Instituto tecnológico de nuevo león.

Ingeniería en sistemas computacionales.

Materia: Lenguajes y Autómatas II

Tarea: Resumen de la unidad III

Alumno: Gaspar Cruz Melendres.

Numero Control: 14480839

Catedrático: Juan Pablo Rosas Baldazo

Guadalupe, Nuevo León. 16 de Abril del 2017

1. Introducción

En este trabajo empezaremos a ver cuales son las diferentes optimizaciones que existe para su mayor manejo dentro de ejecutar programas por las cuales tenemos que ir viendo primero antes que nada que es optimización siendo que, siendo que así esto nos ayudara a identificar bien nuestras operaciones de optimización.

Por lo que a la vez debemos de ir viendo también las estructuras que debemos de tomar para una buena comprensión del código siendo que si tu lo pones digamos en global será mas lenta siendo que con otras optimizaciones podrás superar eso y no será lentas.

Por otro lado, también tendremos que ver las herramientas con las cuales nos ayudaran a comprender mejor nuestro código por lo cual nosotros podremos elegir bien la optimización de los códigos que elaboraremos.

2. Capitulo 1: Tipos de optimización

Las optimizaciones pueden realizarse de diferentes formas. Las optimizaciones 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. La optimización es un proceso que tiene a minimizar o maximizar alguna variable de rendimiento, generalmente tiempo, espacio, procesador, etc.

Dentro de los tipos de optimización se derivan los tipos de optimización local, optimización de ciclo, optimización global y optimización de mirilla

○ *Sección 1.1: 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 solo se ven reflejados en dichas secciones.

Las optimizaciones locales se realizan sobre el bloque básico

• Optimizaciones locales

- Folding.
- Propagación de constantes.
- Reducción de potencia.
- Reducción de subexpresiones comunes.

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.

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 el mejoramiento de consultas en SQL o en aplicaciones remotas (sockets, E/S, etc.).

- ***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.

Puntos esenciales para la optimización global.

- ✓ 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.

Ideas básicas:

- ✓ 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, remplazan).
- ✓ Las nuevas instrucciones son reconsideradas para las futuras optimizaciones.

Ejemplos:

- ✓ Eliminación de cargas innecesarias
- ✓ Reducción de potencia
- ✓ Eliminación de cadenas de saltos

3. Capítulo 2: 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 si 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 en 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 benéfico la mejora.

Sección 2.1: Costo de ejecución (memoria, registros, pilas)

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.

Por ejemplo:

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 la cual, el mejor uso de memoria y otros recursos de hardware tiene mayor rendimiento.

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

Criterios de optimización

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.

- ***Sección 2.3: 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 desambladores. La optimización al igual que la programación es un arte y no se ha podido sistematizar del todo.

4. Conclusiones

Como podemos ver la información de cada uno de las optimizaciones son muy importantes ya que con esos podemos ver las diferencia a de las operaciones cuando el código esta en funcionamiento siendo que para ellos las optimizaciones tendrá un punto esencial en la vida de cada proyecto ya que con esto no ayudara a comprender más un condigo que cuando tiene muchas líneas de código.

Por otro lado, como toda cosa debe de tener herramientas para poder llevar a tener a cabo sus procesos por eso tenemos lo que son los depuradores y desambladores, siendo que con ello tendremos una comprensión de cada diagrama de flujo para así sabes en que procesos estamos haciendo mas de lo normal y que con eso podremos reducir el código de cada uno de ello.

Una de las cosas que mas comprendí es que una optimización podrá tener más optimizaciones dentro de cada optimización hasta que el proceso sea más eficaz (llegara a su objetivo con los recursos menos necesarios para su operación) siendo que por un lado será eficiente (llegara a su objetivo, pero con el tiempo que sea necesario).

Una de las cosas que se puede ver es que en un trabajo tendrás que hacer estos trabajos para poder sacar a delante un código ya que si tiene un montón de código llegara a ser eficaz y con eso contribuye que el código se vea muy amontona y tengas cosas redundantes por lo cual tendrás mas espacio en la memoria que no ayudaría nada el código.

