Раздел 1.
Введение в архитектуру ЭВМ.

Форматы команд и способы адресации ЭВМ .

Способы адресации ЭВМ

Понятия:

1. **Исполнительный адрес операнда –** двоичный код номера ячейки памяти, служащей источником или приемником операнда. Этот код подается на адресные входы ОП, и по нему происходит фактическое обращение к указанной ячейке.
2. **Адресный код команды –** двоичный код в адресном поле команды, из которого необходимо сформировать исполнительный адрес операнда.
3. **Способ адресации –** это метод формирования исполнительного адреса по адресному коду команды**.**

Отсутствие адресной арифметики в КОП **– неявная адресация.**

Единицы информации:

1. **Байт** – минимально адресуемая часть памяти – 8 бит.
2. **Слово** – совокупность нескольких смежных байтов, общее число битов в которых равно разрядности ЭВМ.
3. **Поле** – непрерывная по возрастающим значениям адресов последовательность байтов в ОП. За адрес поля принимается адрес крайнего левого байта поля. Использование поля той или иной длины определяется типом команды, адресующейся к этому полю.
4. **Параграф** – совокупность 16 смежных байтов ОП. Адрес параграфа должен быть кратен 16. Только для x86-ой архитектуры.
5. **Страница –** некоторая совокупность байтов. Величина ее зависит от класса ЭВМ.
6. **Сегмент** – это область ОП, начинающаяся с границы параграфа и имеющая размер до 64Кбайт. Для архитектуры x86

Способы адресации:

1. ***Непосредственная*** – операнд располагается непосредственно в адресном поле команды; используется при выполнении арифметических операций, операций сравнения и для загрузки команд в регистры;
2. ***Прямая*** – в адресном поле команды указывается адрес операнда в памяти (либо номер регистра, содержащего операнд).
3. ***Косвенная*** – код команды указывает адрес ячейки памяти, в которой находится не сам операнд, а его адрес, называемый ***указателем***; для выборки операнда необходимо двукратное обращение к памяти: 1. извлечь адрес операнда; 2. извлечь сам операнд;
4. ***Индексная*** – к формируемому исполнительному адресу прибавляется значение специального регистра – **индекса**.

Способы формирования адресов ячеек памяти можно разделить на:

- **абсолютные** – двоичный код адреса ячейки памяти может быть целиком извлечен либо из адресного поля команды, либо из какой-нибудь другой ячейки в случае косвенной адресации;
- **относительные** – двоичный код адресной ячейки памяти образуется из нескольких составляющих:

Б – код базы,

И – код индекса,

С – код смещения..

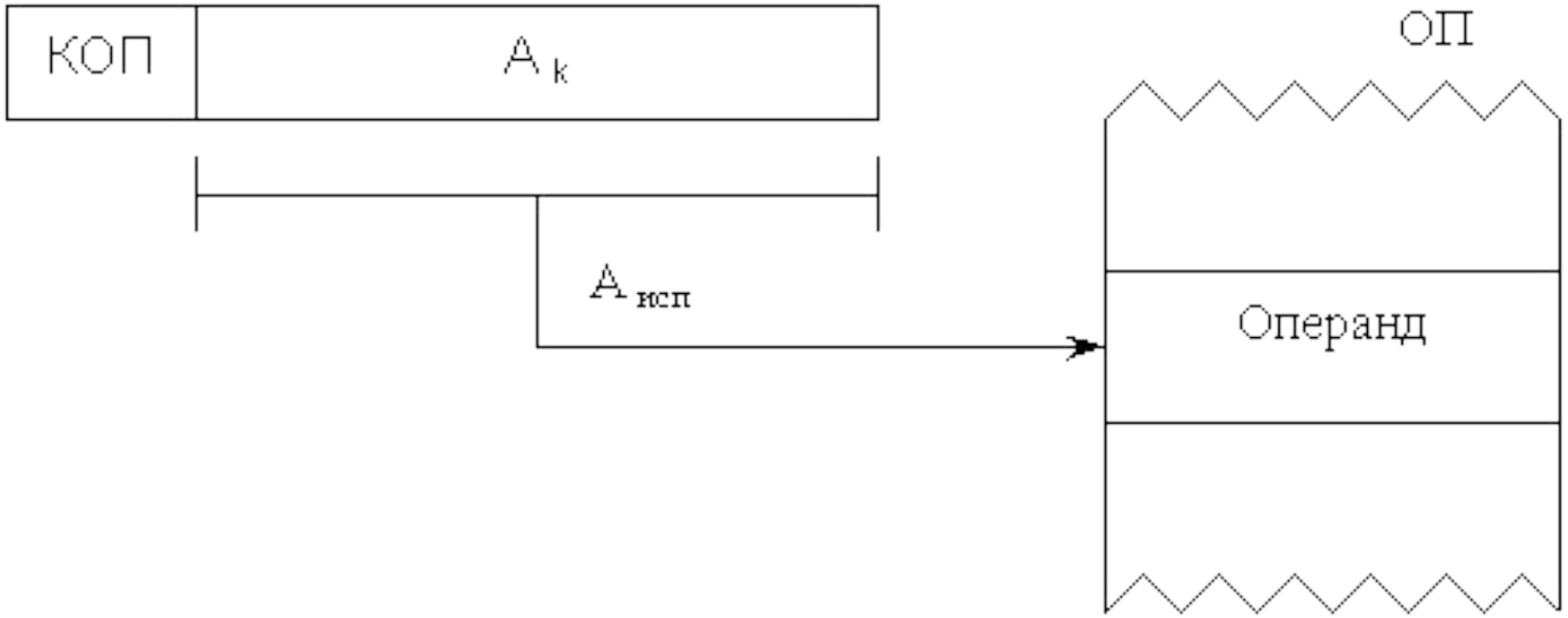
При относительной адресации применяется способ вычисления адреса путем суммирования кодов, составляющих адрес.

