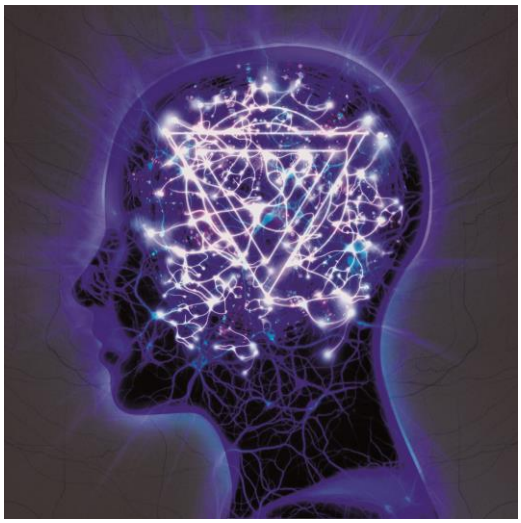
UNIVERSIDAD AUTÓNOMA

DEL ESTADO DE MÉXICO

CENTRO UNIVERSITARIO XICO VALLE DE CHALCO



# MACROS Y PROCEDIMIENTOS

Ensambladores

## Reporte [28-OCT-2019]

Reporte de la práctica 02 para día 28 de octubre en donde se tiene que hacer un código en ensamblador que reciba un número, lo guarde en un arreglo, que lo convierta a hexadecimal y después de nuevo a decimal usando divisiones, macros y procedimientos.


## Javier Enrique Trejo Rodriguez

ICO 0-6 Vespertino

# Índice

## Introducción

El reporte que está a punto de leer a continuación trata del programa de la práctica sobre los macros y los procedimientos en el lenguaje de ensamblador, su declaración y funcionamiento de ambos en un programa, en lo teórico se plantea el uso del seudocódigo y se explica por qué se emplea su uso y como se va desarrollando**,** en la práctica el seudocódigo es transcrito en lenguaje pertinente y compilado en DosBox para mostrar que este realmente funciona y cumple con lo establecido.

Se espera que sea fácil de entender, aún me queda mucho que aprender para tener una mejor manera de explicar todo sobre cualquier programa.

## Objetivo

El alumno estudiante de la materia de Ensambladores por medio de esta práctica y demás ejercicios aprenderá a manejar mejor el lenguaje en su generalidad, así como usa las cadenas, los macros, arreglos, procedimientos y los distintos registros y segmentos del lenguaje de programación Ensamblador de una forma teórica, pero sobre todo y mejor, práctica, para que le ayude en el ámbito laboral en un futuro cuando se dedique a esto de la programación.

## Programa.

## Desarrollo

Realizar un programa en ensamblador que lea un numero de tamaño n, lo guarde en un arreglo, descomponga el numero para “pasarlo a hexadecimal” y de nuevo pasarlo a decimal usando divisiones. El programa debe cumplir con los siguientes requisitos:

a) Ingresar un numero de tamaño >1 en consola.
b) Guardarlo en un arreglo
c) Pasarlo a hexadecimal.
d) Una vez en hexadecimal, volverlo a transformar en decimal

Para una mejor explicación del código, este se va a explicar de manera secuencial para evitar confusiones y posteriormente se mostrará el código haciendo uso de macros y de procedimientos.

## Código

Primero se pone el tamaño que va a ser el código y posterior a eso la librería .data para que funcione bien el programa. Después declaramos un segmento donde van a estar todos nuestros datos que se van a imprimir durante el transcurso de todo el código.

```
5     Cad01 db 10,13, 'TREJO RODRIGUEZ J
6     Cad02 db 10,13,10,13, 'Introduce u
7     Cad03 db 10,13,10,13, 'El numero i
8     Mult dw 1
9     MultD dw 10
10    Contador dw 0
11    Acumulador dw 0
12    ArNumero dw 100 dup(0)
13    ArNum dw 100 dup(0)
14    ArmuN dw 100 dup(0)
15    ArMul dw 100 dup(0)
16    ArMultiplicador dw 100 dup(0)
```

*(Img1. Segmento de datos)*

Las cadenas son los mensaejs que se van a mostrar durante el transcurso del programa.

La variable Mult se asigna en uno para ir llenando el arreglo de multiplicadores, pero posteriormente irá aumentando su valor x10.

