

Lenguaje ensamblador

Guía básica para comenzar a programar

Jorge Luis López Bernal
13111202

LENGUAJES DE INTERFAZ
Ing. Mario Macario Ruiz Grijalva

CONTENIDO

1. INTRODUCCIÓN

¿Qué es el lenguaje ensamblador?	1
¿Cuál es la utilidad del lenguaje ensamblador?	1

2. CONCEPTOS BÁSICOS

Comprendiendo la ejecución de programas	2
Registros	3
Interrupciones	4

3. PROGRAMAS

Estructura básica	5
Elementos básicos en el código	7
Operaciones con cadenas	9
Pantalla	13
Procedimientos	14
Macros	14

4. ANEXOS

Banderas	15
Saltos	15
Interrupciones	16
Tipos de datos	16
Ejemplos	17

1

INTRODUCCIÓN

¿QUÉ ES EL LENGUAJE ENSAMBLADOR?

El principio de la programación se reducía a unos y ceros. Estos formaban datos e instrucciones que son almacenados en memoria y es llamado lenguaje máquina. A través de este idioma, el programador era capaz de comunicarse con la máquina, el cual era un proceso tedioso y muy susceptible a errores.

Por ello, fue necesario hacerse de un lenguaje que fuera más eficiente. Este fue el nacimiento del Lenguaje Ensamblador. Este, se constituye de valores simbólicos que identifican su funcionalidad, además, se incluye en él áreas de trabajo para ayudar a estructurar el programa y hacer más fácil su comprensión.

¿CUÁL ES LA UTILIDAD DE ENSAMBLADOR?

En las actualidad, las áreas de computación y robótica han trabajado de forma arraigada para la creación de nuevas tecnologías productivas y necesarias para el desarrollo humano. Por ejemplo, el control de robots en el área de manufactura, el procesamiento de señales para aparatos médicos y datos sísmicos, y principalmente en los sistemas embebidos presentes en muchos dispositivos electrónicos. Su importancia: trabajar directamente con el microprocesador, lo que permite realizar cualquier tipo de programas que no pueden ser desarrollados con cualquier lenguaje de alto nivel.

2 CONCEPTOS BÁSICOS

COMPRENDIENDO LA EJECUCIÓN DE PROGRAMAS

Al escribir programas en cualquier lenguaje de alto nivel, deben ser traducidos a lenguaje máquina antes de que sean ejecutados. Para esto, se utilizan programas traductores que aceptan archivos de código fuente comprensibles para el humano, y en base a ello genera alguna clase de archivo binario que es comprensible para el CPU.

Como ya se mencionó en la primera parte, el lenguaje ensamblador es un lenguaje de bajo nivel por lo que está muy cerca del lenguaje máquina. Es por esto que el proceso para programar en este lenguaje es el siguiente:



¡Restricciones de edición!

Las instrucciones pueden estar escritas en mayúsculas o minúsculas. Las sentencias pueden comenzar en cualquier columna. No se permite más de 128 caracteres ni líneas múltiples ni códigos de control.

REGISTROS

El procesador está compuesto de contenedores especiales llamados registros. Estos registros tienen diferentes finalidades y ofrecen la ventaja de acceder a su contenido de una forma más rápida. Se pueden clasificar de la siguiente forma:

GENERALES

Son los más utilizados en un programa. Su utilidad fundamental es la de almacenar datos que se usan frecuentemente en el programa. Hay 4 registros de este tipo, cada uno de 16 bits. Cada registro se compone de una parte alta y una parte baja. La parte alta almacena los 8 bytes más significativos, mientras que la parte baja los 8 menos significativos.

Registro acumulador	AX AH AL
Es el registro principal, muy utilizado para operaciones de entrada/salida y operaciones aritméticas.	BX BH BL
Registro base BX	CX CH CL
Utilizado como índice en modos de direccionamiento.	DX DH DL
Registro contador CX	
Es utilizado en las instrucciones de repetición o ciclos, almacena el número de veces que se repetirá una instrucción.	
Registro de datos DX	
Utilizado en operaciones aritméticas y de entrada/salida.	

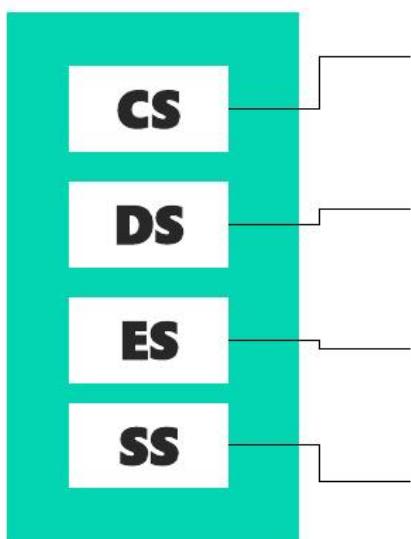
ÍNDICES O APUNTADORES

Su utilidad radica en almacenar la posición de memoria donde se encuentre algún dato necesario para el programa.

SI	Índice fuente, SI. Indica la dirección donde comienzan los datos que se quieren leer de un segmento de datos.
DI	Índice destino, DI. Indica la dirección donde terminan los datos que se quieren leer de un segmento de datos.
BP	Puntero Base, BP. Utilizado por instrucciones que quieren acceder a datos en la pila.

DE SEGMENTO

Un segmento se trata de áreas especiales en un programa con diferente utilidad. El microprocesador dividirá la memoria en segmentos de 64K.



Segmento de código CS

El segmento de código contiene las instrucciones que son ejecutadas. El CS direcciona al segmento de código.

Segmento de datos DS

El segmento de datos contiene los datos, constantes y áreas de trabajo definidos por el programa. El DS direcciona al segmento de datos.

Segmento extra ES

Es una ampliación del segmento de datos.

Segmento de pila SS

El segmento de pila contiene los datos y direcciones guardadas temporalmente. El SS direcciona al segmento de la pila.

ESPECIALES

Estos los utiliza el microprocesador para almacenar sus propios datos.



INTERRUPCIONES*

Las interrupciones son operaciones que suspenden la ejecución de un programa, de manera que el sistema pueda realizar una acción especial. Una vez que se ejecuta la interrupción, se regresa el control al procedimiento que fue interrumpido. Las interrupciones en ensamblador son necesarias para realizar distintas acciones como desplegar mensajes en pantalla o cambiar modo de video.

*NOTA: La lista de banderas y de interrupciones puede ser consultada en los anexos.

3

PROGRAMAS

ESTRUCTURA BÁSICA

Los programas en lenguaje ensamblador consisten en módulos formados por segmentos. Existen dos formas para la definición de segmentos. Utilizando directivas simplificadas o utilizando directivas completas. La sintaxis de cada una de ellas difiere en las palabras reservadas para su declaración.

DECLARATIVAS SIMPLIFICADAS

Las directivas simplificadas son una forma abreviada para definir los segmentos. Para declararlas simplemente es necesario utilizar las palabras reservadas siguientes

Para definir atributos del modelo de memoria. Donde "modeloDeMemoria" puede ser TINY, SMALL, COMPACT, MEDIUM, LARGE.

