Universidad Simón Bolívar
División de Ciencias Físicas y Matemáticas
Departamento de Computación y Tecnología de la Información
Organización del Computador (CI-3815)

Mario Quintero (13-11148)
Andrés Ignacio Torres (14-11082)

# Informe
# Proyecto 2

La interacción entre un computador y los distintos periféricos de entrada/salida (IO) que pueden estar conectados a él se regula a través del manejo de interrupciones, a través de subrutinas programadas para tal fin. El objetivo del presente proyecto es, a través de la estructuración y programación de un videojuego (un clon de **Breakout**, también conocido como **Brick Breaker** o **Arkanoid**), desarrollar un manejador de interrupciones adecuado, en particular de teclado y del timer del procesador.

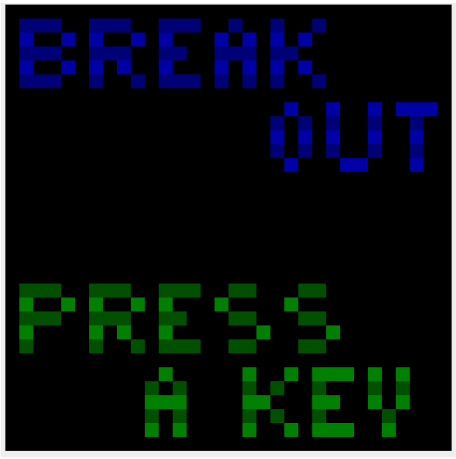


**Figura 1**. Pantalla de inicio del programa, en el Bitmap Display del simulador MARS

**Descripción del programa**

El programa, cuya lógica principal se encuentra en el archivo *breakout.asm*, consta de un juego con tres elementos: una barra en el extremo inferior, una pelota que se desplaza en el tablero y una serie de ladrillos en la parte superior. El objetivo del juego es hacer que la pelota destruya (choque con) los ladrillos verdes, hasta que no quede ninguno, pudiendo manipularla con la barra. El jugador pierde si la

pelota cae del borde inferior, e.g., si no logra alcanzarla con la barra, que se puede desplazar de izquierda a derecha.

**Controles del programa**

El control del programa se maneja mediante las interrupciones del teclado.

- **Q, q:** Salir del programa
- **<Barra espaciadora>**: Pausar o despausar el juego
- **A, a:** Desplazar la barra un bloque a la izquierda
- **D, d**: Desplazar la barra un bloque a la derecha
- **U, u:** Incrementar el valor T que regula la velocidad del juego
- **L, l:** Decrementar el valor T que regula la velocidad del juego

**Diseño de la solución**

Para implementar el juego, se genera en primer lugar una pantalla de "bienvenida" con el nombre del juego. Luego, al recibir una entrada, inicia el juego, en un ciclo principal que verifica si se cumple la condición de ganar (128 ladrillos destruidos), si el juego está pausado, si se ha recibido una interrupción del timer y si alguna tecla de control ha sido presionada, en ese orden, actuando según lo primero que se reciba y repitiendo el ciclo.

Al recibir una interrupción del timer, se salta al procedimiento para actualizar la posición de la pelota en pantalla, calculando a su vez si está en proximidad de la barra de juego o de algún bloque verde y actualizando los valores de velocidad en consecuencia, para el próximo ciclo.

Al ganar, se muestra una pantalla con un mensaje adecuado y se le permite al usuario volver a jugar. Al perder, el juego se reinicia tras presionar cualquier botón, siempre y cuando queden vidas disponibles.