$A = Б + И + С$

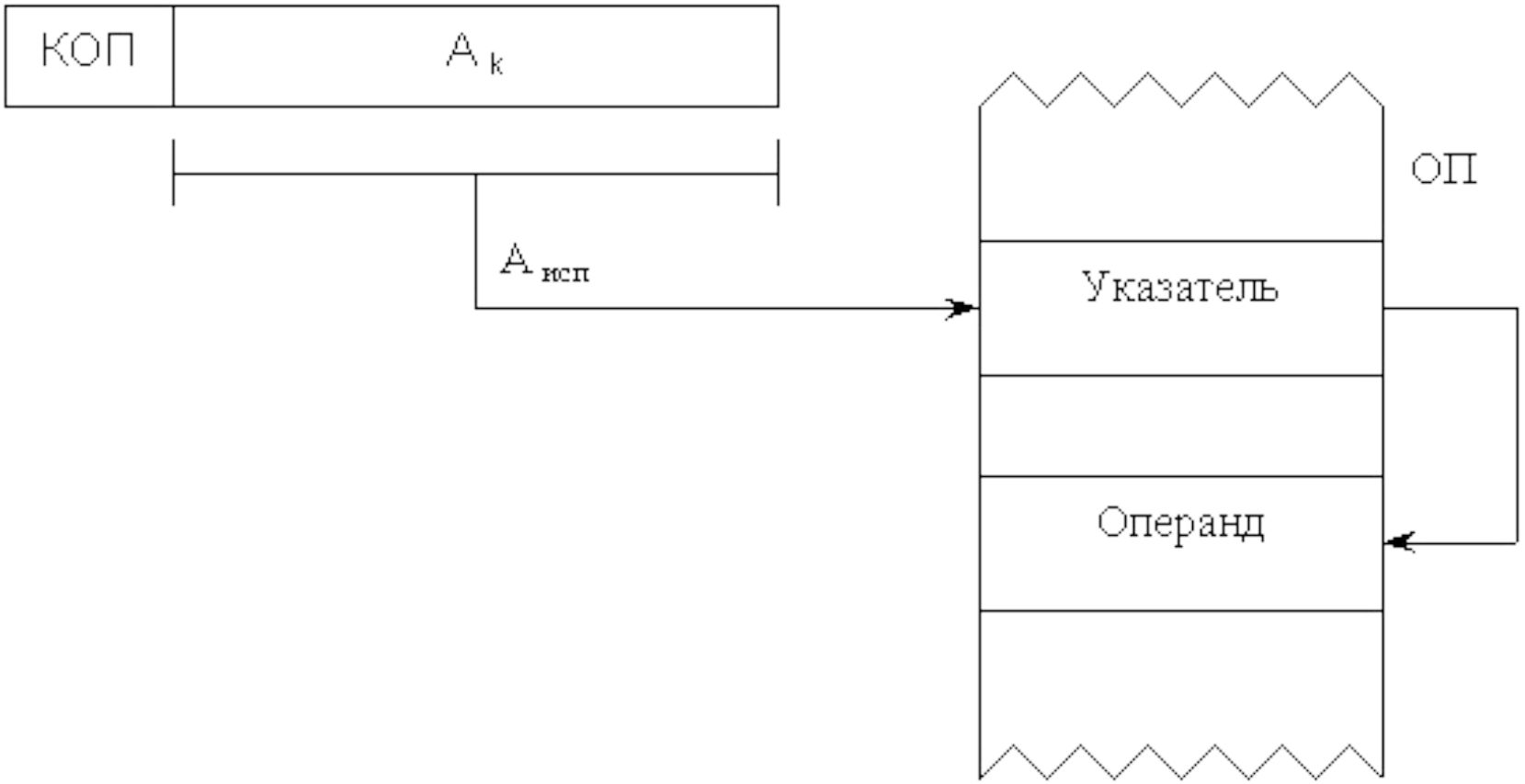
$A = Б + С$

$A = И + С$

Прямая адресация.

