....	295
10.3. Этапы выполнения работы	298
10.4. Содержание пояснительной записки.....	300

ПРИЛОЖЕНИЯ	303
Приложение 1. Список сокращений, используемых в тексте	305
Приложение 2. Описание компакт-диска.....	307
Литература	309
Предметный указатель	311

ГЛАВА 3



Арифметические основы ЭВМ

Безусловно, одним из основных направлений применения компьютеров были и остаются разнообразные вычисления. Обработка числовой информации ведется и при решении задач, на первый взгляд не связанных с какими-то расчетами, например, при использовании компьютерной графики или звука.

В связи с этим встает вопрос о выборе *оптимального представления чисел в компьютере*. Безусловно, можно было бы использовать 8-битное (байтовое) кодирование отдельных цифр, а из них составлять числа. Однако такое кодирование не будет оптимальным, что легко увидеть из простого примера. Пусть имеется двузначное число 13. При 8-битном кодировании отдельных цифр в кодах ASCII его представление выглядит следующим образом: 0011000100110011, т. е. код имеет длину 16 битов; если же определять это число просто в двоичном коде, то получим 4-битную цепочку 1101.

Важно, что представление определяет не только способ записи данных (букв или чисел), но и допустимый набор операций над ними; в частности, буквы могут быть только помещены в некоторую последовательность (или исключены из нее) без изменения их самих; над числами же возможны *операции*, изменяющие само число, например, извлечение корня или сложение с другим числом.

Представление чисел в компьютере имеет две особенности:

- числа записываются в двоичной системе счисления (в отличие от привычной десятичной);
- для записи и обработки чисел отводится конечное количество разрядов (в "некомпьютерной" арифметике такое ограничение отсутствует).

Следствия, к которым приводят эти отличия, и рассматриваются в данной главе.

3.1. Системы счисления

Начнем с некоторых общих замечаний относительно понятия *число* [12]. Можно считать, что любое число имеет значение (содержание) и форму представления.

Значение числа задает его отношение к значениям других чисел ("больше", "меньше", "равно") и, следовательно, порядок расположения чисел на числовой оси. Форма представления, как следует из названия, определяет порядок записи числа с помощью предназначенных для этого знаков. При этом значение числа является инвариантом, т. е. не зависит от способа его представления. Это означает также, что число с одним и тем же значением может быть записано по-разному, т. е. отсутствует взаимно однозначное соответствие между представлением числа и его значением.

В связи с этим возникают вопросы, во-первых, о формах представления чисел и, во-вторых, о возможности и способах перехода от одной формы к другой.

Способ представления числа определяется *системой счисления*.

Определение

Система счисления — это правило записи чисел с помощью заданного набора специальных знаков — цифр.

Людьми использовались различные способы записи чисел, которые можно объединить в несколько групп: унарная, непозиционные и позиционные.

Унарная — это система счисления, в которой для записи чисел используется только один знак — | (вертикальная черта, палочка). Следующее число получается из предыдущего добавлением новой палочки: их количество (сумма) равно самому числу. Унарная система важна в теоретическом отношении, поскольку в ней число представляется наиболее простым способом и, следовательно, просты операции с ним. Кроме того, именно унарная система определяет значение целого числа количеством содержащихся в нем единиц, которое, как было сказано, не зависит от формы представления.

Из *непозиционных* наиболее распространенной можно считать римскую систему счисления. В ней некоторые базовые числа обозначены заглавными латинскими буквами: 1 — I, 5 — V, 10 — X, 50 — L, 100 — C, 500 — D, 1000 — M. Все другие числа строятся комбинаций базовых в соответствии со следующими правилами:

- если цифра меньшего значения стоит справа от большей цифры, то их значения суммируются; если слева — то меньшее значение вычитается из большего;

- цифры I, X, C и M могут следовать подряд не более трех раз каждая;
- цифры V, L и D могут использоваться в записи числа не более одного раза.

Например, запись XIX соответствует числу 19, MDXLIX — числу 1549. Запись чисел в такой системе громоздка и неудобна, но еще более неудобным оказывается выполнение в ней даже самых простых арифметических операций. Отсутствие нуля и знаков для чисел больше M не позволяют римскими цифрами записать любое число (хотя бы натуральное). По указанным причинам теперь римская система используется лишь для нумерации.

В настоящее время для представления чисел применяют, в основном, *позиционные системы счисления*.

