

Universidad Autónoma de Baja California
Facultad de Ciencias Químicas e Ingeniería



ORGANIZACIÓN DE LAS COMPUTADORAS Y LENGUAJE ENSAMBLADOR

Practica 9

Procedimientos en el lenguaje del procesador 8086

Docente: Sanchez Herrera Mauricio Alonso

Alumno: Gómez Cárdenas Emmanuel Alberto

Matricula: 1261509

Contenido

TEORIA.....	3
Conversión numérica.....	3
Conversión decimal a otra base	3
Conversión a decimal desde otra base.....	4
Atajos.....	4
Conversión de hexadecimal a Binario	4
Ejemplo de conversión de hexadecimal a binario	4
Ejemplo de conversión de octal a binario.....	5
DESARROLLO.....	6
Parte 1	6
Parte 2	6
CONCLUSIONES.....	8
REFERENCIAS.....	8
ANEXOS	9

TEORIA

Conversión numérica

La conversión de unidades es la transformación del valor numérico de una magnitud expresado en una unidad de medida en otro valor considerado equivalente pero expresado en otra unidad de medida. En el caso de una conversión numérica se convierte el valor expresado de una base numérica a otra.

Conversión decimal a otra base

Un ejemplo de conversión numérica es la conversión de decimal a cualquier otra base. Para ello se siguen estos pasos:

- Paso 1: Dividir el valor decimal a convertir entre el valor de la nueva base.
- Paso 2: El residuo obtenido de la división es el dígito menos significativo (el de hasta la derecha) del número de la nueva base.
- Paso 3: Divide el resultado de la división anterior entre el valor de la nueva base.
- Paso 4: Se anota el residuo de la división del paso anterior como el dígito siguiente (a la derecha) del número de la nueva base.
- Paso 5: Se repite el paso 3 y 4 hasta que el resultado de la división sea 0. El último dígito obtenido será el dígito más significativo del número de la nueva base.

Para ejemplificar esto se hará una conversión sencilla de decimal a binario del número 52.

- Paso 1: Operación: $\frac{52}{2} \rightarrow$ Resultado: 26 \rightarrow Residuo: 0
- Paso 2: Nuevo número: 0
- Paso 3: Operación: $\frac{26}{2} \rightarrow$ Resultado: 13 \rightarrow Residuo: 0
- Paso 4: Nuevo número: 00
- Paso 5: Se repite el paso 3 y 4
- Paso 6: Operación: $\frac{13}{2} \rightarrow$ Resultado: 6 \rightarrow Residuo: 1
- Paso 7: Nuevo número: 100
- Paso 8: Se repite el paso 3 y 4
- Paso 12: Operación: $\frac{6}{2} \rightarrow$ Resultado: 3 \rightarrow Residuo: 0
- Paso 13: Nuevo número: 0100
- Paso 11: Se repite el paso 3 y 4
- Paso 12: Operación: $\frac{3}{2} \rightarrow$ Resultado: 1 \rightarrow Residuo: 1
- Paso 13: Nuevo número: 10100
- Paso 11: Se repite el paso 3 y 4
- Paso 12: Operación: $\frac{1}{2} \rightarrow$ Resultado: 0 \rightarrow Residuo: 1
- Paso 13: Nuevo número: 110100
- Paso 14: Ya no es necesario repetir los pasos 3 y 4, ya que el resultado fue 0

El valor 52 de base decimal equivale al valor 110100 en base binario.

Conversión a decimal desde otra base

Para convertir desde decimal a cualquier otra base

- Paso 1: Determinar el valor posicional de cada dígito (esto depende de la base a la que se quiera convertir).
- Paso 2: Multiplicar los valores obtenidos por los dígitos en la posición correcta.
- Paso 3: Sumar los productos calculados.

Para ejemplificar esto se hará una conversión sencilla de binario a binario del mismo número obtenido anteriormente (110100).

