Представление данных в микропроцессорных системах

Материалы по дисциплине «Микропроцессорные системы»
Специальность «Компьютерные системы и комплексы»
Составитель: Торгашин Р.Г

ГБПОУ ВО "Борисоглебский техникум промышленных и информационных технологий"

2016 год

Оглавление

Представление данных

Для выполнения программ компьютер временно записывает программу и данные в основную память. Это память, которую люди имеют в виду, когда утверждают, что их компьютер имеет, например, 512К памяти. Компьютер имеет также ряд регистров, которые он использует для временных вычислений.

Бит

Минимальной единицей информации в компьютере является бит. Бит может быть выключен, так что его значение есть нуль, или включен, тогда его значение равно единице. Единственный бит не может представить много информации в отличие от группы битов.

Байт

Группа из девяти связанных битов называется байт. Каждый байт содержит восемь бит для хранения данных и один — для проверки четности.

В некоторых случаях, например при передаче информации по электронным сетям, под байтом понимают группу из восьми связанных бит. В этом случае для проверки четности используется восьмой бит или проверка не ведется. Часто проверка четности реализуется на аппаратном уровне, поэтому в программировании под байтом понимают группу из восьми бит.

Восемь бит позволяют представить до 2^8 (256) сочетаний значений 0 и 1(включено/выключено).

Таблица 1. Структура байта

0	0	0	0	0	0	0	0	1
Биты данных								Бит четности

Согласно правилу контроля четности, количество бит, находящихся в единичном состоянии в одном байте должно быть всегда нечетно.

Например

для 00101010 бит четности 0. Полная форма байта 001010100

для 01100000 бит четности 1. Полная форма байта 011000001

Когда инструкция ссылается на определенный байт в памяти, процессор выполняет проверку выполнения правила четности. Если оно не выполняется — система считает что данные повреждены и выводит сообщение о ошибке проверки четности. Это может быть вызвано сбоем оборудования, помехами и т. п.

Обработка бита четности — автоматическая функция аппаратуры. Поэтому в дальнейшем мы не будем ее учитывать. И будем, если не указано иное, считать что в байте восемь бит.

Биты в байте нумеруются справа налево от 0 до 7:

Таблица 2. Нумерация бит в байте

Содержимое битов	0	1	0	0	0	0	0	1
Номера	7	6	5	4	3	2	1	0

Связанные биты

Программа может рассматривать группу из байт как единицу данных. Группа байт, хранящих определенное значение, обычно называется полем или элементом данных (data item). Процессор тоже поддерживает определенные форматы данных:

Таблица 3. Форматы данных используемые процессором

Формат	Длинна (байт)	Длинна (бит)
Слово (word)	2	16
Двойное слово (doubleword)	4	32

Счетверенное слово (quadword)	8	64
Параграф (paragraph)	16	128
Килобайт (kilobyte)	1024 (2^{10})	
Мегабайт (Megabyte)	1048576 (2^{20})	
Гигабайт (Gigabyte)	1073741824 (2^{30})	

Биты в слове обозначаются числами от 0 до 15 справа налево.

Таблица 4. Нумерация бит в слове

Содержимое бита	0	1	0	1	0	0	0	0	0	1	0	0	0	0	1	1
Номер бита	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Старший бит								Младший бит							

Каждый байт в памяти имеет уникальный адрес. Первый байт — адрес 0. Второй байт -1 и т.д.

Элементы двоичной системы

Компьютер может различать только два состояния — 0 и 1. Которые хранятся в битах. Набор бит может представлять любое числовое значение. Значение двоичного числа определяется наличием единичных бит и их положением в последовательности.

Таблица 5. Зависимость значения бита от положения

Значение	1	1	1	1	1	1	1	1
Номер	7	6	5	4	3	2	1	0
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Вес позиции	128	64	32	16	8	4	2	1

Например:
$\mathbf{01000001}_2=0*2^7+1*2^6+0*2^5+0*2^4+0*2^3+0*2^2+0*2^1+1*2^0=65_{10}$
Также, каждому числовому значению ставится в соответствие определенный символ. Для такого сопоставления используются специальные *таблицы кодировок*. Одной из наиболее популярных таблиц является таблица ASCII. В частности, по таблице ASCII код 65 соответствует латинской «A»

Dec	Hex	Char	Dec	Hex	Char
32	20	Space	64	40	@
33	21	!	65	41	A
34	22	"	66	42	B
35	23	#	67	43	C
36	24	$	68	44	D
37	25	%	69	45	E
38	26	&	70	46	F

Рисунок 1. *Часть таблицы ASCII*

Двоичное число не обязательно должно быть восьми разрядным. 32 разрядный процессор эффективно работает с разрядностью 32. А 64 разрядный с разрядностью 64.

Двоичное число со знаком (применяемое в арифметических операциях) считается положительным если его левый (старший) бит равен 0 и отрицательным если 1. Однако

просто заменить старший бит на 1 недостаточно. Ведь 11000001=193 а не -65. Необходимо записать число в комплементарной (дополняющей) форме. Для этого нужно инвертировать значения битов исходного числа и добавить единицу.
Например:

Исходное число (+65)		01000001
Инвертируем биты		10111110
Добавляем 1	+1	
Результат (-65)		10111111

Двоичное число с знаком отрицательно если его старший бит равен 1. Процессор различает число со знаком и беззнаковые числа и обрабатывает их по разному.
Вычислим десятичное значение числа $10111111_2=191_{10}$. Что определенно не равно -65. Для двоичных чисел с знаком перед конвертированием нужно найти двоичное дополнение.

