Архитектура современных вычислительных устройств / Архитектура ЭВМ и системное ПО Лекции 1–2. Основные концепции

Макаров П. А.

сентябрь 2024 г.

Содержание

1 Литература

1.1 Основная литература

1. *Максимов Н. В., Попов И. И., Партыка Т. П,* Архитектура ЭВМ и вычислительные системы.

2. *Столяров А. В.* Программирование: введение в профессию. Т. 1, 2.

3. *Брайант Р. Э., О'Халларон Д. Р.* Компьютерные системы: архитектура и программирование.

4. *Марек Р.* Ассемблер на примерах. Базовый курс.

1.2 Дополнительная литература

1. *Петцольд Ч.* Код: тайный язык информатики.

2. *Абель П.* Язык Ассемблера для IBM PC и программирования.

3. *Джордейн Р.* Справочник программиста персональных компьютеров типа IBM PC, XT и AT.

4. *Юров В.И.* Assembler. Учебник для вузов.

5. *Калашников О.* Ассемблер — это просто. Учимся программировать.

6. *Керниган Б., Ритчи Д.* Язык программирования С.

7. *Богатырёв А.* Хрестоматия по программированию на Си в Unix.

8. Разработка на платформе Эльбрус. https://dev.mcst.ru/

2 Основные понятия

Определение 1. *Под* ***компьютерами*** *будем подразумевать любые электронные устройства, способные исполнять программы, независимо от их конкретной архитектуры. Часто в качестве синонима используются понятия* ***вычислительная машина/система, ЭВМ****.*

Определение 2. ***Архитектура компьютера*** *— это концептуальная модель компьютерной системы, определяемая её компонентами и их взаимодействиями между собой и окружением. Кроме того, в понятие архитекуры включаются принципы её проектирования и развития.*

Основным компонентом любого компьютера является процессор (в некоторых случаях — контроллер).

Определение 3. ***Процессор*** *— специализированное электронное устройство, предназначенное для выполнения тех или иных операций над некоторой информацией по заданной программе.*

Определение 4. ***Контроллер*** *— это практически то же самое, что и процессор, с той разницей, что программа его работы, как правило, никогда не меняется в процессе функционирования системы.*

В настоящее время различие между двумя этими типами устройств минимальное. Так, существуют и очень сложные универсальные контроллеры, способные выполнять под управлением специальных операционных систем множество различных программ, и содержащие в своём составе ряд тех или иных процессоров.

Процессоры и контроллеры изготовляются по интегральной технологии в миниатюрном исполнении — в виде микросхем (часто используют кальку с английского *chip*). Вместе с тем, в виде микросхем сейчас выпускаются и более сложные по организации устойства, которые называют микрокомпьютерами (μЭВМ). Последние микросхемы помимо процессора включают в себя ОЗУ (RAM), ПЗУ (ROM), системы обработки прерываний, таймеры и другие устройства.

В современных компьютерах применяются различные типы процессоров. Рассмотрим два основных варианта их классификации.

- По используемым видам электрических сигналов:
  - аналоговые;
  - цифровые:

    * ТТЛ (TTL) — «0» $\leftrightarrow U < 0.4$ В, «1» $\leftrightarrow U > 2.0$ В, $U_{\text{пит}} = +5.0$ В;
    * КМОП (CMOS) — «0» $\leftrightarrow U < 7$ В, «1» $\leftrightarrow U > 8$ В.

- По функциональному назначению:
  - CPU;
  - FPU (NPU);
  - GPU.

Определение 5. ***Программа*** *(в широком смысле) — это текст, представляющий собой исчерпывающее описание действий исполнителя и написанный на понятном ему языке.*

В рамках нашего курса мы будем иметь в виду исключительно *компьютерные программы,* то есть тексты исполняемые именно *компьютерами.* Очень часто программы пишут не на языке исполнительного устройства (так казываемом машинном коде), а на других языках (по тем или иным причинам более удобным для программиста), которые при этом называют *языками программирования.*

Определение 6. ***Язык программирования*** *— это формальный язык, предназначенный для записи компьютерных программ. Язык программирования (как и любой естественный язык) представляет собой совокупность лексических, синтаксических и семантических правил.*

В настоящее время существует тысячи языков программирования и множество вариантов их классификации. Приведём несколько основных вариантов классификаций.

- По уровню применяемых абстракций:
  - низкого;
  - среднего;
  - высокого уровня.
- По режиму исполнения программ:
  - транслируемые;
  - интерпретируемые.
- По используемой типизации:
  - со статической;

– либо динамической типизацией.

- По поддержке различных компьютерных архитектур:
  - непереносимые;
  - переносимые;
  - кроссплатформенные.
- По назначению:
  - унивесальные (общего назначения);
  - специализированные.
- По возможности следования парадигме программирования:
  - императивные;
  - процедурные;
  - функциональные;
  - логические;
  - объектно-ориентированные.

Действительно ли являются компьютерными программами тексты, приведённые в листингах 1 и 2?

```cpp
#include <iostream>

using namespace std;

int main() {
    cout << "Hello, world!" << endl;
    return 0;
}
```

Листинг 1: Пример текста, написанного на языке C++

```python
#!/usr/bin/python3.8

print("Hello, world!")
```

Листинг 2: Пример текста, написанного на языке Python

Определение 7. ***Исходный код*** — *это текст компьютерной программы на каком-либо языке программирования или языке разметки,*

который может быть прочтён человеком. Исходный код транслируется в исполняемый код целиком до запуска программы при помощи транслятора либо исполняется непосредственно при помощи интерпретатора. В широком смысле, исходный код — это любые входные данные для транслятора.

Определение 8. ***Транслятор*** *— это программа, выполняющая* ***трансляцию****, то есть преобразование программы, представленной на одном из языков программирования, в программу на другом языке до её исполнения. Трансляция бывает горизонтальной и вертикальной.*

Пример 1. *Процесс трансляции программ, написанных на языке C и C++ включает следующие этапы.*

1. *Обработка исходного текста препроцессором.*
2. *Компиляция.*
3. *Ассемблирование.*
4. *Компоновка.*

А является ли компьютерной программой текст, приведённый в листинге 3?

```
1 ; nasm -f elf64 null.nasm && ld null.o -o null
2
3 %define SYSCALL_EXIT 60
4
5 global _start:
6 _start:
7   mov rax, SYSCALL_EXIT
8   mov rdi, 0
9   syscall
```

Листинг 3: Пример текста, написанного на языке ассемблера NASM

Определение 9. *Таким образом, термин* ***ассемблер*** *применяется в информатике в двух значениях.*

1. *С одной стороны это слово* ***служит общим названием семейства низкоуровневых языков программирования.*** *В этом контексте различают множество различных* ***диалектов ассемблера*** *(в частности, вариации ассемблера x86, ARM, PIC и т. п.), а также два основных* ***синтаксиса:*** *Intel и AT&T.*

2. *С другой стороны, под этим подразумевают именно* ***транслятор, который преобразует исходный код программы,*** *написанный на конкретном языке ассемблера в машинный код некоторого формата (более точно — объектный код) для данной архитектуры компьютера.*

Определение 10. ***Интерпретатор*** *— программа, выполняющая* ***интерпретацию*** *— построчный анализ, обработку и выполнение исходного кода программы непосредственно во время её исполнения.*

Существует две основные разновидности интерпретаторов.

1. Простые.
2. Компилирующего типа.

Алгоритм работы простого интерпретатора:

1. прочитать инструкцию;
2. проанализировать инструкцию и определить соответствующие действия;
3. выполнить соответствующие действия;
4. если не достигнуто условие завершения программы, прочитать следующую инструкцию и перейти к пункту 2.

