

---

**Week 4**

**8088/8086 Microprocessor  
Programming I**

# Example. The PC Typewriter

---

- Write an 80x86 program to input keystrokes from the PC's keyboard and display the characters on the system monitor. Pressing any of the function keys F1-F10 should cause the program to end.
- Algorithm:
  1. Get the code for the key pressed
  2. If this code is ASCII, display the key pressed on the monitor and continue
  3. Quit when a non-ASCII key is pressed
- INT 16, BIOS service 0 – Read next keyboard character
  - Returns 0 in AL for non-ASCII characters or the character is simply stored in AL
- To display the character, we use INT 10, BIOS service 0E- write character in teletype mode. AL should hold the character to be displayed.
- INT 20 for program termination

## Example continued

---

```
AGAIN:  MOV AH,0
        INT 16
        CMP AL,00
        JZ QUIT
        MOV AH, 0E
        INT 10
        JUMP AGAIN
QUIT:   INT 20
```

## Example continued with sign-on messages

---

```
        MOV DX, OFFSET MES
        MOV AH,09
        INT 21
AGAIN:  MOV AH,0
        INT 16
        CMP AL,00
        JZ QUIT
        MOV AH, 0E
        INT 10
        JUMP AGAIN
QUIT:  INT 20
MES    DB 'type any letter, number or punctuation key'
        DB 'any F1 to F10 to end the program"
        DB 0d,0a,0a,'$'
```

# Data Transfer Instructions - MOV

---

Mnemonic	Meaning	Format	Operation	Flags Affected
MOV	Move	MOV D, S	(D) → (S)	None

Destination	Source
Memory	Accumulator
Accumulator	Memory
Register	Register
Register	Memory
Memory	Register
Register	Immediate
Memory	Immediate
Seg reg	Reg16
Seg reg	Mem16
Reg 16	Seg reg
Memory	Seg reg

Memory to  
memory  
is not allowed

# Data Transfer Instructions - XCHG

---

Mnemonic	Meaning	Format	Operation	Flags Affected
XCHG	Exchange	XCHG D,S	(D) ↔ (S)	None

Destination	Source
Accumulator	Reg16
Memory	Register
Register	Register
Register	Memory

Example. XCHG [1234h], BX

# Data Transfer Instructions – LEA, LDS, LES

An important type of data transfer operation is loading a segment and a general purpose register with an address directly from memory

Mnemonic	Meaning	Format	Operation	Flags Affected
LEA	Load Effective Address	LEA Reg16,EA	EA → (Reg16)	None
LDS	Load Register and DS	LDS Reg16, MEM32	(Mem32) → (Reg16) (Mem32 + 2) → (DS)	None
LES	Load Register and ES	LES Reg16, MEM32	(Mem32) → (Reg16) (Mem32 + 2) → (ES)	None

Example. LDS SI, [200h]

## Arithmetic Instructions – ADD, ADC, INC, AAA, DAA

---

Mnemonic	Meaning	Format	Operation	Flags Affected
ADD	Addition	ADD D, S	(S) + (D) → (D) Carry → (CF)	All
ADC	Add with carry	ADC D, S	(S) + (D) + (CF) → (D) Carry → (CF)	All
INC	Increment by one	INC D	(D) + 1 → (D)	All but the carry flag
AAA	ASCII adjust for addition	AAA		
DAA	Decimal adjust for addition	DAA		

# Examples

---

**Ex. 1** ADD AX, 2  
ADC AX, 2

**Ex. 2** AL contains 32 (ASCII code for number 2)  
BL contains 34 (ASCII code for number 4)

ADD AL, BL  
AAA

; has to be adjusted via AL and only an addition instruction

**Ex. 3** AL contains 25 (packed BCD)  
BL contains 56 (packed BCD)

ADD AL, BL  
DAA

## Arithmetic Instructions – SUB, SBB, DEC, AAS, DAS, NEG

---

Mnemonic	Meaning	Format	Operation	Flags Affected
SUB	Subtract	SUB D, S	(D) - (S) → (D) Borrow → (CF)	All
SBB	Subtract with borrow	SBB D, S	(D) - (S) - (CF) → (D)	All
DEC	Decrement by one	INC D	(D) - 1 → (D)	All but the carry flag
NEG	Negate	NEG D		
DAS	Decimal adjust for subtraction	AAA		
AAS	ASCII adjust for subtraction	DAA		