Исходное число (-65)		10111111
Инвертируем биты		10000000
Добавляем 1	+1	
Результат (+65)		01000001

При вычитании чисел вычитаемое нужно преобразовать в двоичное дополнение и сложить с уменьшаемым.

Шестнадцатеричные числа

Так как четыре байта включают в себя 32 бита, то специалисты разработали "стенографический" метод представления двоичных данных. По этому методу каждый байт делится пополам и каждые полбайта выражаются соответствующим значением.

Рассмотрим следующие четыре байта:
Двоичное: 0101 1001 0011 0101 1011 1001 1100 1110.
Десятичное: 5 9 3 5 11 9 12 14
Так как здесь для некоторых чисел требуется две цифры, расширим систему счисления так, чтобы 10=A, 11=B, 12=C, 13=D, 14=E, 15=F. таким образом получим более сокращенную форму, которая представляет содержимое вышеуказанных байт: 59 35 B9 CE. Такая система счисления включает "цифры" от 0 до F, и так как таких цифр 16, она называется шестнадцатеричным представлением.

$0_{hex} = 0_{dec} = 0_{oct}$	0	0	0	0
$1_{hex} = 1_{dec} = 1_{oct}$	0	0	0	1
$2_{hex} = 2_{dec} = 2_{oct}$	0	0	1	0
$3_{hex} = 3_{dec} = 3_{oct}$	0	0	1	1
$4_{hex} = 4_{dec} = 4_{oct}$	0	1	0	0
$5_{hex} = 5_{dec} = 5_{oct}$	0	1	0	1
$6_{hex} = 6_{dec} = 6_{oct}$	0	1	1	0
$7_{hex} = 7_{dec} = 7_{oct}$	0	1	1	1
$8_{hex} = 8_{dec} = 10_{oct}$	1	0	0	0
$9_{hex} = 9_{dec} = 11_{oct}$	1	0	0	1
$A_{hex} = 10_{dec} = 12_{oct}$	1	0	1	0
$B_{hex} = 11_{dec} = 13_{oct}$	1	0	1	1
$C_{hex} = 12_{dec} = 14_{oct}$	1	1	0	0
$D_{hex} = 13_{dec} = 15_{oct}$	1	1	0	1
$E_{hex} = 14_{dec} = 16_{oct}$	1	1	1	0
$F_{hex} = 15_{dec} = 17_{oct}$	1	1	1	1

Рисунок 2. *Таблица перевода шестнадцатеричных (hex), десятичных, восьмеричных (oct) и двоичных чисел*

Шестнадцатеричный формат нашел большое применение. В листингах ассемблированных программ в шестнадцатеричном формате показаны все адреса, машинные коды команд и содержимое констант. Также для отладки при использовании программы DOS DEBUG адреса и содержимое байтов выдается в шестнадцатеричном формате.

Для низкоуровневой работы с данными используются шестнадцатеричные редакторы (Hex-редакторы). Они могут быть как отдельными самостоятельными приложениями, так и компонентами другого, более сложного приложения[1], такого как дизассемблер, отладчик, интегрированная среда разработки, средство восстановления структуры жестких дисков[2] и т.п.[3]

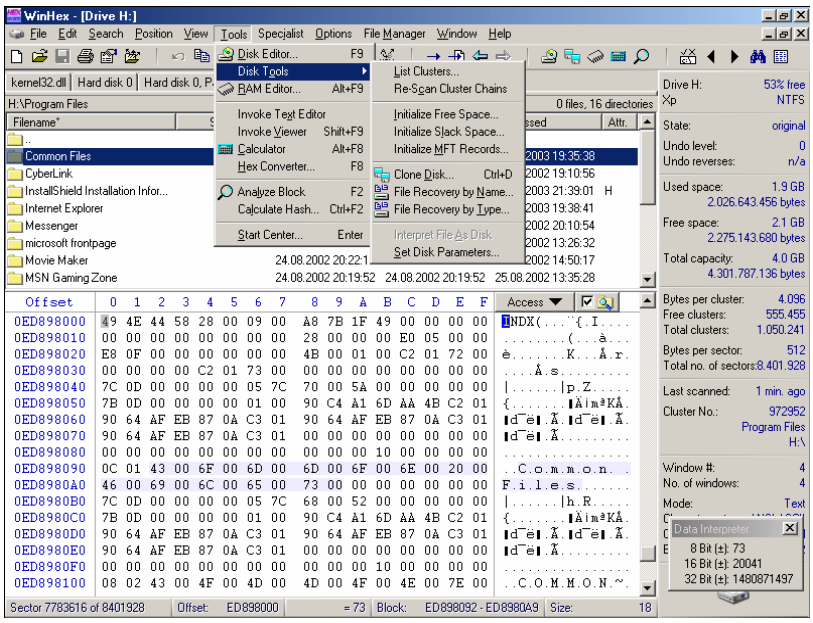

Рисунок 3: Окно редактора WinHex

1WinHEX http://www.winhex.com/winhex/
2DMDE http://dmde.ru/
3Сравнение HEX редакторов http://en.wikipedia.org/wiki/Comparison_of_hex_editors

Для индикации шест. числа в ассемблерной программе непосредственно после числа ставится символ "H", например, 25H (десятичное значение 37). Шест. число всегда начинается с десятичной цифры 0-9, таким образом, B8H записывается как 0B8H.

