**INSTITUTO TECNOLÓGICO DE NUEVO LEÓN**



**Lenguajes y Autómatas 2**

**UNIDAD # 3**

**Nombre de la Unidad: Optimización**

**PROYECTO # 3: OPTIMIZACION**

**Catedrático:** Juan Pablo Rosas Baldazo

**Nombre:** Yessica Yolanda Rodríguez Rosas

**Matrícula**: 14480169

Guadalupe N.L. a 09 de noviembre del 2018

1. Introducción
2. Capítulo 1: Tipos de optimización

* Sección 1.1: Locales
* Sección 1.2: Ciclos
* Sección 1.3: Globales
* Sección 1.4: De mirilla

1. Capítulo 2: Costos

* Sección 2.1 Costo de ejecución(memoria, registros, pilas)
* Sección 2.2: Criterios para mejorar el código
* Sección 2.3: Herramientas para el análisis de flujo de datos

1. Conclusiones
2. Conceptos
3. Bibliografía o Referencias

**INTRODUCCIÓN**

En este resumen y reporte se hablará de los temas como por ejemplo:

* Costos
* Optimización y sus tipos

Podremos saber cuáles son los tipos de optimización que existen en la programación.

Se tocará el tema de los costos del código, que estos a su vez se dividen en costos de memoria, de registros y de pilas.

**Capítulo 1: Tipos de optimización**

Entre los tipos de optimización podemos encontrar se derivan en:

* Optimización local
* Optimización de ciclos
* Optimización global
* Optimización de mirilla

**Sección 1.1: Locales**

Las optimizaciones locales se realizan sobre el bloque básico.

Un bloque básico es un fragmento de código que tiene una única entrada y salida, cuas instrucciones se ejecutan secuencialmente.

Para la optimización local se utiliza alguna de las siguientes optimizaciones:

* Ensamblamiento
* Simplificación algebraica
* Eliminación de subexpresiones comunes
* Propagación de copias
* Propagación de código y reducción de constantes
* Eliminación de código muerto

A continuación se hablará brevemente de cada una de las optimizaciones locales.

**Ensamblamiento o Folding**

Este consiste simplemente en remplazar las expresiones por su resultado cuando se pueden evaluar en un tiempo de codificación. A continuación se muestra un ejemplo del folding:

X= 3+2+T+B

X=5+T+B

**Simplificación algebraica**

Esta optimización consiste en poder eliminar o simplificar declaraciones hechas en el código. A continuación se muestra un ejemplo:

X=x+0

X=x+1

**Eliminación de subexpresiones comunes**

En esta optimización todas las asignaciones en la parte derecha calculan el mismo valor como por ejemplo:

X=y+z x=y+z

W=y+z w=x

**Propagación de copias**

En este un ejemplo sería el siguiente: Si w=x aparece en un bloque todos los usos subsecuentes de w pueden reemplazarse por x.

B=z+y b=z+y

A=b a=b

X=2\*a x=2\*b

**Propagación de código y reducción de constantes**

Se mencionara un ejemplo para el mejor entendimiento de esta optimización

A=5 a=5

X=2\*a x=10

Y=x+6 y=16

**Eliminación de código muerto**

Muerta = no contribuye al resultado del programa. A continuación se muestra un ejemplo de esta optimización:

X=Z+Y B=Z+Y B=Z+Y

A=X A=B X=2\*B

X=2\*A X=2\*B

**Sección 1.2: Ciclos**

La mayoría de las optimizaciones sobre ciclos tratan de encontrar elementos que no deben repetirse en un ciclo.

La optimización de bucles es muy importante por las mejoras en tiempo de ejecución.

Las optimizaciones de ciclos de desencadenan en las siguientes:

* Expansión de bucles
* Reducción de frecuencia
* Reducción de potencia
* Eliminación de variables de inducción

A continuación se mencionará el primer tipo de optimización de ciclos que es el de expansiones de bucles.

**Expansiones de bucles**

Solo se puede aplicar a los bucles cuyo número de iteraciones se conoce en tiempo de compilación.

Se puede aplicar a los bucles:

for(i=1;i<=10;i++)

No se puede aplicar a los siguientes bucles:

for(i=a;i<=b;i++)

El problema de la optimización de ciclos y en general radica en que es muy difícil saber el uso exacto de algunas instrucciones. Así que no todo el código de proceso puede ser optimizado. Otros usos de la optimización puede ser el mejoramiento de consultas SQL o en aplicaciones remotas.

**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 en general de todo el programa. Las optimizaciones globales pueden depender de la arquitectura de la máquina.

En algunos casos es mejor mantener variables globales para agilizar los procesos 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 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.

**Capítulo 2: Costos**

Los costos son el factor más importante a tomar en cuenta a la hora de optimizar.

**Sección 2.1: Costos de ejecución (memorias, 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 del microprocesador son elementos que se deben optimizar.

**Memoria**

La memoria es uno de los recursos más importantes de la computadora y en consecuencia, la parte del sistema operativo responsable de tratar con este recurso, el gestor de memoria, es un componente básico del mismo. El gestor de memoria del sistema operativo debe hacer de puente entre los requisitos de las aplicaciones y los mecanismos que proporciona el hardware de gestión de memoria.

Algunos lenguajes de programación utilizan la pila para almacenar datos que son locales a espacio para los datos locales se asignan a los temas de la pila cuando el procedimiento se introduce y son borradas cuando el proceso termina.

Para datos, básicamente la memoria se divide en:

* Memoria estática
* La pila
* Heap el monton

**Registros**

Los registros de procesador se emplean para controlar instrucciones en ejecución, manejar direccionamiento de memoria y proporcionar capacidad aritmética. Los registros son espacios físicos dentro del microprocesador con capacidad de 4 bits hasta 64 bits dependiendo del microprocesador que emplee. Los registros son direccionales por medio de una viñeta, que es una dirección de memoria.

