

EXPERIMENT 07

AIM :-

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

LO: (LO4): Program to move set of numbers from one memory block to another.

SOFTWARE :- Tasm Software

Theory :-

Instructions used in this program

MOV

The MOV instruction is the most important command in the 8086 because it moves data from one location to another.

Syntax: Mov source, destination

Example: Mov Ax,1234H

LEA

Used to load the address of operand into the provided register. LES – Used to load ES register and other provided register from the memory.

MOVSB

This instruction copies a byte or a word from a location in the data segment [DS:SI] to a location in the extra segment [ES:DI]. The offset in the data segment for the source is to be stored in the SI register and the offset for the destination in the extra segment is to be stored in the DI register.

LOOP

The loop instructions cause the microprocessor to execute a series of instructions repeatedly. Basically, the LOOP instructions are short jump instructions on a condition i.e., when the condition satisfies a short jump is taken whose destination or target address is in the range of -128 bytes to +127 bytes from the instruction address after LOOP instruction.

INTERRUPT

int 21h means, call the interrupt handler 0x21 which is the DOS Function dispatcher. the "mov ah,01h" is setting AH with 0x01, which is the Keyboard Input with Echo handler in the interrupt.

Syntax: int 21H

Example: int 21H

Code :-

```
Assume cs:code,ds:data,es:extra
data segment
blk1 db 20H, 40H, 60H, 80H, 90H
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
back:movsb
loop back
mov ah,4CH
int 21H
code ends
end start
```

Output :-

The image displays two screenshots of the MPLAB IDE interface, showing the execution of assembly code for a CPU 80486. The interface includes a menu bar (File, Edit, View, Run, Breakpoints, Data, Options, Window, Help) and a status bar (READY).

First Screenshot:

- Assembly Code:**

```

cs:0000 B8AD48 mov ax,48AD
cs:0003 8ED8 mov ds,ax
cs:0005 B8AE48 mov ax,48AE
cs:0008 8EC0 mov es,ax
cs:000A BE0000 mov si,0000
cs:000D BF0000 mov di,0000
cs:0010 B90500 mov cx,0005
cs:0013 A4 movsb
cs:0014 E2FD loop 0013
cs:0016 B44C mov ah,4C
cs:0018 CD21 int 21
cs:001A 0000 add [bx+si],al
cs:001C 0000 add [bx+si],al

```
- Registers:**
 - ax: 0000, bx: 0000, cx: 0000, dx: 0000, si: 0000, di: 0000, bp: 0000, sp: 0000, ds: 489D, es: 489D, ss: 48AC, cs: 48AF, ip: 0000
 - Flags: c=0, z=0, s=0, o=0, p=0, a=0, i=1, d=0
- Memory:**
 - ds:0000 CD 20 FF 9F 00 EA FF FF = f 0
 - ds:0008 AD DE E0 01 C5 15 AA 01
 - ds:0010 C5 15 89 02 20 10 92 01
 - ds:0018 01 03 01 00 02 FF FF FF
 - ss:0002 6474
 - ss:0000 0000

Second Screenshot:

- Assembly Code:**

```

cs:0000 B8AD48 mov ax,48AD
cs:0003 8ED8 mov ds,ax
cs:0005 B8AE48 mov ax,48AE
cs:0008 8EC0 mov es,ax
cs:000A BE0000 mov si,0000
cs:000D BF0000 mov di,0000
cs:0010 B90500 mov cx,0005
cs:0013 A4 movsb
cs:0014 E2FD loop 0013
cs:0016 B44C mov ah,4C
cs:0018 CD21 int 21
cs:001A 0000 add [bx+si],al
cs:001C 0000 add [bx+si],al

```
- Registers:**
 - ax: 48AE, bx: 0000, cx: 0000, dx: 0000, si: 0005, di: 0005, bp: 0000, sp: 0000, ds: 48AD, es: 48AE, ss: 48AC, cs: 48AF, ip: 0016
 - Flags: c=0, z=0, s=0, o=0, p=0, a=0, i=1, d=0
- Memory:**
 - es:0000 30 40 50 60 70 00 00 00 00 P p
 - es:0008 00 00 00 00 00 00 00 00
 - es:0010 B8 AD 48 8E D8 B8 AE 48
 - es:0018 8E C0 BE 00 00 BF 00 00
 - ss:0002 6474
 - ss:0000 0000

The screenshot displays the MPLAB IDE interface with the following components:

- Menu Bar:** File, Edit, View, Run, Breakpoints, Data, Options, Window, Help.
- Assembly Code Window (CPU 80486):**

Address	Hex	Assembly	Register/Value	Flag/Status
cs:0000	B8AD48	mov ax,48AD	ax 48AE	c=0
cs:0003	8ED8	mov ds,ax	bx 0000	z=0
cs:0005	B8AE48	mov ax,48AE	cx 0000	s=0
cs:0008	8EC0	mov es,ax	dx 0000	o=0
cs:000A	BE0000	mov si,0000	si 0000	p=0
cs:000D	BF0000	mov di,0000	di 0000	a=0
cs:0010	B90500	mov cx,0005	bp 0000	i=1
cs:0013	A4	movsb	sp 0000	d=0
cs:0014	E2FD	loop 0013	ds 48AD	
cs:0016	B44C	mov ah,4C	es 48AE	
cs:0018	CD21	int 21	ss 48AC	
cs:001A	0000	add [bx+si],al	cs 48AF	
cs:001C	0000	add [bx+si],al	ip 0010	
- Memory Window:**

Address	Hex	Value
ds:0000	30 40 50 60 70	00 00 00 00 00
ds:0008	00 00 00 00 00	00 00 00 00 00
ds:0010	00 00 00 00 00	00 00 00 00 00
ds:0018	00 00 00 00 00	00 00 00 00 00
- Register Window:**

Register	Value
ax	48AE
bx	0000
cx	0000
dx	0000
si	0000
di	0000
bp	0000
sp	0000
ds	48AD
es	48AE
ss	48AC
cs	48AF
ip	0010
- Status Bar:** F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

Conclusion : We learned to build a program on microprocessor using arithmetic and logical instructions and performed BCD addition on 16-bit values.