**Arquitectura Utilizada para Construir el Compilador de Lenguaje C**

Descomposición Modular o Modularización es el proceso de descomposición de un sistema en un conjunto de elementos con un índice bajo acoplamiento (independientes) y alto índice de cohesión (con significado propio).

Consiste en descomponer el problema a resolver en módulos o tareas más simples. Cada tarea representa una actividad completa y se codifica de manera independiente. Facilita el diseño descendente del problema, centrándonos cada vez en la resolución de subproblemas de magnitud inferior.

A la resolución de cada uno de estos subproblemas de complejidad inferior se denomina refinamiento por pasos. Los módulos pueden ser planificados, codificados, comprobados y depurados independientemente, y a continuación se combinan uno a uno con otros módulos.

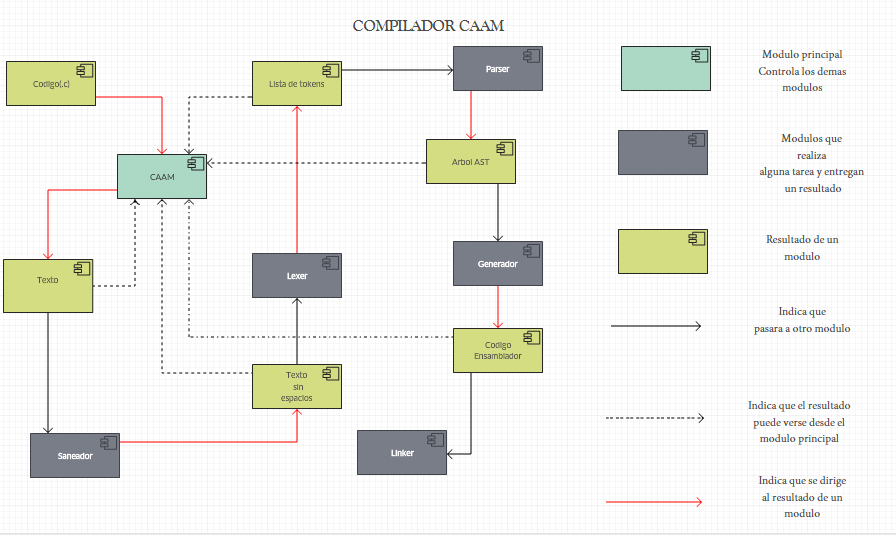
El diseño modular propone dividir el sistema en partes diferenciadas y definir sus interfaces.

Ventajas

* Claridad
* Reducción de costos
* Reutilización
* Se siguen los siguientes pasos para poder realizar la descomposición modular:
* Identificar los módulos
* Describir cada módulDescribir las relaciones entre módulos

Una descomposición modular debe poseer ciertas cualidades mínimas para que se pueda considerar suficiente validad.

* Acoplamiento
* Cohesión
* Comprensibilidad
* Adaptabilidad

**Arquitectura propuesta**

**CAAM**: Este será la estructura principal, la que contendrá a todas las demás estructuras que se diseñan a formato de capas. Des aquí se podrá llamar a cada una de las capas que tiene un trabajo específico, asi se podrá ver lo que realiza cada una de ellas asi como todo el resultado final.

**Saneador**: Tiene la función de quitar los espacios antes del código y después del código, asi sean saltos de línea o tabulaciones.

**Escaner**: Quita los espacios en blanco que haya entre el código y separa las palabras a demás de etiquetarlas para tener en una lista de tokens.

**Parser**: Aquí se revisa la sintaxis del código verificando que la lista de tokens sea la correcta.

**AST:** Es una herramienta que utiliza el parser para poder revisar la sintaxis del código, tambien llamado Árboles De Sintaxis Abstracta, este devuelve el árbol construido y revisado.

**Generador de código**: Con el árbol ya recibido, se procede a generar el código para un ensamblador pasando la sintaxis a su equivalente en código ensamblador.

**Linker:** es el encargado de combinar diferentes archivos con código objeto en un único archivo, y para esto hay dos tareas fundamentales, la resolución de símbolos y el “traslado” (relocation). Por un lado, los archivos de código objeto referencian y usan símbolos, el objetivo de la resolución de símbolos es asociar cada referencia con una única definición del símbolo.

**Estándares de codificación.**

Para los estándares de codificación se utilizará de la siguiente manera.

Para los módulos principales como podría ser el “Main”,”Lexer”, etc. Se utilizará el nombre de la función, sin espacios y cada palabra (de ser el caso) empezando con mayúsculas, también se usara el idioma en inglés, para que nos resulten más fáciles de identificar con los términos vistos en clase o en algún libro de teoría de compiladores, como se muestras con el generador de código, al cual se le llamara siguiendo esto Generador de Código = “Generator”.

Para las definiciones o funciones que se agregan con “def” se utilizara: El nombre de la función que realiza y en que modulo lo hace, todo separado por un guion bajo ( \_ ).

Por ejemplo: “defp compile\_file” --- Esto nos indica que compila el archivo.

defp print\_token\_list --- Este nos dice que mostrara la lista de tokens.

Ahora con los atomos, se nombrara con lo que debe contener dentro de la función, todo esto siguiendo lo visto para las cosas pasadas (Nombre en ingles,separadas con guion bajo),pero ahora para no confundirlas no empezaran con letra mayuscula,

Un ejemplo: :node\_name, :value, :left\_node --- Todo esto muestra que node\_name debe tener el nombre del nodo, value tiene el valor y left\_node nos indica que deberá usarse el nodo izquierdo

Y por último, las variables se llamaran de forma que nos ayuden a tener un entendimiento del código mas fluido, de esta forma se llamaran como lo que se necesita usar, asignar o guardar para poder programar de una manera mejor. Como por ejemplo: file\_name , code\_snippet, words. Etc.