

Universidad Autónoma de Baja California
Facultad de Ciencias Químicas e Ingeniería



Alumno: Eduardo Marcelo Gutiérrez Soto
Profesora: Evangelina Lara Camacho
Practica 7

Resumen sobre conversiones numéricas

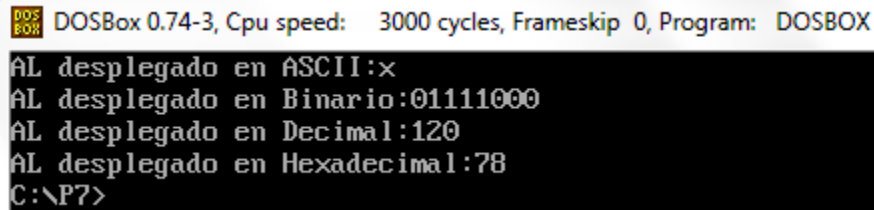
Las conversiones numéricas nos sirven para poder representar un numero de X base en otro numero en X base, como podemos conocer hay una gran variedad de bases numéricas por ejemplo para nombrar algunas.

- Decimal(10)
- Binario(2)
- Hexadecimal(16)
- Octal(8)

estas son las bases numéricas mas comúnmente utilizadas y como podemos saber cada una tiene una representación distinta de los números y cuando queremos saber el valor de un numero de X base en otra base, para eso se realiza una serie de cálculos para poder hacer una conversión de un numero a otro hay mas tipos de bases numéricas pero estas son las que mas se utilizan por lo general, hay una manera de como convertir cualquier base numérica a otra base y es mediante divisiones sucesivas y utilizando como primero una conversión a decimal y ese numero empezarlo a dividir, es una forma muy practica y es la que suelo utilizar para hacer conversiones de cualquier tipo de base. Y al final se utilizan los que son los residuos para poder formar el numero final.

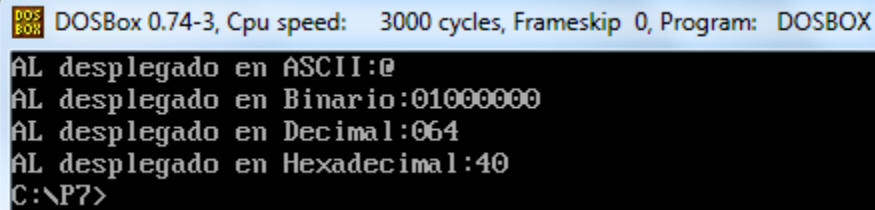
1. Ensamble, encadene y ejecute el programa Prac7.asm que se muestra en el listado 1. El programa realiza un despliegado en pantalla del valor del registro AL en formato binario y hexadecimal. Ensamble y encadene el programa para diferentes valores de AL.

Valor 78H



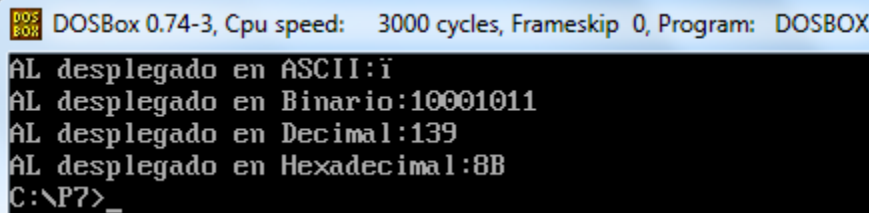
```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
AL desplegado en ASCII:x
AL desplegado en Binario:01111000
AL desplegado en Decimal:120
AL desplegado en Hexadecimal:78
C:\P7>
```

Valor 40H



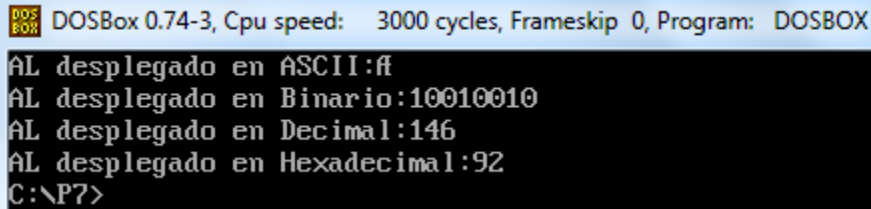
```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
AL desplegado en ASCII:@
AL desplegado en Binario:01000000
AL desplegado en Decimal:64
AL desplegado en Hexadecimal:40
C:\P7>
```