Crea el segmento de la pila y le asigna un tamaño por defecto indicado en bytes o con el número en hexadecimal.

Crea el segmento de datos. Aquí se declaran las variables que se utilizarán en el programa. Puede contener datos cercanos o lejanos. Algunas variaciones de esta declarativa son .DATA, .DATA?, .FARDATA, .FARDATA?

Crea el segmento de código, que contiene las instrucciones que ejecutará el programa.

.MODEL SMALL

.STACK 100H

.DATA

.CODE

.

. ; Código

.

END

DECLARATIVAS COMPLETAS

Las directivas completas son la forma extendida para definir los segmentos. Es necesario iniciar el segmento y cerrarlo. Cualquier segmento es identificado por un nombre seguido de la palabra reservada **SEGMENT**

La directiva **PAGE** designa el número máximo de líneas para listar en una página y el número de caracteres máximo en cada línea.

Se puede combinar la entrada con otros segmentos cuando son enlazados después de ensamblar, estos tipos son **STACK**, **COMMON** y **PUBLIC**.

Se puede asignar un identificador para la entrada clase, la cual agrupa los segmentos cuando se enlazan. Estos pueden ser '**code**' para el segmento de códigos, '**data**' para el segmento de datos y '**stack**' para el segmento de pila.

PAGE 60,132
TITLE Ejemplo

PILA **SEGMENT** STACK 'stack'
; Declaración del tamaño de pila
PILA ENDS

DATOS **SEGMENT** PARA PUBLIC
...
DATOS ENDS

CODIGO **SEGMENT** PARA PUBLIC 'code'
INICIO:

ASSUME DS:DATOS, CS:CODIGO, SS:PILA
; Código ejecutable
CODIGO ENDS
END INICIO

La directiva **TITLE** es utilizada para añadir un título para el programa.

Puede agregarse el tipo de alineación que indicará el límite en el que inicia el segmento, por ejemplo **PARA** alinea el segmento con el límite de un párrafo.

Directiva **ASSUME**. Tiene como propósito indicar al ensamblador cada segmento del programa, porque los segmentos no están definidos de la misma forma que en las declarativas simplificadas

*NOTA: No es necesario indicar la directiva **ASSUME** en las declarativas simplificadas

ELEMENTOS BÁSICOS EN EL CÓDIGO

El código está constituido por las líneas de programa. Cada una suele estar compuesta de cuatro campos diferentes los cuales están separados por una o más tabulaciones



Existen dos tipos de identificadores, uno es el nombre que se refiere a la dirección de un elemento de dato, y la etiqueta, que son expresiones elegidas por el usuario que sirven para identificar una instrucción dentro del programa. Estas no deben llamarse igual que las palabras reservadas del ensamblador.

Este campo es para la definición de áreas de datos y codificación de instrucciones, las cuales indican al microcontrolador la tarea que debe realizar.

El operando le da información a la operación que actúa en él. Este campo puede tener o no, uno o más operandos separados por comas.

Se trata de texto escrito por el programador, con el fin de explicar brevemente la acción que se realiza para que no sea difícil entender o modificar el programa. Se escriben anteponiendo un punto y coma.

TRANSFERENCIA DE DATOS

La instrucción de transferencia de datos **MOV** permite transferir información entre registros y memoria, memoria y registros y entre los propios registros mediante modos de direccionamiento, en el cual, siempre será **MOV** destino, fuente.

OTRAS OPERACIONES IMPORTANTES

Cuando se crean programas en lenguaje ensamblador, es necesario agregar ciertos conjuntos de instrucciones necesarios para el correcto funcionamiento del programa.

Siempre al iniciar el segmento de código habrá que informar al ensamblador cuál será el contenido de los registros de segmento mediante la directiva ASSUME.

ASSUME DS:DATOS, CS:CODIGO, SS:PILA ES:DATOS

Si se trabaja con declarativas completas

**MOV AX, DATOS
MOV DS, AX**

Si se trabaja con declarativas completas y existe un segmento extra

**MOV AX, DATOS
MOV DS, AX
MOV ES,AX**

Si se trabaja con declarativas simplificadas

**MOV AX, @DATA
MOV DS, AX**

No se debe olvidar terminar la ejecución del programa y devolver el control al BIOS. Esto se logra de dos maneras que a fin de cuentas dan el mismo resultado.

Enviándole valores a la parte alta y a la parte baja del registro acumulador.

**MOV AH, 4CH
MOV AL,00H
INT 21H**

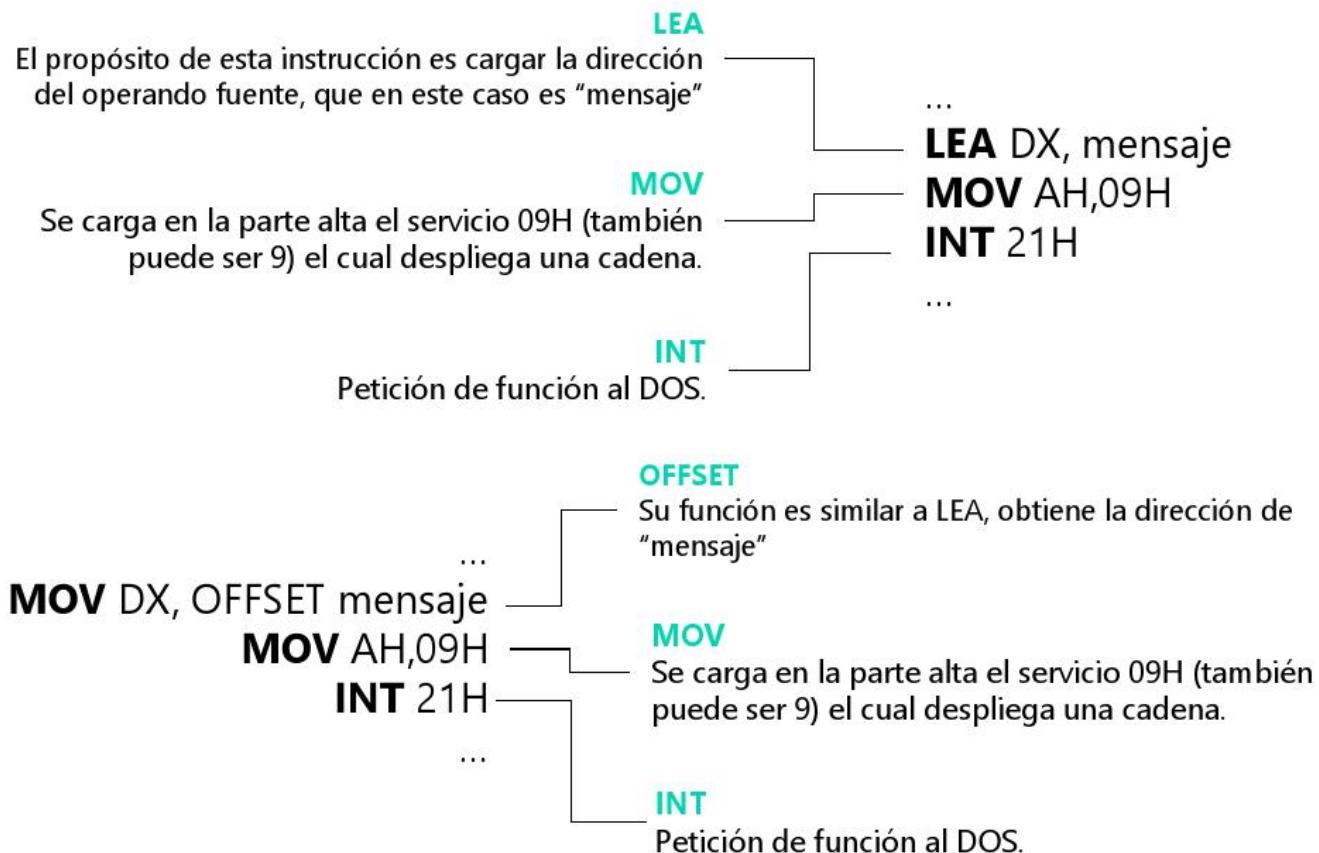
Enviándole el valor completo al registro acumulador.

