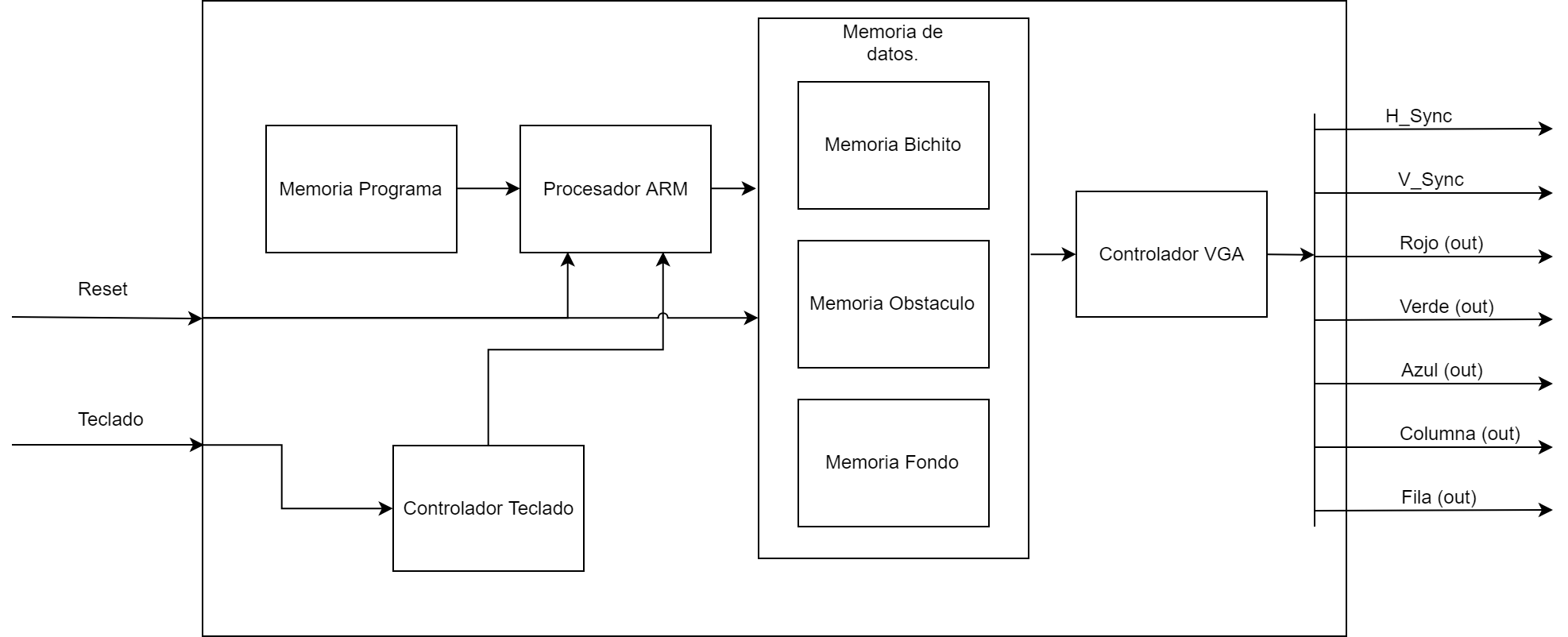
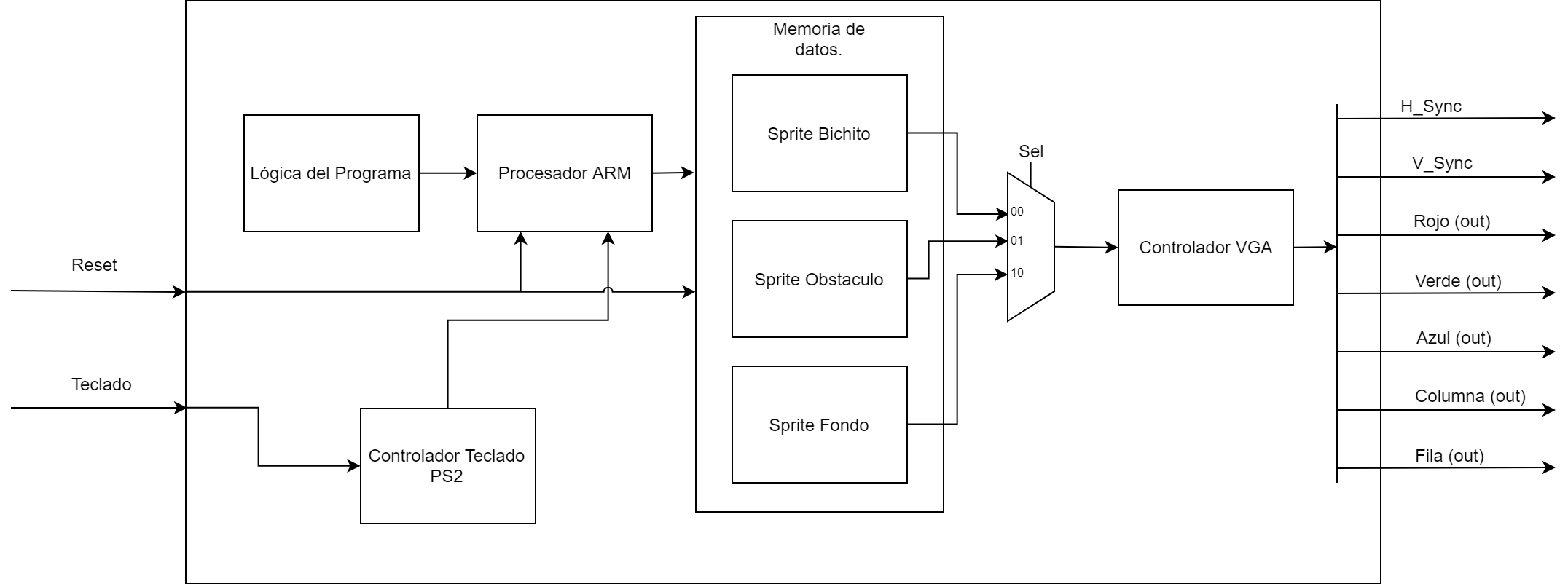
Primer Nivel.



