РЕЖИМЫ АДРЕСАЦИИ

Режимы адресации можно разделить на семь групп:

1.Регистровая адресация, при которой микропроцессор извлекает операнд из регистра или загружает его в регистр.

```
MOV AX,CX
```

2.Непосредственная адресация, которая позволяет указывать 8-или 16-битовое значение константы в качестве операнда - источника.

```
MOV CX,25
```

3.Прямая адресация, при которой исполнительный адрес является составной частью команды. Микропрооцессор добавляет этот исполнительный адрес к сдвинутому содержимому регистра сегмента данных DS и получает 20-битовый физический адрес операнда.

```
MOV AX,DAN1
```

4.Косвенная регистровая адресация, при которой исполнительный адрес операнда содержится в базовом регистре BX, регистре указателе базы BP или индексном регистре (SI или DI). Косвенные регистровые операнды надо заключать в квадратные скобки, чтобы отличить их от регистровых операндов.

```
MOV BX,OFFSET DAN1        ;смещение адреса в BX
MOV AX,[BX]                     ;в AX - содержимое ячейки
                                          ;памяти, адресуемой
```

регистром BX

5.Адресация по базе, когда ассемблер вычисляет исполнительный адрес с помощью сложения значения сдвига с содержимым регистров BX или BP, в которые загружается базовый адрес.

```
MOV AX,[BX]+4
```

6.Прямая индексация с индексированием, при которой исполнительный адрес вычисляется как сумма значений сдвига и индексного регистра DI или SI). Этот тип адресации удобен для доступа к элементам таблицы, когда сдвиг указывает на начало таблицы, а индексный регистр - на ее элемент.

```
MOV DI,5            ;6-й элемент таблицы
MOV AL,TABL[DI]     ;в регистр AL
MOV DI,10           ;6-й элемент таблицы слов
MOV AX,TABL[DI]     ;в регистр AX
```

7.Адресация по базе с индексированием, при которой исполнительный адрес вычисляется как сумма значений базового регистра, индексного регистра и, возможно, сдвига. Этот режим адресации удобен при работе с двумерными массивами, когда базовый регистр содержит начальный адрес массива, а значения сдвига и индексного регистра - смещения по строке и столбцу.

```
MOV AX,VAL[BX][DI]   ;операнды можно заключать в скобки
MOV AX,[BX+2+DI]     ;в любом порядке, а сдвиг можно
MOV AX,[BX][DI+2]    ;сочетать с любым из регистров
```

ТИПЫ КОМАНД

Можно разделить систему команд на семь функциональных групп:

1. Команды пересылки данных.
2. Арифметические команды.
3. Команды манипулирования битами, или команды сдвига
4. Команды передачи управления.
5. Команды обработки строк.
6. Команды прерывания.
7. Команды управления процессором.

КОМАНДЫ ПЕРЕСЫЛКИ ДАННЫХ

Команды пересылки данных осуществляют обмен данными и адресами между регистрами и ячейками памяти или портами ввода-вывода.

Команда MOV может пересылать байт или слово между регистром и ячейкой памяти или между двумя регистрами. Она может также пересылать непосредственно адресуемое значение в регистр или ячейку памяти.

MOV приемник,источник

```
MOV  AX,DAN       ; пересылка из памяти в регистр
MOV  DAN,AX       ; и наоборот
MOV  ES:[BX],AX   ; замена используемого регистра сегмента
MOV  DS,AX        ; пересылка между 16-битовыми регистрами
MOV  BL,AL        ; пересылка между 8-битовыми регистрами
MOV  CL,11        ; пересылка константы в регистр
MOV  STOR,12H     ; или в память
```

Командой MOV нельзя осуществить пересылку данных из одной ячейки памяти в другую. Это можно сделать только через регистр общего назначения:

```
MOV AX,DAN
MOV DAN1,AX
```

Командой MOV нельзя загрузить непосредственно адресуемый операнд в регистр сегмента. Сначала его нужно загрузить в регистр общего назначения.

```
MOV AX,SDATA
MOV DS,AX
```

Нельзя непосредственно переслать значение одного сегментного регистра в другой. Такие пересылки нужно делать через регистр общего назначения.

```
MOV AX,ES   ;регистр DS указывает на тот же сегмент,
MOV DS,ax   ;что и регистр ES
```

Нельзя использовать регистр CS в качестве приемника в команде пересылки.

