# Taller de Interrupciones

Organización del Computador 1

Verano 2018

En este taller vamos a implementar una versión simplificada del clásico juego Nibbles[1] (o Snake) en un procesador Intel 8086 sobre un sistema operativo tipo DOS.
En el archivo `NIBBLES.C` encontrarán una implementación parcial, que comprende la lógica de movimiento y de dibujado del gusanito.
El objetivo será utilizar los siguientes dispositivos presentes en el 8086 para controlar al gusanito y llevarlo a la vida:

- PIT (Programmable Interval Timer): Intel 8253/8254.
- Teclado AT: Intel 8042.

Ambos dispositivos generan interrupciones en los IRQ 0 y 1 respectivamente, que son recibidos por el master PIC (un Intel 8259A).

## 1. Preparación del entorno

Para compilar y ejecutar el programa utilizaremos el emulador DOSBox[2]. En la página de la materia se encuentra un archivo comprimido con el software necesario para editar y compilar el código fuente del taller.
Para preparar dicho entorno, realizar los siguientes pasos:

a) Descomprimir el archivo `paquete_dosbox.tar.gz`. Esto creará una carpeta `dosbox` con el Turbo C++ 3.3 (subcarpeta `TC`) y el Turbo Assembler (subcarpeta `TASM`).

b) Copiar a la carpeta `dosbox/TC/BIN` el archivo `NIBBLES.C`.

c) Para ejecutar DOSBox, desde una terminal posicionada en la carpeta `dosbox` ejecutar el comando `dosbox .`

d) Dentro de DOSBox, en la carpeta `TC\BIN` se encuentra el IDE Turbo C++. El mismo se ejecuta por medio del comando `TC`.

e) Turbo C++ nos permitirá editar, compilar y ejecutar nuestro programa.

## 2. Programación del PIC

El controlador Intel 8259A[3] permite atender por medio de una única línea de interrupción con el procesador las señales de hasta 8 dispositivos. A cada una de las señales se la denomina IRQ (Interrupt Request).
Además, este controlador se podía encadenar a otro Intel 8259A que funcionaba como *esclavo*, permitiendo hasta 16 IRQs (numerados del 0 al 15).
En DOS, cada IRQ genera una interrupción de software de tipo `0x08 + NumIRQ`, donde `NumIRQ` es el número de IRQ correspondiente.
El procedimiento de atención de una interrupción es el siguiente:

- Un dispositivo conectado a una línea determinada envía una señal al 8259A para indicar la interrupción.
- El 8259A envía al procesador el pedido de interrupción por la línea `INTR` y el número de interrupción por las líneas `D0` a `D7`.

[1] `https://www.youtube.com/watch?v=UmeKHtei0qo`
[2] `https://www.dosbox.com/`
[3] `https://en.wikipedia.org/wiki/Intel_8259`

- El procesador es interrumpido y busca en la IVT (Interrupt Vector Table) cuál es la posición de memoria a la que debe saltar, según el tipo de interrupción que se haya generado.
- El procesador guarda la dirección de retorno y las *flags* en la pila y salta a la dirección de memoria indicada en la IVT, comenzando a ejecutar el código allí presente.
- Al finalizar, la rutina de atención **debe** señalar el EOI (End of Interrupt), enviando el valor `0x20` a través el puerto `0x20`. Esto generará una señal `INTA` en el 8259A.
- La rutina de atención debe terminar con la instrucción `iret`, que se encargará de restaurar las *flags* y regresar el program counter al lugar desde donde fue interrmpido.

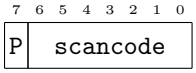
La IVT es un arreglo de punteros a las ISR (Interrupt Service Routine, o rutinas de atención) de las 256 interrupciones por software que soporta el procesador. Dicho arreglo se encuentra a partir de la posición de memoria `0x00000`.

