**UNIVERSIDAD DE SAN CARLOS DE GUATEMALA**

**FACULTAD DE INGENIERIA**

**CIENCIAS Y SISTEMAS**

**ARQUITECTURA DE COMPUTADORES 1**

**ING. OTTO RENE LEIVA**

**AUX. JORGE GUTIERREZ**

**SECCION A**

**MANUAL TECNICO**

**PROYECTO 2 ASSEMBLER (MASM)**

**NOMBRE: OSCAR RENE CUELLAR MANCILLA**

**CARNET: 201503712**

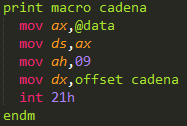
**FECHA: 5 DE NOVIEMBRE DE 2017**

**ENSAMBLADOR USADO EN LA PRACTICA: (MASM)**

El Microsoft Macro Assembler (MASM) es un ensamblador para la familia x86 de microprocesadores. Fue producido originalmente por Microsoft para el trabajo de desarrollo en su sistema operativo MS-DOS, y fue durante cierto tiempo el ensamblador más popular disponible para ese sistema operativo. El MASM soportó una amplia variedad de facilidades para macros y programación estructurada, incluyendo construcciones de alto nivel para bucles, llamadas a procedimientos y alternación (por lo tanto, MASM es un ejemplo de un ensamblador de alto nivel). Versiones posteriores agregaron la capacidad de producir programas para los sistemas operativos Windows. MASM es una de las pocas herramientas de desarrollo de Microsoft para las cuales no había versiones separadas de 16 bits y 32 bits.

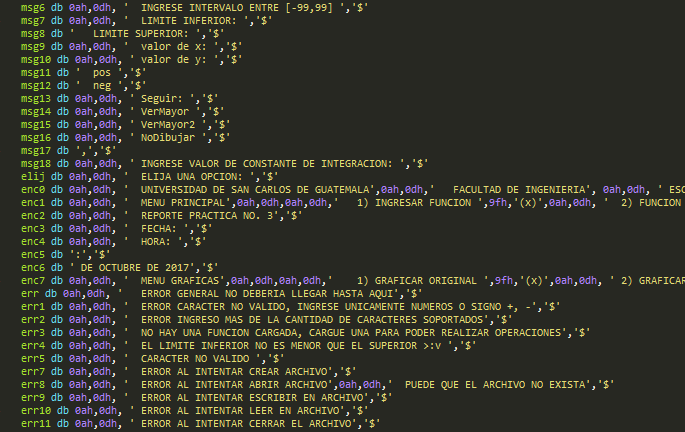
**CÓDIGO RELEVANTE DE LA PRACTICA:**

Durante la realización de la práctica se utilizaron varios macros, el más utilizado sería el macro llamado “print” con el cual mando como parámetro una cadena a imprimir, se manda al registro AX el @data que representa que se va a escribir una cadena, con el mov ah,09 le indico a mi interrupción 21H que iniciare una impresión de cadena en pantalla y al registro DX le mando la dirección donde se almacena mi cadena a imprimir



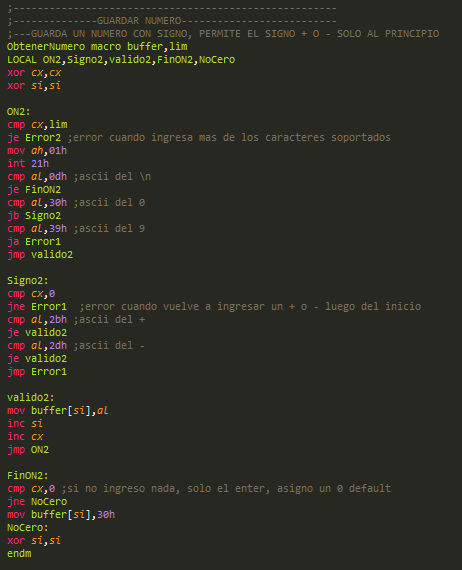
Se declararon todos los mensajes que podían ser mostrados en consola con variables que los representaban de tipo byte:

Cada uno de los cuales es llamado en diferente ocasión y con diferente propósito.