- Paso 1:

Sabemos que el binario es base 2, por lo tanto el valor posicional de cada dígito equivale a 2
 $110100_2 = ((1 * 2^5) + (1 * 2^4) + (0 * 2^3) + (1 * 2^2) + (0 * 2^1) + (0 * 2^0))_{10}$

- Paso 2: $((32 + 16 + 0 + 4 + 0 + 0))_{10}$
- Paso 3: 52_{10}

Atajos

Una manera de hacer una conversión entre 2 bases que no son decimal es primero convertir el número a decimal y después convertirlo a la otra base, sin embargo, esto puede no ser lo más eficiente, sin embargo, para ciertas conversiones entre bases existen “atajos” que te permiten convertir rápidamente desde una base a otra sin necesidad de pasar por la base decimal.

La base binaria fue creada gracias a la facilidad con la que se pueden enviar instrucciones a través de señales eléctricas en un sistema de dos estados. La base octal y hexadecimal fueron creadas ya que son una forma muy sencilla de comprimir datos binarios y representarlos de una manera legible (un solo dígito hexadecimal te permite representar 4 dígitos binarios).

Conversión de hexadecimal a Binario

Este tipo de conversión solo consta de dos pasos:

- Paso 1: Separar el valor por dígitos
- Paso 2: Convertir cada dígito a binario individualmente

Ejemplo de conversión de hexadecimal a binario

Para este ejemplo se va a convertir el valor 78FA hexadecimal a la base binaria

1. $78FAh \rightarrow ???$
2. Paso 1: *Separar los dígitos: 7 – 8 – F – A*
3. Paso 2: *Convertir los dígitos individualmente*
4. $7 = 0111 \rightarrow 8 = 1000 \rightarrow F = 1111 \rightarrow A = 1010$
5. *Valor obtenido 78FA hexadecimal es equivalente a 0111 1000 1111 1010 binario*

El proceso inverso también aplica. La única diferencia es que los dígitos se separan en grupos de 4. Ej. $1001011101011010b \Rightarrow 1001\ 0111\ 0101\ 1010 \Rightarrow 975A$ hexadecimal.

Otro atajo también ocurre cuando se utilizan las bases octal y binario la única diferencia es que los grupos ahora son de 3 bits ya que un dígito octal solo permite representar 3 bits.

Ejemplo de conversión de octal a binario

Para este ejemplo se va a convertir el valor 5246 octal a la base binaria

1. $546 \rightarrow ???$
2. Paso 1: *Separar los dígitos: 5 – 2 – 4 – 6*
3. Paso 2: *Convertir los dígitos individualmente*
4. $5 = 101 \rightarrow 2 = 010 \rightarrow 4 = 100 \rightarrow 6 = 110$
5. *Valor obtenido: 5246 octal es equivalente a 101 010 100 110 binario*

El proceso inverso también aplica. La única diferencia es que los dígitos se separan en grupos de 3 empezando por la derecha.

Ej. 1001011101011010b \Rightarrow 1 001 011 101 011 010 \Rightarrow 113 532 octal.