Некоторые интерпретаторы (например, для языка Python и многих других) могут работать в режиме диалога или так называемого *цикла чтения-вычисления-печати* (REPL — read-eval-print loop). В таком режиме интерпретатор считывает законченную конструкцию языка, выполняет её, печатает результаты, после чего переходит к ожиданию ввода пользователем следующей конструкции.

Определение 11. ***Отладчиком*** *называется инструмент, необходимый для интерактивного анализа исходного кода программы с целью поиска и устранения ошибок или исследования принципа её работы.*

Замечание 1. *Работа с некоторыми отладчиками (в том числе и с отладчиками самого низкого уровня) очень похожа на работу с интерпретатором в режиме диалога.*

3 Инструменты разработчика

- Текстовый редактор.
- Набор транслятора либо интерпретатор.
- Библиотека необходимых функций и компонентов.
- Система сборки.
- Отладчик.
- Профилировщик кода.
- Система контроля версий.
- Генератор документации.

Определение 12. ***Интегрированная среда разработки*** *— это комплекс программных средств, используемый программистами для разработки программного обеспечения (IDE — Integrated Development Environment).*

4 Основы операционных систем

Определение 13. ***Операционная система*** *— это комплекс взаимосвязанных программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем.*

Основные этапы развития компьютеров и операционных систем.

- Пакетный режим. Необходимость оптимального использования дорогостоящих вычислительных ресурсов привела к появлению концепции пакетного режима исполнения программ. Пакетный режим предполагает наличие очереди программ на исполнение, причём система может обеспечивать загрузку программы с внешних носителей данных в оперативную память, не дожидаясь завершения исполнения предыдущей программы, что позволяет избежать простоя процессора.
- Разделение времени и многозадачность. Уже пакетный режим в своём развитом варианте требует разделения процессорного времени между выполнением нескольких программ. Необходимость в разделении времени (многозадачности, мультипрограммировании)

проявилась ещё сильнее при распространении в качестве устройств ввода-вывода телетайпов (а позднее, терминалов с электронно-лучевыми дисплеями) (1960-е годы). Поскольку скорость клавиатурного ввода (и даже чтения с экрана) данных оператором намного ниже, чем скорость обработки этих данных компьютером, использование компьютера в "монопольном" режиме (с одним оператором) могло привести к простою дорогостоящих вычислительных ресурсов. Разделение времени позволило создать "многопользовательские" системы, в которых один (как правило) центральный процессор и блок оперативной памяти соединялся с многочисленными терминалами. При этом часть задач (таких как ввод или редактирование данных оператором) могла исполняться в режиме диалога, а другие задачи (такие как массивные вычисления) — в пакетном режиме.

- Разделение полномочий. Распространение многопользовательских систем потребовало решения задачи разделения полномочий, позволяющей избежать возможности изменения исполняемой программы или данных одной программы в памяти компьютера другой программой (намеренно или по ошибке), а также изменения самой системы прикладной программой. Реализация разделения полномочий в операционных системах была поддержана разработчиками процессоров, предложивших архитектуры с двумя режимами работы процессора — "реальным" (в котором исполняемой программе доступно всё адресное пространство компьютера) и "защищённым" (в котором доступность адресного пространства ограничена диапазоном, выделенным при запуске программы на исполнение).

- Системы реального времени.

- Гибридные системы.

Есть приложения вычислительной техники, для которых операционные системы излишни. Например, встроенные микрокомпьютеры, содержащиеся во многих бытовых приборах, автомобилях, простейших сотовых телефонах и т. д. Операционные системы используются:

- если нужен универсальный механизм хранения данных;

- для предоставления программам системных библиотек с часто используемыми подпрограммами;

- для распределения полномочий;

- необходима возможность имитации “одновременного” исполнения нескольких программ на одном компьютере;
- для управления процессами выполнения отдельных программ.

Таким образом, современные универсальные операционные системы можно охарактеризовать, прежде всего, как:

- использующие файловые системы (с универсальным механизмом доступа к данным);
- многопользовательские (с разделением полномочий);
- многозадачные (с разделением времени).

Определение 14. ***Файловая система*** *— это способ хранения данных на внешних запоминающих устройствах.*

Функции операционных систем.

- Основные функции:
  - загрузка программ в оперативную память и их выполнение.
  - исполнение запросов программ (ввод и вывод данных, запуск и остановка других программ, выделение и освобождение дополнительной памяти и др.).
  - стандартизованный доступ к периферийным устройствам.
  - управление оперативной памятью (распределение между процессами, организация виртуальной памяти).
  - управление доступом к данным на энергонезависимых носителях (жёсткие, оптические диски и др.), организованным в той или иной файловой системе.
  - обеспечение пользовательского интерфейса.
  - сохранение информации об ошибках системы.
- Дополнительные функции:
  - параллельное или псевдопараллельное выполнение задач (многозадачность).
  - эффективное распределение ресурсов вычислительной системы между процессами.
  - разграничение доступа различных процессов к ресурсам.

– организация надёжных вычислений (невозможности одного вычислительного процесса повлиять на вычисления в другом процессе), основана на разграничении доступа к ресурсам.

– взаимодействие между процессами: обмен данными, взаимная синхронизация.

– защита самой системы, а также пользовательских данных и программ от действий пользователей или приложений.

– многопользовательский режим работы и разграничение прав доступа.

Многозадачность и распределение полномочий требуют определённой иерархии привилегий компонентов в самой операционной системе. В составе операционной системы различают три группы компонентов.

- Ядро и планировщик. Драйверы устройств, непосредственно управляющие оборудованием. Сетевая и файловая системы.
- Системные библиотеки.
- Оболочка с утилитами.

Большинство программ, как системных (входящих в операционную систему), так и прикладных, исполняются в непривилегированном ("пользовательском") режиме работы процессора и получают доступ к оборудованию (и, при необходимости, к другим ресурсам ядра, а также ресурсам иных программ) только посредством системных вызовов. Ядро исполняется в привилегированном режиме: именно в этом смысле говорят, что система (точнее, её ядро) управляет оборудованием.

Определение 15. ***Ядро*** — *центральная часть операционной системы, управляющая ресурсами компьютера, выполнением процессов и обеспечивающая координированный доступ процессов к ресурсам.*

Основными ресурсами являются *процессорное время, память* и *устройства ввода-вывода.* Доступ к файловой системе и сетевое взаимодействие также могут быть реализованы на уровне ядра. Перечислим основные объекты ядра операционной системы.

- Процессы.
- Файлы.
- События.

- Потоки.
- Семафоры.
- Мьютексы.
- Каналы.

Как основополагающий элемент операционной системы, ядро представляет собой наиболее низкий уровень абстракции для доступа приложений к ресурсам вычислительной системы, необходимым для их работы. Как правило, ядро предоставляет такой доступ исполняемым процессам соответствующих приложений за счёт использования механизмов межпроцессного взаимодействия и обращения приложений к системным вызовам операционной системы.

Определение 16. ***Системный вызов*** *— это элемент пользовательского интерфейса ядра, представляющий собой обращение прикладной программы к ядру операционной системы для выполнения той или иной операции.*

С целью обеспечения совместимости системные вызовы стандартизируются, а доступ к ним в пользовательском режиме предоставляется с помощью интерфейса в виде стандартной библиотеки. При этом с точки зрения прикладного программиста, системные вызовы реализуются в виде функций, прототипы которых следуют соглашениям API.

Определение 17. ***Программный интерфейс приложения (API — Application Programming Interface)*** *— это описание способов взаимодействия одной компьютерной программы с другими. Обычно входит в описание интернет-протоколов, программных фреймворков или стандарта вызовов функций операционной системы.*

Системный программист, работающий с системными вызовами на уровне ядра, кроме API часто сталкивается с понятием ABI.

