

PROCESAMIENTO DE LOS DATOS

Procesamiento de datos ascii a binario y de binario a ascii

En ensamblador el tipo de dato es un carácter, por lo tanto es necesario procesar este dato y convertir de su correspondiente ascii a su valor binario y viceversa.

Caracter	HEX	BINARIO
'0'	30	00000000
'1'	31	00000001
'2'	32	00000010
'3'	33	00000011
'4'	34	00000100
'5'	35	00000101
'6'	36	00000110
'7'	37	00000111
'8'	38	00001000
'9'	39	00001001
'A'	41	00001010
'B'	42	00001011
'C'	43	00001100
'D'	44	00001101
'E'	45	00001110
'F'	46	00001111

Código para la Subrutina Ascii_Binario

	CMP AL,30h	00110100	34h
	JL ERROR	-00110000	30h
	CMP AL,39h		
	JG LETRA	00000100	4h
	SUB AL,30h ; Restar 30h		
	JMP FIN		
LETRA:	CMP AL,41h	01000100	44h
	JL ERROR	-00110111	37h
	CMP AL,46h		
	JG ERROR	00001101	Dh
	SUB AL,37h ; Restar 37h		
	JMP FIN		
ERROR:	MOV AL,0		
FIN :	RET		

Nota:

Parámetros de entrada: La subrutina toma el dato del **registro AL**, debido a que la subrutina leer al invocar el servicio de lectura de caracter de la int 21h deja el carácter leído en AL.

Parámetro de salida: Deja el dato procesado en el **registro AL**.

Código para la Subrutina Binario_Ascii

```
                CMP DL,9h
                JG SUMA37
                ADD DL,30h
                JMP FIN
SUMA37:        ADD DL,37h
FIN :          RET
```

Nota:

Parámetros de entrada: La subrutina toma el dato del **registro DL**, debido a que la subrutina escribir al invocar el servicio de escritura de carácter de la int 21h solicita que el carácter a escribir esté en el registros DL.

Parámetro de salida: Deja el dato procesado en el **registro DL**.

Ejemplo: Sumar dos numeros de un digito e imprimir su suma (0<=suma<=F)

```
Call LEER_CAR
CALL ASCII_BIN
MOV BL,AL
CALL LEE_CAR
CALL ASCII_BIN
ADD BL,AL
MOV DL,BL
CALL BIN_ASCII
CALL ESCRIBE_CAR
```

Lectura de un número de dos dígitos Hexadecimales (EMPACAR)

Se leen dos caracteres, los cuales se procesan de tal modo que se obtiene su valor binario almacenado solo en un byte.

Por ejemplo: 23h

Car1 Car2
"2" "3"

Ascii	Hexadecimal	Ascii Binario	Procesamiento
"2"	32	00000010	Corrimiento a la izquierda (4): 00100000
"3"	33	00000011	Sumar el primer dato mas el segundo: 00100000 + 00000011 = 0010 0011 nibble: 1º. 2º. valor 2h 3h

Código para la Subrutina Empaqueta

```
Push cx
Call lee_car          ; Leer el primera car
Call ascii_binario    ;Procesa 1er. caracter
Mov cl,04
Shl al,cl             ; Instrucción lógica de corrimiento a la izquierda
Mov ch,al             ; Almacenando el valor de AL a un registro auxiliar
Call lee_car          ; Leer el segundo car
Call ascii_binario    ;Procesa 2o. caracter
Add al,ch             ; Sumar el contenido de los registros
Pop cx
RET
```

Nota :deja el dato en el byte AL

Corrimientos de bits

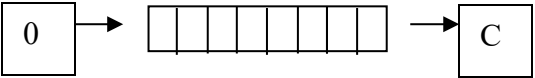


Pueden realizar las siguientes acciones:

- Hacer referencia a un registro o dirección de memoria
- Recorrer bits a la izquierda o la derecha
- Recorrer hasta 8 bits en un byte, 16 bits en una palabra y 32 bits en una palabra doble
- Corrimiento lógico (sin signo), aritmético (con signo)

Sintaxis:

Corrimiento {registro/memoria}, {cl/imm}

El segundo operando es el valor del corrimiento, que es una constante (valor inmediato) o una referencia al registro cl, para el procesador 8088/8086 el valor inmediato solo puede ser 1, si es mayor debe estar contenido en el registro cl.