Команды PUSH и POP непосредственно работают со стеком. Команда PUSH помещает содержимое регистра или ячейки памяти размером 2 байта на вершину стека, а команда POP снимает слово с вершины стека и помещает его в ячейку памяти или регистр.

```
PUSH источник
POP  приемник

PUSH AX
......
POP  AX
```

При работе со стеком следует помнить, что извлекаемый из стека элемент данных - это всегда тот элемент, который был записан туда последним. Если выполняется последовательность команд:

```
PUSH BX
PUSH CX
......
POP  BX
POP  CX
```

то в результате произойдет обмен значений в регистрах BX и CX. Если в команде PUSH был указан регистр BX, это неозначает, что команда POP, указывающая на этот регистр восстанавливает первоначальное значение регистра BX. Команды PUSH и POP должны быть сбалансированы, т.е. каждой команде PUSH должна соответствовать команда POP.

Для помещения в стек и извлечения из стека регистра флагов используются команды PUSHF и POPF без операндов.

Команда LAHF копирует флаги CF,PF,AF,ZF,SF в соответствующие биты регистра AH. Команда SAHF загружает эти 5 битов из регистра AH в регистр флагов.

Команда обмена XCHG меняет между собой значения двух регистров или регистра и ячейки памяти. Однако она не может выполнить обмен значений регистров сегмента.

```
XCHG AX,BX
XCHG AL,BH
XCHG DS,WORD
```

Команда извлечения элемента таблицы XLAT выбирает значение из таблицы и загружает его в регистр AL. Таблица может иметь до 256 элементов.

XLAT имя таблицы

Извлечь 7 байт из таблицы TABL

```
MOV AL,7                    ;номер байта в AL
MOV BX,OFFSET TABL          ;смещение адреса в BX
XLAT TABL                   ;извлечение из таблицы в AL
```

Команды ввода-вывода используются для взаимодействия с ПУ.

```
IN  аккумулятор, порт   ;аккумулятор - регистр
OUT порт,аккумулятор    ;AL или AX
```

Команды пересылки адреса передают не переменные, а их адреса.

Команда LEA пересылает смещение ячейки памяти в любой 16-битовый регистр общего назначения, регистр указателя или индексный регистр.

```
LEA  регистр16,память16
LEA  BX,TABL
```

где операнд память16 должен иметь атрибут типа WORD.

Команда LDS - загрузка указателя и регистра сегмента данных считывает из памяти 32-битовое двойное слово и загружает первые 16 битов в заданный регистр, а следующие 16 битов - в регистр сегмента данных DS.

```
LDS регистр16,память32
```

где регистр16 - любой регистр общего назначения, а память32 - ячейка памяти с атрибутом DD.

Команда LES - загрузка указателя и регистра дополнительного сегмента идентична команде LDS, но загружает номер блока в регистр ES.

АРИФМЕТИЧЕСКИЕ КОМАНДЫ

Микропроцессор может выполнять арифметические команды над двоичными числами со знаком или без знака, а также над десятичными числами без знака как упакованными, так и неупакованными.

Команда сложения ADD и команда сложения с добавлением переноса ADC могут складывать как 8-, так и 16-битовые операнды. Команда ADD складывает содержимое операнда - источника и операнда - приемника и помещает результат в операнд приемник. Команда ADC делает то же, но при сложении использует флаг переноса CF.

```
ADD  AX,CX  ;складываются 16-битовые значения регистров AX CX и
            ;результат - в AX
```

Если длина операндов более 16 бит, то каждый помещается в пару регистров

```
ADD AX,CX ;сначала складываются младшие 16 битов, а затем
ADC BX,DX ;старшие 16 битов и команда ADC добавляет к сумме
          ;[DX]+[BX] любой перенос от сложения [CX]+[AX]
```

Можно добавить значение ячейки памяти к регистру и наоборот, можно сложить константу с содержимым регистра или к ячейки памяти. Но нельзя добавить значение одной ячейки памяти к другой.

