Instruction: A binary pattern designed inside a microprocessor to perform a specific function

Opcode: Operational Code, specifies type of operation performed by CPU.
first field of Assembly Language (Machine language instruction format).
eg 08 is for MOV X, Y

Operand: Data on which operations act on.
Register or memory values 8-bit or 16-bit

Assembler: Converts instruction into sequence of binary bits, so that bits can be read by the processor.

Mnemonics: Symbolic codes for either instructions or commands to perform a particular function.
eg MOV, ADD, SUB

Stack Pointer: 16 bit register, address of data on top of stack

H-L register pair: Higer-Lower register pair, pair denoted by X.
[AX = AH - AL , CX = CH - CL ...]

Port Address: for I/O ports

eg: Multiply AX by 10

```
SHL  AX, 1
MOV  BX, AX
MOV  CL, 2
SHL  AX, CL
ADD  AX, BX
```

Fast Multiplication: Shifting left 1 bit multiplies the number by 2
Shifting left n bits multiplies the number by 2^n

```
eg:  mov dl, 5
     shl dl, 2     :  DL = 20  ( 5 * 2^2 = 20)
```

Fast division: Shifting right n bits divides the operation by 2^n

Arithmetic shifts preserve the number's sign.

```
eg  MOV  DL, -80
    SAR  DL, 1     :  DL = -40 {-80/2^1 = -40}
    SAR  DL, 2        DL = -10 {-40/2^2 = -10}
```

Data copy / transfer instructions

i) MOV dest, source

Moves data from source to destination

Source : register, memory location, data

Destination: register, memory operand

Both source and destination cannot be mem location (037AH eg) or segment registers (SS, DS, ES etc).

Eg MOV AL, BL

ii) Push source

Pushes content of source to stack.

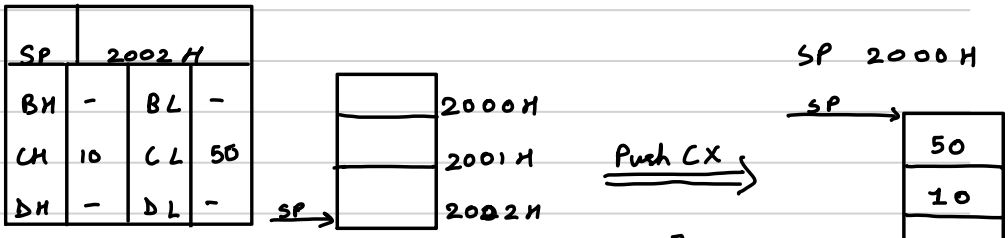
Source: register, segment register, memory

Decrement sp by 2

Higher byte first

Lower byte second

eg Push CX , Push DS, Push AX



[Here CX refers to the C(H-L pair) register]

[Higher byte = H , Lower byte = L]

iii) Pop destination

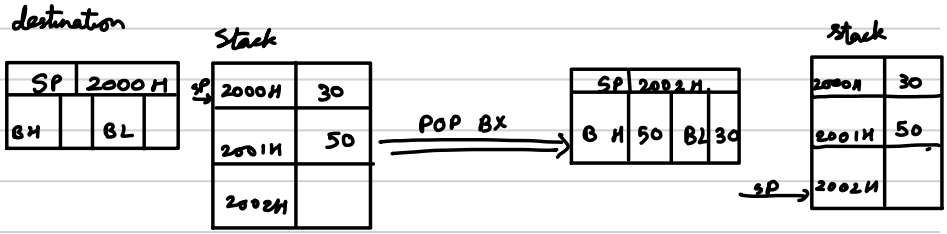
destination : register, segment register , memory

Pops content (removes) from stack to destination

sp increments by 2

Lower first , Higher second

eg POP AX, POP DS, POP [5000H]



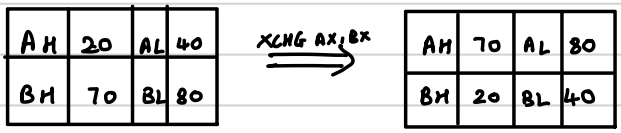
iv) XCHG dest, source

contents of source go to destination and vice versa.

