Arquitectura de la Nintendo Game Boy

José Miguel Moreno López, Marco Pérez Martínez, y Carlos Rascón Herrero

Estudiantes del Grado en Ingeniería Informática de Servicios y Aplicaciones

Campus María Zambrano, Plaza de la Universidad, 1. 40005 Segovia

josemiguel.moreno@alumnos.uva.es
marco.perez@alumnos.uva.es
carlos.rascon@alumnos.uva.es

RESUMEN

La Game Boy es sin duda una de las consolas más exitosas de la historia. Con varios modelos lanzados y más de 1.000 juegos publicados y licenciados, la máquina de Nintendo logró entretener pese a un hardware limitado para su época, lo que redujo su coste de producción notablemente y consiguió que tuviese una gran autonomía, algo que sus competidores no lograron.

Su sencilla arquitectura nos permite conocer cómo son capaces de ejecutar instrucciones desde las máquinas más básicas hasta los ordenadores actuales. A continuación se exploran todos los componentes que hacen funcionar a la Game Boy, desde su procesador hasta la PPU, pasando por la RAM, el controlador de sonido y los dispositivos de entrada y salida.

1. INTRODUCCIÓN

En este artículo se explica la arquitectura de la Game Boy, consola de Nintendo muy popular desde mediados de los 90 hasta la primera década del siglo XXI. En su largo periodo de vida, fue una de las consolas más vendidas de la historia, con un catálogo de juegos amplio y variado que incluía grandes títulos como Tetris o Super Mario Land.

Figura 1. La Game Boy original

De entre las distintas versiones y generaciones que surgieron, en este artículo nos centraremos en la original, conocida como **DMG-01** (*Dot Matrix Game*).

Nuestro objetivo principal es introducir cómo funcionan las máquinas a bajo nivel a través de la Game Boy ya que, pese a sus peculiaridades, su arquitectura es fácil de entender y nos da una vista general de los cimientos de la computación actual.

2. HISTORIA

La Game Boy se puede considerar como uno de los mayores éxitos en toda la historia del videojuego, ya que revolucionó la industria y marcó a toda una generación.

Nintendo se decidió por una máquina monocroma en contraposición a las portátiles de color (Sega Game Gear o Atari Lynx), y esto le permitió una **gran autonomía** de la batería con tan solo cuatro pilas A4.

Su creador, Gunpei Yokoi, dirigió un equipo que se basó fuertemente en las máquinas *Game & Watch* de los años ochenta para crear un sistema de videojuegos portátil que juntara estos dispositivos con la consola de sobremesa **Nintendo Entertainment System** (NES).

Figura 2. Juego *Game & Watch: Parachute*

El modelo original de la consola portátil por excelencia de Nintendo apareció en Japón el 21 de abril de **1989**. Sus creadores (el mismo equipo que desarrolló las *Game & Watch*) apostaron fuerte por ella, pero no imaginaban que años después el éxito de la máquina superaría todas las expectativas y se convertiría en un fenómeno a escala planetaria, inspirando una nueva generación de consolas e influyendo y cambiando por completo la industria del videojuego.

3. ARQUITECTURA

La Game Boy, a diferencia de la mayoría de máquinas de su época, consta de un *System On a Chip* que integra la CPU (*Central Processing Unit*), la PPU (*Pixel Processing Unit*) y otros componentes necesarios para su funcionamiento.

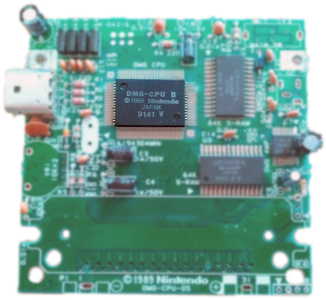

Figura 3. *SoC* de la Game Boy original [1]

3.1. CPU

El procesador utilizado por la Game Boy es el SHARP LR35902, un híbrido entre el Intel 8080 y el Zilog Z80.

El **Intel 8080**, lanzado en 1974, fue el primer microprocesador de 8 bits que tuvo éxito entre la comunidad de programadores, teniendo como principal exponente al MITS Altair 8800.