Contador sirve para guardar el tamaño del numero ingresado y usarlo en todas las iteraciones

Acumulador se usa para guardar la suma de los números en hexaadecimal y posteriormente tambipen se usa para guardar los residuos de las divisiones aa la hora de volver a pasar los números a decimal.

ArNum se usa para guardar el número ingresado, ArmuN para girar ese arreglo y ArNumero sirve para guardar los cocientes de la división cuando se pasa a decimal de nuevo. ArMul se usa para guardar los multiplicadores dependiendo del tamaño del número ingresado, en la primera posición se va guarda un 1, en la segunda un 10, en la tercera un 100 y así suucesivamente y ArMultiplicador es el mismo arreglo peroinvertido, este último lo ocupamos para hacer la división.

Lo que sigue en el código es imprimir la cadena uno que lleva los datos del alumno, despues la cadena dos que es un mensae para que el usuario ingrese un número, después se limpian lor registros índice SI y DI.

En la siguiente imagen (Img2) se muestra la parte del código donde va recibiendo los números y los va guardando en el arreglo hasta que el cmp con enter sea verdadero.

En la línea 37 y 38 se asignan los arreglos pertinentes a un registro índice cada uno.

La etiqueta Leer comienza recibiendo un dato, lo compara con la tecla ENTER, le resta 30H al número ingresado y lo guarda en ArNum. Después limpia los registros Ax y Bx,

```
37    lea si, ArNum   ;Asigna
38    lea di, ArmuN
39
40    leer:
41        mov ah,1
42        int 21h
43        cmp al,13
44
45        je InvierteArreglo
46
47        sub al, 30h
48        mov ah,00h
49        mov [si],al
50
51        mov bx, 00h
52        mov ax, 00h
53        add si,2
54        inc cx
55
56        mov contador,00h
57        mov contador,cx
58
59        jmp leer
```

*(Img2. Segmento que recibe los datos)*

mueve una posición el arreglo e incrementa el contador en Cx

En las líneas 56 y 57 lo que hace es ir actualizando la variable Contador con Cx en cada iteración que se hace dentro de la etiqueta. Ejemplo: Si se ingresa el número 2498, Cx va a ir incrementándose y siempre se va a actualizar en la variable de Contador, en este caso quedaría un 4 en Contador.

En la siguiente imagen (Img3) se muestra a donde salta después que la comparación con la tecla ENTER es verdadera.

```
61      InvierteArreglo:
62          sub si,2
63          mov al,[si]
64          mov [di],al
65          add di,2
66      loop InvierteArreglo
```

Lo que hace es invertir el arreglo, si se metió el numero 2498, lo acomoda en otro arreglo de forma que quede 8942. Esto se hace con el fin de después hacer la multiplicacion para pasarlo a hexadecimal.

*(Img3. Loop invertir arreglo de números)*

En las siguientes líneas se limpian los registros SI, DI Cx, Ax y Bx, se le asigna a Cx el contador y se carga en Si el arreglo ArMul

En la siguiente imagen (Img4) se muestra el código y después se explica lo que hace cada línea a continuación:

```
77      LlenaArregloMultiplicador:
78          mov bx,Mult
79          mov [si],bx
80          mov bx,00h
81
82          mov ax, Mult
83          mul MultD
84          mov ax,ax
85          mov Mult,ax
86          add si,2
87      loop LlenaArregloMultiplicador
```

Cargamos en Bx la variable Mult que en esta primera iteración lleva un 1, se guarda en el registro Si que tiene a ArMul y se pasa alimpiar el registro Bx.

Ahora se carga en Ax la viariable Mult de nuevo, lo multiplicamos por MultD (que siempre tiene un 10), pasamos el valor del resultado a Ax que ahora tendría un 10 y lo guardamos en la variable Mult para que se actualice. En la siguient eiteración sevuelve a multiplicar po 10 y ahora Mult va a tener un 100 y así sucesivamente. Despues solo se aumenta la posición del arreglo en cada iteración.

Al final para un Contador de tamaño 4, el arreglo ArMul quedaría de la siguiente forma: [1] [10] [100] [1000] Pero se ve en hexadecimal de la siguiente forma: [1] [A] [64] [3E8].

En las siguientes líneas se vuelen a limpiar los registros SI, DI Cx, Ax y Bx, se le asigna a Cx el contador, se carga en Si el arreglo ArMul y en Di el arreglo ArmuN (que este último contiene los números ingresados invertidos).

