**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**

**PRACTICA 1 ASSEMBLER (MASM)**

**NOMBRE: OSCAR RENE CUELLAR MANCILLA**

**CARNET: 201503712**

**FECHA: 8 DE SEPTIEMBRE 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.

**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:

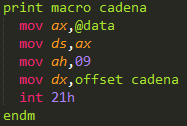
- Introducimos en (AH) el número de función a la que deseamos acceder.

- 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.

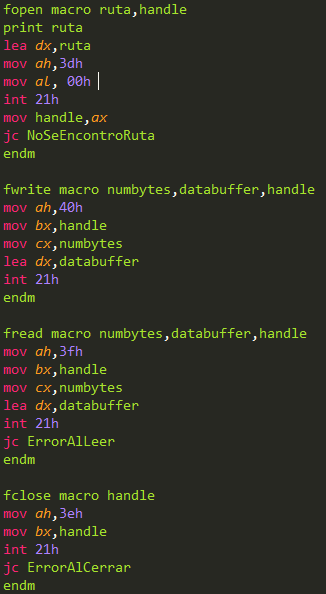
- Llamar a la INT 21H.

**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



También se crearon los macros para el manejo de los archivos, con nombres específicos que le dan un apego al lenguaje C.

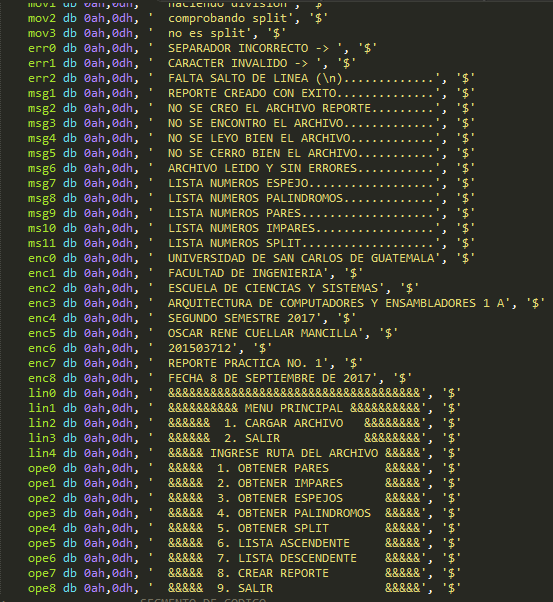


En estos macros se maneja un handle, que es un número que representa de manera única a un archivo para poder ser manejado de una manera más accesible. Para entrar más a detalle acerca de los valores que reciben las interrupciones para poder realizar el manejo de archivos consultar la siguiente página donde se documenta que valores devuelve y pide cada función dada en el registro alto de AX.

Pagina:

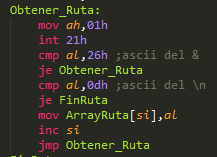
<http://ict.udlap.mx/people/oleg/docencia/ASSEMBLER/asm_interrup_21.html>

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



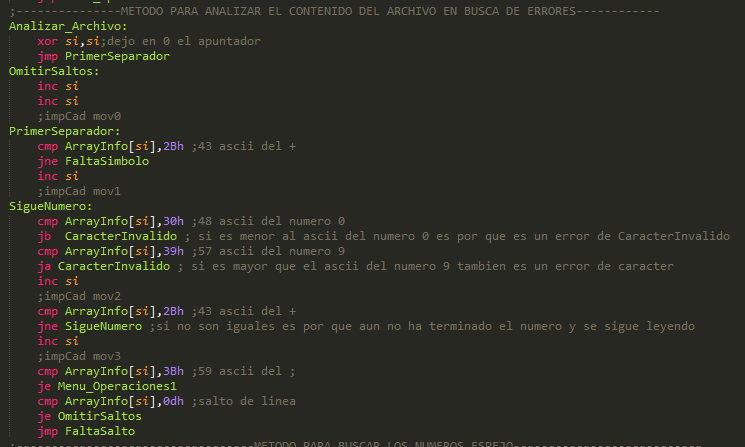
Cada uno de los cuales es llamado en diferente ocasión y con diferente propósito.

Para obtener la ruta que ingresa el usuario desde la consola se utiliza la siguiente función:



En ella leo carácter por carácter que se va ingresando, primero comparo si el carácter es igual a (&), si es igual lo omito y sigo leyendo los demás caracteres, luego si el carácter es un salto de línea es porque el usuario oprimió enter para finalizar el ingreso de la ruta entonces llamo al método FinRuta para salir del ciclo, si aún no se ha presionado enter, entonces guardo el carácter en un arreglo que me representa a mis caracteres de la ruta y aumento el registro de desplazamiento (si) en uno.

Búsqueda de errores en el archivo: para ella realizo el siguiente algoritmo:



El algoritmo anterior pretende leer carácter por carácter el arreglo donde tengo la información del archivo de entrada guardada y realiza un pequeño scanner en el cual espera primero un (+), seguido de los números que sean necesarios y que finalice nuevamente con un (+) si luego del (+) se lee un (\n) es porque sigue leyendo más números y si lo que encuentra es un (;) es porque la lectura llego a su fin y no hubo ningún error, si el análisis es pausado en algún momento se manda a llamar a las etiquetas que muestran el error y mostrando que fue lo que lo produjeron.

Se utilizo una plantilla que tiene como base el analizador para realizar todas las búsquedas que se solicitaban en la práctica como lo son: Números espejo, palíndromos, Split etc… Con ella se sabe en que momento del análisis se encuentra un número para poder realizar posteriores acciones con él.

