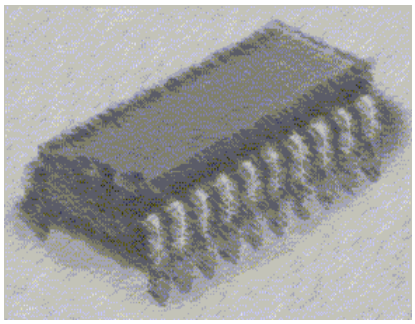


PRACTICE 4:
Machine and assembly languages

Objectives:

When finishing this practice, students are able to relate machine and assembly languages, changing from one to another and knowing addressing modes and memory data storage in deep.

Medios:

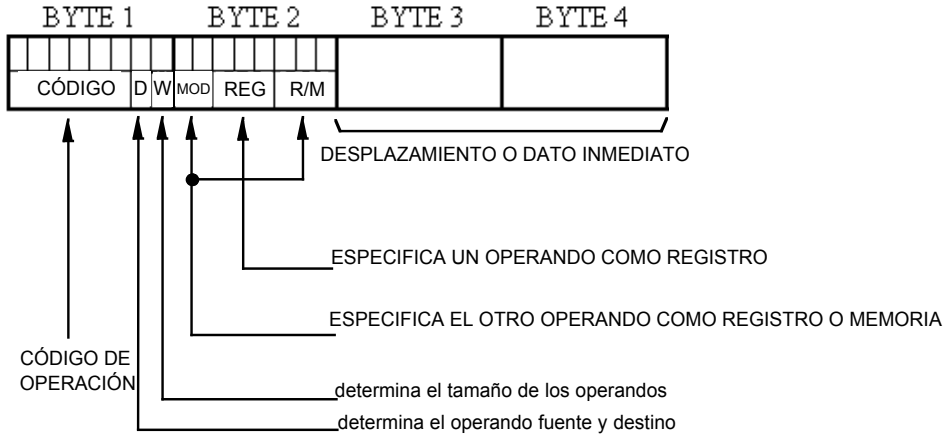
Microsoft Assembler 5.1 is required to program the practice.

INTRODUCCIÓN

i80x86 machine language is used in this practice. Instruction format will be studied. Effective and physical memory addresses will be differentiated

INSTRUCTION FORMAT

i80x86 register-register, register-memory instruction format is shown bellow.



REG	W=0	W=1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI
REG codification table		

MOD = 11			Effective Address			
R/M	**W = 0**	**W = 1**	**R/M**	MOD = 00	**MOD = 01**	**MOD =10**
000	AL	AX	000	[BX]+[SI]	[BX]+[SI] + Shift.8	[BX]+[SI] + Shift.16
001	CL	CX	001	[BX]+[DI]	[BX]+[DI] + Shift.8	[BX]+[DI] + Shift.16
010	DL	DX	010	[BP]+[SI]	[BP]+[SI] + Shift.8	[BP]+[SI] + Shift.16
011	BL	BX	011	[BP]+[DI]	[BP]+[DI] + Shift.8	[BP]+[DI] + Shift.16
100	AH	SP	100	[SI]	[SI] + Shift.8	[SI] + Shift.16
101	CH	BP	101	[DI]	[DI] + Shift.8	[DI] + Shift.16
110	DH	SI	110	Dirección directa	[BP] + Shift.8	[BP] + Shift.16
111	BH	DI	111	[BX]	[BX] + Shift.8	[BX] + Shift.16
R/M codification table depending on used addressing mode MOD						

PRACTICE 4 ACTIVITIES

Exer. #.	Exercise

1	Write, assemble, link and execute the following code:
	dosseg
	.model small.
	.stack 100h
	.data
	Texto DB 'Please, enter a hexadecimal number (maximum two digits) 0 y 9$'
	.code
	Inicio:
	mov ax, @data
	mov ds, ax
	
	mov ah, 9
	lea dx, Texto
	int 21h
	xor bl, bl
	mov ah, 1
	int 21h
	
	mov cl, 4
	mov bl, al
	sub bl, 30h
	shl bl, cl
	
	int 21h
	sub al, 30h
	add bl, al
	
	mov ah, 4Ch
	int 21h
	END Inicio

2	Change view in the Code View program to mixed view: 45B0:0010 B8B345 MOV AX,45B3 45B0:0013 8ED8 MOV DS,AX 45B0:0015 B409 MOV AH,09 45B0:0017 8D160600 LEA DX,Word Ptr [0006] 45B0:001B CD21 INT 21 **45B0:001D 32DB XOR BL,BL** **(Continue)**

