# INSTITUTO TECNOLÓGICO DE NUEVO LEÓN

Ingeniero en sistemas computacionales

Lenguajes Autómatas II

Proyecto #4

Catedrático

Juan Pablo Rosas Baldazo

**Alumno**

Carlos Alberto Martínez Mendoza

**Fecha**

4 de Mayo del 2018

## Introducción

La generación de código es la fase más compleja de un compilador, puesto que no sólo depende de las características del lenguaje fuente sino también de contar con información detallada acerca de la arquitectura objetivo, la estructura del ambiente de ejecución y el sistema operativo que esté corriendo en la máquina objetivo.

La generación de código por lo regular implica también algún intento por optimizar, o mejorar, la velocidad y/o el tamaño del código objetivo recolectando más información acerca del programa fuente y adecuando el código generado para sacar ventaja de las características especiales de la máquina objetivo, tales como registros, modos de direccionamiento, distribución y memoria caché.

En el modelo de análisis y síntesis de un compilador, la etapa inicial traduce un programa fuente a una representación intermedia a partir de la cual la etapa final genera el código objeto.

## Capítulo 1: Registros

Los registros son la memoria principal de la computadora. Existen diversos registros de propósito general y otros de uso exclusivo. Algunos registros de propósito general son utilizados para cierto tipo de funciones. Existen registros acumuladores, puntero de instrucción, de pila, etc.

Los registros son espacios físicos dentro del microprocesador con capacidad de 4 bits hasta 64 bits dependiendo del microprocesador que se emplee.

Antes de nada, para el desarrollo de esta parte hablaremos indistintamente de registros de activación o de marcos de pila. Esto se debe a que en la documentación encontrada sobre el manejo de los registros ebp y esp se hace mención a dicho concepto de marco de pila. Puesto que el lenguaje permite recursividad, los registros de activación se asignan dinámica mente.

La UCP o CPU tiene 14 registros internos, cada uno de ellos de 16 bits (una palabra). Los bits están enumerados de derecha a izquierda, de tal modo que el bit menos significativo es el bit 0.

## Sección 1.1: Registros de datos

AX: Registro acumulador. Es el principal empleado en las operaciones aritméticas.

BX: Registro base. Se usa para indicar un desplazamiento.

CX: Registro contador. Se usa como contador en los bucles.

DX: Registro de datos.

Estos registros son de uso general y también pueden ser utilizados como registros de 8 bits, para utilizarlos como tales es necesario referirse a ellos como por ejemplo: AH y AL, que son los bytes alto (high) y bajo (low) del registro AX. Esta nomenclatura es aplicable también a los registros BX, CX y DX.

## Sección 1.2: Registros de segmentos

CS: Registro de segmento de código. Contiene la dirección de las instrucciones del programa.

DS: Registro segmento de datos. Contiene la dirección del área de memoria donde se encuentran los datos del programa.

SS: Registro segmento de pila. Contiene la dirección del segmento de pila. La pila es un espacio de memoria temporal que se usa para almacenar valores de 16 bits (palabras).

ES: Registro segmento extra. Contiene la dirección del segmento extra. Se trata de un segmento de datos adicional que se utiliza para superar la limitación de los 64Kb del segmento de datos y para hacer transferencias de datos entre segmentos.

## Sección 1.3: Registros punteros de pila

SP: Puntero de la pila. Contiene la dirección relativa al segmento de la pila.

BP: Puntero base. Se utiliza para fijar el puntero de pila y así poder acceder a los elementos de la pila.

## Sección 1.4: Registros índices

SI: Índice fuente.

DI: Índice destino.

# Capítulo 2: Lenguaje Ensamblador

El ensamblador (del inglés assembler) es un traductor de un código de bajo nivel a un código, ejecutable directamente por la máquina para la que se ha generado.

Fue la primera abstracción de un lenguaje de programación, posteriormente aparecieron los compiladores.

## Sección 2.1: Características

El programa lee un archivo escrito en lenguaje ensamblador y sustituye cada uno de los códigos mnemotécnicos por su equivalente código máquina. Los programas se hacen fácilmente portables de máquina a máquina y el cálculo de bifurcaciones se hace de manera fácil.

- El programa lee un archivo escrito en lenguaje ensamblador y sustituye cada uno de los códigos mnemotécnicos por su equivalente código máquina.
- Los programas se hacen fácilmente portables de máquina a máquina y el cálculo de bifurcaciones se hace de manera fácil.

## Sección 2.2: Segunda generación de lenguajes

Versión simbólica de los lenguajes máquina (Urbina, 2011) (MOV, ADD).La comunicación en lenguaje de máquina es particular de cada procesador que se usa, y programar en este lenguaje es muy difícil y tedioso, por lo que se empezó a buscar mejores medios de comunicación con ésta. Los lenguajes ensambladores tienen ventajas sobre los lenguajes de máquina.