Команды ADD и ADC могут воздействовать на шесть флагов: флаг переноса CF; флаг четности PF; вспомогательный флаг переноса AF; флаг нуля ZF; флаг знака SF; флаг переполнения OF.

При выполнения сложения микропроцессор рассматривает операнды как двоичные числа, поэтому при сложении десятичных чисел результат должен быть скорректирован для представления в десятичной форме. Коррекция результата сложения десятичных чисел осуществляется командами AAA и DAA. Эти команды без операндов, но предполагается,что корректируемоезначение находится в регистре AL.

Команда AAA преобразует содержимое регистр AL в правильную неупакованную десятичную цифру в младших четырех битах регистра AL и заполняет нулями старшие четыре бита. Если результат превышает 9, то команда AAA добавляет 1 к содержимому регистра AH и устанавливает флаг CF=1

```
ADD AL,BL  ;сложить неупакованные числа
AAA        ;и преобразовать результат в неупакованное число
```

Команда DAA преобразует содержимое регистра AL в две правильные упакованные десятичные цифры. Если результат превышает 99, то команда DAA добавляет 1 к содержимому регистра AH и устанавливает флаг CF=1

```
ADD AL,BL  ;сложить упакованные числа
DAA        ;и преобразовать результат в упакованное число
```

Команда INC добавляет 1 к содержимому регистра или ячейки памяти,но в отличие от команды ADD не воздействует на флаг переноса CF.

Команды вычитания SUB и вычитания с переносом SBB аналогичны соответствующим командам сложения ADD и ADC, только при вычитании флаг переноса CF действует как признак заема. Команда SUB вычитает операнд-источник из операнда-приемника и возвращает результат в операнд-приемник. Команда SBB делает то же самое, но дополнительно вычитает значения флага переноса CF.

```
SUB AX,CX
SUB AX,CX ;вычитается 32-битовое число, помещенное в регистры CX DX
SBB BX,DX ;из 32-битового числа, помещенного в регистры AX и BX
```

Можно вычитать из содержимого регистра содержимое ячейки памяти и наоборот или вычитать из содержимого регистра либо ячейки памяти не посредственное значение. Но нельзя вычесть значение одной ячейки памяти из другой или вычитать из непосредственного значения содержимое регистра либо ячейки памяти

Команды SUB и SBB могут воздействовать на те же 6 флагов, что и команды ADD и ADC.

Коррекция вычитания двух десятичных чисел осуществляется командами AAS и DAS.

Команда AAS преобразует содержимое регистр AL в правильную неупакованную десятичную цифру в младших четырех битах регистра AL и заполняет нулями старшие четыре бита. Если результат превышает 9, то команда AAS вычитает 1 из содержимого регистра AH и устанавливает флаг CF=1

```
SUB AL,BL  ; вычесть неупакованные числа
DAS        ;и преобразовать результат в неупакованное число
```

Команда DAS преобразует содержимое регистра AL в две правильные упакованные десятичные цифры. Если результат превышает 99, то команда DAS вычитает 1 из содержимого регистра AH и устанавливает флаг CF=1

```
SUB AL,BL  ;вычесть упакованное число BL из AL
DAS        ;и преобразовать результат в упакованное число
```

Команда DEC вычитает 1 из содержимого регистра или ячейки памяти, но при этом не воздействует на флаг переноса CF.