Valor 8BH



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
AL desplegado en ASCII:i
AL desplegado en Binario:10001011
AL desplegado en Decimal:139
AL desplegado en Hexadecimal:8B
C:\P7>_
```

Valor 92H



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
AL desplegado en ASCII:f
AL desplegado en Binario:10010010
AL desplegado en Decimal:146
AL desplegado en Hexadecimal:92
C:\P7>
```

2. En un archivo diferente y siguiendo la plantilla formato.asm diseñe e implemente el procedimiento printNumBase, el cual imprime la palabra dada en el registro AX en el formato según la base dada en el registro BX.

Código:

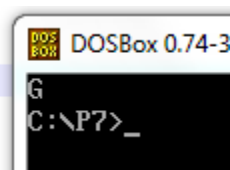
```
-----procedimientos-----
;Procedimiento que recibe una palabra dada en AX y la base a convertir en BX
printNumBase PROC
    push ax
    push bx
    push cx
    push dx

    mov cx,0
    mov dx,0
@@while:cmp ax,0
    je @@end_while
    div bx ;dividimos el numero con la base ingresada
    push dx ;guardamos el residuo a la pila
    inc cx ;incrementamos cx para hacer uso de un loop
    xor dx,dx ;ponemos en cero dx
    jmp @@while
@@end_while:
@@nxt:
    pop dx ;sacamos el residuo de la pila
    add dx,30h ;ajustamos a ascii
    cmp dx,'9' ;comparamos el numero con un 9
    jbe @@print ;
    add dx,7h ;si es mayor a 9 lo ajusta para imprimir las letras correspondientes a HEX
@@print:mov al,dl ;mandamos el dato a desplegar al registro al
    call putchar;
    loop @@nxt

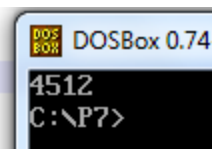
    pop dx
    pop cx
    pop bx
    pop ax
    ret
ENDP
```

Ejecución:

```
mov ax,10h
mov bx,17
call printNumBase
```



```
mov ax,94ah
mov bx,8
call printNumBase
```



hicimos el procedimiento de conversión de manera que sea genérica para cualquier numero y base ingresada por el usuario.

3. Programe el procedimiento atoi, el cual recibe una cadena en BX un apuntador a una cadena terminada en null, que convierte una serie de números ASCII en formato decimal, la cadena es convertida a su correspondiente valor numérico regresandolo en el registro AX.

Ejecución:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
probando la funcion atoi AX=1234 y en HEX AX=04D2
AL desplegado en ASCII:f
AL desplegado en Binario:10010010
AL desplegado en Decimal:146
AL desplegado en Hexadecimal:92
C:\P7>
```

Código:

```

;-----procedimientos-----
;atoi recibe una cadena en bx y retorna el entero en AX
atoi PROC
    push cx          ;salvamos registros
    push bx
    push si
    mov si,bx        ;movemos el valor de bx a si
    xor ax,ax         ;limpiamos el registro AX
    mov cx,10         ; movemos 10 decimal al registro cx
    xor bx,bx         ;limpiamos el registro BX

@@nxt:  cmp byte ptr[si],0 ;verificamos que no sea NULL el caracter de la cadena
        je @@fin          ; si lo es salimos
        mov bl,[si]        ;movemos un byte de CS:BX a bl
        sub bl,30h         ;le restamos el valor ASCII
        mul cx             ;multiplicamos el registro para hacer unidades, decenas, centenas, etc
        add ax,bx          ;le sumamos el dato al registro ax
        inc si             ;apuntamos ahora a la siguiente direccion de la cadena
        jmp @@nxt          ;saltamos a la etiqueta nxt

