**INSTITUTO TECNOLOGICO DE SALTILLO**

**DEPARTAMENTO DE SISTEMAS Y COMPUTACION**

**INGENIERIA EN SISTEMAS COMPUTACIONALES**

**LENGUAJES DE INTERFAZ**

**TRABAJO DE LA MATERIA**

**ALUMNO: CARLOS ALBERTO VANEGAS HERNANDEZ**

**Unidad 1: Introducción al lenguaje ensamblador**

**Tema 1.1. Importancia de la programación en lenguaje ensamblador:**

Primero que nada hablaremos de la historia del lenguaje ensamblador.

En los albores de la historia de la informática y las computadoras, los programadores tenían que escribir su código en lenguaje máquina. Cuando hablamos de lenguaje máquina, nos referimos al código binario, en el cual solo se permiten dos valores 1 y 0 (encendido, apagado – abierto, cerrado – habilitado inhabilitado, etcétera). Al ver que era muy complicado programar de esta manera (teniendo que traducir las instrucciones en lenguaje común a instrucciones en código binario o código máquina) hacia menos eficiente el proceso de codificación, implimentaron un programa (programado en lenguaje máquina) para que la misma máquina convirtiera las instrucciones de entrada en lenguaje usuario, a lenguaje máquina, así, en lugar de escribir las instrucciones en código binario, las escribían en lenguaje común.

La importancia del lenguaje ensamblador radica principalmente que se trabaja directamente con el microprocesador; por lo cual se debe de conocer el funcionamiento interno de este, tiene la ventaja de que en el se puede realizar cualquier tipo de programas que en los lenguajes de alto nivel no lo pueden realizar. Otro punto sería que los programas en ensamblador ocupan menos espacio en memoria.

A continuación mencionaremos las ventajas y desventajas que tiene la programación en lenguaje ensamblador en comparación con otros lenguajes

**Ventajas:**

1.- Como trabaja directamente con el procesador, es procesado más fácilmente.

2.- Al contrario de los lenguajes de alto nivel, los programas en lenguaje ensamblador, ocupan menos espacio, pues no tienen que cargar librerías externas.

3.- El lenguaje ensamblador es flexible, pues se pueden hacer tareas específicas que en un lenguaje de alto nivel no se pueden llevar acabo porque tienen ciertas limitantes que no se lo permite

**Desventajas**

1.- Tiempo de programación: Siendo un lenguaje de bajo nivel, requiere más instrucciones para realiar el mismo proceso, en comparación del lenguaje de alto nivel.

2.- Programas fuente grandes: Por lo mismo que aunenta el tiempo, también crecen los programas fuente.

3.- Peligro de afectar recursos inesperadamente: Con cualquier error que podamos cometer podríamos afectar los recursos de la máquina.

4.- Falta de portabilidad: Para cada máquina existe un lenguaje ensamblador, lo que dificulta codificar en una máquina y llevar los programas a otros sistemas operativos.

**Tema 1.2 El procesador y sus registros internos.**

**1.2.1 Introducción:**

El procesador, conocido como CPU o micro, es el cerebro del PC. Entre sus funciones principales están, la ejecución de las aplicaciones y la coordinación de los diferentes dispositivos que componen un equipo.

Fisicamente, no es más que una pastilla de silicio, que se encuentra colocada sobre la placa base en un conector que se denomina socket, aunque en las laptops lo normal es encontrarlo soldado. La placa base se convierte en la encargada de permitir la conexión con los dispositivos restantes, (memoria RAM, tarjeta gráfica o el disco duro) usando un conjunto de circuitos y chips denominado chipse El encapsulado define como el micro se conecta a la placa base Existen tres modelos, PGA, LGA, BGA. El último caso esta soldado a la placa y por lo tanto será imposible cambiarlo o actualizarlo.

El procesador es de los elementos del PC que más ha evolucionado a lo largo del tiempo. Gracias a las mejoras en la tecnología de fabricación se ha reducido el tamaño de los transistores que se encuentran en su interior, permitiendo integrar un mayor numero de ellos.