Определение

Позиционными называются системы счисления, в которых значение каждой цифры в изображении числа определяется ее положением (позицией) в ряду других цифр.

Наиболее распространенной и привычной является система счисления, в которой для записи чисел используется 10 цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8 и 9. Число представляет собой краткую запись многочлена, в который входят степени некоторого другого числа — основания системы счисления. Например:

$$575,15 = 5 \cdot 10^2 + 7 \cdot 10^1 + 5 \cdot 10^0 + 1 \cdot 10^{-1} + 5 \cdot 10^{-2}.$$

В данном числе цифра 5 встречается трижды, однако значение этих цифр различно и определяется их положением (позицией) в числе. Количество цифр для построения чисел, очевидно, равно основанию системы счисления. Также очевидно, что максимальная цифра на 1 меньше основания. Причина широкого распространения именно десятичной системы счисления понятна — она происходит от унарной системы с пальцами рук в качестве "палочек".

Однако в истории человечества имеются свидетельства использования и других систем счисления — пятеричной, шестеричной, двенадцатиричной, двадцатиричной и даже шестидесятиричной. Общим для унарной и римской систем счисления является то, что значение числа в них определяется посредством операций сложения и вычитания базисных цифр, из которых составлено число, *независимо от их позиции в числе*. Такие системы получили название *аддитивных*.

В отличие от них позиционное представление следует считать *аддитивно-мультипликативным*, поскольку значение числа определяется операциями умножения и сложения. Главной же особенностью позиционного представле-

ния является то, что в нем посредством конечного набора знаков (цифр, разделителя десятичных разрядов и обозначения знака числа) можно записать неограниченное количество различных чисел. Кроме того, в позиционных системах гораздо легче, чем в аддитивных, осуществляются операции умножения и деления. Именно эти обстоятельства обуславливают доминирование позиционных систем при обработке чисел как человеком, так и компьютером.

По принципу, положенному в основу десятичной системы счисления, очевидно, можно построить системы с иным основанием. Пусть p — основание системы счисления. Тогда любое число Z (пока ограничимся только целыми числами), удовлетворяющее условию $Z < p^k$ ($k > 0$, целое), может быть представлено в виде многочлена со степенями (при этом, очевидно, максимальный показатель степени будет равен $k - 1$):

$$Z_p = a_{k-1} \cdot p^{k-1} + a_{k-2} \cdot p^{k-2} + \dots + a_1 \cdot p^1 + a_0 \cdot p^0 = \sum_{j=1}^{k-1} a_j \cdot p^j. \quad (3.1)$$

Из коэффициентов a_j при степенях основания строится сокращенная запись числа:

$$Z_p = (a_{k-1} a_{k-2} \dots a_1 a_0).$$

Индекс p числа Z указывает, что оно записано в системе счисления с основанием p : общее число цифр числа равно k . Все коэффициенты a_j — целые числа, удовлетворяющие условию: $0 < a_j < p - 1$.

Уместно задаться вопросом: каково минимальное значение p ? Очевидно, $p = 1$ невозможно, поскольку тогда все $a_j = 0$ и форма (3.1) теряет смысл. Первое допустимое значение $p = 2$ — оно и является минимальным для позиционных систем.

Система счисления с основанием 2 называется *двоичной*. Цифрами двоичной системы являются 0 и 1, а форма (3.1) строится по степеням 2. Интерес именно к этой системе счисления связан с тем, что, как указывалось выше, любая информация в компьютерах представляется с помощью двух состояний — 0 и 1, которые легко реализуются технически.

Наряду с двоичной в компьютерах используются восьмеричная и шестнадцатеричная системы счисления — причины будут рассмотрены далее.

3.2. Представление чисел в различных системах счисления

Очевидно, что значение целого числа, т. е. общее количество входящих в него единиц, не зависит от способа его представления и остается одинаковым во всех системах счисления; различаются только *формы представления* одного и того же количественного содержания числа.

Например: $|||||_1 = 5_{10} = 101_2 = 5_{16}$.

Поскольку одно и то же число может быть записано в различных системах счисления, встает вопрос о переводе представления числа из одной системы в другую.