Este lenguaje fue usado ampliamente en el pasado para el desarrollo de software, pero actualmente sólo se utiliza encontradas ocasiones, especialmente cuando se requiere la manipulación directa del hardware o se pretenden rendimientos inusuales de los equipos

## Sección 2.3: Clasificación

- **Ensambladores básicos:** Son de muy bajo nivel, y su tarea consiste básicamente, en ofrecer nombres simbólicos a las distintas instrucciones, parámetros y cosas tales como los modos de direccionamiento
- **Ensambladores modulares, o macro ensambladores:** Descendientes de los ensambladores básicos, fueron muy populares en las décadas de los 50 y los 60, fueron antes de la generalización de los lenguajes de alto nivel. Un macroinstrucción es el equivalente a una función en un lenguaje de alto nivel.

## Sección 2.4: Almacenamiento

Una de las principales ventajas del uso del ensamblador, es que se encarga de administrar de manera transparente para el usuario la creación de memoria, las

bifurcaciones y el paso de parámetros. Además nos permite acceder directamente a los recursos de la máquina para un mejor desempeño.

## Sección 2.5: Operaciones básicas

Las operaciones básicas en un lenguaje ensamblador son la suma la resta la multiplicación y la división y Necesitara un poco más de información sobre la arquitectura y SO para el cual programas.

Pero la idea básica es:

- Definir que parámetros tendrá la función.
- Hacer el programa, propiamente dicho, en assembler.

Siguiendo la convención de pasaje de parámetros, manejará registros y posiciones de memoria, devolviendo los resultados en donde deba (una posición de memoria, el registro eax, etc.).

# Capítulo 3: Lenguaje maquina

Es el que proporciona poca o ninguna abstracción del microprocesador de un ordenador. El lenguaje máquina solo es entendible por las computadoras. Se basa en una lógica binaria de 0 y 1, generalmente implementada por mecanismos eléctricos. En general el lenguaje maquina es difícil de entender para los humanos por este motivo hacemos uso de lenguajes más parecidos a los lenguajes naturales.

Se denomina lenguaje máquina a la serie de datos que la parte física de la computadora o hardware, es capaz de interpretar. El lenguaje máquina fue el primero que empleo el hombre para la programación de las primeras computadoras. Una instrucción en lenguaje máquina puede representarse de la siguiente forma: 01101100101001001111010110.

Esta secuencia es fácilmente ejecutada por la computadora, pero es de difícil interpretación, siendo aún más difícil la interpretación de un programa (conjunto de instrucciones) escrito de esta forma.

Esta dificultad hace que los errores sean frecuentes y la corrección de los mismos costosa, cuando no imposible, al igual que la verificación y modificación de los programas.

## Sección 3.1: Características

- El lenguaje máquina realiza un conjunto de operaciones predeterminadas llamadas micro operaciones.

- Las micro operaciones sólo realizan operaciones del tipo aritmética (+,-,*, /), lógicas (AND, OR, NOT) y de control (secuencial, decisión, repetitiva).

- El lenguaje máquina es dependiente del tipo de arquitectura. Así un programa máquina para una arquitectura Intel x86 no se ejecutará en una arquitectura Power PC de IBM (al menos de manera nativa).

- Algunos microprocesadores implementan más funcionalidades llamado
- CISC, pero son más lentos que los RISC ya que estos tienen registros más grandes.

## Sección 3.2: Direccionamiento

Es la forma en cómo se accede a la memoria. Recordar que un programa no puede ejecutarse sino se encuentra en memoria principal. La forma de acceder a la memoria depende del microprocesador, pero en general existen dos tipos de direccionamiento: directo e indirecto.

El direccionamiento directo también recibe el nombre de direccionamiento absoluto y el acceso a las direcciones se hace de manera directa. El direccionamiento indirecto también recibe el nombre de direccionamiento relativo y se basa a partir de una dirección genérica, generalmente el inicio del programa.

Para acceder a una dirección relativa se suma a la dirección base el número de espacios de memorias necesarias.

El direccionamiento relativo hace a los programas relocalizables e independientes. Si la dirección base es el inicio de la memoria fija el direccionamiento pasa a ser un variante de direccionamiento absoluto.

## Sección 3.3: Ventajas

- Mayor adaptación al equipo
- Máxima velocidad con mínimo uso de memoria

## Sección 3.4: Desventajas

- Imposibilidad de escribir código independiente de la maquina
- Mayor dificultad en la programación y en la comprensión de los programas
- El programador debe conocer mas de un centenar de instrucciones
- Es necesario conocer en detalle la arquitectura de la maquina

# Capítulo 4: Administración de memoria

La administración de la memoria es un proceso hoy en día muy importante, de tal modo que su mal o buen uso tiene una acción directa sobre el desempeño de memoria. En general un ensamblador tiene un administrador de memoria más limitado que un compilador; en la mayoría de los lenguajes de programación el uso de punteros no estaba vigilado por lo que se tienen muchos problemas con el uso de memoria. Los lenguajes más recientes controlan el uso de punteros y tienen un programa denominado recolector de basura que se encarga de limpiar la memoria no utilizada mejorando el desempeño.