**MOV AX,4C00H
INT 21H**

OPERACIONES CON CADENAS

MOSTRAR

Esta es una de las acciones más utilizadas en el lenguaje ensamblador, y es gracias a ella que se pueden desplegar mensajes en la pantalla, simplemente teniendo un mensaje que mostrar, el cual puede ser un mensaje previamente definido o alguno ingresado por el usuario.



¿CÓMO DECLARAR UN MENSAJE?

Hay que declarar el mensaje en el segmento de datos:

mensaje DB 'Hola mundo de los Macarios!',10,13,'\$'

El 10,13 es para controlar los saltos de línea.

El símbolo \$ permite terminar la cadena. No se debe olvidar agregarlo.

Para ver los tipos de datos ir al anexo "Tipos de datos"

NOTA: Ir al anexo "Ejemplos" para ver un código completo.

CAPTURA

La captura permite recibir caracteres que el usuario ingresa desde el teclado. Estos se guardan utilizando los índices, variables contadoras, la tabla ASCII y pueden hacerse restricciones utilizando etiquetas y saltos para la captura en base a ella, por ejemplo, que solo acepte números o que solo acepte letras. Es importante señalar que lo que se guarda es una dirección hexadecimal o posición de un carácter, por lo que un número ingresado será tomado como carácter, no como su valor.

Se obtiene el índice de var. Var es un valor entero declarado en el segmento de datos. Se mueve el índice de la variable cad, con esto es posible guardar los caracteres en la variable. Cad es un databyte de tamaño igual a var, en el que se guardarán los caracteres ingresados.

Las etiquetas LEER y SIGA controlan el ciclo de captura.

Las comparaciones marcan el rango de caracteres que se tomarán en cuenta.

Los saltos condicionales son importantes para que el ciclo de captura se realice. Siempre necesita haber una comparación arriba de ellas.

...
MOV CX, var
LEA SI, cad

Esta instrucción nos permite leer el carácter. Ver MODOS DE CAPTURA.

LEER:

MOV AH,01H
INT 21H
CMP AL,"a"
JB SIGA
CMP AL,"z"
J A SIGA

SIGA:

MOV [SI],AL
INC SI
DEC CX
CMP CX,0
JNE LEER

...

MODOS DE CAPTURA

La captura de caracteres se puede realizar de dos formas. Una de ellas es con eco, en la que el carácter será mostrado en la línea de comando. La otra es sin eco, que es lo contrario a la anterior, el carácter no será mostrado. Las instrucciones difieren del servicio que utilizan, 01h o 07h.

Captura con eco

MOV AH,01H
INT 21H

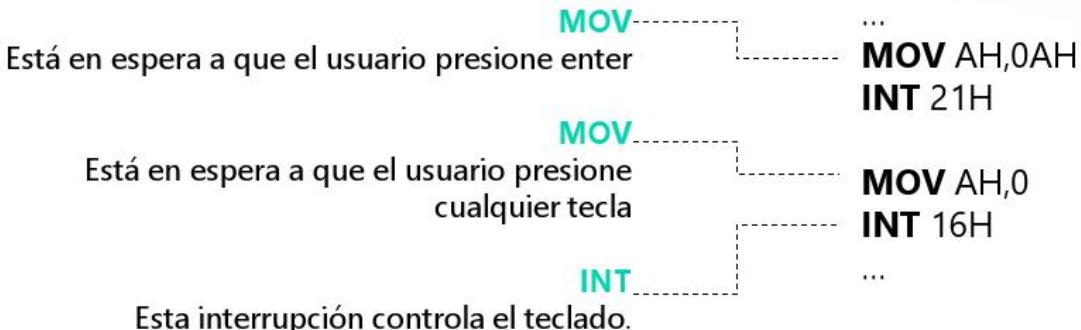
Captura sin eco

MOV AH,07H
INT 21H

NOTA: Ir al anexo “Ejemplos” para ver un código completo., “Saltos” para ver cómo funcionan los saltos.

ESPERA DE TECLADO

De la misma forma en la que se capturan caracteres, existen funciones o servicios* que pueden ayudar al control del programa, por ejemplo, esperar a que el usuario teclee cierta tecla para realizar una acción.



INVERTIR

Con las instrucciones adecuadas se puede lograr invertir una cadena. Esto, podría parecer algo sin utilidad, sin embargo, puede ser utilizado para ver si una cadena es palíndromo, es decir, si se lee igual en cualquier sentido.

Se utiliza la cadena capturada y una extra para guardar la cadena invertida.
Utilizando la pila, se irá agregando cada carácter, y una vez que la cadena termine, pasará los caracteres invertidos a la variable auxiliar.

Al finalizar coloca un '\$' para terminar la cadena.

Se puede observar que el control de estas instrucciones se da por las etiquetas y saltos.
Otra forma de invertir la cadena es utilizando los índices SI Y DI.

Es posible notar que las variables se trabajan como si fueran arreglos de datos.

Pushpila:

**MOV AL,cadena[bx]
PUSH AX
INC BL
CMP BL, longitud
JNE pushpila
MOV BX,0**

Poppila:

**POP AX
MOV invertida[bx], AL
INC BL
GOTOXY 1,14
CMP BL, longitud
JNE poppila
MOV invertida[BX], '\$'**

*SERVICIOS

- 00h lee pulsación de tecla
- 01h obtiene el estado del buffer del teclado
- 02h obtiene el estado del teclado
- 03h establece factor de repetición
- 05h simula la pulsación de una tecla
- 0ah obtiene el id del teclado

NOTA: Ir al anexo "Ejemplos" para ver un código completo., "Saltos" para ver cómo funcionan los saltos.

MAYUSCULAS

Si lo que se quiere es convertir una cadena ingresada por el usuario a mayúsculas, habría que utilizar algo parecido a lo de la captura, agregando una de las dos líneas:

```
AND AL,11011111B  
SUB AL,20H
```

Esta instrucción mueve el índice a lo que sería la posición en la tabla ASCII de los caracteres en mayúscula.