3.2.1. Перевод целых чисел из одной системы счисления в другую

Обозначим преобразование числа Z , представленного в p -ричной системе счисления в представление в q -ричной системе как $Z_p \rightarrow Z_q$. Теоретически возможно произвести его при любых q и p . Однако подобный прямой перевод будет затруднен тем, что придется выполнять операции по правилам арифметики недесятичных систем счисления (полагая в общем случае, что $p, q \neq 10$).

По этой причине более удобными с практической точки зрения оказываются варианты преобразования с промежуточным переводом $Z_p \rightarrow Z_r \rightarrow Z_q$ с основанием r , для которого арифметические операции выполнить легко. Такими удобными основаниями являются $r = 1$ и $r = 10$, т. е. перевод осуществляется через унарную или десятичную систему счисления.

Преобразование $Z_p \rightarrow Z_1 \rightarrow Z_q$

Идея алгоритма перевода предельно проста: положим начальное значение $Z_q := 0$; из числа Z_p вычтем 1 по правилам вычитания системы p , т. е. $Z_p := Z_p - 1$, и добавим ее к Z_q по правилам сложения системы q , т. е. $Z_q := Z_q + 1$. Будем повторять эту последовательность действий, пока не достигнем $Z_p = 0$. Правила сложения с 1 (инкремента) и вычитания 1 (декремента) могут быть записаны так, как представлено в табл. 3.1.

Таблица 3.1. Правила сложения и вычитания 1

Для системы p	Для системы q
$(p-1)-1 = p-2$	$0+1=1$
$(p-2)-1 = p-3$	$1+1=2$
...	...
$1-1=0$	$(q-2)+1 = q-1$
$0-1 = \pi(p-1)$	$(q-1)+1 = \pi.0$

Примечание: π — перенос в случае инкремента или заем в случае декремента.

Промежуточный переход к унарной системе счисления в данном случае осуществляется неявно — используется упоминавшееся выше свойство независимости значения числа от формы его представления. Рассмотренный алгоритм перевода может быть легко реализован программным путем.

Преобразование $Z_p \rightarrow Z_w \rightarrow Z_q$

Очевидно, первая и вторая часть преобразования не связаны друг с другом, что дает основание рассматривать их по отдельности. Алгоритмы перевода $Z_w \rightarrow Z_q$ вытекают из следующих соображений. Многочлен (3.1) для Z_q может быть представлен в виде:

$$Z_q = \sum_{j=0}^{m-1} b_j \cdot q^j = ((\dots(b_{m-1} \cdot q + b_{m-2}) \cdot q + b_{m-3}) \cdot q + \dots + b_1) \cdot q + b_0, \quad (3.2)$$

где m — число разрядов в записи Z_p , а b_j ($j=0, \dots, m-1$) — цифры числа Z_q .

Разделим число Z_q на две части по разряду номер i . Число, включающее $m-i$ разрядов с $(m-i)$ -го по i -й, обозначим γ_i , а число с i разрядами с $(i-1)$ -го по 0-й — δ_i . Очевидно, $i \in [0, m-1]$, $\gamma_0 = \delta_{m-1} = Z_q$.

$$Z_q = (\underbrace{b_{m-1}b_{m-2} \dots b_i}_{\gamma_i} \underbrace{b_{i-1} \dots b_1 b_0}_{\delta_i}).$$

Позаимствуем из языка PASCAL обозначение двух операций: div — результат целочисленного деления двух целых чисел и mod — остаток от целочисленного деления ($13 \text{ div } 4 = 3$; $13 \text{ mod } 4 = 1$).

Теперь если принять $\gamma_{m-1} = b_{m-1}$, то в (3.2) усматривается следующее рекуррентное соотношение: $\gamma_i = \gamma_{i+1} + b_i$, из которого, в свою очередь, получаются выражения:

$$\gamma_{i+1} = \gamma_i \operatorname{div} q; \quad b_i = \gamma_i \operatorname{mod} q. \quad (3.3)$$

Аналогично, если принять $\delta_0 = b_0$, то для правой части числа будет справедливо другое рекуррентное соотношение: $\delta_i = \delta_{i-1} + b_i q^i$, из которого следуют:

$$b_i = \delta_i \operatorname{div} q^i; \quad \delta_{i-1} = \delta_i \operatorname{mod} q^i. \quad (3.4)$$

Из соотношений (3.3) и (3.4) непосредственно вытекают два способа перевода целых чисел из десятичной системы счисления в систему с произвольным основанием q .