Una de las cosas que vi es que cada una de las optimizaciones que existe tiene una función con la cual es beneficiada para cada una de las operaciones que hace el código siendo que no siempre esa optimización es adecuada para cualquier código que hayas realizado, ya que a veces no es necesario tener las cosas en global si no lo ocupas (ejemplo) por lo que debes de tener en cuentas las los beneficios que se obtendrá con cada optimización de códigos.

5. Conceptos

✓ *Compilador.*

Es un programa informático que traduce un programa que ha sido escrito en un lenguaje de programación a un lenguaje común, reúne diversos elementos o fragmentos en una misma unidad.

✓ *Ciclo.*

También se trata de la secuencia de etapas que atraviesa un suceso de características periódicas y del grupo de fenómenos que se reiteran en un cierto orden.

✓ *Módulos.*

Una estructura o bloque de piezas que, en una construcción, se ubican en cantidad a fin de hacerla más sencilla, regular y económica. Todo módulo, por lo tanto, forma parte de un sistema y suele estar conectado de alguna manera con el resto de los componentes.

✓ *Subexpresiones.*

Subexpresión (subexpresiones plural) (matemáticas, programación) Una expresión que es una parte de una expresión mayor.

✓ *Folding.*

El ensamblamiento es remplazar las expresiones por su resultado cuando se pueden evaluar en tiempo de compilación (resultado constante).

✓ *Memoria.*

Es el almacén donde el autómata guarda todo cuanto necesita para ejecutar la tarea de control:

- Datos de programa: Señales de E/S, variables internas y datos alfanuméricos y constantes.
- Datos de control: Instrucciones de usuario (programas) y configuración del autómata.

En general, toda esta información es guardad en memorias de semiconductor. Esta memoria es un dispositivo electrónico capaz de almacenar datos binarios.

✓ *Registros.*

Son los elementos más valiosos y escasos en la fase de generación de código, puesto que el CPU solamente puede procesar datos que se encuentren en registros. Además, las

instrucciones que implican operandos en registros son más cortas y rápidas que las de operandos en memoria.

- ✓ *Propagación de constantes.*

Estas optimizaciones permiten que el programador utilice variables como constantes sin introducir ineficiencias.

- ✓ *Pilas.*

Es un modelo matemático de un sistema que recibe una cadena constituida por símbolos de un alfabeto y determina si esa cadena pertenece al lenguaje que el autómata reconoce.

- ✓ *Depuradores*

Es un programa usado para probar y depurar (eliminar) los errores de otros programas (el programa "objetivo"). El código a ser examinado puede alternativamente estar corriendo en un simulador de conjunto de instrucciones (ISS), una técnica que permite gran potencia en su capacidad de detenerse cuando son encontradas condiciones específicas, pero será típicamente algo más lento que ejecutando el código directamente en el apropiado (o el mismo) procesador.

6. Bibliografía o Referencias

López, N. (2013). *Lenguajes y autómatas II*. Abril 15, 2018, de Blogger. Sitio web: http://noeliy22.blogspot.mx/2013/11/tipos-de-optimizacion.html

Fernández, G. (2013). *Tipos de optimización*. Abril 15, 2018, de Blogger Sitio web: http://gaferz.blogspot.mx/2013/11/tipos-de-optimizacion.html

Carlos, J. (2016). *Optimización Global*. Abril15, 2018, de Blogger. Sitio web: http://juancarlossant.blogspot.mx/2016/11/optimizacion-global.html

Silverio, A.. (2018). *3.2 Costos*. Abril 15, 2018, de Padlet Sitio web: https://padlet.com/nabor_alfredo/92rlc7yhslb1

Sin autor. (2013). *Unidad III*. Noviembre 13, 2018, de S/N Sitio web: http://itpn.mx/recursosisc/7semestre/leguajesyautomatas2/Unidad%20III.pdf

Santiago, J. (2016). *Optimización Local.* Abril 15, 2018, de Blogger. Sitio web: http://juancarlossant.blogspot.mx/2016/11/optimizacion-local.html