عنوان درس:

# برنامه سازی سیستم

## جلسه ۶: عملیات بر روی رشته اطلاعات و جدول

مدرس:
اعظم کبوتری کهنه شهری

# دستورات اساسی رشته ای

| شرح | دستورالعمل دو بایتی (کلمه ای) | دستورالعمل یک بایتی |
|---|---|---|
| کپی یک رشته در رشته دیگر | MOVSW | MOVSB |
| مقایسه یک رشته با رشته دیگر | CMPSW | CMPSB |
| جستجوی یک مقدار در رشته | SCASW | SCASB |
| کپی در ثبات AL یا AX | LODSW | LODSB |
| کپی از ثبات AL یا AX | STOSW | STOSB |

- اگر طول رشته فرد است باید از دستورات یک بایتی و اگر طول رشته زوج است می توان از دستورات دو بایتی استفاده کرد.
- تمام دستورات فوق دوعملوندی هستند، اما چون این عملوندها ثابت و مشخص هستند لذا در جلوی دستورات عملوندها نوشته نمی شوند.

# عملوندهای پیش فرض دستورات رشته ای

| دستورات | اپرند اول (مقصد) | اپرند دوم (مبدأ) |
|---|---|---|
| MOVSW و MOVSB | افست رشته در DI | افست رشته در SI |
| CMPSW و CMPSB | افست رشته در DI | افست رشته در SI |
| SCASW و SCASB | افست رشته در DI | AL یا AX |
| LODSW و LODSB | AL یا AX | افست رشته در SI |
| STOSW و STOSB | افست رشته در DI | AL یا AX |

- AL با دستورات یک بایتی و AX با دستورات دو بایتی استفاده می شود.
- عملوندهای MOVS و CMPS دو رشته و عملوندهای SCAS، LODS و STOS یک رشته و یک مقدار است.
- قبل از استفاده از دستورات رشته ای، باید ثبات های مورد استفاده مقداردهی شوند.

# تنظیم ثبات ها قبل از استفاده از دستورات رشته ای

- برقراری ارتباط بین هر سگمنت و ثبات مربوطه:

```
assume ss:stackSG,ds:dataSG,cs:codeSG,es:dataSG
```

- مقداردهی به ثبات های ds و es (در برنامه های exe اولین دستورات روال main هستند):

```
mov ax,dataSG
mov ds,ax
mov es,ax
```

- تعریف رشته ها در سگمنت داده:

```
str1 DB “string1”
str1 DB “string1”
```

- قرار دادن آدرس رشته ها در SI و DI:

```
LEA SI,str1
LEA DI,str2
```

# نحوه تکرار دستورات رشته ای

- هر دستور رشته ای تنها روی یک بایت/یک کلمه عمل می کند.
- جهت انجام عملیات روی تمام کاراکترهای رشته هر دستورالعمل باید تکرار شود.
- تکرار دستورات با استفاده از دستورات repeat انجام می شود:
  - REP
  - REPE یا REPZ
  - REPNE یا REPNZ
- در دستورات فوق، تعداد تکرار در ثبات CX قرار می گیرد و با هربار اجرای دستور یک واحد از CX کم می شود و وقتی CX=0 شود تکرار خاتمه می یابد.

# نحوه تکرار دستورات رشته ای–ادامه

| دستور | شرط خاتمه تکرار | شرط خاتمه تکرار |
|---|---|---|
| REP | CX=0 | - |
| REPE/REPZ | CX=0 | ZF=0 |
| REPNE/REPNZ | CX=0 | ZF=1 |

- عمل مقایسه روی ZF اثر گذاشته و اگر نتیجه مقایسه مساوی باشد ZF=1 می شود.
- لذا در دستورات رشته ای مقایسه ای مانند CMP و SCAS اغلب از REPE و REPNE (یا معادل آنها) برای تکرار استفاده می شود.
- مثال: جابه جایی ۱۰۰ بایت:

```
MOV CX,100
REP MOVSB
```

- برای مقداردهی CX می توان از اپراتور LENGHTH به صورت نمونه زیر استفاده کرد:

```
str1 DB 50 Dup(?)
…
MOV CX,LENGHTH str1
```

# تعیین جهت عملیات رشته ای

- به هنگام اجرای دستورات رشته ای ثبات های SI و DI که به کاراکتر فعلی در رشته ها اشاره می کنند، به طور خودکار تغییر می کنند تا به کاراکتر بعدی اشاره کنند.
- پرچم DF (Direction Flag) جهت عملیا ت رشته ای را مشخص می کند:
  - اگر DF=0 باشد (مقدار پیش فرض): دستورالعمل های رشته ای به ثبات های DI و یا SI یک/دو واحد اضافه می کنند.
  - اگر DF=1 باشد: دستورالعمل های رشته ای از ثبات های DI و یا SI یک/دو واحد کم می کنند.
- مقداردهی پرچم DF:

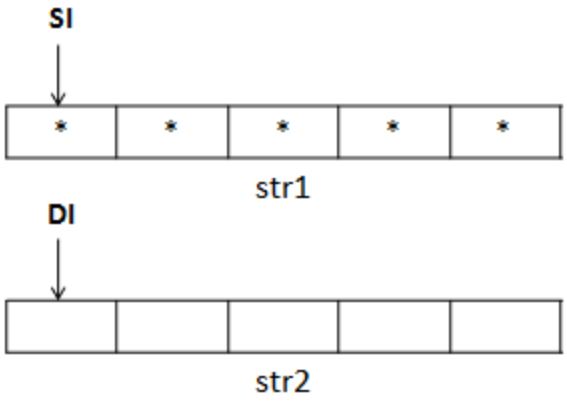
```
STD         ;DF=1
CLD         ;DF=0
```

# دستورات MOVSB و MOVSW

- دستور MOVSB/MOVSW از حافظه ای که SI به آنجا اشاره می کند یک/دو بایت خوانده و در حافظه ای که DI به آنجا اشاره دارد، کپی می کند. همچنین مقدار ثبات های SI و DI را یک/دو واحد افزایش (DF=0) یا کاهش (DF=1) می دهد.

- مثال: دستوراتی که رشته str1 را در رشته str2 کپی می کند:

```
str1 db 5 dup('*')
str2 db 5 dup(' ')
     …
lea si,str1
lea di,str2
mov cx,length str1
cld
rep movsb
```



# دستورات LODSB و LODSW

- دستور LODSB از حافظه ای که SI به آنجا اشاره می کند یک بایت خوانده و در AL کپی می کند. همچنین مقدار ثبات های SI را یک واحد افزایش (DF=0) یا کاهش (DF=1) می دهد.

- دستور LODSW از حافظه ای که SI به آنجا اشاره می کند دو بایت خوانده و در AX کپی می کند. همچنین مقدار ثبات های SI را دو واحد افزایش (DF=0) یا کاهش (DF=1) می دهد.

- کاربرد این دو دستور زمانی است که بخواهیم رشته را خوانده و پردازشی روی کاراکترهای آن انجام دهیم.

# دستورات LODSB و LODSW–ادامه

- مثال: دستوراتی که رشته ای را از صفحه کلید بگیرد و تعداد کاراکترهای خالی آن را شمارش کرده و در ثبات BX ذخیره کند:

```
str   db 20 dup(‘ ’)
              ...
```

دستورات خواندن رشته از صفحه کلید و قراردادن آن در محل حافظه str و قرادادن طول رشته وارد شده در cx

```
      lea si,str
      cld
      mov bx,0
l1:   lodsb
      cmp al,” “
      jne l2
      inc bx
l2:   loop l1          ;(cx=0)خاتمه حلقه
```

- قبل از LODS از REP استفاده نمی شود.

# دستورات STOSB و STOSW

- دستور STOSB مقدار مورد نظر در AL را در حافظه ای که DI به آنجا اشاره می کند کپی می کند. همچنین مقدار ثبات های DI را یک واحد افزایش (DF=0) یا کاهش (DF=1) می دهد.

- دستور STOSW مقدار مورد نظر در AX را در حافظه ای که DI به آنجا اشاره می کند کپی می کند. همچنین مقدار ثبات های DI را دو واحد افزایش (DF=0) یا کاهش (DF=1) می دهد.

- کاربرد این دو دستور زمانی است که بخواهیم مقدار مشخصی را در رشته قرار دهیم.

# دستورات STOSB و STOSW–ادامه

- مثال: دستوراتی که رشته ای را از صفحه کلید بگیرد، تمام کاراکترهای آن را با * پر کند و رشته را دوباره چاپ کند:

```
str  db 20 dup(‘ ’)
          ...
```

دستورات خواندن رشته از صفحه کلید و قراردادن آن در محل حافظه str و قرادادن طول رشته وارد شده در cx

```
lea di,str
cld
mov al,’*’
rep stosb
```

دستورات نمایش رشته روی مانیتور

# دستورات CMPSB و CMPSW

- این دستورات برای مقایسه رشته ها استفاده می شوند.
- برای بررسی تساوی رشته ها از REPE و برای بررسی عدم تساوی رشته ها از REPNE استفاده می کنیم.
- مثال: دستوراتی که دو رشته str1 و str2 را با هم مقایسه کرده درصورت تساوی دو رشته مقدار صفر و گرنه مقدار یک را در AX قرار دهد.

```
  mov ax,0
  lea si,str1
  lea di,str2
  mov cx, length str1
  cld
  repe cmpsb
  je l1
  mov ax,1
l1:
```

# دستورات SCASB و SCASW

- دستور SCASB شبیه دستور مقایسه است و مقدار موجود در AL را در داخل رشته ای که DI به آن اشاره می کند جستجو می کند.