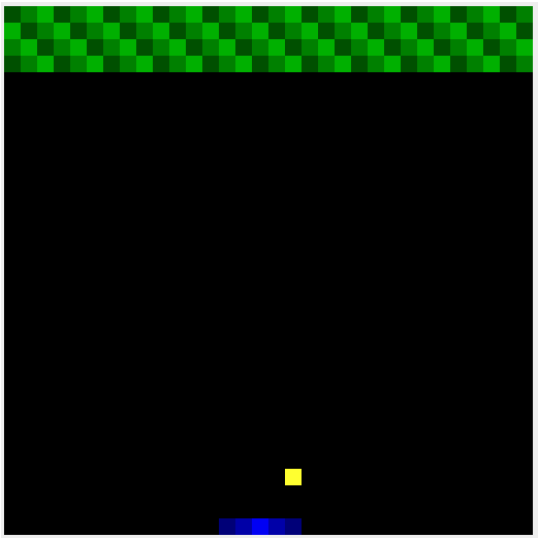

**Figura 2.** Captura de pantalla del juego en funcionamiento

**Detalles de implementación**

La implementación del juego se realizó en el simulador Mars para la arquitectura MIPS, con la modificación para incluir el timer del procesador. A efectos de corrida, se depende de tres elementos fundamentales de implementación:

- El Bitmap Display para simular un monitor y mostrar el juego en pantalla,
- El Keyboard and Display MMIO Simulator, para simular el dispositivo de entrada (teclado),
- El manejador de excepciones (*Exceptions.asm*) incorporado en el proyecto, para manejar la lógica de interrupciones por teclado y timer.

Para el aspecto visual del juego, se reservan 4096 bytes al inicio del segmento de datos, modificándose dinámicamente de acuerdo a lo ocurrido en el juego. Se utilizan 10 colores en todo el juego: el negro, un tono de amarillo, tres tonos de verde y cinco tonos de azul (de los cuales hay dos pares muy similares a la vista).

El juego hace uso de un generador de números pseudoaleatorios para la velocidad y dirección inicial de la pelota. De igual forma, hace uso del generador de tonos MIDI incorporado al simulador para generar sonido al iniciar el juego y al perder.

El parámetro T (cantidad de ciclos de reloj a esperar para el timer) fue **modificado** de lo especificado, debido a que fue muy difícil probar la jugabilidad del programa con el valor por defecto de 10. De esta manera, está incorporado con un valor inicial de 300, pudiéndose modificar con el incremento o decremento deseado según las teclas de control.

**Estado del proyecto**

El proyecto se encuentra en un estado operativo, con todas las funciones básicas incorporadas. Las rutinas de comprobación de estado del juego (ganado, perdido) funcionan, al igual que el manejador de choques con la barra y los ladrillos, y el verificador de bordes (para evitar que la pelota se “salga” del tablero de juegos).

**Instrucciones de corrida**

Para ejecutar satisfactoriamente el juego, se deben seguir los siguientes pasos en el archivo *breakout.asm* (no es necesario abrir *Exceptions.asm*, pero deben estar en la misma carpeta):

1) Abrir Tools > Keyboard and Display MMIO Simulator
2) En Tool Control, hacer click en Connect to MIPS
3) Abrir Tools > Bitmap Display
4) Ajustar los valores:
    OPCIÓN 1 (grande)
    4.1.1) Unit Width in Pixels: 16
    4.1.2) Unit Height in Pixels: 16

4.1.3) Display Width in Pixels: 512
4.1.4) Display Height in Pixels: 512

OPCIÓN 2
4.2.1) Unit Width in Pixels: 8
4.2.2) Unit Height in Pixels: 8
4.2.3) Display Width in Pixels: 256
4.2.4) Display Height in Pixels: 256

4.3) Base address for display: 0x10010000 (static data)

5) Aumentar el tamaño del Bitmap Display para verlo completo
6) En Tool Control, hacer click en Connect to MIPS
7) Ensamblar

**Bibliografía**


- Di Serio, A. *Notas del curso Organización del Computador (CI-3815)***.**
- Hennessy, J., Patterson, D. (2013). *Computer Organizacion and Design*.
- Tanenbaum, A. (2000). *Organización de computadoras: un enfoque estructurado.*