Ejemplos de Codigos

Objetivo: Entender cómo se realizan las comparaciones de datos en lenguaje ensamblador

Programa 1.

Lee un carácter por teclado, se almacena en un registro de propósito general y se muestra desde un registro. Si se lee un numero entre 0-9 mostrara un carácter $, que es de la tabla ASCII

AH, 01h

……

Mov dl, al

```
name "leernumeroporteclado"
; lee dos numeros por teclado y los suma
.model small; se define el modelo de memoria
.stack 64; se define el tamaño de la pila (se define el espacio necesario para el stack)
.data
mensaje1 db "Ingresar numero ",'$'; Se define un tipo de datos bytes, se define una cadena de caracteres
mensaje2 db "Numero digitado para valor es ",'$';Se define un tipo de datos bytes, se define una
;cadena de caracteres


salto db " ",10,13, '$'; se define un salto para que los digitos no esten pegados. Funciona como

Valor1 db 0; se define variable tipo db
Valor2 db 0; se define variable tipo db
.code
inicio proc far
    mov ax, @data
    mov ds,ax ; ds=ax=msg, en el registro del segmento de datos guardamos lo que esta en ax

    ;limpiamos pantalla
    mov ax, 00 ; Establece modo de video espeficado. Funcion de la interrupcion 10h
    mov al, 03h; Modo de video 80 x 25
    int 10h
```

```
    ; mensaje en pantalla Ingresar numero
    mov ah, 09; Function (print string)servicio 9, imprime un string terminado en $
    lea dx, mensaje1; DX= String terminated by '$', carga la direccion efectiva (load efecty address )
    int 21h; Interruptions DOS Function

    ; Leer numero
    mov ah, 01h; funcion 01h de la interrupcion 21h. Entrada de caracter. Normalmente es el teclado
    ; A AH se envia la funcion 01h y en AL se almacena el carácter leido
    ;AL = Código ASCII del Carácter leído y Echo a pantalla. Ah = 01h y AL= codigo ASCII del
    ;caracter leido

    int 21h

    ; salto

    mov ah,09; Function (print string)servicio 9, imprime un string terminado en $
    lea dx,salto; carga la direccion efectiva (load efecty address )
    int 21h

    ;Imprimir mensaje 2.

    mov ah, 09
    lea dx, mensaje2
    int 21h


    ; Mostrar el caracter leido

    mov dl, al; el caracter leido esta en al, este se leyo con la funcion 01h, se mueve a dl para poder
    ;imprimir en pantalla
    mov ah,02h; funcion 02h de la interrupcion para la salida del caracter, DL contiene Código ASCII
    ;a enviar al dispositivo de salida
    int 21h

    ;regresa control a DOS

    mov ax, 4c00h
    int 21h;
    inicio endp
end inicio
```

Programa2.

Lee un número por teclado, el valor guardado en AL se almacena una variable declarada como DB (Valor1) y muestra el mismo número sin convertirlo a ASCII

mov ah, 01h

int 21h

mov valor1, al

Mov dl, valor1

Mov ah, 02h

```
name "leernumeroporteclado"
; lee numero por teclado
.model small; se define el modelo de memoria
.stack 64; se define el tamaño de la pila (se define el espacio necesario para el stack)
.data
mensaje1 db "Ingresar numero ",'$'; Se define un tipo de datos bytes, se define una cadena de caracteres
mensaje2 db "Numero digitado es ",'$';Se define un tipo de datos bytes, se define una
;cadena de caracteres


salto db " ",10,13, '$'; se define un salto para que los digitos no esten pegados. Funciona como

Valor1 db 0; se define variable tipo db
Valor2 db 0; se define variable tipo db
.code
inicio proc far
    mov ax, @data
    mov ds,ax ; ds=ax=msg, en el registro del segmento de datos guardamos lo que esta en ax

    ;limpiamos pantalla
    mov ax, 00 ; Establece modo de video espeficado. Funcion de la interrupcion 10h
    mov al, 03h; Modo de video 80 x 25
    int 10h


    ; mensaje en pantalla Ingresar numero
    mov ah, 09; Function (print string)servicio 9, imprime un string terminado en $
    lea dx, mensaje1; DX= String terminated by '$', carga la direccion efectiva (load efecty address )
    int 21h; Interruptions DOS Function

    ; Leer numero
    mov ah, 01h; funcion 01h de la interrupcion 21h. Entrada de caracter. Normalmente es el teclado
    ; A AH se envia la funcion 01h y en AL se almacena el carácter leido
    ;AL = Código ASCII del Carácter leído y Echo a pantalla. Ah = 01h y AL= codigo ASCII del
    ;caracter leido
    int 21h
    mov valor1, al; se mueve a la variable valor1 el numero digitado y almacenado en AL


    ; salto

    mov ah,09; Function (print string)servicio 9, imprime un string terminado en $
    lea dx,salto; carga la direccion efectiva (load efecty address )
    int 21h


    ;Imprimir mensaje 2.

    mov ah, 09
    lea dx, mensaje2
    int 21h

    ; Mostrar el caracter leido

    mov dl, valor1; el caracter leido esta en al, este se leyo con la funcion 01h, se mueve a dl para pode
    ;imprimir en pantalla
    mov ah,02h; funcion 02h de la interrupcion para la salida del caracter, DL contiene Código ASCII
    ;a enviar al dispositivo de salida
    int 21h

    ;regresa control a DOS

    mov ax, 4c00h
    int 21h;
    inicio endp
end inicio
```

Programa3.

