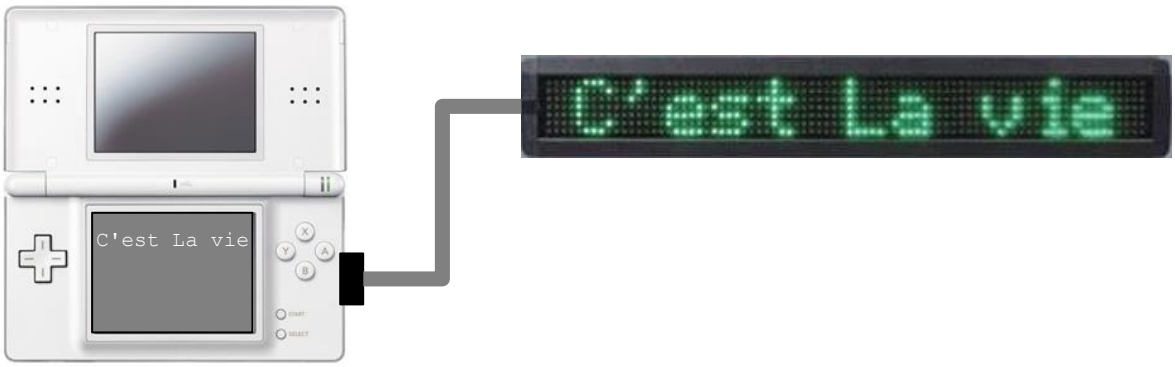
Problema 15: Display de LEDs (Ex. 2ª Conv. 2014-15)

Se propone controlar un display de LEDs con la NDS. El display tiene 7 puntos de altura y 100 puntos de anchura, y los puntos a mostrar se insertan por la columna de la derecha, con un efecto de desplazamiento de todo el contenido hacia la izquierda, columna a columna (píxel a píxel):



El dispositivo se conectará a la NDS mediante el puerto de cartuchos de juegos GBA ROM, y se controlará con un único registro de Entrada/Salida de 16 bits en la dirección simbólica REG_DISP, aunque sólo los 8 bits de menor peso tendrán una funcionalidad específica:

- DATA (bits 6..0): deben contener el valor (0 → apagado, 1 → encendido) de los 7 puntos de una columna a introducir por la derecha, donde el bit 0 corresponde al punto superior y el resto de bits a los sucesivos puntos inferiores,
- STROBE (bit 7): se debe poner a 1 y después a 0 (mín. 2 ms entre cambios) para desplazar el contenido del display una columna a la izquierda, e insertar el estado de los puntos de la columna de más a la derecha según los bits de DATA.

La inserción de las columnas de puntos se debe realizar a una frecuencia de 18 Hercios, lo cual equivaldrá a una velocidad de 3 caracteres por segundo, dado que cada carácter está definido por 7 filas y 6 columnas de puntos (píxeles).

El programa principal debe realizar algunas tareas independientes, además de controlar el carácter a visualizar en todo momento, que se obtendrá de un *string* predeterminado (fijado dentro del programa), almacenado en un vector de códigos ASCII acabados con un carácter centinela '\0'.

Este *string*, que deberá tener al menos 17 caracteres, se tiene que visualizar indefinidamente, de modo que, cuando se llegue al carácter centinela, se empezará de nuevo por el primer carácter. El *string* también se debe mostrar por una pantalla de la NDS de una única vez, sin realizar ningún tipo de desplazamiento horizontal.

Se dispone de las siguientes rutinas, ya implementadas:

Rutina	*Descripción*
`inicializaciones()`	Inicializa el *Hardware* (pantalla, interrupciones, *timer*)
`tareas_independientes()`	Realiza tareas independientes a la visualización del *string* (ej. obtener la temperatura ambiente)
`swiWaitForVBlank()`	Espera hasta el próximo retroceso vertical
`printf(char *format,...)`	Escribe un mensaje por la pantalla inferior de la NDS