...
MOV CX, CUENTA
LEA SI, VAR

LEER:
MOV AH,01H
INT 21H

La línea que cambia a mayúsculas los caracteres es esta
...
CMP AL,"a"
JB SIGA

CMP AL,"b"
JA SIGA
AND AL,11011111B

SIGA:
MOV [SI],AL
INC SI
DEC CX
CMP CX,0
JNE LEER

Otra manera de convertir una cadena en mayúsculas es capturarla y después leerla y

...
MAYUS MACRO
LEA SI, cadena
LEA DI, mayúsculas

SEGUIR:

MOV AL,[SI]
MOV BL,97
CMP BL,AL
JG UPPERCASE
CMP AL,'z'
JA UPPERCASE
SUB AL,20H
MOV [DI],AL

UPPERCASE:

MOV [DI],AL
INC SI
INC DI
MOV CL,0DH
CMP [SI],CL
JNE SEGUIR
MOV [DI], BYTE PTR '\$'
ENDM
...

Se utilizan los índices SI y DI para hacer el cambio a mayúsculas.

Se establece el rango, si el carácter está dentro de él salta y convierte

Se hace el cambio en el carácter, después se mueve el carácter cambiado a la parte alta del registro acumulador y por último se mueve al DI.

Se incrementan los valores de SI y DI y se salta para repetir el ciclo hasta que lo que se encuentra en SI sea un enter (0DH)

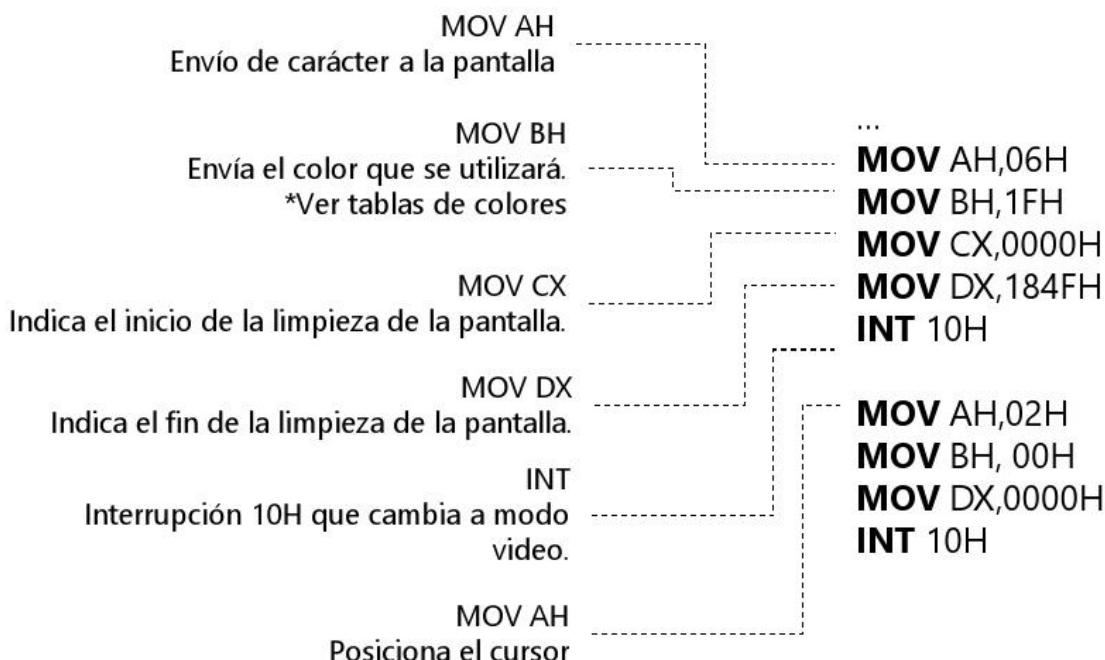
MINUSCULAS

El proceso para convertir en minúsculas es muy similar al de mayúsculas, simplemente es necesario cambiar los límites de la tabla ASCII para que esté entre los caracteres en mayúsculas 'A' y 'Z' y también se reemplaza la línea SUB AL,20H por ADD AL,20H. O bien, se omite la línea AND AL,11011111B.

NOTA: Ir al anexo "Ejemplos" para ver un código completo., "Saltos" para ver cómo funcionan los saltos.

PANTALLA

Las siguientes instrucciones tienen el fin de posicionar el cursor en pantalla, cambiar de color a la pantalla, o limpiar la pantalla, es decir, hacer algo similar al comando `cls` del MS-DOS. Dándole algunas indicaciones será posible lograrlo.



COLORES

OFH - NEGRO
1FH – AZUL
2FH – VERDE
3FH – CYAN
4FH – ROJO
5FH – MAGENTA
6FH – MARRÓN
7FH – GRIS CLARO
8FH – GRIS OSCURO
9FH – AZUL CLARO
0AFH – VERDE CLARO
0BFH – CYAN CLARO
0CFH – ROJO CLARO
0DFH – MAGENTA CLARO
0EFH - BLANCO

POSICIONAR EN PANTALLA

Si lo que se desea es posicionar el cursor en algún lugar de la pantalla, entonces es necesario utilizar las siguientes instrucciones.

XOR BH,BH
MOV DH,POSY
MOV DL,POSY
MOV AH,0
MOV AL,3
INT 10H

La primera línea parametriza o limpia la parte alta del registro base.
Luego se le mandan a la parte alta y baja del registro de datos las posiciones en filas y columnas que quieran recorrer.
Una vez hecho esto, se posiciona el cursor y se utiliza la interrupción de video.

PROCEDIMIENTOS

Un procedimiento es una colección de instrucciones relacionadas que realiza una tarea específica. Puede contener un conjunto de instrucciones que se deseé ejecutar en varias partes del programa. Su diferencia con los procedimientos es necesario llamarlo con la directiva CALL.

SINTAXIS

Nombre PROC NEAR

- La declaración de un procedimiento se realiza asignando un identificador seguido de la palabra reservada PROC y después se agrega NEAR o FAR dependiendo si se trata de un procedimiento cercano o lejano. Luego se incluyen las instrucciones que se requieran.
- La directiva RET regresa al segmento donde el procedimiento fue invocado.
- Se termina el procedimiento con ENDP.

RET

Nombre ENDP

LLAMADA AL PROCEDIMIENTO

CALL Nombre

MACROS

Un macro es un conjunto de instrucciones asociadas a un identificador (el nombre del macro). Este conjunto de instrucciones es invocado como una sola instrucción o macroinstrucción. Normalmente estas instrucciones se repiten varias veces en un programa. La diferencia que tiene con los procedimientos es que puede pasar parámetros directamente para trabajar con ellos.

SINTAXIS

Nombre MACRO arg1,arg2

- Se declara con un identificador seguido de la directiva MACRO y algunos parámetros con los que el macro creado vaya a trabajar.
- Se escriben las instrucciones que se requieran.
- Se termina el macro con ENDM.