Dos años después y con la intención de hacerse con el mercado, Zilog lanzó el Z80, que atrajo a los programadores de la época al ser totalmente compatible con la arquitectura del Intel 8080 y añadir nuevas e interesantes funcionalidades, como un set de instrucciones extendido. El Z80 tuvo tanto éxito que es posible encontrarlo en gran cantidad de máquinas de principios de los ochenta, como el Osborne 1, el ZX Spectrum o el Commodore 128.

Podría decirse, por tanto, que el **Zilog Z80** es una versión mejorada del Intel 8080. Entonces, ¿qué es el **LR35902**?

El microprocesador de SHARP es el resultado de tomar el Zilog Z80 y quitarle la mayoría de instrucciones del set propio y otras pocas heredadas del Intel 8080, mientras se añaden otras nuevas.

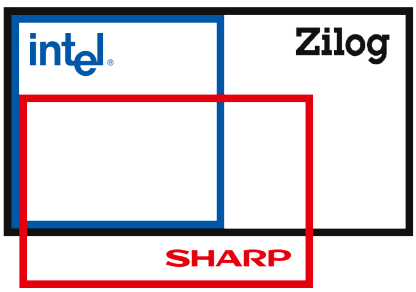

Figura 4. Comparación de procesadores

A continuación se incluye una tabla con las funciones en ensamblador soportadas por cada uno de los procesadores mencionados:

	Intel	Zilog	SHARP
Registros (`AF`, `BC`, `DE`, `HL`)	•	•	•
Segundos registros		•	
Flags `S` y `P`	•	•	
Stack (`push`, `pop`)	•	•	•
Cargar (`ld`)	•	•	•
Aritmética (`add`, `sub`)	•	•	•
Computo con negativos (`neg`)		•	
Bucles para inc. y decremento		•	
Lógica (`and`, `or`, `xor`)	•	•	•
Rotación básica de bits	•	•	•
Manejo de bit (`rlc`, `sla`, `bit`)		•	•
Control flujo (`jp`, `call`, `ret`)	•	•	•
Control de retorno (`retn`)		•	
Saltos relativos (`jr`)		•	•
Interrupciones (HW y SW)	•	•	•
Retorno desde interrup. (`reti*`)		•	•
Bus de E/S (`in`, `out`)	•	•	
Pre y post-decrem. (`hl+`, `hl-`)			•
Página cero (`$ff00`)			•
Intercambio de *nibbles* (`swap`)			•
Ahorro de energía (`stop`)			•

Figura 5. Tabla de comparación de funciones

3.1.1. Características del LR35902

El procesador de la Game Boy se basa en una arquitectura de **8 bits** (esto significa que es capaz de procesar 8 bits en cada ciclo de reloj) con un direccionamiento de memoria de 16 bits (en total puede direccionar 65.536 direcciones de 1 byte cada una, que justo son 8 bits), es decir, 64 KB de memoria.

Este chip corre a 4,19 MHz, aunque en realidad el juego se ejecuta a **1 MHz** al verse el procesador limitado por los chips de memoria.

Este microprocesador hace uso de principalmente **10 registros** (la memoria interna de la CPU), a saber:

- **A** (8 bits): El acumulador, el único capaz de realizar operaciones aritméticas.
- **F** (8 bits): *Flags*, indica el estado del procesador.
- **B, C, D, E, H, L** (8 bits cada uno): Para uso general.
- **SP** (16 bits): El *Stack Pointer* o puntero de la pila, indica la dirección en memoria del elemento superior de la misma.
- **PC** (16 bits): *Program Counter*, indica la dirección en memoria a ejecutar.

Además, la Game Boy permite **juntar los registros en pares** predeterminados para almacenar datos de 16 bits, lo cual es útil al trabajar con direcciones de memoria. Otro caso especial es (HL), que derreferencia la dirección de memoria almacenada como valor en HL y obtiene su valor (por tanto, apunta a 8 bits).

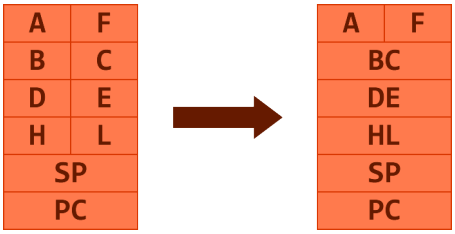

Figura 6. Registros de la Game Boy

El registro F es de solo lectura y toma unos valores determinados en función del **estado de la CPU**:

- ***Zero Flag* (Z)**: Se activa cuando el resultado de la última operación es 0.
- ***Subtraction Flag* (S)**: Se activa cuando la última operación es una resta.
- ***Carry Flag* (C)**: Se activa cuando se da un desbordamiento (*overflow*) en la última operación ejecutada.
- ***Half Carry Flag* (H)**: Similar al anterior, se activa cuando el *overflow* ocurre en los 4 bits menos significativos del resultado.

