Microprocessors and Interfaces: 2021-22
Lecture 14
8086 Logical Instructions : Part-1

By Dr. Sanjay Vidhyadharan

DAA vs. AAA

Normally a BCD number can be coded with 4 Bits (9 being the maximum value in BCD notation).
In unpacked BCD 8 bits are used to represent a BCD digit , with the higher nibble being 0.
The unpacked BCD can easily be converted to ASCII as you just have to add 30H for a byte and 3030H for a word.
If you see the ASCII code, 0 to 9 is represented as 30 to 39, where 3 is the header.
In packed BCD only 4 bits are used to represent a BCD digit and hence a byte represents two BCD digits.

Decimal: 9 1 Binary : 0000 1001 0000 0001 Unpacked
Decimal: 9 1 Binary: 1001 0001 Packed

After addition of two BCD numbers AAA arranges it in unpacked BCD format , While ADD 3030H after that converts it to ASCII.

DAA vs. AAA

```
org 100h
MOV AL, 9H          : AX 0009
ADD AL, 1H          : AX 000A
DAA                 : AX 0010
ret
```

```
org 100h
MOV AL, 9H          : AX 0009
ADD AL, 1H          : AX 000A
AAA                 : AX 0100
ADD AX,3030H        : AX 3130
ret
```

XADD Instruction

XADD dest, source   Exchange (content of operands) and add

XADD BL, CL

After execution both the operand content will change.

Logical Instructions

The logic instructions include

- AND
- OR
- Exclusive-OR
- NOT
- NEG
- Shifts
- Rotates
- TEST (logical compare).

AND Destination, Source

ANDs each bit in the source with the corresponding bit in the destination

- CF and OF both become zero
- PF, SF and ZF affected
- AF undefined

AND

- AND clears bits of a binary number.
  - called **masking**
- AND uses any mode except memory-to-memory and segment register addressing.
- An ASCII number can be converted to BCD by using AND to mask off the leftmost four binary bit positions.

```
   x x x x  x x x x    Unknown number
•  0 0 0 0  1 1 1 1    Mask
 ___________________
   0 0 0 0  x x x x    Result
```

AND

- Ex1:

```
AND CX,[SI]   ; AND word in DS at offset [SI]with word in CX
                 register
              ; result in CX register
```

- Ex2: BX= 10110011 01011110

```
AND BX, 00FFH   ; Mask out upper 8 bits of BX
                  ; Result :BX=00000000 01011110
                  ; CF,OF,PF,SF,ZF=0
```

AND

```
Start: IN AL,80H; B7-Commercial Supply, B6-AC, B5-0 Lights/Fans
MOV BL,AL ;
AND AL,80H;
JNZ J1;
AND BL, BFH;
J1: OUT 81H,BL; (6-0 Pins controlled the loads)
MOV CL,FFH  ;
LOOP: DCR CL;
JNZ LOOP;
JMP Start
HLT
```

OR Destination, Source

ORs each bit in the source with the corresponding bit in the destination

CF and OF both become zero
PF, SF and ZF affected
AF undefined

```
    x x x x  x x x x   Unknown number
+   0 0 0 0  1 1 1 1   Mask
--------------------
    x x x x  1 1 1 1   Result
```

OR

- Ex: CX=00111101 10100101

 OR CX, FF00H ; Result in

 ; CX= 11111111 10100101

 ; The upper byte now all 1's and lower byte not changed.

 CF =OF =0

 PF=1, SF=1, ZF=0

Exclusive-OR

- If inputs of the Exclusive-OR function are both 0 or both 1, the output is 0; if the inputs are different, the output is 1.
- Exclusive-OR is sometimes called a comparator.
- XOR uses any addressing mode except segment register addressing.
- Exclusive-OR is useful if some bits of a register or memory location must be inverted
- A common use for the Exclusive-OR instruction is to clear a register to zero

Exclusive-OR

The operation of the Exclusive-OR function showing how bits of a number are inverted.

$$
\begin{array}{rl}
x\ x\ x\ x\ \ x\ x\ x\ x & \text{Unknown number} \\
\oplus\, 0\ 0\ 0\ 0\ \ 1\ 1\ 1\ 1 & \text{Mask} \\
\hline
x\ x\ x\ x\ \ \overline{x}\ \overline{x}\ \overline{x}\ \overline{x} & \text{Result}
\end{array}
$$

Exclusive-OR

- Ex1: BX= 00111101 01101001

  CX= 00000000 11111111

  XOR BX,CX ; Result: BX= 00111101 10010110

  ; bits in the lower byte are inverted.

Ex2: XOR WORDPTR [BX],00FFH

; XOR the immediate number 00FFH with word at offset [BX] in data segment .Result in memory location [BX].

TEST

TEST Destination, source

- **TEST** performs the AND operation.
  - only affects the condition of the flag register, which indicates the result of the test
  - functions the same manner as a CMP

- Usually the followed by either the JZ (jump if zero) or JNZ (jump if not zero) instruction.
- The destination operand is normally tested against immediate data.

TEST

TEST Destination, source

- Ex: AL= 01010001

```
TEST AL, 80H   ;AND immediate 80H with  AL to test if
                 MSB of AL is 1or 0.
               ; AL = 01010001 (unchanged)
               ; PF=0,SF=0,
               ;ZF=1 because ANDing produced 00H
```

NOT and NEG

- NOT and NEG can use any addressing mode except segment register addressing.
- The **NOT instruction inverts all bits** of a byte, word, or double word.
- **NEG two's complements a number**.
  - the arithmetic sign of a signed number changes from positive to negative or negative to positive

- The NOT function is considered logical
- NEG function is considered an arithmetic operation.

NOT and NEG

- NOT Destination ; Invert each bit of operand
- Ex: NOT BX ; complement the contents of BX register
- NOT BYTEPTR [SI]

- NEG Destination ; Form 2’s complement

Ex: NEG BX ; Replace the contents in BX with it’s 2’s complement

✓ This instruction forms the 2’s complement by subtracting the word or byte indicated in destination from zero.

Thankyou