1. Introducción

Las optimizaciones pueden realizarse de diferentes formas, estas se realizan en base al alcance ofrecido por el compilador. La optimización va a depender del lenguaje de programación y es directamente proporcional al tiempo de compilación; es decir, entre más optimización mayor tiempo de compilación.

1. Capítulo 1:

Sección 1.1: Locales

La optimización local sirve cuando un bloque de programa o sección es crítico por ejemplo: E/S, la concurrencia, la rapidez y confiabilidad de un conjunto de instrucciones. Como el espacio de soluciones es más pequeño la optimización local es más rápida. Como el espacio de soluciones es más pequeño la optimización local es más rápida.

Sección 1.2: Ciclos

Los ciclos son una de las partes más esenciales en el rendimiento de un programa dado que realizan acciones repetitivas, y si dichas acciones están mal realizadas, el problema se hace N veces más grandes. La mayoría de las optimizaciones sobre ciclos tratan de encontrar elementos que no deben repetirse en un ciclo.

Ejemplo: while(a == b) {   
int c = a;   
c = 5;  
 }

Sección 1.3: Globales

La optimización global se da con respecto a todo el código. Este tipo de optimización es más lenta pero mejora el desempeño general de todo programa. Las optimizaciones globales pueden depender de la arquitectura de la máquina.   
En algunos casos es mejor mantener variables globales para agilizar los procesos (el proceso de declarar variables y eliminarlas toma su tiempo) pero consume más memoria. Algunas optimizaciones incluyen utilizar como variables registros del CPU, utilizar instrucciones en ensamblador.

Sección 1.4: De mirilla

La optimización de mirilla trata de estructurar de manera eficiente el flujo del programa, sobre todo en instrucciones de bifurcación como son las decisiones, ciclos y saltos de rutinas. La idea es tener los saltos lo más cerca de las llamadas, siendo el salto lo más pequeño posible.   
  
Instrucciones de bifurcación: Interrumpen el flujo normal de un programa, es decir que evitan que se ejecute alguna instrucción del programa y salta a otra parte del programa. Por ejemplo: el “break”

1. Capítulo 2:

Sección 2.1: Costo de ejecución

Los costos de ejecución son aquellos que vienen implícitos al ejecutar el programa. En algunos programas se tiene un mínimo para ejecutar el programa, por lo que el espacio y la velocidad de microprocesadores son elementos que se deben optimizar para tener un mercado potencial más amplio.  
Las aplicaciones multimedia como los videojuegos tienen un costo de ejecución alto por lo cual la optimización de su desempeño es crítico, la gran mayoría de las veces requieren de procesadores rápidos o de mucha memoria.

Sección 2.2: Criterios para mejorar el código

La mejor manera de optimizar el código es hacer ver a los programadores que optimicen su código desde el inicio, el problema radica en que el costo podría ser muy grande ya que tendría que codificar más y/o hacer su código más legible.   
Los criterios de optimización siempre están definidos por el compilador, algunos de estos criterios pueden modificarse con directivas del compilador desde el código o de manera externa. Este proceso lo realizan algunas herramientas del sistema como los ofuscadores para código móvil y código para dispositivos móviles.

Sección 2.3: Herramientas para el análisis de flujo de datos

Existen algunas herramientas que permiten el análisis de los flujos de datos, entre ellas tenemos:

* Depurador: Permite correr otros programas, permitiendo al usuario ejercer cierto control sobre los mismos a medida que los estos se ejecutan, y examinar el estado del sistema en el momento en que se presente algún problema. El propósito final de un depurador consiste en permitir al usuario observar y comprender lo que ocurre "dentro" de un programa mientras el mismo es ejecutado.
* Desamblador: Es un programa de computadora que traduce el lenguaje de máquina a lenguaje ensamblador, la operación inversa de la que hace el ensamblador. Un desensamblador difiere del descompilador, en que éste apunta a un lenguaje de alto nivel en vez de al lenguaje ensamblador.
* Diagrama de flujo de datos: Es una herramienta de modelización que permite describir, de un sistema, la transformación de entradas en salidas; el DFD también es conocido con el nombre de Modelo de Procesos de Negocios.
* Diccionario de datos: El Diccionario de Datos es un listado organizado de todos los elementos de datos que son pertinentes para el sistema, con definiciones precisas y rigurosas que le permite al usuario y al sistema tener una misma comprensión de las entradas, de las salidas, de los componentes de los repositorios, y también de cálculos intermedios.

