

Roll No: 11
Name: Soham Desai
Xavier ID : 202003021
Date: 23/3/22

EXPERIMENT 7

Aim: Program to move set of numbers from one memory block to another.

LO: 4

LO STATEMENT: Develop the assembly level programming using 8086 loop instruction set

Software and Hardware Requirements: TASM Software

Theory:

1. MOV Instruction

The MOV instruction is the most important command in the 8086 because it moves data from one location to another. It also has the widest variety of parameters; so the assembler programmer can use MOV effectively, the rest of the commands are easier to understand. MOV copies the data in the source to the destination. The data can be either a byte or a word. Sometimes this has to be explicitly stated when the assembler cannot determine from the operands whether a byte or word is being referenced.

Syntax:

Move Destination, Source

Example:

MOV Ax, Bx

2. MOVSB Instruction

MOVSB instruction will perform all the actions in repeat unit loop. The MOVSB instruction will copy a byte from the location pointed to by the Direct Index Register. It will then automatically increment SI to point to the next source location. If you add a special prefix called the repeat prefix in front of the MOVSB instruction, the MOVSB instruction will be repeated and CX decremented until CX is counted down to zero.

Syntax:

Move Destination, Source

Example:

MOVSB Ax, Bx

3. LEA Instruction

LEA is Used to load the address of operand into the provided register. LES – Used to load ES register and other provided register from the memory. The lea instruction places the address specified by its first operand into the register specified by its second operand. Note, the contents of the memory location are not loaded, only the effective address is computed and placed into the register.

Roll No: 11
Name: Soham Desai
Xavier ID : 202003021
Date: 23/3/22

Code:

assume cs:code,ds:code,es:extra

data segment

blk1 db 10H,20H,30H,40H,50H

data ends

extra segment

blk2 db 05 dup(?)

extra ends

code segment

start:

mov Ax,data

mov Ds,Ax

mov Ax,extra

mov es,Ax

lea si,blk1

lea di,blk2

mov Cx,05H

std

back:movsb

loop back

mov AH,4CH

int 21H

code ends

end start

Roll No: 11
 Name: Soham Desai
 Xavier ID : 202003021
 Date: 23/3/22

Output:

For ds:

The screenshot shows the CPU-80486 emulator interface. The assembly code window displays the following instructions:

```

48AF:0000 B8AD48      mov     ax,48AD
48AF:0003 8ED8          mov     ds,ax
48AF:0005 B8AE48      mov     ax,48AE
48AF:0008 8EC0          mov     es,ax
48AF:000A BE0000      mov     si,0000
48AF:000D BF0000      mov     di,0000
48AF:0010 B90500      mov     cx,0005
48AF:0013 FD          std
48AF:0014 A4          movsb
48AF:0015 E2FD      loop    0014
48AF:0017 B44C          mov     ah,4C
48AF:0019 CD21      int     21
48AF:001B 0000          add     [bx+si],al
  
```

The register window on the right shows the following values:

ax	0192	c	1
bx	0018	z	0
cx	092D	s	1
dx	F66C	o	0
si	10AB	p	0
di	3256	a	0
bp	0100	i	1
sp	0106	d	1
ds	2110		
es	114E		
ss	0192		
cs	0000		
ip	0000		

The memory window at the bottom shows the following data:

```

48AD:0000 10 20 30 40 50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
48AD:0008 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
48AD:0010 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
48AD:0018 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  
```

For es:

The screenshot shows the CPU-80486 emulator interface. The assembly code window displays the following instructions:

```

48AF:0000 B8AD48      mov     ax,48AD
48AF:0003 8ED8          mov     ds,ax
48AF:0005 B8AE48      mov     ax,48AE
48AF:0008 8EC0          mov     es,ax
48AF:000A BE0000      mov     si,0000
48AF:000D BF0000      mov     di,0000
48AF:0010 B90500      mov     cx,0005
48AF:0013 FD          std
48AF:0014 A4          movsb
48AF:0015 E2FD      loop    0014
48AF:0017 B44C          mov     ah,4C
48AF:0019 CD21      int     21
48AF:001B 0000          add     [bx+si],al
  
```

The register window on the right shows the following values:

ax	0192	c	1
bx	0018	z	0
cx	092D	s	1
dx	F66C	o	0
si	10AB	p	0
di	3256	a	0
bp	0100	i	1
sp	0106	d	1
ds	2110		
es	114E		
ss	0192		
cs	0000		
ip	0000		

The memory window at the bottom shows the following data:

```

489D:0000 CD 20 FF 9F 00 EA FF FF = f 0
489D:0008 AD DE E0 01 C5 15 AA 01 i 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
489D:0010 C5 15 89 02 20 10 92 01 + 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
489D:0018 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
  
```

Conclusion: From this experiment we learn how to move numbers from one memory block to another.