Segundo Nivel



Tercer Nivel



Cuarto Nivel

a. Nombre del módulo.

b. Diagrama modular (copiar módulo del diagrama de tercer nivel).

c. Objetivo del módulo.

d. Entradas.

e. Salidas.

f. Explicación de la relación con otros módulos.

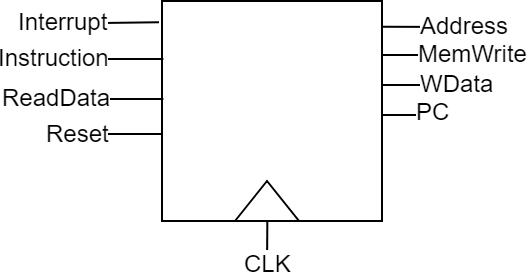
g. Explicación de funcionamiento.

h. Diseño. ¿Cómo se diseña? Justificación, tablas de verdad, simplificaciones, uso de módulos integrados, etc.

i. Diagrama esquemático detallado del diseño (por compuertas lógicas).

j. Diagrama completo de conexiones eléctricas del diseño (por chips).

# Procesador ARM



**Objetivo del módulo.**

Ejecutar instrucciones ARM para la realización de operaciones aritméticas y lógicas, saltos, lectura y escritura en memoria.

**Entradas.**

* **Interrupt:** Señal de interrupción de que se presionó la tecla arriba o abajo del teclado, esta señal activa el código que se ejecuta para manejar la interrupción.
* **Instruction:** Instrucción proveniente desde la memoria de instrucciones que se va a ejecutar en el procesador, son operaciones de tipo aritméticas y lógicas, saltos, etc.
* **ReadData:** Señal proveniente de la memoria de datos y equivale a un dato extraído desde esta memoria.
* **Reset:** Señal de reset proveída desde un botón en la FPGA que setea todas las señales para iniciar/reiniciar el juego.