DESARROLLO

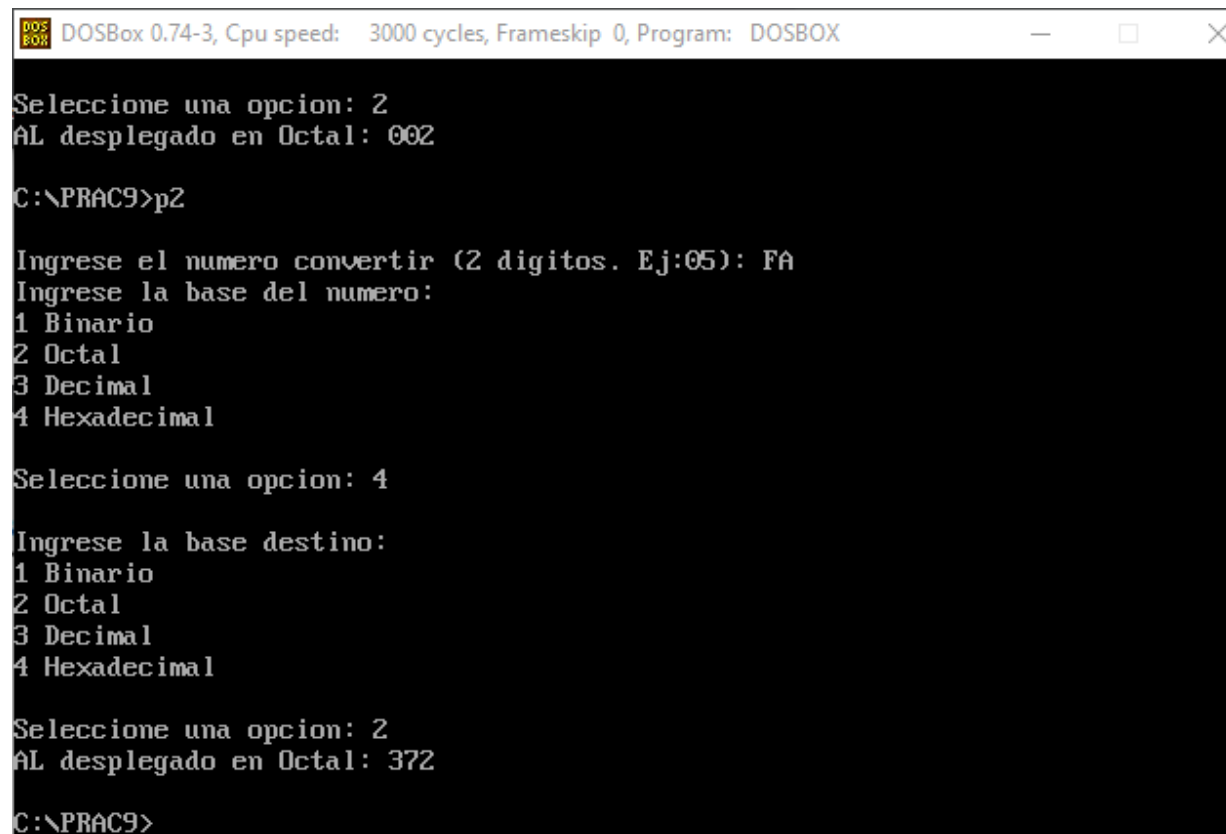
Parte 1

```
AL desplegado en ASCII: .  
AL desplegado en Binario: 00101110  
AL desplegado en Decimal: 046  
AL desplegado en Hexadecimal: 2E  
C:\PRAC9>
```