Определение 18. ***Двоичный интерфейс приложений (ABI — Application Binary Interface)*** *— это набор соглашений для доступа приложения к операционной системе и другим низкоуровневым сервисам, спроектированный для переносимости исполняемого кода между машинами, имеющими совместимые ABI.*

В отличие от API, который регламентирует совместимость на уровне исходного кода, ABI можно рассматривать как набор правил, позволяющих компоновщику объединять откомпилированные модули компонента без перекомпиляции всего кода. Двоичный интерфейс приложений регламентирует:

- конкретный набор инструкций процессора;
- использование регистров процессора;
- соглашение о вызове функций;
- состав и формат системных вызовов и вызовов одного модуля другим;
- форматы исполняемых файлов.

UNIX и UNIX-like ОС, стандартизация и POSIX

К концу 1960-х годов отраслью и научно-образовательным сообществом был создан целый ряд операционных систем, реализующих все или часть очерченных выше функций. К ним относятся Atlas (Манчестерский университет), CTTS и ITS (Массачусетский технологический институт, MIT), THE (Эйндховенский технологический университет), RS4000 (Университет Орхуса) и другие. Всего эксплуатировалось более сотни различных ОС.

Наиболее развитые операционные системы, такие как OS/360 (IBM), SCOPE (CDC) и завершённый уже в 1970-х годах Multics (MIT и Bell Labs), предусматривали возможность исполнения на многопроцессорных компьютерах.

Эклектичный характер разработки операционных систем привёл к нарастанию кризисных явлений, прежде всего, связанных с чрезмерными сложностью и размерами создаваемых систем. Системы были плохо масштабируемыми (более простые не могли использовать все возможности крупных вычислительных систем; более развитые неоптимально исполнялись на малых или не могли исполняться на них вовсе) и полностью несовместимыми между собой, их разработка и совершенствование затягивались.

Задуманная и реализованная в 1969 году Кеном Томпсоном при участии нескольких коллег (включая Денниса Ритчи и Брайана Кернигана), операционная система UNIX (первоначально UNICS, что обыгрывало название Multics) вобрала в себя многие черты более ранних систем, но обладала целым рядом свойств, отличающих её от большинства предшественниц:

- простая метафорика (два ключевых понятия: вычислительный процесс и файл);

- компонентная архитектура: принцип “одна программа — одна функция” плюс мощные средства связывания различных программ для решения возникающих задач (“оболочка”);

- минимизация ядра (кода, выполняющегося в “реальном” (привилегированном) режиме процессора) и количества системных вызовов;

- независимость от аппаратной архитектуры и реализация на машинонезависимом языке программирования (язык программирования Си стал побочным продуктом разработки UNIX);

- унификация файлов.

UNIX, благодаря своему удобству прежде всего в качестве инструментальной среды (среды разработки), обрела популярность сначала в университетах, а затем и в отрасли, получившей прототип единой операционной системы, которая могла использоваться на самых разных вычислительных системах и, более того, могла быть быстро и с минимальными усилиями перенесена на любую вновь разработанную аппаратную архитектуру.

В конце 1970-х годов сотрудники Калифорнийского университета в Беркли внесли ряд усовершенствований в исходные коды UNIX, включая работу с протоколами TCP/IP. Их разработка стала известна под именем BSD (Berkeley Software Distribution).

Задачу разработать независимую (от авторских прав Bell Labs) реализацию той же архитектуры поставил и Ричард Столлман, основатель проекта GNU.

Благодаря конкурентности реализаций архитектура UNIX стала вначале фактическим отраслевым стандартом, а затем обрела статус и стандарта юридического — ISO/IEC 9945 (POSIX).

Только системы, отвечающие спецификации Single UNIX Specification, имеют право носить имя UNIX. К таким системам относятся AIX, HP-UX, IRIX, Mac OS X, SCO OpenServer, Solaris, Tru64 и z/OS.

Операционные системы, следующие стандарту POSIX или опирающиеся на него, называют “POSIX-совместимыми” (чаще встречается словоупотребление “UNIX-подобные” или “семейство UNIX”, но оно противоречит статусу торгового знака “UNIX”, принадлежащего консорциуму The Open Group и зарезервированному для обозначения только операционных систем, строго следующих стандарту). Сертификация на совместимость со стандартом платная, из-за чего некоторые системы не проходили этот процесс, однако считаются POSIX-совместимыми по существу.

К UNIX-подобным относятся операционные системы, основанные на последней версии UNIX, выпущенной Bell Labs (System V), на разработ-

ках университета Беркли (FreeBSD, OpenBSD, NetBSD), на основе Solaris (OpenSolaris, BeleniX, Nexenta OS), а также Linux, разработанная в части утилит и библиотек проектом GNU и в части ядра — сообществом, возглавляемым Линусом Торвальдсом.

Стандартизация операционных систем преследует цель упрощения замены самой системы или оборудования при развитии вычислительной системы или сети и упрощении переноса прикладного программного обеспечения (строгое следование стандарту предполагает полную совместимость программ на уровне исходного текста; из-за профилирования стандарта и его развития некоторые изменения бывают всё же необходимы, но перенос программы между POSIX-совместимыми системами обходится на порядки дешевле, чем между альтернативными), а также преемственность опыта пользователей.

Самым заметным эффектом существования этого стандарта стало эффективное разворачивание Интернета в 1990-х годах.

5 Понятие процесса и его состояния

Под процессом понимается некоторая последовательность действий, составляющая задачу. Он определяется соответствующей программой, т. е. упорядоченным набором машинных команд, реализующих действия, которые должны предприниматься процессом; содержимым рабочей области памяти (набором данных, которые процесс может считывать, записывать, использовать); дескриптором процесса, который описывает текущее состояние любого выделенного ему ресурса. Часто под процессом (Process) или заданием (Task) понимается программа, находящаяся в решении.

Для выполнения работы ОС выделяет процессу процессорное время и другие ресурсы: память, прерывания, каналы прямого доступа и др. В однозадачной ОС присутсвует лишь один пользовательский процесс. В многозадачной системе на ресурсы могут претендовать множество независимых процессов. Планирование процессов — это управление распределением ресурсов между различными конкурирующими процессами путём передачи им управления согласно некоторой стратегии планирования.

Некоторые ОС позволяют одному заданию создавать несколько различных процессов, выполняемых параллельно (одновременно), а также разрешают одной программе быть разделённой между несколькими независимыми процессами.

Процесс создаётся, когда начинается выполнение задания пользователем и уничтожается, когда оно выполнено. Во время существования процесс может находится в трёх состяниях. Процесс активен (выполня-

ется, Running), когда он использует процессор для выполнения своих команд. Процесс блокирован (Blocked) или находится в ожидании, если его выполнение может быть продолжено после наступления некоторого события, например, окончания операции ввода/вывода, обработки промаха по кеш и др. Невыполняющиеся и не блокированные процессы находятся в состоянии готовности (Ready). Они находятся в очереди процессов за ресурсом. Изменение состояния процесса показано на рис.

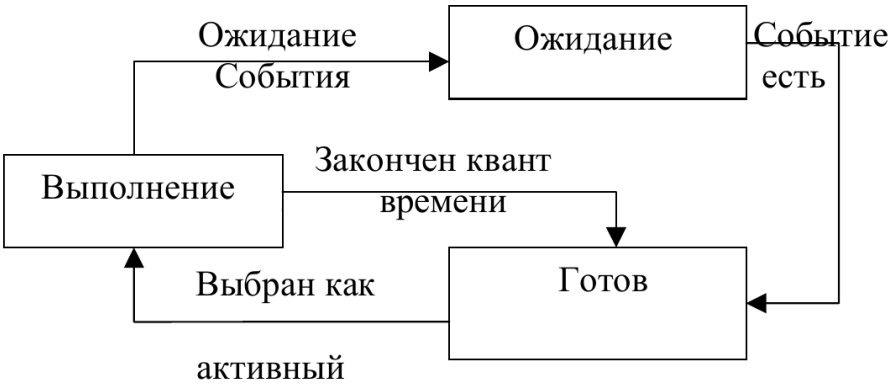

Рис. 1: Переход процесса в разные состояния

Выбор ОС процесса и передача ему управления называют дисптчеризацией. До завершения процесс может многократно переходить из состояния активности в другие состояния и наоборот. Чтобы это не влияло на результаты, его состояние записывается в так называемый дескриптор и востанавливается при возобновлении его активности. В дескриптор записывается точка повторного запуска, состояние регистров, стека и т. д. Выбор процессов для активизации производится циклически, либо по приоритету. Приоритеты могут быть фиксированными, либо меняться в том числе динамически в зависимости от призводительности и загрузки системы.

