Topic 4:
Instruction Set of 8086

Why study instruction set?

- Study of instruction set is required to write instructions in machine code that can be recognized and executed by a central processing unit.
- The knowledge will help you write lines of code or program by which you will be able to tell the processor, the sequence of operations to be performed.

What are we going to study?

Instruction Set

- Different types of instructions with examples
- How to use instructions in assembly language programming

Types of addressing mode in 8086

1. Data Copy/Transfer instructions
2. Arithmetic
3. Logical instructions
4. Shift & Rotate instructions
5. Branch instructions
6. Loop instructions
7. Machine Control instructions
8. Flag Manipulation instructions
9. String instructions

Will be covered later

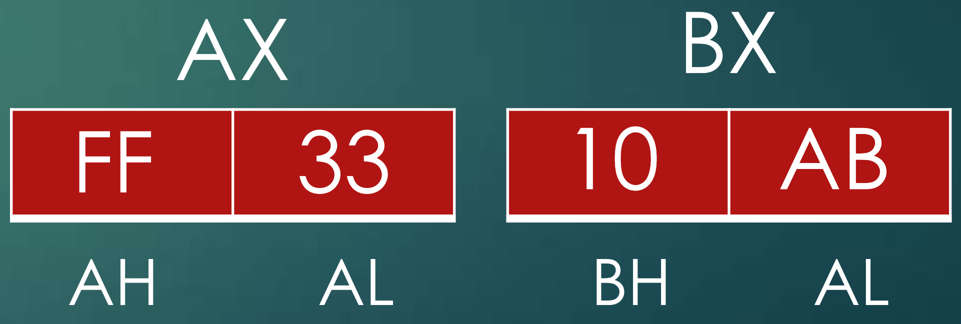
1: Data Copy/Transfer instructions

MOV Destination, Source

- There will be transfer of data from source to destination.

- **Source** can be register, memory location or immediate data.
- **Destination** can be register or memory operand.
- **Both Source and Destination cannot be memory location** or segment registers at the same time.