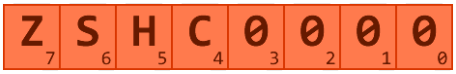

Figura 7. Desglose del byte de *flags*

3.2. Memoria

La Game Boy tiene un espacio de direccionamiento de 64K. Se divide en:

- **[0000-7FFF] ROM**: en la que se carga el juego desde el cartucho. Se divide en dos partes, ROM0 y ROM1. La primera es estática y direcciona hacia el código base del juego. La segunda es un tanto especial, puesto que puede tener muchos bancos de memoria que se pueden intercambiar mediante la técnica de *Memory Banking*, que se explica más adelante.
    - **[0000-00FF] BOOT ROM o BIOS**: En estas direcciones se vuelca una pequeña ROM al encender la consola que se encarga de iniciar el sistema. Principalmente, limpia la RAM, escribe el logotipo de Nintendo en la pantalla y lo desliza hasta el centro de la pantalla, reproduciendo el característico sonido de la Game Boy.
    - **[0100-014F] *Header***: Apunta a la cabecera del cartucho, que contiene los metadatos del juego (nombre, requisitos de versión, etc.).
- **[8000-9FFF] *Video* RAM**: esta división de memoria contiene todos los datos referentes a los fondos y los *sprites*. Cambia dependiendo del cartucho que insertes en la Game Boy, puesto que cada juego tiene sus propios *sprites* y fondos, y puede cambiar en tiempo de ejecución.
- **[A000-BFFF] *External* RAM**: es opcional y apunta al chip de RAM que contenga el cartucho.
- **[C000-DFFF] RAM de trabajo**: es la RAM interna de la consola, de 8K y que puede ser leída y escrita por la CPU.
- **[E000-FDFF] Zona espejo**: debido a la circuitería de la Game Boy, estas direcciones apuntan hacia la RAM de trabajo. No es conveniente su uso.
- **[FE00-FFFF]**: contiene una parte que tiene una subdivisión interna:
    - **[FE00-FE9F] OAM (*Object Attribute Memory*) RAM**: almacena las propiedades de los *sprites* (posición, transparencia, etc.). Más adelante se explica qué son estas entidades.
    - **[FEA0-FEFF]**: es un espacio reservado no usable por el programador.
    - **[FF00-FF7F] I/O (*Input/Output*)**: se encarga de controlar la entrada y salida de datos, como, por ejemplo, los botones de la cruceta, mediante la técnica E/S mapeada.
    - **[FF80-FFFE] *High* RAM**: también denominada página cero, es una RAM de acceso muy rápido, que paradójicamente se encuentra en la zona más alta de la memoria.
    - **[FFFF]**: hay una parte de la memoria que se encarga de controlar las interrupciones del sistema.

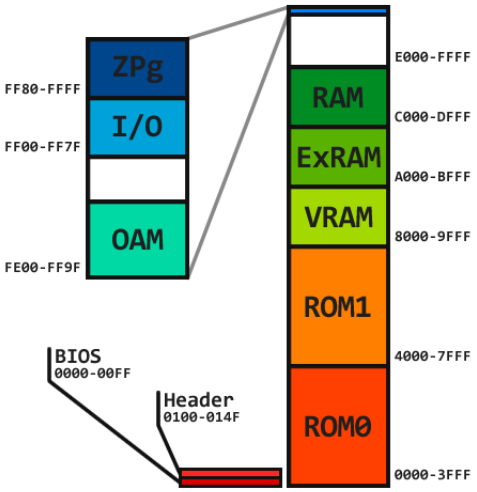

Figura 8. Mapa de memoria de la Game Boy

3.2.1. El sistema de licencias de la Game Boy

Encendida la consola sin introducir ningún cartucho, se visualiza un rectángulo negro en la pantalla en vez del logotipo de Nintendo.

