COURSE: UCS1502 - MICROPROCESSORS AND INTERFACING

Instruction set of 8086 – Part 1 (Data transfer, Arithmetic and Logical instructions)

Dr. K. R. Sarath Chandran
Assistant Professor, Dept. of CSE

This presentation covers

- Instruction set of 8086 (Data transfer, Arithmetic and Logical instructions)

Learning Outcome of this module

- To understand data transfer, arithmetic and logical instructions of 8086

Contents

- Different types of instructions of 8086
- Explanation of all instructions of 8086

Types of instructions in 8086

1. Data transfer instructions
2. Arithmetic and logical instructions
3. Branch instructions
4. Loop instructions
5. Machine control instructions
6. Flag manipulation instructions
7. Shift and rotate instructions
8. String instructions

Data transfer instructions

MOV

Used to copy the byte or word from source to destination

Egs:

```
MOV AX, BX
MOV CH, AH
MOV AX, [BX]
MOV AX, 5000H
MOV [5000], AX
MOV CX, [437A]
```

Direct loading to segment registers with immediate data is not permitted

```
MOV DS, 5000H
```

✗

```
MOV AX, 5000H
MOV DS, AX
```

✓

Data transfer instructions

PUSH

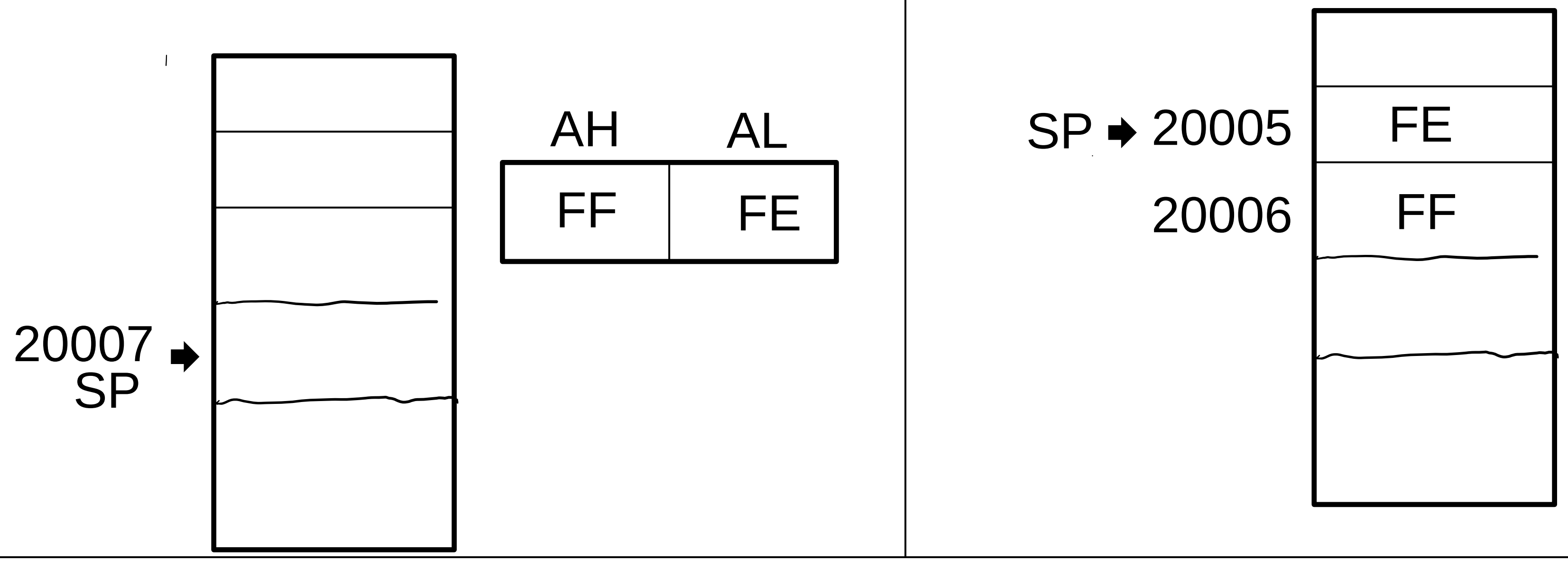
PUSH AL ✗

Used to put a word at the top of the stack

PUSH AX

Before execution

After execution



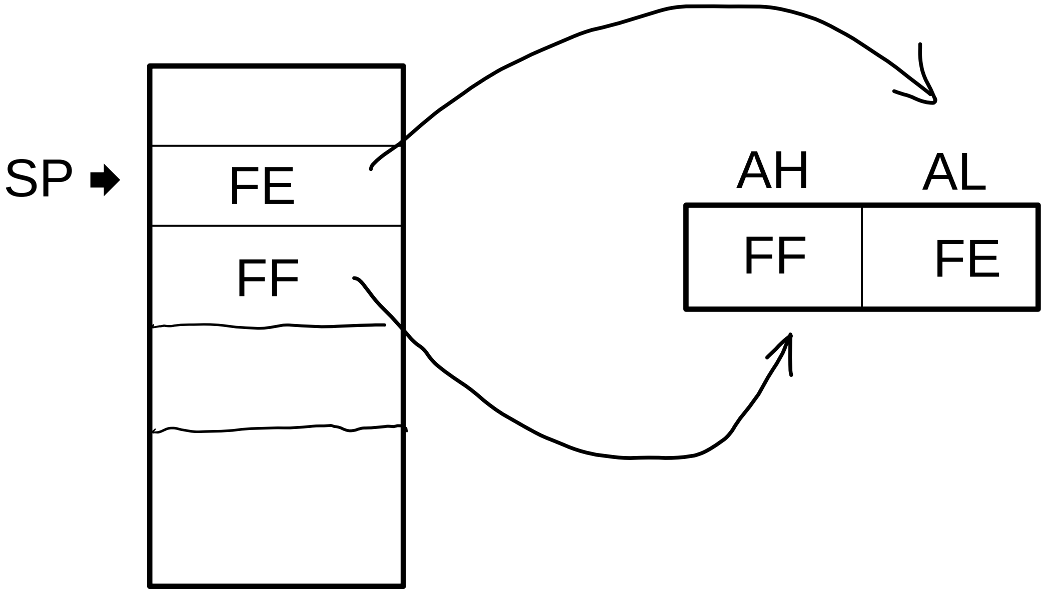
PUSHF – Push flag register

Data transfer instructions

POP

Used to get a word from the top of the stack.

Eg: POP AX



POPF – Pop word from stack top to flag register

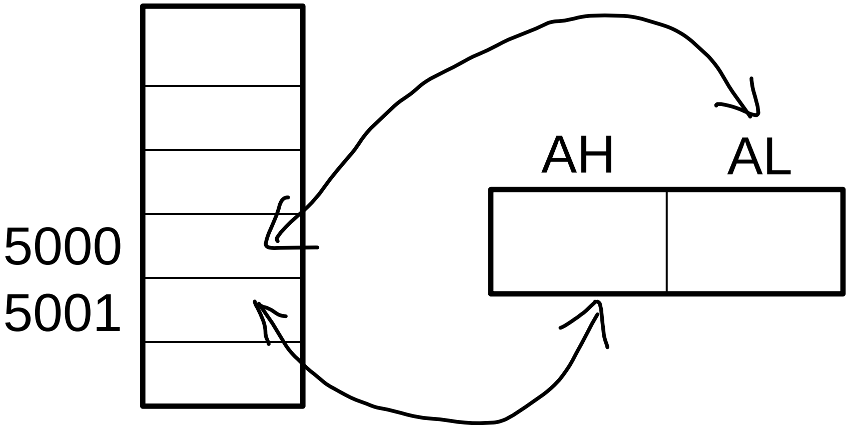
Data transfer instructions

XCHG

Used to exchange the data between source and destinations.

XCHG AX, DX
XCHG [5000], AX

XCHG [5000],[4000] ✗



Data transfer instructions

IN

Used to read a byte or word from the provided port to the accumulator

IN AL, 0C0 : to read 8-bit data from port with address C0 to AL
IN AX, 0C1 : to read 16-bit data from port with address C1 to AX

OUT

Used to send out a byte or word from the accumulator to the provided port.

