АЛЕКСАНДР ПАНОВ

bhv

РЕВЕРСИНГ и ЗАЩИТА ПРОГРАММ ОТ ВЗЛОМА



Наиболее распространенные способы защиты программ и способы их преодоления

Примеры работы с отладчиками и дизассемблерами

Подробное описание отладчика OllyDBg

Методика написания средств защиты программ от взлома

PRO

ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

+CD

Александр Панов

РЕВЕРСИНГ и ЗАЩИТА ПРОГРАММ ОТ ВЗЛОМА

Санкт-Петербург

«БХВ-Петербург»

2006

УДК 681.3.06
ББК 32.973.26-018.1
П16

Панов А. С.

П16 Реверсинг и защита программ от взлома. — СПб.: БХВ-Петербург, 2006. — 256 с.: ил.

ISBN 5-94157-889-X

Подробно изложены современные основные методики защиты программного обеспечения, начиная с составления программ и заканчивая их отладкой. Рассмотрены примеры взлома стандартных защит и даны рекомендации для предотвращения такого взлома. Приведено большое количество рабочих примеров, которые должны помочь программисту решить возникшие перед ним проблемы в защите его интеллектуальной собственности. Подробно описана работа с отладчиком OllyDBg. На сопроводительном компакт-диске находятся описанные в книге программы.

Для программистов

УДК 681.3.06
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Татьяна Темкина*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Наталия Першакова*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.06.06.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 20,64.
Тираж 2500 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 5-94157-889-X

© Панов А. С., 2006
© Оформление, издательство "БХВ-Петербург", 2006

Оглавление

Глава 1. Введение 1

Для кого предназначена эта книга 1
Благодарности 2
Кто такой хакер и как им стать 2
Кто такой крекер и как им стать 3
Что понадобится для работы с книгой 5

Глава 2. Особенности ассемблера для исследования работы программ 7

Структурная схема компьютера 7
Архитектура микропроцессора 9
Организация памяти 12
Представление данных 15
Целые числа 15
Двоично-десятичные числа 16
Числа с плавающей точкой 17
Символьные данные 19
Представление данных сопроцессора 19
Регистры микропроцессора 20
Регистры общего назначения 20
Регистровые указатели и индексные регистры 22
Сегментные регистры 22
Регистры состояния и управления 23
Способы адресации 29
Работа со стеком 34
Логика и организация программы 35
Безусловный переход 35
Команды условного перехода 37
Команды сравнения 38
Команды цикла 39
Логические операции и команды сдвига 41
Организация циклов 44
Вызов подпрограмм 45

Структура COM- и EXE-программ. Размещение программы в памяти 51
Ввод и вывод данных 53
Определение данных 53
Директивы определения данных 54
Команды пересылки и строковые операции 56
Прерывания 63
Система прерываний 64
Обработка прерывания 64
Собственные программы обработки прерываний 65
Установка и чтение адреса обработчика прерывания с помощью функций DOS 68
Прерывания для ввода/вывода символов 69
Арифметические операции 75
Сложение и вычитание 75
Увеличение и уменьшение 75
Умножение и деление 76
Команды *lea* и *xlat* 78
Загрузка исполнительного адреса (команда *lea*) 78
Кодирование по таблице (команда *xlat*) 78

Глава 3. Основные способы ввода/вывода 81

Ввод данных с клавиатуры 81
Преобразование и вывод данных на экран 83

Глава 4. Отладчики 87

Отладчик DEBUG 87
Отладчик Turbo Debugger 88
Отладчик SoftICE 96
Особенности 96
Установка 97
Настройка 99
Окно регистров 101
Окно данных 101
Окно кода 102
Окно команд 102
Точки останова 103
Горячие клавиши 107
Наиболее важные точки останова 108
Интерактивный дизассемблер IDA 109
Отладчик OllyDbg 119
Главное окно программы 120
Горячие клавиши 131
Плагины для отладчика OllyDbg 134

Глава 5. Как именно взламывают программы ... 145

Первые шаги в реверсинге или "С чего начать?" ... 145
Инструменты — главные орудия крекера ... 148
Типовая защита программ "Запрос пароля" ... 154
Без шифрования сообщений ... 154
С шифрованием сообщений ... 164
Улучшенная защита программ с шифрованием сообщений ... 171
Программы с динамической проверкой ввода пароля ... 178