	(Continue) 45B0:001F B401 MOV AH,01 45B0:0021 CD21 INT 21 45B0:0023 B104 MOV CL,04 45B0:0025 8AD8 MOV BL,AL 45B0:0027 80EB30 SUB BL,30 45B0:002A D2E3 SHL BL,CL 45B0:002C CD21 INT 21 45B0:002E 2C30 SUB AL,30 45B0:0030 02D8 ADD BL,AL 45B0:0032 B44C MOV AH,4C 45B0:0034 CD21 INT 21 Left column information means Segment:EffectiveAddress Machine Code. E.g. 45B0:0034 CD21 means r CS=45B0h, instruction effective address will be 0034h (IP value when execute instruction) physical memory address will be 45B0x10h + 0034h = 45B34h in such memory position machine code of INT 21h instruction (CD21h) can be found Run program, step by step and show how IP value change till XOR BL, BL instruction will be reached. ¿What will IP value be to execute the same instruction again? Change IP value with the content which must have to repeat INT21h instruction. (*R IP new value*) Execute instruction again. See it by F4 depressing.
3	Which will be IP value to end program without reading any number? Help: modify IP value to point to end instructions routine.
4	Machine code of LEA DX, Texto is 8D160600. Which is the Texto variable effective address? Which will the effective address to start message in the word *número*?
5	Modify the effective address to star text in *como* word..
6	Despite the program doesn't work, please change source and target register of the MOV BL, AL instruction 8 A D 8 1000 1010 1101 1000 Operation code = 100010 Bit D = 1 (target reg) Bit W = 0 (8 bits) Mod = 11 (register-register) Reg = 011 (BL) R/M = 000 (AL)

	How can MOV BL, AL instruction be changed to MOV AL, BL instruction by modifying only the second byte? (EB 0x45B0:0x0026 *new value*)
7	¿And changing only the first byte? (EB 0x45B0:0x0025 *new value*)

PRACTICE 4
Machine and assembly language

Next program it's a typical login password multiuser system access. Please, write the code and change it as required.

```
DOSSEG
.MODEL SMALL
.STACK 100h
.DATA
      Usuarios      DB "ANTONIO GOMEZ GOMEZ%A23$"
                    DB "LUISA ALONSO LOPEZ%A1SA3$"
                    DB "FERNANDO PEREZ MINGUEZ%2W45$"
                    DB "JOSEFA RUIZ SANCHEZ%ASQ12$"
                    DB "MIGUEL GARCIA GARCIA%S1A$"
      LonUsua       EQU $-Usuarios
      LonParc       DB 5 DUP()          ;Permite indexar la tabla de
                                        ;usuarios guardando la longitud
                                        ;de cada entrada
      Mensaje1      DB "Nombre de usuari@: $"
      Mensaje2      DB "Por favor, introduzca su clave de acceso: $"
      Mensaje3      DB "Bienvenid@ al sistema.$"
      Mensaje4      DB "Nombre de usuari@ o palabra clave incorrectos.$"
      Mensaje5      DB "Ha sobrepasado el numero de intentos de entrada."
                    DB "El sistema se bloquea.",10,13,"$"
      BufUsu        DB 31           ;Buffer para guardar el usuario
      LonUsu        DB 0
                                                  Usuari        DB 31 DUP (?)

      LonCla        DW 0            ;Buffer to store  password
      Clave         DB 5 DUP (?)
      NumInt        DB 0            ;Number of failed atempts
```