Чаще всего процесс представляется в виде некоторого графа, вершины которого представляют состояния процесса, а рёбра — переходы между ними. Рёбра помечают именем или номером события, которое соответствует данному переходу. Корневая вершина сответствует начальному состоянию процесса. Для представления процессов с большим числом состояний используются протоколы. Это конечная последовательность символов, фиксирующая события, в которых процесс принял участие до некоторого момента времени.

6 Информация и кодирование

6.1 Определение и классификации информации

Понятие «информация» является таким же фундаментальным, как понятия «материя», «энергия» и другие философские категории. Это атрибут, свойство сложных систем, связанное с их развитием и самоорганизацией. Известно большое количество различных определений информации, отличие информации от данных, знаний и пр. Ограничимся рассмотрением только некоторых самых практически важных понятий и определений в этом контексте.

На бытовом уровне информацией называют *любые данные или факты,* которые представляют какой-либо интерес. Например, сообщение о событиях, о чьей-либо деятельности и т. п.

В технике под информацией понимают *сообщения,* передаваемые в форме знаков или сигналов.

В компьютерных науках под информацией понимают ту часть *знаний,* которая используется для ориентирования, активного действия, управления, т. е. в целях сохранения, совершенствования, развития системы.

Приведем несколько определений информации:

- отрицание энтропии (Л. Бриллюэн);
- мера сложности структур (Моль);
- отражённое разнообразие (Урсул);
- содержание процесса отражения (Тузов);
- вероятность выбора (Яглом);
- снятая неопределенность наших знаний о чем-то (К. Шеннон);
- обозначение содержания, полученного из внешнего мира в процессе нашего приспособления к нему и приспособления к нему наших чувств (Н. Винер).

Информация может классифицироваться, например, по признакам, отражающим структуру данных и форму их представления. Приведём несколько примеров классов информации.

- По уровням сложности:
  - сигнал;
  - сообщение/документ;

  - информационный массив;
  - информационный ресурс.

- По типу применяемых сигналов:
  - аналоговая (непрерывная);
  - цифровая (дискретная);

- По уровням доступа и организации:
  - данные в регистровой памяти;
  - данные в оперативной памяти;
  - файлы данных на внешних устройствах;
  - базы данных.

- По способам кодирования и представления:
  - цифровая (вычислительные данные, двоичные);
  - символьная или текстовая (алфавитно-цифровая, строчная);
  - графическая.

Многолетнее развитие теории и практики ЭВМ привело к вытеснению (в том числе и на бытовом уровне) аналоговых устройств и сигналов цифровыми, поэтому почти любая информация, с которой имеет дело современный программист, представляет собой те или иные числа. В зависимости от контекста, эти числа могут выступать в разных ролях, например кодировать управляющие команды; являться адресами ячеек памяти и устройств; а также быть непосредственными данными. Таким образом, можно сформулировать следующее.

Определение 19. ***Информация*** *— это данные и контекст.*

Пример 2. *Рассмотрим одну и ту же последовательность символов в разных контекстах.*

```
$ hexdump -C data.dat

00000000  b8 3c 00 00 00 bf 00 00  00 00 0f 05 0a           |.<...........|
0000000d

$ objdump -d null

null:     file format elf64-x86-64
```

```
Disassembly of section .text:

0000000000401000 <_start>:
  401000: b8 3c 00 00 00       mov    $0x3c,%eax
  401005: bf 00 00 00 00       mov    $0x0,%edi
  40100a: 0f 05                syscall
```

6.2 Передача информации

Определение 20. ***Канал передачи*** *— это комплекс технических средств и среды распространения, обеспечивающий передачу сигнала электросвязи в определенной полосе частот или с определенной скоростью передачи между сетевыми станциями и узлами.*

При обмене данными по каналам используется три метода передачи данных:

- симплексная (однонаправленная) передача (телевидение, радио);
- полудуплексная (приём и передача информации осуществляются поочередно);
- дуплексная (двунаправленная), каждая станция одновременно передает и принимает данные.

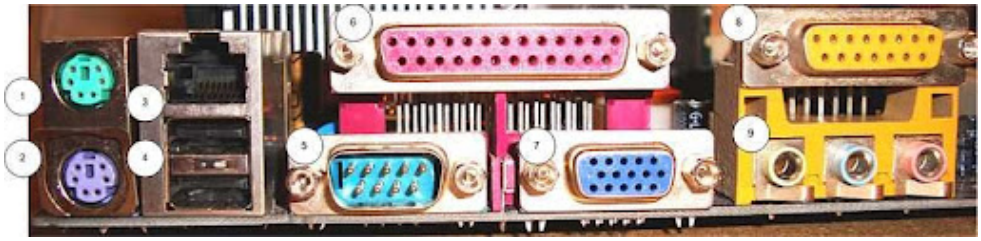

Рис. 2: Примеры некоторых портов ввода/вывода

Для передачи данных в информационных системах наиболее часто применяется последовательная передача. Широко используются асинхронная и синхронная методы последовательной передачи.

При асинхронной передаче каждый символ передается отдельной посылкой.

Стартовые биты предупреждают приёмник о начале передачи. Затем передается символ. Для определения достоверности передачи используется бит чётности (бит четности равен «1», если количество единиц в символе нечетно, и «0» — в противном случае). Последний бит («стоп-бит») сигнализирует об окончании передачи.

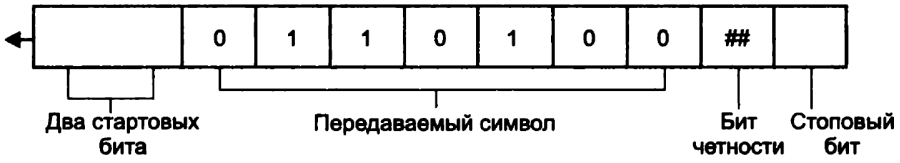

Рис. 3: Асинхронная передача данных

При использовании синхронного метода данные передаются блоками. Для согласования работы приёмника и передатчика в начале блока передаются биты синхронизации. Затем передаются данные, код обнаружения ошибки и символ окончания передачи. При синхронной передаче данные могут передаваться и как символы, и как поток битов. В качестве кода обнаружения ошибки обычно используется код CRC.

Рис. 4: Синхронная передача данных

Практически всегда дискретный сигнал имеет два либо три значения. Нередко его называют также цифровым сигналом. Дискретные сигналы по сравнению с аналоговыми имеют ряд важных преимуществ: помехоустойчивость, лёгкость восстановления формы, простоту аппаратуры передачи данных.

Обычно в цифровых системах используются двоичные сигналы, представляемые двумя значениями: «+» и «0».