OUT 3B, AL : to send 8-bit data from AL to a port with address 3B
OUT 2C, AX : to send 16-bit data from AX to a port with address 2C

Data transfer instructions

XLAT

Used to translate a byte in AL using a look up table in the memory.

The value present in AL before XLAT is replaced by the content of new offset pointed by AL+BX

Look up table should be in data segment. Starting address of look up table should be in BX.

AL = [10HxDS + BX + AL]

```
assume cs:code, ds:data
data segment
             table1 db 0ah, 0bh, 0ch
              result db 00h
data ends
code segment
start:       mov ax,data
             mov ds,ax

             mov al,01
             mov bx,offset table1
             xlat
             mov result, al
             mov ah,4ch
             int 21h
code ends
end start
```

AL=0b

Assembler directives are given in blue

Data transfer instructions

LEA

Load effective address to a register

LEA bx, table1

(similar to
mov bx, offset table1)

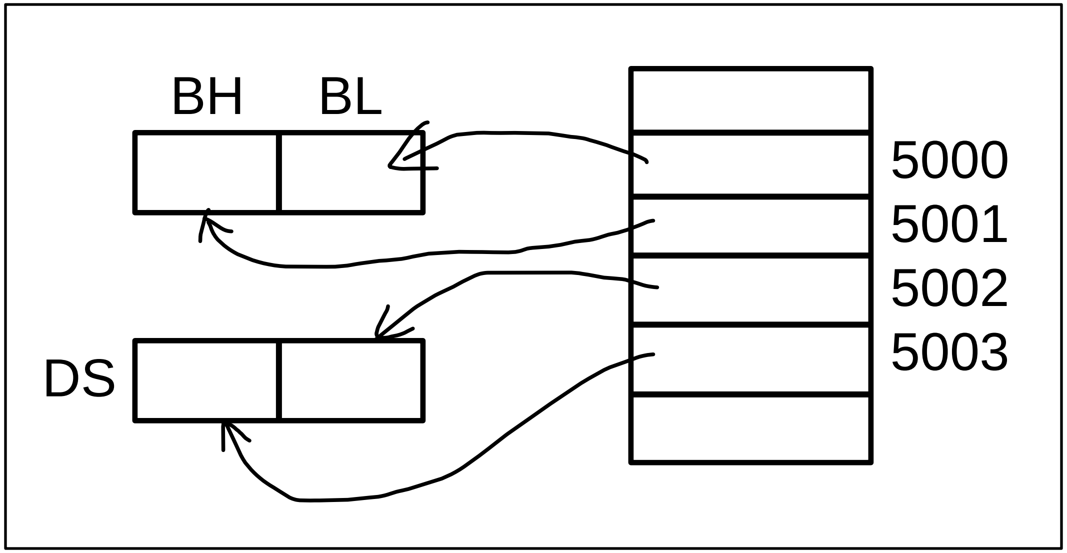
Assembler directive

Data transfer instructions

LDS

Used to load DS register and other provided register from the memory

LDS BX, [5000]



LES

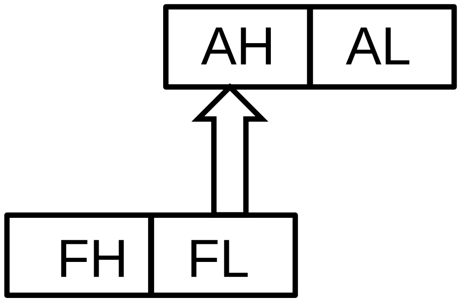
Used to load ES register and other provided register from the memory

LES BX, [5000] (loading pattern is same as LDS instruction)

Data transfer instructions

LAHF

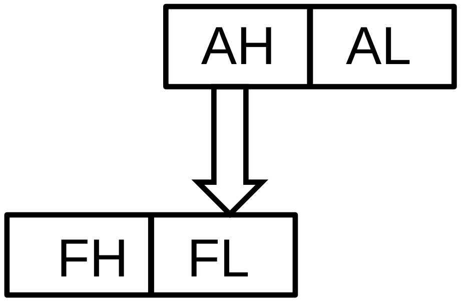
Used to load AH with the lower byte of the flag register.