Eg: MOV AX,BX



AX

FF	33
AH	AL

BX

10	AB
BH	AL

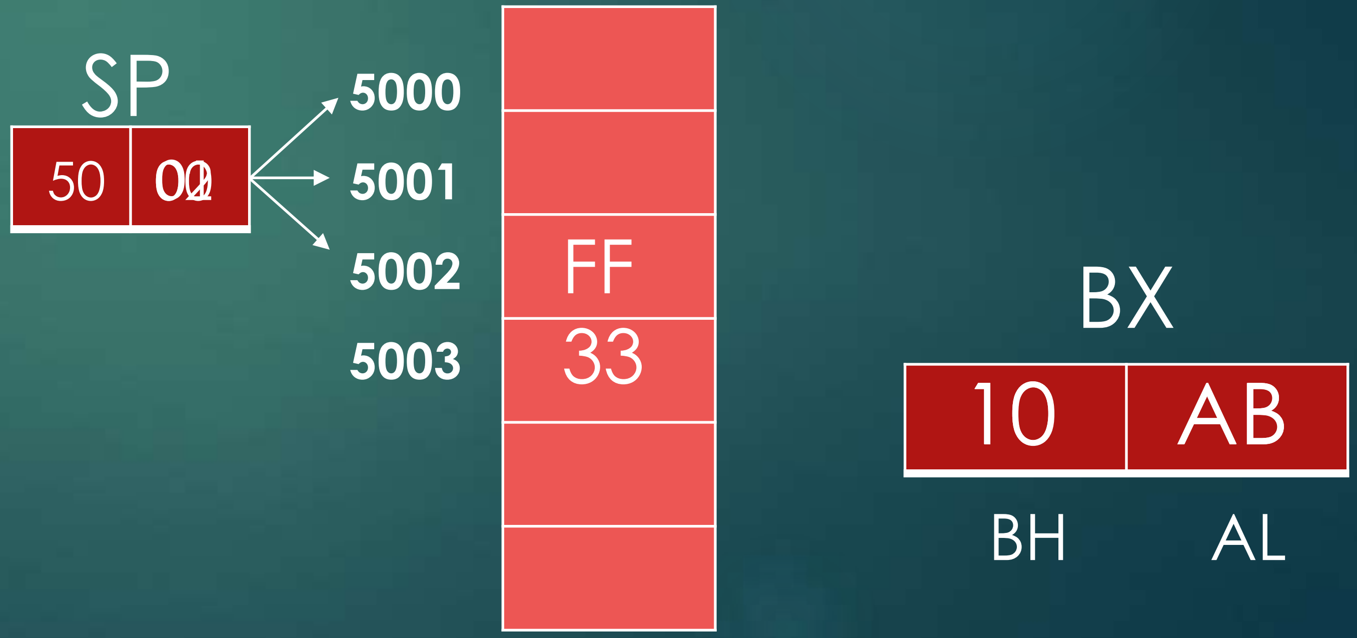
1: Data Copy/Transfer instructions

PUSH Source

- Pushes the 16 bit content specified by source in the instruction on to the stack

- **Pushing** operation decrements stack pointer stack pointer.
- **Stack pointer** is a 16-bit register, contains the address of the data item currently on top of the stack.

Eg: PUSH BX



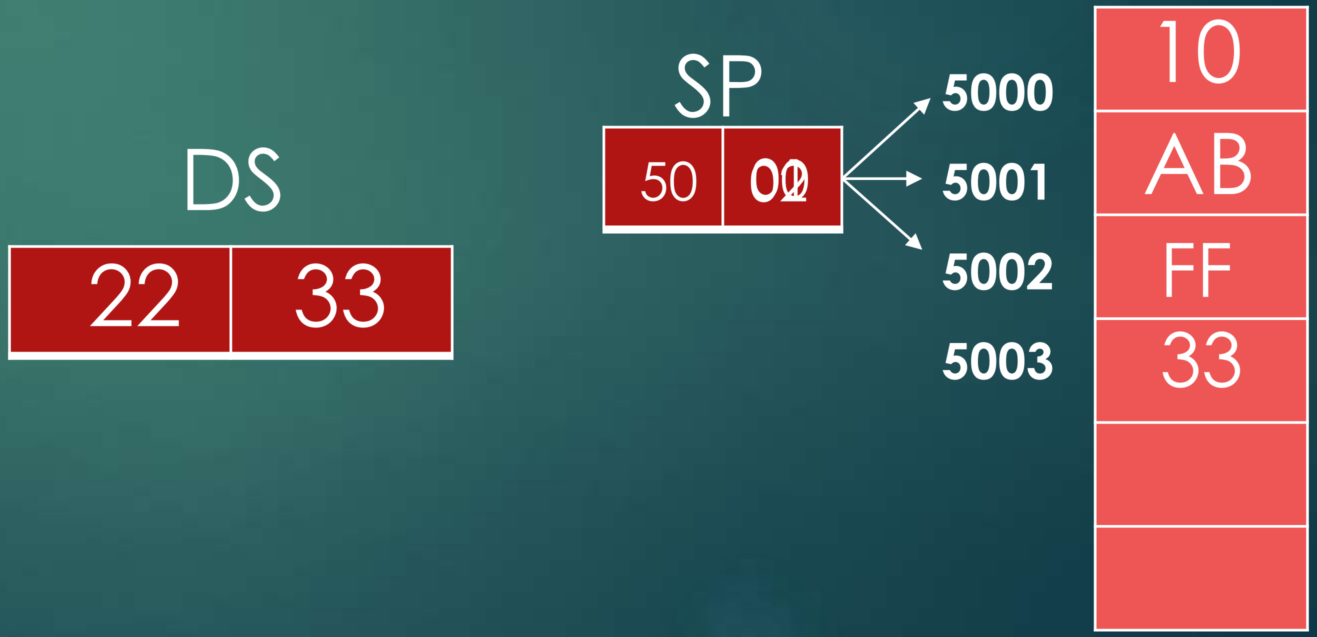
1: Data Copy/Transfer instructions

POP Destination

- Pops the 16 bit content from stack to destination specified in instruction.

Eg: POP DS

- **Popping** operation increments stack pointer stack pointer.