**OBTENER UN NUMERO LEIDO DESDE CONSOLA:**

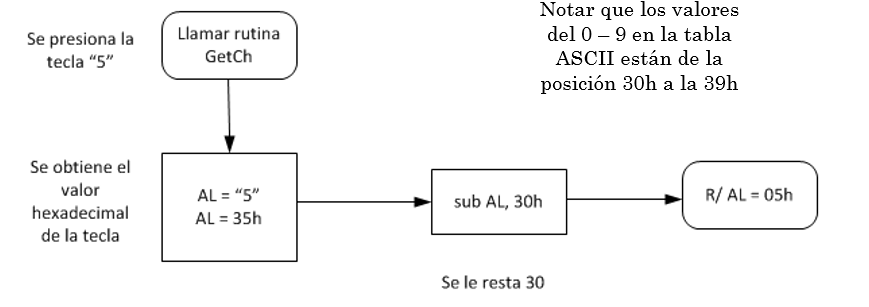
El siguiente algoritmo pide que se ingrese un numero el cual es guardado en un arreglo de tipo byte, el cual posteriormente será mandado al algoritmo de conversión a decimal. Este algoritmo permite el ingreso de signo solo al principio, se podrán ingresar números con el signo + o – solo al comienzo de lo contrario mostrara una advertencia y volverá a solicitar el número, si no se ingresa nada y se ingresa un salto de línea únicamente, este guardara un 0 por default, si se ingresan caracteres incorrectos también se mostrará una advertencia.



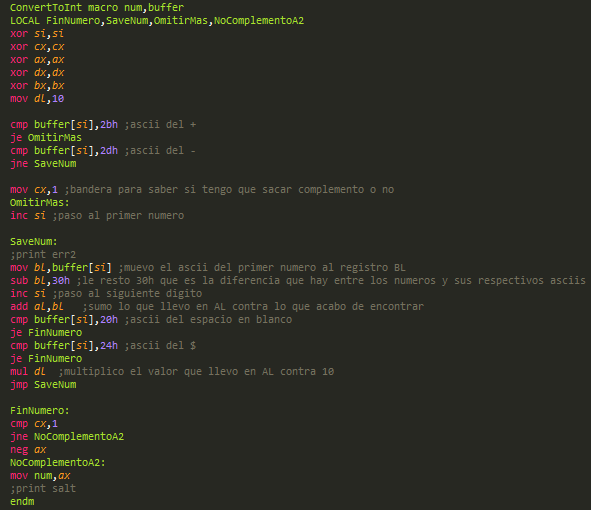
**CONVERTIR UN NUMERO A DECIMAL:**

Al obtener un numero desde consola se pide sea ingresado carácter por carácter, en el siguiente algoritmo se le resta un 30h que es la diferencia entre los números y sus códigos asciis para poder tenerlo en decimal, este número se guarda en una variable de tipo Word. A continuación, se muestra el diagrama de flujo de la conversión y el algoritmo ya implementado en el código:

**DIAGRAMA DE FLUJO DE LA CONVERSION:**



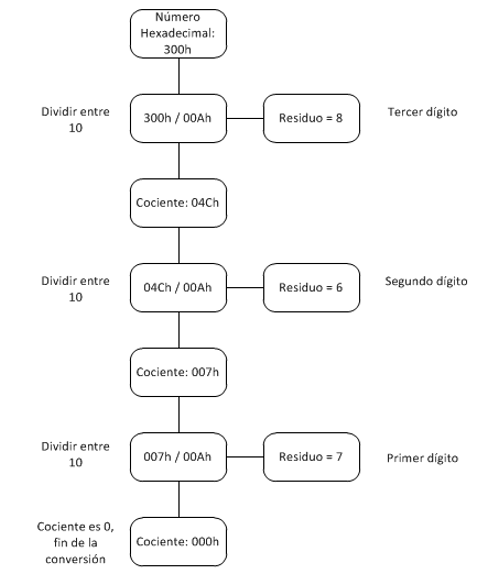
**ALGORITMO IMPLEMENTADO EN EL CODIGO:**



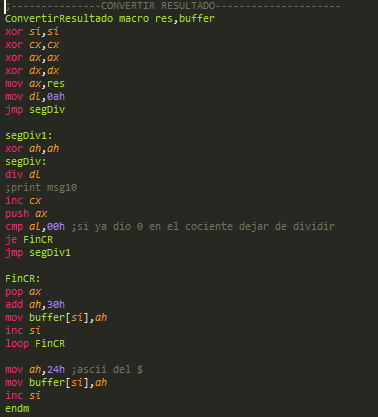
**CONVERTIR UN DECIMAL A SUS CODIGOS ASCIIS:**

Por el lado contrario, cuando realizamos operaciones con números decimales y necesitamos mostrarlos en pantalla hay que hacer un procedimiento inverso para pasar cada digito del número a su respectivo código ascii y asi poder mostrarlo en pantalla. Para poder mostrar el resultado en pantalla se utiliza el algoritmo de convertir un numero entero a sus respectivos asciis, les sumo 30h que es la diferencia que hay entre los números y sus códigos asciis.

