Name: Soham Pawar
No.: 71
Batch: C
XIE ID: 202103006

Experiment No. 6

Aim: Program to evaluate given logical expression.

LO3: Build a program on a microprocessor using arithmetic & logical instruction set of 8086.

Hardware/Software Requirements: TASM Software

Theory:

Introduction to instructions used in this Experiment

1)MOV:

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 it the assembler programmer can use MOV effectively, the rest of the commands are easier to understand.

Syntax: MOV destination, source

The possible combinations of operands are as follows :

destination	source	example
register	register	mov ax,bx
register	immediate	mov ax,10h
register	memory	mov ax,es:[bx]
memory	immediate	mov aNumber,10h
memory	register	mov aDigit,ax

Name: Soham Pawar
No.: 71
Batch: C
XIE ID: 202103006

2) OR:

• It performs OR operation of Destination and Source. • Source can be immediate number, register memory

location.

• Destination can be register or memory location.

• Both operands cannot be memory locations at the same time.

• CF and OF become zero after the operation. • PF, SF and ZF are updated.

Syntax: OR Destination, Source:

Example:

MOV AL, 'A'; AL = 01000001b

OR AL, 00100000b; AL = 01100001b ('a')

RET

Syntax: OR destination, Source ;destination=destination ‘OR’ source

Name: Soham Pawar
No.: 71
Batch: C
XIE ID: 202103006

3) **AND:**

It performs AND operation of Destination and Source.

• Source can be immediate number, register or memory location.

• Destination can be register or memory location. . Both operands cannot be memory locations at the same time.

• CF and OF become zero after the operation. • PF, SF and ZF are updated.

Syntax: AND destination, Source ; destination=destination 'AND' source

Example:

MOV AL, 'a'; AL = 01100001b AND AL, 11011111b; AL = 01000001b ('A')

RET

4) **NOT:**

1. complements each bit Source to produce 1's complement of the specified operand. The operand can be a register or memory location.

• It does not affect the status flags.

Example:

MOV AL, 00011011b

NOT AL; AL = 11100100b

RET

5) **INT:**

INT is an assembly language instruction for x86 processors that generates a software interrupt. It takes the interrupt number formatted as a byte value.[1]
When written in assembly language, the instruction is written like this:

Syntax: When written in assembly language, the instruction is written like this:
Example: INT 13H will generate the 20th software interrupt (0x13 is the number 19 -- nineteen -- written in hexadecimal notation, and the count starts with 0), causing the function pointed to by the 20th vector in the interrupt table to be executed.Example: INT 21H

Given Logical Expression: Y = NOT [(A OR B) AND (C)]

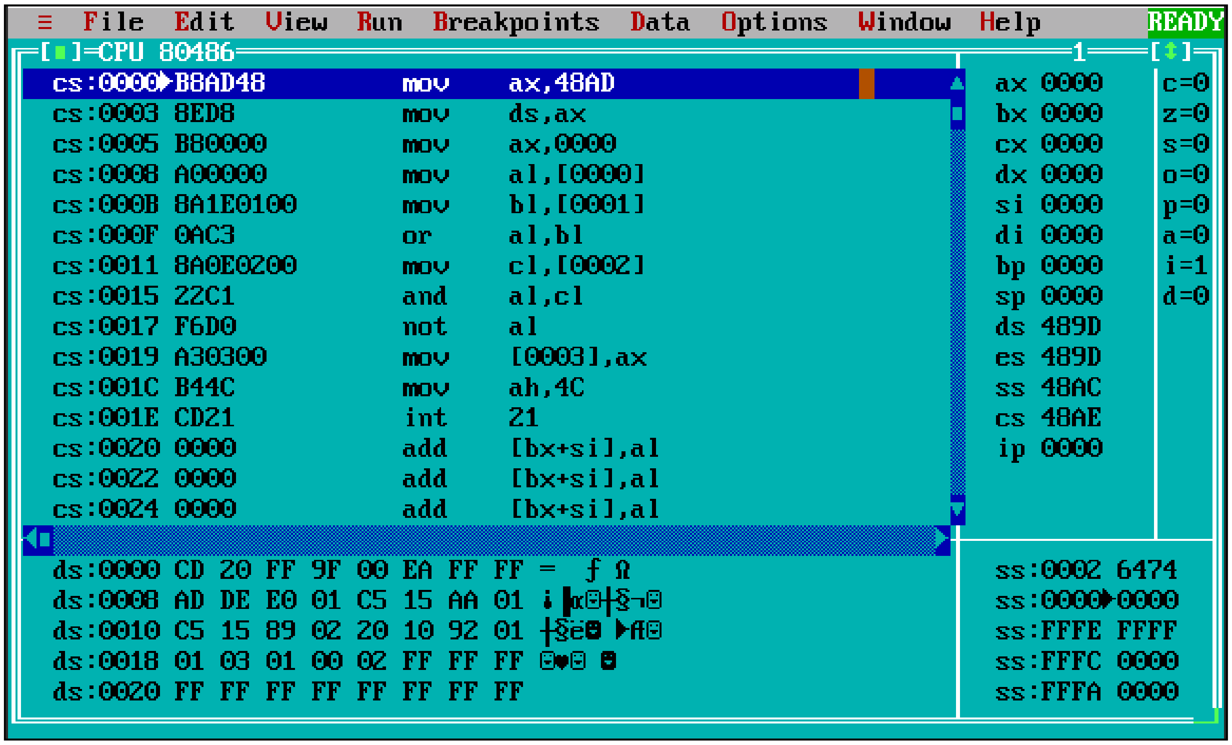
Name: Soham Pawar
No.: 71
Batch: C
XIE ID: 202103006