Адресация данных в памяти

Выделяют два типа внутренней памяти ПК — постоянная память (ПЗУ) и оперативная память (ОЗУ). Последняя также называется памятью с произвольным доступом. Байты в памяти нумеруются последовательно, начиная с 0, и каждый байт имеет уникальный адрес.

Рассмотрим структуру внутренней памяти ПК с процессором 8086. Из первого мегабайта памяти — 640 килобайт отведено на основную оперативную память, большая часть которой доступна для непосредственного использования. Таблица векторов прерываний размещена в нижней области памяти. Выше нее размещена видеопамять.

Таблица 6. Структура адресного пространства 8086

Начало (десятичное)	**Адрес (шестнадцатеричный)**	**Объем и назначение**
960 К	F0000	64К основное ПЗУ
768 К	C0000	192К расширенное ПЗУ
640 К	A0000	128К видеопамять (ОЗУ)
0	00000	640К (ОЗУ) Область BIOS. Таблица векторов прерываний

В зависимости от модели процессор может обращаться к одному или более байтам памяти. Например:

1315_{10}=0529H, занимает два байта (одно слово) в памяти. Слово состоит из старшего (05) и младшего (29) байта.

Процессор хранит данные в инверсном порядке. То-есть младший байт в ячейке с меньшим адресом а старший- в ячейке с большим адресом.

Тогда он передаст 0529H из регистра в память по адресам 04A26H и 04A27H следующим образом:

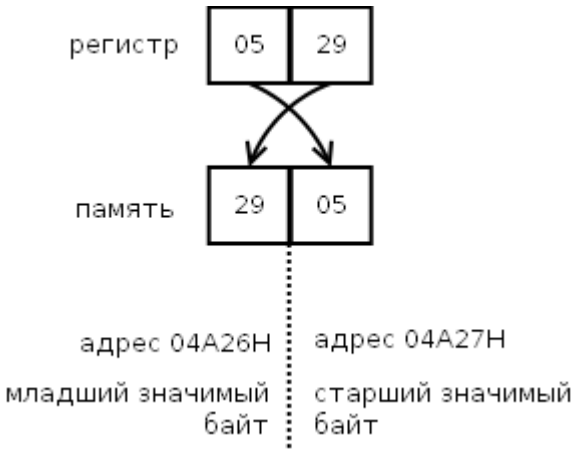

Рисунок 4 *Передача слова из регистра в память*

Процессор подразумевает, что данные в памяти хранятся в инверсном порядке, и обрабатывает их соответственно. Когда он извлекает данные из памяти, процессор переставляет байты.

Адресация операндов

Большая часть команд процессора работает с кодами данных (операндами). Одни команды требуют входных операндов (одного или двух), другие выдают выходные операнды (чаще один операнд). Входные операнды называются еще операндами-источниками, а

выходные называются операндами-приемниками. Все эти коды операндов (входные и выходные) должны где-то располагаться.

- Они могут находиться во внутренних регистрах процессора (наиболее удобный и быстрый вариант).
- Они могут располагаться в системной памяти (самый распространенный вариант).
- Они могут находиться в устройствах ввода/вывода (наиболее редкий случай).

Определение места положения операндов производится кодом команды. Причем существуют разные методы, с помощью которых код команды может определить, откуда брать входной операнд и куда помещать выходной операнд. Эти методы называются методами адресации . Эффективность выбранных методов адресации во многом определяет эффективность работы всего процессора в целом.

Есть две схемы адресации:

Абсолютный адрес. - например 04A26, представляющий собой 20 разрядное двоичное число, прямо указывающее на ячейку памяти с операндом.

Адрес в системе сегмент-смещение, состоящий из начального адреса сегмента и значения смещения.

Методы адресации

Количество методов адресации в различных процессорах может быть от 4 до 16.

Непосредственная адресация предполагает, что операнд (входной) находится в памяти непосредственно за кодом команды. Операнд обычно представляет собой константу, которую надо куда-то переслать, к чему-то прибавить и т.д. Например, команда может состоять в том, чтобы прибавить число 6 к содержимому какого-то внутреннего регистра процессора. Это число 6 будет располагаться в памяти, внутри программы в адресе, следующем за кодом данной команды сложения.

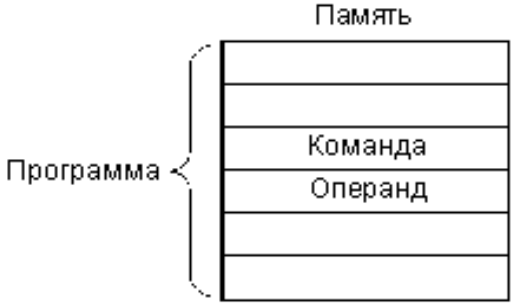

Рисунок 5 *Непосредственная адресация*

П**рямая (она же абсолютная) адресация** предполагает, что операнд (входной или выходной) находится в памяти по адресу, код которого находится внутри программы сразу же за кодом команды. Например, команда может состоять в том, чтобы очистить (сделать нулевым) содержимое ячейки памяти с адресом 1000000. Код этого адреса 1000000 будет располагаться в памяти, внутри программы в следующем адресе за кодом данной команды очистки.