Lee dos números digitados por teclado y los suma

```
; lee dos numeros por teclado y los suma. Solo se convierte al final la suma SUB 30h
.model small; se define el modelo de memoria
.stack 64; se define el tamaño de la pila (se define el espacio necesario para el stack)
.data
; la palabra .data es una directiva y comunica al ensamblador que a continuación se define un conjunto
; de datos  (variables) donde se almacenará la información
mensaje1 db "Ingresar primer numero ",'$'; Se define un tipo de datos bytes, se define una cadena de caract
mensaje2 db "Ingresar segundo numero ",'$';Se define un tipo de datos bytes, se define una cadena de caract
mensaje3 db "El resultado de la suma es... ",'$'; Se define un tipo de datos bytes, se define una cadena d

salto db " ",10,13, '$'; se define un salto para que los digitos no esten pegados. Funciona como
; un ENTER
Valor1 db 0
Valor2 db 0
.code
inicio proc far; inicia proceso, proc es una directiva que define procedimientos
                ; un segmento de codigo puede tener cualquier numero de procedimientos
                ; todos distinguidos por PROC y ENDP
                ; far es un operando, en este caso esta relacionado con la ejecucion del programa.
                ; En este caso el operador FAR informa al sistema que la direccion indicada
                ; es el punto de entrada para la ejecucion del programa
    ;Direccionamos el segmento de datos
    mov ax, @data
    mov ds,ax ; ds=ax=msg, en el registro del segmento de datos guardamos lo que esta en ax


    ;limpiamos pantalla
    mov ax, 00 ; Establece modo de video espeficado. Funcion de la interrupcion 10h
    mov al, 03h; Modo de video 80 x 25
    int 10h

    ;imprimir primer mensaje1 (Ingresar numero)
    mov ah, 09; Function (print string)servicio 9, imprime un string terminado en $
    lea dx, mensaje1; DX= String terminated by '$', carga la direccion efectiva (load efecty address )
    int 21h; Interruptions DOS Function

    ; Leer y almacena el primer digito en la variable valor1
    mov ah, 01h; funcion 01h de la interrupcion 21h. Entrada de caracter. Normalmente es el teclado
    ; A AH se envia la funcion 01h y en AL se almacena el carácter leido
    ;AL = Código ASCII del Carácter leído y Echo a pantalla. Ah = 01h y AL= codigo ASCII del
    ;caracter leido
    int 21h
    mov valor1, al; se mueve el digito almacenado en AL a la variable valor1

    ; imprimir salto, tipo Enter
    mov ah, 09
    lea dx, salto
    int 21h
```

```
    ;imprimir Segundo mensaje (Ingresar numero)
    mov ah, 09; Function (print string)servicio 9, imprime un string terminado en $
    lea dx, mensaje2; DX= String terminated by '$', carga la direccion efectiva (load efecty address )
    int 21h; Interruptions DOS Function

    ; Leer y almacena el segundo digito en la variable valor2
    mov ah, 01h;
    int 21h
    mov valor2, al; se almacena el segundo numero digitado en la variable valor2

    mov ah, 09
    lea dx, salto
    int 21h
    ; Realiza la suma
    mov bl, valor2; se carga el registro BL con el segundo numero
    add bl, valor1; se suma el segundo numero en BL. BL=BL+valor1
    ; imprimir mensaje 30h
    mov ah, 09
    lea dx, mensaje3
    int 21h

    ; realiza conversion para imprimir

    mov dl, bl; se mueve la suma al registro DL
    sub dl, 30h; se convierte el resultado restandole 30h. Se resta para obtener el numero
    mov ah, 02h; funcion 02h de la interrupcion para la salida del caracter, DL contiene Código ASCII
    ;a enviar al dispositivo de salida
    int 21h

    ;regresa control a DOS

    mov ax, 4c00h
    int 21h;
    inicio endp
end inicio
```

Programa 4

Lee dos números por teclado y los multiplica con la instrucción "mul"

```
name multiplica dos numeros
; lee dos numeros por teclado y los multiplica.
.model small; se define el modelo de memoria
.stack 64; se define el tamaño de la pila (se define el espacio necesario para el stack)
.data
; la palabra .data es una directiva y comunica al ensamblador que a continuación se define un conjunto
; de datos  (variables) donde se almacenará la información
mensaje1 db "Ingresar primer numero a multiplicar",'$'; Se define un tipo de datos bytes, se define una ca
mensaje2 db "Ingresar segundo numero ",'$';Se define un tipo de datos bytes, se define una cadena de carac
mensaje3 db "El resultado de la multiplicacion es... ",'$'; Se define un tipo de datos bytes, se define un

salto db " ",10,13, '$'; se define un salto para que los digitos no esten pegados. Funciona como
; un ENTER
Operando1 db 0
Operando2 db 0
.code
inicio proc far; inicia proceso, proc es una directiva que define procedimientos
                ; un segmento de codigo puede tener cualquier numero de procedimientos
                ; todos distinguidos por PROC y ENDP
                ; far es un operando, en este caso esta relacionado con la ejecucion del programa.
                ; En este caso el operador FAR informa al sistema que la direccion indicada
                ; es el punto de entrada para la ejecucion del programa
    ;Direccionamos el segmento de datos
    mov ax, @data
    mov ds,ax ; ds=ax=msg, en el registro del segmento de datos guardamos lo que esta en ax
```

```
;limpiamos pantalla
mov ax, 00 ; Establece modo de video espeficado. Funcion de la interrupcion 10h
mov al, 03h; Modo de video 80 x 25
int 10h

;imprimir primer mensaje1 (Ingresar numero)
mov ah, 09; Function (print string)servicio 9, imprime un string terminado en $
lea dx, mensaje1; DX= String terminated by '$', carga la direccion efectiva (load efecty address )
int 21h; Interruptions DOS Function

; Leer y almacena el primer digito en la variable operando1
mov ah, 01h; funcion 01h de la interrupcion 21h. Entrada de caracter. Normalmente es el teclado
; A AH se envia la funcion 01h y en AL se almacena el carácter leido
;AL = Código ASCII del Carácter leído y Echo a pantalla. Ah = 01h y AL= codigo ASCII del
;caracter leido
int 21h
sub al, 30h; restar 30h para obtener el numero
mov operando1, al; se mueve el digito almacenado en AL a la variable valor1

; imprimir salto, tipo Enter
mov ah, 09
lea dx, salto
int 21h
```

```
;imprimir Segundo mensaje (Ingresar numero)
mov ah, 09; Function (print string)servicio 9, imprime un string terminado en $
lea dx, mensaje2; DX= String terminated by '$', carga la direccion efectiva (load efecty address )
int 21h; Interruptions DOS Function

; Leer y almacena el segundo digito en la variable operando2
mov ah, 01h;
int 21h
sub al, 30h;restar 30h para obtener el numero
mov operando2, al; se almacena el segundo numero digitado en la variable valor2

; imprimir salto, tipo Enter
mov ah, 09
lea dx, salto
int 21h

;imprimir tercer mensaje (Ingresar numero)
mov ah, 09; Function (print string)servicio 9, imprime un string terminado en $
lea dx, mensaje3; DX= String terminated by '$', carga la direccion efectiva (load efecty address )
int 21h; Interruptions DOS Function
```

```
; realiza conversion para imprimir

mov al, operando1; se mueve el operando1 a AL, AL es utilizado para la multiplicacion
mov bl, operando2; se mueve el operando 2 a BL
mul bl; AL=AL*BL


mov dl, al; el resultado de la multiplicacion se almacena en DL
add dl,30h; se suma 30h para obtener el numero
mov ah,2h
int 21h


;regresa control a DOS

mov ax, 4c00h
int 21h;
inicio endp
end inicio
```

