

# INSTITUTO TECNOLÓGICO DE CIUDAD GUZMAN

**Ingeniería en Sistemas Computacionales**

**Departamento de Sistemas e Informática**

**Materia: Lenguajes de Interfaz**

**Grupo: A**

**Unidad 3**

**Trabajo: “Biblioteca de Macroinstrucciones”**

**Fecha: 31 de mayo del 2021**

**Profesor: Estanislao Castillo Horta**

**Integrantes del Equipo:**

**Juan Fernando Brambila Rivera**

**José Eduardo Peña Jiménez**

**Aram Missael Guzmán Boiso**

**Luis Ángel Guzmán García**

## INDICE

|   |    |
|---|----|
| 1. MACROINSTRUCCIÓN: <b>Conversión a Decimal</b> .....        | 2  |
| 2. MACROINSTRUCCIÓN: <b>Conversión a Hexadecimal</b> .....    | 4  |
| 3. MACROINSTRUCCIÓN: <b>Conversión a Binario</b> .....        | 6  |
| 4. MACROINSTRUCCIÓN: <b>Leer una Cadena</b> .....             | 8  |
| 5. MACROINSTRUCCIÓN: <b>Compactar una Cadena</b> .....        | 9  |
| 6. MACROINSTRUCCIÓN: <b>Calculo de la raíz Cuadrada</b> ..... | 11 |
| 7. MACROINSTRUCCIÓN: <b>Validación de Números</b> .....       | 13 |

## 1. MACROINSTRUCCIÓN: **Conversión a Decimal**

**Función:** Macroinstrucción que tenga como entrada un valor de 16 bits y que imprima en decimal dicho valor.

- Como parámetro un valor de 16 bits

**Parámetros de entrada:** Recibe una dirección numérica de 16 bits definida como “ent” y que se almacena en el registro AX.

**Registros Utilizados:** Los registros manipulados para la operación de esta macro es: AX, BX, CX, DX.

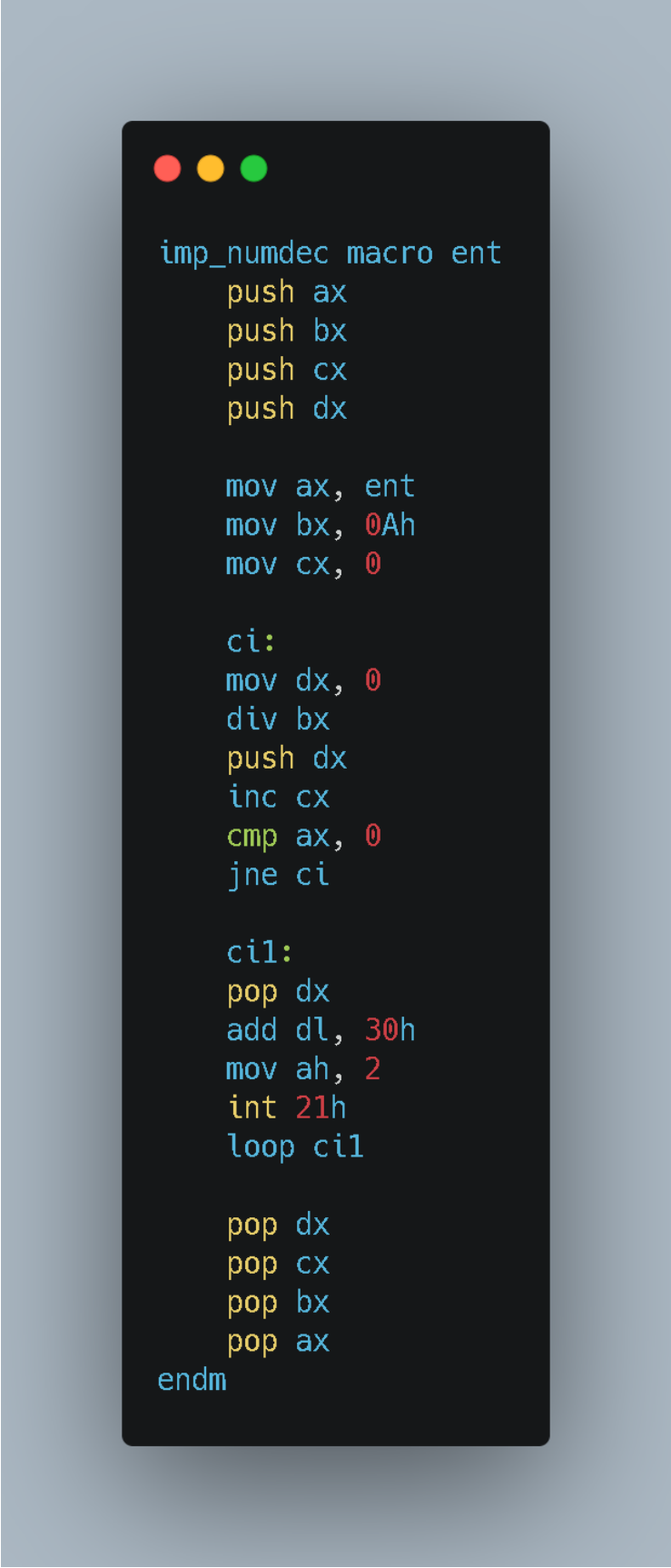
**Que puede aceptar:** Solo acepta direcciones numéricas sin signo y entre el rango de 0 a 65,535 es decir de 16 bits.