Рисунок 6 *Прямая адресация*

Регистровая адресация предполагает, что операнд (входной или выходной) находится во внутреннем регистре процессора. Например, команда может состоять в том, чтобы переслать число из нулевого регистра в первый. Номера обоих регистров (0 и 1) будут определяться кодом команды пересылки.

***Рисунок** 7 Регистровая адресация*

Косвенно-регистровая (она же косвенная) адресация предполагает, что во внутреннем регистре процессора находится не сам операнд, а его адрес в памяти. Например, команда может состоять в том, чтобы очистить ячейку памяти с адресом, находящимся в нулевом регистре. Номер этого регистра (0) будет определяться кодом команды очистки.

Автоинкрементная адресация очень близка к косвенной адресации, но отличается от нее тем, что после выполнения команды содержимое используемого регистра увеличивается на единицу или на два. Этот метод адресации очень удобен, например, при последовательной обработке кодов из массива данных, находящегося в памяти. После обработки какого-то кода адрес в регистре будет указывать уже на следующий код из массива. При использовании косвенной адресации в данном случае пришлось бы увеличивать содержимое этого регистра отдельной командой.
Автодекрементная адресация работает похоже на автоинкрементную, но только содержимое выбранного регистра уменьшается на единицу или на два перед выполнением команды. Эта адресация также удобна при обработке массивов данных. Совместное использование автоинкрементной и автодекрементной адресаций позволяет организовать память стекового типа.

Из других распространенных методов адресации можно упомянуть об индексных методах, которые предполагают для вычисления адреса операнда прибавление к содержимому регистра заданной константы (индекса). Код этой константы располагается в памяти непосредственно за кодом команды.

Отметим, что выбор того или иного метода адресации в значительной степени определяет время выполнения команды. Самая быстрая адресация — это регистровая, так как она не требует дополнительных циклов обмена по магистрали. Если же адресация требует обращения к памяти, то время выполнения команды будет увеличиваться за счет длительности необходимых циклов обращения к памяти. Понятно, что чем больше внутренних регистров у процессора, тем чаще и свободнее можно применять регистровую адресацию, и тем быстрее будет работать система в целом.

Адресация байт и слов

Многие процессоры, имеющие разрядность 16 или 32, способны адресовать не только целое слово в памяти (16-разрядное или 32-разрядное), но и отдельные байты. Каждому байту в каждом слове при этом отводится свой адрес.
Так, в случае 16-разрядных процессоров все слова в памяти (16-разрядные) имеют четные адреса. А байты, входящие в эти слова, могут иметь как четные адреса, так и нечетные. Например, пусть 16-разрядная ячейка памяти имеет адрес 23420, и в ней хранится код 2A5E (рис 8).

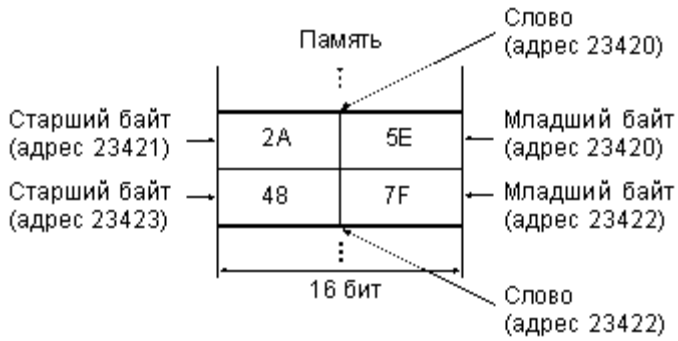

Рисунок 8 *Адресация байтов и слов*

При обращении к целому слову (с содержимым 2A5E) процессор выставляет адрес 23420. При обращении к младшему байту этой ячейки (с содержимым 5E) процессор выставляет тот же самый адрес 23420, но использует команду, адресующую байт, а не слово. При обращении к старшему байту этой же ячейки (с содержимым 2A) процессор выставляет адрес 23421 и использует команду, адресующую байт. Следующая по порядку 16-разрядная ячейка памяти с содержимым 487F будет иметь адрес 23422, то есть опять же четный. Ее байты будут иметь адреса 23422 и 23423.

Для различия байтовых и словных циклов обмена на магистрали в шине управления предусматривается специальный сигнал байтового обмена. Для работы с байтами в систему команд процессора вводятся специальные команды или предусматриваются методы байтовой адресации.

Сегментирование памяти

Сегментировании памяти применяется в некоторых процессорах, например в процессорах IBM PC-совместимых персональных компьютеров.

В процессоре Intel 8086 сегментирование памяти организовано следующим образом.

Вся память системы представляется не в виде непрерывного пространства, а в виде нескольких кусков — сегментов заданного размера (по 64 Кбайта), положение которых в пространстве памяти можно изменять программным путем.

Для хранения кодов адресов памяти используются не отдельные регистры, а пары регистров:

- сегментный регистр определяет адрес начала сегмента (то есть положение сегмента в памяти);
- регистр указателя (регистр смещения) определяет положение рабочего адреса внутри сегмента.