**Tipos de registros**

* Registro de segmento
* Registros de apuntador de instrucción
* Registros apuntadores
* Registros de propósito general
* Registros índices
* Registros de banderas

**Pila**

La aparición de lenguajes con estructura de bloque trajo consigo la necesidad de técnicas de alojamiento en memoria más flexibles, que pudieran adaptarse a las demandas de memoria durante la ejecución del programa.

**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 principio.

El problema radica en que el costo podría ser muy grande ya que tendría que codificar más y/o hacer su código 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 de 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 desensambladores. La optimización al igual que la programación es un arte y no se ha podido sistematizar del todo.

**Depuradores**

Un depurador es una aplicación que permite correr otros programas, permitiendo al usuario ejercer cierto control sobre los mismos a medida que los están ejecutando 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.

**Desensambladores**

Es un programa de computador que traduce el lenguaje ensamblador, la operación inversa de la que hace el ensamblador. La salida de un desensamblador, es a menudo formateada para la legibilidad humana en vez de ser adecuada para la entrada a un ensamblador, haciendo que este sea principalmente una herramienta de ingeniería inversa.

**CONCLUSIONES**

Cuando se trata de la programación debemos de optimizar código, optimizar memoria para que así el tiempo de ejecución sea más rápido.

Cuando hablamos de tipos de optimización podemos contar con 4 tipos, podemos utilizar ya sea locales, globales, de mirilla y de ciclos, cada una de esas optimizaciones nos dice cuales es la mejor o la que podemos utilizar o simplemente sin darnos cuenta la estamos utilizando.

**CONCEPTOS**

**Optimización:** Buscar la mejor manera de realizar una actividad, como por ejemplo optimización de software la cual busca adaptar programas informáticos para que realicen sus tareas de la forma más eficiente posible.

**Bloque básico:** Es un fragmento de código que tiene una única entrada y salida, cuas instrucciones se ejecutan secuencialmente.

**Ensamblamiento:** Fase de compilación de un programa, consiste en transformar un archivo escrito en lenguaje ensamblador.

**Instrucciones de bifurcación:** Estas 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.

**Heap El Monton:** Cuando el tamaño de un objeto a colocar en memoria puede variar en tiempo de ejecución, no es posible ubicarlo en memoria estática, ni siquiera en pila.

**REPORTE**

Capítulo 1 Tipos de optimización

Cuando hablamos de tipos de optimización nos referimos a los siguientes:

* Locales
* Globales
* De mirilla
* De ciclos

En la sección 1.1 se habló de los tipos de optimización pero se empezó a describir la optimización local, en la cual nos dice que hay diferentes tipos de optimización local, los cuales son:

* Propagación de copias
* Eliminación de código muerto
* Ensamblamiento
* Simplificación algebraica

En la de Ensamblamiento nos dice que se tiene que reducir la expresión como por ejemplo si tenemos x= 5+8+y+z podemos modificarlo para que el código sea x=13+y+z ocupando así menos espacio y optimizando nuestros recursos.

En la sección 1.2 tratamos otro de los tipos de optimización el cual es la optimización de ciclos en ella nos dice que es muy importante para la reducción de tiempo de ejecución del código, en esta optimización también se pueden desglosar algunos tipos de optimizaciones de ciclos, está por mencionar una la expansión de bucles en la cual nos menciona como es que se puede realizar esta así como los ciclos en los que aplica y los que no, especificándonos con un ejemplo muy fácil como aplica y como no.

En la sección 1.3 se tocó el tema de la optimización global, que esta nos dice que hablamos con ella de todo el código.

También nos dice que a veces es mejor mantener algunas variables globales para así poder agilizar los procesos, pero que también tiene una parte negativa la cual es que nos consumirá más memoria.

Sección 1.4 hablamos de la optimización de mirilla, en esta optimización nos dice que aquí se trata de estructurar de una manera eficiente el flujo de los programas y más en las instrucciones de bifurcación, con esto se quiere decir que estas instrucciones evitan que se ejecute una instrucción del programa interrumpiendo así el flujo del mismo.

Capítulo 2 Costos

A la hora de optimizar los costos son un factor importante que debemos de tener en cuenta.

Sección 2.1 Los costos que se consideran primordiales son los de registros, pilas y memoria, siendo la memoria el costo más importante a la hora de optimizar.

Para datos en memoria se dividen en 3 los cuales son los siguientes:

* Memoria estática
* Pilas
* Heap el monton

También existen varios tipos de registros los cuales son:

* Registros de apuntadores
* Registro de bandera
* Registro de segmento

Por mencionar algunos de estos tipos de registros.

Sección 2.2 Nos dice que tenemos que tener varios puntos en cuenta a la hora de optimizar el código, como por ejemplo que debemos hacer que los programadores hagan que su código este optimizado desde el inicio del mismo, aunque como hablamos de costo, en este sería costoso debido a que el código debe ser claro y sería un poco largo.

Sección 2.3 nos dice que las herramientas que podemos utilizar para el flujo de análisis de datos son los depuradores y los desensambladores.

**BIBLIOGRAFIA O REFERENCIAS**

<https://prezi.com/ywsyzson5rda/optimizacion-local/>

<https://prezi.com/whrzzyslwovn/optimizacion-de-ciclos/>

<https://prezi.com/f2ankz9ryt2m/optimizacion-de-mirilla/>

<https://prezi.com/iu8b3klzevha/tipos-de-optimizacion-global/>

<https://prezi.com/m-ft53psccpy/321-costo-de-ejecucion-memoria-registros-pilas/?webgl=0>

<https://padlet.com/nabor_alfredo/92rlc7yhslb1>