Un Modelo de Referencia para la Generación de Informes

José Ignacio Sbruzzi, *Padrón Nro. 97.452*
jose_sbruzzi@hotmail.com
Leandro Huemul Desuque, *Padrón Nro. 00.000*
dirección de e-mail
Nombre y Apellido de Autor, *Padrón Nro. 00.000*
dirección de e-mail
Nombre y Apellido de Autor, *Padrón Nro. 00.000*
dirección de e-mail

Grupo Nro. 0 - 2do. Cuatrimestre de 2006
66.20 Organización de Computadoras
Facultad de Ingeniería, Universidad de Buenos Aires

Resumen

Este artículo es un modelo que proporciona a los alumnos las instrucciones necesarias para preparar sus informes para la asignatura *66.20 Organización de Computadoras*. El informe deberá contener un resumen de no más de 150 palabras. La primera página del artículo deberá seguir el formato que se ilustra en el presente modelo y deberá contener el título, los nombres de los autores, sus números de padrón, sus direcciones de e-mail, número de grupo y el resumen. La primera página del informe no debe ser numerada.

1. Introducción

Se desarrolló un programa en C que simula el Juego de la Vida de Conway, habiéndose implementado una parte específica en assemly MIPS. El objetivo del presente trabajo fue principalmente familiarizarse con el entorno GXEmul y con LaTeX.

El Juego de la vida es... GXEmul es...

2. desarrollo

2.1. Documentación del código C

2.2. Documentación del código assembly

2.3. Dificultades

La construcción de una imagen en formato PBM presentó algunas dificultades ya que se usó PBM en vez de plain PBM, y en la descripción del formato PBM no había ejemplos de archivos PBM (sí los había de plain PBM). Un punto relativamente problemático del formato PBM es que cada pixel está asociado a un bit. Para evitar esta dificultad, se decidió que cada celda correspondería a un cuadrado de 8x8 pixeles, simplificando el fragmento del programa que guarda las imágenes.

3. Resultados

3.1. Medidas de tiempo de ejecución

3.2. Corridas de prueba

A continuación se detalla el resultado de las corridas de prueba de glider, pento y sapo para 10 operaciones en una matriz de 20 por 20 tal como pedido.

3.2.1. Glider

Glider es una configuración del autómata que se mueve por la pantalla sin destruirse: atravieza un ciclo de cuatro estados, al cabo de los cuales regresa a la configuración inicial, pero desplazada. En la figura **??** se muestra la configuración inicial del glider junto con 10 iteraciones. El glider se mueve, en cada ciclo, una celda hacia abajo y una celda a la derecha, como puede notarse en la figura **??**.

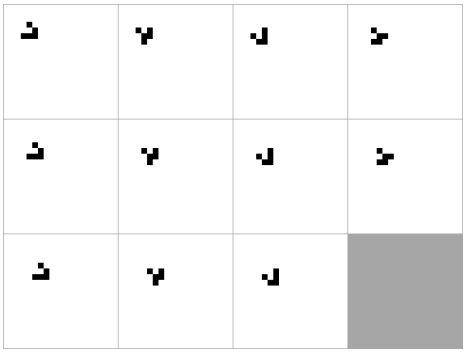

Figura 1: Desde la esquina izquierda superior: 1:Estado inicial descripto en el archivo glider. 2 a 11: Iteraciones 1 a 10

Figura 2: El movimiento del glider: se superponen la iteración 4 y 8

3.2.2. Pento

Pento es un patrón que se estabiliza luego de cierta cantidad de iteraciones. El resultado final varía según el tamaño de la matriz. Para una matriz de 20 por 20, se muestra la configuración inicial, las primeras 10 iteraciones y el resultado final estable, alcanzado en la iteración 60, en la figura **??**. En la imagen **??** puede verse la configuración final de Pento para una matriz de otro tamaño.

Figura 3: Desde la esquina superior izquierda: 1: estado inicial de Pento, 2 a 11: primeras 10 iteraciones, 12: estado final luego de 60 iteraciones. El tamaño de la matriz es 20x20

3.2.3. Sapo

El patrón Sapo, al igual que Glider, atravieza un ciclo, pero de 2 estados (no de 4), y no se desplaza. En la figura **??** se muestran el estado inicial y cuatro iteraciones de Sapo. No se muestran más iteraciones porque los estados se repiten constantemente.

Figura 4: La configuración final de Pento en una matriz de 30 por 30.

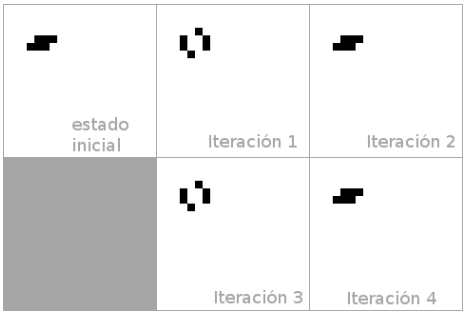

Figura 5: La configuración final de Pento en una matriz de 30 por 30.

4. Conclusiones

Se presentó un modelo para que los alumnos puedan tomar como referencia en la redacción de sus informes de trabajos prácticos.

Referencias

[1] Intel Technology & Research, "Hyper-Threading Technology," 2006, http://www.intel.com/technology/hyperthread/.

[2] J. L. Hennessy and D. A. Patterson, "Computer Architecture. A Quantitative Approach," 3ra Edición, Morgan Kaufmann Publishers, 2000.

[3] J. Larus and T. Ball, "Rewriting Executable Files to Mesure Program Behavior," Tech. Report 1083, Univ. of Wisconsin, 1992.