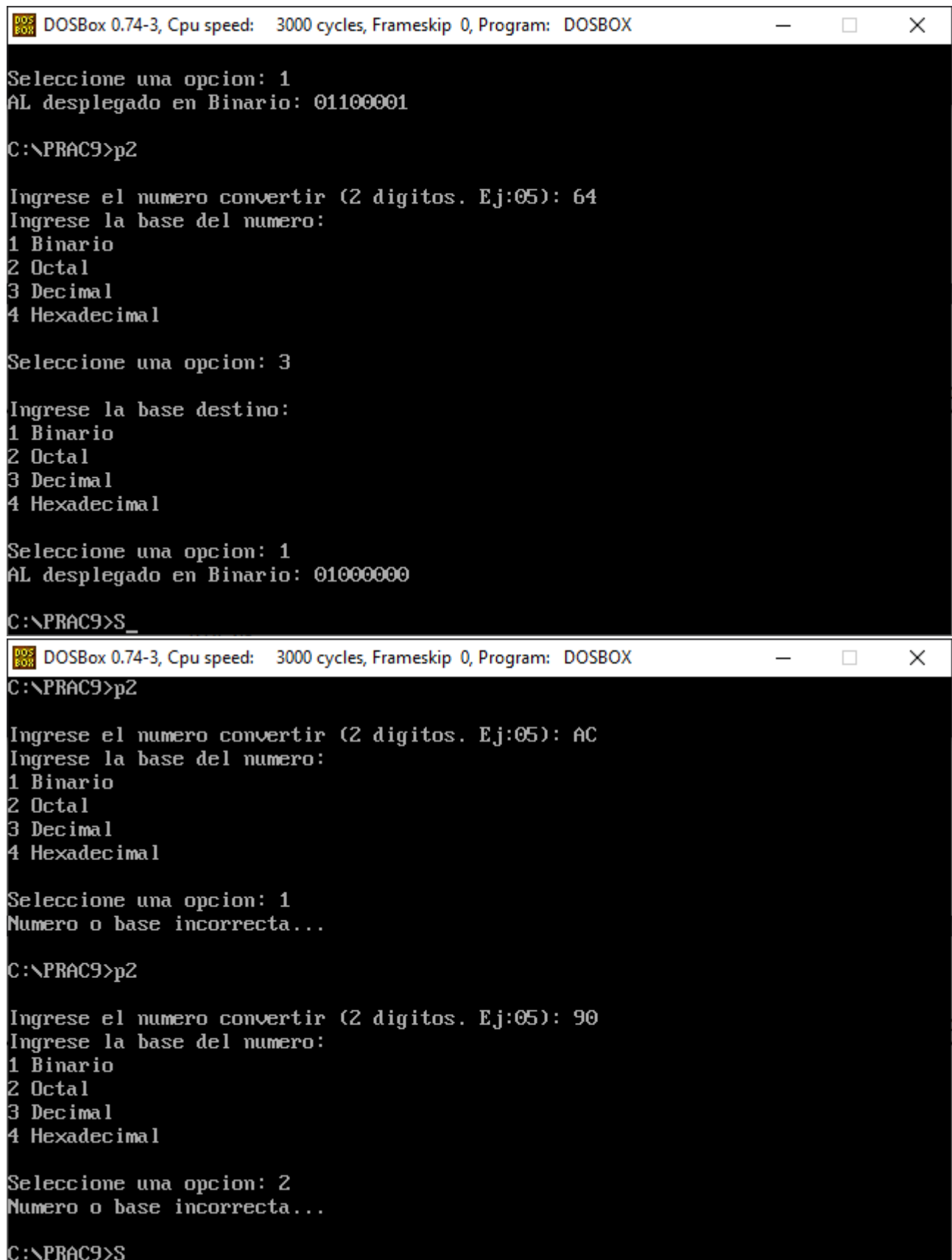
```
AL desplegado en ASCII: L  
AL desplegado en Binario: 01001100  
AL desplegado en Decimal: 076  
AL desplegado en Hexadecimal: 4C  
C:\PRAC9>
```

```
AL desplegado en ASCII: 5  
AL desplegado en Binario: 00110101  
AL desplegado en Decimal: 053  
AL desplegado en Hexadecimal: 35  
C:\PRAC9>_
```

Parte 2



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX  
Seleccione una opcion: 2  
AL desplegado en Octal: 002  
C:\PRAC9>p2  
Ingrese el numero convertir (2 digitos. Ej:05): FA  
Ingrese la base del numero:  
1 Binario  
2 Octal  
3 Decimal  
4 Hexadecimal  
Seleccione una opcion: 4  
Ingrese la base destino:  
1 Binario  
2 Octal  
3 Decimal  
4 Hexadecimal  
Seleccione una opcion: 2  
AL desplegado en Octal: 372  
C:\PRAC9>
```



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Seleccione una opcion: 1
AL desplegado en Binario: 01100001

C:\PRAC9>p2

Ingrese el numero convertir (2 digitos. Ej:05): 64
Ingrese la base del numero:
1 Binario
2 Octal
3 Decimal
4 Hexadecimal

Seleccione una opcion: 3

Ingrese la base destino:
1 Binario
2 Octal
3 Decimal
4 Hexadecimal

Seleccione una opcion: 1
AL desplegado en Binario: 01000000

C:\PRAC9>S_

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\PRAC9>p2

Ingrese el numero convertir (2 digitos. Ej:05): AC
Ingrese la base del numero:
1 Binario
2 Octal
3 Decimal
4 Hexadecimal

Seleccione una opcion: 1
Numero o base incorrecta...

C:\PRAC9>p2

Ingrese el numero convertir (2 digitos. Ej:05): 90
Ingrese la base del numero:
1 Binario
2 Octal
3 Decimal
4 Hexadecimal

Seleccione una opcion: 2
Numero o base incorrecta...

C:\PRAC9>S_
```

