RESUMEN CAPITULO 1 - TRADUCTORES/COMPILADORES

LENGUAJES DE PROGRAMACION (Comunicación hombre-maquina)
Las computadoras operan sobre bits (unos y ceros) y las personas nos entendemos por medio de idiomas. El vehículo de comunicación entre una pc y un humano son los lenguajes de programacion.

Lo podemos definir a un lenguaje de programación como un conjunto de símbolos junto a un conjunto de reglas para combinar los símbolos de forma que permita al usuario crear programas que serán entendidos por la computadora para realizar alguna tarea.
Los lenguajes de programación los podemos clasificar en lenguajes de bajo nivel y de alto nivel.

LENGUAJES DE BAJO NIVEL: Son totalmente dependientes de la maquina, por lo tanto no se puede migrar a otras maquinas. Cada instrucción corresponde a una acción ejecutable por la PC

Dentro de estos se encuentra:
- Lenguaje de maquina: Las instrucciones se representan por medio de código binario. Cada cifra del código binario corresponde a un pulso eléctrico. Cada operación requiere una determinada combinación de "señales". Por esto el código binario es el lenguaje que la PC entiende.
- Lenguaje ensamblador: Las instrucciones se presentan por medio de palabras en ves de código binario, es decir las instrucciones se presentan en forma simbólica

LENGUAJE DE ALTO NIVEL: se encuentran mas cercano al lenguaje natural humano que al de maquina. Son independientes de la maquina por lo cual se puede migrar de una a otra.
Cada instrucción corresponde a varias acciones ejecutables por la PC.
Estos lenguajes permiten al programador olvidarse del funcionamiento interno del PC. Estos programas necesitan de un traductor.

TRADUCTOR: Los programas fuente escrito en un lenguaje de alto nivel necesita de un proceso de traducción al lenguaje de maquina. A esto se lo llama proceso de traducción y al programa que realiza el proceso traductor

PROGRAMA FUENTE ———-> **TRADUCTOR** ———-> **PROGRAMA OBJETO**
Programa fuente: programa escrito por el programador
Programa objeto: conjunto de instrucciones que entiende el ordenador

Un lenguaje de programación tiene un léxico (símbolos o vocabulario), una sintaxis (reglas) y una semántica (significado).

Existen 2 tipos de traductores: los compiladores y los interpretes

INTEPRETE: Programa informatico que procesa el código fuente de un programa durante su tiempo de ejecución y actúa como una interfaz entre ese programa y el procesador. Procesa el código fuente linea por linea. El proceso de conversion no finaliza hasta que se ha interpretado todo el código, solo se interrumpe si se produce un fallo durante el procesamiento (error), de esta forma se simplifica la resolución de errores.
Python, Ruby o PHP son algunos de los lenguajes de programación que dependen de un interprete para su traducción a código binario.

COMPILADOR: Es un programa informatico que traduce todo el código fuente antes de su ejecución. Algunos lenguajes compilados son C, C++ o Pascal.

depuestos clave que caracterizan intérpretes y compiladores:

	Intérprete	**Compilador**
Momento en que se traduce el código fuente	Durante el tiempo de ejecución del software	Antes de ejecutar el software
Procedimiento de traducción	Línea por línea	Siempre todo el código
Presentación de errores de código	Después de cada línea	En conjunto, después de toda la compilación
Velocidad de traducción	Alta	Baja
Eficiencia de traducción	Baja	Alta
Lenguajes típicos	Perl, PHP, Python, Ruby	C, C++, Pascal, GO

SOLUCION HIBRIDA DE COMPILADOR E INTERPRETE:
Durante el proceso de compilación tiene lugar un paso intermedio, antes de generar la traducción final en código maquina, se convierte el código fuente en un código intermedio (código objeto) para luego se utilizado por un interprete para ser compatible en diversas plataformas. Un ejemplo es JAVA.

