8086 Instruction SET

MOV - MOV Destination, Source

The MOV instruction copies a word or byte of data from a specified source to a specified destination. The destination can be a register or a memory location. The source can be a register, a memory location or an immediate number. The source and destination cannot both be memory locations.

→ MOV CX, 037AH Put immediate number 037AH to CX

→ MOV BL, [437AH] Copy byte in DS at offset 437AH to BL

→ MOV AX, BX Copy content of register BX to AX.

→ MOV DL, [BX] Copy byte from memory at [BX] to DL

→ MOV DS, BX Copy word from BX to DS register

→ MOV Result [BP], AX Copy AX to two memory location.

→ MOV ES: RESULTS [BP], AX Same as the above inst, but physical addres = EA + ES, beacuse of the segment override prefix

LEA-LEA Register, Source

This instruction determines the offset of the variable or memory location named as the source and puts this offset in the indicated 16-bit reg. LEA does not affect any flag.

→ LEA BX, PRICES Load DX with offset of PRICE in DS

→ LEA BP, SS: STACK_TOP Load BP with offset of STACK TOP in

→ LEA CX, [BX][DI] LOAD CX with EA = [BX] + [DI]

ADD- ADD Destination, Source
ADC - ADC Destination, Source

These inst add a number from some source to a number in some destination and put the result in the specified destination. The ADC also adds the status of the carry flag to the result. The source and the destination must copy the byte to a word location and fill the upper byte of the word with 0's before adding.

→ ADD AL, 74H Add immediate number 74H

→ ADC CL, BL Add Content of BL Plus Carry

→ ADD DX, BX Add content of BX to content DX

→ AD DX, [SI] Add word from memory

→ ADC AL, PRICES [BX] Add byte from effective

→ ADD AL, PRICES Add content of memory at effective

SUB - SUB Destination, source
SBB - SBB " , source

These inst subtract the number in some source from the number in some destination and put the result in the destination. The SBB instruction also subtracts the content of carry flag from the destination. You must first move the byte to a word location such as 16 bit reg and fill the upper byte of the word with 0's.

→ SUB CX, BX CX - BX ; Result in CX
→ SBB CH, AL subtract content of AL and
→ SUB AX, 3427H subtract immediate num 3427H
→ SBB BX, [3427H] subt word at 3427H in DS
→ SUB PRICES [BX], 04H subt 04 from PRICE [BX]
→ SBB CX, TABLE [BX] subt TABLE [BX]
→ SBB TABLE [BX], CX Sub CX from TABLE [BX]

MUL- MUL Source

This instruction multiplies an unsigned byte in some source with an unsigned byte in AL Register or an unsigned word in multiplied by the content of AX, the result is put in DX and AX register.

If you want to multiply a byte with a word, you must first move the byte to a word location such as an extended register and fill the upper byte ...

→ MUL BH — Multiply AL with BH, result in AX

→ MUL CX; — Multiply AX with CX

→ MUL BYTE PTR [BX] — Multiply AL with byte in DS pointed by BX

→ MUL FACTOR [BX] — Multiply AL with byte at effective address FACTOR

→ MOV AX, MCAND_16 ; Load 16-bit multiplicand in AX
MOV CL, MPLIER_8 ; Load 8 bit multiplier in CL
MOV CH, 00H ; Set upper byte of CX to all 0's
MUL CX ; AX times CX; 32 bit result in DX and AX

DIV - DIV Source

This instruction is used to divided an unsigned word by a byte of divid an unsigned double word (32 bits) by a word. When a word is divided by a byte, the word must be in the AX register. The devison can be in a register memory, the 8086 will generate a type 0 interrupt. All flags are undefind after a DIV instruction.

If you want to divide a byte by a byte, in AX and fill DX with all 0's.

→ DIV BL — Divide word in AX by byte in DL

→ DIV CX — Divide down word in DX and AX by word in CX;

→ DIV SCALE — AX/(byte at effective address)
if SCALE [BX] is of type word

INC-INC Destination:

The INC instruction adds 1 to a specified register or to a memory location. AF, OF, PE, SF and ZF are updated, but CF is not affected. FFH or a 16 bit destination containing FFFH is incremented, the result will be all 0's with no carry.

