NAME : Soham Manoj Pawar **Class : SE-IT**
ROLL NO :71 **MPL LAB**

EXPERIMENT 09

AIM :-

Program to search for a given number

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.

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

JMP

NAME : Soham Manoj Pawar **Class : SE-IT**
ROLL NO :71 **MPL LAB**

The basic instruction that transfers control to another point in the program is JMP.

Syntax: JMP label

To declare a label in your program, just type its name and add ":" to the end, label can be any character combination but it cannot start with a number, for example here are 3 legal label definitions:

label1:

label2:

a:

<u>JNZ</u>

The jnz (or jne) instruction is a conditional jump that follows a test.

It jumps to the specified location if the Zero Flag (ZF) is cleared (0).

jnz is commonly used to explicitly test for something not being equal to zero whereas jne is commonly found after a cmp instruction.

Syntax

jnz location

jne location

Example

jnz short loc_4010E5 ; if function InternetReadFile des not return 0, jump to loc_4010E5

<u>JZ</u>

Checks the state of one or more of the status flags in the EFLAGS register (CF, OF, PF, SF, and ZF) and, if the flags are in the specified state (condition), performs a jump to the target instruction specified by the destination operand. A condition code (cc) is associated with each instruction to indicate the condition being tested for. If the condition is not satisfied, the jump is not performed and execution continues with the instruction following the Jcc instruction.

Syntax

jz location

je location

DEC

Subtracts 1 from the destination operand, while preserving the state of the CF flag. The destination operand can be a register or a memory location. This instruction allows a loop counter to be updated without disturbing the CF flag. (To perform a decrement operation that updates the CF flag, use a SUB instruction with an immediate operand of 1.)

INC

Increment Register or memory by 1.

Adds 1 to the destination operand, while preserving the state of the CF flag. The destination operand can be a register or a memory location. This instruction allows a loop counter to be updated without disturbing the CF flag. (Use a ADD instruction with an immediate operand of 1 to perform an increment operation that does updates the CF flag.)

This instruction can be used with a LOCK prefix to allow the instruction to be executed atomically.

Example: MOV AL, 4 INC AL ; AL = 5

CMP

The CMP instruction can be used to compare two 8-bit or two 16-bit numbers.

Whenever a compare operation is performed the result of such an operation reflects in one of the six status flags CF, AF, OF, PF, SF and ZF.

The CMP operation is also known as the subtraction method as it uses two`s complement for it.

Syntax

CMP destination, source

Example

```
CMP DX,     00  ; Compare the DX value with zero
JE  L7     ; If yes, then jump to label L7
.
.
L7: ...
```