CONCLUSIONES

Esta práctica sirvió como un acercamiento aún más a lo que es el lenguaje ensamblador y el su uso de procedimientos y funciones, también sirvió para aprender a utilizar la pila y que el uso de esta es recomendado y, o necesario dependiendo de lo que haga la función, si es un procedimiento es muy importante utilizarla para guardar el estado de los registros antes de modificarlos (siempre y cuando sean modificados).

REFERENCIAS

Number System Conversion - Tutorialspoint. Tutorialspoint.com. (2020). from https://www.tutorialspoint.com/computer_logical_organization/number_system_conversion.htm.

Why is sixteen so sweet? | NASA Space Place – NASA Science for Kids. Spaceplace.nasa.gov. (2020). from <https://spaceplace.nasa.gov/binary-code3/en/>.

ANEXOS

```

MODEL small
.STACK 100h
    ;----- Insert INCLUDE "filename" directives here
    ;----- Insert EQU and = equates here
LOCALS

.DATA    ;Mensaje para interactuar con el usuario
    mens_ascii db 10,13,'AL desplegado en ASCII: ',0
    mens_base  db 10,13,'Ingrese la base del numero: ',10,13,0
    mens_bases db '1 Binario',10,13,'2 Octal',10,13,'3 Decimal',10,13,'4 Hexadecimal',10,13,10,13,'Seleccione una opcion: ',0
    mens_bin    db 10,13,'AL desplegado en Binario: ',0
    mens_convert db 10,13,'Ingrese la base destino:',10,13,0
    mens_dec     db 10,13,'AL desplegado en Decimal: ',0
    mens_hex     db 10,13,'AL desplegado en Hexadecimal: ',0
    mens_oct     db 10,13,'AL desplegado en Octal: ',0
    mens_number  db 10,13,'Ingrese el numero convertir (2 digitos. Ej:05): ',0
    xcption     db 10,13,'Numero o base incorrecta...',0

.CODE ;----- Insert program, subroutine call, etc., here
Principal PROC
    mov ax,@data        ;Inicializar DS al la direccion
    mov ds,ax           ; del segmento de datos (.DATA)

    call changeBase     ;

    mov ah,04ch         ; fin de programa
    mov al,0
    int 21h
Principal ENDP
; --- procedimientos ---
changeBase PROC
    push ax             ;Guarda registros a modificar
    push bx
    push cx
    push dx
    push di
    mov di,offset mens_number
    call putStr
    mov bx,00h          ;Resetea bx
    mov cx,00h          ;Resetea cx
    mov dx,00h          ;Resetea dx

```

```

    call getChar          ;Obtiene el primer digito y lo guarda en ch
    mov ch,al
    call getChar          ;Obtiene el segundo digito y lo guarda en cl
    mov cl,al

    mov di,offset mens_base
    call putStr
    mov di,offset mens_bases
    call putStr
    call getChar          ;Captura el valor de la base y lo guarda en bh
    mov bh,al

    call all2int           ;Convierte el valor de base 'bh' a base hexadecimal
    cmp bl,0               ; Resetea bl si ocurre un error
    je @@exception
    call putNewline

    mov di,offset mens_convert
    call putStr            ;Obtiene la base a convertir y la guarda en dl
    mov di,offset mens_bases
    call putStr
    call getChar
    mov dl,al

    call convert           ;Convierte el numero de base hexadecimal a la base en
contrada en dl

    cmp bl,0               ; Resetea bl si ocurre un error
    je @@exception
    jmp @@end

@@exception:mov di,offset xcption
    call putStr            ;Hace indicar al usuario que hubo un error
@@end:      call putNewline
            pop di          ;Recupera los registros modificados
            pop dx
            pop cx
            pop bx
            pop ax
            ret
changeBase ENDP

;*****
;Convierte el valor en cl a la base encontrada en dl
convert PROC                ;1 = binario, 2 = octal, 3 = decimal, 4 = hexadecimal