En la siguiente imagen (Img5) se muestra un loop llamado Sacar y multplicar, se muestra y se explica a continuación:

```
 99      SacarYMultiplicar:
100          mov bx,[si]
101          mov ax,[di]
102          mul bx
103          mov ax,ax
104
105          mov [di],ax
106          add Acumulador,ax
107
108          add si,2
109          add di,2
110
111          mov ax,00h
112          mov bx,00h
113      loop SacarYMultiplicar
```

*(Img5. Saca números y multiplica)*

Recordamos que se cargó en Si el arreglo ArMul y en Di el arreglo ArmuN, entonces pasamos a Bx la primera posición de el ArMul que tiene un 1, en Ax lo mismo, pero con el arreglo de los números invertidos, que en este caso para el ejemplo anterior, tendría un 8. Se multiplica en la línea 102 y se carca el resultado en Ax en la línea siguiente.

En la línea 105 guardamos el resultado de esa multiplicación, se sobre escribe en Di en la posición donde estaba. En la línea siguiente se guarda el resultado en la variable de acumulador (en las siguientes iteraciones se suman los resultados con o que trae ya el acumulador para que al final vuelva a dar el número 2,498 y quede guardado todo en esa variable).

En las últimas dos líneas siguientes solo se limpian las variables Ax y Bx para usarlas de nuevo con seguridad y que no modifiquen nada mal en la siguiente iteración.

En las siguientes líneas se vuelven a limpiar los registros SI, DI Cx, Ax y Bx, se le asigna a Cx el contador, se carga en Di el arreglo ArMultiplicador y recordamos que Si tiene precargado el arreglo ArMult que tiene los multiplicadores.

En la siguiente imagen (Img6) se muestra el loop que se llama InvierteMultiplicador y su función es, como su nombre lo indica, invertir el arreglo de los números multiplicadores. Su funcionamiento se muestra a continuación:

```
123      InvierteMultiplicador:
124          sub si,2
125          mov ax,[si]
126          mov [di],ax
127          add di,2
128      loop InvierteMultiplicador
```

*(Im6. Invierte el arreglo multiplicador)*

Este loop solo invierte el arreglo, ya antes se había mostrado el funcionamiento de uno ciclo así.

Lo que hace este loop es sacar los multilicadores de Si y los va metiendo en el arreglo de Di de forma tal que quede algo asi: [1] [10] [100] [1000] -> [1000] [100] [10] [1].

Esto se hace con el fin de preparar el arreglo para el siguiente loop.

En las siguientes líneas se vuelven a limpiar los registros SI, DI Cx, Ax y Bx, se le asigna a Cx el contador, se carga en Si el arreglo ArMultiplicador y en Di el arreglo ArNumero que apenas se va a utilizar, ergo no tiene nada guardado.

En la siguiente imagen (Img7) se muestra el último loop del programa, lo que hace es la parte de convertir de hexadecimal a decimal de nuevo y guardarlo en el arreglo ArNumero.

```
140     DivideSuma:
141         mov ax,Acumulador
142         mov bx,[si]
143         div bx
144
145         mov ax,ax
146         mov bx,dx
147
148         mov Acumulador,00h
149         mov Acumulador,bx
150
151         add ax,30h
152         mov [di],ax
153
154         add si,2
155         inc di
156
157         mov ax,00h
158         mov bx,00h
159         mov dx,00h
160     loop DivideSuma
```

*(Im7. Pasa de hexadecimal a decimal)*

El loop empieza asignándole el valor de acumulador al registro Ax [Recordando que acumulador tiene el resultado de la sumatoria de los números multiplicados en exadecimal, es decir, tiene el 2,498 o en hexadecimal se vería como 9C2].

En la siguiente línea pasa a Bx el valor del arreglo Si, que en este caso es el multiplicador invertido, es decir, se le pasa 1,000 Bx.

En la línea 143 se hace la división de Bx entre Ax, el cociente se queda guardada en Ax y el residuo por alguna razón queda en Dx, ero en la línea 146 lo pasamos a Bx.

En las líneas 148 y 149 se actualiza acumulador, primero se limpia en ceros y después se le asigna el valor del residuo que está en Bx.