ENDM

LLAMADA AL MACRO

Nombre

NOTA: Ir al anexo “Ejemplos” para ver un código completo con el uso de macros y procedimientos.

4

ANEXOS

BANDERAS

OF (Desbordamiento): indica desbordamiento de un bit de orden AL después de una operación aritmética.

DF (dirección): designa la dirección hacia la izquierda o hacia la derecha para mover o comparar cadenas.

IF (Interrupción): indica interrupción externa.

TF (Trampa): Permite la operación del procesador en modo de un paso.

SF (Signo): contiene el signo resultante de una operación aritmética.

ZF (Cero): Indica el resultado de una operación aritmética o de comparación.

AF (Acarreo): Contiene un acarreo externo del bit 3 en un dato de 8 bits.

PF (Paridad): Contiene el acarreo de orden más alto después de una operación aritmética.

SALTOS

JZ (Jump if zero): Salta si cero

JNZ (Jump if not zero): Salta si no es cero

JC (Jump if carry): Salta si acarreo

JNC (Jump if not carry): Salta si no acarreo

JO (Jump if overflow): Salta si
desbordamiento

JNO (Jump if not overflow): Salta si no
desbordamiento

JS (Jump if sign): Salta si signo

JNS (Jump if not sign): Salta si no signo

JP/JPE (Jump if parity /parity even): Salta si
hay paridad (paridad par)

JNP/JPO (Jump if not parity/parity odd):
Salta si no hay paridad (paridad non)

JA (Jump if above): Salta si por encima

JAE (Jump if above or equal): Salta si por
encima o igual

JB (Jump if below): Salta si por debajo

JBE (Jump if below or equal): Salta si por
debajo o igual

JE (Jump if equal): Salta si igual

JG (Jump if greater): Salta si mayor

JGE (Jump if greater or equal): Salta si
mayor o igual

JL (Jump if less): Salta si menor

JLE (Jump if less or equal): Salta si menor
o igual

INTERRUPCIONES

INTERRUPCIONES DE BIOS

INT 00H: División entre cero. Llamada por un intento de dividir entre cero.

INT 01H: Un solo paso. Avanza un paso a través de la ejecución.

INT 02H: Interrupción no enmascarable. Para condiciones graves de hardware.

INT 03H: Punto de ruptura. Detiene ejecución por depuración de programas.

INT 04H: Desbordamiento.

INT 05H: Imprime pantalla. Hace que el contenido de la pantalla se imprima.

INT 08H: Sistema de cronómetro. Actualiza la hora del sistema y fecha.

INT 09H: Interrupción del teclado. Provocada por presionar o soltar tecla.

INT 0BH, INT 0CH: Control de dispositivo serial. COM1 y COM2.

INT 0DH, INT 0FH: Control de dispositivo paralelo. LPT1 y LPT2.

INT 0EH: Control de disco flexible. Señala actividad de disco flexible.

INT 10H: Despliegue en video.

INT 11H: Determinación del equipo.

INT 12H: Determinación del tamaño de la memoria.

INT 13H: Entrada/Salida de disco.

INTERRUPCIONES DE DOS

INT 20H: Termina el programa.

INT 21H: Petición de función al DOS.

INT 22H: Dirección de terminación.

INT 23H: Dirección de Ctrl + Break para transferir rutina del DOS.

INT 24H: Manejador de error crítico.

INT 25H: Lectura absoluta de disco.

INT 26H: Escritura absoluta de disco.

INT 27H: Termina pero reside en memoria.

INT 2FH: Interrupción de multiplexión.

INT 33H: Manejador del ratón.

TIPOS DE DATOS

DB (Define Byte): Reserva datos de tamaño byte (8 bits)

DW (Define Word): Reserva datos de tipo palabra (16 bits)

DD (Define Doubleword): Reserva datos de tipo doble palabra (32 bits)

DQ (Define Quadword): Reserva datos de tipo cuádruple palabra (64 bits)

DT (Define Terabyte): Reserva datos de tipo terabyte (80 bits o 10 bytes)

EJEMPLOS

PEDIR CADENA AL USUARIO Y MOSTRARLA EN MAYÚSCULA Y MINÚSCULA

```
12 DATO SEGMENT
13     CUENTA=10
14     MENSAJE DB "Introduce cadena1",10,10,"$"
15     MENSAJE2 DB "Introduce cadena2",10,10,"$"
16     VAR DB CUENTA DUP(0)
17     VAR2 DB CUENTA DUP(0)
18     CRLF DB 10,10,"$"
19 DATO ENDS
20 ;-----
21 ;Segundo de codigo
22 ;-----
23 CODIGO SEGMENT
24     ASSUME DS:DATO, CS:CODIGO
25
26     MAIN:
27         MOV AX,DATO
28         MOV DS,AX
29
30     ; INSTRUCCIONES PARA LEER LA PRIMER CADENA
31         MOV CX,CUENTA           ;Mueve al registro contador el valor de cuenta (10)
32         LEA SI,VAR              ;Lee caracter por caracter hasta que la cadena esté completa
33
34         MOV BX,OFFSET MENSAJE   ;Le manda al registro de dato el mensaje para que sea mostrado
35         MOV AH,09H
36         INT 21H
37 LEER:
38         MOV AH,01H               ;Etiqueta LEER, se regresa aquí para la entrada del teclado caracter por caracter
39         INT 21H                 ;Permite capturar lo que el usuario teclea
40         CMP AL,"a"             ;Compara que esté entre a
41         JB SIGA                ;Si se cumple la comparación va a la etiqueta SIGA
42         CMP AL,"z"              ;si no, lo compara con z
43         JA SIGA                ;si se cumple va a la etiqueta SIGA
44         SIGA: MOV [SI],AL        ;Guarda los caracteres. No continúa si no es una letra.
45         INC SI                 ;Incremento del índice
46         DEC CX                 ;Decremento del contador
47         CMP CX,0                ;Compara que el contador no sea 0
48         JNE LEER                ;Si no es igual a 0, regresa a la etiqueta LEER
49
50     ;USO DEL ESPACIO
51         MOV AH,09H
52         MOV BX,OFFSET CRLF
53         INT 21H
54     ; INSTRUCCIONES PARA LEER LA PRIMER CADENA
55         MOV CX,CUENTA           ;Vuelve a asignar 10 al registro contador
56         LEA SI,VAR2              ;La variable que se llena ahora es var2
57
58         MOV BX,OFFSET MENSAJE2   ;Le manda al registro de dato el mensaje para que sea mostrado
59         MOV AH,09H
60         INT 21H
61 LEER2:
62         MOV AH,01H               ;Se repiten las instrucciones para leer la segunda cadena
63         INT 21H                 ;Permite capturar
64         CMP AL,"a"             ;compara con a
65         JB SIGA2               ;si se cumple va a etiqueta SIGA2
66         CMP AL,"z"              ;si no se cumple, compara con z
67         JA SIGA2                ;si se cumple va a etiqueta SIGA2
```

```

69 SIGA2: MOV [SI],AL           ;guarda los caracteres, no continua si no es letra
70     INC SI
71     DEC CX
72     CMP CX,0           ;incrementa SI
73     JNE LEER2          ;decrementa el contador
74                                     ;compara con 0, si no es igual
75                                     ;vuelve a la etiqueta LEER2
76
77                                     ;USO DEL ESPACIO
78     MOV AH,09H
79     MOV DX,OFFSET CRLF
80     INT 21H
81
82                                     ;USO DEL ESPACIO
83     MOV AH,09H
84     MOV DX,OFFSET CRLF
85     INT 21H
86
87                                     ;MOSTRAR CADENA 1 Y 2 EN MINÚSCULAS
88     MOV AX,0DH           ;manda al acumulador la interrupción para mostrar en pantalla
89     MOV DX,OFFSET VAR2
90     INT 21H               ;se envía al registro de dato la variable VAR2
91
92                                     ;CONVIERTA A MAYÚSCULA SEGUNDA CADENA
93     ;Algunas de las siguientes instrucciones no han sido vistas en clase
94     ;pero son resultado de buscar una manera de pasar la cadena a mayúsculas
95
96     MOV SI,0              ;le manda al registro indice el valor 0
97     MOV CX,CUENTA         ;el registro de
98     MAYUS: MOV AL,VAR2[SI] ;se puede ver como un arreglo de caracteres
99     SUB AL,20H             ;resta 20h para convertir en mayúscula
100    MOV VAR2[SI],AL        ;Mueve el resultado anterior al arreglo de caracteres VAR2
101    INC SI                ;Incrementa el valor del registro indice
102    LOOP MAYUS            ;Genera un ciclo en el programa, retornando a la etiqueta MAYUS
103
104    ;"Subtract 20h to make uppercase"
105    ;MOSTRAR CADENA 2 EN MAYÚSCULAS
106    MOV AX,0DH           ;Manda al acumulador la interrupción para mostrar en pantalla
107    MOV DX,OFFSET VAR2
108    INT 21H               ;se envía al registro de dato la variable VAR2 convertida en mayúsculas
109
110    CODIGO ENDS          ;Finalización
111    END MAIN              ;Cierra el segmento de código
                                ;Cierra el main