```

```

    push ax                ;Guarda registros a modificar
    push cx
    cmp dl,'1'            ;Compara el valor dl y selecciona la base a convertir

    je @@binario
    cmp dl,'2'
    je @@octal
    cmp dl,'3'
    je @@decimal
    cmp dl,'4'
    jne @@exception      ;Si ninguna base es aceptada, llama el error
;Hexadecimal
    mov di,offset mens_hex
    call putStr
    mov al,cl
    call printHex        ;Imprime el valor de al en base hexadecimal
    jmp @@end
@@binario:  mov di,offset mens_bin
    call putStr
    mov al,cl
    call printBin        ;Imprime el valor de al en base hexadecimal
    jmp @@end

@@octal:   mov di,offset mens_oct
    call putStr
    mov al,cl
    call printOct        ;Imprime el valor de al en base hexadecimal
    jmp @@end

@@decimal: mov di,offset mens_dec
    call putStr
    mov al,cl
    call printDec        ;Imprime el valor de al en base hexadecimal
    jmp @@end
@@exception:mov bl,0      ;Resetea bl si hubo un error
@@end:     pop cx         ;Recupera registros modificados
    pop ax
    ret
convert    ENDP
;*****
;Convierte el valor en al ascii, tomado en cuenta como un valor de base bh
all2int    PROC
    push ax              ;Guarda registros a modificar a excepcion de bx y cx
    sub cl,'0'           ;Elimina el valor ascii de cx
    sub ch,'0'

```

```

    mov al,bh
    cmp bh,'1'           ;Selecciona la conversion dependiendo el valor en b
h
    je @@bin2int

    cmp bh,'2'
    je @@oct2int

    cmp bh,'3'
    je @@dec2int

    cmp bh,'4'
    je @@hex2int

    jmp @@exception      ;Si no se selecciono una opcion, manda error

@@bin2int: mov al,ch
            call isBin    ;Compara que ch sea un digito binario valido (0-1)
            cmp bl,00h
            je @@exception
            mov al,cl      ;Compara que cl sea un digito binario valido (0-
1)
            call isBin
            cmp bl,00h
            je @@exception
            mov al,ch      ;Copia ch a cl
            mov bl,02h     ;Copia el valor de base a bl
            jmp @@end      ;Salta al final

@@oct2int: mov al,ch
            call isOct    ;Compara que ch sea un digito octal valido (0-7)
            cmp bl,00h
            je @@exception
            mov al,cl      ;Compara que cl sea un digito octal valido (0-7)
            call isOct
            cmp bl,00h
            je @@exception
            mov al,ch      ;Copia ch a cl
            mov bl,08h     ;Copia el valor de base a bl
            jmp @@end      ;Salta al final

@@dec2int: mov al,ch
            call isDec    ;Compara que ch sea un digito decimal valido (0-
9)
            cmp bl,00h