En la línea 151 se le suma 30H al cociente de la división y en la siguiente se mete el numero a la posición pertinente del arreglo ArNumero. En la línea 154 se mueve la posición del arreglo multiplicador para que en la siguiente iteración se saque el siguiente valor, y en la línea siguiente se hace lo mismo para Di.

En las últimas tres líneas del loop se limpian todos los registros para un uso eficiente en las siguientes iteraciones.

```
162         mov [di], byte ptr"$"
```

En esta línea se le pone un “$” al final del arreglo que tiene guardado el número después de la división, esto se hace para que a la hora de imprimir no saque basura.

En las últimas líneas solo imprime un mensaje y después imprime el numero guardado en el areglo ArNumero después de hacer la división.

Para pasarlo a macros y procedimientos solo se tiene que ir dividiendo el código original, poner las etiquetas de los macros y ordenar todo en los procedimientos tal como se vio en clase. El código final queda de la siguiente manera:

```
include PP1013M.lib
.model small
.data

Datos

.stack
.code

Inicio:
        call PIDE_DATOS
        call HEXADECIMAL
        call DECIMAL
        ImprimeResultado ax,dx,Cad03,ArNumero

        PIDE_DATOS:
                Cargar
                Imprime Cad01, Cad02
                Limpia_SIDICX si,di,cx
                lea si, ArNum
                lea di, ArmuN
                leer:
                        mov ah,1
                        int 21h
                        cmp al,13
                        je InvierteArreglo
                        LeerMac ax,bx,cx,si
                        jmp leer
                InvierteArreglo:
                        Invertir ax,si,di
                loop InvierteArreglo
                Limpia_Todo si,di,cx,ax,bx
                mov cx,contador
                lea si,ArMul
                LlenaArregloMultiplicador:
                        LlenaArregloMultiplicadorMacro
                        Mult,MultD,ax,bx,si
                loop LlenaArregloMultiplicador
        RET

HEXADECIMAL:
                Limpia_Todo si,di,cx,ax,bx
                mov cx,contador
                lea si,ArMul
                lea di,ArmuN
                SacarYMultiplicar:
                        SacarYMultiplicarMacro
                        ax,bx,si,di,Acumulador
                loop SacarYMultiplicar
                xor di,di
                xor cx,cx
                mov ax,00h
                mov bx,00h
                mov cx,contador
                lea di,ArMultiplicador
        RET


        DECIMAL:
                InvierteMultiplicador:
                        Invertir ax,si,di
                loop InvierteMultiplicador
                Limpia_Todo si,di,cx,ax,bx
                mov cx,contador
                lea si,ArMultiplicador
                lea di,ArNumero
                DivideySuma:
                        DivideySumaMacro
                        ax,bx,dx,si,di,Acumulador
                loop DivideySuma
                mov [di], byte ptr"$"
        RET

END Inicio
```

E*l documento de los MACROS queda de la siguiente manera:*

```
Datos MACRO
        Cad01 db 10,13, 'TREJO RODRIGUEZ JAVIER ENRIQUE ICO
O-6$'
        Cad02 db 10,13,10,13, 'Introduce un numero:  $'
        Cad03 db 10,13,10,13, 'El numero introducido fue:  $'
        Mult dw 1
        MultD dw 10
        Contador dw 0
        Acumulador dw 0
        ArNumero dw 100 dup(0)
        ArNum dw 100 dup(0)
        ArmuN dw 100 dup(0)
        ArMul dw 100 dup(0)
        ArMultiplicador dw 100 dup(0)
ENDM
Cargar MACRO
        mov dx,@DATA
        mov ds,dx
ENDM
Imprime MACRO Cad01, Cad02
        lea dx,Cad01
        mov ah,09h
        int 21h
        lea dx,Cad02
        mov ah,09h
        int 21h
ENDM
Limpia_SIDICX MACRO si,di,cx
        xor si,si
        xor di,di
        xor cx,cx
ENDM
Limpia_AB MACRO ax, bx
        mov bx, 00h
        mov ax, 00h
ENDM
Limpia_Todo MACRO si,di,cx,ax,bx
        xor si,si
        xor di,di
        xor cx,cx
        mov bx, 00h
        mov ax, 00h
ENDM
LeerMac MACRO ax,bx,cx,si
        sub al, 30h
        mov ah,00h
        mov [si],al
        mov ax,00h
        mov bx,00h
        add si,2
        inc cx
        mov contador,00h
        mov contador,cx
ENDM
Invertir MACRO ax,si,di
        sub si,2
        mov ax,[si]
        mov [di],ax
        add di,2
ENDM
LlenaArregloMultiplicadorMacro MACRO Mult,MultD,ax,bx,si
        mov bx,Mult
        mov [si],bx
        mov bx,00h
        mov ax, Mult
        mul MultD
        mov ax,ax
        mov Mult,ax
        add si,2
ENDM
SacarYMultiplicarMacro MACRO ax,bx,si,di,Acumulador
        mov bx,[si]
        mov ax,[di]
        mul bx
        mov ax,ax
        mov [di],ax
        add Acumulador,ax
        add si,2
        add di,2
        mov ax,00h
        mov bx,00h
ENDM
DivideySumaMacro MACRO ax,bx,dx,si,di,Acumulador
        mov ax,Acumulador
        mov bx,[si]
        div bx
        mov ax,ax
        mov bx,dx
        mov Acumulador,00h
        mov Acumulador,bx
        add ax,30h
        mov [di],ax
        add si,2
        inc di
        mov ax,00h
        mov bx,00h
        mov dx,00h
ENDM
ImprimeResultado MACRO ax,dx,Cad03,ArNumero
        mov dx, offset Cad03
        mov ah, 09h
        int 21h
        mov dx, offset ArNumero
        mov ah,09h
        int 21h
        mov ax, 4c00h
        int 21h
ENDM
```