При этом физический 20-разрядный адрес памяти, выставляемый на внешнюю шину адреса, образуется так, как показано на рис. 9, то есть путем сложения смещения и адреса сегмента со сдвигом на 4 бита. Положение этого адреса в памяти показано на рис. 9

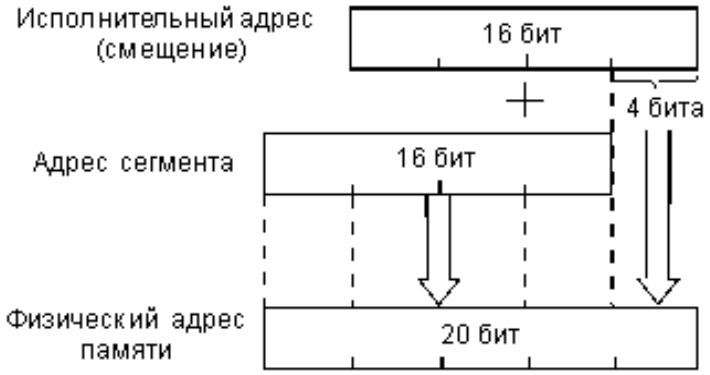

Рисунок 9 *Формирование физического адреса памяти из адреса сегмента и смещения*

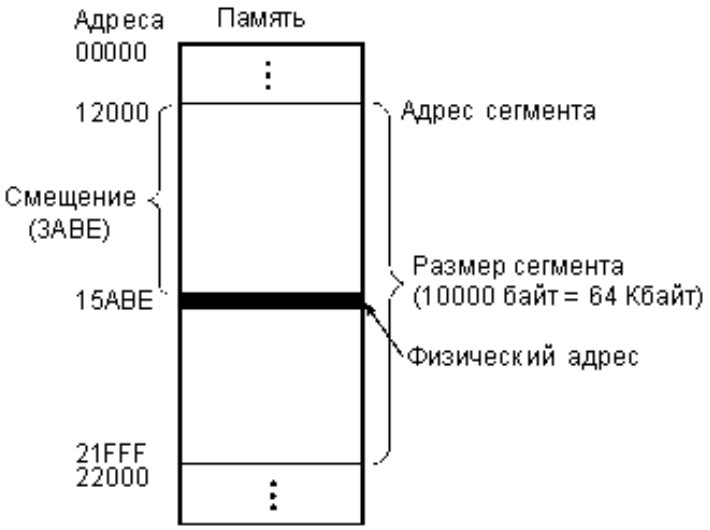

Рисунок 10 *Физический адрес в сегменте*

Сегмент может начинаться только на 16-байтной границе памяти (так как адрес начала сегмента, по сути, имеет четыре младших нулевых разряда, как видно из рис. 10), то есть с адреса, кратного 16. Эти допустимые границы сегментов называются границами параграфов.

Отметим, что введение сегментирования, прежде всего, связано с тем, что внутренние регистры процессора 16-разрядные, а физический адрес памяти 20-разрядный (16-разрядный адрес позволяет использовать память только в 64 Кбайт, что явно недостаточно). В появившемся в то же время процессоре MC68000 фирмы Motorola внутренние регистры 32-разрядные, поэтому там проблемы сегментирования памяти не возникает.

Применяются и более сложные методы сегментирования памяти. Например, в процессоре Intel 80286 в так называемом защищенном режиме адрес памяти вычисляется в соответствии с рис. 11.