## Example

---

- 32-bit subtraction of 2 32 bit numbers X and Y that are stored in the memory as
  - X = (DS:203h)(DS:202h)(DS:201h)(DS:200h)
  - Y = (DS:103h)(DS:102h)(DS:101h)(DS:100h)
- The result X = Y to be stored where X is saved in the memory

MOV SI, 200h

MOV DI, 100h

MOV AX, [SI]

SUB AX, [DI]

MOV [SI], AX ;save the LS word of result

MOV AX, [SI] +2

SBB AX, [DI]+2

MOV [SI] +2, AX

Ex. 12 34 56 78 – 23 45 67 89 = EF EE EE EE

# Multiplication and Division

---

<b>Multiplication (MUL or IMUL)</b>	<b>Multiplicant</b>	<b>Operand (Multiplier)</b>	<b>Result</b>
Byte * Byte	AL	Register or memory	AX
Word * Word	AX	Register or memory	DX :AX
Dword * Dword	EAX	Register or Memory	EDX :EAX

<b>Division (DIV or IDIV)</b>	<b>Dividend</b>	<b>Operand (Divisor)</b>	<b>Quotient : Remainder</b>
Word / Byte	AX	Register or memory	AL : AH
Dword / Word	DX:AX	Register or memory	AX : DX
Qword / Dword	EDX: EAX	Register or Memory	EAX : EDX

# AAM, AAD, CBW, CWD

---

- AAM: Adjust AX for multiply
- AAD: Adjust AX for divide
  - MOV AL, 07 ; first unpacked number
  - MUL AL, 09 ; second unpacked number
  - AAM ; AX should have 06 03
- Division instructions can also be used to divide an 8 bit dividend in AL by an 8 bit divisor.
  - In order to do so, the sign of the dividend must be extended to fill the AX register
  - AH is filled with zeros if AL is positive
  - AH is filled with ones if the number in AL is negative
- Automatically done by executing the CBW (convert byte to word) instruction
- CWD (convert word to double word)
  - Ex. MOV AL, 0A1h
  - CBW
  - CWD

# Example

---

- Write a program that calculates the average of five temperatures and writes the result in AX

```
DATA DB    +13,-10,+19,+14,-18    ; 0d, f6,13,0e, ee
        MOV    CX,5                ;LOAD COUNTER
        SUB    BX,BX                ;CLEAR BX, USED AS ACCUMULATOR
        MOV    SI,OFFSET SIGN_DAT ;SET UP POINTER
BACK:    MOV    AL,[SI]             ;MOVE BYTE INTO AL
        CBW                          ;SIGN EXTEND INTO AX
        ADD    BX,AX                ;ADD TO BX
        INC    SI
        DEC    CX                    ;INCREMENT POINTER
        JNZ    BACK                 ;LOOP IF NOT FINISHED
        MOV    AL,5                  ;MOVE COUNT TO AL
        CBW                          ;SIGN EXTEND INTO AX
        MOV    CX,AX                ;SAVE DENOMINATOR IN CX
        MOV    AX,BX                ;MOVE SUM TO AX
        CWD                          ;SIGN EXTEND THE SUM
        IDIV   CX                   ;FIND THE AVERAGE
```

# Compare (CMP)

---

Write a program to find the highest among 5 grades and write it in DL

```
DATA    DB    51, 44, 99, 88, 80        ; 13h,2ch,63h,58h,50h
        MOV    CX,5                    ;set up loop counter
        MOV    BX, OFFSET DATA ;BX points to GRADE data
        SUB    AL,AL                    ;AL holds highest grade found so far
AGAIN:  CMP    AL,[BX]                  ;compare next grade to highest
        JA     NEXT                     ;jump if AL still highest
        MOV    AL,[BX]                  ;else AL holds new highest
NEXT:   INC    BX                        ;point to next grade
        LOOP  AGAIN                     ;continue search
        MOV    DH, AL
```