## Resultados

Primero se ingresa el numero 2,498 como lo muestra la imagen (Img 9) y se va guardando en el arreglo.

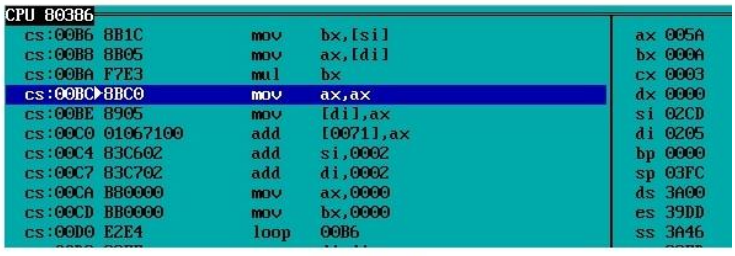

*(Im9. Se ingresa el número 2,498)*

Asumimos que ya guardó el número en el arreglo, el contador se quedó con 4 y todos los loops serán con 4 iteraciones.

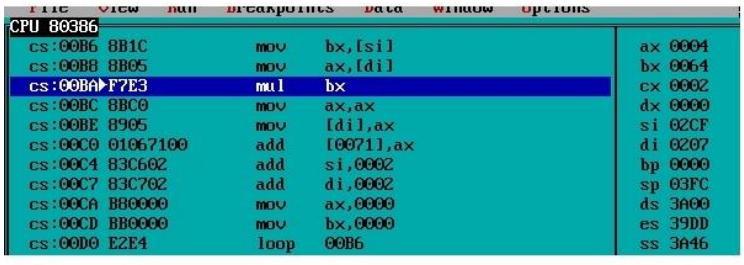
Para la primera parte se muestran en la siguiente imagen los resultados del cómo se va llenando el arreglo ArMul tal como lo muestra la imagen 10

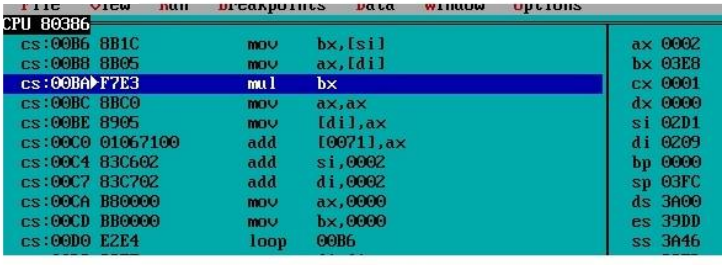

*(Im10. Llenado del arreglo ArMul)*

En la siguiente imagen (Img 11) se muestra cómo se va multiplicando el número en unidades por el multiplicador, es decir, 8*1, 9*10, 4*100 y 2*1,000 y la sumatoria se guarda en Acumulador.