Code :-

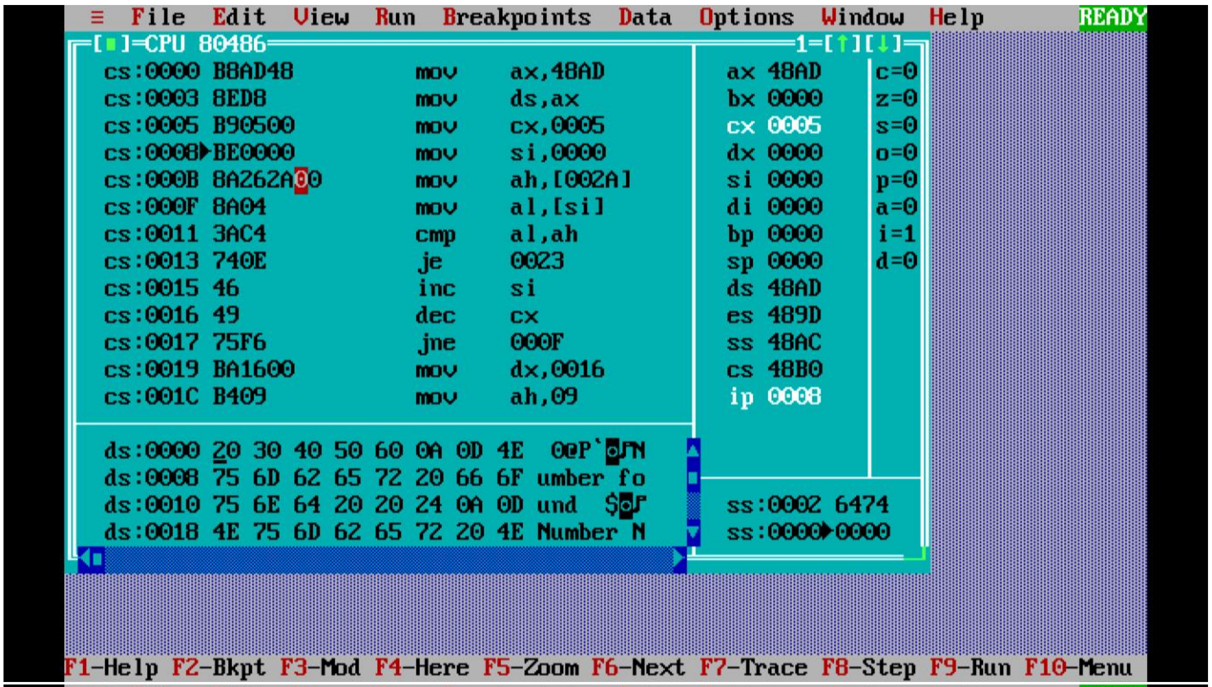
```
Assume CS:Code, DS:Data
Data Segment
a1 db 10H, 20H, 30H, 40H, 50H
msg1 db 0AH, 0DH, 'Number Found $'
msg2 db 0AH, 0DH, 'Number Absent $'
num db 03H
Data Ends
Code Segment
Start: MOV AX, Data
MOV DS, AX
MOV CX, 05H
LEA Si, a1
```

```
MOV AH, num

TO:MOV AL, [Si]

CMP AL, AH

JZ Me

INC Si

DEC CX

JNZ TO

LEA DX, msg2

MOV AH, 09H

INT 21H

JMP LAST

ME:LEA DX, msg1

MOV AH, 09H

INT 21H

LAST:MOV AH, 4CH

INT 21H

Code Ends

End Start.
```

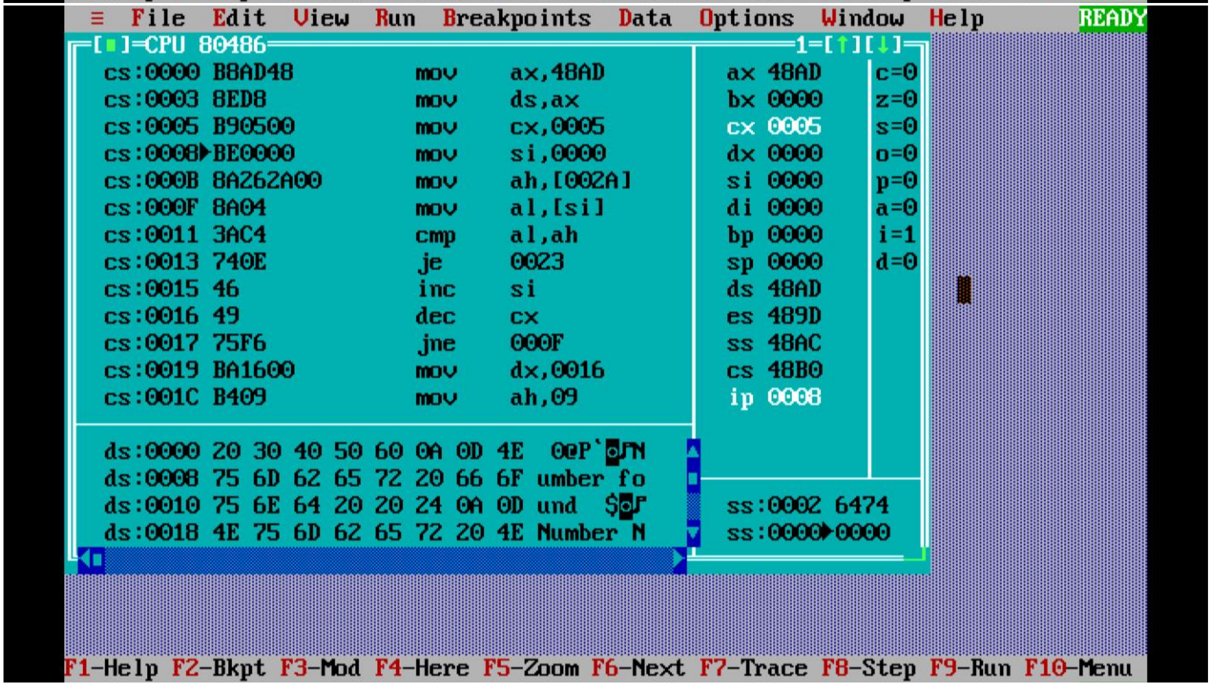
NAME : Soham Manoj Pawar **Class : SE-IT**

