30/12/2021

8086 Microprocessor

Divide Instruction

DIV Source

→ Division of 16 bit or 32 bit number by.
8 bit or ~~bit~~ 16 bit number.

① AL ← (AX ÷ Source (8-bit)) => 16 bit operand / 8-bit operand

AL ← Quotient
AX ← operand 16bit.
Source (8-bit) ← operand 8-bit.
operand 8-bit. → any 8-bit register. or 8-bit data from memory.

AH ← Remainder.

For eg:- ① DIV BL

AL ← (AX) ÷ (BL)

AL (quotient)
AH (remainder)

② DIV [0248H]

MA = BA + EA
= DS x 16_{10} + 0248

AL ← (AX) ÷ (MA)

AL (Quotient)
AH (remainder)

② AX ← (DX : AX) ÷ (Source (16-bit))

AX ← Quotient

(DX : AX) ← operand 32-bit

(Source (16-bit)) ← operand 16-bit. → 16-bit register.

DX ← remainder

32 bit operand / 16 bit operand.

For eg:- DIV BX

(Quotient) AX ← (DX : AX) ÷ (BX)

DX (remainder)

Note :- ① If the quotient is larger than the size of AL register (in case of 16-bit / 8-bit) or the size of AX register (in case of 32-bit / 16-bit) then a divide by zero condition occurs which is handeled by a special interrupt.

② If denominator in a division is zero than also divide by zero interrupt occurs.

→ DIV is used for unsigned division operation.

IDIV is used for signed division operation.

DAA (Decimal adjust after addition)

↳ No operand

↳ operation: AL ← (AL adjusting for BCD addition).

```
MOV AL, 54H
MOV BL, 26H
ADD AL, BL
DAA
```

AL = 54H.
BL = 26H.

```
  1
 54
 26
---
 80
```

```
0101 0100 → AL
0010 0110 → BL
+
---------
0111 1010   7AH.
```

not valid BCD.

```
+ 0000 0110
-----------
1000 0000
  8    0
-----------
```

* DAA will always be used after addition instruction only.

DAS (Decimal adjust after Subtraction)

DAS instruction after subtraction instruction to adjust the result for BCD value.

Logical Instructions of 8086

→ Basic logic operations : NOT, AND, OR, XOR

→ Bit by Bit shift operation such as
- SHL (Shift logical left)
- SHR (Shift logical right)
- SAL (Shift arithmetic left)
- SAR (Shift arithmetic right)

→ Rotate operations
- ROR (rotate right without carry)
- ROL (rotate left without carry)
- RCR (rotate right through carry)
- RCL (rotate left through carry)

→ Compare Instructions.

NOT destination (1's complement of destination)

↓

register

memory.

For eg: MOV AL, 0AH

NOT AL

(BE)

AL = 0000 1010

AL = 1111 0101

(AE)

AND destination, Source (Logical AND)

operation : destination operand ← (destination operand) AND (source operand)

bitwise logical AND operation.

Source operand → immediate data ; Register ; memory.

destination operand → Register ; memory.

For eg:

```
MOV BL, 0FH
MOV CL, F0H
AND CL, BL
```

BL = 0000 1111 ⌐ AND
CL = 1111 0000 ⌐
CL = 0000 0000

OR destination, source (Logical OR operation)

operation: destination operand ← (destination operand) OR (source operand)

(destination operand) → Register memory

(source operand) → Immediate data Register memory.

For eg:

```
MOV CX, 01F1H
OR CX, F001H
```

CX = 0000 0001 1111 0001
1111 0000 0000 0001

1111 0001 1111 0001

F 1 F 1

CX after OR instruction.

XOR destination, Source (Exclusive logical OR)

operation: (destination operand) ← (destination operand) ⊕ (source operand)

destination operand → Register memory

source operand → Immediate data Register memory.