5000	10
5001	AB
5002	FF
5003	33

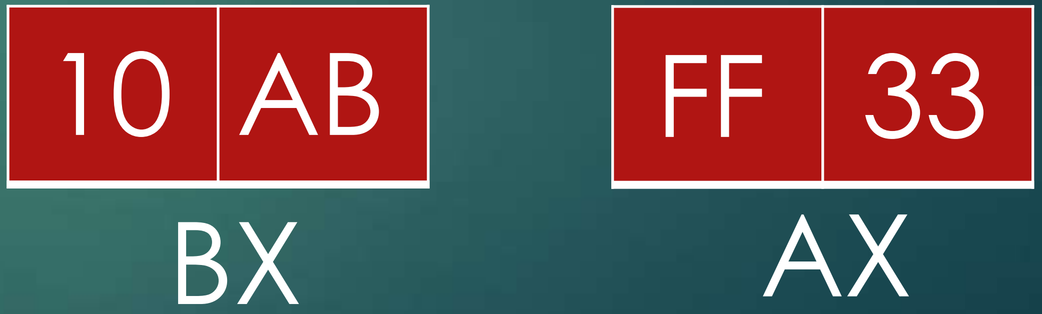
1: Data Copy/Transfer instructions

XCHG Destination, Source

- This instruction exchanges contents of Source with destination.

- It cannot exchange two memory locations directly.

Eg: XCHG BX,AX



10 AB

BX

FF 33

AX

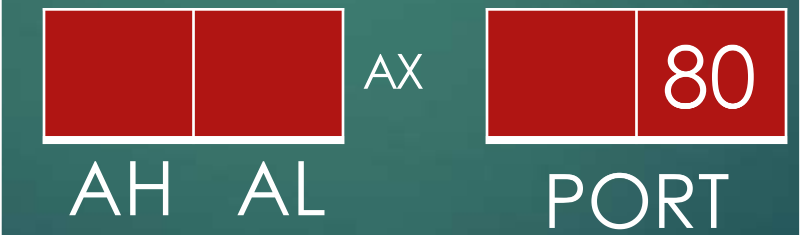
1: Data Copy/Transfer instructions

IN AL/AX, 8-bit/16-bit port address

- It copies data to accumulator from a port with 8-bit or 16-bit address.

- DX is the only register is allowed to carry port address.