# Logical Instructions

---

- AND
  - Uses any addressing mode except memory-to-memory and segment registers
  - Especially used in clearing certain bits (masking)
    - xxxx xxxx **AND** 0000 1111 = 0000 xxxx (clear the first four bits)
  - Examples: AND BL, 0FH; AND AL, [345H]
- OR
  - Used in setting certain bits
    - xxxx xxxx **OR** 0000 1111 = xxxx 1111
- XOR
  - Used in inverting bits
    - xxxx xxxx **XOR** 0000 1111 = xxxx yyyy
- **Ex.** Clear bytes 0 and 1, set bits 6 and 7, invert bit 5

```
AND CX, 0FCH  1111 1100
OR CX, 0C0H   1100 0000
XOR CX, 020H  0010 0000
```

# TEST

---

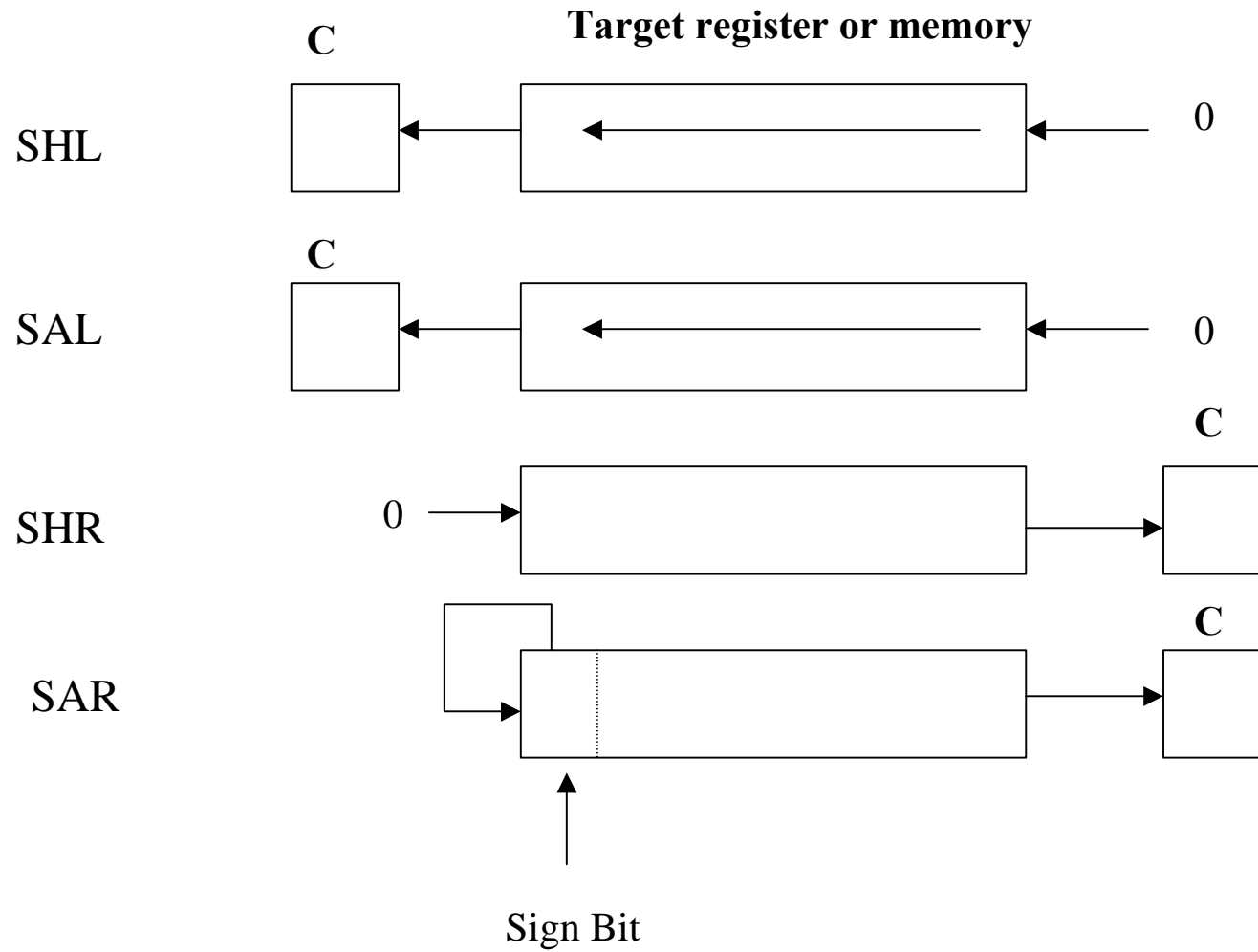
- TEST instruction performs the AND operation but it does not change the destination operand as in AND but only the flags register.
- Similar to CMP bit it tests a single bit or occasionally multiple bits.
- **Ex.** TEST DL, DH ; TEST EAX, 256

```
TEST AL, 1 ; test right bit
JNZ RIGHT ; if set
TEST AL, 128 ; test left bit
JNZ LEFT ; if set
```