Способ 1 является следствием соотношений (3.3), предполагающий следующий алгоритм перевода:

1. Целочисленно разделить исходное число (Z_{10}) на основании новой системы счисления (q) и найти остаток от деления — это будет цифра 0-го разряда числа Z_q .
2. Частное от деления снова целочисленно разделить на q с выделением остатка; процедуру продолжать до тех пор, пока частное от деления не окажется меньше q .
3. Образовавшиеся остатки от деления, поставленные в порядке, обратном порядку их получения, и представляют Z_q .

Пример 3.1

Выполнить преобразование $123_{10} \rightarrow Z_5$. Результат — на рис. 3.1.

$$\begin{array}{r} 123 \quad | \quad 5 \\ \hline 120 \quad 24 \quad | \quad 5 \\ \hline 3 \quad 20 \quad 4 \\ \hline \quad \quad 4 \end{array}$$

Рис. 3.1. Результат выполнения примера 3.1

Остатки от деления (3, 4) и результат последнего целочисленного деления (4) образуют обратный порядок цифр нового числа. Следовательно, $123_{10} = 443_5$.

Необходимо заметить, что полученное число нельзя читать как "четыре-ста сорок три", поскольку десятки, сотни, тысячи и прочие подобные обозначения чисел относятся только к десятичной системе счисления. Прочитывать число следует простым перечислением его цифр с указанием системы счисления ("число четыре, четыре, три в пятеричной системе счисления").

Способ 2 вытекает из соотношения (3.4), действия производятся в соответствии со следующим алгоритмом:

1. Определить $m-1$ — максимальный показатель степени в представлении числа по форме (3.1) для основания q .
2. Целочисленно разделить исходное число (Z_{10}) на основание новой системы счисления в степени $m-1$ (т. е. q^{m-1}) и найти остаток от деления; результат деления определит первую цифру числа Z_q .
3. Остаток от деления целочисленно разделить на q^{m-2} , результат деления принять за вторую цифру нового числа; найти остаток; продолжать эту последовательность действий, пока показатель степени q не достигнет значения 0.

Продемонстрируем действие алгоритма на той же задаче, что была рассмотрена выше.

Определить $m-1$ можно либо путем подбора ($5^0 = 1 < 123$; $5^1 = 5 < 123$; $5^2 = 25 < 123$; $5^3 = 125 > 123$, следовательно, $m-1 = 2$), либо логарифмированием с оставлением целой части логарифма ($\log_5 123 = 2,99$, т. е. $m-1 = 2$).

Далее:

$$b_2 = 123 \operatorname{div} 5^2 = 4 \qquad \delta_1 = 23 \operatorname{mod} 5^2 = 23 \qquad i = 2 - 1 = 1$$

$$b_1 = 23 \operatorname{div} 5^1 = 4 \qquad \delta_0 = 23 \operatorname{mod} 5^1 = 3 \qquad i = 0$$

Алгоритмы перевода $Z_g \rightarrow Z_w$ явно вытекают из представлений (3.1) или (3.2): необходимо Z_p представить в форме многочлена и выполнить все операции по правилам десятичной арифметики.

Пример 3.2

Выполнить преобразование $443_5 \rightarrow Z_{10}$.

Решение:

$$443_5 = 4 \cdot 5^2 + 4 \cdot 5^1 + 3 \cdot 5^0 = 4 \cdot 25 + 4 \cdot 5 + 3 \cdot 1 = 123_{10}.$$

Необходимо еще раз подчеркнуть, что приведенными алгоритмами удобно пользоваться при переводе числа из десятичной системы в какую-то иную или наоборот. Они работают и для перевода между любыми иными системами счисления, однако преобразование будет затруднено тем, что все арифметические операции необходимо осуществлять *по правилам исходной* (в первых алгоритмах) или *конечной* (в последнем алгоритме) системы счисления.

По этой причине переход, например, $Z_3 \rightarrow Z_8$ проще осуществить через промежуточное преобразование к десятичной системе $Z_3 \rightarrow Z_{10} \rightarrow Z_8$. Ситуация, однако, значительно упрощается, если основания исходной и конечной систем счисления оказываются связанными соотношением $p = q^r$, где r — целое число (естественно, большее 1) или $r = 1/n$ ($n > 1$, целое) — эти случаи будут рассмотрены далее.