Eg: IN AL, 80H



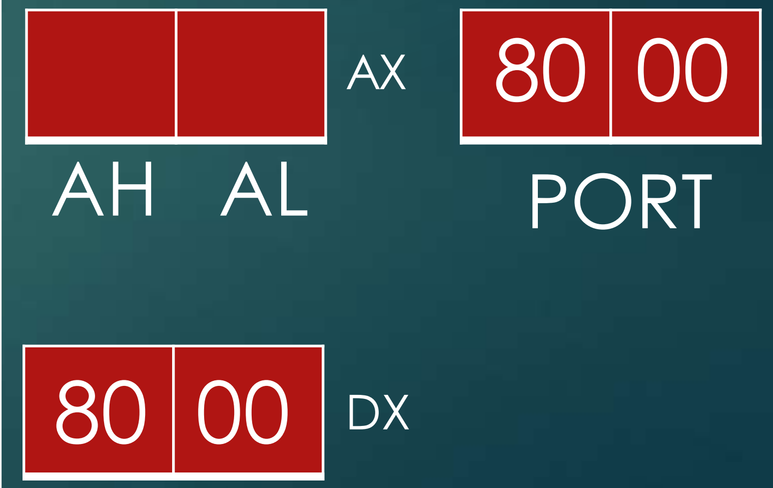
AX

AH AL

80

PORT

Eg: IN AX, DX



AX

AH AL

80 00

PORT

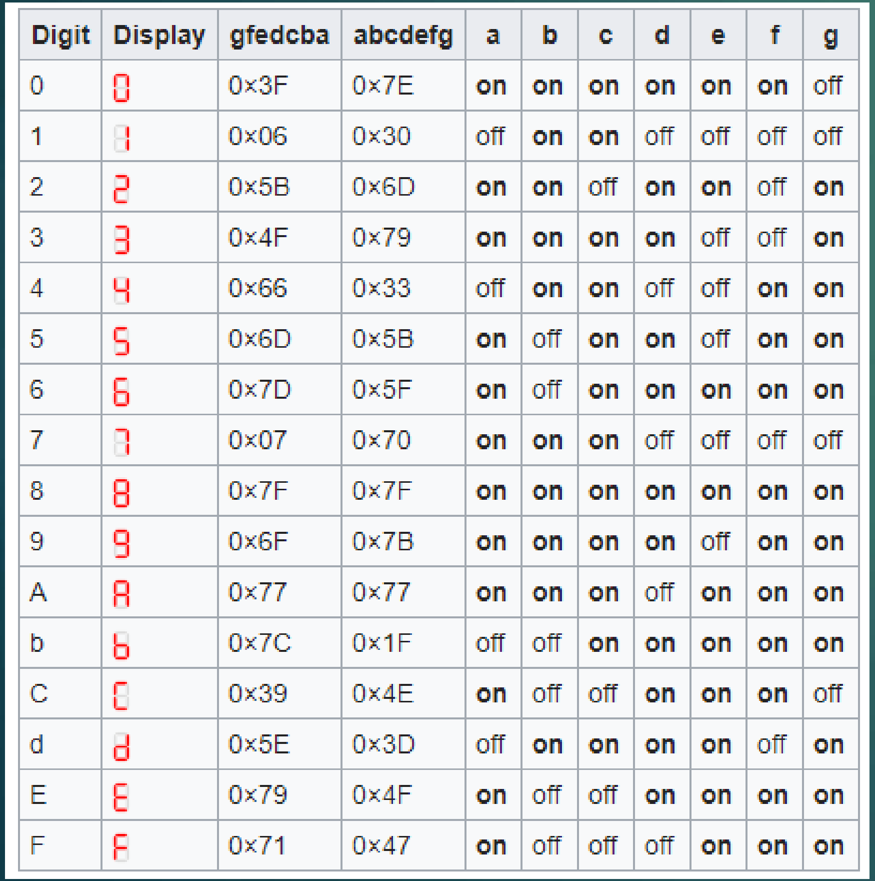
80 00

DX

1: Data Copy/Transfer instructions

XLAT

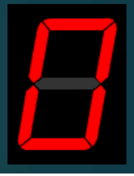
- Translate instruction is used to find out codes in case of code conversion.

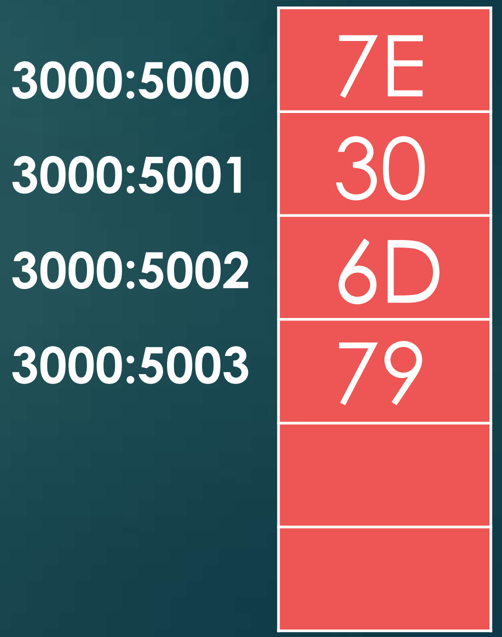


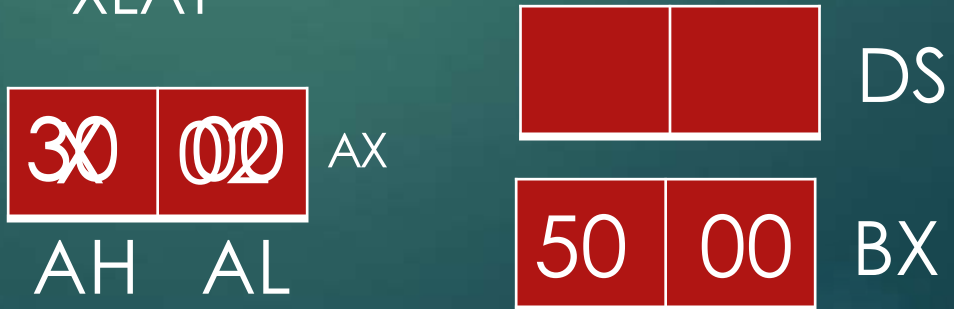
Digit	Display	gfedcba	abcdefg	a	b	c	d	e	f	g
0	0	0×3F	0×7E	on	on	on	on	on	on	off
1	1	0×06	0×30	off	on	on	off	off	off	off
2	2	0×5B	0×6D	on	on	off	on	on	off	on
3	3	0×4F	0×79	on	on	on	on	off	off	on
4	4	0×66	0×33	off	on	on	off	off	on	on
5	5	0×6D	0×5B	on	off	on	on	off	on	on
6	6	0×7D	0×5F	on	off	on	on	on	on	on
7	7	0×07	0×70	on	on	on	off	off	off	off
8	8	0×7F	0×7F	on	on	on	on	on	on	on
9	9	0×6F	0×7B	on	on	on	on	off	on	on
A	A	0×77	0×77	on	on	on	off	on	on	on
b	b	0×7C	0×1F	off	off	on	on	on	on	on
C	C	0×39	0×4E	on	off	off	on	on	on	off
d	d	0×5E	0×3D	off	on	on	on	on	off	on
E	E	0×79	0×4F	on	off	off	on	on	on	on
F	F	0×71	0×47	on	off	off	off	on	on	on

