22. Форматы данных и методы адресации микропроцессоров фирмы INTEL

Новые команды процессоров 386+ поддерживают БИТОВЫЕ ДАННЫЕ:

- БИТ – одиночный двоичный разряд.
- БИТОВОЕ ПОЛЕ – группа до 32-х битов.
- ЦЕПОЧКА БИТОВ (СТРОКА) – набор последовательных битов, длиной до 4 Гбит.

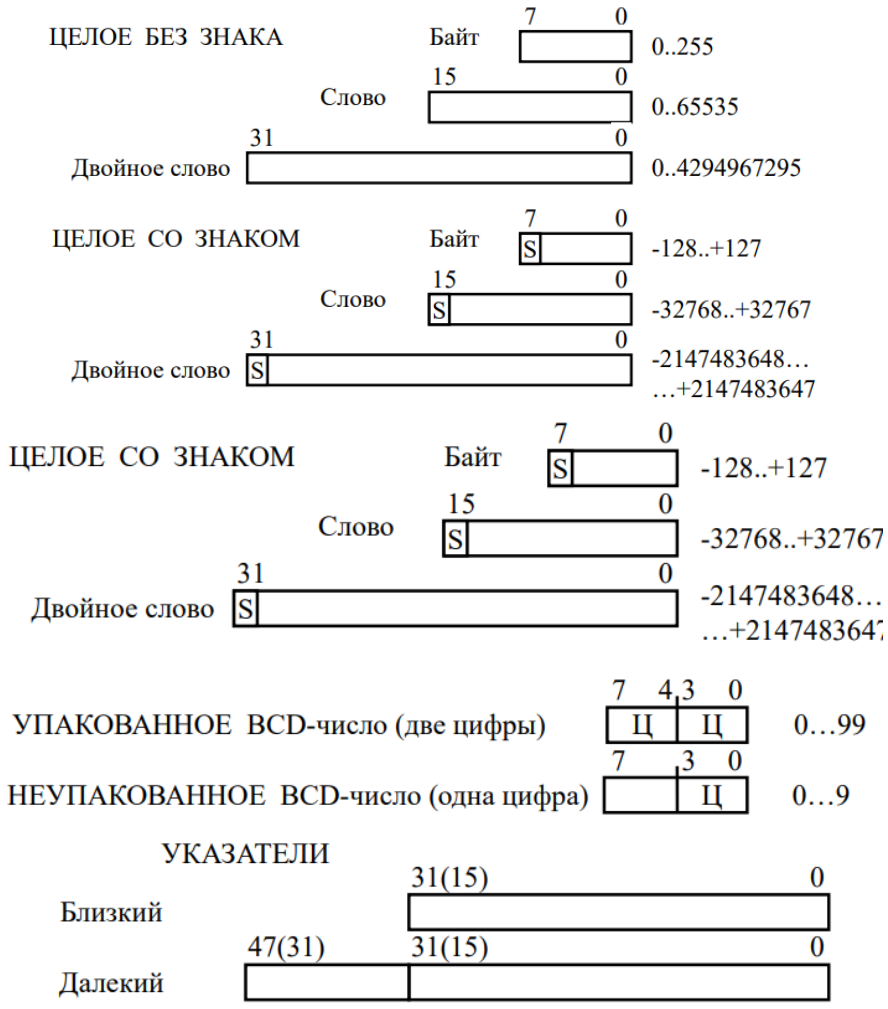

Рис. 4 – *Типы данных 32-х разрядных процессоров (386+)*

Процессор может легко оперировать с цепочками бит, байт, слов и двойных слов. Под ЦЕПОЧКОЙ (string) понимается последовательность практически любой длины отдельных, но взаимосвязанных элементов данных, ХРАНЯЩИХСЯ ПО СОСЕДНИМ АДДРЕСАМ. УКАЗАТЕЛИ применяются для обращения к некоторым объектам в памяти, например, адресам

подпрограмм. Близкие (NEAR) или внутрисегментный указатель (см. рис. 4) – это 16-ти битовое или 32-х битовое смещение внутри текущего сегмента. Далекий (FAR) или межсегментный указатель применяется в тех случаях, когда программа осуществляет передачу управления в другой сегмент. Такой указатель определяет новый сегмент (с помощью селектора) и 16-ти или 32-х битовое смещение внутри этого сегмента. При размещении операндов в памяти необходимо учитывать, что процессоры 386+ не накладывают ограничения на размещение данных. Однако производительность процессора повышается, если слова размещены по четным адресам, а двойные слова – по адресам, кратным четырем. Такой принцип называется ВЫРАВНИВАНИЕ АДРЕСОВ по границам слов или двойных слов. Выравнивание особенно важно для стека, который работает только со словами или двойными словами.