**Que no puede aceptar:** No se admiten como parámetros de entrada direcciones de cadena (símbolos, letras o nulos).

**Que retorna:** No retorna valores, solamente realiza procedimiento para la conversión de un valor de entrada de 16 bits en a su valor decimal.

**Código:** Ver Figura 1. Macroinstrucción: *Conversión a decimal*

**Hecho Por:** Brambila Rivera Juan Fernando, Guzmán Boiso Aram Misael, Peña Jiménez José Eduardo, Guzmán García Luis Ángel.



```
imp_numdec macro ent
    push ax
    push bx
    push cx
    push dx

    mov ax, ent
    mov bx, 0Ah
    mov cx, 0

    ci:
    mov dx, 0
    div bx
    push dx
    inc cx
    cmp ax, 0
    jne ci

    cil:
    pop dx
    add dl, 30h
    mov ah, 2
    int 21h
    loop cil

    pop dx
    pop cx
    pop bx
    pop ax
endm
```

Figura 1. Macroinstrucción: Conversión a decimal

**Hecho Por:** Brambila Rivera Juan Fernando, Guzmán Boiso Aram Misael, Peña Jiménez José Eduardo, Guzmán García Luis Ángel.

## 2. MACROINSTRUCCIÓN: **Conversión a Hexadecimal**

**Función:** Macroinstrucción que tenga como entrada un valor de 16 bits y que imprima en hexadecimal dicho valor.

- Como parámetro un valor de 16 bits

**Parámetros de entrada:** Recibe una dirección numérica de 16 bits definida como “ent” y que se almacena en el registro AX.

**Registros Utilizados:** Los registros manipulados para la operación de esta macro es: AX, BX, CX, DX.

**Que puede aceptar:** Solo acepta direcciones numéricas decimales sin signo y entre el rango de 0 a 65,535 es decir de 16 bits.

**Que no puede aceptar:** No se admiten como parámetros de entrada direcciones de cadena (símbolos, letras o nulos).

**Que retorna:** No retorna valores, solamente realiza procedimiento para la conversión de un valor de entrada de 16 bits en a su valor hexadecimal.

**Código:** Ver Figura 2. Macroinstrucción: *Conversión a Hexadecimal*

**Hecho Por:** Brambila Rivera Juan Fernando, Guzmán Boiso Aram Misael, Peña Jiménez José Eduardo, Guzmán García Luis Ángel.

