Descripción breve

En este documento se hará un resumen de la unidad 4.

Resumen unidad 3

Jesus Oviedo Ovalle #13480497

**Tipos de optimización**

Las optimizaciones pueden realizarse de diferentes formas, estas se realizan en base a el 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.

Como el tiempo de optimización es gran consumidor de tiempo (dado que tiene que recorrer todo el árbol de posibles soluciones para el proceso de optimización) la optimización se deja hasta la fase de prueba final. Algunos editores ofrecen una versión de depuración y otra de entrega o final.

**Locales**

La optimización local se realiza sobre módulos del programa. En la mayoría de las ocasiones a través de funciones, métodos, procedimientos, clases, etc.

La característica de las optimizaciones locales es que sólo se ven reflejados en dichas secciones. Esta sirve cuando un bloque de programa o sección es crítico, por ejemplo; la 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.

**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.

El problema de la optimización en ciclos y en general radica en que es muy difícil saber el uso exacto de algunas instrucciones. Así que no todo código de proceso puede ser optimizado. Otro uso de la optimización puede ser en el mejoramiento de consultas en SQL o en aplicaciones remotas (sockets, E/S, etc.).

**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.

Grafo del flujo de ejecución: antes de realizar una optimización global es necesario crear el grafo de flujo de ejecución. El grafo de flujo de ejecución representa todos los caminos posibles de ejecución del programa. La optimización global a partir del análisis del grafo del flujo de ejecución permite:   
-Una propagación de constantes fuera del bloque básico.  
-Eliminación del código no utilizado  
-Una mejor asignación de los registros.

**De mirilla**

La optimización de mirilla trata de estructurar de manera eficiente el flujo del programa, sobre todo en instrucciones de bifurcaciones 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. Algunas ideas básicas son:

-Se recorre el código buscando combinaciones de instrucciones que pueden ser reemplazadas por otras equivalentes más eficientes.

-Se utiliza una ventana de “n” instrucciones y un conjunto de patrones de transformación (patrón, secuencias, etc.).

-Las nuevas instrucciones son reconsideradas para las futuras optimizaciones.

Algunos ejemplos con la eliminación de cargas innecesarias la reducción de potencia y la eliminación de cadenas de saltos.

**Costos**

Los costos son el factor más importante a tomar en cuenta a la hora de optimizar ya que en ocasiones la mejora obtenida puede verse no reflejada en el programa final, pero sin ser perjudicial para el equipo de desarrollo. La optimización de una pequeña mejora tal vez tenga una pequeña ganancia en tiempo o en espacio, pero sale muy costosa en tiempo generarla.

Pero en cambio si esa optimización se hace por ejemplo en un ciclo, la mejora obtenida puede ser “N” veces mayor por lo cual el costo se minimiza y es benéfica la mejora.

**Costo de ejecución (memoria, registros, pilas)**

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 los 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 crítica, la gran mayoría de las veces requieren de procesadores rápidos (e.g. tarjetas de video) o de mucha memoria. Otro tipo de aplicaciones que deben optimizarse son las aplicaciones para dispositivos móviles.

Los dispositivos móviles tienen recursos más limitados que un dispositivo de cómputo convencional razón por lo cual, el mejor uso de memoria y otros recursos de hardware tiene mayor rendimiento. En algunos casos es preferible tener la lógica del negocio más fuerte en otros dispositivos y hacer uso de arquitecturas descentralizadas como cliente/servidor o **P2P**.

**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.

Muchos 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.

**Herramientas para el análisis del flujo de datos**

Existen algunas herramientas que permiten el análisis de los flujos de datos, entre ellas tenemos los **depuradores** y **desensambladores**. La optimización al igual que la programación es un arte y no se ha podido sistematizar del todo.

[2] Gabriela Fernández Espinoza. (2013). Tipos de optimización. 15/04/18, de Blogger.com Sitio web: <http://gaferz.blogspot.mx/2013/11/tipos-de-optimizacion.html>

[3] Juan Carlos Olivares Rojas. (-). Optimización. 15/04/2018, de IT Morelia Sitio web: <http://dsc.itmorelia.edu.mx/~jcolivares/courses/ps207a/ps2_u7.pdf>

[4] ITPN. (2012). Optimización. 15/04/18, de ITPM.mx Sitio web: <http://itpn.mx/recursosisc/7semestre/leguajesyautomatas2/Unidad%20III.pdf>