Esto se debe al sistema de licencias implantando, que requería a los fabricantes autorizados por Nintendo tener su logotipo en la memoria del cartucho. Si la consola no detecta unos bytes determinados en la cabecera de la ROM, se abortaba el proceso de carga.

Este sistema de licencias fue poco eficaz, puesto que algunos fabricantes no autorizados consiguieron los bytes necesarios para renderizar el logo de Nintendo, por lo que consiguieron vender juegos piratas que funcionaban perfectamente en la consola, sin permiso de Nintendo.

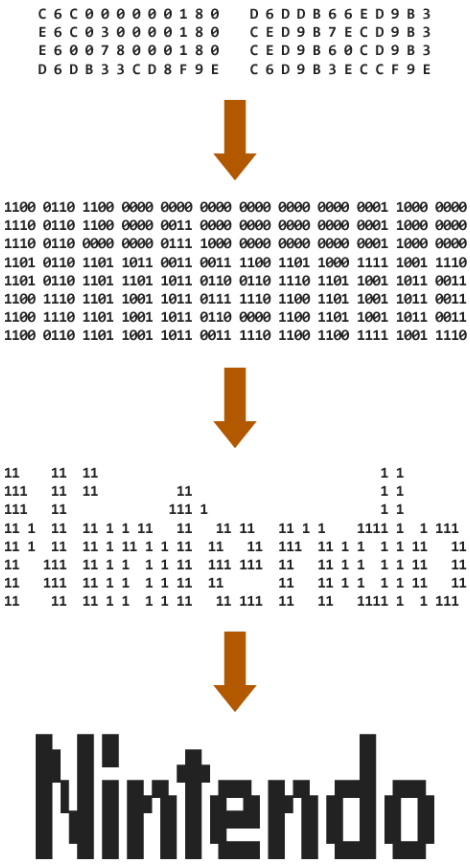

Figura 9. Renderización del logotipo de Nintendo

Otro dato curioso es que la Game Boy no quita el logotipo de la pantalla antes de pasar el control al cartucho (limpia la RAM pero **no la vRAM**), por lo que algunos juegos lograron introducir curiosas animaciones modificando los bytes existentes en memoria.

Figura 10. Ejemplo de *logo hacks* en la Game Boy

3.2.2. Cartuchos y *memory banking*

Al carecer de sistema operativo, los juegos de la Game Boy se ejecutan directamente desde el cartucho, que contiene las **instrucciones en ensamblador** listas para ser leídas por el procesador de la consola. Al encender la Game Boy y tras realizar la secuencia de inicio, el procesador coloca el contador de programa en la dirección `0x0100` (que se encuentra en el área de ROM0) y desde ahí sigue la ejecución.

Dado que esta área tiene un tamaño de **32 KB**, ¿quiere decir que no podemos hacer juegos con un tamaño superior?

Algunos juegos como el Tetris no necesitan más espacio, pero otros, como Pokémon Crystal, necesitan más para poder leer todo su contenido. Esto se puede hacer porque algunos cartuchos tienen un chip llamado MBC (*Memory Bank Controller*) que es capaz de **intercambiar páginas de memoria durante la ejecución**. Esto se hace en la segunda parte de la ROM (paginada como `ROM1`), en la que se intercambian los diferentes bancos de memoria para poder ejecutar juegos de gran tamaño.

Figura 11. Pokémon Crystal alcanza 1 MB de tamaño

3.3. Gráficos

Este apartado se centra principalmente en la Unidad de Procesamiento de Píxeles o PPU *(Pixel Processing Unit*), que se encarga de renderizar el juego en la pantalla.

La Game Boy dispone de una pantalla de **160x144 píxeles** a 4 colores (escala de grises, para ser más exactos). Dado que la consola solo dispone de 8 KB de VRAM, no es posible almacenar todos los píxeles de un fotograma y cambiarlos con suficiente rapidez. Aquí se introduce el concepto de *tiles*.

3.3.1. *Tiles*

Para sobrepasar esta limitación de memoria surgen los *tiles*: **bloques de 8x8 píxeles indivisibles** con los que se pueden elaborar gráficos. Esta imposición implica no poder pintar píxeles en una posición determinada de la pantalla, que puede apreciarse aplicando una cuadrícula sobre cualquier fotograma de cualquier juego.

