

03/01/2022

8086 Microprocessor

CBW (Convert from byte to word)

→ This instruction does not have any operand.

→ This instruction copies the sign bit of AL register (MSB of AL register) into the 8-bits of AH register.

(Sign Extension).

What is Sign Extension?

→ if assume you have a signed binary number 0110 (4-bit).

↑
Sign bit

0000000110

Sign extended

→ if we have a -ve number

10110 (5-bit).

after sign extension.

1110110 (7-bit).

both have Same Value.

MOV AL, 45H.
CBW

AL = 0100 0101
↑
Sign bit.

AH = 0000 0000

BE of CBW
AL = 45H AH = ? \Rightarrow AX = ?? 45H.

AE of CBW
AL = 45H AH = 00H \Rightarrow AX = 0045H.

BX AL
16-bit 8-bit
 ↓ + ↓

8086 we cannot add them directly.

ADD BX, AL (NOT VALID because source
and destination operand
Size donot match).

CBW
ADD BX, AX (VALID).

024
+ 163

CWD (Convert from word to double word)
(16-bit) (32-bit)

* Convert a 16-bit value in AX register to a 32-bit value in DX:AX register combination by sign extension of sign bit of AX register.

TEST instruction

TEST operand1, operand2. (Syntax).

The TEST instruction performs bitwise logical AND operation of operand1 and operand2 without saving the result, just the status flags is modified based on result.

For eg:

MOV AL, 0AH

MOV BL, 0AH.

TEST AL, BL.

AL = 0000 1010

BL = 1010 0000

0000 1010

1010 0000

0000 0000

Logical
AND

result
will
be
not
stored

TEST will

simply

modify

status
flag

values
based on
result

Zero flag = 1

CF = 0

AF = 0

SF = 0

OF = 0

PF = 1

COMPARE INSTRUCTION

CMP operand1, operand2, (Syntax).

↓ ↘

Register
memory.

Register
memory.

→ This instruction performs subtraction of operand 2 from operand 1.

→ Result of subtraction is not stored anywhere but the status of zero flag and carry flag indicates comparison.

* When $\text{operand1} = \text{operand2} \rightarrow \text{Zero flag} = 1.$

operand 1 > operand 2 \rightarrow Zero flag = 0
Carry flag = 0

operand 1 < operand 2 \rightarrow Zero flag = 0
Carry flag = 1.

MASM \rightarrow assembler

Assembler Directives.

Jump Instruction \rightarrow unconditional
 \rightarrow conditional.

① unconditional jump.

JMP target address \rightarrow generally provided using label.

This instruction unconditionally transfer the control of execution to the specified target address using 8-bit or 16-bit displacement.

target address \rightarrow CS + displacement.

$$MA = BA + EA$$

\downarrow

CS

code
segment

\downarrow

displacement

\downarrow is given with JMP
as operand.

For eg:

MOV AL, 00H.

AGAIN: MOV BL, 01H.

ADD AL, BL

JMP AGAIN \rightarrow label.

to indicate
displacement.

\downarrow JMP AGAIN

$$MA = BA + EA$$
$$= CS + IP$$

AGAIN
 \downarrow

$$MA = BA + EA$$
$$= CS + IP \pm \text{displacement.}$$
$$= CS + IP - 3$$