Programa 5.

Ciclo Loop (bucles). Realiza n iteraciones mientras el registro CX sea mayor que cero.

Su procedimiento es el siguiente:

mov CX, tamaño; se establece el tamaño del ciclo, es decir el número de iteraciones a realizar

etiqueta: ; la etiqueta que denota el inicio del cuerpo del bucle

……

Código ensamblador; código ensamblador, parte del cuerpo del ciclo a iterar

……

loop etiqueta ; Instrucción loop que retorna hacia la etiqueta que marca el inicio del cuerpo del ciclo mientras que CX > 0. En cada iteraccion CX decrementa en 1.

```
name bucles (lopp)
; repite un mensaje N veces
.model small; se define el modelo de memoria
.stack 64; se define el tamaño de la pila (se define el espacio necesario para el stack)
.data
; la palabra .data es una directiva y comunica al ensamblador que a continuación se define un conjunto
; de datos  (variables) donde se almacenará la información

mensaje1 db "Universidad Nacional Abierta y a Distancia",0dh,0ah,'$'; Se define un tipo de datos bytes,
;se define una cadena de caracteres. 0dh y 0ah indican un salto
.code
inicio proc far; inicia proceso, proc es una directiva que define procedimientos
                ; un segmento de codigo puede tener cualquier numero de procedimientos
                ; todos distinguidos por PROC y ENDP
                ; far es un operando, en este caso esta relacionado con la ejecucion del programa.
                ; En este caso el operador FAR informa al sistema que la direccion indicada
                ; es el punto de entrada para la ejecucion del programa
    ;Direccionamos el segmento de datos
    mov ax, @data
    mov ds,ax ; ds=ax=msg, en el registro del segmento de datos guardamos lo que esta en ax

    ;limpiamos pantalla
    mov ax, 00 ; Establece modo de video espeficado. Funcion de la interrupcion 10h
    mov al, 03h; Modo de video 80 x 25
    int 10h
```

```
    mov cx, 000ah; equivalente a 10 en decimal, se puede escribir directamente el 10 o
    ;en su defecto el equivalente en hexadecimal que es 000ah
    imprimirmensaje: ; colocamos una etiqueta para el codigo. Toda etiqueta finaliza en dos puntos ":"
        mov ah,09h
        lea dx,mensaje1; imprime el mensaje en cada ciclo, se repite 10 veces
        int 21h
    loop imprimirmensaje; en cada ciclo se decrementa el registro CX inicializado en 10.
    ; cuando CX sea cero, sale del bucles


    mov ax, 4c00h
    int 21h;
    inicio endp
end inicio
```

Programa 6. Comparar si un número es cero o uno

Digitado un número por teclado determina si es cero o uno. Se debe digitar solo cero o uno.