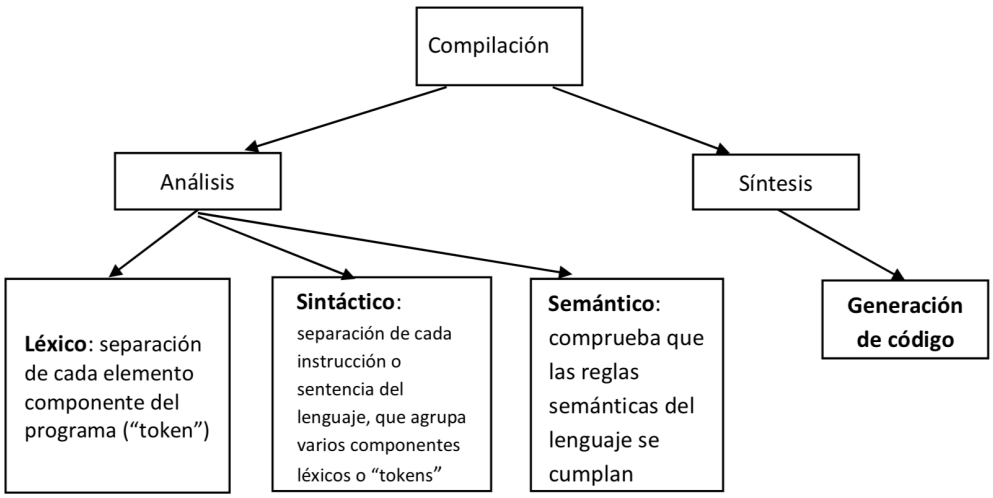
Una vez finalizada la compilación y obtenido el programa objeto, es sometido a otro proceso para llegar al ejecutable. Por lo tanto podemos decir que un compilador no es un programa que funciona de manera aislada, si no que necesita de otros programas, algunos son:

- preprocesador: se ocupa de incluir archivo, eliminar comentarios, y otras tareas similares
- Enlazador: o linker, se encarga de construir el archivo ejecutable
- Depurador o debugger: permite seguir pasar a paso la ejecución de un programa
- Ensamblador: convierte el programa objeto en ejecutable mediante un ensamblador.

CONCEPTOS:
TOKEN: es una categoria a la que pertenece cada cadena o palabra del código fuente. También se llaman categorías léxicas. Los tokens pueden variar de un programa a otro pero en general son palabras reservadas, identificadores, constantes, operadores de asignación, operadores aritméticos, etc.
<categoria lexica o token, valor lexico>
Valor léxico es el dato relaciónado con el token, es decir el valor del token. También se lo llama lexema.

PATRON: es la Telga de generación de caracteres que podría representar a un determinado token. La descripción formal de un patron ser realiza mediante expresiones regulares

ATRIBUTO: caracteristica propia de cada tipo de token



LAS FACES DE UN COMPILADOR:
Tiene dos etapas: una inicial en la que el proceso depende del código fuente y es independiente de PC. Y otra etapa final donde el proceso se hace dependiente de la maquina y indpendiente del lenguaje fuente.

La etapa de análisis o etapa inicial tiene los procesos de Análisis léxico, Análisis sintactico y Análisis semántico.

Todos los errores se informan al final de la etapa de análisis y recién ai se detiene el proceso de compilación.

ANALISIS LEXICO: El AL reconoce para cada cadena del código fuente si concuerda o no con algún token y lo clasifica por su categoría

- **ENTRADA**: cadena de caracteres (código fuente)
- **PROCESO**: se agrupan los caracteres, se separan los token
- **SALIDA**: secuencia de <Tokens, valor> correspondientes

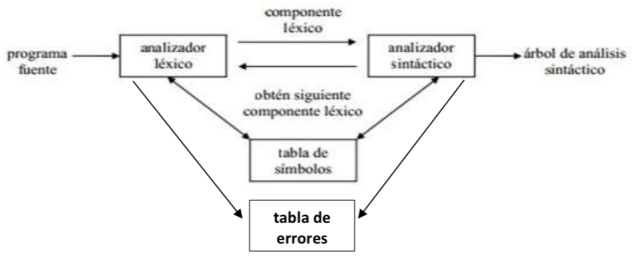
Cuando el AL detecta un identificador o constance almacena los token con sus atributos en la tabla de símbolos.

El aAL opera bajo petición del AS, devolviendo un token cuando el AS lo va necesitando
ERRORES: detecta un error cuando los caracteres que toma el buffer no corresponden a ningún token valido del lenguaje.

ANALISIS SINTACTICO: La escritura del código fuente se rige por un conjunto de reglas gramáticas. Esto se denomina sintaxis del lenguaje y permite determinar matemáticamente si un programa es correcto o no desde la sintaxis.
La tarea del AS es determinar si la estructura sintáctica del del programa es correcta o no.

- **ENTRADA**: tokens producidos por el AL
- **PROCESO**: examina los tokens analizando la categoria lexica del mismo y comprueba que van llegando en el orden especificado por la gramática.
- **SALIDA**: Arbol sintactico

Los dos analizadores (AL y AS) trabajan juntos, de hecho el AL es una subrutina del AS
El accionar de estos analizadores apunta a comprobar si el programa esta bien construido respecto a las reglas del lenguaje.