Gracias a estas mejoras, se pueden incluir mas bloques funcionales en su interior. Se inició con el controlador de memoria, después con la tarjeta gráfica, y en un futuro cercano, pasaremos del concepto de procesador a lo que se denomina SOC, es decir, un chip con todos los elementos de la placa base en su interior.

Aun cuando no todos los procesadores son iguales, la mayoría incluye:

**Núcleos:** Son procesadores en miniatura. Al tener varios dentro del mismo procesador, podrá trabajar con más de una aplicación al mismo tiempo.

**Caché:** Parte del sistema de memoria, la memoria chache, es el mas cercano al micro; se usa para mejorar la velocidad de los accesos a la memoria RAM. Si un dato se encuentra en la cache no tendrá que buscarse en toda la RAM, y el CPU sera mas rápido.

La cache se encuentra organiada en varios niveles, cada uno más lento y grande que el anterior. Será tarea del CPU dejar los datos mas usados más cerca del micro. La memoria RAM almacena tanto los datos como las instrucciones de los programas y utilidades.

**Controlador de memoria.** Al integrar este controlador en el interior del procesador se consigue aumentar la velocidad de la memoria RAM. El inconveniente es que sólo se puede trabajar con el tipo de memoria para la que el procesador de la computadora esté preparado. Anteriormente la memoria dependía de la placa base y no era raro que esta estuviera preparada para poder funcionar con varios tipos de memoria RAM.

**Tarjeta gráfica:** Integrando este componente ya no se habla de CPUs, sino de APUs. La inclusión de este elemento es fundamental ya que las tarjetas gráficas están compuestas de pequeños núcleos en su interior que pueden usarse para acelerar cierto tipo de aplicaciones. Al pasar al interior del micro pueden hacer su trabajp de una forma mucho mas eficiente.

**Otros elementos.** Los micros han incorporado aun mas funcionalidad que antes se encontraba sobre la placa base. Por ejemplo, el controlador de PCI Express, aumentando la velocidad con la que el micro es capa de comunicarse, por ejemplo, con una tarjeta gráfica discreta.

**Tema 1.2.2 Registros del procesador**

Los registros del procesador se emplean para controlar instrucciones en ejecución, manejar direccionamiento de memoria y proporcionar capacidad aritmética. Los registros son direccionables por medio de un nombre. Los bits por convención, se numeran de derecha a izquierda, como en:

... 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Los registros internos del procesador se pueden clasificar en 6 tipos diferentes:

1. Registros de segmento
2. Registros de propósito general
3. Registros de apuntadores
4. Registros de banderas
5. Registros de Puntero de instrucción
6. Registros de Pila

**Registros de segmento:** Un registro de segmento tiene 16 bits de longitud y facilita un área de memoria para direccionamiento conocida como el segmento actual.

**Registro CS.** El DOS almacena la dirección inicial del segmento de código de un programa en el registro CS. Esta dirección de segmento, mas un valor de desplazamiento en el registro apuntador de instrucción (IP), indica la dirección de una instrucción que es buscada para su ejecución.

**Registro DS.** La dirección inicial de un segmento de datos de programa es almacenada en el registro DS. En términos sencillos, esta dirección, mas un valor de desplazamiento en una instrucción, genera una referencia a la localidad de un byte especifico en el segmento de datos.

**Registro SS.** El registro SS permite la colocación en memoria de una pila, para almacenamiento temporal de direcciones y datos. El DOS almacena la dirección de inicio del segmento de pila de un programa en el registro SS. Esta dirección de segmento, más un valor de desplazamiento en el registro del apuntador de pila (SP), indica la palabra actual en la pila que está siendo direccionada.

**Registros ES.** Algunas operaciones con cadenas de caracteres (datos de caracteres) utilizan el registro extra de segmento para manejar el direccionamiento de memoria. En este contexto, el registro ES está asociado con el registro DI (índice). Un programa que requiere el uso del registro ES puede inicializarlo con una dirección de segmento apropiada.

**Registros FS y GS.** Son registros extra de segmento en los procesadores 80386 y posteriores.

**Tema 1.3 La memoria principal (RAM)**

La memoria principal está formada por un conjunto de unidades llamadas palabras. Dentro de cada palabra se guarda la información que constituye una instrucción o un dato o parte de un dato.