Utiliza cmp, JE y JZ

```
name comparanumerocerooUno
; lee un numero por teclado y lo compara con cero o uno
.model small; se define el modelo de memoria
.stack 64; se define el tamaño de la pila (se define el espacio necesario para el stack)
.data
; la palabra .data es una directiva y comunica al ensamblador que a continuación se define un conjunto
; de datos  (variables) donde se almacenará la información
mensaje1 db "Ingresar numero ",'$'; Se define un tipo de datos bytes, se define una cadena de caracteres
mensaje2 db "El numero es CERO ",0dh,0ah,'$';Se define un tipo de datos bytes, se define una cadena de car
mensaje3 db "El numero es UNO ",0dh,0ah,'$'; Se define un tipo de datos bytes, se define una cadena de car

salto db " ",10,13, '$'; se define un salto para que los digitos no esten pegados. Funciona como
; un ENTER
numero db 0

.code
inicio proc far; inicia proceso, proc es una directiva que define procedimientos
                ; un segmento de codigo puede tener cualquier numero de procedimientos
                ; todos distinguidos por PROC y ENDP
                ; far es un operando, en este caso esta relacionado con la ejecucion del programa.
                ; En este caso el operador FAR informa al sistema que la direccion indicada
                ; es el punto de entrada para la ejecucion del programa
    ;Direccionamos el segmento de datos
    mov ax, @data
    mov ds,ax ; ds=ax=msg, en el registro del segmento de datos guardamos lo que esta en ax
```

```
    ;limpiamos pantalla
    mov ax, 00 ; Establece modo de video espeficado. Funcion de la interrupcion 10h
    mov al, 03h; Modo de video 80 x 25
    int 10h

    ;imprimir primer mensaje1 (Ingresar numero)
    mov ah, 09; Function (print string)servicio 9, imprime un string terminado en $
    lea dx, mensaje1; DX= String terminated by '$', carga la direccion efectiva (load efecty address )
    int 21h; Interruptions DOS Function

    ; Leer y almacena el digito en la variable numero
    mov ah, 01h; funcion 01h de la interrupcion 21h. Entrada de caracter. Normalmente es el teclado
    ; A AH se envia la funcion 01h y en AL se almacena el carácter leido
    ;AL = Código ASCII del Carácter leído y Echo a pantalla. Ah = 01h y AL= codigo ASCII del
    ;caracter leido
    int 21h

    sub al,30h; restar 30h para obtener el numero
    mov numero, al; se mueve el digito almacenado en AL a la variable valor1

    ;comparamos con cero
    cmp al,0
    jz cero
    ;comparamos con uno
    cmp al,1
    je uno


    cero:
    ; imprimir salto, tipo Enter
    mov ah, 09
    lea dx, salto
    int 21h

    mov ah, 09
    lea dx, mensaje2; imprime que el numero digitado es cero
    int 21h
    jmp fin


    uno:
    ; imprimir salto, tipo Enter
    mov ah, 09
    lea dx, salto
    int 21h

    mov ah, 09
    lea dx, mensaje3; imprime que el numero digitado es uno
    int 21h

    fin:
    ;regresa control a DOS

    mov ax, 4c00h
    int 21h;
    inicio endp
end inicio
```

Programa 7.

Se deben capturar dos números por teclado num1 y num2; num1 es el numero que se va a multiplicar en cada iteraccion y num2 es la cantidad de veces que se va a multiplicar

Recordemos que CX es clave porque es el contador que decrementa cada vez en el ciclo

```
mov cx, num2
mov ax, num1
inicio:
 mov bx,num1
 mul bx ;ax = ax * bx
 loop inicio ;c--
```

Programa 8.

Digitar un numero por teclado y determinar si es par, impar o cero. Utiliza CALL y etiquetas. Verifica bit de paridad

```
;*************************************************
;Autor: XXXXXXXX
;Fecha: XXXXXXXX
;Este codigo fuente determina
;si un digito decimal ingresado en teclado es
;par impar, o cero. Se basa en la lectura del bit0
;y en la comparacion con 0 de el registro AL donde esta
;el digito
;
.model small
.stack
.data
 cadena db 10,13,'Ingrese numero:$'
 es_impar  db 10,13, 'El numero ingresado es IMPAR$'
 es_par    db 10,13, 'El numero ingresado es PAR$'
 es_cero   db 10,13, 'El numero ingresado es CERO$'


.code

  mov dx,@data       ;inicializa posicion del segmento
  mov ds,dx          ;de datos, carga en dx la direccion de origen de los caracteres
                     ;para que se puedan direccionar mas adelante
  Inicio:

lea dx,cadena        ;bucle principal que
mov ah,09h           ;solicita numero, compara
int 21h              ;e imprime resultado

call lee_nums
call Compara

lee_nums:    ; 01 es el numero de una funcion(leer un caracter en formato ASCII del teclado
                     ; para poder leer ese caracter escrito en teclado se debe especificar
                     ; en el registro AH (mov AH, valor)
                     ; ese numero de funcion y luego llamar la interrupcion con la
                     ;instruccion INT seguida del numero de interrupcion(21h)
                     ; para que el contador de programa salte a esa area reservada de memoria
                     ; a leer las instrucciones que conforman esa funcion y se lea el valor
                     ; ASCII del teclado, y se guarde en el registro AL
                     ;esta(s) instruccion (es) tiene(n) el mismo significado en todo este progr
                     ;en la parte en que se repita

            ;Si se revisa el codigo ASCII que tiene 0-255 caracteres, se vera que el caracter "0
            ;ocupa la posicion 48, osea, es el carater numero 48. 48 en hexadecimal se escribe:
            ;30h. Como las operaciones aritmeticas se hacen con numerosn en formato binario,
            ;los caracteres que se ingresen por teclado al ser codificados en ASCII
            ;no representan el valor en binario de numero al que representan;
            ;ejemplo: el numero 0 digitado en el teclado, llegara
            ;al registro del microprocesador no como 00h, sino como 30h, osea en binario:0011000
            ; que no es la cantidad 0, porque cero es 00000000
            ;pero si se le resta a ese codigo ASCII el valor de 30h en hexa,(48 en decimal)
            ;o 00110000 en binario
            ; el valor da : 30h-30h=0  o en binario 00110000-00110000= 00000000
            ;que si es el valor binario de la cantidad "0"
            ;Si ahora por ejemplo, se introduce el numero 5, como se hizo por teclado, el teclad
            ;lo manda codificado en ascii y el resultado que entra al registro del microprocesad
            ;es 35h, que es 53 en decimal, obviamente en binario el valor no representa a la
            ;cantidad 5, y cualquier operacion tiene un resultado diferente al real.
            ;entonces, si el '5' en ASCII es igual a 35h o 01110101 en binario
            ;al restarle otra vez, 30h en hexa (48 en decimal) resulta el numero binario
            ;que si equivale a esa cantidad, veamos:
            ;35h-30h=5h
            ;01110101-00110000 =00000101
            ; Es por eso que se resta 30h o 48 en decimal a la entrada del teclado
            ; restarle esos 30h al valor de entrada es practicamente "convertir de ASCII a binar
```

```
                ; en programas que reciben mas de un digito se convierte cada digito
                ; e internamente se procesan los digitos para construir el valor binario
                ;
mov ah,01       ;lee el digito como ASCII
int 21h         ;le asigna el valor numerico ASCII->BIN en AL
sub al,30h      ;para poder operar aritmeticamente
ret             ;0 (hex)=00h  0 (ASCII)= 48/30h...9 (ASCII)=57/39h


Compara:
cmp AL,0h       ;compara si todo AL=0
je  cero
and AL,01h      ;lee solo el bit0, es una simple tecnica de mascara: https://es.wikipedia.org/wiki/0
cmp AL, 01h     ;comprueba el  bit0 para 0(par) o 1(impar)
jz  impar

jmp par         ; si el bit no es 1, y AL no vale 0
                ;es par


                impar:
                lea dx,es_impar
                          ; 09 es el numero de una funcion(escribir una cadena de caracteres/string
                          ; en pantalla) contenida dentro del vector de interrupcion 21h
                          ; para poder escribir la cadena se debe especificar en el registro AH
                          ;(mov AH, valor)
                          ; ese numero de funcion y luego llamar la interrupcion con la
                          ;instruccion INT seguida del numero de interrupcion(21h)
                          ; para que el contador de programa salte a esa area reservada de memoria
                          ; a leer las instrucciones que conforman esa funcion y se escriban a panta
                          ; los caracteres desde la direccion cargada en el registro DX
                          ; (lo que se hizo mediante la instruccion LEA que es CARGAR DIRECCION EFEC
                          ; cargamos la direccion exacta de los caracteres del mensaje "es par"
                          ;esta(s) instruccion(es) tiene(n) el mismo significado en todo este progra
                                                ;en la parte en que se repita

                          mov ah,09 ;escribir en pantalla
                          int 21h   ;que es impar
                          jmp Inicio
                          par:
                          lea dx,es_par
                          mov ah,09 ;escribir en pantalla
                          int 21h   ;que es par
                          jmp Inicio
                          cero:
                          lea dx,es_cero
                          mov ah,09 ;escribir en pantalla
                          int 21h   ;que es cero
                          jmp Inicio


                          end
```