ВНИЗ ∨

3.1 РЕЖИМЫ (МЕТОДЫ) АДРЕСАЦИИ

Процессоры 386+ обеспечивает 13 режимов адресации, которые рассчитаны на эффективное выполнение программ, написанных на языках высокого уровня (ЯВУ) типа: C++, Фортран и др..

НЕЯВНАЯ АДРЕСАЦИЯ. Операнд адресуется неявно, если в команде нет специальных полей для его определения, т.е. операнд задается

полем команды. В ассемблерных кодах с неявной адресацией поле операнда пустое. Примеры команд с неявной адресацией:

```
AAA     ; Коррекция регистра AL после сложения
CMC     ; Инверсия флага переноса
STD     ; Установить в 1 флаг направления.
```

РЕЖИМ РЕГИСТРОВОЙ АДРЕСАЦИИ и
РЕЖИМ НЕПОСРЕДСТВЕННОЙ АДРЕСАЦИИ – предназначены, соответственно, для адресации одного из регистров регистрового блока или непосредственного операнда в команде с разрядностью 8, 16 или 32 бита :

```
INC     esi         ; Инкремент регистра ESI
SUB     ECX, ECX    ; Сбросить регистр ECX
MOV     EAX, CR0    ; Передать в EAX содержимое CR0.

MOV     EAX, 0F0F0F0F0h     ; Загрузить константу в EAX
AND     AL, 0FH     ; Выделить младшую тетраду регистра AL
BT      EDI, 3      ; Передать во флаг CF третий бит
                    ; регистра EDI
```

Имеется 10 режимов АДРЕСАЦИИ ПАМЯТИ. Исполнительный адрес включает в себя два компонента адреса ячейки памяти – сегмент и эффективный адрес (внутрисегментное смещение). ЭФФЕКТИВНЫЙ АДРЕС (EA) вычисляется суммированием следующих элементов :

- СМЕЩЕНИЕ (отклонение) – целая 8-ми или 32-х битовая величина со знаком, непосредственно задаваемая в команде (16-ти битовые отклонения могут использоваться при помощи префикса);
- БАЗА – содержимое любых РОНов. Базовые регистры обычно используются компиляторами в качестве точки отсчета локальной области памяти;
- ИНДЕКС – содержимое любых РОНов, исключая ESP. Индексные регистры используются для доступа к элементам строк или массивов.
- МНОЖИТЕЛЬ f - указывает шаг (1, 2, 4 или 8) для индексного регистра. Шаг индексации позволяет успешно адресовать массивы или структуры, содержащие многобайтовые операнды.

EA = БАЗА + ИНДЕКС * (ШАГ ИНДЕКСАЦИИ) + ОТКЛОНЕНИЕ.

Вычисление эффективного адреса (EA) практически не ухудшает производительность процессора из-за использования конвейерного режима.

РЕЖИМЫ АДРЕСАЦИИ ПАМЯТИ :

ПРЯМАЯ АДРЕСАЦИЯ – смещение (отклонение) адреса операнда содержится в 8, 16 или 32 разрядах команды :

```
MOV     AL, [2000h]          ; Передать байт в регистр AL
```



```
INC     dword prt [123456h]  ; Инкремент двойного слова
                             ; в памяти.
```

РЕГИСТРОВЫЙ КОСВЕННЫЙ МЕТОД АДРЕСАЦИИ – базовый или индексный регистр содержат адрес операнда :

```
MOV     AL, [ECX]        ; Передать в AL байт по адресу из ECX
DEC     word prt [ESI]   ; Декремент слова по адресу из ESI.
```

БАЗОВАЯ АДРЕСАЦИЯ – базовый регистр суммируется с отклонением:

```
MOV     EAX, [EBX+4]     ; Передать двойное слово из памяти
ADD     [ECX+10h], DX    ; Прибавить к слову в памяти.
```

