66:20 Organización de Computadoras
Trabajo práctico 1: conjunto de instrucciones MIPS

1. Objetivos

Familiarizarse con el conjunto de instrucciones MIPS32 y el concepto de ABI[1], escribiendo un programa portable que resuelva el problema descripto en la sección 6.

2. Alcance

Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

3. Requisitos

El trabajo deberá ser entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes.

Además, es necesario que el trabajo práctico incluya (entre otras cosas, ver sección 9), la presentación de los resultados obtenidos, explicando, cuando corresponda, con fundamentos reales, las causas o razones de cada resultado obtenido.

El informe deberá respetar el modelo de referencia que se encuentra en el grupo, y se valorarán aquellos escritos usando la herramienta TEX / LATEX.

4. Recursos

Usaremos el programa GXemul [1] para simular el entorno de desarrollo que utilizaremos en este y otros trabajos prácticos, una máquina MIPS corriendo una versión reciente del sistema operativo NetBSD [3]. GXemul se puede hacer correr bajo Windows, en el entorno Cygwin [2].

[1] Application Binary Interface

5. Introducción

El "Juego de la Vida" de Conway es un autómata celular, diseñado por el matemático británico John Conway en 1970 [5], y si bien su funcionamiento es simple, computacionalmente es equivalente a una máquina de Turing [6]. Se trata básicamente de una grilla en principio infinita, en cada una de cuyas celdas puede haber un organismo vivo (se dice que la celda está viva, o encendida) o no, en cuyo caso se dice que está muerta o apagada. Se llama "vecinos" de una celda a las ocho celdas adyacentes a ésta. Esta matriz evoluciona en el tiempo en pasos discretos, o estados, y las transiciones de las celdas se realizan de la siguiente manera:

- Si una celda tiene menos de dos o más de tres vecinos encendidos, su siguiente estado es apagado.
- Si una celda encendida tiene dos o tres vecinos encendidos, su siguiente estado es encendido.
- Si una celda apagada tiene exactamente tres vecinos encendidos, su siguiente estado es encendido.
- Todas las celdas se actualizan simultáneamente.

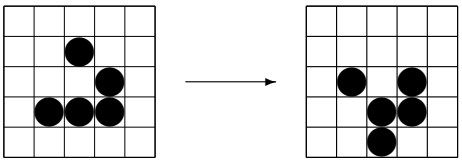

Figura 1: Ejemplo de transición entre estados

6. Programa

Se trata de una versión en lenguaje C del "Juego de la Vida". El programa recibirá por como argumentos tres números naturales, i, M y N, y el nombre de un archivo de texto con coordenadas en una matriz de MxN, y escribirá i archivos .PBM [7] representando i estados consecutivas del "Juego de la Vida" en una matriz de MxN, tomando como celdas 'vivas' iniciales las que están en las coordenadas del archivo de entrada (el primer estado a representar es el inicial). De haber errores, los mensajes de error deberán salir exclusivamente por `stderr`.

6.1. Condiciones de contorno

Dados los problemas que acarrearía tratar con una matriz infinita, se ha optado por darle un tamaño limitado. En este caso, para las filas y columnas de los 'bordes' de la matriz, hay dos maneras básicas de calcular los 'vecinos':

1. **Hipótesis del mundo rectangular:** Las posiciones que caerían fuera de la matriz se asumen apagadas. Sencillamente no se computan.

2. **Hipótesis del mundo toroidal:** La fila $M-1$ pasa a ser vecina de la fila 0, y la columna $N-1$ pasa a ser vecina de la columna 0. Entonces, el vecino superior de la celda $[0, j]$ es el $[M-1, j]$, el vecino izquierdo de la celda $[i, 0]$ es el $[i, N-1]$, y viceversa; de esta manera, nunca nos salimos de la matriz. Esta es la opción más interesante.

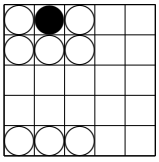

Figura 2: Ejemplo de mundo toroidal: la celda (0,1) y sus vecinos.

6.2. Comportamiento deseado

Primero, usamos la opción `-h` para ver el mensaje de ayuda:

```
$ conway -h
Uso:
  conway -h
  conway -V
  conway i M N inputfile [-o outputprefix]
Opciones:
  -h, --help     Imprime este mensaje.
  -V, --version  Da la versión del programa.
  -o  Prefijo de los archivos de salida.
Ejemplos:
 conway 10 20 20 glider -o estado
 Representa 10 iteraciones del Juego de la Vida en una matriz de 20x20,
 con un estado inicial tomado del archivo ‘‘glider’’.
 Los archivos de salida se llamarán estado_n.pbm.
 Si no se da un prefijo para los archivos de salida,
 el prefijo será el nombre del archivo de entrada.
```

Ahora usaremos el programa para generar una secuencia de estados del Juego de la Vida.

```
$ conway 5 10 10 glider -o estado
Leyendo estado inicial...
Grabando estado_1.pbm
Grabando estado_2.pbm
Grabando estado_3.pbm
Grabando estado_4.pbm
Grabando estado_5.pbm
Listo
```

El formato del archivo de entrada es de texto, con los números de fila y columna separados por espacios, a una celda por línea. Ejemplo: si el archivo `glider` representa un planeador en el centro de una grilla de $10x10$, se verá de la siguiente manera:

```
$ cat glider
5 3
5 4
5 5
3 4
4 5
```

El programa deberá retornar un error si las coordenadas de las celdas encendidas están fuera de la matriz ($[0..M-1], [0..N-1]$), o si el archivo no cumple con el formato.