```

CAMBIAR EL COLOR DE LA PANTALLA

```

8 .MODEL SMALL
9 .STACK 100H
10 .DATA
11     TITULO DB 13,10,'Ejercicio 3. Uso de procedimientos',13,10
12     .... DB 13,10,'Pinta la pantalla. Presiona "r" para rojo o "m" para magenta.'
13     .... DB 13,10,'$'
14
15     MSJM DB 13,10,'Se ha elegido el color magenta','$'    ;Las siguientes son líneas que aparecerán
16     MSJR DB 13,10,'Se ha elegido el color rojo','$'    ;al inicio del programa para que el usuario
17                                         ;seleccione una opción.
18
19 .CODE
20     INICIO:
21         MOV AH,0FH      ;Habilita el modo video
22         INT 10H        ;Ejecuta
23         MOV AH,0        ;Limpia la pantalla
24         INT 10H        ;Ejecuta
25
26         MOV AX,@DATA    ;Apunta al .DATA declarado, el acumulador recibe una dirección de memoria
27         MOV DS,AX        ;Se envía al segmento de datos
28
29         ;Muestra título
30         MOV BX,OFFSET TITULO    ;Muestra el título
31         MOV AH,9H
32         INT 21H
33
34         MOV AH,0        ;Prepara el teclado para recibir ingreso de caracteres. Lee con eco. 0ih
35         INT 16H
36         CMP AL,'r'      ;Compara la parte baja del acumulador con el carácter 'r'
37         JE ROJO        ;Si la comparación anterior es igual, salta a la etiqueta
38         CMP AL,'m'      ;Compara la parte baja del acumulador con el carácter 'm'
39         JE MAGENTA     ;Si la comparación anterior es igual, salta a la etiqueta
40
41         JNE INICIO
42
43 ROJO:          ;ETIQUETA ROJO
44     CALL CROJO
45 MAGENTA:       ;ETIQUETA MAGENTA
46     CALL CMAGENTA
47
48     CROJO PROC NEAR
49         MOV AH,0EH
50         MOV BH,4FH
51         MOV CX,0000H
52         MOV DX,184FH
53         INT 10H
54
55         MOV AH,02H
56         MOV BH,00H
57         MOV CX,0000H
58         INT 10H
59
60         MOV DX,OFFSET MSJR    ;muestra mensaje
61         MOV AH,9H
62         INT 21H
63
64         MOV AX,4C00H    ;termina programa
65         INT 21H
66         RET
67
68 CROJO ENDP
69 CMAGENTA PROC NEAR
70     MOV AH,0EH
71     MOV BH,5FH
72     MOV CX,0000H
73     MOV DX,184FH
74     INT 10H
75
76     MOV AH,02H
77     MOV BH,00H
78     MOV CX,0000H
79     INT 10H
80
81     MOV DX,OFFSET MSJM    ;muestra mensaje
82     MOV AH,9H
83     INT 21H
84
85     MOV AX,4C00H    ;termina programa
86     INT 21H
87     RET
88 CMAGENTA ENDP
89 END INICIO