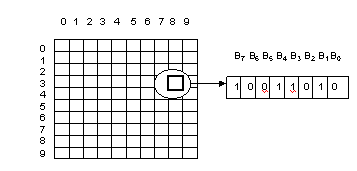
A la cantidad de palabras que forman la memoria principal se denomina capacidad de memoria. Cuanto mayor sea el numero de palabras, mayor será el numero de instrucciones o datos que podrá almacenar la computadora.

Una palabra esta formada a su vez por unidades mas elementales llamadas bits; al igual que en el español, donde cada palabra se forma por letras. Cada bit puede tener solo dos valores, el 1 o el 0.

El numero de bits que forman una palabra se llama longitud de palabra.

En la figura 2.5 se muestra como se puede estar organizada una Memoria Principal.

**Figura 2.5.** Organización de una unidad de memoria.



Las palabras forman una matriz de 10 filas y 10 columnas. La primera palabra corresponderá con la dirección 00, la segunda con la 01, y la última, con la 99. La capacidad de la memoria será de 10 \* 10 = 100 palabras. Tembièn se muestra la longitud de la palabra 38, que es de 8 bits, al igual que las demás, y la información que contiene, que es el valor binario 10011010.

Las palabras se distinguen entre si por la posición que ocupan en la memoria principal (MP), y se puede guardar la información y volverá a recuperar indicando el numero de dicha posición. A los números que señalan las posiciones de memoria se les denomina direcciones de memoria.

La acción de guardar información en una palabra se llama acceso de escritura, y a la acción de recuperar esa información se le llama acceso de lectura. Los accesos son coordinados por la UC. La secuencia de órdenes que debe generar la UC se indica en la tabla 2.1.

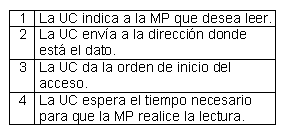
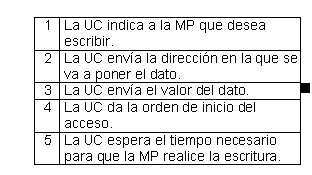
En la tabla 2.1 a se muestra un acceso de escritura. Obsérvese que la UC debe indicar además de la posición del dato, el valor del dato y las indicaciones de control que le digan a la memoria que se desea guardar el dato y el momento en que debe iniciarse la operación de escritura.

Esta ultima orden la dará la UC cuando esté segura de que los datos anteriores han llegado correctamente a la MP. Despues de esta ultima orden, la UC espera un tiempo para asegurarse que se ha escrito la información en la MP.

En la parte B de la misma tabla se muestra un acceso de lectura. En este caso, la UC no indica el dato, puesto que es precisamente lo que espera recibir. Los demás pasos son idénticos a los del acceso de escritura.

Desde que se inicia la secuencia hasta que finaliza transcurre un tiempo, denominado tiempo de acceso, cuya duración depende de la tecnología con que esta fabricada la MP.

**Tabla 2.1.** Secuencia de acceso a la memoria.



                       (a) Acceso de escritura.                    (b) Acceso de lectura

**Tema 1.4 El concepto de interrupciones**

Esta sección estudia el concepto base de la programación de PCs: el uso de las interrupciones. Básicamente, las interrupciones son las puertas de acceso a las funciones del sistema operativo, el cual controla los recursos del sistema.

El término **interrupción**  evolucionó como una descripción de lo que pasa dentro del microprocesador. Una interrupción detiene la ejecución de un programa, para invocar otra rutina. La rutina invocada por efecto de la interrupción es llamada rutina manejadora de interrupción (*interrupt handler*).

La rutina manejadora de interrupción es un programa que puede usar cualquiera de las interrupciones disponibles en el microprocesador. El final de la rutina es marcado por la instrucción **iret** (interrupt return). Tal y como lo señala el nombre de la instrucción, en este punto el control es retornado al programa original. Este mecanismo es similar a la invocación de una subrutina. La diferencia radica en que la interrupción es generada por un ente externo aal microprocesador y no por el propio programa.

Un programa puede ser interrumpido en cualquier momento, pero debe ser reactivado sin modificaciones que afecten su funcionamiento. Por ello hay que restaurar los registros del microprocesador. Sin embargo, es muy difícil que una instrucción no altere al menos uno de los registros, por lo que la rutina manejador de interrupción deberá encargarse del proceso de salvar y restaurar el contenido de los mismos.