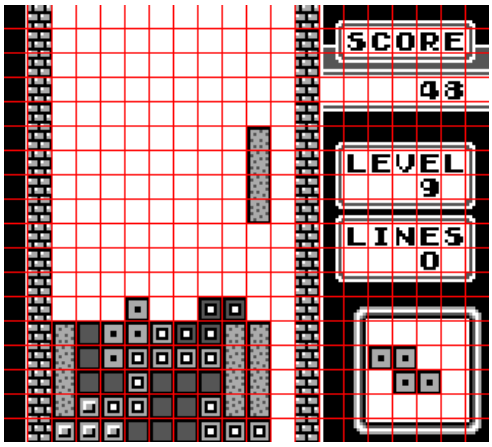

Figura 12. *Tiles* usados por Tetris

Los *tiles* pueden dedicarse para mapas (el fondo del juego) o para *sprites* (objetos que se verán más adelante). En total, la Game Boy admite hasta 384 *tiles*, de los cuales 128 son reservados para mapas y otros 128 para *sprites*, dejando el resto compartidos para ambos usos.

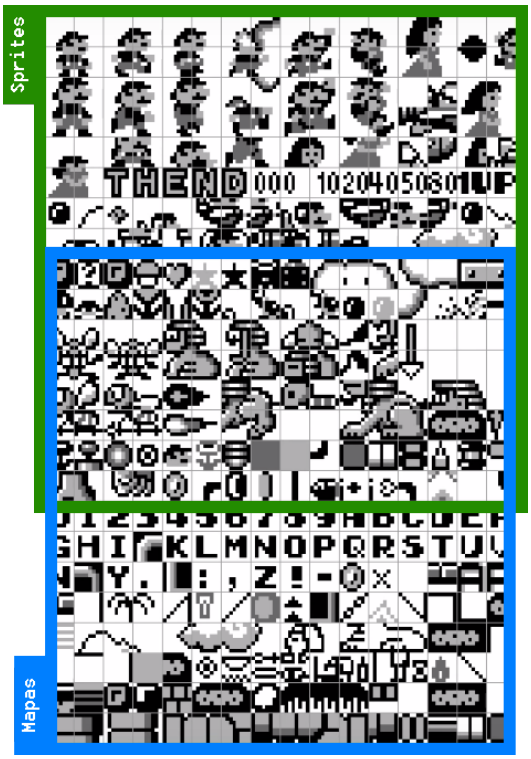

Figura 13. Representación gráfica de la VRAM

3.3.2. Mapas

La PPU divide los elementos a mostrar en la pantalla en tres capas: el fondo, los *sprites* y la ventana. Los mapas son el conjunto de *tiles* que se renderizan en la capa del fondo.

Estos mapas pueden tener un tamaño máximo de 32x32 tiles, lo que se traduce a **256x256 píxeles**. Dado que la resolución de la pantalla es menor que el tamaño del mapa, la PPU hace uso de dos variables (**SCX** y **SCY**) que establecen el desplazamiento en píxeles (horizontal y vertical, respectivamente) del área a renderizar.

Hay que tener en cuenta que **los mapas son cíclicos**, es decir, cuando acaba por la derecha vuelve a empezar por la izquierda, y lo mismo en el eje vertical. Esto puede utilizarse para crear niveles infinitos, como Super Mario Land, que hace uso de esta técnica para renderizar el mapa unos píxeles más adelante de donde se encuentra la cámara, creando la sensación de un mapa de gran tamaño.

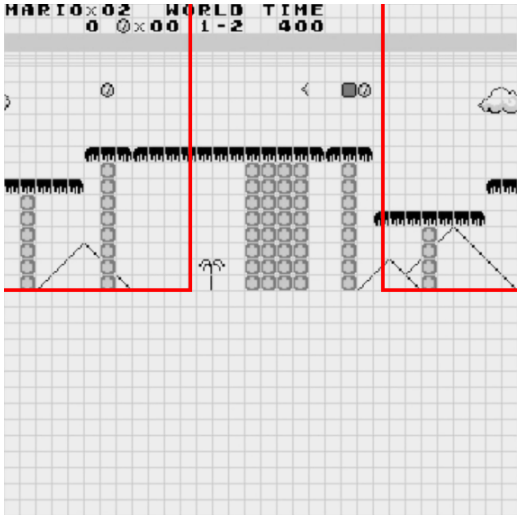

Figura 14. *Horizontal Scrolling* en Super Mario Land

3.3.3. *Sprites*

Los *sprites* (oficialmente llamados **objetos** por Nintendo) son entidades que se renderizan encima del fondo y que pueden posicionarse **en cualquier lugar de la pantalla**. Es por eso que son usados para representar al jugador, enemigos u objetos **móviles**.

