Tarea 1 - Programación y algoritmos

Giovanni Gamaliel López Padilla

1. **Resumen de la publicación *S. Kumar and P. K . Singh "An overview of modern cache memory and performance analysis of replacement policies" 2016 IEEE International Conference on Engineering and Technology (ICETECH), 2016, pp. 210-214, doi: 10.1109/ICETECH.2016.7569243***

   La cache de un procesador es una memoria más rápida que la memoria principal. La existencia de la cache evita que el sistema busque una misma información en la memoria principal. El rendimiento de la cache es basada en tres parámetros: *miss rate, miss penalty y average access time.*

   El sistema de memoria de un uniprocesador esta diseñada por niveles de cache, el cual suministra datos e instrucciones a un solo procesador. Para multiprocesadores, el cache es solo un componente del sistema de memoria, el cual se le añade una interconexión entre chips, coherencia y consistencia de datos. En procesadores multicore cada core esta conectado a registros de ALU, pipeline, unidades de control, L1 y cache.

   Las tres principales unidades de un procesador es la unidad de instrucción, unidad de ejecución y unidad de almacenamiento. La unidad de instrucción es la responsable de organizar busquedad y decodificaciones de un programa. La unidad de ejecución es la responsable de la lógica, aritmética y ejecutar las acciones que le da la unidad de instrucción. La unidad de almacenamiento establece una conexión entre la unidad de instrucción y ejecución.

   Existen técnicas de mapeo para determinar la organización del cache. Las técnicas de mapeo son usadas para mapear un número largo en los bloques de la memoria principal en pocas lineas de la memoria cache y asignar bits. Las tres técnicas de mapeo son *direct mapping, associative mapping y set associative mapping.* La técnica que es considerada mejor es set associative mapping, esto porque tiene un mejor hit rate y access time.

   Las políticas del cache deciden la consistencia y mantenimiento entre lineas de la cache correspondientes a los bloques de memoria principal. *Write Caching, Write Back y Write Through* son las principales políticas de la cache. En la política write back las operaciones son realizadas únicamente en la cache, la memoria principal es actualizada cuando la cache donde se escribirá esta llena. Con la política Write Through las operaciones son realizadas en la memoria principal. Las operaciones realizadas por las dos políticas pueden obtener resultados inconsistentes, esto surge porque si dos memorias cache contienen la misma linea de datos y una es actualizada, la cache que no fue actualizada contiene datos invalidos. A menos que exista un monitor que realice notificaciones cuando una linea es actualizada. Write Caching es una mezcla de write back y write caching donde una pequeña full associative cache es incrustrada detrás de la write Through cache.

   Los algoritmos para remplazar datos en la cache juegan un rol importante en el diseño de la memoria cache porque decide que linea de datos será cambiada con el bloque de memoria principal. Los algoritmos de remplazamiento más usados son *First in First Out (FIFO) , Least Frequently (LFU) y Least Recently Used (LRU).* LRU es el algoritmo más efectivo porque es facil de implementar. Actualmente los procesadores incluyen multiple niveles de cache y la asociatividad entre ellos provoca que se realize un chequeo de las políticas de remplazamiento. Estos procesadores emplean los algoritmos LRU, FIFO, LFU y random. El funcionamiento del algortimo LRU necesita el número del estatus de cada bit para llevar un registro de de los bloques a los que se accedió. Cuando la política de LRU golpea o falla, el bloque que iba a ser remplazado requiere más energía y tiempo. Es por ello, que el algoritmo random puede ser usado para reducir la complejidad y costo a comparación de LRU.

El algoritmo random escoge un bloque candidato para ser descartado y así usar esa linea para remplazar los datos de la memoria principal. Este algoritmo no mantiene el histórico de los accesos a memoria.

Se realizo la comparación del parámetro hit rate para los algoritmos FIFO, LFU, LRU y random para distintos benchmarks. En el cual se visualizo que el algoritmo LRU es más confiable, ya que fue el que tuvo valores más altos en la mayoria de los benchmarks pero que este puede ser remplazado por el algoritmo random o LFU.

2. **Describir detalladamente los paradigmas de computación principales y su relacion. Dar ejemplos de cada paradigma.**

   Los paradigmas de programación indica un método de realizar cómputos ,la estructura y organización de las tareas que se deben realizar en un programa. Los paradigmas principales están basados en diferentes modelos de cómputo, por lo tanto afectan a las construcciones más básicas de un programa. Los paradigmas principales son:

   - Estructurado.
   - Lógico.
   - Funcional.
   - Orientado a objetos.

Estructurado

El paradigma estructurado consiste en en una serie de sentencias ejecutadas según un control de flujo explícito que modifican el estado del programa. [1]

Ejemplo

```
total=0;
for (i=1; i<=10; i++)
{
      total=total+1
}
```

Lógico

El paradigma lógico se describe en términos de su respuesta, pero no se define explícitammente qué propiedades se espera que presente el resultado [2].Contiene un control del flujo que esta asociado a una composición de funciones, recursividad y/o técnicas de reescritura y unificación. [3, 4]

Ejemplo

```
mujer(maria).
mujer(luisa).
mujer(juana).
mujer(alicia)
toca_guitarra(luisa).
toca_guitarra(maria).
oye_musica(X) :- toca_guitarra(X).

input:      ?-mujer(maria)
output:     true

input:      ?-mujer(ana)
output:     false
```

Funcional

En las matemáticas se establecio el concepto de *función*, las cuales establecen una relación entre los parámetros y el resultado. Un programa consiste en la definición de una o más funciones. Para la ejecución del programa se establecen parámetros y el programa realiza las operaciones necesarias para obtener el resultado. [5] Hace uso del cálculo lambda.

Ejemplo

```
1       void suma(int n1, int n2){
2             printf("La suma es %d",n1+n2);
3       }
4       int main(){
5             suma(1,2);
6       }
```

Orientado a objetos

Un lenguaje de programación orientado a objetos puede contener el paradigma imperativo, funcional o lógico. Lo que caracteriza este estilo es la forma de manejar la información basada en los conceptos clase, objeto y herencia. [1]

- Clase
  Tipo de dato con propiedades determinadas con funciones.
- Objeto
  Entidad de una clase con una serie de parámetros capaces de interactuar con otros objetos.
- Herencia
  Propiedad por la que es posible construir una nueva clase a partir de una ya existente.

Ejemplo

```
class person:
      def __init__(self,name: str):
            self.name=name

person1=person("Carlos")
print(person1.name)

>Carlos
```

Referencias

[1] J. J. Rodríguez Sala, L. Santamaría Arana, A. Rabasa Dolado, and O. Martínez Bonastre, *introducción a la programación. Teoría y práctica*. Gamma, 2003.

[2] J. Daintith, *A dictionary of computing*. Oxford New York: Oxford University Press, 2008.

[3] C. Sartori, *Computación y programación funcional : introducción al cálculo lambda y la programación funcional usando Racket y Python*. Barcelona: Marcombo, 2021.

[4] I. Bratko, *Prolog programming for artificial intelligence*. Harlow, England New York: Addison-Wesley, 2012.

[5] J. Fokker, *Programación Funcional*. Universidad de Utrecht, 1996.