Programa 9.

Digitar un numero por teclado y determinar si es par, impar o cero teniendo en cuenta el modulo

```
.model small; Definimos el modelo de memoria
.stack 64; Definimos el tama?o de la pila
.data
;INFORMACION
ESPACIO db 13,10,'$';salto de linea
LECTURA db 13,10,'PROGRAMA QUE CONSISTE EN LEER UN NUMERO DE UN DIGITO POR TECLADO','$'
LECTURA1 db 13,10,'Y EVALUAR SI EL NUMERO INGRESADO ES: PAR, IMPAR O CERO','$'
LECTURA2 db 13,10,'TRABAJO REALIZADO POR JORGE ARMANDO SANTOS TORRES','$'
LECTURA3 db 13,10,'CODIGO: xxxxxxx','$'
LECTURA4 db 13,10,'UNAD' ,'$'
LECTURA5 db 13,10,'MES ADO' ,'$'
MENSAJE db 13,10,'DIGITAR NUMERO DE UN SOLO DIGITO: ','$'
MEN_PAR db 13,10,'-------------------EL NUMERO INGRESADO ES PAR ----------------------','$'
MEN_IMPAR db 13,10,'---------------- EL NUMERO INGRESADO ES IMPAR --------------------','$'
MEN_CERO db 13,10,'------------------ EL NUMERO INGRESADO ES CERO ---------------------','$'
ESPACIO1 db 13,10,'$'  ;salto de linea
VALOR db 0              ;Variable para el numero digitado
NUM db 2              ;Variable de division por 2
.code
.startup              ;Inicio de la codificacion del programa
mov ax, @data         ;Segmento de datosmov ds,ax
mov ah,09h
lea dx, ESPACIO         ; salto de linea
int 21h
mov ah,09h
lea dx, LECTURA         ;Mensaje
int 21h
mov ah,09h
lea dx, LECTURA1        ;Mensaje
int 21h
mov ah,09h
lea dx, LECTURA2        ;Mensaje
int 21h
mov ah,09h
lea dx, LECTURA3        ;Mensaje
int 21h
mov ah,09h
lea dx, LECTURA4         :Mensaje

int 21h
mov ah,09h
lea dx, LECTURA5        ;Mensaje
int 21h
mov ah,09h
lea dx, ESPACIO        ;Fin... Mensaje con informacion del programa y un salto de linea
int 21h
mov ah,09h
lea dx, MENSAJE         ;Se muestra el mensaje de ingresar un numero de un digito
int 21h
call leer             ;se Lee el numero digito
sub al,30h            ;Se captura el numero en al
mov VALOR, al           ;Se pasa el numero de al a la variable VALOR
mov ah,09h            ;Salto de linea en el programa
lea dx, ESPACIO
int 21h
mov bl,VALOR            ;Se pasa el valor de la variable VALOR a bl
cmp bl,0              ;Se compara el valor de bl con 0
je cero               ;Si la comparacion es verdadera, salto al procedimiento Cero con la etiqueta .
int 21h
xor ax,ax             ;Se limpia el registro AX
mov al,NUM            ;se mueve el valor de la variable NUM  a al
mov bl,al             ;se mueve  al a bl
mov al,VALOR            ;se coloca el valor de la variable VALOR, el numero capturado a al
div bl                ;se realiza la division
cmp ah,0              ; residuo se almacena en ah y se realiza la comparacion
jz par                ;Salto al proedimiento par
jmp impar             ;salto al procedimiento impar
;procedimientos
leer proc near
mov ah,01h
int 21h
ret
CERO:                 ;Salto en caso que sea Cero
mov ah,09h
lea dx,MEN_CERO
int 21h
```

```
jmp salir
PAR:                    ;Salto si es par
mov ah,09h
lea dx,MEN_PAR
int 21h
jmp salir
IMPAR:                  ;Salto si es impar
mov ah,09h
lea dx,MEN_IMPAR
int 21h
jmp salir
SALIR:
mov ax, 4c00h  ; Funcion para terminar proceso
int 21h        ; Llamar a NUM
end
```