Eg:

```
MOV AX, TABLE SEGMENT ADDRESS
MOV DS, AX
MOV AL, DISPLAY CODE
MOV BX, OFFSET TABLE
XLAT
```







2: Arithmetic Instructions

- Addition
- Subtraction
- Increment
- Decrement
- Multiply
- Divide

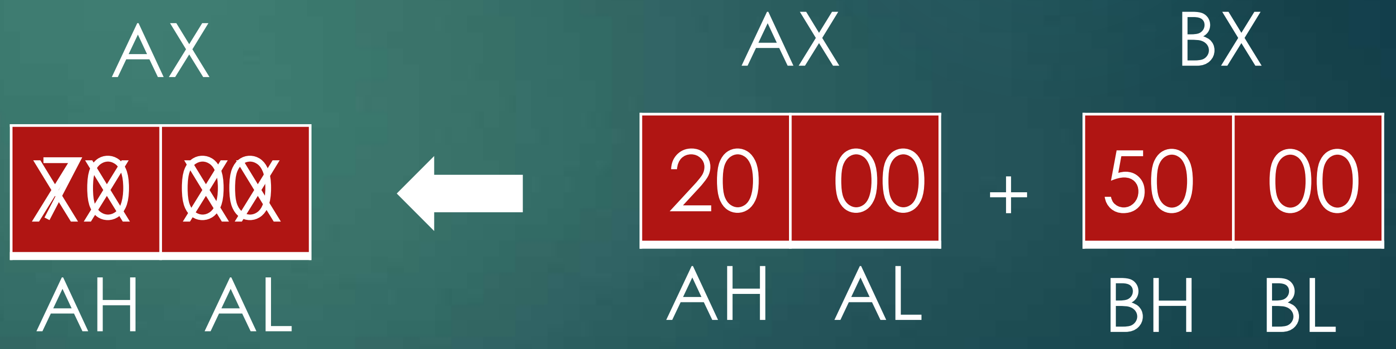
2: Arithmetic Instructions

ADD Destination, Source

- This instruction adds the contents of source operand with the contents of destination operand. The result is stored in destination operand.

- The source may be immediate data, memory location or register.
- The destination may be memory location or register.
- AX is the default destination register.

Eg: ADD AX,BX



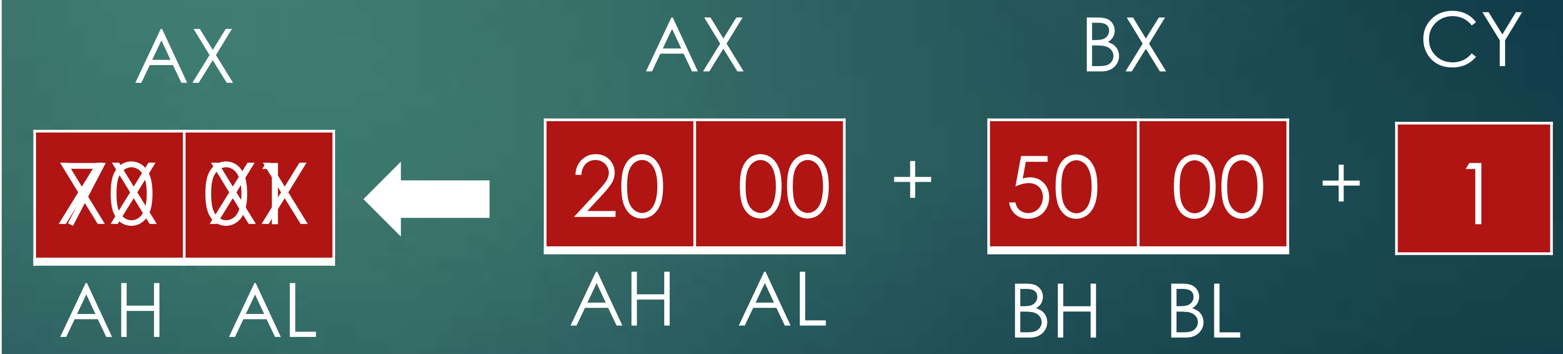
2: Arithmetic Instructions

ADC Destination, Source

- This instruction adds the contents of source operand with the contents of destination operand with carry flag bit.