→ INC BL Add 1 to contains BL Register

→ INC CX Add 1 to contains of CX Register

→ INC BYTE PTR [BX] increment byte in Data

→ INC WORD PTR [BX].

→ INC TEMP[BX] increment [BX] PRICE

→ INC PRICE[BX] increment [PRICE]

DEC-DEC Destination!

This instruction subtracts 1 from the destination word or byte. The destination can be a register or a memory location. AF OF SF PF and ZF are updated but CF is not affected. This means that if an 8-bit destination containing 00H or 16-bit 0000H is decremented.

→ DEC CL
→ DEC BP
→ DEC BYTE PTR [BX]
→ DEC WORD PTR [BP]
→ DEC COUNT

Subtracted 1 from CL
" 1 " BP Reg
" 1 " offset [BX] is DS
" 1 " by BP is SS
" 2 " DD

MUL - MUL Source

This instruction multiplies an unsigned byte in some source with an unsigned byte in AL Register or an unsigned word in multiplied by the content of AX, the result is put in DX and AX register.

If you want to multiply a byte with a word, you must first move the byte to a word location such as an extended register and fill the upper sym..

→ MUL BH Multiply AL with BH, result in AX

→ MUL CX; Multiply AX with CX

→ MUL BYTE PTR [BX] Multiply AL with byte in DS pointed by BX

→ MUL FACTOR [BX] Multiply AL with byte at effective address FACTOR

→ MOV AX, MCAND_16 ;Load 16-bit multiplicand in AX
MOV CL, MPLIER_8 ;Load 8 bit multiplier in CL
MOV CH, 00H ;set upper byte of CX to all 0's
MUL CX AX times CX; 32 bit result in DX and AX

DAA (DECIMAL ADJUST AFTER BCD ADDITION)

This instruction is used to make sure the result of adding two Packed BCD numbers is adjust to be a legal BCD. If the result in the upper midble of AL is now greater than 9 or if the carry flag must be set by the addition or correction, then the DAA instruction will add 60H to AL.

→ Let AL = 59 BCD, and BL = 35 BCD

ADD AL, BL

AL = 8EH, Now
AL = 9D BCD, CF = 0

DAA

→ Let AL = 88 BCD, and BL = 49 BCD

ADD AL, BL

AL = D1H; AF = 1, add 06H to AL

DAA

AL = D7H; Upper nibble > 9, add 60H to AL

AL = 37 BCD. CF = 1

The DAA inst updates AF, CF, SF, PF and ZF

AAA (ASCII ADJUST FOR ADDITTION)

Numerial data coming into a computer from a terminal is usually is ASCII code. In this code, the numbers 0 to 9 are represented by the ASCII codes 30H to 39H. The 3086 allows you to add the ASCII the AAA instruction is used to make sure the result is the correct unpacked BCD;

→ Let AL = 0011 0101 (ASCII 5), and BL = 0011 1001 (ASCII 9)

ADD AL, BL

AAA

AL = 0111 0111 0 (6EH, which is incorrect BCD)

AL = 0000 0100 (unpacked BCD 4)

CF = 1 indicates answer is 14 decimal

Logical Inst

AND- AND Destination, Source

This instruction ANDs each bit in a source byte or word with the same number bit in a destination byte or word.
The source can be immediate number, the content of a register, or the content of a memory location.
The destination can be a register or a memory location.
CF and OF are both 0 after AND. PF, SF and ZF are updated by the AND inst.

→ AND CX, [SI] AND word in DS at offset [SI] with word in CX register

→ AND BH, CL AND byte in CL with byte in BH; Result in BH

→ AND BX, 00FFH 00FFH Masks upper byte, Leaves lower byte unchanged.

OR-OR Destination, Source

This instruction ORs each bit in a source byte or word with the same numbered bit in a destination byte or word. The result is put in the specific destination.

The source and the destination cannot both be memory locations. CF and OF are both 0 After AND. F.F, SF and ZF are updated by the AND instruction.

→ AND CX, [SI] AND word in DS at offset [SI] with word CX register

→ AND BH, CL AND byte in CL with byte in BH.

→ AND BX, 00FFH 00FFH Masks upper byte, leaves lower byte unchanged.

