مجموعة التعليمات Instruction Set

للـ 8086 أنماط للتعليمات6

1. تعليمات نقل البيانات Data Transfer Instructions
2. التعليمات الحسابية Arithmetic Instructions
3. التعليمات المنطقية Logical Instructions
4. تعليمات معالجة السلاسل String manipulation Instructions
5. تعليمات التحكم بالعملية Process Control Instructions
6. تعليمات التحكم بالنقل Control Transfer Instructions

جَامعة المَنارة MANARA UNIVERSITY

4. تعليمات معالجة السلاسل
4. String Manipulation Instructions

- السلسلة هي تتالي من البايتات أو الكلمات String : Sequence of bytes or words
- تضم مجموعة تعليمات الـ8086 مجموعة تعليمات نقل ومقارنة ومسح وتحميل وتخزين السلاسل instruction set includes instruction for string movement, comparison, scan, load and store.
- وتستخدم بادئة REP لإعادة تنفيذ تعليمة السلسلة REP instruction prefix : used to repeat execution of string instructions
- تعليمةالسلسلة تنتهي بـS أو SB أو SW. String instructions end with S or SB or SW.
  S represents string, SB string byte and SW string word.
- تخزن إزاحة العنوان الفعال إمّا في المسجل SI للمصدر أو في DI للهدف. Offset or effective address of the source operand is stored in SI register and that of the destination operand is stored in DI register.
- ويتم تحديث قيمة كل من SI و DI آلياً وفقاً لقيمة العلم DF Depending on the status of DF, SI and DI registers are automatically updated.
- DF = 0 ⇒ SI and DI are incremented by 1 for byte and 2 for word.
- DF = 1 ⇒ SI and DI are decremented by 1 for byte and 2 for word.

REP	
REPZ/ REPE **(Repeat CMPS or SCAS until ZF = 0)**	**While CX ≠ 0 and ZF = 1, repeat execution of string instruction and (CX) ← (CX) – 1**
REPNZ/ REPNE **(Repeat CMPS or SCAS until ZF = 1)**	**While CX ≠ 0 and ZF = 0, repeat execution of string instruction and (CX) ← (CX) - 1**

Mnemonics: REP, MOVS, CMPS, SCAS, LODS, STOS

تعريف السلسلة $-string
ويمكن حساب طول السلسلة str_len

repe/repz

```
while (ECX != 0)
      execute the string instruction;
      CX := CX–1;
      if (ZF = 0)
      then
      exit loop
      end if
end while
```

repne/repnz

```
while (ECX != 0)
      execute the string instruction;
      CX := CX–1;
      if (ZF = 1)
      then
      exit loop
end i
```

rep

```
while (ECX != 0)
   execute the string instruction;
   ECX := ECX–1;
end while
```

Movs تعليمة نقل سلسلة

movs dest_string,source_string
movsb
movsw
movsd

علم الاتجاه DF

std set direction flag (DF = 1) توضع علم الاتجاه
cld clear direction flag (DF = 0) تصفير علم الاتجاه

MOVS	
MOVSB	**MA = (DS) x 16_{10} + (SI)** **MA_E = (ES) x 16_{10} + (DI)** **(MA_E) ← (MA)** **If DF = 0, then (DI) ← (DI) + 1; (SI) ← (SI) + 1** **If DF = 1, then (DI) ← (DI) - 1; (SI) ← (SI) - 1**
MOVSW	**MA = (DS) x 16_{10} + (SI)** **MA_E = (ES) x 16_{10} + (DI)** **(MA_E ; MA_E + 1) ← (MA; MA + 1)** **If DF = 0, then (DI) ← (DI) + 2; (SI) ← (SI) + 2** **If DF = 1, then (DI) ← (DI) - 2; (SI) ← (SI) - 2**

movsb — move a byte string
ES:EDI := (DS:ESI) ; copy a byte
if (DF = 0) ; forward direction
then
    ESI := ESI+1
    EDI := EDI+1
else ; backward direction
    ESI := ESI–1
    EDI := EDI–1
end if
Flags affected: none

مثال اكتب برنامج لنقل سلسلة من 4 بايت
من موقع DS:SI = 0000:2000
الى الموقع ES:DI = 0000:2400

Example:

```
CLD ;          clear the direction flag to auto increment SI and DI
MOV AX, 0000H ;
MOV DS, AX ;          initialize data segment register to 0
MOV ES, AX ;          initialize extra segment register to 0
MOV SI, 2000H ;       Load the offset of the string1 in SI
MOV DI, 2400H ;       Load the offset of the string2 in DI
MOV CX, 04H ;          load length of the string in CX
REP MOVSB ;           decrement CX and MOVSB until CX will be 0
```

مثال اكتب برنامج لنقل سلسلة من موقع string1 الى موقع string2

```
string1 db 'The original string',
string2 resb 80
```

```
DATA
string1 db 'The original string',0
strLen EQU $ - string1
.UDATA
string2 resb 80
.CODE
.STARTUP
        mov AX,DS ;          set up ES
        mov ES,AX ;          to the data segment
        mov ECX,strLen ;     strLen includes NULL
        mov ESI,string1
        mov EDI,string2
        cld ; forward direction
rep movsb
```