- The source may be immediate data, memory location or register.
- The destination may be memory location or register.
- The result is stored in destination operand.
- AX is the default destination register.

Eg: ADD AX,BX



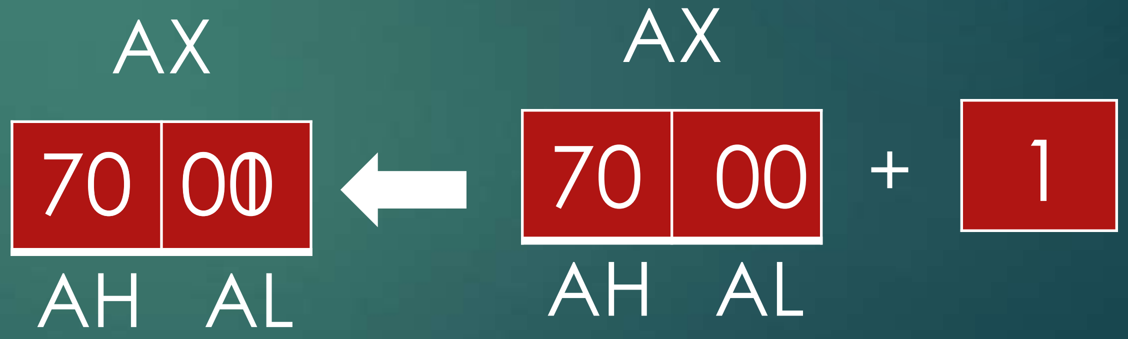
2: Arithmetic Instructions

INC source

- This instruction increases the contents of source operand by 1.

- The source may be memory location or register.
- The source can not be immediate data.
- The result is stored in the same place.

Eg: INC AX



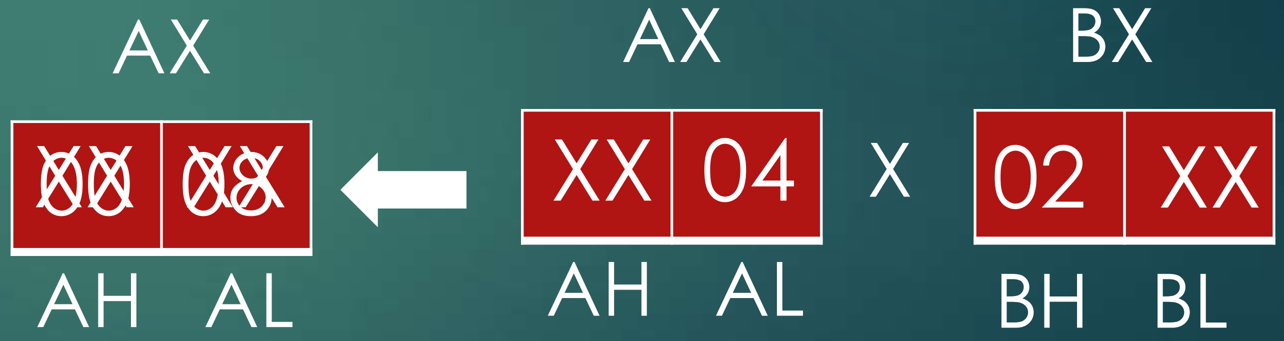
2: Arithmetic Instructions

MUL operand

- This instruction will multiple unsigned operand 8-bit/16-bit with AL/AX and store the result in AX/DX-AX.

- Operand may be general purpose register or memory location.
- If operand is of 8-bit then multiply it with contents of AL.
- If operand is of 16-bit then multiply it with contents of AX.
- Result is stored in accumulator AX in 8 bit operation and DX-AX in 16bit operation.