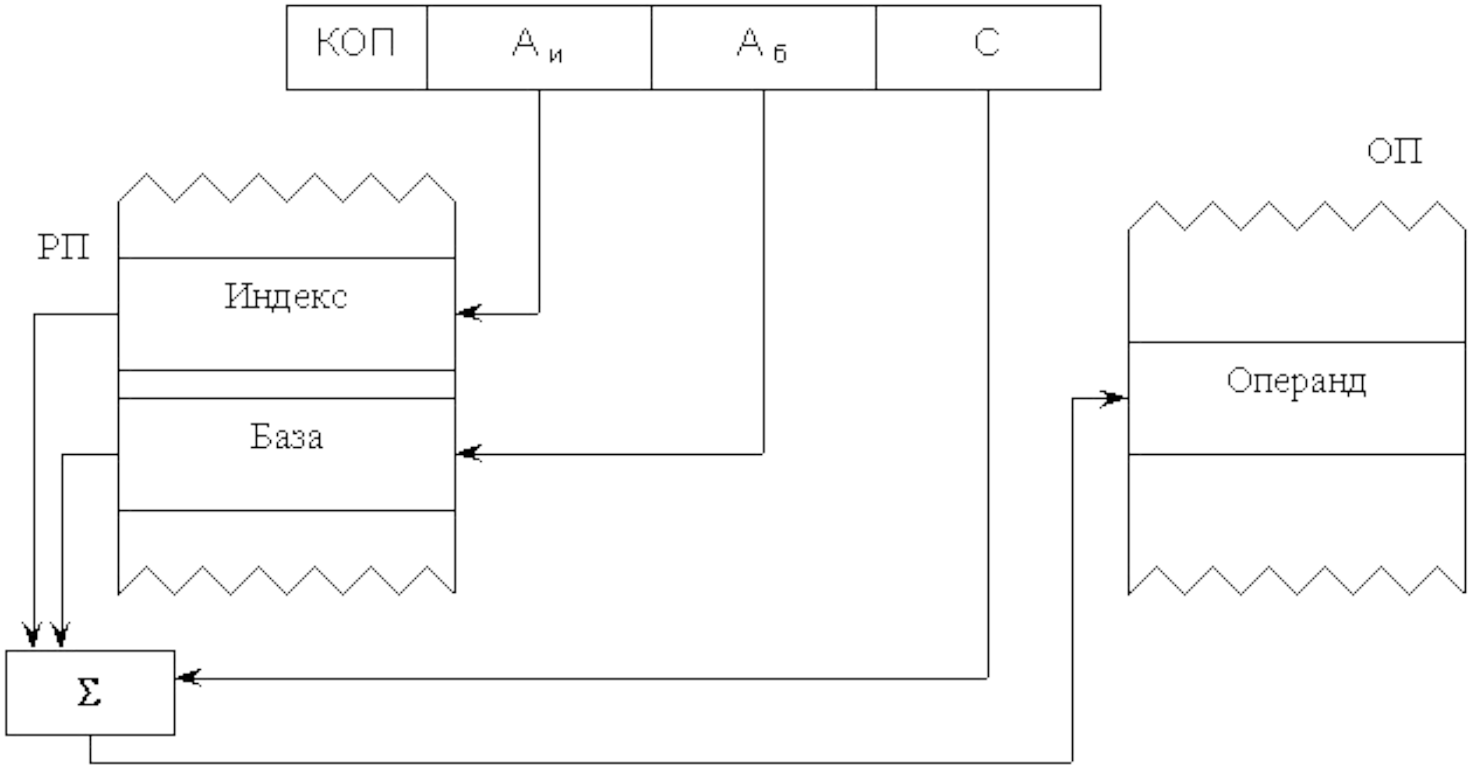


Косвенная адресация.



При косвенной адресации код команды указывает адрес ячейки памяти, в которой находится не сам операнд, а его адрес, называемый ***указателем***.

Индексная адресация.



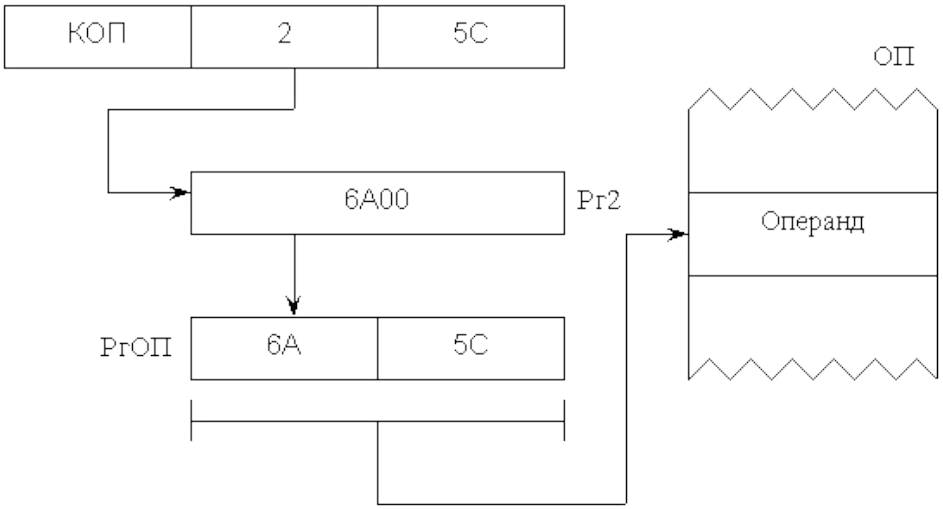
Используется для работы программ с массивами, требующими однотипных операций над элементами массива.

Адрес *i*-того операнда в массиве определяется как сумма начального адреса массива операнда, задаваемого **смещением *S***, и **индекса *I*** , записанного в одном из регистров регистровой памяти, называемым ***индексным регистром***.

