9. Обработка массивов. Индексная и косвенная адресация

При обработке массивов надо прежде всего выделить область памяти для хранения элементов массива. Это выполняется обычным образом с помощью директив объявления данных и директивы дублирования DUP:

```
MasW   DW   50   DUP  (?)      ; 50 элементов длиной в слово
MasB   DB   200  DUP  (?)      ; 200 байтовых элементов
MasD   DD   100  DUP  (?)      ; 100 двойных слов
```

В этих объявлениях имена MasW, MasB и MasD определяют адреса размещения в памяти первых элементов массива. Для перехода ко вторым элементам начальные адреса надо увеличить на размер элемента, т.е. использовать следующие выражения: MasW+2, MasB+1, MasD+4. Аналогично – для перехода к последующим элементам.

Ясно, что данные выражения имеют очень ограниченное использование, поэтому для организации доступа к любому элементу массива в архитектуру процессоров была введена так называемая индексная адресация. Ее суть состоит в том, что один из РОНов объявляется индексным и хранящееся там значение трактуется процессором как байтовое смещение относительно первого элемента массива. Изменяя значение индексного регистра, можно менять базовый адрес за счет прибавления к нему некоторого смещения. Динамическое изменение адресов называется их модификацией. Она выполняется процессором при обработке команды: вычисляется так называемый исполнительный (эффективный) адрес как сумма базового адреса и содержимого индексного регистра, и именно по этому исполнительному адресу выполняется выборка операнда:

исполнительный адрес = базовый адрес + (индексный регистр)

В качестве индексных можно использовать только регистры BX, BP, SI, DI. При этом имя индексного регистра в операнде команды задается после имени массива с использованием квадратных скобок. Использование индексных регистров включает следующие шаги:

- занесение в индексный регистр начального значения (чаще всего – 0)

- использование текущего значения индексного регистра в команде
- изменение индексного регистра на 1, 2 или 4 байта

Реализация двух последних шагов в теле цикла позволяет организовать последовательный доступ к элементам массива.

Пример.

```
MOV   BX, 4          ; (BX) = 4
MOV   AX, MasW       ; (AX) = первое число в массиве (с номером 0)
MOV   AX, MasW[BX]   ; (AX) =  третье число в массиве (с номером 2)
                     ;  исполнительный адрес = MasW + 4
ADD   BX, 2          ;  (BX) = (BX) + 2 = 6
MOV       AX, MasW[BX]   ; (AX) =  четвертое число в массиве
```

Подавляющее большинство программ обработки массивов используют циклы и индексные регистры, поэтому перед разработкой программы приходится выполнять распределение регистров, закрепляя их специализацию в программе.

Пример. Найти сумму элементов массива из 50 слов.

Необходимые регистры:

- счетчик числа повторений - CX
- индексный регистр для доступа к элементам массива - BX
- регистр для накопления суммы элементов - AX

Основной фрагмент программы:

```
      MOV   AX, 0      ; обнуление регистра-суммы
      ;  XOR  AX, AX – другой способ обнуления
      MOV   CX, 50     ; установка счетчика цикла
      MOV   BX, 0      ; начальное значение индексного регистра
L:    ADD   AX, MasW[BX]     ;  (AX) = (AX) + (MasW + (BX))
      ADD   BX,  2           ;  (BX) = (BX) + 2
      LOOP  L                ;  повторяем цикл
```

Пример. Круговая перестановка элементов массива, содержащего 100 байтовых чисел: a[0] ↔ a[99], a[1] ↔ a[98], a[2] ↔ a[97] и т.д. Особенность данной программы состоит в том, что одновременно необходим доступ к начальным и конечным элементам массива, и поэтому приходится использовать сразу два индексных регистра: один изменяет свое значение от 0 до 49, второй – от 99 до 50.