1. Conclusión:

En conclusión la optimización nos permite mejorar el proceso de un sistema mediante la maximización o reducción de algunos las variables que interactúan con el proceso (Local, global, de mirilla y ciclos). Esto nos sirve para reducir costos que puedan llegar a ser beneficiosos para el proceso del programa. Una herramienta que es de mucha utilidad para reconocer la complejidad de nuestro programa es la utilización de herramientas de flujo de datos, estos además nos describen los pasos que lleva a cabo nuestro sistema.

1. Conceptos:

**Optimización:** La optimización es un proceso que tiene a minimizar o maximizar alguna variable de rendimiento, generalmente tiempo, espacio, procesador, etc.

**Bifurcación:** División de una cosa en dos ramales, brazos o puntas.

**Ofuscadores:** Se refiere al acto deliberado de realizar un cambio no destructivo, ya sea en el código fuente de un programa informático o código máquina cuando el programa está en forma compilada o binaria, con el fin de que no sea fácil de entender o leer.

**CPU:** Unidad central de procesamiento

**Arquitectura:** Es el diseño conceptual y la estructura operacional fundamental de un sistema de Computadora.

1. Bibliografía:

Unidad III. (s.f.-b). Recuperado 7 noviembre, 2018, de http://itpn.mx/recursosisc/7semestre/leguajesyautomatas2/Unidad%20

Aho (2006), et. al. Compiladores: Principios y Técnicas. Segunda Edición.

Aho, Lam, Sethi, Ullman Addison (2008) Compiladores: Principios, técnicas y herramientas. Segunda Edición Addison

A. Garrido, J. Iñesta, F. Moreno y J. Pérez. (2002) Diseño de compiladores. Edita Universidad de Alicante

Reporte

La optimización en general busca aumentar o reducir algún estado del proceso, en este caso se principal propósito es reducir la complejidad del programa para tener un mejor rendimiento. Los tipos de optimización son:

* Local: Esta se realiza dentro de los módulos del programa, se utiliza cuando un bloque de programa o sección es crítico.
* Ciclos: Se utilizan cuando son necesarias las repeticiones.
* Global: Se aplica en todo el código, su misión es mejorar el desempeño de todo el programa.
* De mirilla: Se encarga de darle una estructura a la orientación del programa, ya que te dirigirá respecto a las decisiones que se tomen.

En costos hablan principalmente de las formas en las que se puede reducir tiempo y espacio a la hora de realizar el código.   
Un ejemplo es un programa que mande imprimir 5 números aleatorios con una declaración de impresión por número y el segundo programa con un ciclo while que mande a imprimir números aleatorios. Con 5 números no se notaría mucho la diferencia en tiempo, pero si en lugar de 5 fueran 500 aumentarían los recursos del primer programa debido a que se tendrían que escribir 500 declaraciones, en cambio en el segundo programa solamente se repetiría la misma declaración 500 veces.

Algunos de las herramientas para el análisis de diagramas de flujo son:

Depurador: El cual permite revisar el estado del programa referente a estados de entrada/salida, errores y tiempos.

Desamblador: Traduce el lenguaje maquina a lenguaje ensamblador.

Modelo de diagrama de flujo: Esta herramienta permite describir el proceso de un sistema.

Diccionario de datos: Guarda un conjunto de datos necesario para el proceso del sistema.