- Bit Test Instructions (80386 thru Pentium 2) BT/BTC/BTR/BTS
  - BT AX,4 tests bit position 4 in AX and result is loaded in the carry flag, C = 1 if bit 4 is 1, 0 otherwise

```
BTS CX, 9 ; tests and sets bit 9
BTR CX, 10 ; tests and resets bit 10
BTC CX, 12 ; tests and complements bit 12
```

# Shift



# Examples

---

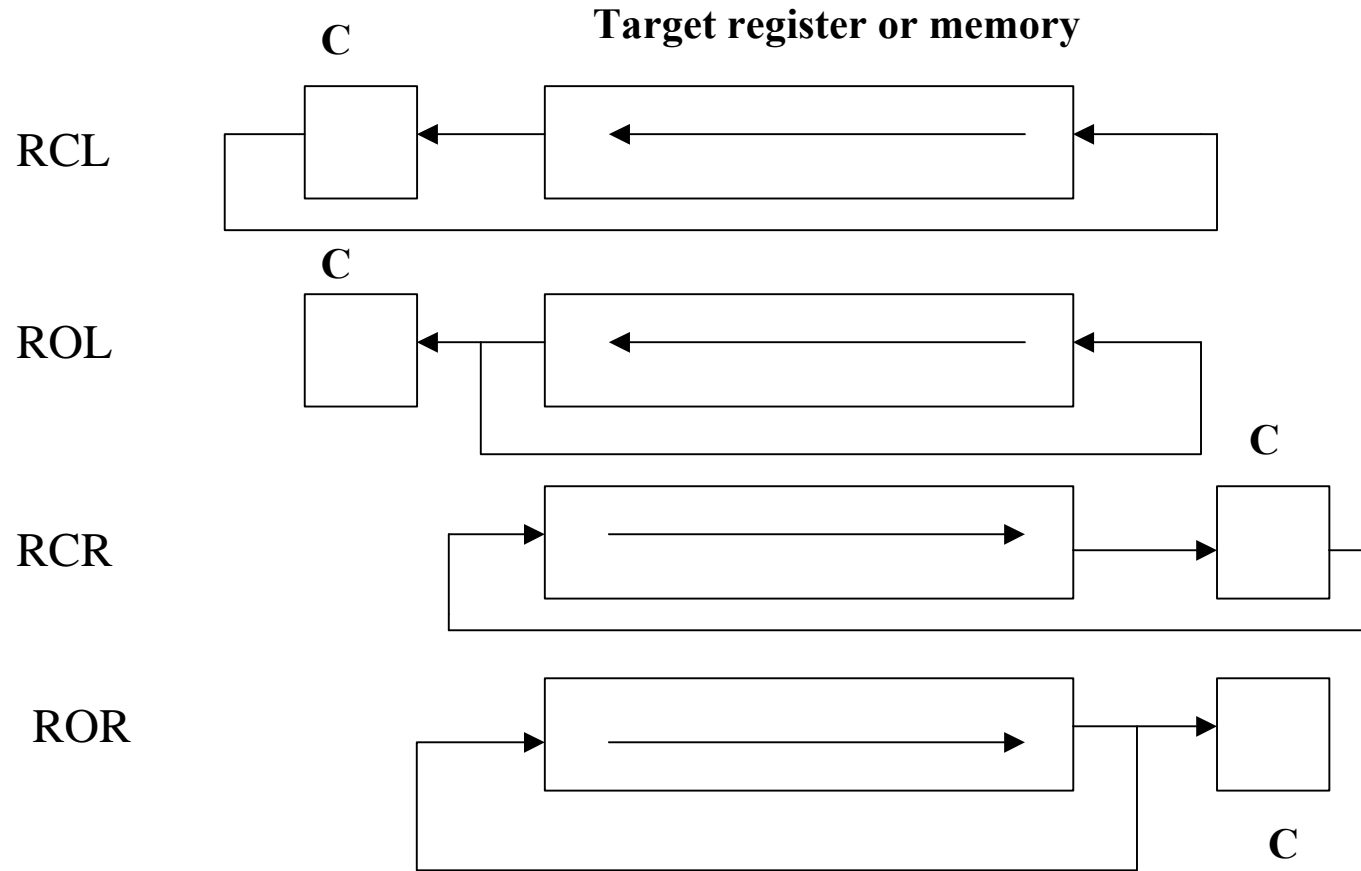
Examples    SHL AX,1  
              SHR BX,12  
              SAL DATA1, CL ; shift count is a modulo-32 count  
              SAR EDX, 14