Команда обращения знака NEG вычитает значение операнда приемника из нулевого значения. Она оказывает на флаг то же действие, что и команда SUB. Команда NEG полезна для вычитания значения регистра или ячейки памяти из непосредственного значения. Т.к. недопустима команда

```
SUB 100,AL
```

ее можно заменить парой команд

```
NEG AL       ;обращаем знак содержимого регистра AL и
ADD AL,100   ;добавляем к нему 100
```

Команда сравнения CMP сравнивает два числа, вычитая одно из другого, но в отличие от команды SUB не сохраняет результат вычитания. Она целиком предназначена для установки значений флагов, на основании которых команды условного перехода передают значения.

Команды умножения чисел без знака MUL и команда умножения целых чисел со знаком IMUL могут умножать как байты, так и слова. Они имеют формат:

```
MUL   источник
IMUL  источник
```

где источник - регистр общего назначения или ячейка памяти размером в байт или слово. В качестве второго операнда команду умножения используют содержимое регистра AL при операциях над байтами или содержимое регистра AX при операциях над словами. Произведение имеет двойной размер и возвращается следующим образом: при умножении байтов в регистре AH старший байт и в регистре AL младший байт; при умножении слов в регистре DX старшее слово и в регистре AX младшее слово.

```
MUL  BX     ;умножить BX на AX без знака
MUL  BSTOR  ;умножить яч.памяти 1 байт на AL без знака
IMUL DL     ;умножить DL на AL со знаком
INUL WSTOR  ;ячейку памяти 2 байта на AX со знаком
```

Команды MUL и IMUL не позволяют в качестве операнда использовать непосредствен-ное значение.

Для коррекции результатов умножения используется команда AAM, которая преобразует результат предшествующего умножения байтов в два правильных неупакованных десятичных операнда.

Команда DIV выполняет деление чисел без знака, а команда IDIV выполняет деление целых чисел со знаком. Эти команды имеют формат:

```
DIV   источник
IDIV  источник
```

где источник - делитель размером в байт или слово, находящийся в регистре общего назначения или в ячейке памяти. Делимое должно иметь двойной размер. Оно извлекается из регистров AH и AL при делении на 8-битовое число или из регистров DX и AX при делении на 16-битовое число. Результаты возвращаются следующим образом:

Если операнд-источник представляет собой байт, то частное возвращаетсяв регистре AL, а остаток в регистре AH. Если операнд-источник представляет собой слово, то частное возвращается в регистре AX, а остаток - в регистре DX.

```
 DIV BX   ;разделить DX:AX на BX без знака
 DIV MEMB ;разделить AH:AL на байт памяти без знака
IDIV DL   ;разделить AH:AL на DL со знаком
IDIV MEMW ;разделить DX:AX на слово памяти со знаком
```

Команды DIV и IDIV не позволяют разделить на непосредственное значение, его надо предварительно загрузить в регистр или ячейку памяти.

Команда коррекции деления AAD в отличие от других команд десятичной коррекции, должна выполняться непосредственно перед операцией деления. Команда ADD преобразует неупакованное делимое в двоичное значение и загружает его в регистр AL, а затем обнуляет регистр AH.

КОМАНДЫ МАНИПУЛИРОВАНИЯ БИТАМИ

К командам манипулирования битами относятся логические команды, команды сдвига и команды циклического сдвига.

Операндами логических команд AND (И), OR (ИЛИ), XOR (Исключающее ИЛИ) могут быть байты или слова. В них можно сочетать два регистра, регистр с ячейкой памяти, или непосредственное значение с регистром или ячейкой памяти.

При исполнения команды AND биты операнда приемника становятся равным 0 всюду, где операнд источник содержит 0, и сохраняются там, где операнд источник содержит 1.

```
AND AX,BX
AND BL,1101B
```

Команда OR полагает равным 1 те биты операнда-приемника, в позициях которых хотя бы один из операндов содержит 1.

OR BX,0C00H

Команда XOR полагает равным 1 все те биты приемника, в позициях которых операнды имеют различные значения.

Команда NOT (НЕ) обращает состояние каждого бита регистра или ячейки памяти и ни на какие флаги не воздействует.