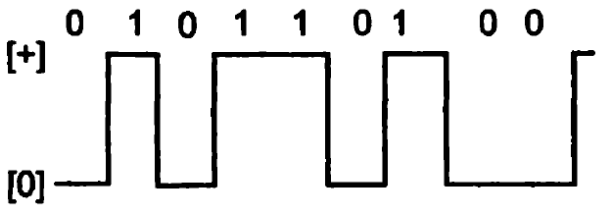

Рис. 5: Двоичные сигналы

Вместе с тем при передаче данных в большинстве случаев применяются троичные сигналы со значениями «+», «0», «−». Здесь «логическая единица» представляется отсутствием потенциала в канале, тогда как «логический нуль» характеризуется положительным либо отрицательным импульсом. При этом полярность импульсов, представляющих

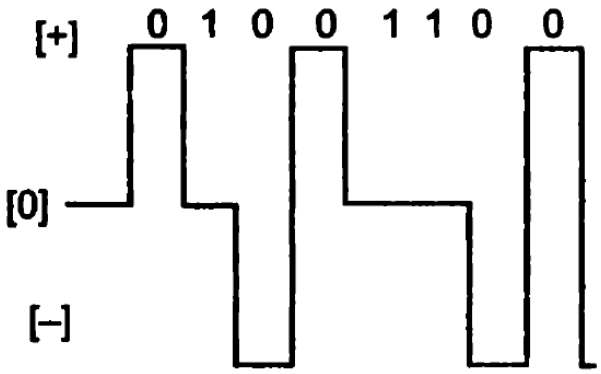

Рис. 6: Троичные сигналы

«нули», должна чередоваться, т. е. за положительным «+» импульсом должен следовать отрицательный «−» и наоборот.

В форме троичного сигнала осуществляется не только кодирование передаваемых данных, но также обеспечивается синхронизация работы канала и проверка целостности данных.

Также в каналах передачи данных часто используются биполярные сигналы. Здесь единицы представляются чередующимися положитель-

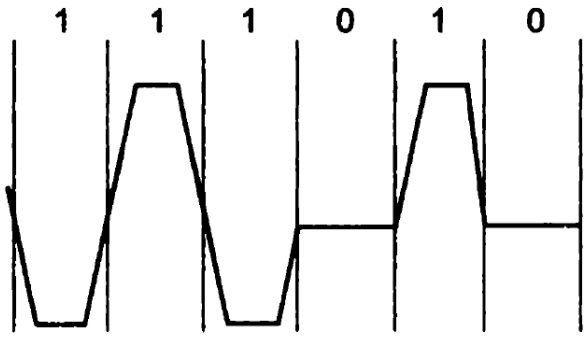

Рис. 7: Биполярные сигналы

ными и отрицательными импульсами. Отсутствие импульсов определяет состояние «логический нуль». Биполярное кодирование обеспечивает обнаружение одиночной ошибки. Так, если вместо нуля появится единица либо единица ошибочно сменится на нуль, то это легко обнаруживается. В обоих случаях нарушается чередование полярности импульсов.

Любой реальный импульсный сигнал (см. рис. 8) характеризуется амплитудой U_0, длительностью импульсов $t_и$, передним фронтом $t_{ф1}$, задним фронтом $t_{ф2}$ и периодом повторения T. Отношение длительности периода к длительности импульсов $t_и$ называется скважностью $Q = T/t_и$.

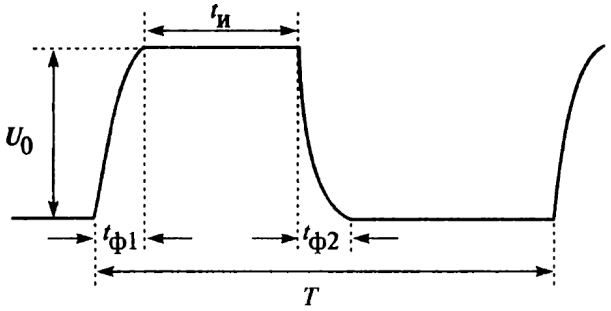

Рис. 8: Реальный вид импульсных сигналов

6.3 Измерение количества информации

Определение 21. ***Битом*** *(bit) информации называется один двоичный разряд.* ***Байт*** *(byte) — это группа из восьми (октет) битов*[1]*.* ***Машинное слово*** *(word) — это двоичное число определенной разрядности, зависящей от архитектуры вычислительной машины.*

Как правило, словом называют единицу информации, имеющую размер в два байта, но встречаются и другие варианты. Часто используются более крупные единицы информации: двойные, учетверённые, восьмикратные и т. п. слова. Лексемы ассемблера `db`, `dw`, `dd`, `dq` и `dt` используются для резервирования областей памяти соответствующих размеров (байт, слово, двойное слово и т. д.).

Приведём также более точную систему определений.

Определение 22. ***Бит*** *(от англ.* ***BI****nary digi****T*** *— двоичная единица) — единица измерения количества информации, равная количеству информации, содержащемуся в опыте, имеющем два равновероятных исхода. Это наименьшая единица информации в цифровом компьютере, принимающая значения «*0*» или «*1*».*

Определение 23. ***Машинное слово*** *(МС) – упорядоченное множество двоичных разрядов, используемое для хранения команд программы и обрабатываемых данных. Каждый разряд, называемый битом, — это двоичное число, принимающее значения только «*0*» или «*1*». Разряды в МС обычно нумеруются справа налево начиная с* 0*. Количество разрядов в МС называется размерностью МС или его разрядностью. Типовая длина МС — 16 или 32 бита.*

Определение 24. ***Байт*** *— машинное слово минимальной размерности, адресуемое в процессе обработки данных.*

[1] В исторической ретроспективе, байт не всегда формировался именно восемью битами, но на данный момент иное уже не имеет места.

Встречаются также более крупные производные единицы информации: килобайт (Кбайт, 1 KB = 10^3 B), мегабайт (Мбайт, 1 MB = 10^3 KB), гигабайт (Гбайт, 1 GB = 10^3 MB), терабайт (Тбайт, 1 TB = 10^3 GB), петабайт (Пбайт, 1 PB = 10^3 TB).

Замечание 2. *Часто возникает неоднозначность в интерпретации, например, 1 Мбайта, который может рассматриваться и как 1 000 Кбайт (десятичный мегабайт), так и 1 024 Кбайт (бинарный мегабайт). С ростом числа увеличивается расхождение, вызванное неоднозначным пониманием использованного префикса. В частности, разница между «двоичным» и «десятичным» килобайтом составляет 2,4%, но между «двоичным» и «десятичным» гигабайтом — уже более 7%.*

Для разрешения этой неоднозначности, в марте 1999 г. Международной электротехнической комиссией (International Electrotecnical Comission — IEC) был предложен новый стандарт для обозначения двоичных чисел. Префиксы IEC схожи с привычными префиксами СИ (Международной системы измерения физических единиц) — они начинаются одинаково, но второй слог двоичных префиксов — би (от англ. binary — «двоичный»).

Стандарт был утвержден, но введенные названия используются нерегулярно, очевидно, из-за их необычности: «килобит» звучит привычнее, нежели «кибибит». Российский ГОСТ 8.417-2002 («Единицы величин») также определяет свое написание двоичных префиксов для байтов. При этом приставки К-, М- и Г- (записанные прописными буквами) применительно к байтам имеют двоичное значение.

Для описания скорости передачи данных можно использовать термин бод. Число бод равно количеству значащих изменений сигнала (потенциала, фазы, частоты), происходящих в секунду. Первоначально бод использовался в телеграфии. Для двоичных сигналов нередко принимают, что бод равен биту в секунду, например 1 200 бод = 1 200 бит/с. Однако единого мнения о правильности использования этого термина нет, особенно при высоких скоростях, где число бит в секунду не совпадает с числом бод.

6.4 Кодирование символьной информации

Определение 25. ***Код (code)*** *— совокупность знаков, символов и правил представления информации.*

Рассмотрим основные понятия и методы дискретного представления информации, или кодирования[2], не вдаваясь слишком подробно в теорию

[2]Которые появились задолго до вычислительных машин.

кодирования.