Ex.           ; Multiply AX by 10  
              SHL AX, 1  
              MOV BX, AX  
              SHL AX,2  
              ADD AX, BX

Ex.           What are the results of SAR CL, 1 if CL initially contains B6H?

Ex.           What are the results of SHL AL, CL if AL contains 75H  
              and CL contains 3?

# Rotate



**Ex.** What is the result of ROL byte ptr [SI], 1 if memory location 3C020 contains 41H?

# Example

---

;write a program that counts the number of 1's in a byte and writes it into BL

```
DATA1 DB 97                ; 61h
      SUB  BL,BL            ;clear BL to keep the number of 1s
      MOV  DL,8             ;rotate total of 8 times
      MOV  AL,DATA1
AGAIN: ROL  AL,1            ;rotate it once
      JNC  NEXT            ;check for 1
      INC  BL              ; if CF=1 then add one to count
NEXT:  DEC  DL              ;go through this 8 times
      JNZ  AGAIN          ;if not finished go back
      NOP
```

# BCD and ASCII Numbers

---

- BCD (Binary Coded Decimal)
  - Unpacked BCD: One byte per digit
  - Packed BCD: 4 bits per digit (more efficient in storing data)
- ASCII to unpacked BCD conversion
  - Keyboards, printers, and monitors all use ASCII.
  - Digits 0 to 9 are represented by ASCII codes 30 – 39.
- **Example.** Write an 8086 program that displays the packed BCD number in register AL on the system video monitor
  - The first number to be displayed should be the MSD
  - It is found by masking the LSD and then rotating the MSD into the LSD position
  - The result is then converted to ASCII by adding 30h
  - The BIOS video service is then called

# Program

---

```
Mov bl,al  
And al,f0h  
Mov cl,4  
Ror al,cl  
Add al,30h  
Mov ah,0eh  
Int 10h
```

```
Mov al,bl  
And al,0fh  
Add al,30h  
Int 10h  
Int 20h ; return to DOS
```

# Example

---

- Write an 8086 program that adds two packed BCD numbers input from the keyboard and computes and displays the result on the system video monitor
- Data should be in the form 64+89= The answer 153 should appear in the next line.

Mov dx, buffer address

Mov ah,0a

Mov si,dx

Mov byte ptr [si], 8

Int 21

Mov ah,0eh

Mov al,0ah

Int 10 ; BIOS service 0e line feed position cursor

## Example Continued

---

```
sub byte ptr[si+2], 30h  
sub byte ptr[si+3], 30h  
sub byte ptr[si+5], 30h  
sub byte ptr[si+6], 30h
```

```
Mov cl,4  
Rol byte ptr [si+3],cl  
Rol byte ptr [si+6],cl  
Ror word ptr [si+2], cl  
Ror word ptr [si+2], cl
```

```
Mov al, [si+3]  
Add al, [si+6]  
Daa  
Mov bh,al  
Jnc display  
Mov al,1  
Call display  
Mov al,bh  
Call display  
Int 20
```