```

OPERACIONES CON CADENA

```
6 TITLE Menu de opciones para operaciones con cadenas.
7
8 ;Las siguientes variables se declaran para ser utilizadas en el
9 ;espaciado del programa, y con esto evitar estar agregandolas
10 ;cada vez
11 cr EQU 10
12 lf EQU 13
13 ;-----
14 ; SEGMENTO DE DATOS: Contiene las variables a utilizar
15 ;-----
16 VARIABLES SEGMENT PARA PUBLIC
17     ;mensaje de cabecera
18     c1 DB 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX$'
19     c2 DB 'XXXXXXXXXXXX B I E N V E N I D O XXXXXXXX$'
20     c3 DB 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX$'
21     c4 DB 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXJL$'
22
23     ;mensaje para pedir la cadena
24     m1 DB 'Ingrese una cadena de caracteres: $'
25
26     ;menu de opciones
27     m2 DB 'Que desea hacer con la cadena? ',10,13
28         DB ' a) Invertirla ',10,13
29         DB ' b) Convertirla a mayusculas ',10,13
30         DB ' Elija una opcion: $'
31
32     ;mensajes de resultados y opciones de salida
33     m3 DB 'LA CADENA CAMBIADA ES: $'
34     m4 DB 'CADENA EN MAYUSCULAS: $'
35     m5 DB 'PRESIONE ENTER PARA SEGUIR...$'
36     m6 DB 'DESEA SALIR DEL PROGRAMA? (S/N)$'
37
38     max_caracteres DB 40 ;maximo de caracteres a introducir
39     lencad DB 0           ;longitud de cadena
40     cadena DB 40 DUP(0)   ;guarda la cadena
41     girat DB 40 DUP(0)    ;guarda la cadena invertida
42     mayus DB 40 DUP(0)    ;guarda la cadena en mayusculas
43     espacio DB cr,lf,'$'  ;variable de espacio
44
45     max_caracteres2 DB 2 ;longitud maxima
46     lencad2 DB 0          ;longitud leida
47     cadena2 DB 2 DUP(0)   ;buffer que contendrá el texto a introducir
48
49     max_caracteres3 DB 2 ;
50     opcion DB 2 DUP(0)   ;PARA ELEGIR OPCION DEL MENU
51     CUENTA=40            ;variable cuenta para el contador
52
53 VARIABLES ENDS
54 ;-----
55 ; SEGMENTO DE CODIGO
56 ;-----
57 CODIGO SEGMENT PARA PUBLIC 'code'    ;Declaración del segmento de código público
58     main PROC FAR             ;procedimiento main
59         ASSUME CS:CODIGO,DS:VARIABLES,SS:PILA,ES:VARIABLES      ;Asignación de segmentos
60
61         ;Movimientos obligatorios antes de comenzar
62         MOV AX,VARIABLES
63         MOV DS,AX
64         MOV ES,AX
65
```

```

66 ;*****
67 ; MACROS
68 ;*****
69 ;***** LIMPIAR LA PANTALLA *****
70 LIMPIAR MACRO :Para limpiar la pantalla
71     MOV AX,0600H :limpiar la pantalla
72     MOV BH,4FH :envia el color
73     MOV CX,0000H :delimita el área a limpiar
74     MOV DX,184FH :delimita el área a limpiar
75     INT 10H :interrupcion de video
76 ENDM
77
78 ;***** POSICIONA EL CURSOR *****
79 GOTOXY MACRO x,y :Para posicionar el cursor
80     XOR BH,BH :limpia la parte alta del registro base
81     MOV DL,x :envia argumentos como coordenadas en x
82     MOV DH,y :y al registro de datos
83     MOV AH,02H
84     INT 10H :interrupcion de video
85 ENDM
86
87 ;***** IMPRIME UNA CADENA *****
88 IMPRIME MACRO arg1 :Para imprimir cadena
89     PUSH AX :Agrega el registro acumulador a la pila
90     PUSH DX :Agrega el registro de datos a la pila
91     LEA DX,arg1 :Carga el argumento que recibe este macro en la dirección del registro de datos
92     MOV AH,9
93     INT 21H
94     POP AX :Saca el registro acumulador de la pila
95     POP DX :Saca el registro de datos de la pila
96 ENDM
97
98 ;***** LEE UNA CADENA *****
99 LEE MACRO arg1 :Para leer la cadena
100    PUSH AX :Agrega el registro acumulador a la pila
101    PUSH DX :Agrega el registro de datos a la pila
102    LEA DX,arg1 :Carga el argumento que recibe este macro en la dirección del registro de datos
103    MOV AH,10
104    INT 21H :Codigo de funcion
105    POP DX :Saca el registro de datos de la pila
106    POP AX :Saca el registro acumulador de la pila
107 ENDM
108
109 ;***** INVIERTE LA CADENA *****
110 INVERTIR MACRO :Para invertir la cadena
111     pushpila:
112         MOV AL,cadena[bx] :
113         PUSH AX :Agrega el registro acumulador a la pila
114         INC BL :incrementa el valor de la parte baja del registro base
115         CMP BL,lencad :compara la variable de la longitud leida con la parte baja del registro base
116         JNE pushpila :si no es igual, dirige a la etiqueta pushpila
117         MOV BX,0 :
118
119     poppila:
120         POP AX :Saca el contenido del registro acumulador de la pila
121         MOV girat[bx],al :mete los caracteres de forma inversa en la variable girat
122         INC BL :incrementa el valor de la parte baja del registro base
123         GOTOXY 1,15 :posiciona el cursor
124         CMP BL,lencad :compara la variable de la longitud leida con la parte baja del registro base
125         JNE poppila :repite hasta que se inviertan todos los caracteres
126         MOV girat[bx],'$' :le pasa un $ para darle fin a la cadena
127
128     IMPRIME m3 :llama a macro y le envia el mensaje 3 para mostrar
129     IMPRIME girat :llama a macro y le envia la variable invertida a mostrar
130     IMPRIME espacio :muestra espacio
131     JMP CONTINUAR :regresa a continuar para capturar opcion de salida
132 ENDM

```

```

134 ;***** CONVIERTA A MAYUSCULAS *****
135 MAYUSCULAS MACRO :Para convertir la cadena en mayúsculas
136
137     UPPERCASE:
138         MOV AL,[cadena] ;Mueve lo que está en la cadena en el índice de BX
139         SUB AL,20H ;Esta instrucción convierte lo que está en AL en mayúscula
140         MOV mayus[BX],AL ;Una vez que lo convierte lo agrega a la cadena mayus
141         INC BL ;Incrementa la parte baja del registro base
142         CMP BL,lencad ;la compara con lencad
143         JNE UPPERCASE ;si no es igual, regresa a la etiqueta MAYUS
144         MOV mayus[BX],'$' ;Una vez que termina, agrega el carácter de fin de la cadena
145         ... ;y en las siguientes instrucciones la imprime
146         GOTONY 1,15
147         IMPRIME espacio
148         IMPRIME m4
149         IMPRIME mayus
150         JMP CONTINUAR
151
152 ENDM
153
154 ;***** INICIO Y PROCEDIMIENTOS *****
155
156 INICIO:
157     LIMPIAR
158     GOTONY 1,0 ;todos los GOTONT positionan el cursor
159     IMPRIME c1 ;imprime cabecera
160
161     GOTONY 1,2
162     IMPRIME c2 ;imprime cabecera
163
164     GOTONY 1,3
165     IMPRIME c3 ;imprime cabecera
166
167     GOTONY 1,5
168     IMPRIME c4 ;imprime cabecera
169
170     GOTONY 1,6
171     IMPRIME m1 ;Pide que se introduzca un mensaje
172
173     GOTONY 1,7
174     LEE max_caracteres ;llama a macro que lee, envía argumento max_caracteres
175     IMPRIME espacio
176     MOV BX,0 ;se da al registro base el valor 0
177
178     GOTONY 1,9
179     IMPRIME m2 ;Imprime menu de opciones
180
181     GOTONY 1,13
182     :LEE max_caracteres3
183
184     MOV AH,01H
185     INT 21H
186     CMP AL,'a' ;Compara con a minúscula
187     JE INVIERTE_CAD ;Si la comparación es igual, dirige a la etiqueta
188     CMP AL,'A' ;Compara con a mayúscula
189     JE INVIERTE_CAD
190     CMP AL,'b' ;Compara con b minúscula
191     JE MAYUS_CAD ;Si la comparación es igual, dirige a la etiqueta
192     ... ; ...