Глава 6. Основные способы защиты программ ... 187

Проверка даты и времени ... 187
Практический пример ... 187
Инструмент InqSoft Sign Of Misery ... 191
Фиксированное количество запусков программы ... 192
Программы, защищенные CRC ... 201
Более сложный алгоритм проверки CRC ... 210
Классические алгоритмы нахождения CRC и методика их применения в защите ПО ... 214
Некоторые менее распространенные способы защиты программ ... 217
Шифрование всей программы или ее части ... 217
"Отлов" пошаговой отладки программы ... 219
Проверка на основе динамически изменяющихся параметров ... 219
Регистрация он-лайн ... 220

Глава 7. Методы противодействия и способы защиты программ от взлома ... 221

Как не надо проектировать защиту программ ... 222
Как "обмануть" отладчики и дизассемблеры ... 226
Функция *IsDebuggerPresent* ... 227
Функция *CreateFileA* ... 229
Обман дизассемблера ... 230
Усложнение листинга ... 231

Заключение ... 233

Приложение. Описание компакт-диска ... 235

Список литературы ... 237

Предметный указатель ... 239



Глава 2

Особенности ассемблера для исследования работы программ

В этой главе кратко описаны принципы работы персонального компьютера и ассемблера (языка Assembler). Для более подробного изучения языка ассемблер я рекомендую вам прочитать мою книгу "Assembler: Экспресс курс". Это пригодится вам для того, чтобы лучше понять, как именно можно спроектировать защиту программы или как ее можно снять (это зависит от целей, которые вы перед собой ставите).

Желающим побыстрее начать что-то делать рекомендую перейти сразу к четвертой главе; если понадобится, вы сможете вернуться к этой главе позднее. Но если вы ступили на путь реверсинга программ впервые, то я рекомендую вам читать все по порядку. Это поможет вам понять многие принципы, сэкономив много времени.

Приведенные здесь основы ассемблера позволят лучше понять, что же именно мы будем делать, как устроена защита той или иной программы и т. д. Я не буду объяснять весь синтаксис ассемблера. Это нам ни к чему.

Мы рассмотрим принципы работы компьютера и микропроцессора на примере микропроцессора 8086. Сейчас у многих имеются компьютеры с гораздо более совершенной архитектурой (AMD Athlon XP, Intel Pentium и др.) но все, что я расскажу о микропроцессоре 8086, справедливо и для всех остальных моделей процессоров (естественно, речь идет только об архитектуре PC).

Структурная схема компьютера

Структурная схема персональной ЭВМ, работающей на базе микропроцессора, представлена на рис. 2.1.

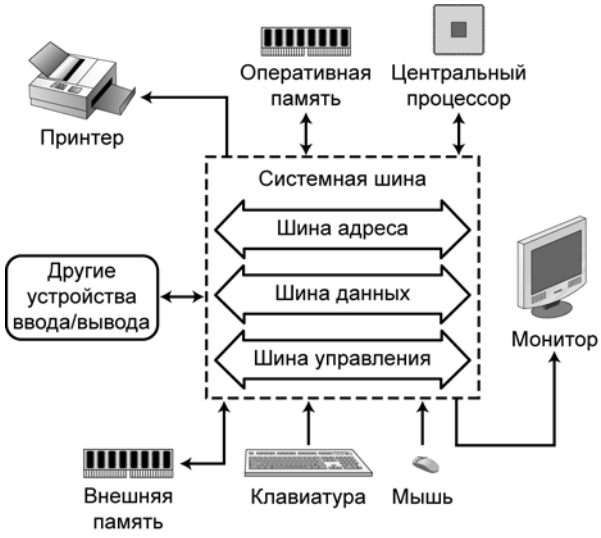


Рис. 2.1. Структурная схема микрокомпьютера

При работе персонального компьютера (также называемого *микрокомпьютером*) выполняются следующие действия:

1. Программа или данные вводятся с помощью устройств ввода из внешней памяти (жесткий диск, дискета и т. д.) в оперативную память.
2. Микропроцессор выбирает команды программы из оперативной памяти в строгом порядке, в котором их следует выполнять. Микропроцессор считывает указанные в этих командах данные и производит указанные командами действия. Адреса и данные передаются через *системную шину*, соединяющую центральный процессор, память и устройства ввода/вывода (рис. 2.1).
3. Данные, полученные в результате обработки команд, микропроцессор пересылает в оперативную память или на устройство вывода (дисплей, принтер, внешняя память и т. д.).