Адрес индексного регистра задается в команде полем адреса индекса A_u.

В каждом *i*-том цикле содержимое индексного регистра изменяется на постоянную величину, как правило, это 1.

В некоторых моделях ЭВМ относительная адресация выполняется без суммирования по следующей схеме:



Автоиндексная адресация

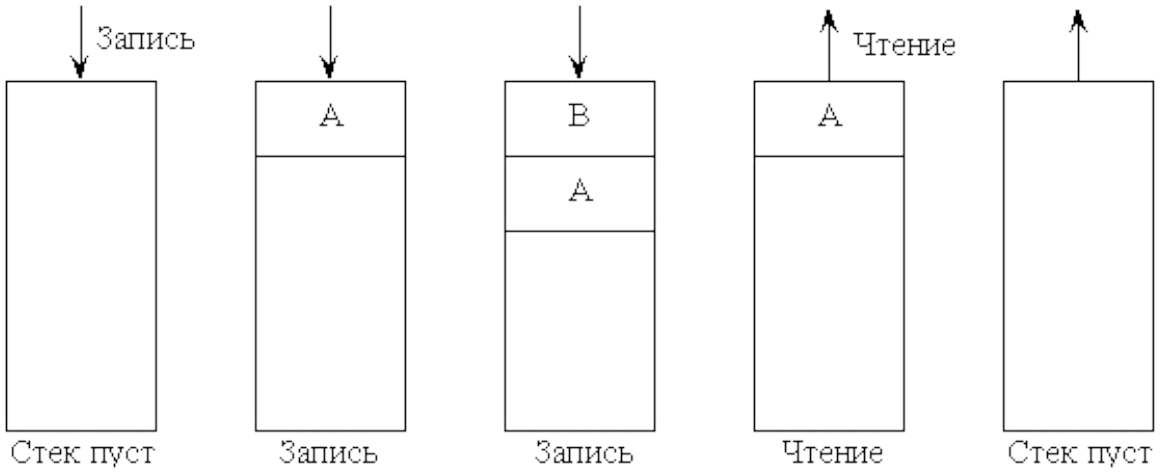
При автоиндексации косвенный адрес, находящийся в регистре РП, автоматически увеличивается (автоинкрементная адресация), или уменьшается (автодекрементная адресация) на постоянную величину до или после выполнения операции.