Instrucción	Descripción	Función
SHR	Desplazamiento lógico a la derecha	
SAR	Desplazamiento aritmético a la derecha	
SHL	Desplazamiento lógico a la izquierda	
SAL	Desplazamiento aritmético a la izquierda	

Al terminar una operación de corrimiento, puede utilizar la instrucción JC (salta si existe acarreo) para examinar el bit desplazado a la bandera de acarreo

Escritura de un número de dos dígitos Hexadecimales (DESEMPACAR)

El valor está almacenado en un byte, el procesamiento para obtener los ASCII asociados consiste:

Binario	Procesamiento
0001 0010 = 12h	Aplicar 4 corrimientos a la derecha : 0000 0001 = 1
0001 0010 = 12h	Aplicar máscara AND F al valor original: 0001 0010 0000 1111 <hr/> 0000 0010 = 2

Código para la Subrutina Desempaqueta

```
Push dx
Push cx
Mov dh,dl    ; Guardando el valor original en DH
Mov cl,4
Shr dl,cl    ; Cuatro corrimientos a la derecha
Call Binario-Ascii
Call Escribe_car
Mov dl,dh    ; Recuperando el dato de DH
And dl,0Fh   ; Aplicando máscara
Call Binario_ascii
Call Escribe_car
Pop cx
Pop dx
RET
```

Ejemplo:

Sumar dos números de 2 dígitos cada uno

```
Call Empaqueta
Mov dl,al
Call empaqueta
Add dl,al
Call desempaqueta ; Obsérvese que la suma no debe ser mayor a un byte
```

Práctica 3:

Codifique un programa con las siguientes opciones:

1. Hallar los pares de un total de datos leídos desde teclado, el total de datos se solicita al usuario, Mostrar en pantalla los números pares e imprimir el total de pares leídos. (datos cuya longitud máxima sea de un byte)
2. Sumar los números pares hallados en el intervalo 1d ... 100d y mostrar en pantalla su suma (2550d=9F6h)

3. Codifique un programa que sume n números aleatorios, los números a sumar serán de longitud de un byte. El programa solicitará el total de números a sumar, al final imprimirá la suma total


Ejemplo:

Codificar el **comando E** permitiendo el almacenamiento de datos byte a byte.

Entrada:

>E Dir_desplazamiento

Salida:

DS:Dir_desplazamiento  Sale del comando

DS:Dir_desplazamiento ‘ ‘ Incrementa en uno al desplazamiento :
DS:Dir_desplazamiento +1

DS:Dir_desplazamiento ‘-‘ Decrementa en uno al desplazamiento :
DS:Dir_desplazamiento -1

DS:Dir_desplazamiento dato	Incrementa en uno al
dato ; es un byte que solo almacena	desplazamiento :
caracteres en el rango del ‘0’...’9’	DS:Dir_desplazamiento +1
y de la ‘A’...’F’	Y almacena el byte leído en la
	dirección corriente

