Logotipo, Icono

Descripción generada automáticamente

**Instituto Tecnológico y de Estudios Superiores de Occidente.**

**Práctica 2: Snake.**

Yael Alejandro Rodríguez Barreto

Alberto Rentería Camacho.

Profesor Aurelio Antonio Villalpando Bravo.

Organización y arquitectura de computadoras.

(O2023\_ESI3913B).

30 de noviembre de 2023.

**Práctica 2: Snake.**

* **Diagrama de flujo**

Diagrama

Descripción generada automáticamente

* **Diseño**

En este caso el código trabaja mediante un arreglo que guarda la posición de cada una de las partes de la serpiente, iniciando con la cabeza en la posición 0 y añadiendo detrás de este el resto de las direcciones dentro de los leds. Cabe destacar que en dicho arreglo está directamente el número que representa la posición en los leds en lugar de las coordenadas X y Y por separado, por lo que para obtener estos valores es necesario usar módulo y división.

Para conocer si una manzana fue consumida o si alguna colisión con el resto del cuerpo sucedió basta con comparar los valores de la cabeza con estos, por lo que se necesita recorrer toda la serpiente tanto para buscar colisiones como para actualizar las nuevas posiciones de cada uno de estos.

Al iniciarse, el juego se encarga de limpiar por completo el tablero de leds, por lo que se puede ejecutar varias veces el código sin la necesidad de cerrarlo y volverlo a abrir, una vez hecho esto se muestra la serpiente completamente estática con 2 segmentos en la esquina superior de la pantalla y no es sino hasta que se interactúa con el dpad para que la víbora empiece a moverse y que aparezca la primera de las manzanas.

* **Requisitos del simulador**

Imagen que contiene Patrón de fondo

Descripción generada automáticamenteDadas las características de nuestro programa y las especificaciones dadas en un inicio es necesario que el tablero de leds tenga unas medidas divisibles entre 2, es decir, pueden ir desde 4 hasta 256 pero tanto la altura como el ancho deben ser múltiplos de 2, esto como ya se mencionó es a causa de que tanto las manzanas como la serpiente trabajan mediante bloques de 2x2, por lo que usar saltos o atributos de solo 1 tamaño en los leds complicaría enormemente las cosas, en este caso mostramos un tablero de 16x16.

Forma

Descripción generada automáticamenteA su vez es necesaria la creación de un dpad para controlar los movimientos de la serpiente, este ya por su parte tiene por defecto el añadido de ser controlado por las teclas WASD o las flechas, por ende, en caso de requerir mayor velocidad con las direcciones estas teclas pueden ser usadas, cabe destacar que gracias al código no importa si se quiera ir a una dirección invalida, como querer ir a la derecha mientras se avanza a la izquierda, dado que dicha entrada simplemente se ignorará.

A su vez poseemos dos versiones del código en caso de que quieran cosas extras, es decir, una versión trabaja únicamente con la serpiente y las manzanas mientras que otra añade paredes que puedan llegar a matar a la serpiente y se mueven cada vez que la serpiente crece.

Texto

Descripción generada automáticamenteFinalmente, nuestro sistema de puntajes es algo sencillo dado que solo se muestra al final de la ejecución, es decir, al chocarse consigo mismo, y da como resultado la cantidad de manzanas consumidas durante la partida, dicho puntaje aparece simplemente en la consola.

* **Configuraciones de cache (Primera parte)**

Para esta parte del reporte y en base al requisito mínimo dado en la actividad, las siguientes pruebas dentro del caché serán realizadas en un tablero de leds de tamaño 26x36, además de que como está especificado al final de la actividad, cada uno de los bloques o vías tendrán un tamaño de 2 palabras. Al finalizar las pruebas, nos dimos cuenta de que el caché totalmente asociativo ofrece mejores resultados.

**- Mapeo directo 4 líneas/bloques:**

Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Como se aprecia en las imágenes, al utilizar un caché de mapeo directo de 4 líneas obtenemos un hit rate de 94.32% = .9432, lo que nos deja con un misss rate de 0.0568.

**- Asociativa con 2 conjuntos y 2 vías (ways):**

Tabla

Descripción generada automáticamente

Gráfico

Descripción generada automáticamente

Para el caso de la caché asociativa con 2 conjuntos y dos vías, tenemos un peor resultado (El cual no es malo), con un hit rate de 92.1% = .921, dejando un miss rate de 0.079.

**Totalmente asociativa con 4 líneas/bloques:**

Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Finalmente, para el caso de la caché totalmente asociativa con 4 bloques, tenemos muy buenos resultados, con hit rate de 99.33% = .9933, dejando el miss rate con .0067. Sin duda la caché Totalmente Asociativa nos deja con mejores resultados.