Существует достаточное количество способов адресации с использованием **базы** и **смещения**. В этом случае исполнительный адрес формируется путем сложения двух компонент – **базового адреса** и **смещения**. Базовый адрес загружается в специальный **регистр базы**. Для определения смещения используют уже перечисленные способы адресации.

Разновидностью прямой и косвенной адресации является ***регистровая*** – адресное поле команды указывает не на ячейку памяти, а на регистр:

1. ***Прямая регистровая*** – операнд непосредственно размещается в регистре;
2. ***Косвенная регистровая*** – регистр содержит адрес операнда памяти.

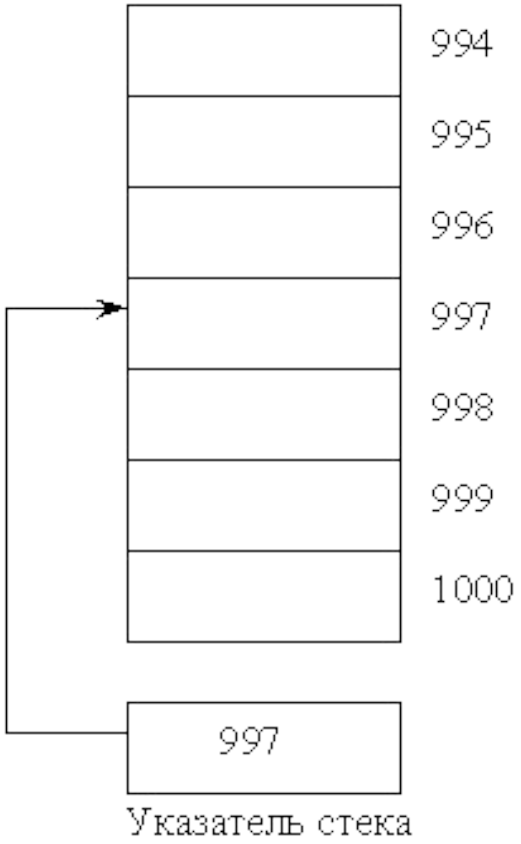
Стековая адресация



Запись в стек производится с использованием **автодекрементной адресации**, а чтение - с использованием **автоинкрементной адресации**.

Для чтения записи доступен только один регистр **v** вершина стека. Этот способ адресации используется, в частности, системой прерывания программ при вложенных вызовах подпрограмм.

Стековая память реализуется на основе обычной памяти с использованием ***указателя стека*** и ***автоиндексной адресации***.



Представление данных и машинные операции.

Операнды – это данные, которыми оперируют машинные команды.

К наиболее **общим (базовым) типам операндов** можно отнести:

1. адреса,
2. числа,
3. символы,
4. логические данные.

ВМ обеспечивает обработку и более **сложных информационных единиц**:

- графических изображений,
- аудио-,
- видео-,
- анимационной информации.

Среди цифровых данных можно выделить две группы:

2. **целые типы**, используемые для представления целых чисел, внутри могут быть представлены несколькими форматами;
3. **вещественные типы** для представления рациональных чисел, для представления чисел используется форма с плавающей запятой (ПЗ).

Беззнаковые и знакопеременные целые числа.