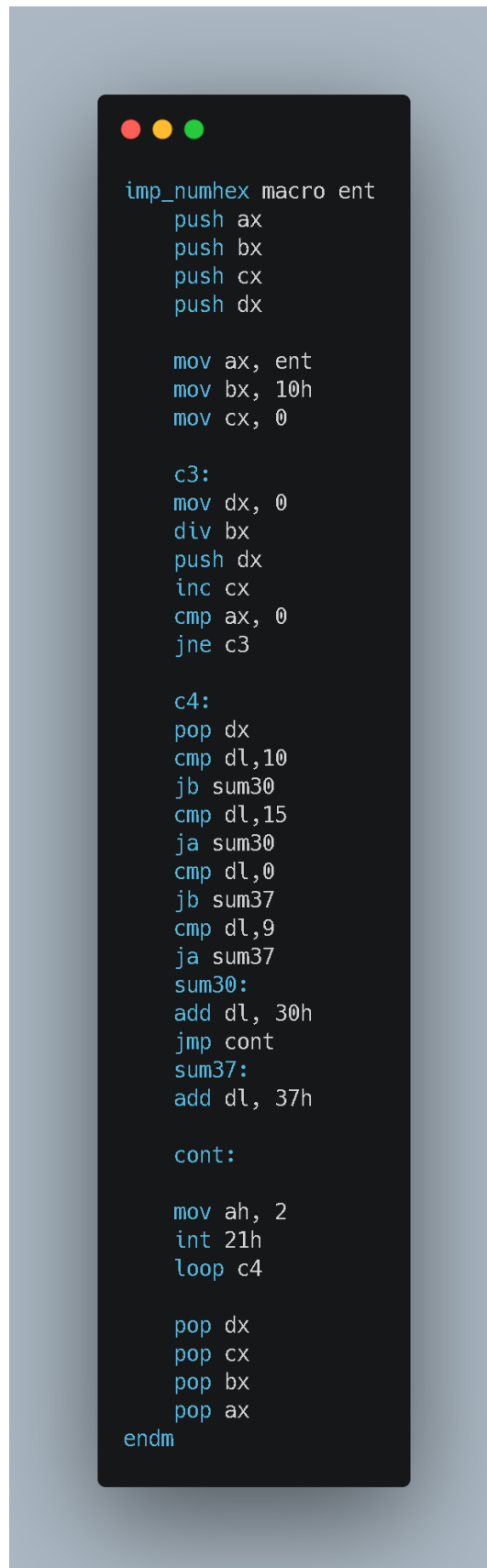


Figura 2. Macroinstrucción: Conversión a Hexadecimal

**Hecho Por:** Brambila Rivera Juan Fernando, Guzmán Boiso Aram Misael, Peña Jiménez José Eduardo, Guzmán García Luis Ángel.

### 3. MACROINSTRUCCIÓN: **Conversión a Binario**

**Función:** Macroinstrucción que tenga como entrada un valor de 16 bits y que imprima en binario dicho valor.

- Como parámetro un valor de 16 bits

**Parámetros de entrada:** Recibe una dirección numérica de 16 bits definida como “ent” y que se almacena en el registro AX.

**Registros Utilizados:** Los registros manipulados para la operación de esta macro es: AX, BX, CX, DX.

**Que puede aceptar:** Solo acepta direcciones numéricas decimales sin signo y entre el rango de 0 a 65,535 es decir de 16 bits.

**Que no puede aceptar:** No se admiten como parámetros de entrada direcciones de cadena (símbolos, letras o nulos).

**Que retorna:** No retorna valores, solamente realiza procedimiento para la conversión de un valor de entrada de 16 bits en a su valor binario.

**Código:** Ver Figura 3. Macroinstrucción: *Conversión a Binario*

**Hecho Por:** Brambila Rivera Juan Fernando, Guzmán Boiso Aram Misael, Peña Jiménez José Eduardo, Guzmán García Luis Ángel.