(**Continue)**

(**Continue)**

```
.CODE
;Macro to change video mode
ModoVideo MACRO modo
        mov ah,0
        mov al,modo
        int 10h
        ENDM

;Macro to change cursor position
PonCursor MACRO fil,col
        mov ah,2
        xor bx,bx
        mov dh,BYTE PTR fil
        mov dl,BYTE PTR col
        int 10h
        ENDM

;Procedure to display a string on the screen
SacaMens  PROC
        push bp
        mov bp,sp
        push dx
        push ax
        mov ah,9
        mov dx,[bp+4]
        int 21h
        pop ax
        pop dx
        pop bp
        RET  2
SacaMens  ENDP


Inicio: mov ax,@DATA
       mov ds,ax

;Ask for username and password
otrave: ModoVideo 3
       PonCursor 10,15
       lea ax, Mensaje1
       push ax
       call SacaMens
       lea dx, BufUsu
       mov ah,0Ah
       int 21h
       PonCursor 12,15
       lea ax, Mensaje2
       push ax
       call SacaMens
       lea bx, Clave     ;reading passwor
       xor si,si
```

(**Continue)**

(**Continue)**

```
lazo1:  mov ah,8
        int 21h
        mov [bx+si],al
        cmp al,13
        je salir
        mov ah,0Eh
        mov al,'*'
        int 10h
        inc si
        cmp si,5
        jbe lazo1
salir:  mov LonCla,si          ;password length stored
;Indexar tabla de usuarios
        xor si,si
        xor di,di
        mov dl,1               ;each user type is taken into account to calculate length
        lea bx, Usuarios
        mov cx,LonUsua
        mov al,'$'
lazo2:  cmp al,[bx+si]
        jne seguir
        mov LonParc[di],dl
        xor dl,dl
        inc di
seguir: inc si
        inc dl
        loop lazo2
;Check username with allowed stored names
        lea bx, Usuarios
        xor di,di
        xor si,si
        xor cx,cx
        mov cl,LonParc                ;First entry length
otrous: xor si,si
lazo3:  mov al,Usuari[si]
        cmp al,20h                    ;If space conversion is not required
        je saltar
        and al,11011111b              ;Username uppercase conversion
saltar: cmp al,[bx+si]
        jne salfin
        inc si
        loop lazo3
salfin: cmp BYTE PTR[bx+si],'%'
        je nomcorr
        mov cl,LonParc[di]
        add bx,cx
        inc di
        cmp bx,LonUsua         ;Ends
        jb otrous
        jmp nominc
```

(**Continue)**

(**Continue)**

```
;Correct name, then password chacking
nomcorr:
        xor cx,cx
        mov cl,LonParc[di]      ;Password length
        sub cx,si               ;(LonParc[di]-si)-2
        dec cx
        dec cx
        cmp cx,LonCla           ;password too long
        jb nominc               ;despite of first character matching
        inc si
        xor di,di
lazo4:  mov al,Clave[di]
        cmp al,[bx+si]
        jne nominc
        inc si
        inc di
        loop lazo4
;Display welcome message
        PonCursor 14,15
        lea ax, Mensaje3
        push ax
        call SacaMens
        jmp final
;Display Username or password incorrect. Block access
nominc: PonCursor 14,15
        lea ax, Mensaje4
        push ax
        call SacaMens
        mov al,NumInt
        inc al
        cmp al,3
        jae bloquea
        mov NumInt,al
        jmp otrave
bloquea:PonCursor 16,4
        lea ax, Mensaje5
        push ax
        call SacaMens
        jmp bloquea
final:  mov ah,4Ch
        int 21h
        END Inicio
```

Please, do next activities:

1.- Convert *PonCursor* macro into a procedure. Row and column parameters must be passed through the stack.

2.- Convert *SacaMens* procedure into a macro. String address is passed as parameter.

3.- JE machine code is 74h and the JNE one is 75h. The effective address that must be added to IP is code in the next byte. Please, make necessary changes in the machine code so the program jumps the next *Salir*: *XOR DI, DI* instruction. Run the program and see the changes

4.- Change first byte of the *add bx,cx* instruction (after *Salfin* label) so source and target operand will be inverted.

5.- Change the second bye of the machine code to obtain the same effect as above.

6.- Change first byte of the machine code of the *je nomcorr* instruction (after *Salfin* label) to jump to the Welcome message when user introduces an incorrect password.

7.- Change second byte of the machine code of the *je nomcorr* instruction (after *Salfin* label) to jump to the User or password incorrect when the introduced password is correct.

8.- Change the effective address in the machine code of the character string to start in *“clave de acceso”* when executing *Sacamens* procedure.

9.- Change effective address when using 0Ah function of the 21h intterrupt, to point to *Usuari* instead of *BufUsu.*