7. Implementación

El programa a implementar deberá satisfacer algunos requerimientos mínimos, que detallamos a continuación.

7.1. Portabilidad

Pese a contenter fragmentos en assembler MIPS32, es necesario que la implementación desarrollada provea un grado mínimo de portabilidad.

Para satisfacer esto, el programa deberá proveer dos versiones de `conway()`, incluyendo la versión MIPS32, pero también una versión C, pensada para dar soporte genérico a aquellos entornos que carezcan de una versión más específica.

7.2. API

Gran parte del programa estará implementada en lenguaje C. Sin embargo, la función `vecinos()` estará implementada en assembler MIPS32, para proveer soporte específico en nuestra plataforma principal de desarrollo, NetBSD/pmax.

El programa en C deberá interpretar los argumentos de entrada, pedir la memoria necesaria para la matriz de estado A, popular la matriz con los

contenidos del archivo de entrada, y computar el siguiente estado para cada celda valiéndose de la siguiente función:

```
unsigned int vecinos(unsigned char *a,
                     unsigned int i, unsigned int j,
                     unsigned int M, unsigned int N);
```

Donde `a` es un puntero a la posición $[0, 0]$ de la matriz, i y j son la fila y la columna respectivamente del elemento cuyos vecinos queremos calcular, y M y N son las cantidades de filas y columnas de la matriz A. El valor de retorno de la función `vecinos` es la cantidad de celdas vecinas que están encendidas en el estado actual. Los elementos de A pueden representar una celda cada uno, aunque para reducir el uso de memoria podrían contener hasta ocho cada uno. Después de computar el siguiente estado para la matriz A, el programa deberá escribir un archivo en formato PBM [7] representando las celdas encendidas con color blanco y las apagadas con color negro [2].

7.3. ABI

El pasaje de parámetros entre el código C (`main()`, etc) y la rutina `vecinos()`, en assembler, deberá hacerse usando la ABI explicada en clase: los argumentos correspondientes a los registros `$a0-$a3` serán almacenados por el *callee*, siempre, en los 16 bytes dedicados de la sección "function call argument area" [4].

7.4. Algoritmo

El algoritmo a implementar es el algoritmo del Juego de la Vida de Conway[5], explicado en clase.

8. Proceso de Compilación

En este trabajo, el desarrollo se hará parte en C y parte en lenguaje Assembler. Los programas escritos serán compilados o ensamblados según el caso, y posteriormente enlazados, utilizando las herramientas de GNU disponibles en el sistema NetBSD utilizado. Como resultado del enlace, se genera la aplicación ejecutable.

9. Informe

El informe deberá incluir:

- Este enunciado;

[2] Si se utiliza sólo un pixel por celda, no se podrá apreciar el resultado a simple vista. Pruebe haciendo que una celda sea representada por grupos de por ejemplo 16x16 pixels.

- Documentación relevante al diseño e implementación del programa, incluyendo un diagrama del stack de la función `vecinos`;
- Corridas de prueba para diez iteraciones, en una matriz de $20x20$, de los archivos de entrada `glider`, `pento` y `sapo`, con los comentarios pertinentes;
- El código fuente completo, en dos formatos: digitalizado e impreso en papel.

10. Mejoras opcionales

- Una versión de terminal, que permita ver en tiempo real la evolución del sistema (y suprima los archivos de salida).
- Un editor de pantalla, de modo texto, a una celda por caracter. Esto permite experimentar con el programa, particularmente combinado con la versión de tiempo real.

11. Fecha de entrega

Primera entrega: Jueves 29 de Septiembre.
Revisión: Jueves 6 de Octubre.
Última fecha de entrega: Jueves 13 de Octubre.

Referencias

[1] GXemul, `http://gavare.se/gxemul/`.

[2] Installing the MIPS Environment over Cygwin on Windows, `http://faculty.cs.tamu.edu/bettati/Courses/410/2006B/Projects/gxemulcygwin.html`

[3] The NetBSD project, `http://www.netbsd.org/`.

[4] System V application binary interface, MIPS RISC processor supplement (third edition). Santa Cruz Operations, Inc.

[5] Juego de la Vida de Conway, `http://es.wikipedia.org/wiki/Juego_de_la_vida`.

[6] Máquina de Turing, implementada en el Juego de la Vida de Conway, `http://rendell-attic.org/gol/tm.htm`.

[7] `http://netpbm.sourceforge.net/doc/pbm.html`