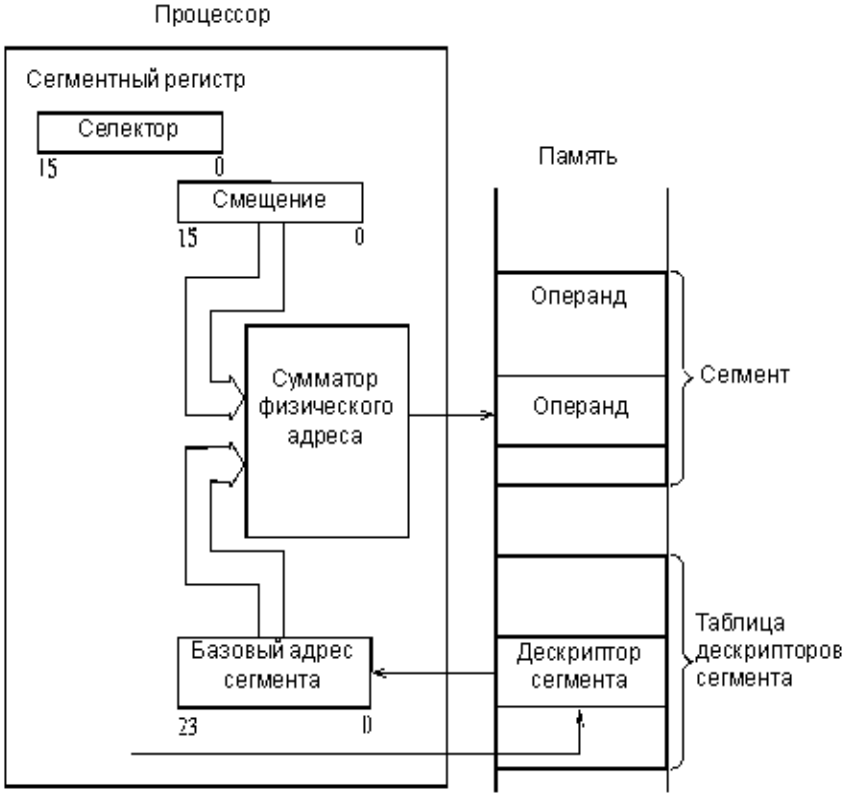

Рисунок 11 *Адресация памяти в защищенном режиме процессора Intel 80286*

В сегментном регистре в данном случае хранится не базовый (начальный) адрес сегментов, а коды селекторов, определяющие адреса в памяти, по которым хранятся дескрипторы (то есть описатели) сегментов. Область памяти с дескрипторами называется таблицей дескрипторов. Каждый дескриптор сегмента содержит базовый адрес сегмента, размер сегмента (от 1 до 64 Кбайт) и его атрибуты. Базовый адрес сегмента имеет разрядность 24 бит, что обеспечивает адресацию 16 Мбайт физической памяти.

Таким образом, на сумматор, вычисляющий физический адрес памяти, подается не содержимое сегментного регистра, как в предыдущем случае, а базовый адрес сегмента из таблицы дескрипторов.

Еще более сложный метод адресации памяти с сегментированием использован в процессоре Intel 80386 и в более поздних моделях процессоров фирмы Intel. Этот метод иллюстрируется рис. 12.

Адрес памяти (физический адрес) вычисляется в три этапа. Сначала вычисляется так называемый эффективный адрес (32-разрядный) путем суммирования трех компонентов: базы, индекса и смещения (Base, Index, Displacement), причем возможно умножение индекса на масштаб (Scale). Эти компоненты имеют следующий смысл:

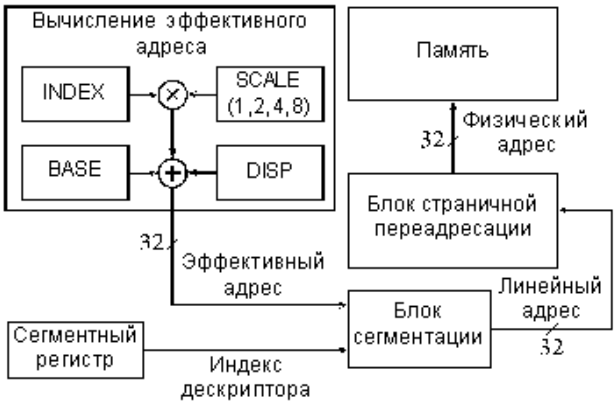

Рисунок 12. *Формирование физического адреса памяти процессора 80386 в защищенном режиме.*

- смещение — это 8-, 16- или 32-разрядное число, включенное в команду.
- база — это содержимое базового регистра процессора. Обычно оно используется для указания на начало некоторого массива.
- индекс — это содержимое индексного регистра процессора. Обычно оно используется для выбора одного из элементов массива.
- масштаб — это множитель (он может быть равен 1, 2, 4 или 8), указанный в коде команды, на который перед суммированием с другими компонентами умножается индекс. Он используется для указания размера элемента массива.

Затем специальный блок сегментации вычисляет 32-разрядный линейный адрес, который представляет собой сумму базового адреса сегмента из сегментного регистра с эффективным адресом. Наконец, физический 32-битный адрес памяти образуется путем преобразования линейного адреса блоком страничной переадресации, который осуществляет перевод линейного адреса в физический страницами по 4 Кбайта.

В любом случае сегментирование позволяет выделить в памяти один или несколько сегментов для данных и один или несколько сегментов для программ. Переход от одного сегмента к другому сводится всего лишь к изменению содержимого сегментного регистра. Иногда это бывает очень удобно. Но для программиста работать с сегментированной памятью обычно сложнее, чем с непрерывной, несегментированной памятью, так как приходится следить за границами сегментов, за их описанием, переключением и т. д.

Регистры процессора

Внутренние регистры процессора представляют собой сверхоперативную память небольшого размера, которая предназначена для временного хранения служебной информации или данных. Количество регистров в разных процессорах может быть от 6—8 до нескольких десятков. Регистры могут быть универсальными и специализированными.

Специализированные регистры, которые присутствуют в большинстве процессоров, — это регистр- счетчик команд, регистр состояния (PSW), регистр указателя стека. Остальные регистры процессора могут быть как универсальными, так и

специализированными.

Например, в 16-разрядном процессоре Т-11 фирмы DEC было 8 регистров общего назначения (РОН) и один регистр состояния. Все регистры имели по 16 разрядов. Из регистров общего назначения один отводился под счетчик команд, другой — под указатель стека. Все остальные регистры общего назначения полностью взаимозаменяемы, то есть имеют универсальное назначение, могут хранить как данные, так и адреса (указатели), индексы и т.д. Максимально допустимый объем памяти для данного процессора составлял 64 Кбайт (адрес памяти 16-разрядный).

В 16-разрядном процессоре MC68000 фирмы Motorola было 19 регистров: 16-разрядный регистр состояния, 32-разрядный регистр счетчика команд, 9 регистров адреса (32-разрядных) и 8 регистров данных (32-разрядных). Два регистра адреса отведены под указатели стека. Максимально допустимый объем адресуемой памяти — 16 Мбайт (внешняя шина адреса 24-разрядная). Все 8 регистров данных взаимозаменяемы. 7 регистров адреса – тоже взаимозаменяемы.