* **Configuraciones de cache (Segunda parte)**

**- Mapeo directo 16 líneas/bloques:**

Tabla

Descripción generada automáticamenteInterfaz de usuario gráfica, Gráfico

Descripción generada automáticamente con confianza media

Para esta configuración de caché de mapeo directo con 16 bloques, tenemos un hit rate del 99.57% = .9957, dejando un miss rate de .0043.

**- Asociativa con 4 conjuntos y 4 vías (ways):**

Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica, Gráfico

Descripción generada automáticamente con confianza media

Para el caso de la caché asociativa con 4 conjuntos y 4 vías, tenemos un hit rate casi idéntico al anterior, con un 99.56% = .9956, dejando el miss rate en .0044.

**- Totalmente asociativa con 16 líneas/bloques:**

Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente

Finalmente, tenemos la caché totalmente asociativa, que nos deja con resultados similares, con un hit rate de 99.53% = .9953 y un miss rate de .0047.

Al finalizar estas configuraciones de caché, podemos darnos cuenta de que ofrecen resultados muy similares, sin embargo, por una muy pequeña ventaja, el caché de mapeo directo es el que nos ofreció un mejor hit rate.

* **Optimización de caché**

Para esta parte, se jugó un snake de 40x40

Configuración con 32 bloques por conjuntos y un solo índice, es decir, la caché totalmente asociativa. Tiene un hit rate de 99.78%

Gráfico

Descripción generada automáticamente

Configuración con 16 bloques por conjunto y dos índices. Tiene un hit rate de 99.79%

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Configuración con 8 bloques por conjunto y 4 índices. Tiene un hit rate de 99.82%

Interfaz de usuario gráfica, Gráfico

Descripción generada automáticamente

Configuración con 4 bloques por conjunto y 8 índices. Tiene un hit rate de 99.76%

Interfaz de usuario gráfica

Descripción generada automáticamente

Configuración con 2 bloques por conjunto y 16 índices. Tiene un hit rate de 99.69%

Interfaz de usuario gráfica, Gráfico

Descripción generada automáticamente

Configuración con un bloque por conjunto y 32 índices. Tiene un hit rate de 63.89%

Interfaz de usuario gráfica, Gráfico

Descripción generada automáticamente

**Resultados:**

Caché totalmente asociativa – Hit rate 99.78%

|  |  |
| --- | --- |
| Configuración | Hit Rate |
| 16 bloques por conjunto, 2 índices | 99.79% |
| 8 bloques por conjunto, 4 índices | 99.82% |
| 4 bloques por conjunto, 8 índices | 99.76% |
| 2 bloques por conjunto, 16 índices | 99.69% |
| 1 bloque por conjunto, 32 índices | 63.89% |

En base a estos resultados, nos dimos cuenta de que la configuración que nos ofrece un mejor hit rate es utilizar 8 bloques por conjunto con 4 índices, y la peor configuración que hay es solo utilizar un bloque por conjunto y 32 índices, siendo 2 configuraciones las que lograron superar a la opción original.

* **Conclusiones**

**Alberto Renteria Camacho:**

Esta práctica nos ayudó a aterrizar los temas de la caché que vimos en clase, además, nos dimos cuenta de la relación con la restricción con no utilizar memoria dinámica con los accesos a la memoria caché, ya que al utilizar memoria estática, pudimos ver de una mejor manera el comportamiento de la caché al ejecutar nuestro programa.

En relación con la elaboración de la práctica, no hubo gran complicación en realizar el snake, sin embargo, al inicio tuve algunas complicaciones en entender las formulas utilizadas para calcular las coordenadas de cada elemento de la serpiente, pero fuera de eso, lo demás se realizó sin mucha dificultad.

**Yael Alejandro Rodríguez Barreto:**

Si bien el desarrollo de la práctica se aleja de las cosas vistas en el anterior trabajo, sí se acerca en gran medida a todo lo repasado durante esta última parte del semestre, pasando por cosas como de nuevo trabajar con la herramienta Ripes, pero en esta ocasión mediante el lenguaje C, el manejo de las entradas y salidas de datos en este como lo son los leds y el pad, hasta cosas todavía más reciente como el funcionamiento de la caché en la lectura y escritura de datos dentro de la memoria.

En este caso y como era de esperarse gran parte del foco de la actividad se centra tanto en el análisis como lectura de información dentro del caché, haciendo uso de todo lo aprendido tanto para entender como este trabaja (Cosas como cache de mapeo directo, asociativa y por conjunto) y cuáles de estos métodos son mejores para la optimización de memoria.

Cabe destacar que, además de lo anterior, no solo basta con entender cómo es que estos trabajan, sino también de cómo se conforman para realizar un análisis completo y acorde a lo solicitado desde un inicio.