Распределение регистров:

- счетчик числа повторений цикла – CX
- первый индексный регистр – SI
- второй индексный регистр - DI

Фрагмент программы:

```
        M   DB   100   DUP (?)     ;  исходный массив
        . . . . . . . . . . . . . . . . . . . . .
        MOV  CX, 50         ;  (CX) = 50 – число повторений
        MOV  SI, 0          ;  (SI) = 0
        MOV DI, 99          ;  (DI) = 99
L:      MOV   AH, M[SI]   ; (AH) = M[0], M[1], M[2], …, M[49]
        MOV   BH, M[DI]   ; (BH)  = M[99], M[98], M[97], …, M[50]
        MOV   M[SI], BH
        MOV  M[DI], AH
        INC   SI                ;   (SI) = (SI) + 1
        DEC  DI                 ;   (DI) = (DI) – 1
        LOOP   L
```

При необходимости, механизм индексных регистров можно применить для обработки двухмерных массивов. Допускается вычислять исполнительный адрес с использованием одновременно двух индексных регистров. При этом не все комбинации индексных регистров разрешены - можно одновременно использовать только пары BX и SI, BX и DI, BP и SI, BP и DI. Например:

```
MOV  AX, M2D [BX] [SI]    ; исполнит. адрес =  M2D + (BX) + (SI)
MOV  CX, M2D [BP] [DI]    ; исполнит. адрес =  M2D + (BP) + (DI)
```

Здесь регистр BX или BP можно использовать для отслеживания адреса строки, а SI или DI – для адреса элемента в этой строке.

Пример. Найти в двухмерном массиве M2D количество появлений заданного числа (например – 99). Распределение регистров:

- счетчик числа повторений цикла - CX
- искомое число появлений - AL
- индексный регистр для изменения строк - BX
- индексный регистр для изменения элементов в строке - SI

Фрагмент программы:

```
     M2D   DW   20   DUP  (10  DUP (?))    ;  20 строк по 10 чисел
     . . . . . . . . . . . . . . . . . . . . . . . . . . . .
     MOV  AL, 0     ; обнуление счетчика появлений
     MOV CX, 20     ; число повторений внешнего цикла
     MOV BX,  0     ; начальный индекс по строкам
L:   MOV  DX, CX    ; сохранить внешний счетчик в регистре DX
     MOV  CX, 10    ; внутренний счетчик по строке
     MOV  SI,  0    ; начальный индекс элемента в строке
L2:  CMP   M2D [BX] [SI], 99        ; сравнение M2D[i, j] с 99
     JNE   L1          ; обход совпадения
     INC  AL           ; увеличение счетчика появлений
L1:  ADD  SI, 2        ; увеличение индекса элемента в строке
     LOOP   L2         ; внутренний цикл
     MOV   CX, DX      ; восстановление внешнего счетчика
     ADD  BX, 20    ;  (BX) = (BX) + 2 * 10, т.е. переход к следующей строке
     LOOP   L          ; внешний цикл
```

Пример. Найти максимальное число в массиве из 100 байтов и его порядковый номер. Необходимые регистры:

- счетчик числа повторений цикла - CX
- текущее максимальное значение - AX
- номер текущего максимума - BX
- индексный регистр - SI

Фрагмент программы:

```
        Mas   DB   100   DUP (?)     ;  исходный массив
        . . . . . . . . . . . . . . . . . . .
        MOV   AX,   Mas   ;  (AX) = Mas [0] – начальный максимум
        MOV   BX,  0          ;  (BX) = 0   - индекс первого элемента
        MOV   SI,  1          ;  (SI) = 1  - шаг изменения индекса-адреса
        MOV   CX,  99         ;   счетчик цикла
L1:   CMP   Mas [SI],  AX        ;  сравнение Mas [i] с регистром AX
        JLE    L                  ;  обход, если Mas [i] ≤ (AX)
        MOV   AX,  Mas [SI]      ;  (AX) = новый максимум
        MOV   BX,   SI        ;  (BX) = индекс нового максимума
L:      ADD    SI, 1          ;  увеличение индекса
        LOOP   L1
```

Косвенная адресация возникает, когда в команде задается не адрес операнда, а место, где этот адрес хранится - регистр или область памяти. Выполнение такой команды происходит в 2 шага:

- выборка из регистра или памяти хранящегося там адреса
- обращение по этому адресу для получения операнда.