**DIAGRAMA DE FLUJO DE LA CONVERSION:**

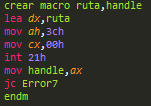
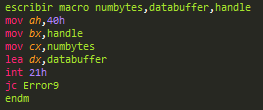


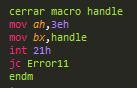
**ALGORITMO IMPLEMENTADO:**



**CREACION DEL REPORTE:**

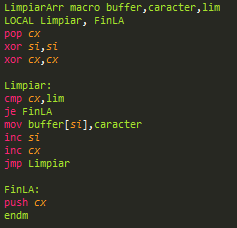
Para la creación del reporte se utilizaron los macros para crear archivos y modificar su contenido, para ello se utilizaron los siguientes macros:



**LIMPIAR ARREGLO:**

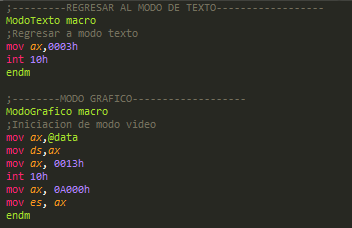
Para que no se encuentren errores a la hora de reutilizar un arreglo con la basura digital, se utiliza el siguiente macro que limpia un arreglo de tipo byte con un carácter específico y la cantidad de espacios específicas.



**REQUERIMIENTOS MINIMOS:**

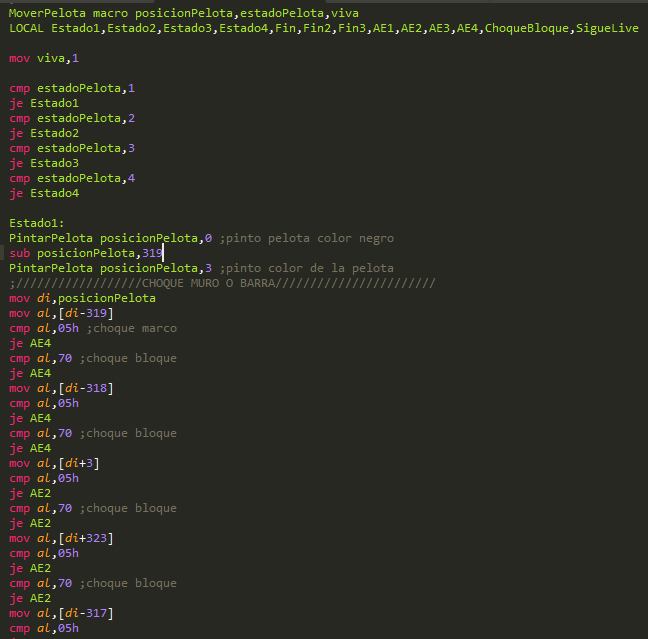
**MACROS PARA EL MODO DE VIDEO:**

En los siguientes macros se obtiene del modo de video su propiedad y la matriz que proporciona de 320 x 200 para el manejo de las imágenes.



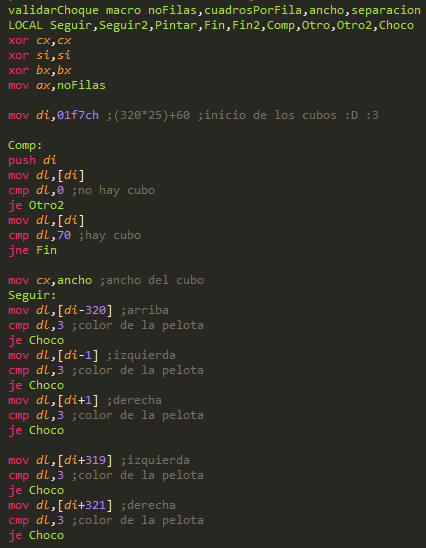
**MOVIMIENTO DE LA PELOTA:**

Para mover la pelota se utilizaron 4 estados para ella, los cuales son movimientos diagonales, para arriba izquierda, arriba derecha, abajo izquierda y abajo derecha. Estos movimientos siempre son de 45 grados por lo cual se aumentaba o disminuía las variables “x” y “y” de la pelota. El algoritmo implementado se muestra a continuación.