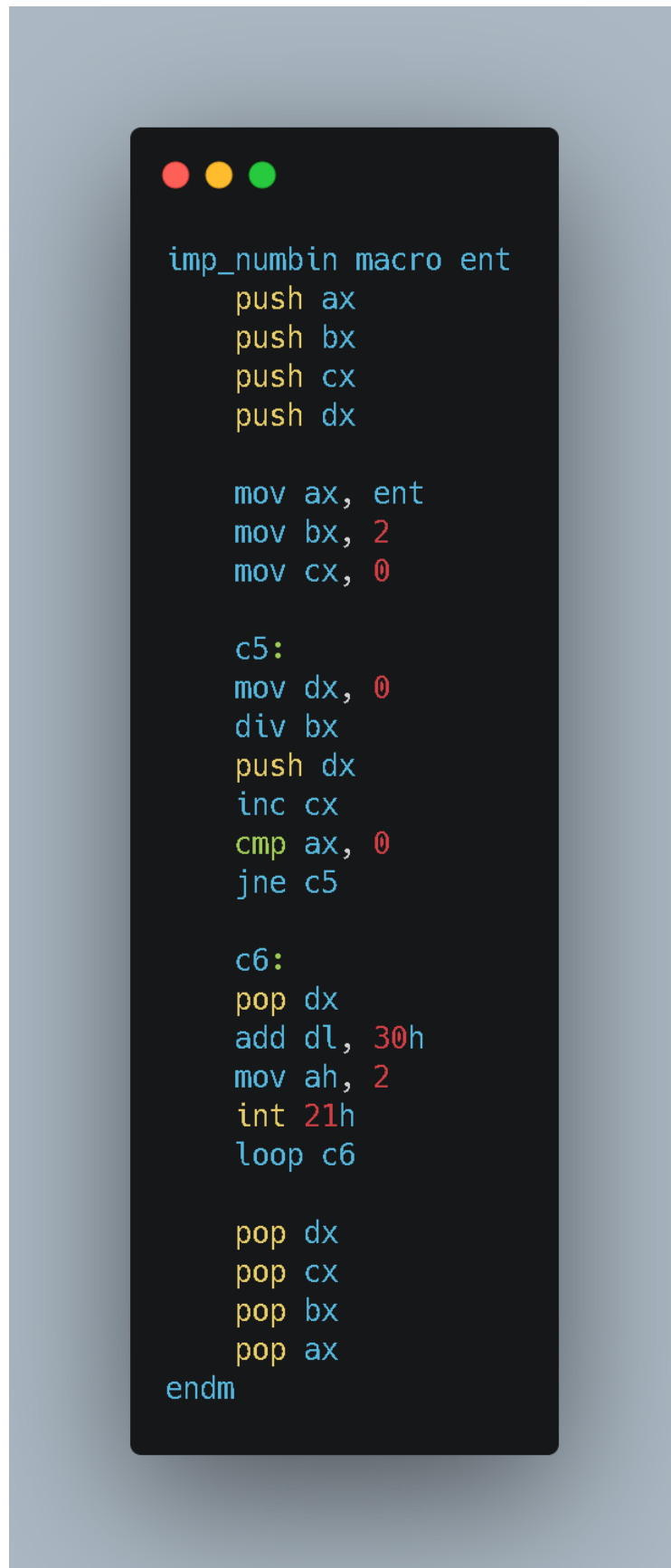


Figura 3. Macroinstrucción: Conversión a Binario

**Hecho Por:** Brambila Rivera Juan Fernando, Guzmán Boiso Aram Misael, Peña Jiménez José Eduardo, Guzmán García Luis Ángel.



## 4. MACROINSTRUCCIÓN: Leer una Cadena

**Función:** Macroinstrucción que permite introducir una cadena desde el teclado.

- El parámetro de entrada debe ser la dirección base de la cadena.

**Parámetros de entrada:** Recibe una dirección de cadena de 16 bits definida como “ent” y que se almacena en el registro DX.

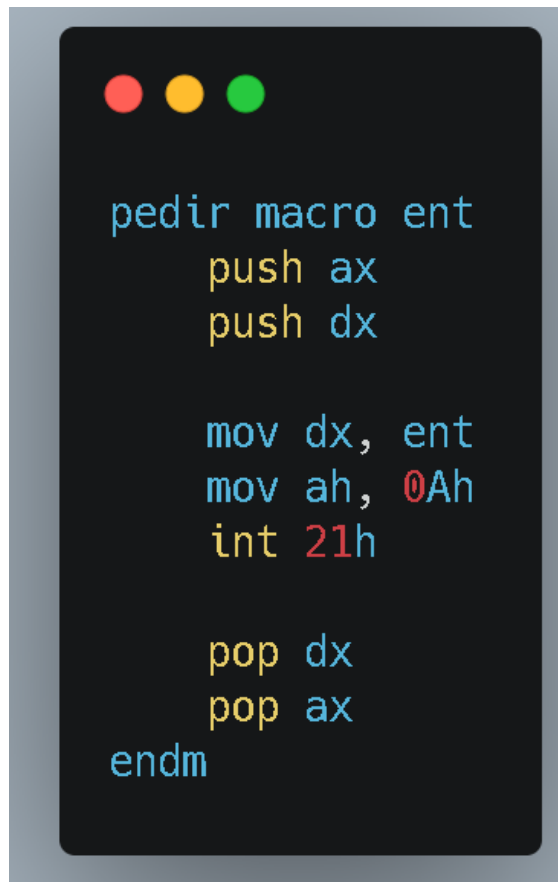
**Registros Utilizados:** Los registros manipulados para la operación de esta macro es: AX, DX.