```

```

        je @@exception
        mov al,cl          ;Compara que cl sea un digito decimal valido (0-
9)

        call isDec

        cmp bl,00h

        je @@exception

        mov al,ch          ;Copia ch a cl
        mov bl,0Ah         ;Copia el valor de base a bl
        jmp @@end          ;Salta al final

@@hex2int: mov al,ch
        call isHex         ;Compara que ch sea un digito hexadecimal valido (0-
F)

        cmp bl,00h
        je @@exception
        cmp bl,02h         ;Si el digito es un numero
        jne @@continue     ;continua
        sub ch,7           ;Si el digito es una letra, restale 7
@@continue: mov al,cl
        call isHex         ;Compara que cl sea un digito hexadecimal valido (0-
F)

        cmp bl,00h
        je @@exception
        cmp bl,02h         ;Si el digito es un numero
        jne @@end2         ;continua
        sub cl,07h         ;Si el digito es una letra, restale 7
        jmp @@end2

@@end2:  mov bl,10h         ;Copia el valor de base a bl
        mov al,ch          ;Copia el valor de ch a al
        jmp @@end

@@exception: mov bl,0h     ;En caso de haber un error, resetea bl

@@end:   mul bl             ;Multiplica al por la base de conversion
        add cl,al          ;Agrega al (ch multiplicado por la base) a cl
        pop ax             ;Recupera registros
        ret

all2int  ENDP
;*****
printBin PROC
        push ax            ; salvar registros a utilizar
        push bx
        push cx

```

```

        mov cx,8           ; inicializar conteo a 16
        mov ah,al          ; BX sera el registro a desplegar
@@nxt:   mov al,'0'         ; preparar a AL para imprimir ASCII
        shl ah,1           ; pasar el MSB de AH a la bandera de acarreo
        adc al,0            ; sumar a AL el valor del acarreo
        call putchar
        loop @@nxt         ; continuar con el proximo bit
        pop cx             ; recuperar registros utilizados
        pop bx
        pop ax
        ret
printBin ENDP
;*****
printOct PROC
        push ax             ;Guarda registros a modificar
        push cx
        mov cl,al           ;Guarda el valor de al en cl
        and al,0C0h         ;Se obtienen los ultimos 2 bits de al 'xx00 0000' (bits mas significativos)
        rol al,1            ;Se acomodan los 2 bits en el lugar mas significativo (0000 00xx)
        rol al,1
        add al,'0'          ;Se le agrega el formato ascii al numero para imprimirlo
        call putChar        ;Se imprime el ascii obtenido

        mov al,cl           ;Se recupera el valor de al guardado en cl
        and al,38h          ;Se obtienen los bits 4, 5 y 6 de al '00xx x000'
        shr al,1            ;Se acomodan los 3 bits en el lugar mas significativo (0000 0xxx)
        shr al,1
        shr al,1

        add al,'0'          ;Se le agrega el formato ascii al numero para imprimirlo
        call putChar        ;Se imprime el ascii obtenido

        mov al,cl           ;Se recupera el valor de al guardado en cl
        and al,07h          ;Se obtienen los primeros 3 bits de al '0000 0xxx'

```

```

        add al,'0'                ;Se le agrega el formato ascii al numero para imprimi
rlo
        call putchar              ;Se imprime el ascii obtenido
        pop cx                    ;Recupera registros a modificar
        pop ax
        ret
printOct ENDP
;*****
printDec PROC
        push ax                  ; salvar registro a utilizar
        push bx
        push cx
        push dx
        mov cx,3                 ; inicializar conteo a 3 (cent-dec-unida)
        mov bx,100               ; iniciar con centenas
        mov ah,0                 ; asegurar AX = AL
@@nxt:  mov dx,0                  ; asegurar DX=0 para usar div reg16
        div bx                   ; dividir DX:AX entre BX
        add al,'0'               ; convertir cociente a ASCII
        call putchar             ; desplegar digito en pantalla
        mov ax,dx                ; pasar residuo (DX) a AX
        push ax                  ; salvar temporalmente AX
        mov dx,0                 ; ajustar divisor para nuevo digito
        mov ax,bx                ; la idea es:
        mov bx,10                ; BX = BX/10
        div bx
        mov bx,ax                ; pasar cociente al BX para nuevo digito
        pop ax                   ; recupera AX
        loop @@nxt               ; proximo digito
        pop dx
        pop cx
        pop bx
        pop ax
        ret
printDec ENDP
;*****
printHex PROC
        push ax                  ; salvar registros a utilizar
        push bx
        push cx
        mov ah,0                 ; asegurar AX = AL
        mov bl,16
        div bl                   ; dividir AX/16 --> cociente en AL y residuo AH
        mov cx,2                 ; para imprimir dos digitos hex
@@nxt:  cmp al,10                ; verifica si cociente AL es menor a 10