تعليمة مفارنة سلسلتين تتالف من بايت او من كلمة word

Compare two string byte or string word

CMPS	
CMPSB	**MA = (DS) x 16_{10} + (SI)** **MA_E = (ES) x 16_{10} + (DI)** **Modify flags ← (MA) - (MA_E)** **If (MA) > (MA_E), then CF = 0; ZF = 0; SF = 0** **If (MA) < (MA_E), then CF = 1; ZF = 0; SF = 1**
CMPSW	**If (MA) = (MA_E), then CF = 0; ZF = 1; SF = 0** **<u>For byte operation</u>** **If DF = 0, then (DI) ← (DI) + 1; (SI) ← (SI) + 1** **If DF = 1, then (DI) ← (DI) - 1; (SI) ← (SI) - 1** **<u>For word operation</u>** **If DF = 0, then (DI) ← (DI) + 2; (SI) ← (SI) + 2** **If DF = 1, then (DI) ← (DI) - 2; (SI) ← (SI) - 2**

cmpsb — compare two byte strings
Compare the two bytes at DS:ESI and ES:EDI and set flags
if (DF = 0) ; forward direction
then
    ESI := ESI+1
    EDI := EDI+1
else ; backward direction
    ESI := ESI−1
    EDI := EDI−1
end i

مثال اكتب برنامج لمقارنة سلسلتين وايجاد موقع الخلاف بين السلسلتين

السلسلة الأولى

string1 db 'abcdfghi',

السلسلة الثانية

string2 db 'abcdefgh',

```
DATA
string1 db 'abcdfghi',0
strLen EQU $ - string1
string2 db 'abcdefgh',0
.CODE
        .STARTUP
        mov AX,DS ; set up ES
        mov ES,AX ; to the data segment
        mov ECX,strLen
        mov ESI,string1
        mov EDI,string2
        cld ; forward direction
        repe cmpsb
```

leaves ESI pointing to g in string1 and EDI to f in string2.
Therefore, adding

```
        dec ESI
        dec EDI
```

Scan (compare) a string byte or word with accumulator
تعليمة مسح سلسلة من بايت أو word تخزن في المجمع

SCAS	
SCASB	**MA_E = (ES) x 16_{10} + (DI)** **Modify flags ← (AL) - (MA_E)** **If (AL) > (MA_E), then CF = 0; ZF = 0; SF = 0** **If (AL) < (MA_E), then CF = 1; ZF = 0; SF = 1** **If (AL) = (MA_E), then CF = 0; ZF = 1; SF = 0** **If DF = 0, then (DI) ← (DI) + 1** **If DF = 1, then (DI) ← (DI) – 1**
SCASW	**MA_E = (ES) x 16_{10} + (DI)** **Modify flags ← (AL) - (MA_E)** **If (AX) > (MA_E ; MA_E + 1), then CF = 0; ZF = 0; SF = 0** **If (AX) < (MA_E ; MA_E + 1), then CF = 1; ZF = 0; SF = 1** **If (AX) = (MA_E ; MA_E + 1), then CF = 0; ZF = 1; SF = 0** **If DF = 0, then (DI) ← (DI) + 2** **If DF = 1, then (DI) ← (DI) – 2**

scasb — scan a byte string
Compare AL to the byte at ES:EDI and set flags
if (DF = 0) ; forward direction
then
    EDI := EDI+1
else ; backward direction
    EDI := EDI−1
end if
Flags affected: As per cmp instruction

مثال اكتب برنامج لمسح سلسلة وايجاد الحرف 'e' وموقعه في السلسلة

This program leaves DI pointing to e in string1.

```
DATA
string1 db 'abcdefgh',0
strLen EQU $ - string1
.CODE
        .STARTUP
        mov AX,DS ;         set up ES
        mov ES,AX ;         to the data segment
        mov CX,strLen
        mov DI,string
        mov AL,'e' ;        character to be searched
        cld ;               forward direction
        repne scasb;
        dec DI;
```

. The following example can be used to skip the initial blanks

```
DATA
string1 db ' abc',0
strLen EQU $ - string1
.CODE
        .STARTUP
        mov AX,DS ;             set up ES
        mov ES,AX ;             to the data segment
        mov ECX,strLen
        mov EDI,string1
        mov AL,' ' ;            character to be searched
        cld ;                   forward direction
        repe scasb
        dec EDI
```

Load string byte in to AL or string word in to AX

LODS	
LODSB	MA = (DS) x 16_{10} + (SI) (AL) ← (MA) If DF = 0, then (SI) ← (SI) + 1 If DF = 1, then (SI) ← (SI) – 1
LODSW	MA = (DS) x 16_{10} + (SI) (AX) ← (MA ; MA + 1) If DF = 0, then (SI) ← (SI) + 2 If DF = 1, then (SI) ← (SI) – 2

```
lodsb — load a byte string
AL := (DS:ESI);      copy a byte
if (DF = 0) ;        forward direction
      then
      ESI := ESI+1
else ;               backward direction
      ESI := ESI−1
end if
Flags affected: non
```

Store byte from AL or word from AX in to string

STOS	
STOSB	$MA_E = (ES) \times 16_{10} + (DI)$ $(MA_E) \leftarrow (AL)$ If DF = 0, then (DI) ← (DI) + 1 If DF = 1, then (DI) ← (DI) – 1
STOSW	$MA_E = (ES) \times 16_{10} + (DI)$ $(MA_E ; MA_E + 1) \leftarrow (AX)$ If DF = 0, then (DI) ← (DI) + 2 If DF = 1, then (DI) ← (DI) – 2

مثال اكتب برنامج لتخزين1- في مصفوفة من 100 عنصر

```
UDATA
array1 resw 100
.CODE
        .STARTUP
        mov AX,DS ;         set up ES
        mov ES,AX ;         to the data segment
        mov ECX,100
        mov EDI,array1
        mov AX,-1
        cld ;               forward direction
        rep stosw
```