eXCHanGE of contents / data

2 memory locations cannot directly perform exchange

eg XCHG BX, AX XCHG [5000H], AX



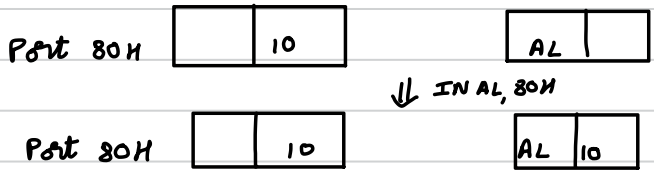
v) IN AL/AX, 8-bit/16-bit port address

reads from specified port address

Copies data to accumulator from a port

DX only register for port address

eg IN AL, 08H IN AX, DX



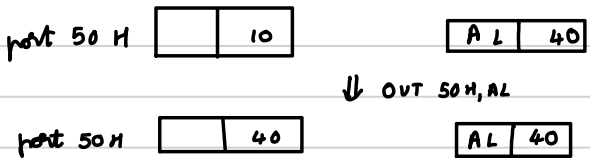
vi) OUT 8-bit/16-bit port address

writes to the specified port address

copies data to port from accumulator

DX only register for port address

eg OUT 80H, AL OUT DX, AX



viii) XLAT

translate instruction

for code conversion

replaces AL register contents

viiii) LEA 16-bit register (source), address (dest)

Load Effective Address.

EA formed by the destination into source register.

eg LEA BX, Address LEA SI, Address[BX]

LEA Reg16, EA

EA → (Reg 16)

ix) LDS 16 bit register (source), address (dest)

Load Data Segment

Contents of DS & destination to source register contents

eg LDS BX, 5000H

LDS Reg 16, Mem32

Mem32 → Reg16

Mem 32+2 → DS

x) LES 16-bit register (source), address (dest)

Load Extra Segment

Contents of ES & destination to source register contents

eg LES BX, 5000H

Mem32 → Reg16

Mem 32+2 → ES

xi) LAHF

loads AH from the contents of lower byte of flag register

observes status of all conditional flags of flag register

xii) SAHF

sets or resets all conditional flags of flag register wrt corresponding bit.

bit position in AH is 1 then related flag is set otherwise reset.

xiii) PUSHF

decrements sp by 2

copies content from flag register to memory location pointed by SP.

xiv) POPF

increments sp by 2

copies content of memory location pointed by SP to flag register.

Arithmetic Instructions

i) ADD dest, source

adds source content to destination content

AH	10	AL	10

ADD AX, 2020H →

AH	30	AL	30

source: data, memory location, register

destination: memory location, register (Stores result)

AX is default destination

eg ADD AX, 2020H ADD AX, BX

ii) ADC dest, source

adds contents of source with carry bit to destination

CY	1

AH	10	AL	10

ADC AX, 2020H →

AH	30	AL	31

source: data, memory location, register

destination: memory location, register

Ax default destination

eg ADC AX, 2020H ADC Ax, BX [added to lower byte]

iii) INC source

increments contents by 1

5000H | 1010 — INC [5000H] → 5000H | 1011

not immediate data, can be memory location or register

result same place

eg INC AX INC [5000H] [added to lower byte]

iv) DEC source

decrements contents by 1

5000H | 1010 — DEC [5000H] → 5000H | 100F

not immediate data, can be memory location or register

result same place

eg DEC Ax DEC [5000H] [added to lower byte]

v) SUB destination, source

subtracts source content from destination content

AH	20	AL	00

SUB AX, 1000H →

AH	10	AL	00

source: data, memory location, register

destination: memory location, register (Stores result)

AX is default destination

eg SUB AX, 1000H SUB AX, BX

vi) SBB dest, source

B	1

AH	20	AL	20

SBB AX, 1000H →

AH	10	AL	19

Subtract with borrow

subtract the contents of source & borrow from contents of destination

source: data, memory location, register

destination: memory location, register (Stores result) eg: SBB AX, 1000H SBB AX, BX

Machine Control Instructions

i) NOP

No operation to be performed

ii) HLT

Halt, processor in stable condition (do nothing)

Flag Manipulation Instructions

i) CMC

Complement carry flag

Inverts carry flag

CF = 1 $\xrightarrow{CMC}$ CF = 0

ii) CLI

Clear interrupt flag

reset condition of interrupt flag

$\xrightarrow{CLI}$ IF = 0

iii) STC

Set carry flag

$\xrightarrow{STC}$ CF = 1

iv) CLD

Clear direction flag

reset condition for direction flag

$\xrightarrow{CLD}$ DF = 0

Refer ppt for : String, Logical, Rest of Arithmetic, Branching