В 16-разрядном процессоре Intel 8086, который стал базовым в линии процессоров, используемых в персональных компьютерах, реализован принципиально другой подход. Каждый регистр этого процессора имеет свое особое назначение, и заменять друг друга регистры могут только частично или же не могут вообще. Остановимся на особенностях этого процессора подробнее.

Регистры процессора 8086

Процессор 8086 имеет 14 регистров разрядностью по 16 бит. Из них четыре регистра (AX, BX, CX, DX) — это регистры данных, каждый из которых помимо хранения операндов и результатов операций имеет еще и свое специфическое назначение:

- регистр AX — умножение, деление, обмен с устройствами ввода/вывода (команды ввода и вывода);
- регистр BX — базовый регистр в вычислениях адреса;
- регистр CX — счетчик циклов;
- регистр DX — определение адреса ввода/вывода.

Для регистров данных существует возможность раздельного использования обоих байтов (например, для регистра AX они имеют обозначения AL – младший байт и AH — старший байт).

Следующие четыре внутренних регистра процессора — это сегментные регистры, каждый из которых определяет положение одного из рабочих сегментов (рис. 13):

- регистр CS (Code Segment) соответствует сегменту команд, исполняемых в данный момент;
- регистр DS (Data Segment) соответствует сегменту данных, с которыми работает процессор;
- регистр ES (Extra Segment) соответствует дополнительному сегменту данных;
- регистр SS (Stack Segment) соответствует сегменту стека.

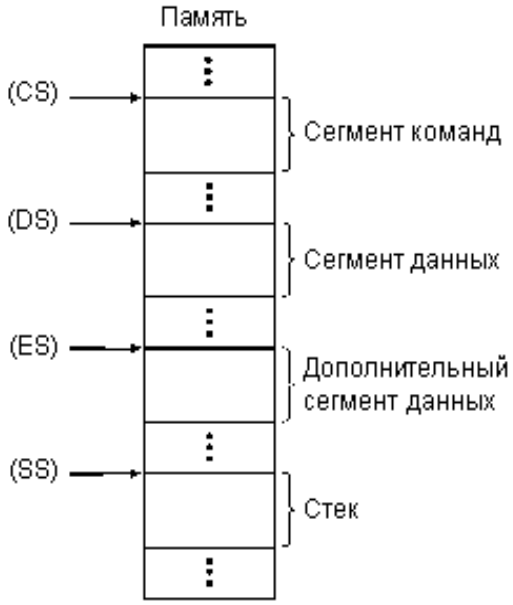

Рисунок 13 *Сегментные регистры*

В принципе, все эти сегменты могут и перекрываться для оптимального использования пространства памяти. Например, если программа занимает только часть сегмента, то сегмент данных может начинаться сразу после завершения работы программы (с точностью 16 байт), а не после окончания всего сегмента программы.
Следующие пять регистров процессора
SP — Stack Pointer,
BP — Base Pointer,
SI — Source Index,
DI — Destination Index,
IP —Instruction Pointer
служат указателями (то есть определяют смещение в пределах сегмента). Например, счетчик команд процессора образуется парой регистров CS и IP, а указатель стека — парой регистров SP и SS. Регистры SI, DI используются в строковых операциях, то есть при последовательной обработке нескольких ячеек памяти одной командой.

Последний регистр FLAGS — это регистр состояния процессора (PSW). Из его 16 разрядов используются только девять (рис. 14): CF (Carry Flag) — флаг переноса при арифметических операциях, PF (Parity Flag) — флаг четности результата, AF (Auxiliary Flag) — флаг дополнительного переноса, ZF (Zero Flag) — флаг нулевого результата, SF(Sign Flag) — флаг знака (совпадает со старшим битом результата), TF (Trap Flag) — флаг пошагового режима (используется при отладке), IF (Interrupt-enable Flag) — флаг разрешения аппаратных прерываний, DF (Direction Flag) — флаг направления при строковых операциях, OF (Overflow Flag) — флаг переполнения.

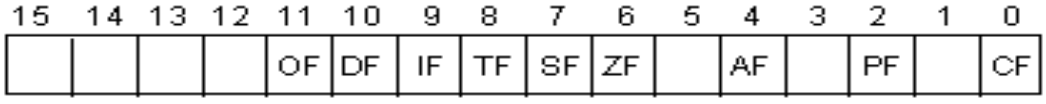

Рисунок 14 *Регистр состояния*

Биты регистра состояния устанавливаются или очищаются в зависимости от результата исполнения предыдущей команды и используются некоторыми командами процессора. Биты регистра состояния могут также устанавливаться и очищаться специальными командами процессора (о системе команд процессора будет рассказано в следующем разделе).

Во многих процессорах выделяется специальный регистр,

называемый аккумулятором (то есть накопителем). При этом, как правило, только этот регистр-аккумулятор может участвовать во всех операциях, только через него может производиться взаимодействие с устройствами ввода/вывода. Иногда в него же помещается результат любой выполненной команды (в этом случае говорят даже об "аккумуляторной" архитектуре процессора).

Например, в процессоре 8086 регистр данных AX можно считать своеобразным аккумулятором, так как именно он обязательно участвует в командах умножения и деления, а также только через него можно пересылать данные в устройство ввода/вывода и из устройства ввода/вывода.