Flag register

SAHF

Used to store AH register to lower byte of the flag register.



Flag register

Arithmetic instructions

ADD

Used to add the provided byte to byte/word to word.

ADD AL, BL ⇨ AL = AL + BL
ADD AX, BX
ADD AX, [SI]
ADD AX, [5000]
ADD AX, 012AH
ADD [5000],[4000] ✗

ADC

add with carry.

ADC AL, BL ; AL=AL+BL+CY
ADC AX, BX
ADC AX, [SI]
ADC AX, [5000]

INC

Increment the provided byte/word by 1.

INC AL
INC AX
INC [BX]
INC [5000]

DEC

Used to decrement the provided byte/word by 1.

DEC AL
DEC AX
DEC [BX]
DEC [5000]

Arithmetic instructions

SUB

To subtract the byte from byte/word from word.

SUB AL, BL ⇨ AL = AL - BL

SUB AX, BX

SUB AX, [SI]

SUB AX, [5000]

SUB AX, 012AH

SUB [5000],[4000] ✗

SBB

to perform subtraction with borrow.

SBB AL, BL ⇨ AL = AL – (BL + CY)

SBB AX, BX

SBB AX, [SI]

SBB AX, [5000]

SBB AX, 012AH

CMP

Used to compare 2 provided byte/word. Source and destination values are not changed after execution. Only flags are affected.

CMP AX, BX

CMP AX, [5000]

CMP CX, BX

	CF	ZF	SF
If CX=BX	0	1	0
If CX>BX	0	0	0
If CX<BX	1	0	1

Arithmetic instructions

AAA – ASCII adjust after addition

Used to adjust ASCII after addition.
It allows to add a ASCII values without masking “3”. It works only with AL.

Before execution AL = 32 BL = 33 **ADD AL,BL** **AAA** After execution CF=0, AL=05	Before execution AL = 35 BL = 39 **ADD AL,BL** **AAA** After execution CF=1, AL=04

AAS – ASCII adjust after subtraction

Used to adjust ASCII after subtraction.

Before execution AL = 35 BL = 39 **SUB AL,BL** **AAS** After execution CF=1, AL=04	Before execution AL = 39 BL = 35 **SUB AL,BL** **AAS** After execution CF=0, AL=04

Arithmetic instructions

AAM – BCD adjust after multiplication

Before execution
AL = 05
BH = 09

MUL BH
AAM

After execution
AH=04, AL=05

2Dh=45 in decimal

Before execution
AL = 03
BL = 05

MUL BH
AAM

After execution
AH=01, AL=05

AAD

BCD to binary conversion before division.
Converts 2 unpacked BCD in AH and AL to the binary equivalent in AL

Before execution
AH = 06, AL = 07

AAD

After execution
AH=00, AL=43

67 decimal = 43H

Before execution
AH = 06, AL = 07
CH =09

AAD
DIV CH

After execution
AH=04, AL=07

Arithmetic instructions

DAA – Decimal adjust accumulator. Converts invalid BCD to valid BCD. Sum should be in AL.

Before execution AL = 53 CL = 29 **ADD AL,CL** **DAA** After execution AL=82	Before execution AL = 80 CL = 80 **ADD AL,CL** **DAA** After execution CF=1, AL=60

DAS – Decimal adjust after subtraction

Before execution AL = 75, BH = 46 **SUB AL, BH** **DAS** After execution CF=0, AL=29	Before execution AL=38, CH=61 **SUB AL, CH** **DAS** After execution CF=01, AL=77 (10’s complement of 23 = 77)

Arithmetic instructions

NEG – to find the 2’s complement

NEG AL ; AL = 2’S COMPLEMENT OF AL
NEG AX

NEG BYTE PTR[BX]

MUL – UNSIGNED MULTIPLICATION

MUL BH ; AX = AL x BH
MUL CX; DXAX = AX x CX

IMUL – SIGNED MULTIPLICATION

IMUL BH ; AX = AL x BH

IMUL CX; DXAX = AX x CX

CBW – Used to fill the upper byte of the word with the copies of sign bit of the lower byte. (Convert byte to word)

AL=11011010

CBW

AX = 11111111 11011010

Arithmetic instructions

CWD – Used to fill the upper word of the double word with the sign bit of the lower word.
(Convert word to double word)

Copies the sign bit of AX to
All bits of DX.