Определение 26. ***Кодируемые (обозначаемые)*** *элементы входного алфавита обычно называют* ***символами.***

Символом (служит условным знаком какого-нибудь понятия, явления), как правило, является цифра, буква, знак пунктуации или иероглиф естественного языка, знак препинания, знак пробела, специальный знак, символ операции. Кроме этого, учитываются управляющие («непечатные») символы.

Определение 27. ***Кодирующие (обозначающие)*** *элементы выходного алфавита называются* ***знаками.*** *Количество различных знаков в выходном алфавите называют* ***значностью (-арностью, -ичностью,*** *например «бинарный» или «двоичный» код). Количество знаков в кодирующей последовательности для одного символа —* ***разрядностью кода.***

Пространственно-временное расположение знаков кода приводит к понятиям параллельных или последовательных кодов.

Определение 28. *При* ***последовательном коде*** *каждый временно́й такт предназначен для отображения одного разряда символа. Здесь все разряды символа фиксируются по очереди одним и тем же элементом и проходят через одну и ту же линию передачи (например, радио- или оптические сигналы либо передача данных по двум проводам, двухжильному кабелю).*

Определение 29. *При* ***параллельном коде*** *все знаки символа представляются в одном временном такте, каждый знак проходит через отдельную линию (например, по четырём проводам, четырёхжильному кабелю), образуя символ (т. е. символ передается в один приём, в один момент времени).*

Для последовательного кода характерно *временно́е разделение* каналов при передаче информации, для параллельного — *пространственное.* В зависимости от применяемого кода различаются устройства параллельного и последовательного действия.

Пример 3. *Широко известным примером кода является азбука Морзе, в которой буквы латиницы (или кириллицы) и цифры кодируются сочетаниями из «точек» и «тире». Применительно к азбуке Морзе:*

- *символами являются элементы языкового алфавита (буквы A–Z или А–Я) и цифровой алфавит (здесь — цифры 0–9);*

- *знаками являются «·» и «—» (или «+» и «−» либо «1» и «0», короче — любые два разных знака);*
- *поскольку знаков два, то азбука Морзе является двузначным (бинарным, двоичным) кодом;*
- *поскольку число знаков в азбуке Морзе колеблется от 1 (буквы E, T) до 5 (цифры), здесь имеет место код с переменной разрядностью (часто встречающиеся в тексте символы обозначены более короткими кодовыми комбинациями, нежели редкие символы);*

Поскольку знаки передаются последовательно (электрические импульсы, звуковые или оптические сигналы разной длины, соответствующие «точкам» и «тире»), то азбука Морзе — есть последовательный код.

Первые опыты телеграфной и радиосвязи осуществлялись именно посредством азбуки Морзе, причём приемное устройство записывало импульсы переменной длины в виде «·» и «—» на движущуюся телеграфную ленту, однако уже в начале XX в. был осуществлен переход на 5-разрядный (5-битовый) телеграфный код.

Наименование	Расшифровка	Другие названия	Разрядность
Baudot	Код Бодо	IA-1	5
M2	CCITT-2	IA-2	5
ASCII-7	American Standard Code for Information Interchange	ISO-7, IA-5, USASCII, ANSI X3.4	7
ASCII-8	—		8
EBCDIC	Expanded Binary Coded Decimal Information Code		8
Hollerith	Код Холлерита	Код перфокарт	12
UNICODE	UNIversal CODE		16

Таблица 1: Некоторые наиболее известные коды

В табл. 1 приводится перечень наиболее известных кодов,некоторые из которых использовались первоначально для связи, кодирования данных, а затем для представления информации в ЭВМ.

6.5 Избыточные коды

При записи и передаче данных часто используются избыточные коды, т. е. такие, которые за счёт усложнения структуры позволяют повысить надёжность передачи данных.

6.5.1 Коды с обнаружением ошибок

Распространённым методом обнаружения ошибок является **контроль чётности.** В этом случае при записи байта информации в запоминающее устройство генерируется дополнительный контрольный бит, в который записывается «0», если количество единиц в исходном байте информации чётное, и «1», если оно нечётное. Если при чтении ранее записанного байта вновь получить контрольный бит и сравнить его с уже имеющимся, то можно судить о достоверности получаемой информации.

Широко используется для обнаружения ошибок в блоках данных также код с циклическим контролем — **циклический избыточный код обнаружения ошибок (CRC — Cyclic Redundance Check).** Здесь вычисляется контрольная сумма содержимого блока данных перед его передачей, которая включается в одно из полей блока, а затем она повторно вычисляется после передачи. Несовпадение результатов свидетельствует об ошибке в передаваемом содержимом.

6.5.2 Корректирующие коды

В ответственных приложениях, требующих повышенной надёжности хранения информации, применяются более серьёзные, чем контроль чётности, методы обеспечения целостности данных. К ним относятся корректирующие коды (ECC — Error Correction Code), позволяющие не только обнаруживать ошибки, но и восстанавливать искажённую информацию за счет её избыточности. Так, существуют модули памяти со схемами ECC, в которых для хранения контрольной информации используется не один, а два бита, в которых хранится остаток от деления числа на 4 (деление по модулю 4). Благодаря этим данным схема ECC умеет обнаруживать и исправлять одиночные искажённые биты, а также обнаруживать (но не исправлять) двойные ошибки. Модули памяти с ECC обычно стоят заметно дороже и применяются в основном в серверах. В общем случае ECC применяются во всех современных дисковых и ленточных накопителях. За счёт информационной избыточности закодированных данных удается восстанавливать поврежденные блоки информации длиной в сотни байт. Наиболее широко применяются помехоустойчивые коды Рида — Соломона (Reed — Solomon), а также код Хемминга, позволяющий исправлять одиночные ошибки, появляющиеся в блоках данных.

7 Математические основы

7.1 Представление чисел в позиционных системах счисления

В современной информатике для записи чисел в подавляющем большинстве случаев используются позиционные системы счисления.

Определение 30. ***Система счисления*** *— это совокупность правил записи чисел цифровыми символами.*

Определение 31. ***Позиционными системами счисления*** *называются такие системы, в которых действительное значение цифры зависит от её позиции (порядкого номера) в записи числа. Другими словами, каждая позиция в записи имеет свой вес.*

Определение 32. ***Непозиционная система счисления*** *— это система, в которой значение символа не зависит от его положения в числе.*

Пример 4. *Римская система счисления*[3]*: IV, V, VI, XIV, ...*

Основные характеристики позиционной системы счисления:

1. основание системы — это число различных цифр в системе: B;
2. наибольшая цифра: $B - 1$.

Любое число M в позиционной системе может быть представлено в следующей форме:

$$M = \sum_{i=-k}^{n-1} A_i \cdot B^i = A_{n-1}A_{n-2}\dots A_1A_0.A_{-1}A_{-2}\dots A_{-k+1}A_{-k}. \qquad (1)$$

Здесь B — основание системы счисления, A_i — значение i-го разряда, n — число разрядов целой части, k — число разрядов дробной части, B^i — вес i-го разряда.

Пример 5.

$$\pi \approx 3.1415 = 3 \cdot 10^0 + 1 \cdot 10^{-1} + 4 \cdot 10^{-2} + 1 \cdot 10^{-3} + 5 \cdot 10^{-4}. \qquad (2)$$

[3]Строго говоря, это пример смешанной системы счисления.

В принципе, для создания позиционной системы счисления может быть использовано любое число цифр, начиная от двух. Используя в качестве основания системы число n, можно задействовать цифры от 0 до $n-1$. В таком случае, в любой системе счисления запись числа 1000_n означает n^3.