# 3. Programación del PIT

Los controladores Intel 8253 u 8254[4] proveen un mecanismo para contar de manera periódica en 3 registros diferentes. Ya no forman parte de la arquitectura de los procesadores actuales, pero su funcionalidad sigue existiendo de manera emulada.
El PIT tiene un clock propio que funciona a aproximadamente 1,19 MHz (8253) o 10 MHz (8254). El primer registro está asociado al IRQ 0 (utilizado como timer), mientras que el tercero se utiliza para modular el PC Speaker con el que las viejas PC emitían sonidos.
Al iniciar la máquina, se configura para que emita una interrupción con una frecuencia de 18,2 Hz (la menor velocidad posible). Este valor es modificable ajustando ciertos registros.
En DOS, el PIT era utilizado (siempre a la velocidad mínima) para mantener actualizada la hora del sistema, por lo que los programas que quisieran utilizar el timer para ejecutarse periódicamente debían implementar una rutina de atención para la interrupción por software `0x1C`, la cual se invoca desde la que atiende al IRQ 0 (la `0x08`). Esto hace que no sea necesario enviar el EOI al PIC desde la `0x1C`, porque ya lo hace la `0x08` original.

# 4. Programación del teclado

El Intel 8042[5] es un controlador que cuenta con puertos de hardware de lectoescritura en las direcciones `0x60` y `0x64`, a través de los cuales el procesador puede enviar comandos, recibir datos, preguntar el estado y hablar directamente con el teclado.
Cada tecla que se presiona o se suelta genera un código (*rawcode*) que puede ser leído en el puerto `0x60` de a un byte a la vez. Las teclas se codifican de manera diferente dependiendo del tipo[6] (normales o extendidas). La codificación de una tecla siendo presionada se denomina *make code*, mientras que al soltarse se denomina *break code*.
Una tecla normal ocupa un byte y su *rawcode* se codifica de la siguiente manera:




El bit `P` representa si la tecla se encuentra presionada (make, 0) o no (break, 1), mientras que `scancode` indica los 7 bits correspondientes a la tecla que fue presionada.
En el caso de las teclas extendidas, tanto los *make codes* como los *break codes* ocupan más de un byte. Esto implica que para reconstruir la tecla presionada o soltada es necesario realizar varias lecturas sucesivas al puerto `0x60`, debiendo bufferearse los códigos leídos en la memoria de la máquina. El indicador para reconocer una tecla extendida es el *rawcode* `0xE0`. El siguiente byte leído es interpretado como una tecla normal, con su correspondiente indicador de *make* o *break* en el bit más significativo[7].
El controlador del teclado genera además una interrupción en el IRQ 1 por cada byte generado que pueda ser leído en el puerto `0x60`.

[4] `https://en.wikipedia.org/wiki/Intel_8253`
[5] `https://en.wikipedia.org/wiki/Intel_MCS-48`
[6] Ver lista completa de códigos en `http://stanislavs.org/helppc/make_codes.html`
[7] Hay teclas que se codifican con más de 2 bytes, pero quedan fuera del alcance de este taller.

# 5. Consigna

a) Almacenar en las variables globales `orig_teclado` y `orig_timer` las posiciones de memoria de las ISRs correspondientes a las interrupciones de tipo `TECLADO` (`0x09`) y `TIMER` (`0x1C`).

**Ayuda**: utilizar la función `get_isr` para leer la IVT.

b) Modificar la tabla de interrupciones para que se invoque a la función `rutina_teclado` cada vez que ocurra una interrupción de tipo `TECLADO` (`0x09`). Utilizar la rutina de atención del teclado para terminar el programa al presionar la tecla `Esc`.

**Ayuda**: utilizar la función `set_isr` para modificar la IVT.

**Importante**: si no restauran la rutina original al terminar el programa, no tendrán más control del teclado y deberán cerrar DOSBox.

c) Modificar la tabla de interrupciones para que se invoque a la función `rutina_reloj` cada vez que ocurra una interrupción de tipo `TIMER` (`0x1C`). Recordar restaurar la rutina original al terminar el programa.

d) Utilizar la rutina de atención del reloj para avanzar el gusanito un paso cada aproximadamente 275 ms.

e) Utilizar la rutina de atención del teclado para cambiar la dirección del gusanito según la flecha presionada (ver variable `direccion_gusanito`).

f) En este punto utilizaremos el comando `TCC` con la opción `-S` para generar código assembler anotado a partir del archivo `NIBBLES.C`.

   i. Compilar a assembler el código terminado.

   ii. Modificar los prototipos de las funciones `rutina_timer` y `rutina_teclado`, eliminando la keyword `interrupt`. Volver a compilar a assembler.

   iii. Comparar el código generado en ambos casos. ¿Qué instrucciones cambiaron? ¿Sigue funcionando la versión sin la keyword `interrupt`? ¿Por qué?