La memoria principal puede ser considerada como un arreglo lineal de localidades de almacenamiento de un byte de tamaño. Cada localidad de almacenamiento tiene asignada una dirección que la identifica

Se distinguen los siguientes propósitos del sistema de administración de memoria:

## Sección 4.1: Protección

Si varios programas comparten la memoria principal, se debería asegurar q ue el programa no sea capaz de cambiar las ubicaciones no pertenecientica él. Aunque una acción de escritura puede tener efectos más graves que una de lectura, esta última tampoco debería estar permitida, para proporcionar algo de privacidad al programa.

## Sección 4.2: Compartimiento

Este objetivo parece contradecir al anterior, sin embargo a veces es necesario para los usuarios poder compartir y actualizar información (por ejemplo, en una base de datos) y, si se organiza la tarea de entrada a la misma, se puede evitar el tener varias copias de la rutina.

## Sección 4.3: Reubicación

La técnica de multiprogramación requiere que varios programas ocupen la memoria al mismo tiempo. Sin embargo no se sabe con anticipación donde será cargado cada programa por lo que no es práctico usar direccionamiento absoluto de memoria.

## Sección 4.4: Organización física

Debido al costo de una memoria principal rápida, éste se usa en conjunto con una memoria secundaria mucho más lenta (y por consiguiente, barata) a fines de extender su capacidad.

## Sección 4.5: Organización lógica

Aunque la mayor parte de las memorias son organizadas linealmente con un direccionamiento secuencial, esto difícilmente concuerde con el camino seguido por el programa, debido al uso de procedimientos, funciones, subrutinas, arreglos, etc.

## Conclusión

Como vimos en este resumen, conocimos como el código intermedio puedes irse optimizando hacia el código objeto de bajo nivel. Y que cada equipo tiene un diferente procesador para su lenguaje máquina, por lo cual se crean los lenguajes ensambladores, que a pesar de ya no ser tan utilizadas en la actualidad, fue de vital importancia en el pasado para el desarrollo de software y cuando se requería la manipulación directa del hardware o en rendimientos inusuales de los equipos.

Y cabe recalcar que algo importante es la administración de la memoria de tu ordenador, ya que todo ensamblador tiene por defecto un administrador de memoria, este es un poco más limitado si es comparado con el de un compilador. En la actualidad los lenguajes que más se utilizan tienen un control de recolector de basura que se encarga de limpiar la memoria no utiliza o de segundo plano

## Conceptos

**Código intermedio:** El Front-end traduce el programa fuente en una representación de código intermedio, y el back-end traduce esta representación en código final

**Código objeto:** Al código que resulta de la compilación del código fuente. Puede ser en lenguaje maquina o bytecode, y puede distribuirse en varios archivos que corresponden a cada código fuente compilado

**Lenguaje de bajo nivel:** Aquel en el que sus instrucciones ejercen un control directo sobre el hardware y están condicionados por la estructura física de las computadoras que lo soportan

**Arquitectura computacional:** Es el diseño conceptual y la estructura operacional fundamental de un sistema de computadoras

**Memoria:** Es el dispositivo que retiene, memoriza o almacena datos informáticos durante algún periodo de tiempo

**Registros acumuladores:** Son almacenados temporalmente los resultados aritméticos y lógicos intermedios que serán tratados por el circuito operacional de la unidad aritmético-lógica

**Microprocesador:** Procesador de muy pequeñas dimensiones en el que todos los elementos están agrupados en un solo circuito integrado

**Bit:** Es un digito del sistema de numeración binario

**Archivo de datos:** Conjunto de bits que son almacenados en un dispositivo

**Archivos de programa:** Es un sistema Microsoft Windows se utiliza como un lugar de almacenamiento para programas y algunos otros archivos binarios

**Código maquina:** Es el único que entiende directamente la computadora, utiliza el alfabeto binario que consta de los dos únicos símbolos 0 y 1

**Operaciones lógicas:** Son expresiones matemáticas cuyo resultado es un valor booleano

**Memoria secundaria:** Es el conjunto de dispositivos y soportes de almacenamiento de datos que conforman el subsistema de memoria de la computadora, junto con la memoria principal

## Bibliografía


Beck. Software de Sistemas, Introducción a la programación de Sistemas. Addison-Wesley Iberoamericana.

Guerra Crespo. Héctor. Compiladores. Ed. Tecnológica didáctica.

Vázquez Gaudioso. Elena, García Saiz. Tomas. Introducción a la teoría de autómatas, gramática y lenguajes. Ed. Universitaria Ramón Areces

Meloni. Brenda, Giro. Juan, Vázquez. Juan, Constable. Leticia. Lenguajes formales y teoría de autómatas. Ed. Alfaomega

Millan Borrajo. Daniel, Fernández Martínez. Paloma, Isasi Viñuela. Pedro. Lenguajes, Gramáticas y Autómatas: Un enfoque práctico. Ed. Addison Wesley


.