Команда TEST - проверить выполняет операцию AND над операндами, но воздействует только на флаги и не изменяет значения операндов.

У микропроцессора есть семь команд, осуществляющих сдвиг 8-битового и 16-битового содержимого регистров или ячеек памяти на одну или несколько позиций влево или вправо. Для всех команд сдвига флаг CF преобретает значение бита, сдвинутого за один из концов операнда. Команды сдвига вправо помещают во флаг CF значение нулевого бита. Команды сдвига влево помещают в него значение бита 7 при операциях над байтами или бита 15 при операциях над словами.

Логические команды сдвига используются для действий над числами без знака или над нечисловыми значениями. Арифметические команды сдвига используются для действий над числами со знаком и сохраняют старший, знаковый бит операнда.

КОП приемник,счетчик

где приемник - 8- или 16-битовый регистр общего назначения или ячейка памяти, а счетчик может быть цифрой 1 или значением без знака в регистре CL.

Команды SAL сдвиг влево арифметический и SAR сдвиг вправо арифметический сдвигают числа со знаком. Команда SAR сохраняет знак операнда, распространяя его при выполнении сдвига. Команда SAL не сохраняет знак, но заносит 1 во флаг переполнения OF в случае изменения знака операнда. При каждом сдвиге операнда команда SAL заносит 0 в вакантный нулевой бит этого операнда.

Команды SHL - сдвиг влево логический и SHR - сдвиг вправо логический сдвигают числа без знака. Команда SHL идентична команде SAL. При каждом сдвиге операнда команда SHR заносит 0 в вакантный старший бит этого операнда.

Для экономии времени команды сдвига можно применять вместо команд умножения и деления.

Команды циклического сдвига в отличие от команд сдвига сохраняют сдвинутые за пределы операнда биты, помещая их обратно в операнд. Сдвинутый за пределы операнда бит запоминается во флаге переноса CF. При исполнении команды ROL - сдвинуть влево циклически и ROR - сдвинуть вправо циклически вышедший за пределы операнда бит входит в него с другого конца. При исполнении команды RCL - сдвинуть вправо циклически вместе с флагом переноса в противоположный конец операнда помещается значение флага переноса CF.

КОМАНДЫ ПЕРЕДАЧИ УПРАВЛЕНИЯ

Команды передачи управления обеспечивают переход от одной части программы в другую. Эти команды можно подразделить на три группы: команды безусловной передачи управления, команды условной передачи управления и команды управления циклами.

Регистры CS:IP показывают, какая следующая команда должна быть выполнена. Команды перехода модифицируют указатель команды IP и, возможно, регистр сегмента команд CS. Если команда перехода изменяет только регистр IP, это - внутрисегментный переход. Он также называется близким NEAR - переходом. Если переход изменяет регистр CS, это межсегментный или далекий FAR - переход. Любая программная метка на языке ассемблера имеет атрибут либо NEAR, либо FAR. Ассемблер использует эту информацию для того, чтобы определить, какой тип команды перехода или вызова сгенерировать при переходе к этой метке. Существует два метода вычисления адреса перехода. Если в команде указано значение адреса, это - абсолютный переход.Команда может указать место перехода как некоторое расстояние от нее самой. Этот переход называется относительным.

Безусловный переход - это такой переход, который передает управление всякий раз, когда он выполняется.Условный переход проверяет текущее состояние, чтобы определить, передавать управление или нет.

Все команды вызова процедуры CALL безусловны. Команды CALL осуществляют функции запоминания адреса возврата и передачи управления процедуре. Она помещает в стек адрес возврата, занимающий 2 байта, если процедура определена с атрибутом NEAR, и 4 байта, если она определена с атрибутом FAR.

Командам CALL соответствуют команды возврата RET. Все возвраты косвенные переходы, поскольку они извлекают адрес перехода из вершины стека.

Команда безусловного перехода JMP идентична команде CALL по возможностям адресации.

JMP имя

где операнд имя может иметь атрибут NEAR или FAR, быть прямым или косвенным.