DIV– UNSIGNED DIVISION

DIV BL ; AX/BL; quotient in AL, reminder in AH
DIV CX; DXAX/CX, quotient in AX, reminder in DX

IDIV - SIGNED DIVISION

Logical instructions

AND – bitwise AND

AND AX,BX
AND AX, 0008H
AND AX, [5000]
Both operands in memory is not allowed.

OR – bitwise OR
NOT – complement - NOT AX
XOR

TEST – Logical compare

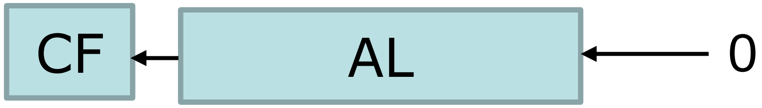
Performs logical AND.
Source and destination remain same. Only flags will be affected

TEST AX, BX
TEST AX, 0008H
TEST AX, [5000]

Logical instructions

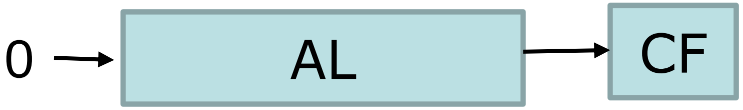
SHL (SAL) – Shift left

MOV CL, 04
SHL AL, CL ; shift AL left CL bits (here 4 bits)



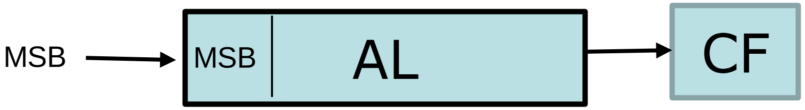
SHR – Shift right

MOV CL, 03
SHR AL, CL ; shift AL right CL bits (here 3 bits)



SAR – Shift arithmetic right

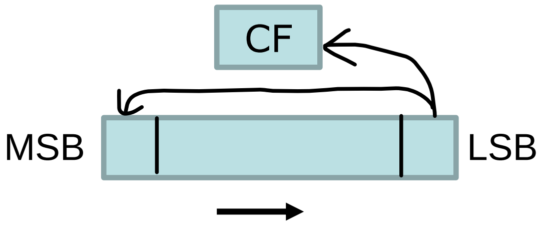
MOV CL, 02
SAR AL, CL ; arithmetic shift AL right CL bits



Logical instructions

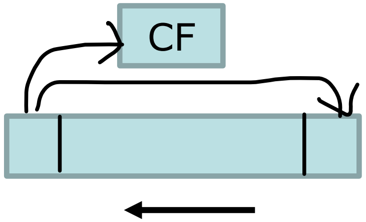
ROR – Rotate right without carry

MOV CL, 04
ROR AL, CL ; Rotate AL right , CL bits without carry (here 4 bits)



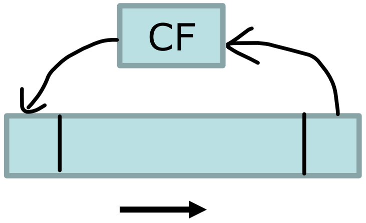
ROL – Rotate left without carry

MOV CL, 04
ROL AL, CL ; Rotate AL left , CL bits without carry (here 4 bits)



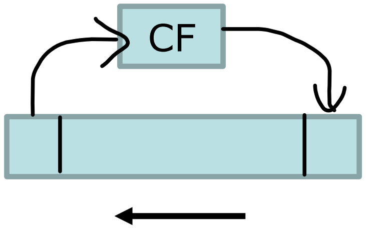
RCR – Rotate right with carry

MOV CL, 04
RCR AL, CL ; Rotate AL right , CL bits with carry (here 4 bits)



RCL – Rotate left with carry

MOV CL, 04
RCL AL, CL ; Rotate AL left , CL bits with carry (here 4 bits)



References

- Doughlas V. Hall, “Microprocessors and Interfacing, Programming and Hardware”, Second Edition, TMH, 2012.

Thank you