**Que puede aceptar:** Solo acepta direcciones de cadena (letras, símbolos o nulo) o numéricas introducidas por teclado.

**Que no puede aceptar:** Cualquier dirección de cadena (letras, símbolos o nulo) o numérica es aceptada.

**Que retorna:** Nada.

**Código:** Ver Figura 4. Macroinstrucción: *Leer una cadena*

A screenshot of a code editor window with a dark background and light-colored text. The code is written in assembly language and defines a macro named 'pedir macro ent'. The code includes instructions to push 'ax' and 'dx' onto the stack, move the value of 'ent' into 'dx', set 'ah' to 0Ah, and interrupt 21h. After the interrupt, it pops 'dx' and 'ax' from the stack and ends the macro with 'endm'. The code is color-coded: 'pedir macro ent' is in light blue, 'push ax' and 'push dx' are in yellow, 'mov dx, ent' is in light blue, 'mov ah, 0Ah' has '0Ah' in red, 'int 21h' has '21h' in red, 'pop dx' and 'pop ax' are in yellow, and 'endm' is in light blue.

```
pedir macro ent
    push ax
    push dx

    mov dx, ent
    mov ah, 0Ah
    int 21h

    pop dx
    pop ax
endm
```

Figura 4. Macroinstrucción: Leer una cadena

**Hecho Por:** Brambila Rivera Juan Fernando, Guzmán Boiso Aram Misael, Peña Jiménez José Eduardo, Guzmán García Luis Ángel.

## 5. MACROINSTRUCCIÓN: **Compactar una Cadena**

**Función:** Macroinstrucción que compacta la cadena (el número introducido desde el teclado) en un valor de 16 bits.

- Tiene un parámetro de entrada que es la dirección base de la cadena que representa el número.
- Tiene como parámetro de salida el valor numérico de 16 bits calculado en el procedimiento.

**Parámetros de entrada:** Recibe una dirección base de 16 bits definida como “ent” y que se almacena en el registro BX.

**Registros Utilizados:** Los registros manipulados para la operación de esta macro es: AX, BX, CX, DX.

**Que puede aceptar:** Solo puede aceptar direcciones base de una cadena únicamente de 16 bits, de tipo numérica y sin signo.

**Que no puede aceptar:** No se admiten como parámetros de entrada direcciones de cadena (símbolos, letras o nulos).

**Que retorna:** El valor en Hexadecimal ya compactado de la cadena extraída del registro AX y almacenada en el parámetro “sali”, siempre y cuando no halla error. Si retorna un 0 es que el numero es nulo, y si retorna un 1 significa que el numero es mayor a 65,536 (16 bits).

**Código:** Ver Figura 5. Macroinstrucción: *Compactar cadena*

**Hecho Por:** Brambila Rivera Juan Fernando, Guzmán Boiso Aram Misael, Peña Jiménez José Eduardo, Guzmán García Luis Ángel.

```

dechex macro ent, sali
    push ax
    push bx
    push cx
    push dx
    push di
    push si

    mov bx, ent
    inc bx
    push bx
    mov ch, 0
    mov cl, [bx]
c1:
    inc bx
    sub [bx], byte ptr 30h
    loop c1

    pop si
    mov cl, [si]
    mov ch, 0
    dec cx

    inc si
    mov al, [si]
    cmp al, 00
    je fin8
    mov ah, 0
    cmp cl, 0
    je fin7
    mov di, 0ah
c2:
    mul di
    jc fin9
    inc si
    mov bl, [si]
    mov bh, 0
    add ax, bx
    jc fin9
    loop c2

    jmp fin7

fin9:
    mov sali, 1001h
    jmp fin6
fin8:
    mov sali, 0
    jmp fin6

fin7:
    mov sali, ax
    jmp fin6

fin4:
    pop bx
    mov sali, 0
    jmp fin6

fin5:
    mov sali, 1

fin6:
    pop si
    pop di
    pop dx
    pop cx
    pop bx
    pop ax
endm

```

Figura 5. Macroinstrucción: Compactar cadena

**Hecho Por:** Brambila Rivera Juan Fernando, Guzmán Boiso Aram Misael, Peña Jiménez José Eduardo, Guzmán García Luis Ángel.