Programa 10.

Multiplicación de dos números a través de sumas sucesivas

```
.model small
.stack 100h

.data

a db ?
msg1 db 'Ingrese el primer numero: $'
msg2 db 'Ingrese el segundo numero: $'
msg3 db 'El resultado de la multiplicacion es: $'
salto db 10,13," ",'$'
Valor1 db 0
Valor2 db 0
multiplicacion db 0

.code
.startup

inicio:
        mov ax,@data
        mov ds,ax

        ;limpiar pantalla
        mov ah,00
        mov al,03h
        int 10h

        mov ah,9;  mostrar mensaje
        lea dx,msg1 ;muestra el mensaje para el ingreso del primer numero
        int 21h

        ;lee el pimer digito y lo convierte a decimal
        mov ah,01h
        int 21h ;usa la interrupcion 21
        sub al, 30h ;resta 30 al registro al, obtener numero decimal
        mov Valor1, al  ;guarda el contenido de al en Valor1

        ;salto de linea para el siguiente mensaje
        mov ah, 09
```

```
        lea dx, salto
        int 21h


         mov ah,9;  mostrar mensaje para que ingrese el segundo numero
         lea dx,msg2
        int 21h

        ;lee el segundo digito y lo convierte a decimal
        mov ah,01h
        int 21h
        sub al, 30h
        mov Valor2, al ;guarda el contenido de al en Valor2
        ;salto de linea
        mov ah, 09
        lea dx, salto
        int 21h

        mov cx, 00h ; carga el registro cx con 0 hexa
        mov al, 00h ;carga el registro al con 0 hexa

;etiqueta de nombre mull
mull:   inc cx  ;incrementa el valor del registro cx
        add al, Valor1  ;suma el Valor 1 al registro al y lo almacena en al
        cmp cl, Valor2  ;compara el registro cl con el dato guardado en Valor2
        JNZ mull    ;si la comparacion no es igual a cero va a la etiqueta mull
                    ;si es cero salta a la siguiente instruccion
        mov multiplicacion, al  ;guarda el contenido de al en la variable multiplicacion

        mov ah, 09h         ;carga el registor ah con 9 hexa
        lea dx, msg3        ;muestra el mensaje 3
        int 21h
        mov dl, multiplicacion  ; carga el contenido de multiplicacion en rel registor dl
        add dl, 30h             ;le suma 30 al registro dl para sacar el valor en decimal
        mov ah, 02              ;carga el registro ah con el valor de 02
        int 21h
```

```
        ;imprimir segundo mensaje

        fin:
        mov ah, 4ch
        int 21h
end     inicio
```

Programa 11.