Code:

```
Assume CS:code,DS:data
  data segment
  A db 03H
  B db 08H
  C db 05H
  Y dw ?
  data ends
  code segment
Start:MOV AX,data
      MOV DS,AX
      MOV AX,00H
      MOV AL,A
      MOV BL,B
      OR AL,BL
      MOV CL,C
      AND AL,CL
      NOT  AL
      MOV Y,AX
      MOV AH,4CH
      INT 21H
      code ends
      end start
```

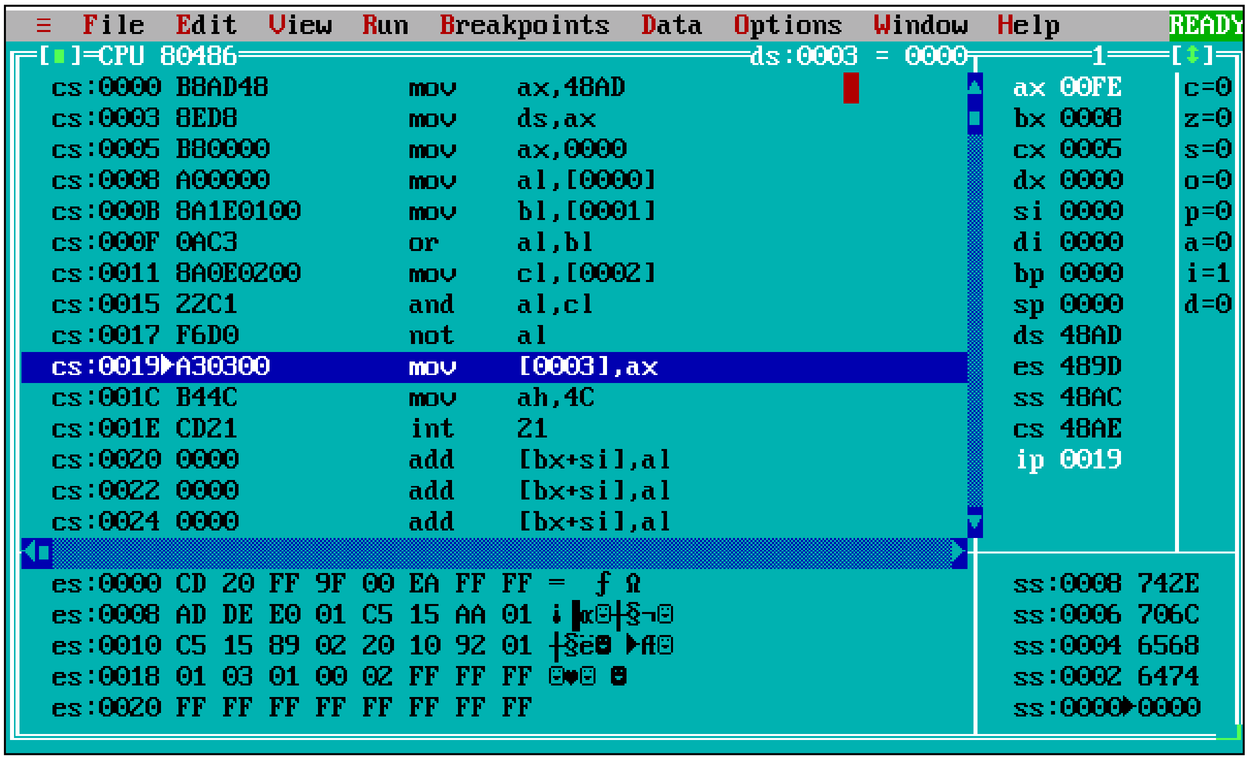
Name: Soham Pawar
No.: 71
Batch: C
XIE ID: 202103006