## 6. MACROINSTRUCCIÓN: **Calculo de la raíz Cuadrada**

**Función:** Macroinstrucción que calcula la raíz cuadrada de un número.

- El parámetro de entrada es un valor de 16 bits que representa el valor al cual se le calculará la raíz cuadrada.
- Parámetro de salida es un valor de 16 bits.

**Parámetros de entrada:** Recibe una dirección numérica de 16 bits definida como “ent” y que se almacena en el registro AX.

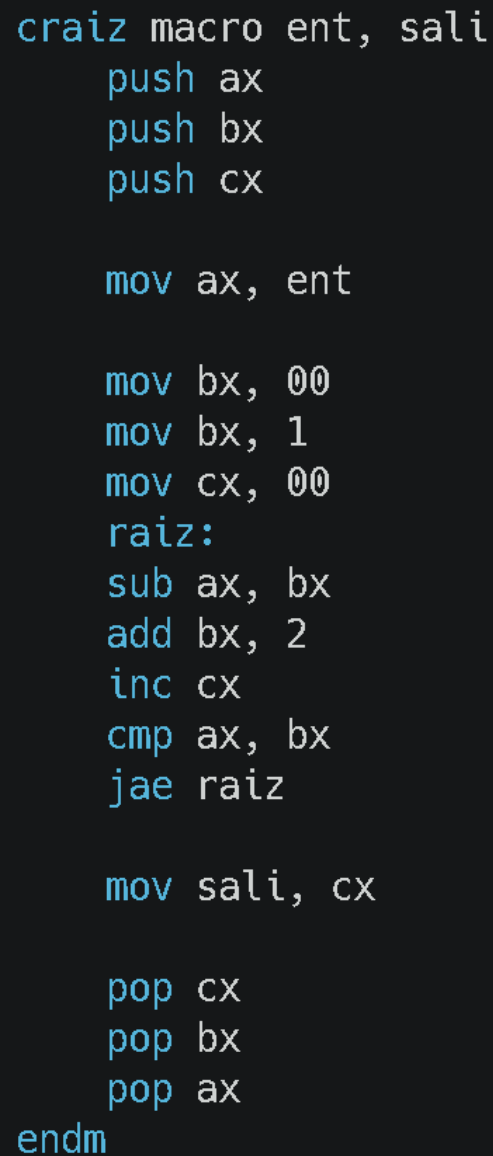
**Registros Utilizados:** Los registros manipulados para la operación de esta macro es: AX, BX, CX.

**Que puede aceptar:** Solo acepta direcciones numéricas decimales sin signo y entre el rango de 0 a 65,535 es decir de 16 bits.

**Que no puede aceptar:** No se admiten como parámetros de entrada direcciones de cadena (símbolos, letras o nulos).

**Que retorna:** Retorna el valor de la raíz ya calculada extraída del registro CX de 16 bits y almacenada en el parámetro “sali.”.

**Código:** Ver Figura 6. Macroinstrucción: *Calculo de la raíz cuadrada*



```
craiz macro ent, sali
    push ax
    push bx
    push cx

    mov ax, ent

    mov bx, 00
    mov bx, 1
    mov cx, 00
    raiz:
    sub ax, bx
    add bx, 2
    inc cx
    cmp ax, bx
    jae raiz

    mov sali, cx

    pop cx
    pop bx
    pop ax
endm
```

Figura 6. Macroinstrucción: Calculo de la raíz cuadrada

**Hecho Por:** Brambila Rivera Juan Fernando, Guzmán Boiso Aram Misael, Peña Jiménez José Eduardo, Guzmán García Luis Ángel.