Выделение специального регистра-аккумулятора упрощает структуру процессора и ускоряет пересылки кодов внутри процессора, но в некоторых случаях замедляет работу системы в целом, так как весь поток информации должен пройти через один регистр-аккумулятор. В случае, когда несколько регистров процессора полностью взаимозаменяемы, таких проблем не возникает.

Регистры процессора 80386

Начиная с модели 80386 процессоры Intel предоставляют 16 основных регистров для пользовательских программ и ещё 11 регистров для работы с мультимедийными приложениями (MMX) и числами с плавающей точкой (FPU/NPX). Все команды так или иначе изменяют содержимое регистров. Как уже говорилось, обращаться к регистрам быстрее и удобнее, чем к памяти. Поэтому при программировании на языке Ассемблера регистры используются очень широко.

Таблица 7. Основные регистры процессора 80386.

Название	Разрядность	Основное назначение
EAX	32	Аккумулятор
EBX	32	База
ECX	32	Счётчик
EDX	32	Регистр данных
EBP	32	Указатель базы
ESP	32	Указатель стека
ESI	32	Индекс источника
EDI	32	Индекс приёмника
EFLAGS	32	Регистр флагов
EIP	32	Указатель инструкции (команды)
CS	16	Сегментный регистр
DS	16	Сегментный регистр
ES	16	Сегментный регистр
FS	16	Сегментный регистр
GS	16	Сегментный регистр
SS	16	Сегментный регистр

Регистры EAX, EBX, ECX, EDX – это регистры общего назначения. Они имеют определённое назначение (так уж сложилось исторически), однако в них можно хранить любую информацию.

Регистры EBP, ESP, ESI, EDI – это также регистры общего назначения. Они имеют уже более конкретное назначение. В них также можно хранить пользовательские данные, но делать это нужно уже более осторожно, чтобы не получить «неожиданный» результат.

Ранее процессоры были 16-разрядными, и, соответственно, все их регистры были также 16-разрядными. Для совместимости со старыми программами, а также для удобства программирования некоторые регистры разделены на 2 или 4 «маленьких» регистра, у каждого из которых есть свои имена. Пример такого регистра.

Рисунок 15 *Делимый регистр EAX*

Из этого следует, что можно написать в программе, например, такие команды:

MOV AX, 1

MOV EAX, 1

Обе команды поместят в регистр AX число 1. Разница будет заключаться только в том, что вторая команда обнулит старшие разряды регистра EAX, то есть после выполнения второй команды в регистре EAX будет число 1. А первая команда оставит в старших разрядах регистра EAX старые данные. И если там были данные, отличные от нуля, то после выполнения первой команды в регистре EAX будет какое-то число, но не 1. А вот в регистре AX будет число 1.

Ниже приведён список регистров общего назначения, которые можно поделить описанным выше способом и при этом к «половинкам» и «четвертинкам» этих регистров можно обращаться в программе как к отдельному регистру.

Таблица 8. «Делимые» регистры.

Регистр	Старшие разряды	Имена 16-ти и 8-ми битных регистров	
	31…16	15…8	7…0
EAX	...	AX	
		AH	AL
EBX	...	BX	
		BH	BL
ECX	...	CX	
		CH	CL
EDX	...	DX	
		DH	DL
ESI	...	SI	
EDI	...	DI	
EBP	...	BP	
ESP	...	SP	
EIP	...	IP	

Вопросы для контроля

Базовый уровень

Что такое бит?
Что такое байт?
Опишите структуру байта с учетом бита четности.
Сформулируйте правило контроля четности
Как нумеруются биты в слове
Как информация хранящаяся в битах может быть преобразована в понятный человеку формат?
Что такое шестнадцатеричные числа?
Как слово передается из регистра процессора в память?
Что такое абсолютный адрес операнда?
Что такое адрес в системе сегмент-смещение?
Опишите метод непосредственной адресации
Опишите метод прямой адресации
Опишите метод регистровой адресации
Опишите метод косвенно-регистровой адресации
Опишите метод автоинкрементной адресации
Опишите метод автодекрементной адресации
Для чего применяют сегментирование памяти?
Как работает сегментирования память процессора 8086?
Какой адрес называется логическим?
Что такое линейный адрес?
Что такое физический адрес?
Для чего используются дескрипторы сегмента?
Как выполняется адресация байт в слове?
Что такое «внутренние регистры процессора»?
Что такое «делимый» регистр?

Расширенный уровень

Перечислите и охарактеризуйте форматы данных используемые процессором
Как кодируются двоичные числа с знаком?
Как работает сегментирования память процессора 80286?
В чем разница между GDT и LDT таблицами?
Какие три блока содержит селектор сегмента?
Для чего служат регистры AX, BX, DX, CX?
Для чего служат регистры CS, DS, ES, SS?
Для чего служат регистры SP, BP, SI, DI, IP?
Для сего служит регистр FLAGS?
Каковы роли битов в регистре FLAGS?

Источники

Основы микропроцессорной техники. Новиков Юрий Витальевич, Скоробогатов Петр Константинович — Url: http://www.intuit.ru/studies/courses/3/3/info
Ассемблер. Язык и программирование для IBM PC. Питер Абель.