Замечание 3. *Система счисления определяет только правила записи числа. Само число и его свойства от выбора системы не зависят, т. е. простое число всегда останется простым, чётное-чётным и т. д.*

Система	Основание	Обозначение	Использование в C
Двоичная	2	b (binary)	Префикс `0b`
Восьмеричная	8	o (octal)	Ведущий `0`
Десятичная	10	d (decimal)	По умолчанию
Шестнадцатеричная	16	h (hexadecimal)	Префикс `0x`

Таблица 2: Основные позиционные системы счисления

Пример 6.

$$25_{10} = 11001_2 = 31_8 = 19_{16}. \tag{3}$$

В информатике базовой является именно двоичная система счисления, т. к. именно в виде нулей и единиц в памяти компьютера хранится вся информация.

Теорема 1. *Если основание одной системы счисления представляет собой натуральную степень n основания другой системы счисления, то одна цифра первой системы в точности соответствует n цифрам второй системы.*

На практике эта теорема применяется для преобразования представления числа в двоичной ($B = 2$), восмеричной ($B = 8 = 2^3$) и шестнадцатеричной ($B = 16 = 2^4$) системах. Примеры использования приведены в таб. 3.

7.2 Числа конечной точности

Так как информация в ВТ хранится в ячейках ограниченной разрядности, то программисту приходится иметь дело с числами конечной точности, не замкнутыми относительно арифметических операций ($+$, $-$, $\cdot$, $/$). Это приводит к появлению следующих ошибок при вычислениях:

oct	bin	hex	bin	hex	bin
0	000	0	0000	8	1000
1	001	1	0001	9	1001
2	010	2	0010	A	1010
3	011	3	0011	B	1011
4	100	4	0100	C	1100
5	101	5	0101	D	1101
6	110	6	0110	E	1110
7	111	7	0111	F	1111

Таблица 3: Связь представлений чисел в двоичной, восьмеричной и шестнадцатеричной системах счисления

1. ошибки переполнения;
2. ошибки потери значимости;
3. результат операции не является числом (NaN — Not a Number).

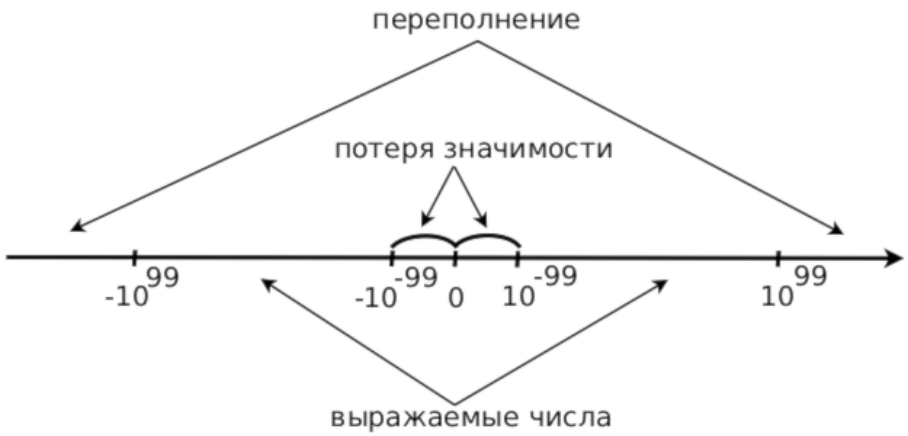

Рис. 9: Числа конечной точности как подмножество $\mathbb{R}$ на примере двузначного порядка

Пример 7. *Для чисел конечной точности в общем случае не имеет места ассоциативность и дистрибутивность:*

$$a + (b - c) \neq (a + b) - c, \tag{4}$$

$$a \cdot (b - c) \neq a \cdot b - a \cdot c. \tag{5}$$

Ещё одно отличие чисел конечной точности от действительных чисел состоит в том, что они заполняют числовую ось с переменной плотностью, что может приводит к ошибкам округления и накоплению погрешности вычислений.

Определение 33. ***Машинный ноль*** *— числовое значение с таким отрицательным порядком, которое воспринимается машиной как ноль*[4].

Определение 34. ***Машинный эпсилон*** ε *— это числовое значение, меньше которого невозможно задавать относительную точность для любого алгоритма, возвращающего вещественные числа. Абсолютное значение* ε *зависит от разрядности регистров ЭВМ, типа (разрядности) используемых чисел и от принятой в конкретном трансляторе структуры представления вещественных чисел (количества бит, отводимых на мантиссу и на порядок). Формально* ε *определяют двумя альтернативными способами:*

1. *как минимальное из чисел* ε*, для которого* $1 + \varepsilon > 1$ *при машинных расчётах с числами данного типа;*

2. *либо как максимальное* ε*, для которого справедливо равенство* $1 + \varepsilon = 1$.

Пример 8. *В языке Си машинный эпсилон определяют константы* `FLT_EPSILON`*,* `DBL_EPSILON` *и* `LDBL_EPSILON`*.*

7.3 Представление чисел в компьютерах

В ЭВМ применяются две формы представления чисел:

- естественная форма, или форма с фиксированной точкой;
- нормальная форма, или форма с плавающей точкой.

7.3.1 Числа с фиксированной точкой

В форме представления с фиксированной точкой числа изображаются в виде последовательности цифр с постоянным для всех чисел положением точки, отделяющей целую часть от дробной.

Пример 9. *Пусть для представления чисел в основных системах счисления используется пять разрядов в целой части числа (до точки) и три — в дробной части (после точки). Записанные в такую разрядную сетку значения могут иметь вид:*

```
+01221.075;
+000B0.125h;
-70301.123o.
```

[4]На рис. 9 этому значению соответствует любое число $x : |x| < 10^{-99}$.

Эта форма наиболее проста, естественна, но имеет небольшой диапазон представления чисел. Диапазон значащих чисел N в системе счисления с основанием B при наличии n разрядов в целой части и k разрядов в дробной части числа (без учета знака числа) будет таким:

$$B^{-k} \leqslant N \leqslant B^n - B^{-k}. \tag{6}$$

Пример 10. *При $B = 2$, $n = 10$ и $k = 6$ представимые числа изменяются в диапазоне*

$$0.015625 \leqslant N \leqslant 1023.984375. \tag{7}$$

Если в результате операции получится число, выходящее за допустимые пределы, произойдет переполнение разрядной сетки, и дальнейшие вычисления теряют смысл. В современных компьютерах естественная форма представления используется как вспомогательная и только для целых чисел.

В памяти ЭВМ числа с фиксированной точкой чаще всего хранятся в трёх форматах:

- полуслово — это обычно 16 бит или 2 байта (`short, dw`);
- слово — 32 бита или 4 байта (`int, dd`);
- двойное слово — 64 бита или 8 байтов (`long int, dq`).

7.3.2 Запись отрицательных чисел

Отрицательные числа с фиксированной точкой записываются в разрядную сетку чаще всего в дополнительном коде, который образуются прибавлением единицы к младшему разряду обратного кода. Обратный код получается заменой единиц на нули, а нулей на единицы в прямом двоичном коде.

Таким образом, существует четыре основных системы представления отрицательных чисел[5].

1. Знаковая система. Старший бит кодирует знак числа. Обычно считается, что нуль используется для записи положительных чисел, а единица обозначает отрицательные.

2. Дополнение до единицы (или обратный код). Знаковый бит + инверсия всех остальных разрядов.

[5]Некоторые подробности можно прочитать здесь: http://www.cs.emory.edu/~cheung/Courses/255/Syllabus/5-repr/excess-n.html.

3. Дополнение до двух (или дополнительный код). Знаковый бит + инверсия + добавление 1.

4. Система со смещением (как правило, используются варианты Excess-2^{n-1} или Excess-$2^{n-1}-1$). Сумма исходного числа со смещением. Здесь предполагается, что для записи числа используется n двоичных разрядов.

Пример 11. *Пусть требуется записать отрицательное число* -27_{10} *в* $n = 8$ *двоичных разрядах. Тогда в знаковой системе ему соответствует представление* `10011011`*, а во второй, третьей и четвёртой —* `11100100`*,* `11100101` *и* `01100101`*, соответственно*[6].