Un *sprite* toma los píxeles de un *tile*, con la sutil diferencia de que este puede tener **transparencia** al sacrificar uno de 4 colores de la paleta (el primero). Este efecto se le indica a la PPU a través de una de las propiedades de los *sprites*: su prioridad.

Otra propiedad interesante de estas entidades es que pueden ser renderizadas **en espejo** activando el bit *X-Flip*, que lo voltea verticalmente, y/o el bit *Y-Flip*. De esta forma, basta con un solo *sprite* para representar a un enemigo andando de izquierda a derecha y viceversa. Todas estas propiedades se almacenan en la *Object Attribute Memory* (OAM).

La Game Boy puede contener un máximo de 40 *sprites* en memoria, aunque existe otra limitación que impide renderizar más de **10 *sprites* en una misma línea**.

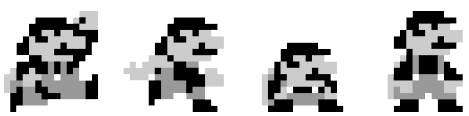

Figura 15. Varios *sprites* de Mario

3.3.4. Ventanas

Por encima de todas las capas del juego se puede incluir, de forma **opcional**, una ventana. Esta no se ve afectada por el desplazamiento de la cámara y **no puede tener transparencia**, por lo que se suele posicionar en la parte superior o lateral de la pantalla, dejando oculto el resto de la ventana al estar fuera del área de renderización.

Solo puede haber una ventana a la vez y suele utilizarse para mostrar la puntuación y el número de vidas al jugador.

3.4. Audio

El controlador de sonido de la Game Boy es capaz de generar audio en **estéreo**, aunque éste solo puede ser apreciado a través de la salida de auriculares, ya que la consola solo tiene un altavoz.

La Game Boy dispone de **4 voces o canales**: dos para pulsos, uno para ondas personalizadas y el último para efectos de sonido (ruido). Cada canal ocupa en memoria 5 bytes (un byte por registro), en el que cada registro define una propiedad del sonido, a saber: control, frecuencia, volumen, longitud y *sweep*.

El primer canal de pulso es el único que soporta el registro *sweep*, que permite hacer efectos de cambio de tono. El tercer canal tiene a su vez 16 bits adicionales para definir las características de la onda que tiene asociada.

Todos los canales pueden activarse o desactivarse manualmente con el **bit de *toggle*** y automáticamente después de un número de ciclos a través de la propiedad *length* (longitud).

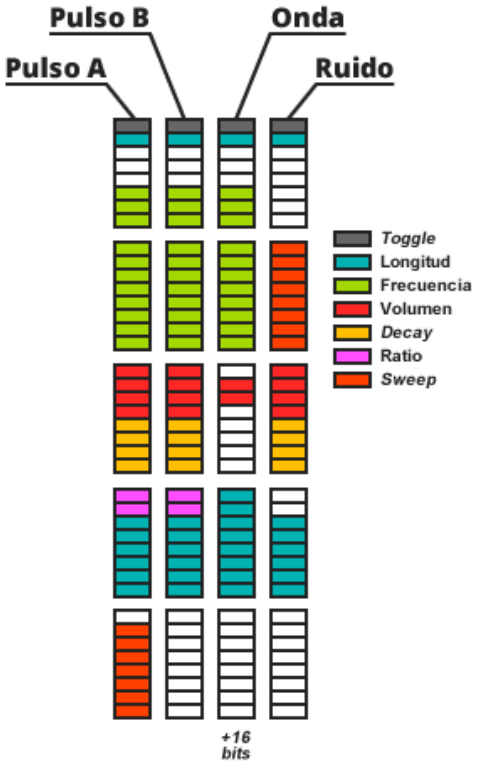

Figura 16. Mapa de registros de audio

3.5. Entrada y salida

En esta sección se describen las entradas y salidas de la Game Boy, principalmente el teclado y el *link cable*.