Если известно, что переход делается на метку,лежащую в диапазоне +-128 байт от текущего места, то можно указать тип операнда SHORT корокий. JMP SHORT METKA

Команды условного перехода проверяют текущее состояние регистра флагов и в зависимости от кодов условия команда делает или не делает переход. Команды условного перехода не устанавливают флаги, а только проверяют их текущее состояние.

команда	описание	условие перехода
JA	перейти, если выше	CF=0 и ZF=0
JAE	перейти, если выше или равно	CF=0
JB	перейти, если ниже	CF=1
JBE	перейти, если ниже или равно	CF=1 илиZF=1
JC	перейти, если перенос	CF=1
JCXZ	перейти, если значение регистра CX=0	CX=0
JE	перейти, если равно	ZF=1
JG*	перейти, если больше	ZF=0 и SF=OF
JGE*	перейти, если больше или равно	SF=OF
JL*	перейти, если меньше	SF<>OF
JLE*	перейти, если меньше или равно	ZF=1 или SF<>OF
JNA	перейти, если не выше	CF=1 или ZF=1
JNAE	перейти, если не выше и не равно	CF=1
JNB	перейти, если не ниже	CF=0
JNBE	перейти, если не ниже и не равно	CF=0 и ZF=0
JNC	перейти, если нет переноса	CF=0
JNE	перейти, если не равно	ZF=0
JNG*	перейти, если не больше	ZF=1 или SF<>OF
JNGE*	перейти, если не больше и не равно	SF<>OF
JNL*	перейти, если не меньше	SF=OF
JNLE*	перейти, если не меньше и не равно	ZF=0 и SF=OF
JNO*	перейти, если нет переполнения	OF=0
JNP	перейти, если нет четности	PF=0
JNS*	перейти, если знаковый разряд нулевой	SF=0
JNZ	перейти, если не нуль	ZF=0
JO	перейти, если переполнение	OF=0
JP	перейти, если есть четность	PF=1
JPE	перейти, если сумма битов четная	PF=1
JPO	перейти, если сумма битов нечетная	PF=0
JS	перейти, если знаковый бит равен 1	SF=1
JZ	перейти, если если нуль	ZF=1

знаком * отмечены команды,относящиеся к действиям над числами со знаком.

Командам условной передачи управления могут предшествовать любые команды, изменяющие состояние флагов, но чаще всего они используются с командой сравнения CMP.

```
          ADD AL,BL  ;переход к метке M1, если при сложении
          JC  M1     ;возник перенос
          CMP AL,BL  ;переход к метке M2, если значения регистров
          JE  M2     ;AL и BL одинаковы
```

перейти на метку M3, если содержимое регистра BX имеет большее значение, чем содержимое регистра AX

```
          CMP BX,AX  ;
          JA  M3     ;если операнды не имеют знака

          CMP BX,AX  ;
          JG  M3     ;если операнды со знаком.
```

Команды управления циклами обеспечивают условные передачи управления при организации циклов. Регистр CX служит счетчиком числа повторений цикла. Команда LOOP уменьшает содержимое регистра CX на 1 и передает управления операнду близкая метка, если содержимое регистра CX не равно 0. Если счетчик в регистре CX=0, программа продолжает выполнятся после команды LOOP. Тело цикла выполняется столько раз, сколько было задано содержимым регистра CX

```
        MOV CX,COUNT
M1:     ....
        ....

        LOOP M1
```

Если в регистр CX перед началом работы цикла был загружен 0, то цикл выполнится 65536 раз, т.к. после выполнения первой команды LOOP содержимое регистра CX уменьшается от 0 до 0FFFFH. Команда JCXZ - пе реход, если содержимое регистра CX равно 0 - проверяет текущее содержимое регистра CX и делает переход, если оно равно 0.

```
        MOV CX,COUNT
        JCXZ ENDLOOP
M1:     ...
        ...
        LOOP M1
```

Команда LOOPE - повторять цикл, пока равно, имеющая синоним LOOPZ - повторять цикл, если 0 - завершает повторение цикла, если содержимое регистра CX = 0, или флаг ZF = 0, или оба они равны 0.