@@fin:
    pop si
    pop bx
    pop cx
    ret
ENDP
;*****

```

Verificación de numero para el funcionamiento de ATOI

```

mov dx,offset mens
call puts
mov bx,offset num      ;mandamos el offset de la cadena por BX
call atoi              ;llamamos la funcion atoi
cmp ax,1234            ;comparamos el dato convertido a decimal con atoi
jne @@FIN              ;si el dato no es el mismo que el de la cadena se termina el programa
mov dx,offset mens2    ;caso contrario imprime que el dato si es igual
call puts

```

4. Programe el procedimiento esAutomatico, el cual recibe una palabra en el registro DX y retorna un 1 en AX si el numero es automorfico, caso contrario retorna un 0.

Creamos un código para verificar si un numero es automorfico para eso verificaremos los números automorficos que hay hasta el numero soportado por el 8088 que es el 9376.

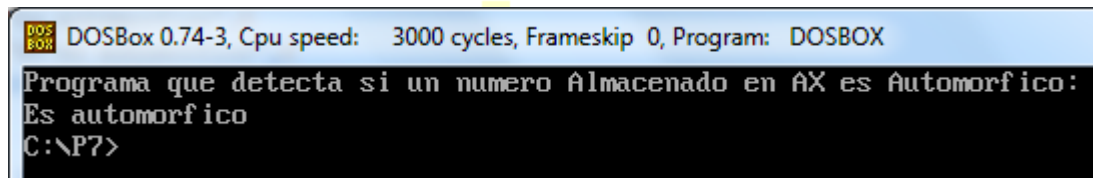
Código.

```
;procedimiento que detecta si un numero es automorfico recibe una palabra en DX
;si lo ex AX retorna 1 si no lo es 0
esAutomatico PROC
    push bx
    push cx ;salvamos registros a utilizar
    xor ax,ax ;aseguramos el registro AX en cero
    xor bx,bx ;aseguramos BX en cero
    mov ax,dx ;movemos el valor de DX a AX
    mov bx,dx ;respaldamos el dato de DX en BX
    mov cx,10 ;movemos un 10 al registro para hacer divisiones de 10
    cmp ax,9 ;comparamos el numero ingresado que sea menor que 9 para poder hacer divisiones de 10
    ja @@nxt ;si esta arriba es que ax es mayor a 9
    mul ax ;elevamos el numero almacenado en ax^2
    div cx ;dividimos el numero por 10
    cmp bx,dx ;comparamos el residuo que esta en DX con BX si es igual sigue el procedimiento
    jne @@no_es
    mov ax,1 ;es amorfico
    jmp @@si_es
@@nxt: cmp ax,99 ;comparamos el numero con 99 para dividirlo entre 100 si es menor se divide entre 100
    ja @@nxt1 ;si no es verificamos el numero con millares
    mov cx,100 ;movemos un 100 a cx para dividirlo
    mul ax ;elevamos ax^2
    div cx ;dividimos el numero entre 100
    cmp bx,dx ;comparamo el numero original con el residuo
    jne @@no_es ;si no es igual mandamos un 0 por AX
    mov ax,1 ;si lo es mandamos un 1
    jmp @@si_es
@@nxt1: cmp ax,999 ;comparamos el numero con 999 para dividirlo entre 1000 si es menor se divide
    ja @@nxt2
    mov cx,1000 ;movemos a cx 1000
    mul ax ;elevamos ax^2
    div cx ;lo dividimos entre 1000
    cmp bx,dx ;comparamos si el numero anteriormente ungresado es igual a el residuo
    jne @@no_es
    mov ax,1
    jmp @@si_es
@@nxt2: cmp ax,9999
    ja @@no_es ;brincamos directamente a no es por que nuestro procesado nos limita a dos bytes
    mul ax ;se eleva al AX^2 y como resulta a una multiplicacion de 16 bits nos de un num de 32 DX-AX
    and ax,2400
```

Ahora veremos una serie de ejemplos con los números que son automorficos.

Ejemplo con 5:

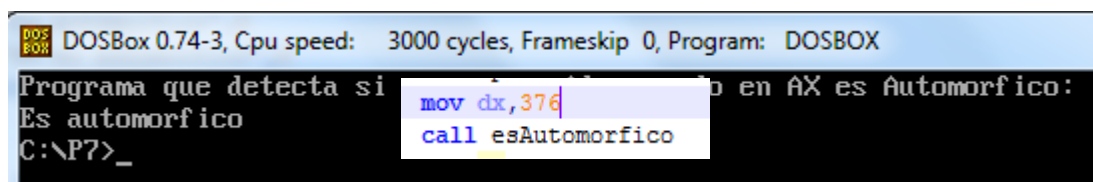
```
mov dx,5  
call esAutomorfico
```



Si el numero no fuera automorfico mandaría por AX un 0

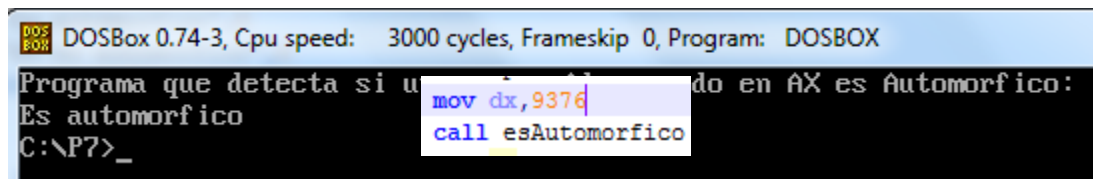
Ejemplo con 25:

```
mov dx,25  
call esAutomorfico
```



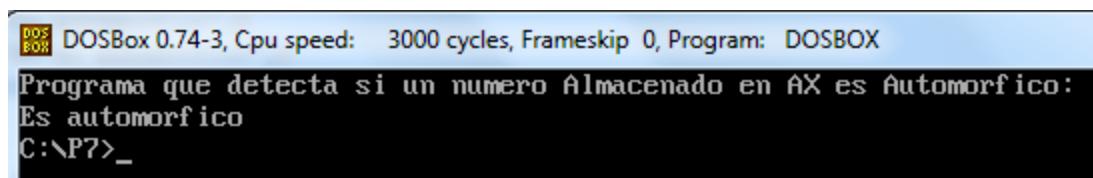
Ejemplo con 376:

```
mov dx,9376  
call esAutomorfico
```



Ejemplo con 9376:

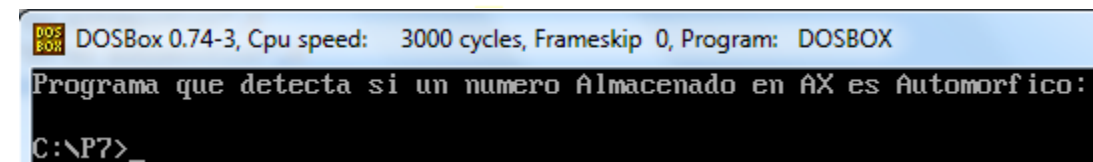
```
mov dx,44  
call esAutomorfico
```



Ejemplo con numero no automorfico:

```
mov dx,44  
call esAutomorfico
```

si no es



automorfico solo se sale del programa y manda en AX = 0.

Conclusión

con esta practica logramos darle un uso mas a la lógica para programar en ensamblador y el como interpreta los valores el procesador, creamos una serie de programas muy interesantes como lo es el atoi que te convierte una cadena de números a su valor decimal y el de los números automorficos teniendo en cuenta las limitaciones del procesador lo logramos hacer, la única complicación fue en el printNumBase pero no fue tanto una dificultad si no un descuido yo sabia que el código estaba bien y hasta lo desensable con el debug para correrlo paso a paso y funcionaba bien ya que debug asume que los registros están en cero, entonces mi error fue de como hago divisiones de 16 bits pues nunca puse en cero el registro DX cuando empezaba el programa y por eso no funcionaba nada y nomas le puse MOV DX,0 y todo funciono correctamente.