**VALIDAR CHOQUES:**

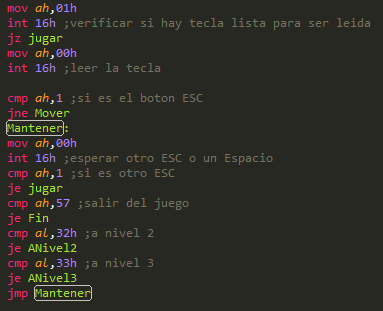
Para validar si hubo choque con el marco o algún cubo se utiliza el siguiente macro:



En el macro anterior se compara cada uno de los pixeles de cada cubo, por ejemplo sus pixeles extremos superiores e inferiores y los laterales, uno por uno y se compara la siguiente posición, se verifica el color que hay en la siguiente posición para saber si la pelota está chocando con alguno de ellos o no, este proceso se repite para cada uno de los cubos hasta que se termina de verificar con todos o cuando se encuentre con cuál de ellos fue que choco, luego de encontrar con cual chocó se procede a eliminarlo y aumentar la variable que lleva control de cuantos cubos fueron eliminados.

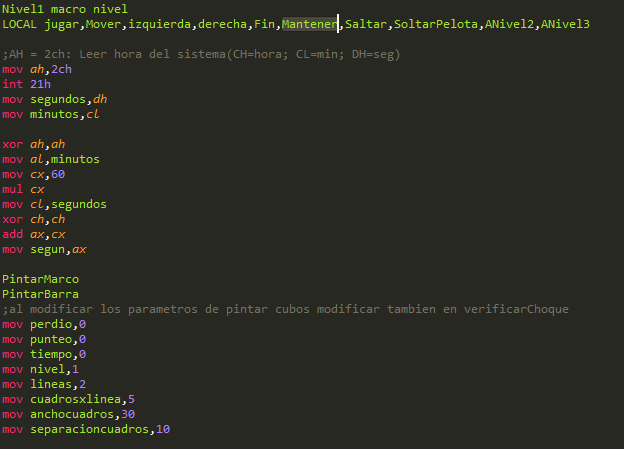
**PAUSAR JUEGO:**

Para pausar el juego se utilizó la interrupción 16h con la cual primero leo del teclado si hay disponible una tecla para leer, luego si la hay, la leo, posteriormente comparo el código de la tecla especial oprimida, si no es otro ESC entonces mantengo leyendo la tecla hasta que se presione otro ESC o la Barra espaciadora para finalizar el juego.



**NIVEL 1:**

El nivel 1 del juego está compuesto de 10 bloques y de una pelota, el objetivo de este es destruir los 10 bloques sin que la pelota toque el borde inferior de la pantalla, al lograr romper los 10 bloques nos mandara al nivel 2 y así sucesivamente.



**INTERRUPCIONES USADAS EN LA PRACTICA:**

**Interrupción al sistema 21H:**

La mayoría de servicios ó funciones del sistema operativo MS-DOS se obtienen a través de la interrupción software 21H. Es por esto que se le denomina DOS-API: DOS-APPLICATION-PROGRAM-INTERFACE La INT 21H está compuesta por un grupo de funciones. Cuando se accede a la INT 21H, hay que indicar el número de función que queremos ejecutar. La llamada a la INT 21H se realizará como sigue:

- Introducimos en (AH) el número de función a la que deseamos acceder.

- En caso de que deseemos acceder a una sub-función dentro de una función, debemos indicarlo introduciendo en (AL) el número de esa sub-función.

- Llamar a la INT 21H.

**Interrupción al sistema 10H (FUNCION 13H):**

El modo gráfico 13h nos permite manejar la pantalla como una matriz de 320 píxeles de ancho por 200 píxeles de alto. Cada píxel puede tomar uno de 256 colores, estos colores están definidos en una paleta de colores la cual podemos configurar.

Usamos la interrupción 10h, servicio 0Ch para modificar los píxeles; el parámetro AL más que especificar directamente el color, indica la entrada de la paleta de colores que se debe usar para el píxel ubicado en la fila DX y la columna CX.