3.2.2. Перевод дробных чисел из одной системы счисления в другую

Вещественное число, в общем случае содержащее целую и дробную часть, всегда можно представить в виде суммы целого числа и правильной дроби. Поскольку в предыдущем разделе проблема записи натуральных чисел в различных системах счисления уже была решена, можно ограничить рассмотрение только алгоритмами перевода правильных дробей.

Введем следующие обозначения: правильную дробь в исходной системе счисления p будем записывать в виде $0, Y_p$, дробь в системе q — $0, Y_q$, а преобразование — в виде $0, Y_p \rightarrow 0, Y_q$.

Последовательность рассуждений весьма напоминает проведенную ранее для натуральных чисел. В частности, это касается рекомендации осуществлять преобразование через промежуточный переход к десятичной системе, чтобы избежать необходимости производить вычисления в "непривычных" системах счисления, т. е. $0, Y_p \rightarrow 0, Y_{10} \rightarrow 0, Y_q$.

Это, в свою очередь, разбивает задачу на две составляющие: преобразование $0, Y_p \rightarrow 0, Y_{10}$ и $0, Y_{10} \rightarrow 0, Y_q$, каждое из которых может рассматриваться независимо.

Алгоритмы перевода $0, Y_{10} \rightarrow 0, Y_q$ выводятся путем следующих рассуждений. Если основание системы счисления q , простая дробь содержит n цифр, и b_k — цифры дроби ($1 < b_k < n$, $0 < b_k < n-1$), то она может быть представлена в виде суммы:

$$0, Y_q = \sum_{k=1}^n b_k \cdot q^{-k} = \frac{1}{q} \left(b_1 + \frac{1}{q} (b_2 + \dots + \frac{1}{q} (b_{n-1} + \frac{1}{q} b_n) \dots) \right), \quad (3.5)$$

$$0, Y_q = (0, b_1 b_2 \dots \underbrace{b_i b_{i+1} \dots b_n}_{\varepsilon_i}).$$

Часть дроби от разряда i до ее конца обозначим ε_i и примем $\varepsilon_n = b_n/q$. Очевидно, $\varepsilon_1 = 0, Y_q$, тогда в (3.5) легко усматривается рекуррентное соотношение:

$$\varepsilon_i = \frac{1}{q} (b_i + \varepsilon_{i+1}). \quad (3.6)$$

Если вновь позаимствовать в PASCAL обозначение функции — на этот раз `trunc`, производящей округление целого вещественного числа путем отбрасывания его дробной части, то следствием (3.6) будут соотношения, позволяющие находить цифры новой дроби:

$$b_i = \text{trunc}(q \cdot \varepsilon_i); \quad \varepsilon_{i+1} = q \cdot \varepsilon_i - \text{trunc}(q \cdot \varepsilon_i). \quad (3.7)$$

Соотношения (3.7) задают алгоритм преобразования: $0, Y_{10} \rightarrow 0, Y_q$:

1. Умножить исходную дробь в десятичной системе счисления на q , выделить целую часть — она будет первой (старшей) цифрой новой дроби; отбросить целую часть.
2. Для оставшейся дробной части операцию умножения с выделением целой и дробных частей повторять, пока в дробной части не окажется 0 или не будет достигнута желаемая точность конечного числа; появляющиеся при этом целые будут цифрами новой дроби.
3. Записать дробь в виде последовательности цифр после нуля с разделителем в порядке их появления в пп. 1 и 2.

Пример 3.3

Выполнить преобразование $0,375_{10} \rightarrow 0, Y_2$. Результат — на рис. 3.2.

Таким образом, $0,375_{10} \rightarrow 0,011_2$.

$$\begin{array}{r|l} 0,375 \times 2 = & \mathbf{0}, & 750 \\ 0,75 \times 2 = & \mathbf{1}, & 50 \\ 0,5 \times 2 = & \mathbf{1}, & 0 \end{array}$$

Рис. 3.2. Результат выполнения примера 3.3

Перевод $0, Y_p \rightarrow 0, Y_{10}$, как и в случае натуральных чисел, сводится к вычислению значения формы (3.5) в десятичной системе счисления. Например:

$$0,011_2 = 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 0 + 0,25 + 0,125 = 0,375_{10}.$$

Следует сознавать, что после перевода дроби, которая была конечной в исходной системе счисления, она может оказаться бесконечной в новой системе. Соответственно, рациональное число в исходной системе может после перехода превратиться в иррациональное. Справедливо и обратное утверждение: число иррациональное в исходной системе счисления в иной системе может оказаться рациональным.