## 7. MACROINSTRUCCIÓN: Validación de Números

**Función:** Macroinstrucción que valida si un valor introducido desde el teclado contiene exclusivamente dígitos (no cadena nula ni símbolos que no sean dígitos).

- La entrada es la dirección base del valor introducido desde el teclado (como una cadena).
- Salida: 0 si el valor es correcto (solo contiene dígitos); 1 si es cadena nula; 2 si contiene símbolos que no sean dígitos.

**Parámetros de entrada:** Recibe una dirección de cadena de 16 bits definida como “ent” y que se almacena en el registro BX.

**Registros Utilizados:** Los registros manipulados para la operación de esta macro es: AX, BX, CX.

**Que puede aceptar:** Solo acepta direcciones numéricas o de cadena (letras, símbolos o nulo) introducidas por teclado.

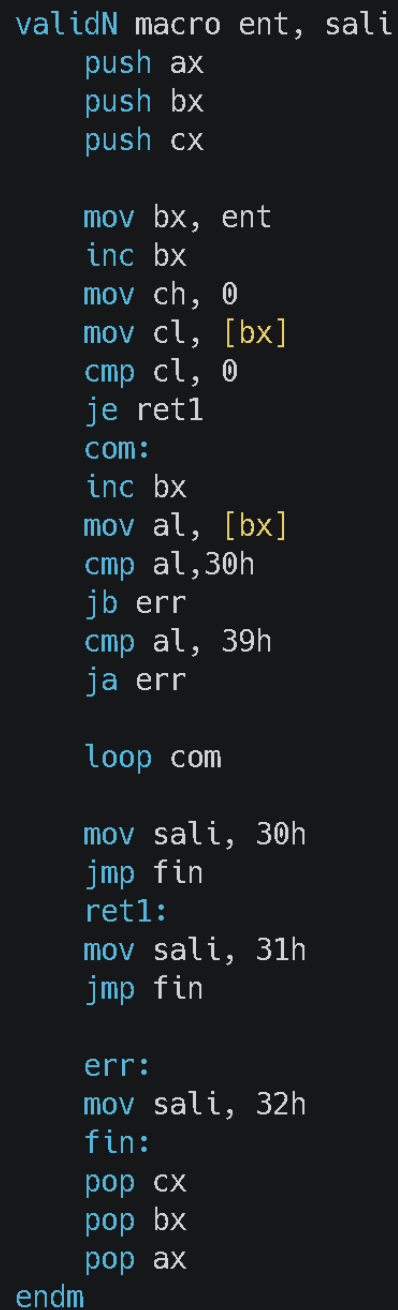
**Que no puede aceptar:** Al ser una macroinstrucción cuya función es validar, solo acepta tanto registros de cadena (letras, símbolos o nulo) como numéricos, siempre y cuando sean de 16 bits.

**Que valida:** Valida principalmente que sea una cadena de puros números sin signo y en un rango de 0-9 cada uno por separado, por medio de la numeración ASCII y con cmp, para así diferenciar si el valor es nulo, un símbolo o si es un dígito.

**Que retorna:** Retornara un valor 0, en caso de ser correcta la cadena, 1 en caso de que sea una cadena vacía o un 2 si la cadena contiene símbolos que no sean validados como dígitos.

**Código:** Ver Figura 7. Macroinstrucción: *Validación de números*

**Hecho Por:** Brambila Rivera Juan Fernando, Guzmán Boiso Aram Misael, Peña Jiménez José Eduardo, Guzmán García Luis Ángel.



```
validN macro ent, sali
    push ax
    push bx
    push cx

    mov bx, ent
    inc bx
    mov ch, 0
    mov cl, [bx]
    cmp cl, 0
    je ret1
com:
    inc bx
    mov al, [bx]
    cmp al, 30h
    jb err
    cmp al, 39h
    ja err

    loop com

    mov sali, 30h
    jmp fin
ret1:
    mov sali, 31h
    jmp fin

err:
    mov sali, 32h
fin:
    pop cx
    pop bx
    pop ax
endm
```

Figura 7. Macroinstrucción: Validación de números

**Hecho Por:** Brambila Rivera Juan Fernando, Guzmán Boiso Aram Misael, Peña Jiménez José Eduardo, Guzmán García Luis Ángel.