*(Im11. Multiplicando las unidades por multiplicador)*

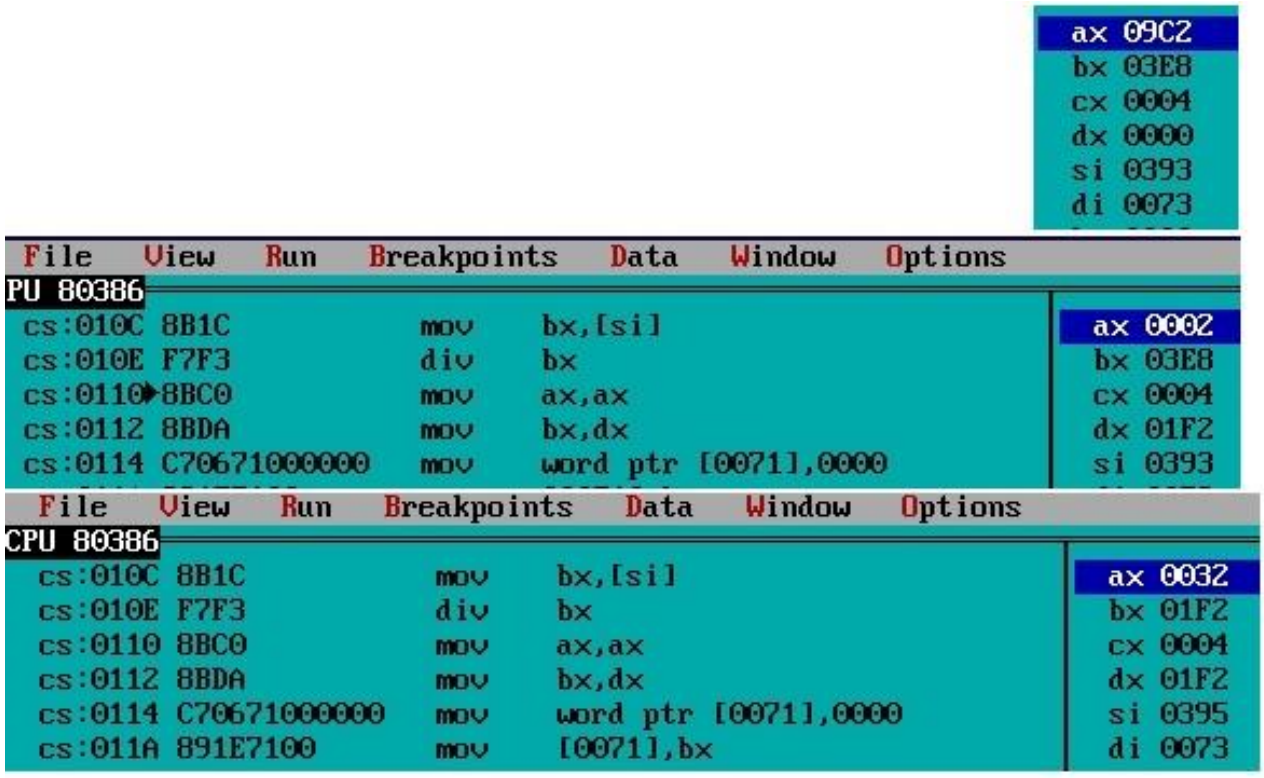


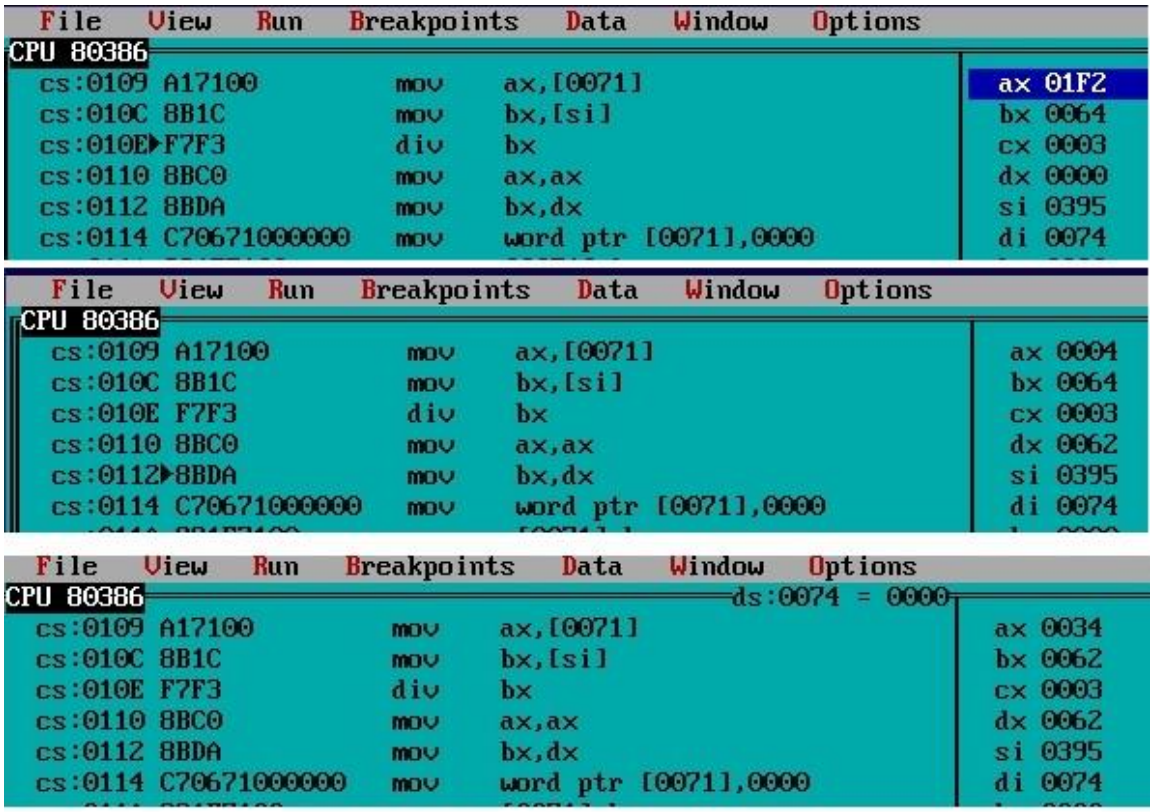
En las siguiente 4 imágenes, se muestra el procedimiento de la división. Tal como se explicó anteriormente, primero se carga el Acumulador en Ax para que tome el papel del dividendo, después el multiplicador en Bx pero ahora este tomará el papel de divisor tal como se muestra en la primera parte de cada imagen.