ROLL NO :71 **MPL LAB**

Output :-

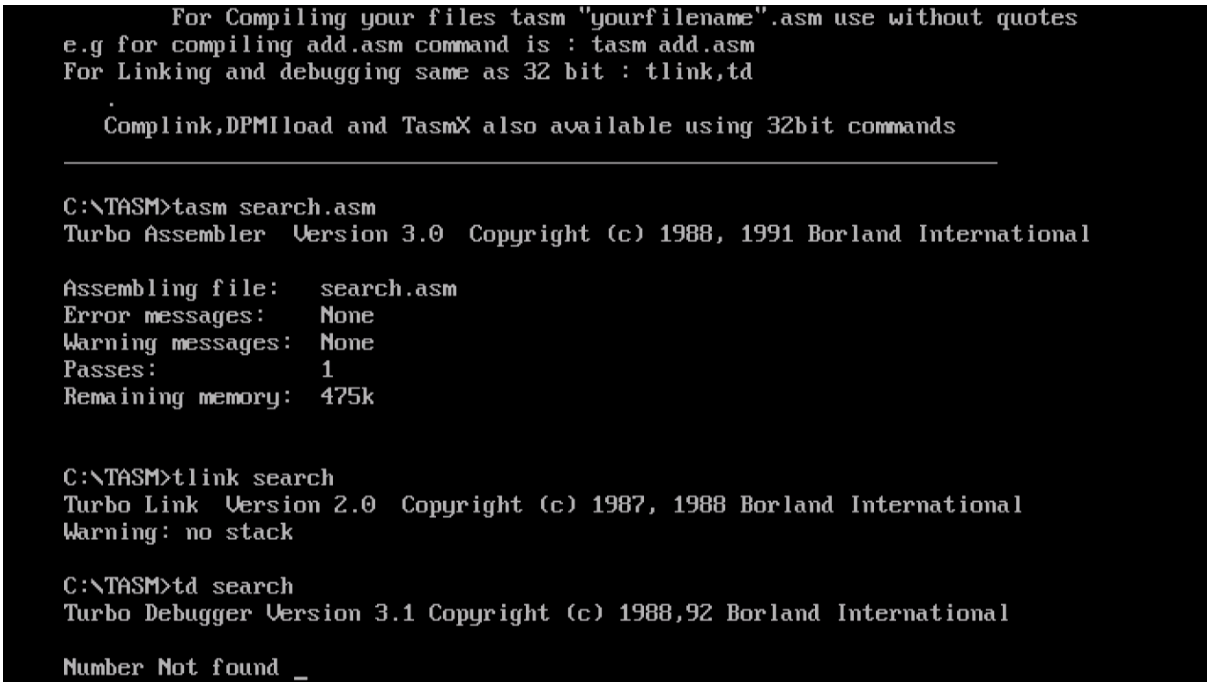


```
 ≡  File  Edit  View  Run  Breakpoints  Data  Options  Window  Help       READY
╔═[■]═CPU 80486═════════════════════════════════════1═[↑][↓]═╗
║ cs:0000 B8AD48         mov    ax,48AD      ║ ax 48AD  │c=0
║ cs:0003 8ED8           mov    ds,ax        ║ bx 0000  │z=0
║ cs:0005 B90500         mov    cx,0005      ║ cx 0005  │s=0
║ cs:0008►BE0000         mov    si,0000      ║ dx 0000  │o=0
║ cs:000B 8A262A00       mov    ah,[002A]    ║ si 0000  │p=0
║ cs:000F 8A04           mov    al,[si]      ║ di 0000  │a=0
║ cs:0011 3AC4           cmp    al,ah        ║ bp 0000  │i=1
║ cs:0013 740E           je     0023         ║ sp 0000  │d=0
║ cs:0015 46             inc    si           ║ ds 48AD  │
║ cs:0016 49             dec    cx           ║ es 489D  │
║ cs:0017 75F6           jne    000F         ║ ss 48AC  │
║ cs:0019 BA1600         mov    dx,0016      ║ cs 48B0  │
║ cs:001C B409           mov    ah,09        ║ ip 0008  │
║─────────────────────────────────────────────║          │
║ ds:0000 20 30 40 50 60 0A 0D 4E  0@P`◙♪N   ║──────────
║ ds:0008 75 6D 62 65 72 20 66 6F umber fo   ║
║ ds:0010 75 6E 64 20 20 24 0A 0D und  $◙♪   ║ ss:0002 6474
║ ds:0018 4E 75 6D 62 65 72 20 4E Number N   ║ ss:0000►0000
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
```



```
 ≡  File  Edit  View  Run  Breakpoints  Data  Options  Window  Help       READY
╔═[■]═CPU 80486═════════════════════════════════════1═[↑][↓]═╗
║ cs:0000 B8AD48         mov    ax,48AD      ║ ax 48AD  │c=0
║ cs:0003 8ED8           mov    ds,ax        ║ bx 0000  │z=0
║ cs:0005 B90500         mov    cx,0005      ║ cx 0005  │s=0
║ cs:0008►BE0000         mov    si,0000      ║ dx 0000  │o=0
║ cs:000B 8A262A00       mov    ah,[002A]    ║ si 0000  │p=0
║ cs:000F 8A04           mov    al,[si]      ║ di 0000  │a=0