```

```

192     CMP AL,'B'          ;Compara con b mayuscula
193     JE MAYUS_CAD
194
195     JMP CONTINUAR      ;Salto a etiqueta continuar
196
197     INVIERTE_CAD:      ;esta etiqueta dirige al macro para invertir cadena
198         INVERTIR        ;llamada al macro
199     MAYUS_CAD:          ;esta etiqueta dirige al macro para convertir en mayúsculas
200         MAYUSCULAS       ;llamada al macro
201
202     ;*****
203     ; OPCION DE SALIDA
204     ;*****
205     CONTINUAR:          ;etiquete para mostrar la confirmación de salida
206         GOTOKY 24,17    ;sposicionamiento del cursor
207         IMPRIME m6      ;imprime el mensaje 6
208
209         GOTOKY 39,18    ;sposicionamiento del cursor
210         LEE max_caracteres2 ;llama al macro para leer opción de salida
211
212         CMP cadena2[0],'s' ;compara con s minuscula para salir
213         JE SALIR
214         CMP cadena2[0],'S' ;compara con s mayuscula para salir
215         JE SALIR
216
217     JMP INICIO          ;si no se cumplen las condiciones, salta a la etiqueta inicio
218
219     SALIR:
220
221     MOV AX,4C00H
222     INT 21H
223     ;*****
224     ; FIN
225     ;*****
226 main ENDP
227 CODIGO ENDS
228
229 ;-----
230 ; SEGMENTO DE PILA
231 ;-----
232 PILA SEGMENT PARA STACK 'stack'
233     DB 128 DUP(0)      ;definicion del tamaño de la pila
234 PILA ENDS
235
236 END main

```

ESCRIBIR CARÁCTER EN POSICIÓN ESPECÍFICA DE LA PANTALLA

```
6 :Permitir ir a una cadena de la pantalla y escribir cualquier carácter
7
8 .gotoxy macro fila,col ;recibe parametros para posicionar segun ellos
9     mov ah,02h
10    mov dh,fila ;se indica la posición en fila al registro de datos en su parte alta
11    mov dl,col ;se indica la posición en columnas al registro de datos en su parte baja
12    mov bh,0h ;manda un nulo al registro base en la parte baja
13    int 10h
14
15 .endm
16
17 pantalla macro que :macro que muestra el que
18     mov ah,02h ;servicio para mostrar en pantalla
19     mov dl,que ;muestra el que al registro de datos en su parte baja
20     int 21h
21
22 .endm
23
24 .model small
25 .data
26 .code
27 .startup:
28     ;movimientos iniciales obligatorios
29     mov ax,@data
30     mov ds,ax
31     mov ax,0003h ;limpia la pantalla
32     int 10h ;interrupción de video
33     gotoxy 28,19 ;posiciona 10,10
34     pantalla 41h ;imprime carácter
35     mov ah,01h ;despliega y muestra
36     int 21h
37     mov ax,4c00h ;termina el programa
38     int 21h
39 .end startup
```

CONVIerte A HEXADECIMAL

```
4 .MODEL SMALL           ;Modelo de memoria
5 .CODE                  ;área de código
6 BEGIN:                ;Etiqueta de inicio del programa
7     MOV AX,@DATA         ;Inicializa el registro DS con la dirección DATA
8     MOV DS,AX             ;por @DATA (segmento de datos)
9
10    MOV DX,OFFSET Titulo ;obtiene la dirección de la cadena de caracteres
11    MOV AH,09              ;Usamos la función 09H de la interrupción 21H
12    INT 21H               ;Para desplegar la cadena cuya dirección obtuvimos
13
14    MOV CX,1E              ;contador de caracteres que se mostrarán //ES COMO INICIALIZAR VARIABLES
15    MOV BX, OFFSET Cadena ;Permite acceso a la cadena donde se encuentran los valores a desplegar //SE PONE EN EL PRIMER ÍNDICE DE LA CADENA
16
17 CICLO:                ;Etiqueta para generar un ciclo
18    MOV AL,CX              ;Coloca en AL el número a traducir y lo traduce //CARGA LA PARTE BAJA DEL CONTADOR EN LA PARTE BAJA DEL ACUMULADOR
19    XLAT                   ;Usando la instrucción XLAT //
20
21    MOV DL,AL              ;Coloca en DL el valor a ser desplegado por medio de la //LO QUE ESTÁ EN LA PARTE BAJA DEL ACUMULADOR, LO MANDA A LA PARTE BAJA DEL DE DATOS
22
23    MOV AH,02              ;función 2 de la interrupción 21h //02 NO ES INTERRUPCIÓN. LE MANDA UN 2 AL ACUMULADOR
24    INT 21H                ;despliega el carácter
25
26    MOV DL,10              ;Salta una linea desplegando el carácter 10 //MANDA 10 A PARTE BAJA DE DATOS
27    INT 21H                ;Despliega el carácter
28
29    MOV DL,13              ;Produce un retorno de carro desplegando el carácter 13 //MANDA 13 A PARTE BAJA DE DATOS
30    INT 21H                ;Despliega el retorno de carro
31
32    LOOP CICLO             ;Decrementa en uno la CX y brinda a la etiqueta ciclo
33    ;siempre y cuando CX no sea igual a cero
34
35    MOV AH,4CH              ;Utiliza la función 4C de la interrupción 21h para
36    INT 21H                ;finalizar el programa
37
38 ;inicio del segmento de datos
39 .DATA                 ;DEFINE SEGMENTO DE DATOS
40     Titulo DB 13,10,'Lista de numeros hexadecimales del 0 al 15'
41     |     DB 13,10,'$'   ;Cadena a desplegar al inicio del programa
42     Cadena DB ' FEDCBABEDCBA9876543210' ;Cadena con los dígitos hexadecimales //asignación de bytes, se puede ver como un arreglo o pila, el $ es para mandar algo.
43
44 .STACK                ;Declaración del segmento de la pila
45
46 END BEGIN             ;Declaración del final del programa
```

Enlaces

Conceptos básicos

http://dac.escet.urjc.es/docencia/ETC-ITIG_LADE/teoria-cuat1/tema9_conceptos_basicos_ensamblador.pdf

<http://www.15dejuniomnr.com.ar/blog/apunteca/Ciclo%20Superior/ECA/Digital%202/Teoria/Conceptos%20Basicos%20Sobre%20la%20Programacion%20en%20Assembler.pdf>

<http://ensam1.blogspot.mx/2005/09/51elementos-basicos.html>

<http://www.circuitoselectronicos.org/2012/02/introduccion-al-lenguaje-ensamblador.html>

Registros

<ftp://ftp.ehu.es/cidira/dptos/depjt/Apuntes/Estructura%20computadores%20I/pdf/capitulo6%20Los%20registros%20del%20microprocesador.pdf>

Servicios de INT

https://es.wikipedia.org/wiki/Int_16h#Lista_de_servicios_de_la_INT

Interrupciones

http://ict.udlap.mx/people/oleg/docencia/ASSEMBLER/asm_interrup_21.html

Directivas

<http://www.ingelec.uns.edu.ar/icd2759/docs/Materiales%20de%20consulta/DIRECTIVAS%20DEL%20ASM86.pdf>

Colores

<http://stackoverflow.com/questions/28790368/how-to-get-16-colors-for-background-in-mcga-bios-text-mode-al-03h/28790456#28790456>