En la segunda parte de cada imagen se ve que el cociente queda en Ax, Bx se queda como estaba y el residuo de la operación se queda en Dx

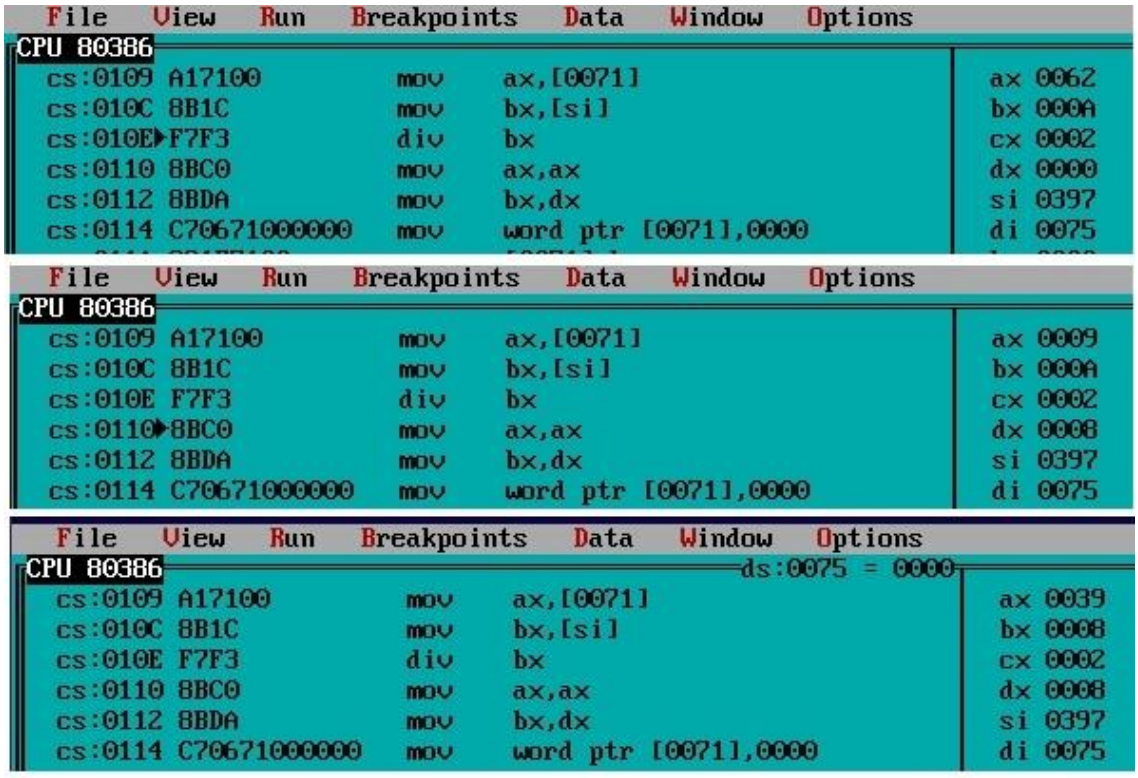
En la tercera parte de cada imagen se muestra que el residuo se pasa al registro Bx para posteriormente guardarlo en el arreglo ArNumero, que es el que va a imprimir los resultados.