Микропроцессор расположен на *системной плате*, где также расположены микросхемы оперативной памяти и микросхемы *постоянного запоминающего устройства (ПЗУ)*, содержащего программу начальной загрузки памяти ЭВМ и другие вспомогательные программы. Различные компоненты компьютера подсоединяют к системной шине, состоящей из адресной шины, шины данных и шины управления. Связь между процессором, памятью и устройствами ввода/вывода осуществляется тоже через системную шину. Процессор подсоединяется к шине через блок логики управления шиной, устройства — через интерфейсы.

Архитектура микропроцессора

Главное, что должен делать центральный процессор, — упрощать работу со следующими объектами вычислительной системы:

- присваивания и арифметические выражения;
- безусловные/условные переходы;
- логические выражения;
- циклы;
- массивы/структуры данных;
- подпрограммы;
- устройства ввода/вывода.

Стандартная архитектура центрального процессора, обладающего такими возможностями, должна включать:

- набор регистров для адресации данных и выполнения арифметических операций;
- устройство управления для исполнения команд;

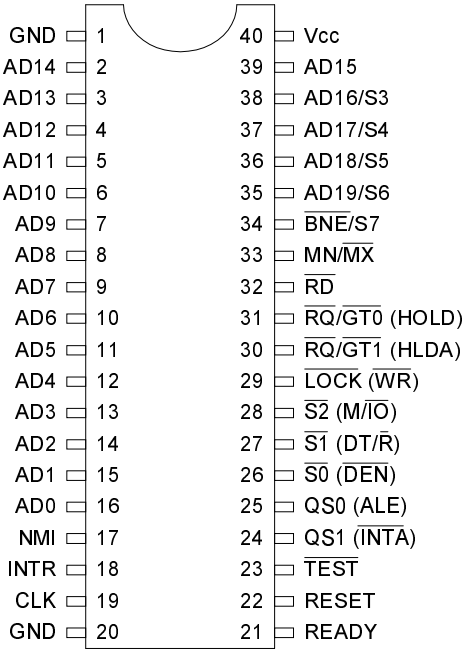


Рис. 2.2. Разводка контактов микропроцессора Intel 8086

- арифметико-логическое устройство (АЛУ) для выполнения арифметических и логических операций;
- секцию управления вводом/выводом.

Под *регистром* мы будем понимать запоминающее устройство для временного хранения целого числа байт, расположенное внутри микропроцессора.

Рассмотрим эти компоненты на примере микропроцессора Intel 8086 (рис. 2.2).

В компьютерах на базе данных процессоров используется системная шина, называемая *мультиплексной*, поскольку по ее линиям передаются как адреса, так и данные.

Сигналы принимаются и передаются через мультиплексную шину микропроцессором, который делится на две части — операционное устройство и шинный интерфейс (рис. 2.3).