```

```

        jb @@print
        add al,7
@@print: add al,30h          ; si es menos a 10 sumar 30h de lo contrario 37h
        call putchar
        mov al,ah          ; pasa residuo (AH) a AL para imprimirlo
        loop @@nxt         ; proximo digito
        pop cx
        pop bx
        pop ax             ; recupera registros utilizados
        ret
prinHex  ENDP
;*****
isBin    PROC ;Revisa que al sea un digito binario valido (0-1)
        cmp al,1          ;Compara que al sea 1 o 0
        jg @@exception
        cmp al,0
        jl @@exception
        mov bl,1h
        jmp @@endCheck
@@exception:
        mov bl,0h         ;En caso de haber un error, resetea bl
@@endCheck: ret
isBin    ENDP
;*****
isOct    PROC ;Revisa que al sea un digito octal valido (0-7)
        cmp al,7          ;Compara que al se encuentre dentro del rango (0-7)
        jg @@exception
        cmp al,0
        jl @@exception
        mov bl,1h
        jmp @@endCheck
@@exception:
        mov bl,0h         ;En caso de haber un error, resetea bl
@@endCheck: ret
isOct    ENDP
;*****
isDec    PROC ;Revisa que al sea un digito decimal valido (0-9)
        cmp al,9          ;Compara que al se encuentre dentro del rango (0-9)
        jg @@exception
        cmp al,0
        jl @@exception
        mov bl,1h
        jmp @@endCheck
@@exception:
        mov bl,0h         ;En caso de haber un error, resetea bl

```



```

@@endCheck: ret
isDec      ENDP
;*****
isHex      PROC ;Revisa que al sea un digito hexadecimal valido (0-F)
            mov bl,1h ;Resetea bl en 1 (si es un digito hexadecimal valido)
            cmp al,16h
            jg @@exception
            cmp al,10h
            jl @@continue ;Si se encuentra dentro del rango (10h-16h)
            mov bl,02h ;Se resetea bl en 2
            jmp @@endCheck
@@continue: cmp al,0h ;Compara que al se encuentre dentro del rango (0-9)
            jl @@exception
            cmp al,9h
            jle @@endCheck
            jmp @@endCheck
@@exception:
            mov bl,0h ;En caso de haber un error, resetea bl
@@endCheck: ret
isHex      ENDP
;*****
putNewline PROC ;Funcion para imprimir un salto de linea
            push ax
            mov al, 0Ah ;Salto de linea
            call putChar
            mov al, 0Dh ;Retorno de carro
            call putChar
            pop ax
            ret
putNewline ENDP
;*****
putChar    PROC ;Funcion para imprimir un char almacenado en al
            push ax ;Guarda el valor del registro a modificar
            mov ah,0Eh ;Selecciona el servicio 0Eh
            int 10h ;Llama la interrupcion 10h
            pop ax ;Recupera registro
            ret
putChar    ENDP
;*****
putStr     PROC ;Funcion para imprimir un string
            push ax ;Guarda registros a modificar
            push di
@@putStr:
            mov al,[di] ;Obtiene el valor de la direccion di y la guarda en al
            cmp al,0h ;Si es caracter de terminacion

```

```
        je @@endputStr ; Dejar de imprimir
        mov ah,0Eh     ;Selecciona el servicio 0Eh
        int 10h        ;Llama la interrupcion 10h
        inc di         ;decrementa di
        jmp @@putStr

@@endputStr:
        pop di         ;Recupera registros modificados
        pop ax
        ret
putStr   ENDP
;*****
getChar  PROC          ;Funcion para leer un cahar y almacenarlo en al
        mov ah,1h     ;Selecciona el servicio 01h
        int 21h        ;Llama la interrupcion 21h
        ret
getChar  ENDP
        END
```