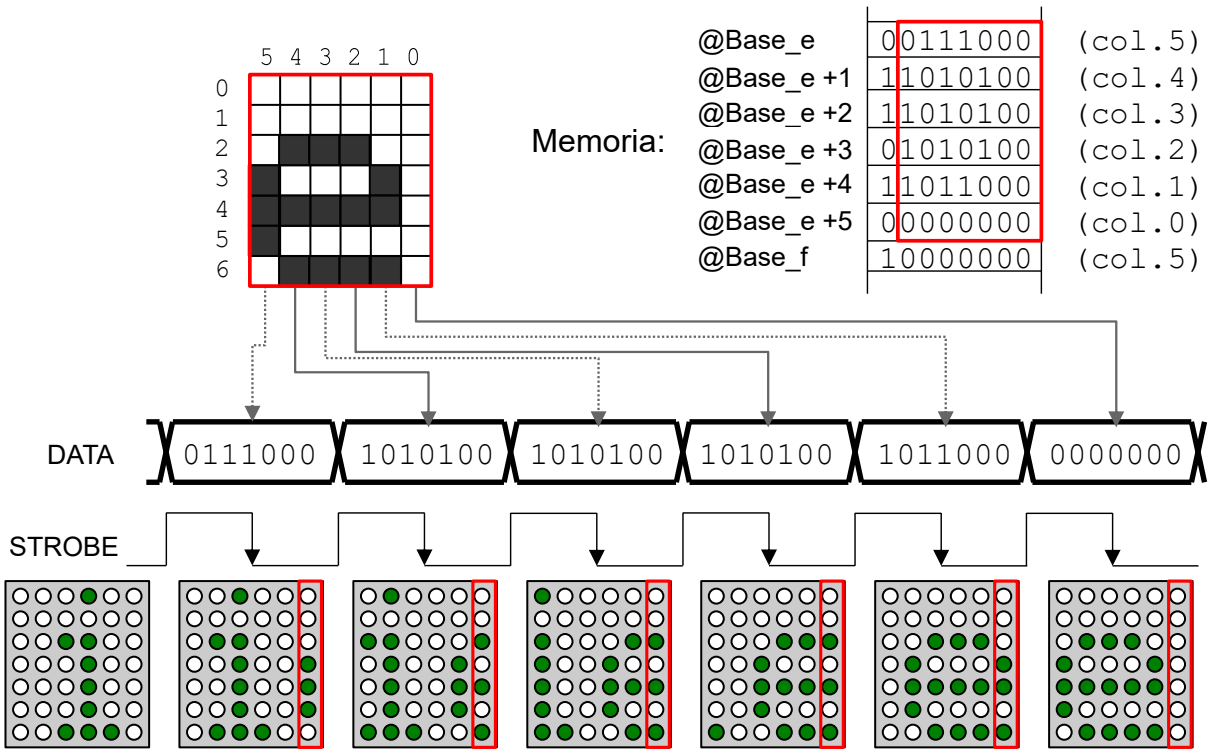
Para poder transferir las columnas de puntos de cada carácter de forma concurrente con el programa principal, se debe utilizar la RSI del *timer* 0, que se programará (por la rutina de inicializaciones) para realizar 36 interrupciones por segundo.

Esta RSI accederá al carácter actual a través de una variable global `currentChar`, y gestionará la columna actual a visualizar mediante otra variable global `num_col`. Además, tiene que llamar a una rutina auxiliar que hay que implementar, la cual retornará el estado de los puntos de una columna de un carácter, a partir del código ASCII del carácter y del número de columna actual que se pasarán por parámetro:

```
char obtener_puntos(char caracter, char num_columna);
```

Además de enviar el estado de los puntos, la RSI debe generar la señal de *strobe*, es decir, poner el bit 7 a 1 y, después de un cierto tiempo, poner el bit 7 a 0, para indicar al display que debe introducir una nueva columna.

A continuación se muestra la secuencia de introducción de la letra 'e' (detrás de una 'i'), así como el contenido en memoria que determina el estado de los puntos y el desplazamiento de las 6 columnas de puntos por la derecha (empezando por la columna de más a la izquierda):



Hay que observar que cada carácter se almacenará a partir de una posición de memoria específica y ocupará 6 bytes consecutivos correspondientes al estado de sus 6 columnas, ocupando dicho estado los 7 bits de menor peso de cada byte, mientras que el bit de más peso del byte puede contener cualquier valor. Los datos de todos los caracteres se almacenarán consecutivamente a partir de una dirección base de memoria, cuyo nombre simbólico será `base_ASCII`. El primer carácter almacenado corresponderá al código ASCII 32 (espacio en blanco).

Se pide:

Programa principal en C, RSI del *timer* 0 y rutina `obtener_puntos()` en ensamblador.