Пример 3.4

Выполнить преобразование $5,3(3)_{10} \rightarrow Y_8$.

Перевод целой части, очевидно, дает: $5_{10} = 12_3$. Перевод дробной части: $0,3(3)_{10} \rightarrow 0,1$. Окончательно: $5,3(3)_{10} \rightarrow 12,1_3$.

Как уже было сказано, значение целого числа не зависит от формы его представления и выражает количество входящих в него единиц. Простая дробь имеет смысл доли единицы, и это "дольное" содержание также не зависит от выбора способа представления. Другими словами, треть пирога остается третью в любой системе счисления.

3.2.3. Перевод чисел между системами счисления $2 \leftrightarrow 8 \leftrightarrow 16$

Интерес к двоичной системе счисления вызван тем, что именно она используется для представления чисел в компьютере. Однако двоичная запись оказывается громоздкой, поскольку содержит много цифр и, кроме того, плохо воспринимается и запоминается человеком из-за зрительной однородности (все число состоит из нулей и единиц). Поэтому в нумерации ячеек памяти компьютера, записи кодов команд, нумерации регистров и устройств и пр. используются системы счисления с основаниями 8 и 16. Выбор именно этих

систем счисления обусловлен тем, что переход от них к двоичной системе и обратно осуществляется, как будет показано далее, весьма простым образом.

Двоичная система счисления имеет основанием 2 и, соответственно, две цифры: 0 и 1.

Восьмеричная система счисления имеет основание 8 и цифры 0, 1, ..., 7.

Шестнадцатеричная система счисления имеет основание 16 и цифры 0, 1, ..., 9, А, В, С, D, E, F. При этом знак А является шестнадцатеричной цифрой, соответствующей числу 10 в десятичной системе, $B_{16} = 11_{10}$, $C_{16} = 12_{10}$, $D_{16} = 13_{10}$, $E_{16} = 14_{10}$ и $F_{16} = 15_{10}$. Другими словами, в данном случае А, ..., F — это не буквы латинского алфавита, а *цифры шестнадцатеричной системы счисления*.

Докажем две теоремы [12].

Теорема 1. Для преобразования целого числа $Z_p \rightarrow Z_q$ в том случае, если системы счисления связаны соотношением $q = p^r$, где r — целое число, большее 1, достаточно Z_p разбить справа налево на группы по r цифр и каждую из них независимо перевести в систему q .

Доказательство. Пусть максимальный показатель степени в записи числа p по форме (3.1) равен $k-1$, причем, $2r > k-1 > r$.

$$Z_p = (a_{k-1} \dots a_1 a_0) = a_{k-1} \cdot p^{k-1} + a_{k-2} \cdot p^{k-2} + \dots a_1 \cdot p^1 + a_0 \cdot p^0.$$

Вынесем множитель p^r из всех слагаемых, у которых $j \geq r$. Получим:

$$Z_p = (a_{k-1} \cdot p^{k-1-r} + a_{k-2} \cdot p^{k-2-r} + \dots a_{r+1} \cdot p^1 + a_r \cdot p^0) \cdot p^r + (a_{r-1} \cdot p^{r-1} + a_{r-1} \cdot p^{r-2} + \dots a_1 \cdot p^1 + a_0 \cdot p^0) \cdot p^0 = b_1 \cdot q^1 + b_0 \cdot q^0,$$

где

$$b_1 = a_{k-1} \cdot p^{k-1-r} + a_{k-2} \cdot p^{k-2-r} + \dots a_{r+1} \cdot p^1 + a_r \cdot p^0 = (a_{k-1} \dots a_r)_p,$$

$$b_0 = a_{r-1} \cdot p^{r-1} + a_{r-1} \cdot p^{r-2} + \dots a_1 \cdot p^1 + a_0 \cdot p^0 = (a_{r-1} \dots a_0)_p.$$

Таким образом, r -разрядные числа системы с основанием p оказываются записанными как цифры системы с основанием q . Этот результат можно обобщить на ситуацию произвольного $k-1 > r$ — в этом случае выделятся не две, а больше (m) цифр числа с основанием q . Очевидно, $Z_q = (b_m \dots b_0)_q$.