Eg: MUL BH



AX

XX 00 | XX 08

AH AL

AX

XX | 04

AH AL

X

BX

02 | XX

BH BL

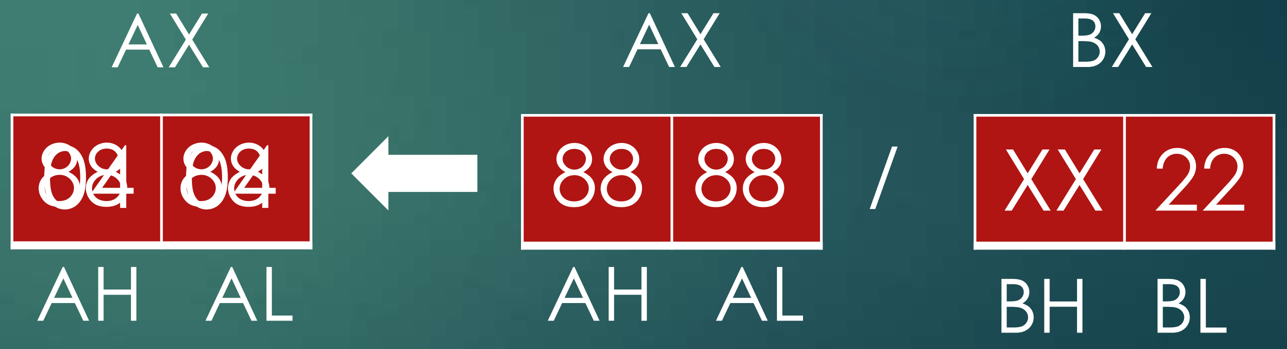
2: Arithmetic Instructions

DIV Operand

- This instruction will divide unsigned operand AX/DX-AX by 8-bit/16-bit number and store the result in AX/DX-AX

- Operand may be general purpose register or memory location.
- AL=AX/Operand (8-bit)
- AL= Quotient, AH=Remainder.
- AX=DX-AX/Operand (16-bit)
- AX= Quotient, DX=Remainder.

Eg: DIV BL



3: Bit Manipulation Instructions (LOGICAL Instructions)

- AND
- OR
- XOR
- NOT

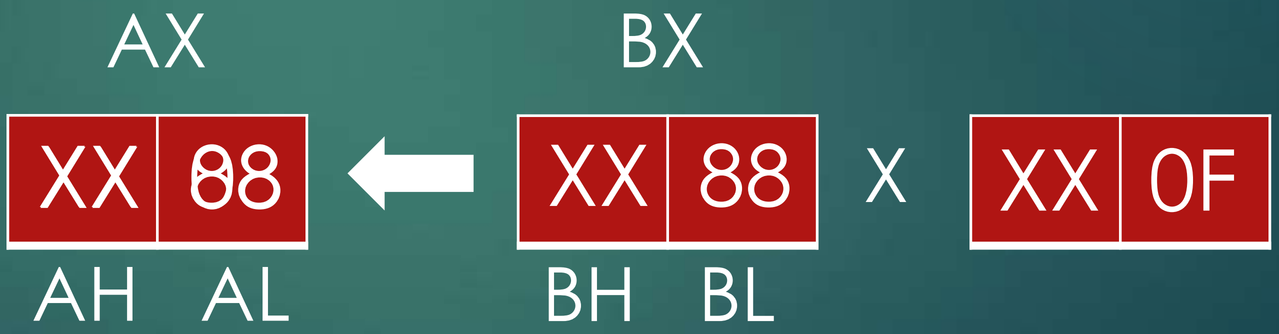
3: Bit Manipulation Instructions (LOGICAL Instructions)

AND

- Especially used in clearing certain bits (masking)

xxxx xxxx AND 0000 1111 = 0000 xxxx

Eg: AND BL, 0FH



Assesement

You just studied about the AND operator is used to clear certain bits. Which operator would be used to set certain bits without effecting the other bits

Pause and write your answer in your course journal

Assessment (Answer)

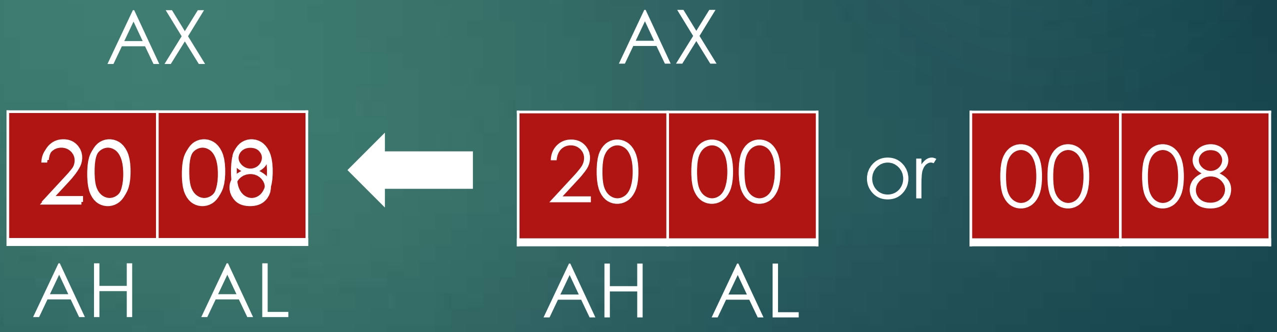
OR

- Used to multiply each bit in a byte/word with the corresponding bit in another byte/word.