X - число

q – основание системы счисления или база

$X = \pm a^{n-1}\ldots a^1 a^0 a^{-1} a^{-2}\ldots a^{-r}$

Целая часть числа ***Дробная часть числа***

q^{n-1} q^1 q^0 q^{-1} q^{-2} q^{-r}

Знак	a_{n-1}	…	a_1	a_0	.	a_{-1}	a_{-2}	…	a_{-r}

$Q^{-r} = |X| = q^n - q^{-r}$

Формат без знакового разряда

2^{-1} 2^{-2} $2^{(n-3)}$ $2^{(n-2)}$

0 1 n-2 n-1

$2^{-(n-2)} <= x <= 1 - 2^{-(n-2)}$

Формат со знаковым разрядом

2^{-1} $2^{(n-2)}$ $2^{(n-1)}$

Знак				

0 1 n-2 n-1

$2^{-(n-1)} <= abs(x) <= 1 - 2^{-(n-1)}$

Формат без знакового разряда

2^{n-1} 2^{n-2} ... 2^1 2^0

0 1 ... n-2 n-1

$0 <= x <= 2^n - 1$

Формат со знаковым разрядом

2^{n-2} ... 2^1 2^0

0 1 ... n-2 n-1

$0 <= abs(x) <= 2^n - 1$

Пример: $A_2 = 11110000_2$

1	1	1	1	0	0	0	0

- в ячейках памяти

$2^n - 1$ - максимальное значение целого, т.е.

$$A = 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 1 \times 2^8 - 1 = 255_{10}$$

$2002_{10} = 11111010010_2$ - пример в прямом коде

0	0	0	0	0	1	1	1	1	1	0	1	0	0	1	0

- в ячейках памяти

$A = 2^{n-1} - 1$ - максимальное положительное число

$2^n - |A| + |A| = 0$ - дополнительный код

Прямой код модуля	$\lvert -2002_{10} \rvert$	0000011111010010_2
Обратный код	Инвертирование	1111100000101101_2
	Прибавление единицы	1111100000101101_2 + 0000000000000001_2
Дополнительный код		$1111100000101110 1_2$

$2^{n-1} - |A|$ - отрицательное число в дополнительном коде

$A = - 2^{n-1}$ - минимальное отрицательное число

$A = 2^{31} - 1 = 2\ 147\ 483\ 647_{10}$ - минимальное целое положительное число для длинных целых чисел со знаком

$A = -2^{31} = - 2\ 147\ 483\ 648_{10}$ - минимальное целое отрицательное число для длинных целых чисел со знаком

Достоинства представления чисел в формате с ***фиксированной запятой:***
а) простота и наглядность представления чисел,
б)простота алгоритмов реализации арифметических операций.
Недостаток представления чисел в формате с ***фиксированной запятой*** - небольшой диапазон представления величин.

Целочисленные форматы с фиксированной запятой, принятые в микропроцессорах фирмы Intel:

1. Байт (целое со знаком) – 8 бит, бит с номером 7 – знак;
2. Слово (целое со знаком) – 16 бит, бит с номером 15 – знак;
3. Двойное слово (целое со знаком) – 32 бита, бит с номером 31 – знак;
4. Байт (целое без знака) – 8 бит;
5. Слово (целое без знака) – 16 бит;
6. Двойное слово (целое без знака) – 32 бита;
7. Ближний указатель – 32 бита, хранит смещение или линейный адрес;
8. Длинный указатель или логический адрес – 48 бит, биты с номерами 47-32 хранят селектор сегмента, а биты с номерами 31-0 – смещение.

Упакованные целые числа

Формат предполагает упаковку в пределах достаточно длинного слова (обычно 64-разрядного) нескольких небольших целых чисел, а соответствующие команды обрабатывают все эти числа параллельно.

В микропроцессорах фирмы Intel, начиная с Pentium MMX присутствуют специальные команды для обработки мультимедийной информации (**MMX-команды**), оперирующие целыми числами, упакованными в **квадрослова** (64-разрядные слова).

Предусмотрены три формата:

1. упакованные байты (восемь 8-разрядных чисел);
2. упакованные слова (четыре 16-разрядных числа);
3. упакованные двойные слова (два 32-разрядных числа).

Байты в формате упакованных байтов нумеруются от 0 до 7, причем байт 0 располагается в младших разрядах квадрослова. Аналогичная система нумерации и размещения упакованных чисел применяется для упакованных слов (номера 0-3) и упакованных двойных слов (номера 0-1).

Идентичные форматы упакованных данных применяются также в другой технологии обработки мультимедийной информации, предложенной фирмой AMD и реализованной в микропроцессорах данной фирмы.