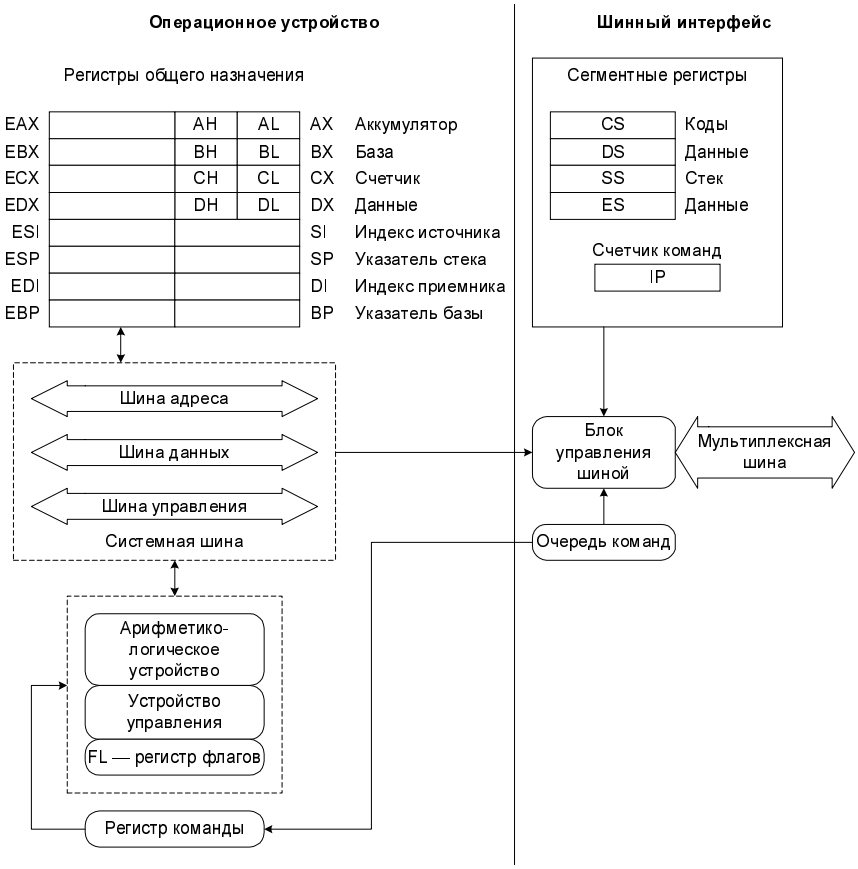


Рис. 2.3. Внутренняя архитектура микропроцессора Intel 8086

Операционное устройство предназначено для выполнения команд. Оно содержит АЛУ, устройство управления и 10 регистров.

Шинный интерфейс подготавливает команды и данные для выполнения и состоит из блока управления шиной, очереди команд и сегментных регистров. Он управляет передачей данных на операционное устройство, в память и на устройства ввода/вывода. В шинном интерфейсе адрес формируется с помощью сегментных регистров. Шинный интерфейс также имеет доступ к командам, находящимся в оперативной памяти, осуществляет выборку этих команд и помещает их в очередь команд. При этом шинный интерфейс должен как бы "заглядывать вперед" и выбирать команды так, чтобы всегда существовала непустая очередь команд, готовых для выполнения.

Согласно рис. 2.3 микропроцессор 8086 имеет 14 16-разрядных регистров, доступных для программиста. Начиная с микропроцессора i80386 пользовательских регистров стало 16 — добавились два сегментных регистра FS и GS. Остальные регистры расширились до 32 бит. Основные регистры микропроцессоров семейства 80x86 показаны на рис. 2.4. Регистры позволяют свести операции к командам, имеющим не более одного операнда, находящегося в памяти.

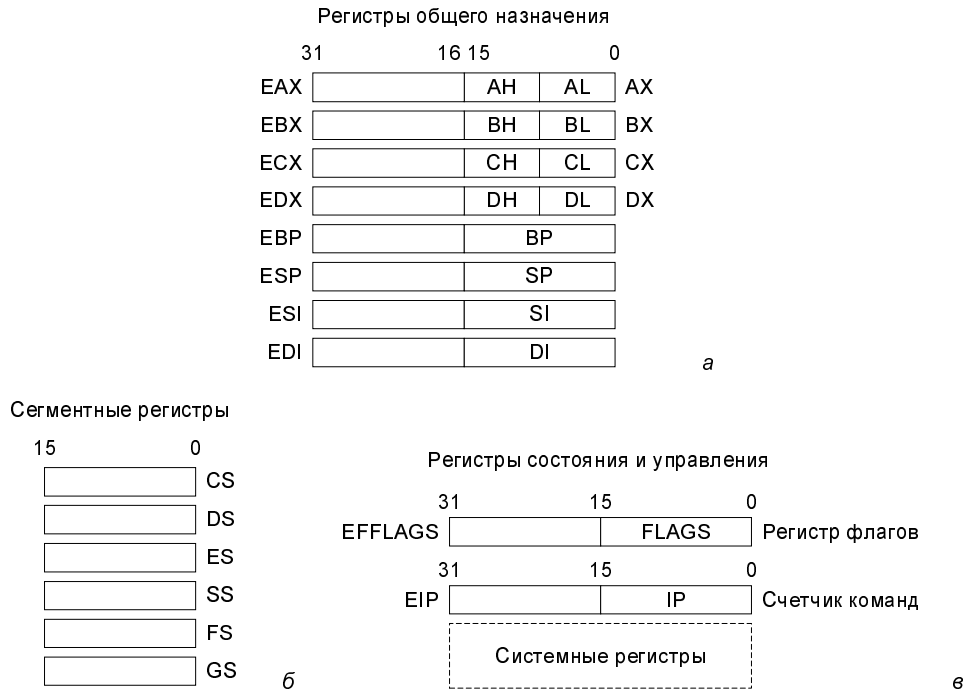


Рис. 2.4. Регистры микропроцессоров семейства Intel 80x86

Организация памяти

Внутренняя память микрокомпьютера делится на два типа.

