Строковые команды

Строковые команды

- Назначение - эффективное выполнение простых операций над массивами

- Могут выполнять действия над двумя операндами в памяти!

Общее для всех строковых команд

- Источник – DS:[(E)SI] (сегмент можно заменить)
- Приемник – ES:[(E)DI] (сегмент нельзя заменить)

Общее для всех строковых команд

- Каждая команда имеет 4 варианта для данных разного размера:

размер, байт	**суффикс**
авто (где это возможно)	**нет**
1	**b**
2	**w**
4	**d**

Общее для всех строковых команд

- Синтаксис для автоопределения размера операндов

```
команда переменная1, переменная2
```

- Размер переменных должен совпадать!
- Переменные – только для определения размера операндов
- Всегда используются данные по адресу DS:[ESI] и ES:[EDI]!
- Можно указать любые другие переменные такого же размера (но это снизит читаемость)

Общее для всех строковых команд

После выполнения:

- если используется источник, то

  ESI := ESI + шаг

- если используется приемник, то

  EDI := EDI + шаг

- где шаг = { размер данных, если DF = 0
  - размер данных, если DF = 1

  (DF – Direction Flag)

Префиксы строковых команд

- Записываются перед строковой командой:

```
префиксСС строковая_команда
```

- Это аналогично

```
метка:
    строковая_команда
    loopСС метка
```

- но эффективней

Префиксы строковых команд

- Условия **CC**:
  - те же, что и у **loop**
  - используются с командами поиска

- Команда повторяется пока

  (E)CX≠0 и (выполняется условие **CC**)

Префиксы строковых команд

префикс	дополнит. условие	команды
rep	нет	movs, lods, stos ins, outs
repe, **repz**	ZF = 0	cmps, scas
repne **repnz**	ZF != 0	

Копирование

- movs (movsb, movsw, movsd) – копирует источник в приемник

- Пример:

```
a    dw 10 dup (?)
b    dw 10 dup (?)
…
     mov cx, 10
     lea si, a
     lea di, b
     rep movsw      ; или rep movs a,b
```

Поиск

- **scas** (**scas**b, …) – ищет значение в приемнике (эталон – al, ax, eax)

- Пример – найти дину строки, заканчивающейся нулем

Пример


```
st  db "string", 0
…
    mov ecx, 1000     ; макс. дина
    lea edi, st
    xor al, al        ; сравниваем с 0
    repne scas st     ; пока не 0
        ; edi – адрес первого 0 в st
        ; длина = edi – st - 1
    lea eax, st+1
    sub edi, eax ; edi = длина
```

Сравнение

- **cmps** (**cmps**b, ...) – сравнивает источник и приемник

- Пример – найти первый с конца несовпадающий элемент

Пример

```
a    dw 10 dup (?)
b    dw 10 dup (?)
…

     std          ; DF=1 – обратное направление
     mov ecx, 10
     lea esi, a[9]
     lea edi, b[9]
     repe cmps a, b

     je метка ; если все совпадают

; ECX – номер отличного элемента, начиная с 0
; [ESI+2], [EDI+2] – адрес элементов
```

Загрузка и сохранение

- **lods** (**lods**b, lodsw, lodsd) – копирует источник в AL, AX или EAX
- **stos** (**stos**b, stosw, stosd) – сохраняет AL, AX или EAX в приемник

- Применение 1. Без префиксов внутри циклов для более сложной обработки данных

Пример

- Расширить массив байт до массива слов

```
a    db 10 dup (?)
b    dw 10 dup (?)
…
     mov ecx, 10
     lea esi, a
     lea edi, b
fori:
     lods a
     cbw
     stos b
     loop fori
```

Загрузка и сохранение

- Применение 2 – заполнение памяти константой

```
a    dd 100 dup (?)
…

     mov ecx, 100
     lea edi, a
     xor eax, eax ; заполняем нулем
     rep stos a
```