Output:



```
 ≡  File  Edit  View  Run  Breakpoints  Data  Options  Window  Help        READY
[■]=CPU 80486==========================================================1===[↕]=
 cs:0000►B8AD48         mov    ax,48AD                     ax 0000  c=0
 cs:0003 8ED8           mov    ds,ax                       bx 0000  z=0
 cs:0005 B80000         mov    ax,0000                     cx 0000  s=0
 cs:0008 A00000         mov    al,[0000]                   dx 0000  o=0
 cs:000B 8A1E0100       mov    bl,[0001]                   si 0000  p=0
 cs:000F 0AC3           or     al,bl                       di 0000  a=0
 cs:0011 8A0E0200       mov    cl,[0002]                   bp 0000  i=1
 cs:0015 22C1           and    al,cl                       sp 0000  d=0
 cs:0017 F6D0           not    al                          ds 489D
 cs:0019 A30300         mov    [0003],ax                   es 489D
 cs:001C B44C           mov    ah,4C                       ss 48AC
 cs:001E CD21           int    21                          cs 48AE
 cs:0020 0000           add    [bx+si],al                  ip 0000
 cs:0022 0000           add    [bx+si],al
 cs:0024 0000           add    [bx+si],al

 ds:0000 CD 20 FF 9F 00 EA FF FF =  ƒ Ω                  ss:0002 6474
 ds:0008 AD DE E0 01 C5 15 AA 01 ¡▐α☺┼§¬☺                ss:0000►0000
 ds:0010 C5 15 89 02 20 10 92 01 ┼§ë☻ ►Æ☺                ss:FFFE FFFF
 ds:0018 01 03 01 00 02 FF FF FF ☺♥☺ ☻                    ss:FFFC 0000
 ds:0020 FF FF FF FF FF FF FF FF                          ss:FFFA 0000
```



```
 ≡  File  Edit  View  Run  Breakpoints  Data  Options  Window  Help        READY
[■]=CPU 80486========================ds:0003 = 0000=====================1===[↕]=
 cs:0000 B8AD48         mov    ax,48AD                     ax 00FE  c=0
 cs:0003 8ED8           mov    ds,ax                       bx 0008  z=0
 cs:0005 B80000         mov    ax,0000                     cx 0005  s=0
 cs:0008 A00000         mov    al,[0000]                   dx 0000  o=0
 cs:000B 8A1E0100       mov    bl,[0001]                   si 0000  p=0
 cs:000F 0AC3           or     al,bl                       di 0000  a=0
 cs:0011 8A0E0200       mov    cl,[0002]                   bp 0000  i=1
 cs:0015 22C1           and    al,cl                       sp 0000  d=0
 cs:0017 F6D0           not    al                          ds 48AD
 cs:0019►A30300         mov    [0003],ax                   es 489D
 cs:001C B44C           mov    ah,4C                       ss 48AC
 cs:001E CD21           int    21                          cs 48AE
 cs:0020 0000           add    [bx+si],al                  ip 0019
 cs:0022 0000           add    [bx+si],al
 cs:0024 0000           add    [bx+si],al

 es:0000 CD 20 FF 9F 00 EA FF FF =  ƒ Ω                  ss:0008 742E
 es:0008 AD DE E0 01 C5 15 AA 01 ¡▐α☺┼§¬☺                ss:0006 706C
 es:0010 C5 15 89 02 20 10 92 01 ┼§ë☻ ►Æ☺                ss:0004 6568
 es:0018 01 03 01 00 02 FF FF FF ☺♥☺ ☻                    ss:0002 6474
 es:0020 FF FF FF FF FF FF FF FF                          ss:0000►0000
```

Name: Soham Pawar
No.: 71
Batch: C
XIE ID: 202103006

Conclusion: From this experiment we learned how Logical Instruction set and implemented the code of the given logical expression using assembly language in TASM software.