ADMINISTRACION DE LA TABLA DE SIMBOLOS Y TIPOS

TABLA DE TIPOS: es esencial para que el Ase pueda trabajar. Se inicializan todos los tipos primitivos y se agregan los tipos definidos por el programador.

TABLA DE SIMBOLOS: es una estructura de datos contenedora de l información que el programa procesa. A lo largo de la ejecución de un programa algunos de estos símbolos cambian de valor pero su dirección de memoria es fija.

TABLA DE TIPOS

Código del tipo	nombre	dimensión
0	*char*	*1*
1	*int*	*1*
2	*Ar*	*10*

TABLA DE SIMBOLOS

Código del símb.	nombre	categoria	tipo	dirección
0	*x*	*variable*	*2*	*9000*
1	*car*	*variable*	*0*	*9020*
2	*z*	*variable*	*1*	*9021*

ANALIZADOR SEMANTICO: es la fase mas compleja, comprueba que exista coherencia en el programa. El AS invoca a los procesos del Ase. Para validar el significado de los tokens el Ase recurre a sus atributos.

- **ENTRADA**: árbol jerárquico resultado del AS
- **PROCESO**: comprueba el significado de lo que va leyendo sea valido.

- **SALIDA**: árbol semantico.

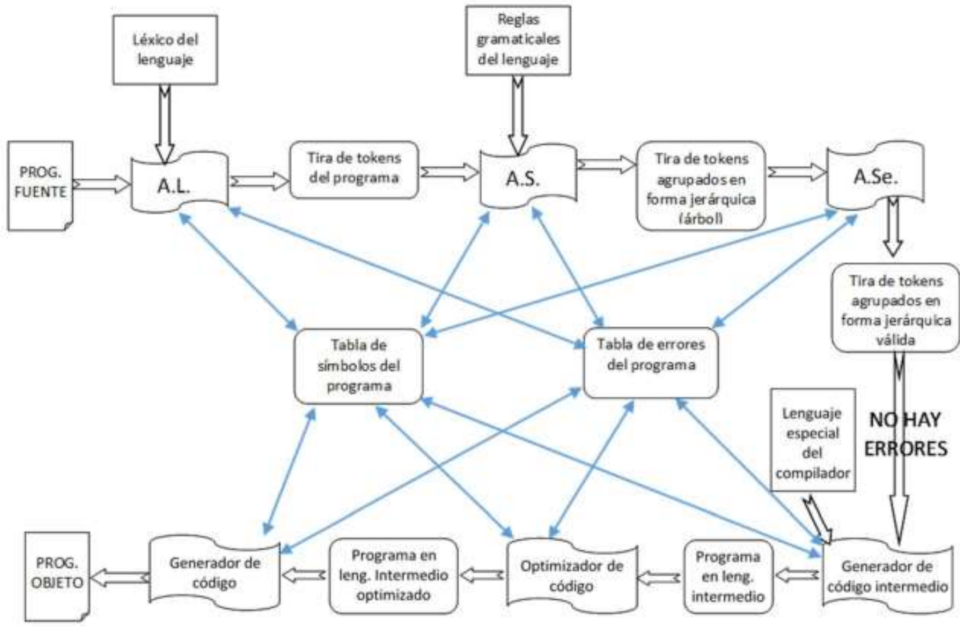
ERRORES: detecta un error cuando una construcción corresponde a una operatoria no valida (aunque fuera sintácticamente correcta).

GENERACION DE CODIGO INTERMEDIO: Después del AS y Ase algunos compiladores generaron un representación intermedia del programa fuente,
Este código intermedio es de facil traducción al lenguaje de maquina al que responde un determinado procesador.

GENERACION DE CODIGO OBJETO: Es la fase final, cada una de las instrucciones se traduce a una secuencia de instrucciones de maquina.

El programa objeto puede ser :

- Lenguaje de maquina absoluto: el programa se coloca en una posición fija de memoria y se genera el .exe para ejecutarlo.
- Lenguaje de maquina redireccionable: las posiciones de memorias no son fija. El .obj se debe someter al proceso de linkedicion para generar el .exe
- Lenguaje simbólico: es util en arquitecturas con poca memoria



RESUMEN CAPITULO 2 - INTRODUCCION A LOS LENGUAJES FORMALES

Lenguaje Natural -> a todo lenguaje hablado o escrito que es utilizado por los humanos para comunicarse entre si. Estos lenguajes tienen características

- Evolucionan con el paso del tiempo agregando nuevos términos y reglas gramaticales
- Cada palabra tiene una semantica (significado)
- Los lenguajes naturales presentan ambigüedades.