ИНДЕКСНАЯ АДРЕСАЦИЯ – индексный регистр (любой РОН кроме ESP) суммируется с отклонением :

```
SUB     array[ESI], 2    ; Вычесть 2 из элемента массива
IMUL    vector[ECX]      ; Умножить EAX на элемент массива.
```

ИНДЕКСНАЯ АДРЕСАЦИЯ С ШАГОМ – содержимое индекс-ного регистра умножается на шаг «f» и суммируется с отклонением :

```
MOV     EAX, vec[ECX*4]          ; Переслать в EAX двойное слово
                                 ; из массива.
```

БАЗОВО-ИНДЕКСНАЯ АДРЕСАЦИЯ. – EA = БАЗА + ИНДЕКС :

```
ADD     EAX, [EBX][ESI]          ; Прибавить к EAX двойное
                                 ; слово  из памяти.
```

БАЗОВО-ИНДЕКСНАЯ АДРЕСАЦИЯ С ШАГОМ. – EA = БАЗА + ИНДЕКС * ШАГ:

```
INC     word prt [EDX][EDI*4]  ; Инкремент ячейки памяти.
```

БАЗОВО-ИНДЕКСНАЯ АДРЕСАЦИЯ С ОТКЛОНЕНИЕМ. – EA = БАЗА + ИНДЕКС + ОТКЛОНЕНИЕ:

```
MOV     AX, [ECX][ESI+20h]     ; Переслать слово из памяти
```

БАЗОВО-ИНДЕКСНАЯ АДРЕСАЦИЯ С ОТКЛОНЕНИЕМ И С ШАГОМ. – EA = БАЗА + ИНДЕКС * ШАГ + ОТКЛОНЕНИЕ:

```
ADD     AX, [EDX][EDI*4+10h]; Сложить AX с ячейкой памяти.
```

СТЕКОВАЯ АДРЕСАЦИЯ (можно рассматривать как вариант регистровой косвенной адресации) – в указателе стека ESP (SP) формируется 32-х битовое (16-ти битовое) внутрисегментное смещение для операнда в стековом сегменте :

```
PUSH    ECX     ; Включить в стек содержимое регистра
PUSHFD          ; Включить в стек содержимое EFLAGS
```



```
PUSH    4000h   ; Включить в стек константу
POP     EDX     ; Извлечь из стека в регистр
POPFD           ; Извлечь из стека в регистр EFLAGS
POP     [ESI]   ; Извлечь из стека в ячейку памяти
```

В таблице 1 показана разница в использовании базовых и индексных регистров для 16-ти и 32-х битовых адресов.

Для обеспечения совместимости ПО процессоров необходимо программы (с 16-ти битовыми командами МП 86 и 286) выполнять на МП 386+ в реальном или защищенном режимах. Процессор определяет размерность адреса, анализируя бит **D** (Default) в дескрипторе сегмента. Если D=0, то все длины операндов и эффективных адресов составляют 16 бит. Если D=1, – 32 бита. В реальном режиме – 16 бит.

Изменение размерности адреса и данных, задаваемых битом **D**, обеспечивают два префикса, выбираемые перед командами:

- ПРЕФИКС РАЗМЕРНОСТИ ОПЕРАНДА (OperandSize),
- ПРЕФИКС ДЛИНЫ АДРЕСА (AddressSize).

Наличие префикса коммутирует (переключает) размер операнда или размер эффективного адреса на значение, противоположное принимаемому по умолчанию (по биту **D**).

Префиксы могут использоваться совместно с любой инструкцией и в любом режиме – реальном, виртуальном и V86. Префикс длины адреса не обеспечивает размерность адреса более 64 Кбайт в режиме реальной адресации. Адрес свыше 0FFFFh будет рассматриваться как ошибка.

Таблица 1 – *Базовые и индексные регистры для 16-ти и 32-х битовых адресов*

	16-ти битовый адрес	32-х битовый адрес
Базовый регистр	BX, BP	Любой 32-х битовый РОН
Индексный регистр	SI, DI	Любой 32-х битовый РОН, исключая ESP
Шаг индексации «f»	нет	1, 2, 4, 8
Смещение	0, 8, 16 бит	0, 8, 32 бит