SEGMENTO DE CÓDIGO PARA LEER_DIR_OFFSET

```
CALL EMPAQUETA
MOV BH,AL
CALL EMPAQUETA
MOV BL,AL
```

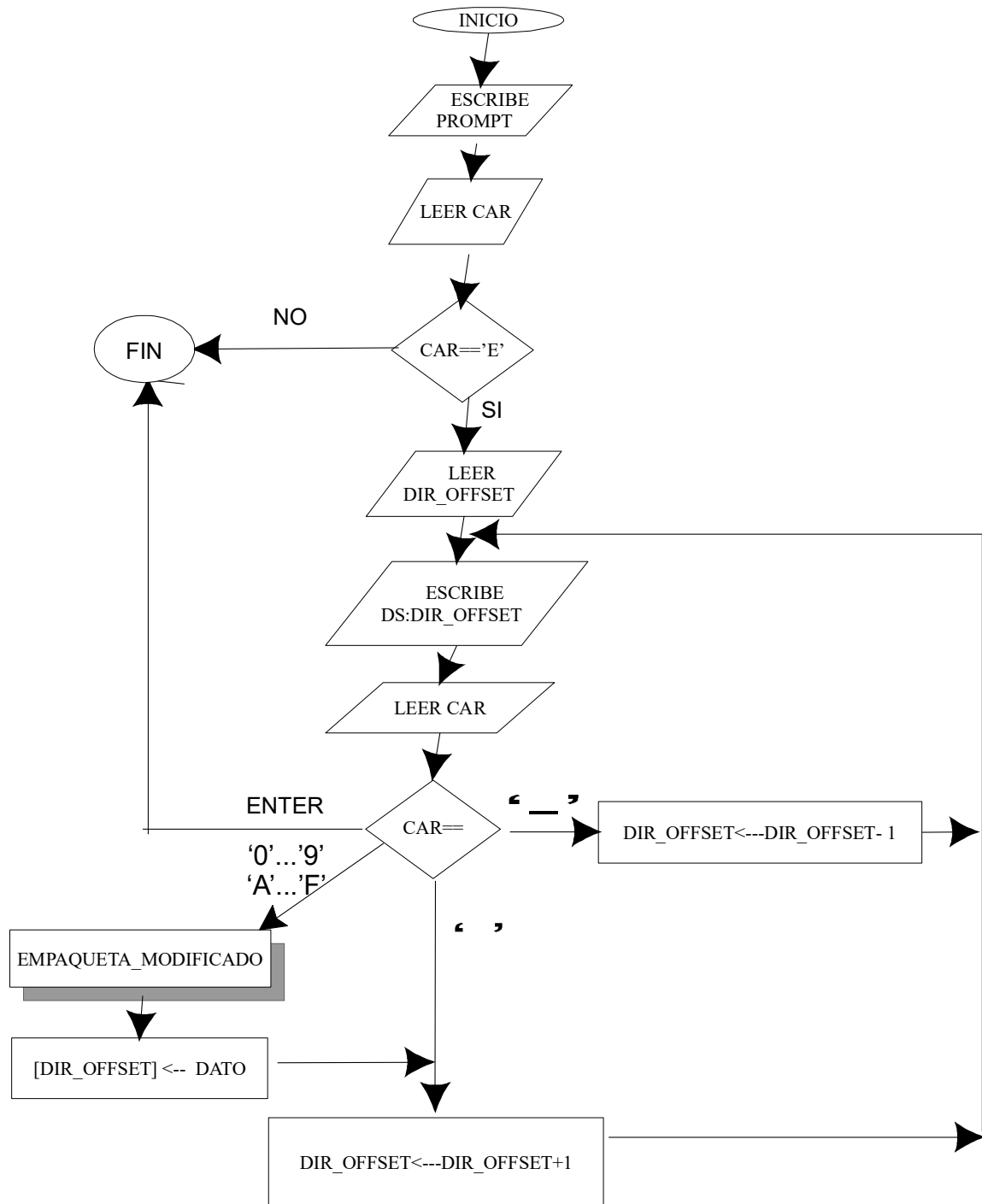
SEGMENTO DE CÓDIGO PARA ESCRITURA_DE_DIR_DE_SEGMENTO

```
MOV AX,DS
MOV DL, AH
CALL DESEMPAQUETA
MOV DL, AL
CALL DESEMPAQUETA
```

Subrutina EMPAQUETA_MODIFICADO

```
PUSH CX
CALL ASCII_BINARIO
MOV CL,4
SHL AL,CL
MOV CH,AL
CALL LEE ; Validar si es un carater entre '0'...'9', enter, espacio o menos
CALL ASCII_BINARIO
ADD AL,CH
POP CX
RET
```

Diagrama de flujo del comando E del depurador



PROYECTO: Codificar el comando E y el comando D