Первый тип — это память, доступная только для чтения и называемая *постоянной памятью* (Read Only Memory, ROM). ROM-память представляет собой специальную микросхему, находящуюся на материнской плате, данные в которую записываются на заводе — изготовителе материнской платы. Основное назначение ROM-памяти — поддержка процедуры начальной загрузки. Для программиста наиболее важным элементом ROM-памяти является *базовая система ввода/вывода* (Basic Input/Output System, BIOS).

Память второго типа называется *памятью произвольного доступа* (Random Access Memory, RAM), или *оперативной памятью*. Эта память доступна для чтения и записи и является памятью, с которой работает программист. Содержимое оперативной памяти теряется при выключении питания компьютера. Поэтому для сохранения данных и программ требуются средства *внешней памяти* (обычно жесткий диск или дискета).

Память ЭВМ состоит из последовательности ячеек, каждая из которых имеет *адрес*. Таким образом, *память* — это область хранения информации, доступ к которой осуществляется по шине адреса.

Минимальная компьютерная единица информации называется *битом*. Бит может принимать только одно значение из двух — 0 или 1. Память разбивается на блоки различных размеров — байт, слово, сегмент, страница и др. Рассмотрим их подробнее.

Байтом называется минимальная адресуемая единица информации. Байт состоит из восьми бит. Таким образом, чтобы изменить один бит памяти, надо записать, по крайней мере, восемь бит в байт, содержащий этот бит. Каждому байту соответствует его собственный физический адрес.

Словом называется последовательность бит, число которых равно размеру регистра-аккумулятора. Например, в микропроцессоре 8086 регистр AX имеет длину 16 бит, т. е. слово состоит из 16 бит, или содержит 2 байта. В более современных компьютерах аккумулятором является регистр EAX, а это значит, что слово состоит из 32 бит (4 байта). В этом случае его называют *двойным словом*, оставляя название "слово" за областью памяти, состоящей из 16 бит.

Примечание

В 16-разрядных процессорах: слово — 16 бит, двойное слово — 32 бита, в 32-разрядных процессорах: слово — 32 бита, двойное слово — 64 бита.

Микропроцессор работает в одном из следующих режимов:

- *реальный режим* — режим, в котором работал процессор 8086;
- *защищенный режим* — режим, позволяющий максимально реализовать возможности процессоров 80*x*86 (начиная с процессора 80286 в этом режиме физический адрес формируется не так, как в реальном режиме);
- *режим виртуального 8086* — режим, возможный при работе процессора в защищенном режиме; в этом режиме могут работать несколько программ, разработанных для 8086; процесс формирования физического адреса для этих программ производится как в реальном режиме;
- *режим системного управления* — режим, появившийся в микропроцессоре Pentium; функционирование процессора в этом режиме по общим принципам подобно его работе в реальном режиме.

Мы только начинаем изучать язык ассемблера, поэтому для простоты работы будем рассматривать адресацию памяти в реальном режиме.

Сегментом называется независимый, поддерживаемый на аппаратном уровне блок памяти, имеющий размер не больше 2^{16} байт (поскольку $2^{16} = 2^6 \cdot 2^{10} = 64 \cdot 1024$, а 1024 байт = 1 Кбайт, максимальный размер сегмента будет равен 64 Кбайт).

Параграфом называется минимальный размер сегмента 16 байт.

Адресная шина процессора 8086 состоит из 20 линий, поэтому максимальный адресуемый объем памяти в реальном режиме равен 2^{20} байт = 2^{10} Кбайт = = 1 Мбайт. В защищенном режиме для процессора 80286 он равен 2^{24} байт = 16 Мбайт, а для процессоров 80386 и выше — 2^{32} байт = 4 Гбайт.

Память разделяется на сегменты. Адрес ячейки формируется подобно почтовому адресу (сегментный адрес указывает "улицу", а смещение — "номер дома"). Физический адрес байта, имеющего смещение (т. е. номер байта в сегменте) *x* относительно начала сегмента, указанного с помощью сегментного регистра S, будет равен $16 \times S + x$. Например, регистр CS (сегмент кодов) используется для указания сегмента, в котором находится код программы. Так как содержимое счетчика команд IP равно смещению текущей команды относительно начала сегмента, физический адрес EA текущей команды будет равен $16 \cdot CS + IP$.