OR-OR Destination, Source :

The instruction ORs each bit in source byte or word with the same numbered bit in a destination byte or word. The result is put in the specified destination. The source can be an immediate number, the content of a register, or the content of a memory location. AF is undefined. PF has meaning only for an 8 bit operand.

→ OR AH, CL — CL ORed with AH, result in AH, CL not changed.

→ OR BP, SI — SI ORed with BP, result in BP, SI

→ OR SI, BP — BP ORed with SI

→ OR BL, 80H — BL ORed with word

→ OR BL, 80H

→ OR CX, TABLE [SI] — CX ORed with word from content of memory

XOR-XOR Destination, Source:

The inst Exclusive-ORs each bit in a source byte or word with the same numbered bit in a destination byte or word. The source can be an immediat number, the content of a register, or the content of a memory location. PF has meaning only for an 8-bit operand. AF is undefind.

→ XOR CL, BH — Byte in BH ORed in CL

→ XOR BP, DI — word in DI exclusive-ORed with

→ XOR WORD PTR [BX] 00FFH — Exclusive-OR immediate number 00FFH with word at result in memory,

CMP - CMP Destination, Source:

This instruction compares a byte/word in the specified source with byte/word in the specified destination. The source can be a immediate number, a register or a memory location. The destination. AF, OF, SF, ZF, PF and CF are updated by the CMP instruction. For the instruction. follows:

	CF	ZF	SF	
CX = BX	0	1	0	Result of subtract is 0
CX > BX	0	0	0	No borrow required so CF = 0
CX < BX	1	0	1	Subtract requires borrow; so CF = 1

→ CMP AL, 01H — Compare 01H — AL

→ CMP BH, CL — Compare byte in CL in BH

→ CMP CX, TEMP — Compare DS at TEMP add CX

→ CMP PRICES [BX], 49H PRICES

TEST- TEST Destination, Source

This instruction AND the byte / word in the specified source with the byte/word in the specified destination. Flags are updated, but neither operand is changed.

The source can be an immediate number, the content of a register, or the content of a memory location. AF is be undefined.

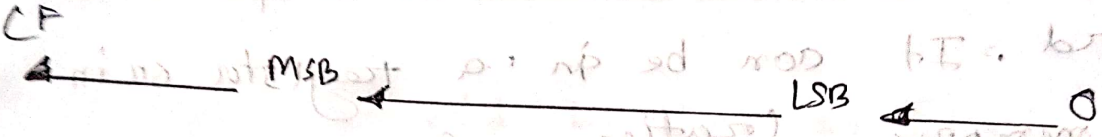
→ TEST AL, BH AND BH with AL

→ TEST CX, 0001H AND CX with immediate 0001

→ TEST BP, [BX] [D1] AND word are offset

SAL - SAL Destination, Count
SHL - SHL " , Count

SAL and SHL are two mnemonic for the same instruction. The instruction shifts each bit in the specified destination some number of bit positions to the left. As a bit is shifted out of the LSB operation,



the destination operand can be a byte or a word, and put "CL" in the count position of the instruction.

→ SAL BX, 1 ; Shift word in BX 1 bit positi

→ MOV CL, 02h ; Load desired number of sh
SAL BP, CL ; Shift word in BP

→ SAL BYTE PTR [BX], 1 ; Shift byte in DX at offset [BX] 1 bit

SAR - SAR Destination, count :

The instruction shifts each bit in the specified destination some number of bit positions to the right. Bits shifted into CF previously will be lost.

The destination operand can be a byte or a word. It can be in a register or in a memory location. If you put "CL" in count position of the instruction.

→ SAR DX, I Shift word in DI on

→ MOV CL, 02H Load desired number of shift in CL

SAR WORD PTR [BP], CL Shift word at offset [BP]

SHR - SHR Destination, Count

This instruction shifts each bit in the specified destination some number of bit positions to the right. As a bit is shifted out of MSB position. shifted into CF previously will be lost.

The flags are affected by SHR as follow:

CF Contains the bit most recently shifted out from LSB, For a count of one, OF will be 1 if the two MSBs are not both 0's.

→ SHR BX,1 Shift word in BP one bit position right, 0 in MSB

→ MOV CL,03H Load desired number of shifts into CL

→ SHR.BYTE PTR [BX] Shift byte in DS at offset [BX] 3 bits right, 0's in 3 MSBs