Programa que realizar la multiplicación a traves de sumas sucesivas y la potencia a traves de multiplicaciones sucesivas

```
include 'emu8086.inc' ; archivo con funciones/librerias varias hecha para el microproc
.model small  ;distribucion de memoria  data+code en mismo segmento
.stack     ;asigna tamano a pila  1K
.data

msj1 db 10,13,10,13,"Ingrese el numero 1:", '$ '    ;cadenas de caracteres
msj2 db 10,13,"Ingrese el numero 2:", '$ '
msj3 db 10,13,"Resultado de multiplicacion:", '$ '
msj4 db 10,13,'Resultado de potenciacion:$ '


var1      db 0             ;primer num
var2      db 0             ;segundo num
potencia  db 1             ;la potencia vale 1 inicialmente en caso de elevar a 0, el resultado d
;base      db 0
;exponente db 0
diez      dw 10            ; se usa como /divisor
                           ; en  la funcion  PRINT_NUM_UNS.


.code                   ;programa

inicio:
    mov dx,@data       ;inicializa la direccion del segmento datos
    mov ds,dx          ;pone en el puntero del segmento datos DS la direccion de memoria
    xor ax,ax          ;donde se han guardan los valores de los operandos ingresados por teclado
    xor bx,bx          ;y los grupos de caracteres para mostrar en pantalla(cada mensaje msj1..msj
    xor cx,cx          ;hacer XOR con el mismo registro tiene el efecto de poner el registro
    mov var1,0         ;en ceros  porque por algebra de Boole a XOR a=0 ;ejemplo: 0010 XOR 0010=0
    mov var2,0         ;se hizo luego de ensayos, para que no aparecieran resultados de
                       ;operaciones anteriores, que estaban apareciendo
                       ;ya que el programa se hizo para que vuelva a pedir los numeros desde tecla
                       ; y debe borrar los resultados anteriores y comenzar con los nuevos datos
                       ; es una tecnica llamada inicializar (en esto caso iniciar en 0)
    mov ah, 09h     ; despliega usando la funcion 9(valor cargado en AH) de la interrupcion 21h
    lea dx, msj1    ; el mensaje1 'Ingrese numero1'enviando los caracteres uno por uno a pantalla
    int 21h         ; hasta que detecte el signo $ y terminar de enviar


     mov ah, 01h    ;se carga en AH el valor 1 (numero de la funcion)de la interrupcion 21h del mi
     int 21h        ;que obtiene/lee el numero de 1 digito (un caracter ASCIIdel teclado)en AL
     sub al, 30h    ;pasa el caracter ASCII a numero binario compensando el offset que hay entre
     mov var1, al   ;el codigo ASCII de los digitos decimales(0..9) y el valor binario
                    ;que es de 48 posiciones (30h en hexa)
                    ;var1=numero 1

    mov ah, 09h     ; despliegua mensaje 2 usando la funcion 9 de la interrupcion 21h
    lea dx, msj2    ; 'Ingrese numero 2'escribiendo todos los caracteres hasta que encuentre '$'
    int 21h

    mov ah, 01h     ;de nuevo obtiene el numero de 1 digito con la funcion 1 de interrupcion21h
    int 21h         ;en el registro AL y lo guarda en la ubicaciin var2
    sub al, 30h     ;ASCII -> BIN
    mov var2, al    ;var2=numero 2   y lo carga en el registro var2 para mantenerlo de referencia
                    ;en este punto ya tenemos los dos numeros y vamos a operar

                    ;Si el num2=0,se evita multiplicar y se va al paso de escribir resultado
                    ; el resultado de multiplicar esta en BX que inicialmente es 0
                    ; si no se multiplico (porque num2=0) entonces BX todavÝa vale 0
    cmp  var2,0     ; y este es el resultado que sale a pantalla num1*0=0.
    jz   sr         ;Si el numero 2 no vale 0, s multiplica normalmente
    mov cl,var2     ;contamos las veces que se debe sumar el numero1, usando el registro CL
                    ;como un contador y la intruccion LOOP que en conjunto hacen a CL
                    ;un contador decreciente. cuando se combinan (una funcionalidad del micro)
                    ;el numero1 se suma n veces=numero2. Cada vez que se suma, el valor decrementa
                    ;y cuando es 0, se sale del bucle que repite la suma, y va a la instruccion
                    ;de mostrar el resultado

    Multi:          ;la operacion de multiplicacion suma sucesivamente  el numero1
    add bl,var1     ;hay que sumar el numero 1 a si mismo, la cantidad de veces del numero2
    loop Multi      ;cuando decremente CL=0 salir a imprimir resultado

sr:
    mov ah, 09h     ;escribe en pantalla con la funcion 9 de la interrupcion 21h
```

```
    lea dx, msj3    ;'resultado de multiplicacion:'
    int 21h         ;LEA carga la direccion efectiva de donde esta el mensaje 3 en DX
                    ;y la funcion 9 de la  interrupcion 21h toma ese argumento en DX
                    ;para poder ir a buscar los caracteres que se deben imprimir en pantalla
                    ;eso es lo que hace el bloque mov ah,9 -lea dx,msj-int21h que se usa varias
                    ;veces en este programa ensamblador para escribir mensajes al usuario


     mov ax,bx      ;carga  en AX el resultado de la multiplicacion que esta en BX
     call PRINT_NUM ;llama a un procedimiento/rutina que convierte el valor del resultado
                    ;en los digitos decimales hasta 4 cifras
                    ;esta funcion que imprime el valor en pantalla del numero en ese registro
                    ;viene en las librerias de funciones predisenadas del 8086, archivo emu8086.i
                    ;la explicacion de funcionamiento esta el final del programa

    mov ah, 09h     ;despliega con la funcnion 9 de interrupcion 21h el mensaje
    lea dx, msj4    ;'resultado de la potenciacion:'
    int 21h

    xor ax,ax       ;Se dejan en 0 los registros para volver a hacer operaciones (ya se explico la
    xor bx,bx       ;en ellos, la potenciacion por multiplicaciones sucesivas
    xor cx,cx       ; DX se dejo igual, para no afectar la direccion de donde busca los mensajes

     mov al, potencia
    cmp var2,0
    jz  sh

    mov cl,var2



                           ;desde este punto se realiza la potenciacion  elevando el numero1
     Pot:                  ;a el valor del numero2. Se hace a traves de multiplicaciones suces
     mul var1                ;   si se eleva a 0 osea, var2=0, que vaya a fin e imprima el

                             ;   valor del registro potencia que es 1 por defecto
     loop Pot                ;   y al final del programa que vuelva a dejar en 1 ese registro
```

```
sh:

     call PRINT_NUM

    jmp inicio          ; vuelve al inicio del programa a solicitar los dos digitos
                        ; el programa se repite como un bucle




; Este procedimiento escribe en pantalla el numero en AX,
; se emplea con PRINT_NUM_UNS para escribir numeros con signo:
PRINT_NUM       PROC    NEAR    ; las subrutinas que emplea estan en posiciones cercanas NEAR