Команда LOOPNE - повторять цикл, пока не равно, имеющая синоним LOOPNZ - повторять цикл, пока не нуль - завершает повторение цикла, если содержимое регистра CX = 0, либо флаг ZF=1, либо выполнено и то,и другое.

КОМАНДЫ ОБРАБОТКИ СТРОК

Команды обработки строк позволяют производить действия над строками байтов или слов памяти. Эти строки могут иметь длину до 64Кбайт и состоять из числовых значений,алфавитно-цифровых,а также из любых других значений, которые могут храниться в памяти в виде двоичных кодов.

Команды обработки строк предоставляют возможность выполнения пяти основных операций, называемых примитивами, которые обрабатывают строку по одному элементу (байту или слову) за прием. Автоматическое уменьшение или увеличение адреса дает возможность быстрой обработки строковых данных. Флаг направления в регистре флагов управляет направлением обработки. Когда он установлен равным 1, адрес уменьшается, если флаг сброшен в 0 - то увеличивается. Байтовые команды обработки строк изменяют адрес на 1 после каждой операции, а команды обработки строк над словами изменяют адрес на 2.

Микропроцессор предполагает, что строка-источник находится в сегменте данных, а строка-приемник - в дополнительном сегменте. Процессор адресует строку-источник через пару регистров DS:SI, а строку-приемник через пару регистров ES:DI.

Команда загрузки строки LODS при работе со строкой байтов загружает в регистр AL байт, на который указывает пара регистров DS:SI. Затем она изменяет регистр SI на 1. При работе со строкой слов команда LODS загружает регистр AX и изменяет регистр SI на 2.

LODS строка-источник (типа BYTE или WORD)

Команда LODS имеет сокращенные формы LODSB - загрузить строку байтов и LODSW - загрузить строку слов.

Команда STOS - сохранить строку - строго противоположна команде LODS и пересылает байт из регистра AL или слово из регистра AX в ячейку памяти, находящейся в дополнительном сегменте и адресуемой парой регистров ES:DI, а затем изменяет значениу регистра DI так, чтобы оно указывало на следующий элемент строки.

STOS строка-приемник

Сокращенные формы команды - STOSB - сохранить строку байт и STOSW - сохранить строку слов.

Префиксы повторения заставляют микропроцессор повторять команду обработки строк. Число повторений извлекается из регистра CX.

REP STOSB

Эта команда повторяется до тех пор, пока содержимое регистра CX не уменьшится до нуля. Команда STOSB записывает байт из регистра AL в ячейку памяти, которая адресуется парой регистров ES:DI и изменяет регистр DI, затем префикс REP уменьшает регистр CX и, если он не равен 0, повторяет команду.

```
 PROB    DW  25 DUP (?)
         .....
         MOV AX,0
         MOV CX,25
         LEA DI,PROB
 REP     STOSW
```

Префикс REPE - повторять пока равно, имеющий синоним REPZ повторять, пока 0, повторяет команду, пока флаг ZF=1 и значение регистра CX не равно 0.

Действие префикса REPNE - повторять, пока не равно, имеющего синоним REPNZ - повторять, пока не нуль, противоположно префиксу REPE. Иначе, префикс REPNE обеспечивает повторение модифицированной им команды, пока флаг ZF=0 и значение регистра CX не равно 0.

Команда пересылки строки MOVS копирует байт или слово из одной части памяти в другую.

MOVS строка-приемник,строка-источник

Здесь строка-источник - строка в сегменте данных, а строка-приемник - строка в дополнительном сегменте. Оба операнда должны быть одинаковых типов.

Сокращенные команды MOVSB - переслать строку байтов и MOVSW - переслать строку слов.

Комбинация команды MOVS с префиксом REP дает эффективную команду пересылки блока.