7.3.3 Числа с плавающей точкой

Любое такое число M может быть представлено в виде

$$M = \pm f \cdot B^e, \tag{8}$$

где f — это дробь (мантисса) числа ($|f| < 1$), которая определяет точность, с которой мы знаем M; e — экспонента (целое число), задающая порядок (область значений) числа.

Пример 12. *Преобразование двоичного числа с плавающей точкой к десятичному представлению выполняется очевидно*

$$101.01101_2 = 4 + 1 + 1/4 + 1/8 + 1/32 = 5.40625_{10}. \tag{9}$$

Теперь продемонстрируем обратное преобразование $5.40625_{10} \to x_2$. *В первую очередь запишем целую часть исходного числа в двоичной системе* $5_{10} = 101_2$. *После этого будем последовательно умножать на два дробную часть исходного числа и всех чисел, получающихся в этом процессе, выписывая значения их целых частей. Этот процесс продолжим до тех пор пока дробная часть не обратится в нуль.*

$$\begin{aligned} 0.40625 \cdot 2 &= 0.8125 \\ 0.8125 \cdot 2 &= 1.625 \\ 0.625 \cdot 2 &= 1.25 \\ 0.25 \cdot 2 &= 0.5 \\ 0.5 \cdot 2 &= 1.0 \end{aligned}$$

$$\Rightarrow 0.40625_{10} = 0.01101_2$$

[6] Обратите внимание на связь третьего и четвёртого представлений.

Затем, складывая целую и дробную части, окончательно получаем $5.40625_{10} = 101.01101_2$.

Замечание 4. *В виде конечной двоичной дроби представимы такие и только такие рациональные числа, знаменатель которых является степенью двойки.*

Определение 35. ***Нормализованными*** *называются такие числа с плавающей точкой* M*, для которых выполнено условие*

$$B^{-1} \leqslant |f| < 1. \tag{10}$$

Пример 13. *Числа, использованные в Примере 9, в нормализованной форме будут выглядеть так:*

$$\begin{aligned} +01221.075 &= +0.1221075 \cdot 10^4; \\ +000B0.125h &= +0.B0125h \cdot 16^2; \\ -70301.123o &= -0.70301123o \cdot 8^5. \end{aligned} \tag{11}$$

Числа с плавающей точкой хранятся в ВМ как правило в нормализованном виде (хотя возможно использование и денормализованных чисел), так как нормальная форма представления обеспечивает большой диапазон отображения чисел.

Утверждение 1. *Правила представления чисел с плавающей точкой регламентирует международный стандарт IEEE 754-2008 (см. рис. 10), согласно которому нормализованными двоичными числами называют такие числа для которых* $1 \leqslant |f| < 2$.

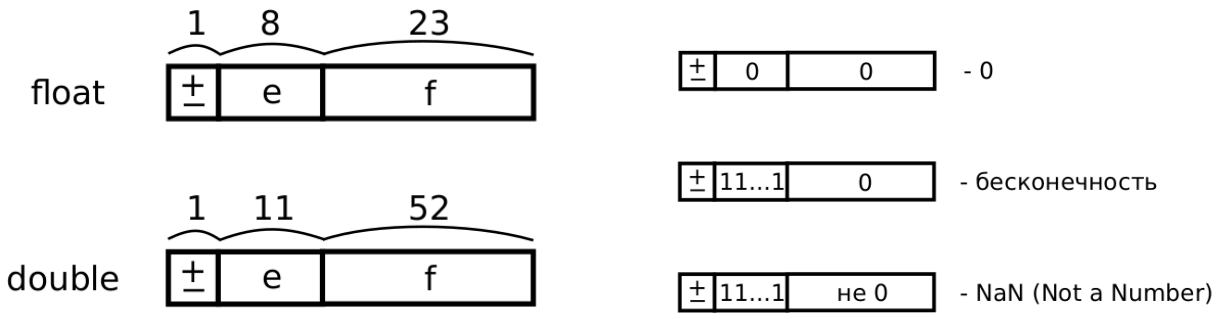

Рис. 10: Некоторые соглашения стандарта IEEE 754-2008

Подробности о стандарте IEEE 754 можно прочитать здесь: http://www.softelectro.ru/ieee754.html.

7.4 Разновидности «машинной арифметики»

- Обычная двоичная арифметика.
- Двоично-десятичная арифметика.
- Циклическая арифметика (WrapAround).
- Арифметика с насыщением (Saturation Arithmetic — SArith).
- Векторные операции над упакованными данными («упакованная арифметика» — packed arithmetic).
- «Горизонтальная» и «вертикальная» арифметика.

7.5 Логические основы ВМ

Начало исследований в области формальной логики было положено работами Аристотеля в IV веке до нашей эры. Однако строго формализованный подход к проблеме впервые был предложен Дж. Булем. В честь него алгебру высказывания называют булевой алгеброй, а логические значения — булевыми. Основу математической логики составляет алгебра высказываний. Алгебра логики используется при построении основных узлов компьютеров (дешифратор, сумматор, шифратор).

Алгебра логики оперирует высказываниями. Под высказыванием понимают повествовательное предложение, относительно которого можно утверждать, истинно оно или ложно.

Высказывания (логические переменные) принято обозначать буквами латинского алфавита. Если высказывание x истинно, это обозначается как $x = 1$ (true), а если оно ложно, то $x = 0$ (false).

В алгебре логики над высказываниями можно производить определенные логические операции, в результате которых получаются новые высказывания. Истинность результирующих высказываний зависит от истинности исходных и использованных для их преобразования логических операций.

8 Контрольные вопросы и задания

1. С какими архитектурами компьютеров вы работали? Каковы их основные особенности? Наидите и изучите информацию о неизвестных вам архитектурах.

$x = 0$, если $x \neq 1$	$x = 1$, если $x \neq 0$
$x = 0,\ \overline{x} = 1$	$x = 1,\ \overline{x} = 0$
$1 \cdot 0 = 0 \cdot 1 = 0$	$1 + 0 = 0 + 1 = 1$
$1 \cdot 1 = 1$	$0 + 0 = 0$
$0 \cdot 0 = 0$	$1 + 1 = 1$
$x \cdot 1 = x$	$x + 0 = x$
$x \cdot 0 = 0$	$x + 1 = 1$
$x \cdot x = x$	$x + x = x$
$x \cdot \overline{x} = 0$	$x + \overline{x} = 1$
$\overline{\overline{x}} = x$	
$x \cdot y = y \cdot x$	$x + y = y + x$
$x + x \cdot y = x$	$x \cdot (x + y) = x$
$(x + \overline{y}) \cdot y = x \cdot y$	
$x \cdot y + x \cdot \overline{y} = x$	$(x + y)(x + \overline{y}) = x$
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	$(x + y) + z = x + (y + z)$
$(x + y) \cdot (x + z) = x + y \cdot z$	$x \cdot y + x \cdot z = x \cdot (y + z)$
$\overline{x \cdot y \cdot z} = \overline{x} + \overline{y} + \overline{z}$	$\overline{x + y + z} = \overline{x} \cdot \overline{y} \cdot \overline{z}$

Таблица 4: Основы булевой алгебры

2. Каковы, на ваш взгляд, преимущества и недостатки использования IDE для разработки ПО? С какими инструментами разработчика у вас есть опыт работы?

3. Назовите существующие интерфейсы взаимодействия пользователя с операционной системой. Каковы их преимущества и недостатки?

4. Ознакомьтесь более подробно с теми определениями информации, которые были перечислены в разделе 6.1. Какой смысл в понятие «информация» вкладывали различные авторы?

5. Найдите и изучите материалы об использовании десятичных и двоичных единиц измерения информации в разных областях компьютерных наук. Что вы можете сказать о двоично-десятичных единицах измерения информации?