        PUSH    DX              ; el ensamblador dispone de saltos cortos dentro del procedimient
        PUSH    AX              ;se respaldan los datos en DX y AX en pila (un segmento de memori

        CMP     AX, 0           ; se compara si AX=0
        JNZ     not_zero        ; si no es cero, salta a not_zero
                                ; si es 0 imprime el numero 0 en pantalla
        PUTC    '0'
        JMP     printed         ;salta a printed que es el final, y restablece los valores AX y D

not_zero:
        ; verifica el  SIGNO de AX,
        ; si es negativo, halla el valor absoluto:
        CMP     AX, 0
        JNS     positive  ;si el valor es positvo, no se escribe el signo '-'
        NEG     AX

        PUTC    '-'     ;si dio negativo, se escribe en pantalla - y el valor absoluto

positive:
        CALL    PRINT_NUM_UNS    ;si es positivo va a rutina de imprimir sin signo
printed:
```

```
        POP     AX              ; aqui se sale de esta rutina, y se restablecen
        POP     DX              ;los antiguos valores de Ax,DX
        RET
PRINT_NUM       ENDP


; este procedimineto depsliega un numero, escribe primero el digito mas alto,hasta el menor (4dig
; sin signo cargado en  AX (mas de un digito)
; admite valores de  0 a 65535 (FFFFh)  numeros de 16 bit, dado que AX es de 16 bit
PRINT_NUM_UNS   PROC    NEAR ;de igual manera el procedimiento no llama otros segmentos de memori
        PUSH    AX           ;se respaldan los valores actuales de los registros
        PUSH    BX           ;para poder cargar los valores de los numeros que esta en BIN , y pr
        PUSH    CX           ;para imprimir en pantalla como ASCII 0..9
        PUSH    DX

        ; se usa como bandera para evitar escribir ceros a la izquierda: Ej. 8 en lugar de 008
        MOV     CX, 1  ;mientras sea 1 , no se escribira esa posicion en pantall

        ; (dividir por "/ 10000" arroja un numero menor o igual a 9).
        MOV     BX, 10000       ; 2710h - valor hexa del divisor.
                                ;ya que 65535 tiene 5 cifras, se divide por el numero 10mil
                                ;para obtener la cifra alta, por 1000 para la suguiente
        ; AX =0?                ;100 la cifra de centenas, 10 para la cifra decenas
                                ;cada vez que se divide se va guardando el residuo
                                ;para seguir dividiendo y obtener el valor de cada cifra
                                ;decenas de mil, unidades de mil, centenas, decenas y unidad
                                ;la tecnica es escribir primero las cifras altas del numero, ya c
                                ;las primeras que se obtienen
                                ;hay una rutina que indica si un digito es cero, para evitar
                                ;escribir ceros a la izquierda
        CMP     AX, 0
        JZ      print_zero  ;si AX vale 0, evitar convertir y simplemente va a escribir 0

begin_print:

        ; verifica el estado del divisor (si es 0 va al fin de rutina/ end_print):
        CMP     BX,0

        JZ      end_print

        ;evita escribir ceros a la izquierda:
        CMP     CX, 0
        JE      calc
        ; Si  AX<BX El resultado de DIVision sera cero:
        CMP     AX, BX
        JB      skip    ;Jump if Below: si AX es menor a BX (por debajo) va a skip
calc:
        MOV     CX, 0   ; fija la bandera.

        MOV     DX, 0   ;DX vale 0, y el valor a dividir es lo contenido en AX, DX tendra el resi
        DIV     BX      ; AX = DX:AX / BX   (DX=residuo). ;aqui hace la division
                        ; por ejemplo, si el numero es 9 al dividir por 10000,
                        ; dara 0, y se comienza a calcular que hay digitos superiores que valen 0
                        ; (unidades de mil)  resultando 00009

        ; escribir en pantalla el ultimo digito
        ; AH=0 simpre porque siempre resulta 0..9 ocupando AL y por eso se ignora AH
        ADD     AL, 30h     ; convertir a codigo ASCII (ya se ha explicado ASCII a BIN)
        PUTC    AL          ;ahora se suma el offset, en lugar de restar, para hacer lo inverso
                            ;se obtiene el digito n en AL y se paso a ASCII, para luego mandar a
                            ;pantalla

        MOV     AX, DX  ; Obtener el residuo de la ultima DIVision para continuar.
                        ;y a 1 para poder crear las cifras del numero que se pone en pantalla
                        ; si tenemos 1206, al dividir por 1000 da '1' (primer cifra/unid de mil)
                        ;y sobra 206, ya terminamos las unids de mil
                        ;ahora lo que queda se divide por 100 y daria '2', sobrando 6
                        ;luego ese 6 se divide por 10
skip:
        ; divide el divisor BX=BX/10  ;divide el valor de BX por 10,porque va pasando de
        ; 1000 a 100 a 10

        PUSH    AX          ;guarda el residuo para volverlo a dividir mas adelante y obtener las
        MOV     DX, 0       ;cifras
        MOV     AX, BX
```

```
        DIV     diez            ; AX = DX:AX / 10   (DX=residuo).
        MOV     BX, AX
        POP     AX


        JMP     begin_print

print_zero:
        PUTC    '0'             ;envia el caracter ASCII 0 cuando AX=0

end_print:
                                  ;restablece los valores que guardo en la pila
        POP     DX                ;y retorna de la rutina
        POP     CX                ;debe hacerlo en orden contrario con la intruccion POP
        POP     BX                ;de desapilar, ya que apilar y desapilar se realizan en sentido
        POP     AX                ;contrario con deplazamientos en direcciones de la memoria
                                  ; van en sentido ascendente y descente respectivamente
        RET
PRINT_NUM_UNS   ENDP


    ;la siguiente macro de la libreŕa escribe en
;pantalla el equivalente en ASCII del valor que
;se cargue en el registro char

;*************************************
;Explicacion de la macro PUTC usada por el procedimiento de PRINT_N
;*********************************************************************
;PUTC    MACRO   char
;        PUSH    AX          ;guarda el actual valor de AX para poder usarlo
;        MOV     AL, char    ;carga en AL el caracter ASCII
;        MOV     AH, 0Eh     ;la funcion 0E de la interrupccion 10h
;        INT     10h         ;escribe el caracter ASCII cargado en AL
;        POP     AX          ; y corre un espacio [posiciona para otro caracter]
                            ;finalmente, con POP se devuelve el valor que tenŕa AX
                            ;antes de usarlo en esta macro

;ENDM                        ;directiva que indica el final de la macro




            end
```