Команда сравнения строк CMPS сопоставляет операнд-источник с операндом-приемником и возвращает результат через флаги. Команда CMPS не изменяет значения операндов.

CMPS строка-приемник,строка-источник

Подобно команде CMP команда CMPS сравнивает операнды с помощью их вычитания. Однако, CMP вычитает операнд-источник из операнда-приемника, а команда CMPS вычитает операнд-приемник из операнда-источника. Это означает, что указываемые после команды CMPS команды условной передачи управления должны отличаться от тех, которые бы следовали за командой CMP в аналогичной ситуации.

Для сравнения нескольких элементов команду CMPS надо использовать с префиксом повторения REPE(REPZ) или REPNE(REPNZ), но не имеет смысла использовать префикс REP. После команды CMPS удобнее всего использовать команды условной передачи управления, проверяющие значение флага ZF т.е. команды JE или JNE.

```
         CLD
         MOV CX,100
 REPNE   CMPS DEST,SOURCE  ;сравнивать, пока не равно
         JNE NF            ;совпадение обнаружено?
         .....             ;да
NF:      .....             ;нет
```

Ассемблер преобразует команду CMPS либо в команду CMPSB при сравнении байтов или в команду CMPSW при сравнении слов. Эти команды, не имеющие операндов, можно использовать самостоятельно.

Команды сканирования строк позволяют осуществить поиск заданного значения в строке, находящейся в дополнительном сегменте. Смешение адреса первого элемента строки должно быть помещено в регистр DI. При сканировании строки байтов искомое значение должно находиться в регистре AL, а при сканировании строки слов - в регистре AX.

Основная команда сканирования строк SCAS имеет формат

SCAS строка приемник

Как и в случае команды CMPS, для выполнения действий более чем над одним элементом строки надо воспользоваться префиксами повторения REPE(REPZ) или REPNE(REPNZ). Команды SCASB - поиск байта и команда SCASW - поиск слова ориентированы на конкретный размер элементов и не имеют операндов.

КОМАНДЫ УПРАВЛЕНИЯ ПРОЦЕССОРОМ

Эти команды позволяют управлять работой микропроцессора из программы. Сюда относятся команды управления флагами, команды внешней синхронизации и команда холостого хода.

Команда STC устанавливает флаг переноса CF, а команда CLC обнуляет флаг переноса CF.

Команда CMC - обратить флаг переноса - переводит флаг CF в состояние 0, если он имел состояние 1 и наоборот.

Команда STD устанавливает флаг направления DF, а команда CLD обнуляет флаг направления DF. Эти команды используются для указания направления обработки строк.

Команда CLI обнуляет флаг прерывания IF, что заставляет микропроцессор игнорировать маскируемые прерывания, инициируемые внешними устройствами.

Команда CTI переводит флаг прерываний IF в состояние 1, что разрешает микропроцессору реагировать на прерывания, инициируемыми внешними устройствами.

Команды внешней синхронизации используются в основном для синхронизации действий микропроцессора с внешними событиями.

Команда HLT - остановиться переводит микропроцессор в состояние останова, при котором он находиться на холостом ходу и не выполняет никакие команды. Микропроцессор выходит из состояния останова только в том случае, если его снова инициировать.

Команда WAIT - ожидать также переводит микропроцессор на холостой ход, но при этом через каждые пять тактов он проверяет активность входной линии по имени TEST. При этом он способен обрабатывать прерывания, но по завершении программы обработки прерывания МП вновь возвратится в это состояние. Если линия TEST становиться активной, МП переходит к следующей за WAIT команде. Назначение команды WAIT – останов микропроцессора на то время, пока некоторое внешнее устройство не завершит свою работу.

Команда ESC заставляет микропроцессор извлечь содержимое указанного в ней операнда и передать его на шину данных. Используется чаще всего для передачи команд математическому сопроцессору.

Команда NOP - нет операции - не выполняет никакой операции. Она не действует ни на флаг, ни на регистры, ни на ячейки памяти; она только увеличивает значение указателя команд IP.