Эта адресация – основа механизма адресных указателей и динамических структур данных. Отличие косвенной адресации от обычной можно показать на следующих двух внешне похожих командах:

```
MOV   BX, AX      ; (BX) = значение из регистра AX
MOV   BX, [AX]    ; (BX) = значение из памяти по адресу в регистре AX
```

Этот способ эффективен, когда при написании программы адрес операнда в памяти не известен, известно лишь, что этот адрес находится в

определенном регистре. Тогда можно динамически во время работы программы загружать этот регистр необходимыми адресами и с помощью косвенной адресации работать с разными областями памяти.

При использовании косвенной адресации возникает ряд проблем. Первая связана с необходимостью занесения в регистр или область памяти значения-адреса. Для этого в систему команд введена специальная команда с именем LEA (от Load Effective Address, т.е. Загрузить Исполнительный Адрес). Команда имеет формат

LEA регистр, операнд

и она вычисляет исполнительный адрес операнда и записывает его в регистр.

Рассмотрим пример.

```
X1    DW   1, 2, 3   ; объявление трех чисел-слов со значением 1, 2 и 3
      . . . . . .
      MOV  BX,  X1      ;  в BX – значение по адресу X1, т.е. число 1
      LEA   BX, X1      ; в BX – адрес области с именем X1
      MOV  SI,  2       ; загрузка индексного регистра SI значением 2
      LEA  AX,  X1[SI]  ; в AX – адрес области X1 + 2, т.е. адрес числа 2
```

Из последнего примера видно, что косвенная адресация может комбинироваться с индексной.

Вторая проблема связана с тем, что адрес, который хранится в регистре, никак не определяет размер адресуемой области (байт? слово? двойное слово?). Поэтому часто в командах с косвенной адресацией приходится уточнять размер адресуемой области. Для этого есть специальный оператор указания типа PTR:

<тип> PTR <выражение>

Например:

```
BYTE    PTR  0    ;  взять 0 как байт (т.е. 00 h)
WORD    PTR  0    ;  взять 0 как слово (т.е. 00 00 h)
DWORD   PTR  0    ;  взять 0 как двойное слово (т.е. 00 00 00 00 h)
BYTE    PTR  A1   ;  имя A1 адресует байт
```

```
WORD    PTR  A2   ;  имя A2 адресует слово
BYTE    PTR  500  ;  ОШИБКА!!!
```

С помощью оператора PTR можно не только уточнять тип операнда, но и изменять его. Для правильного использования косвенной адресации надо с помощью оператора PTR указывать тип косвенно-адресуемой памяти:

```
MOV  BYTE  PTR  [BX],  0     ; обнуление байта по адресу (BX)
MOV  WORD  PTR  [BX],  0     ; обнуление слова по адресу (BX)
```

Практические задания к теме №9.

Обработка массивов с помощью индексной адресации

Задание 1. Вычисление суммы элементов в 10-элементом числовом массиве

Задание 2. Вычисление суммы произведений элементов двух числовых 10-элементных массивов

Задание 3. Нахождение максимального числа и его номера в 10-элементом числовом массиве

Задание 4. Круговая перестановка элементов массива, содержащего 10 чисел

Задание 5. Задать текстовую строку длиной 15-20 символов и подсчитать число появлений в ней какого-либо символа

Задание 6. Задать текстовую строку, содержащую несколько слов и заменить все пробелы символом *

Задание 7. Определить строку, содержащую 20-30 букв и цифр и подсчитать число цифр (код нуля 30h, код девятки 39h)

Задание 8. Определить небольшой числовой двухмерный массив (3 строки по 5 элементов) и найти количество появлений заданного числа, используя вложенные циклы

Выполнить все задания с использованием косвенной адресации