Lenguaje Formal -> están conformados por un conjunto de cadenas y estas cadenas están constituidas por símbolos de un alfabeto en particular. A diferencia de un lenguaje natural, los lenguajes formales tienen estas características

- Las cadenas que constituyen el lenguaje no tienen una semántica asociada
- Nunca es ambiguo
- Esta definido por reglas gramaticales prestablecidas

SIMBOLO o CARACTER -> entidad indivisible, los símbolos son letras (a,b,..,z), dígitos (1,2,3….9) y otros caracteres (+,-,/…..)
ALFABETO o VOCABULARIO -> conjunto finito de símbolos. Se definen como una enumeración de los símbolos. Se referencia con la letra V

V1 = {a, b, c, d, ….., z} Alfabeto castellano V2 = {0,1} alfabeto binario

CADENA SOBRE ALFABETO -> toda secuencia finita de símbolos de un determinado alfabeto. Un símbolo puede aparecer varias veces en una cadena. También se lo llama string, palabra. Se referencia con la letra s

s1 = abcsca -> cadena sobre el alfabeto V1
s2 = 01111011 -> cadena sobre el alfabeto V2

LONGITUD DE CADENA -> cantidad de símbolos de s. Se indica |s| ——> |s1| = 6 ; |s1| = 8

CADENA VACIA -> cadena de longitud cero (no pose simbolos). Se referencia con e |e| = 0

PARTES DE UNA CADENA

PREFIJO -> se define como prefijo a aquella cadena q no es ni s ni e.
ejemplo -> s = sintaxis ————> sin

SUFIJO -> se define como aquella cadena que ni es ni s ni e.
ejemplo -> s = sintaxis ————> taxis

SUBCADENA -> toda cadena que se obtiene de eliminar un prefijo y un sufijo sin ser s ni e.
ejemplo -> s = sintaxis ————> inda

SUBSECUENCIA -> cualquier cadena que se forme eliminado 0 o mas símbolos de s
ejemplo -> s = sintaxis ————> nax

Con un alfabeto finito puedo hacer infinitas cadenas.

OPERACIONES ENTRE CADENAS

CONCATENACION DE CADENAS -> Sean x e y dos cadenas sobre un alfabeto V, la concatenación de x e y se representa como x.y a una nueva cadena que se obtiene agregando y a x.

x = vice ; y = director ————> x.y = vicerrector && y.x = directorvice
Por lo tanto x.y != y.x

POTENCIA ENTERA n >= -> estamos en presencia de una definición por recurrencia
S = ja
S^o = e
S^1 = ja ————————- S^2 = S.S = jaja ———————— S^3 = S.S.S = jajaja

LENGUAJE -> conjunto (finito o infinito) de cadenas que se forman con un alfabeto V. Se representa con la letra L.

V = {0,1,2,3,4,5,6,7,8,9}
L1(V) = {n de 2 cifras}
L2(V) = {todos los numeros}
L3(V) = {nos binarios}

LENGUAJE VACIO -> es un conjunto vacío y que se denota con O o { }. No debe confundirse con un lenguaje que conténga una sola cadena y esta sea la cadena vacía.

DESCRIPCION DE UN LENGUAJE -> Los lenguajes formales puede ser descriptos por:

ENUMERACION o EXTENSION -> L1 = {0,1} ; M = {b, bb, bbbb, bbbbbb}

COMPRENSION -> M = {cadenasque contengan entre 1b y 5b inclusive}
N = {Nombres de los colores primarios}

EXPRESIONES REGULARES o GRAMATICAS FORMALES

OPERACIONES SOBRE LENGUAJES:

UNION -> L U M = {s / s esta en en L o esta en M}

CONCATENACION -> L . M = {s / s = r . T con r en L y t en M}

ejemplo R . S = {no, si} . {33, 9a} = {no33, no9a, si33, si9a}

POTENCIA -> L^0 = {e} ——— $L^i = L^{i-1} . L$

CERRADURA DE KLEENE -> lenguaje infinito que contiene al propio lenguaje y que contiene a la cadena vacía

$L^* = \{e\} \cup L^1 \cup L^2 \cup \ldots\ldots L^n$

$V^* = \{e\} \cup V^1 \cup V^2 \cup \ldots\ldots V^n$

{e} -> cadenas de longitud 0

V -> cadenas de longitud 1

V . V -> cadena de longitud 2

V . V . V -> cadenas de longitud 3 y así sucesivamente

CERRADURA POSITIVA DE L^+

$L^+ = L^1 \cup L^2 \cup \ldots\ldots L^n$