- دستور SCASW مقدار موجود در AX را در داخل رشته ای که DI به آن اشاره می کند جستجو می کند.

- اگر بخواهیم جستجو تا زمانی که شرط تساوی برقرار است ادامه یابد از REPE استفاده می کنیم.

- اگر بخواهیم جستجو تا زمانی که شرط تساوی برقرار نیست ادامه یابد از REPNE استفاده می کنیم.

# دستورات SCASB و SCASW–ادامه

- مثال: دستوراتی که یک رشته را از ورودی بگیرد، در داخل رشته دنبال کاراکتر P بگردد و درصورت پیداشدن آن را با Q عوض کند:

```
str   db 20 dup(‘ ’)
            ...
```

دستورات خواندن رشته از صفحه کلید و قراردادن آن در محل حافظه str و قرادادن طول رشته وارد شده در cx

```
      lea di,str
      cld
      mov al,’P’
      repne scasb
      jne l1
      dec di
      mov byte ptr[di],’Q’
l1:   دستورات نمایش رشته روی مانیتور
```

# دستور XLAT

- XLAT(Translate a byte in AL)
  - در اکثر برنامه ها به جدولی نیاز هست که یک سری اطلاعات در آن ذخیره شود.
  - جهت دستیابی به این جدول از دستور XLAT استفاده می شود.
  - قبل از استفاده از این دستور باید آدرس جدول که در سگمنت داده تعریف شده در ثبات BX و شماره بایت مورد نظر جدول در ثبات AL قرار گیرد.
  - دستور XLAT که بدون پارامتر است AL را به عنوان افست جدول به کار برده و محتوای آن خانه جدول را در AL ذخیره می کند.
  - این دستور روی پرچم ها تأثیری ندارد.

# دستور XLAT–ادامه

- مثال: فرض کنید می خواهیم مقدار $y=x^2$ وقتی x بین صفر تا ۹ است بدست آوریم، مثلا اگر $x=5$ را دادیم برنامه مقدار ۲۵ را به ما بدهد:

```
table1 db 0,1,4,9,16,25,36,49,64,81
…
mov bx,offset table1
mov al,05
xlat
```

# دستور XLAT–ادامه

- مثال: فرض کنید می خواهیم اعداد صفر تا ۱۵ را به معادل هگزادسیمالشان تبدیل کنیم:

```
table1 db ‘0’,’1’,’2’,’3’,’4’,’5’,’6’,’7’,’8’,’9’
       db ‘A’,’B’,’C’,’D’,’A’
dec_val db ?
hex_val db ?

…
mov bx,offset table1
mov al, dec_val
xlat
mov dec_val, al
```

# بدست آوردن آدرس کامل (LDS و LES)

- دستور LDS افست اپرند دوم خود را در ثباتی که در اپرند اول مشخص کرده ایم کپی کرده و آدرس سگمنت داده را در DS قرار می دهد.

```
LDS  DX,VAL
```

- دستور LES افست اپرند دوم خود را در ثباتی که در اپرند اول مشخص کرده ایم کپی کرده و آدرس سگمنت داده را در DS قرار می دهد.

```
LES  DX,VAL
```

- این دو دستور روی پرچم ها اثری ندارند.

# تمرین

۱. دستوراتی بنویسید که یک رشته را از ورودی بخواند، حروف کوچک آن را به حروف بزرگ تبدیل کند و سپس رشته را دوباره چاپ کند. (با LODSB و STOSB)

۲. دستوراتی بنویسید که دو رشته هم طول را از ورودی بخواند، اگر مساوی هستند پیام "Yes" وگرنه پیام "No" را چاپ کند. (با CMPSB)

۳. دستوراتی بنویسید که ALIrEZA را پردازش کرده حرف r آن را با R جایگزین کرده و نمایش دهد. (با SCASB)

۴. دستوراتی بنویسید که یک رشته را از ورودی بخواند و تعداد حروف بزرگ آن را شمارش کرده و نمایش دهد. (با LODSB)

۵. دستوراتی بنویسید که به کمک دستور XLAT حروف کوچک را به حروف بزرگ تبدیل کند.