xxxx xxxx **OR** 0000 1111 = xxxx 1111

Eg: MOV AX, 2000h

OR AX, 0008h



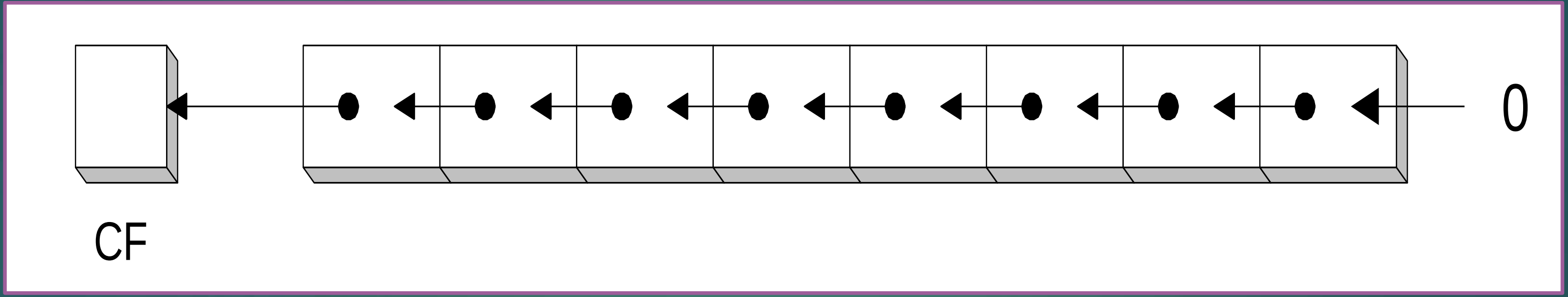
0010 0000 **OR** 0000 1000 = 0010 1000

This sets bit 4 in the AX register

4: Instructions to perform shift operations

SHL

- The SHL (shift left) instruction performs a logical left shift on the destination operand, filling the lowest bit with 0.



Eg: MOV BL,5d

SHL BL,1

After

XX	0A
BH	BL

0	0	0	0	1	0	1	0

←

Before

XX	05
BH	BL

0	0	0	0	0	1	0	1

5: Branch instructions

- CALL – Used to call a procedure and save their return address to the stack.
  - Eg: CALL Label
- RET – Used to return from the procedure to the main program.
- JMP – Used to jump to the provided address to proceed to the next instruction.
  - JMP Label

5: Branch instructions

Instructions to transfer the instruction during an execution with some conditions

- **JA/JNBE** – Used to jump if above/not below/equal instruction satisfies.
- **JAE/JNB** – Used to jump if above/not below instruction satisfies.
- **JBE/JNA** – Used to jump if below/equal/ not above instruction satisfies.
- **JC** – Used to jump if carry flag CF = 1
- **JE/JZ** – Used to jump if equal/zero flag ZF = 1
- **JG/JNLE** – Used to jump if greater/not less than/equal instruction satisfies.
- **JGE/JNL** – Used to jump if greater than/equal/not less than instruction satisfies.
- **JL/JNGE** – Used to jump if less than/not greater than/equal instruction satisfies.
- **JLE/JNG** – Used to jump if less than/equal/if not greater than instruction satisfies.
- **JNC** – Used to jump if no carry flag (CF = 0)
- **JNE/JNZ** – Used to jump if not equal/zero flag ZF = 0
- **JNO** – Used to jump if no overflow flag OF = 0
- **JNP/JPO** – Used to jump if not parity/parity odd PF = 0
- **JNS** – Used to jump if not sign SF = 0
- **JO** – Used to jump if overflow flag OF = 1
- **JP/JPE** – Used to jump if parity/parity even PF = 1
- **JS** – Used to jump if sign flag SF = 1

Summery

What we have learnt

- Studied 5 types of instruction set.
- How different instructions can manipulate data in different ways.