║ cs:0011 3AC4           cmp    al,ah        ║ bp 0000  │i=1
║ cs:0013 740E           je     0023         ║ sp 0000  │d=0
║ cs:0015 46             inc    si           ║ ds 48AD  │
║ cs:0016 49             dec    cx           ║ es 489D  │
║ cs:0017 75F6           jne    000F         ║ ss 48AC  │
║ cs:0019 BA1600         mov    dx,0016      ║ cs 48B0  │
║ cs:001C B409           mov    ah,09        ║ ip 0008  │
║─────────────────────────────────────────────║          │
║ ds:0000 20 30 40 50 60 0A 0D 4E  0@P`◙♪N   ║──────────
║ ds:0008 75 6D 62 65 72 20 66 6F umber fo   ║
║ ds:0010 75 6E 64 20 20 24 0A 0D und  $◙♪   ║ ss:0002 6474
║ ds:0018 4E 75 6D 62 65 72 20 4E Number N   ║ ss:0000►0000
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
```

NAME : Soham Manoj Pawar **Class : SE-IT**
ROLL NO :71 **MPL LAB**



```
        For Compiling your files tasm "yourfilename".asm use without quotes
    e.g for compiling add.asm command is : tasm add.asm
    For Linking and debugging same as 32 bit : tlink,td
       .
       Complink,DPMIload and TasmX also available using 32bit commands
    ____________________________________________________________________

    C:\TASM>tasm search.asm
    Turbo Assembler  Version 3.0  Copyright (c) 1988, 1991 Borland International

    Assembling file:   search.asm
    Error messages:    None
    Warning messages:  None
    Passes:            1
    Remaining memory:  475k


    C:\TASM>tlink search
    Turbo Link  Version 2.0  Copyright (c) 1987, 1988 Borland International
    Warning: no stack

    C:\TASM>td search
    Turbo Debugger Version 3.1 Copyright (c) 1988,92 Borland International

    Number Not found _
```

<u>**Conclusion :**</u> We learned to built a program on microprocessor using arithmetic and logical instructions and performed BCD addition on 16-bit values.