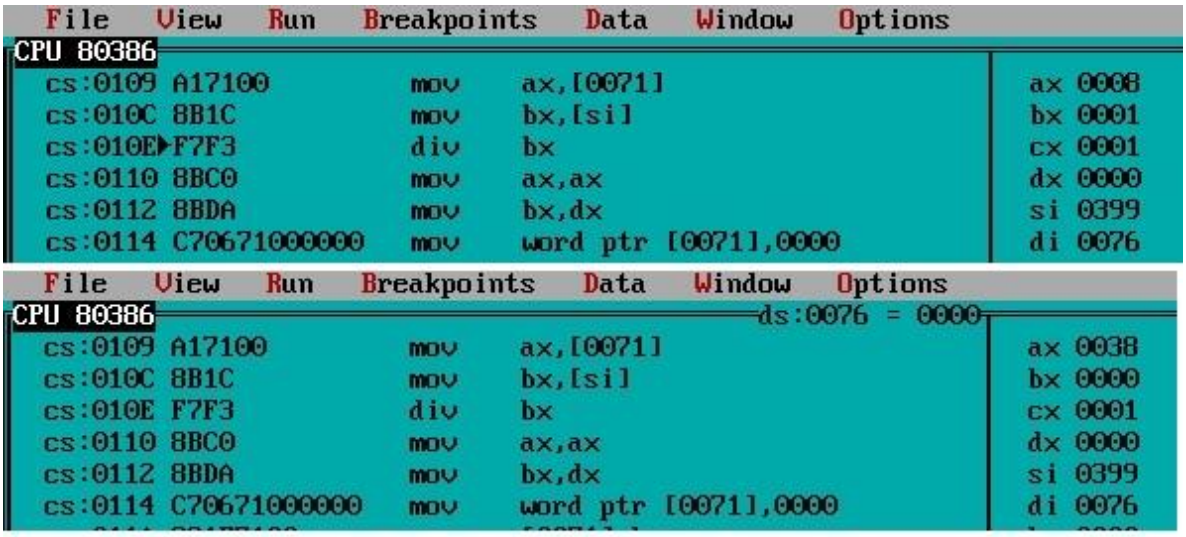

Iteración 1




Iteración 2




Iteración 3




Iteración 4

## Conclusión

De esta forma se concluye esta práctica de este programa en ensamblador, el cual consistió en recibir un numero y pasarlo a hexadecimal y de nuevo a decimal. Para hacerla el alumno tuvo que partirse la cabeza pensando en la lógica del cómo se tenían que mover los registros y qué operaciones hacer para que esta funcionara de la manera correcta y que hiciera lo que el profesor pidió.

No cabe duda de que el alumno aprendió demasiado sobre el uso de arreglos, registros, macros y procedimientos pues, todos estos elementos fueron usados para realizar esta práctica y es necesario un buen dominio sobre estos para seguir continuando el curso, dominio que creo que si tengo pues esta práctica me dejó mucho aprendizaje.

Se usaron imágenes de solo los puntos más importantes del programa, esto para que el reporte no quedara más largo y así evitar explicaciones redundantes o a veces hasta innecesarias.