3.5.1. Teclado

El teclado consta de 8 botones, de los cuales cuatro son para los cursores de dirección (el *gamepad* o cruceta).

En un principio cualquiera pensaría que al haber ocho botones se necesitarían 8 bits (un byte) para representar el estado del teclado. Sin embargo, Nintendo consigue el mismo resultado con **solo 6 bits**.

La información de estado de estos botones se almacena en la dirección de memoria `FF00`. Para poder leer el estado de la cruceta los bits 5 y 4 de este byte deben tener el valor `01` en binario (respectivamente).

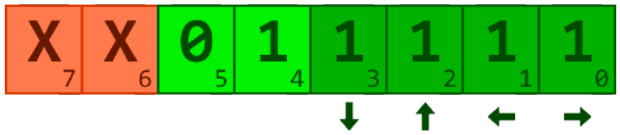

Figura 17. Botones del *gamepad*

De esta forma, los cuatro bits menos significativos indicarán el estado de las fechas abajo, arriba, izquierda y derecha, en ese orden. La única peculiaridad es que el `1` se interpreta como no pulsado y **el `0` como tecla pulsada**.

Similarmente, para acceder a las teclas `Start`, `Select`, `A` y `B`, los bits de selección tendrán que establecerse a `10`.

Figura 18. Resto de botones

Este diseño se debe a la circuitería interna del chip de lectura del teclado, que se estructura en una cuadrícula de dos columnas y cuatro filas.

De esta forma, cuando el bit 4 está activado y el bit 5 desactivado, la columna de la derecha es ignorada, y al leer cualquiera de los bits 0 al 3 se obtiene el estado de la columna izquierda (en este caso se corresponde con las flechas del *gamepad*).

Paralelamente, si se invierte el estado de los bits 4 y 5 podremos leer la columna de la derecha por el mismo motivo.

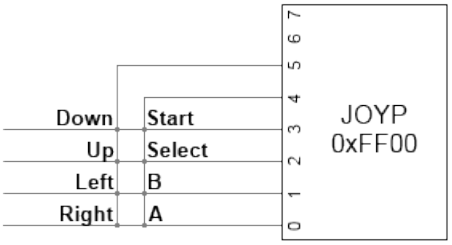

Figura 19. Circuitería del teclado [2]

3.5.2. *Link cable*

Es un accesorio que permite a los jugadores conectar dos consolas Game Boy, generalmente para jugar en modo multijugador. Está compuesto por 3 cables, dos para datos y uno para el reloj.

El reloj indica a ambas consolas cuándo tienen que leer la información que está siendo transmitida, para así evitar corrupción de datos. Al inicio de la comunicación, las dos máquinas **deben acordar cuál enviará los pulsos de reloj** y cuál los leerá para sincronizar la información.

Los otros dos cables sirven para transmitir la información de forma **bidireccional** (un cable para un sentido y el otro para el contrario). El link cable envía los datos **serializadamente**, es decir, utiliza un solo carril (cable) para cada sentido de la comunicación, enviando bit a bit la información de un byte.

Este dispositivo fue utilizado en juegos como Tetris para partidas multijugador, o en la primera generación de juegos Pokémon para el intercambio de ítems.

4. CONCLUSIONES

La Game Boy es sin duda una máquina suficientemente potente para ejecutar programas informáticos acordes a su época.

Pese a sus años, su arquitectura se sigue asemejando a la base de los sistemas informáticos actuales, por lo que comprender su funcionamiento nos puede servir de base para entender máquinas mucho más complejas.

REFERENCIAS

[1] Blume, O. (2014). DMG-01: Overview. Disponible en http://dot-matrix-game.blogspot.com /2014/01/overview.html

[2] Nazar, I. (2010). GameBoy Emulation in JavaScript: The CPU. Disponible en http://imrannazar.com/GameBoy-Emulation-in-JavaScript:-The-CPU

[3] Steil, M. (2016). The Ultimate Game Boy Talk (33c3). Disponible en https://www.youtube.com/watch?v=HyzD8pNlpwI

BIOGRAFÍAS

José Miguel Moreno, **Marco Pérez** y **Carlos Rascón** son estudiantes de Ingeniería Informática de Servicios y Aplicaciones en el Campus María Zambrano de la Universidad de Valladolid en Segovia.