Вернемся к рис. 2.3 и рассмотрим процессы выполнения команд и обмена данными между микропроцессором и оперативной памятью.

1. По содержимому регистров CS и IP, хранящих адрес команды в виде базового адреса, т. е. адреса начала сегмента (CS), и смещения (номера байта в сегменте) (IP), определяется местонахождение (EA) команды, выполняемой на следующем шаге.

2. Вычисленный адрес запоминается в стеке.
3. Из оперативной памяти считывается команда с этим адресом и пересылается в процессор.
4. Команды помещаются в очередь команд и выполняются в порядке очередности.
5. Местоположение данных, участвующих в командах, определяется в шинном интерфейсе по информации, содержащейся в адресной части команды, в регистрах сегментов DS, SS и ES, а также в регистрах SP, BP, SI, DI.
6. Данные, прочитанные из памяти, пересылаются в АЛУ или регистр данных (DS) и обрабатываются текущей командой, адрес которой был найден на шаге 2. Результаты обработки данных помещаются в оперативную память или в какой-либо из регистров.

Циклическое повторение описанных процедур составляет выполнение программы. Номер сегмента называется *адресом сегмента (сегментным адресом)*, а номер байта в сегменте — *смещением*. На практике указывают лишь смещение, а также по умолчанию предполагается, что значение регистра DS известно.

Надо помнить, что любой сегмент имеет объем до 64 Кбайт и имеется 4 сегментных регистра; таким образом, доступна память 256 Кбайт. Но в действительности в программе возможно любое количество сегментов. Для того чтобы адресовать любой сегмент, достаточно изменить значение сегментного регистра.

Память RAM включает в себя первые три четверти внутренней памяти, а ROM — последнюю четверть. В памяти объема 1 Мбайт ROM начинается с 768-го килобайта. Распределение памяти при работе в операционной системе MS-DOS показано на рис. 2.5 (слева указаны начальные адреса блоков памяти).

F0000	Основная системная память
C0000	Дополнительная постоянная память
A0000	Видеобуфер
	Пользовательские программы
50000	ROM BIOS
40000	Расширение оперативнй памяти в канале I/O
0	Основная оперативная память

Рис. 2.5. Распределение памяти в MS-DOS

Представление данных

Данные в памяти ЭВМ имеют двоичное представление. В самом общем случае мы можем отвести под целое число любое количество соседних байтов. Программисту приходится работать с числами разных форматов (не только с целыми). Рассмотрим некоторые наиболее часто используемые типы данных:

- целые числа;
- двоично-десятичные числа;
- неупакованный BDC-формат;
- упакованный BDC-формат;
- числа с плавающей точкой;
- символьный тип.

Для быстрого старта этого достаточно, а подробную информацию о любых типах данных вы можете получить из специальных источников.

Целые числа

Система команд ЭВМ поддерживает работу с целыми числами *без знака* и *со знаком*, занимающими байт, слово или двойное слово.

Целое число без знака в двоичной записи состоит из k бит, где $k = 8$, 16 или 32.

Байтом 1010 1100 представляется число $2^7 + 2^5 + 2^3 + 2^2 = 172$, а словом 0111 0111 0000 1001 — число $7 \cdot 16^3 + 7 \cdot 16^2 + 9 = 30\,473$.

Самое большое число, которое можно представить байтом, равно 255, а наибольшее число, представляемое словом, — $2^{16} - 1 = 65\,535$.

Диапазоны значений целых чисел со знаком, имеющих разную длину:

- байт: от –128 до 127;
- слово: от –32 768 до 32 767;
- двойное слово: –2 147 483 648 до 2 147 483 647.

Целые числа со знаком записываются в дополнительном коде: положительное число записывается как беззнаковое (его старший бит равен 0), а отрицательное число $-x$ (при $x > 0$) представляется как $2^k - x$, где k — количество разрядов ячейки памяти, отведенной под число (k = 8,16 или 32). Если в этой ячейке записано некоторое беззнаковое число z, старший бит которого равен 1, то это число представляет число со знаком, равное $-(2^k - z)$.

Например, байт 10101100 является беззнаковым числом 172, которое превращается в –(256 – 172) = –84, если рассматривать его как число со знаком.