**Tema 1.4.1 Tipos de Interrupciones**

Las interrupciones se dividen en dos tipos: hardware y software.

Interrupciones Por Software

Una interrupción por software es una interrupción acivada por una instrucción especial: la instrucción Int. Esta es usada siempre en conjunto con el numero de la interrupción a ser invocada. Esto permite que un programa invoque una rutina de la cual no conoce la ubicación exacta en la memoria. Las rutinas pertenecientes al BIOS y al núcleo del DOS (*kernel*), pueden ser accesadas por este mecanismo.

El propósito mas importante de dichas rutinas, es hacer que las diferencias en el hardware entre una computadora y otra sea transparente a los programas de aplicación. Las rutinas del *BIOS* y del *DOS* actúan como intermediarios entre las aplicaciones y el hardware del equipo. Esto garantiza cierta independencia de los programas con respecto a la diversidad de configuraciones de hardware disponibles.

Las rutinas manejadoras de interrupciones por software generalmente requieren de parámetros que le proprociones instrucciones especificas sobre el trabajo a realizar. Estos son normalmente transferidos por medio de los registros internos del microprocesador.

Interrupciones por hardware

Las interrupciones por hardware son procesos activados por los distintos componentes del sistema. Con ello el dispositivo solicita la atención del microprocesador. La direccion de la rutina correspondiente es obtenida de la tabla de vectores de interrupción. Estos valores son inicializados por el *BIOS.*

Entre las interrupciones por hardware existen dos tipos: las enmascarables y las no enmascarables. Las interrupciones enmascarables son aquellas que pueden ser suprimidas por medio de la instrucción **cli (clear interrupt flag).** Esto significa que la rutina correspondiente no sera invocada al presentarse la interrupción. Las interrupciones no enmascarables no pueden ser suprimidas y son empleadas para servir eventos críticos del sistema.

**Tema 1.4.2 Instrucciones.**

Las instrucciones del procesador 80x86 pueden ser agrupadas en 7 renglones según sus funciones:

Instrucciones de Transferencia de Datos.

Son aquellas que llevan a cabo funciones de transferencia de informacion y puede ser divididas en dos categorías generales: instrucciones que mueven datos de un registro a otro registro, o entre localidades de memoria y registros , e instrucciones que mueven datos hacia o desde la pila. Ejemplos de estas instrucciones son: mov, xchg, push, pop, etc.

Instrucciones Aritméticas.

Llevan a cabo operaciones aritméticas y de comparación. Se dividen en cinco tipos: Instrucciones de adición, substraccion, multiplicación, división y comparación. Entre las instrucciones pertenecientes a este grupo tenemos: add, sub, mut, div, cmp, etc.

Instrucciones Logicas

El 8086 provee instrucciones para llevar a cabo las 4 funciones lógicas básicas. Estas son AND, NOT, OR y XOR. Adicionalmente incluye la función TEST, la cual ejecuta una operación AND sin alterar los operandos.

Instrucciones para Manejo de Cadena.

Las instrucciones de manejo de cadenas llevan a cabo funciones ejecutadas normalmente por un conjunto de instrucciones dentro de un lazo. Cada una de estas instrucciones actualiza los registros apuntadores involucrados en la misma. En cada iteración los registros apuntadores serán incrementados si el valor del bit de direccion del registro de estado esta en cero y decrementados en el caso contrario. Los regustros apuntadores afectados serán modificados por uno o por dos ya sea que la operación involucre bytes o palabras. Existen cinco instrucciones para manejo de cadenas:

**MOVS:** Mueve 8 o 16 bits de datos desde una localidad de memoria a otra.

**LODS:** Carga un dato de 8 o 16 bits desde una localidad de memoria al registro AL o AX.

**STOS:** Almacena el contenido de AL (operación de 8 bits) o de AX (operación de 16 bits) en una localidad de memoria.

**SCAS:** Compara el contenido de AL o AX con el de una localidad de memoria.

**CMPS:** Compara el contenido de una localidad de memoria con el de otra.