**Salidas.**

* **Address:** Esta señal es el resultado de la ALU cuando se realiza un cálculo de una dirección a la memoria de datos.
* **MemWrite:** Señal generada por la Unidad de Control que indica a la memoria de Datos si se va a escribir en dicha memoria**.**
* **WData:** Dato que se va a guardar en memoria de datos.
* **PC:** Señal del registro del PC que indica a la memoria de instrucciones la dirección de la siguiente instrucción que se debe ejecutar.

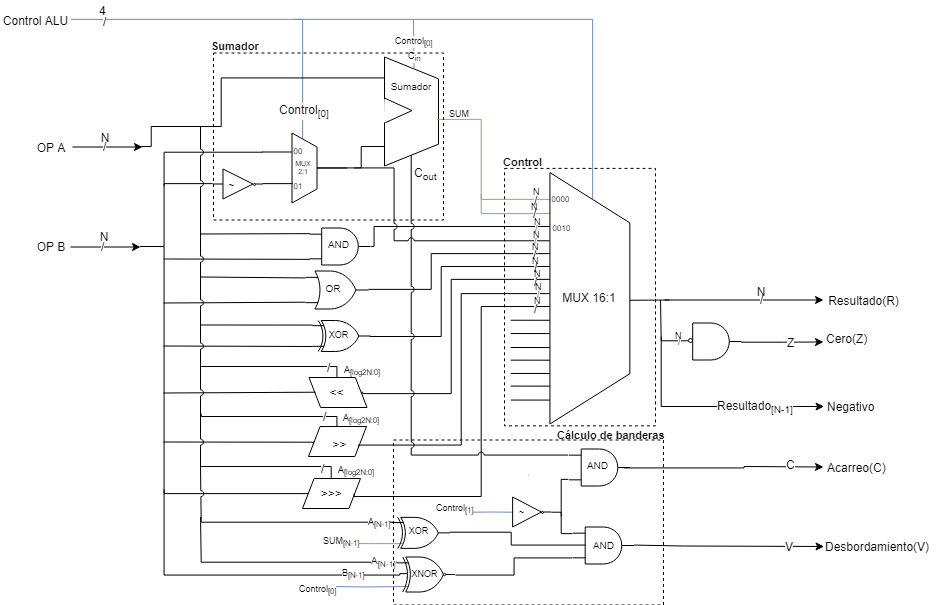
**Explicación de la relación con otros módulos.**

* **Memoria de Instrucciones:** Como su nombre lo dice, esta memoria almacena las instrucciones que se van a ejecutar como programa (el juego del bichito), el procesador debe indicar a la memoria cuál es la siguiente instrucción que se va a ejecutar una vez ejecutada la instrucción actual, esto lo hace pasando mediante la señal del PC la dirección de memoria donde está almacenada la siguiente instrucción.
* **Memoria de datos:** El procesador utiliza esta memoria para leer y almacenar datos, se tiene que generar una señal para saber cual de las dos operaciones se va a realizar, la dirección de memoria donde se va a aplicar la operación y en caso de escritura el dato que se va a escribir. Caso aparte con la lectura, que la memoria envía el dato que se está accediendo.

**Explicación de funcionamiento.**

El procesador va ejecutando las instrucciones almacenadas en la memoria de instrucciones de acuerdo a la lógica de las mismas, realiza las operaciones aritméticas, lógicas, saltos, lecturas y escrituras en memoria para modificar los datos de acuerdo a las necesidades del usuario expresadas en las instrucciones.

# ALU



**Objetivo del módulo.**

Este módulo se encarga de controlar la ALU, ya que todos los módulos se encuentran trabajando al mismo tiempo. El multiplexor recibe la salida de los módulos y mediante los 4 bits de control selecciona cual de estos resultados se verán en la salida de la ALU. Para poder realizar las operaciones lógicas y aritméticas correspondientes.

**Entradas.**

* **C1**: Es el bit 0 de selección.
* **C2**: Es el bit 1 de selección.
* **C3**: Es el bit 2 de selección.
* **C4**: Es el bit 3 de selección.
* **OP A**: Operando A
* **OP B**: Operando B

**Salidas.**

* **Resultado**: El valor de entrada seleccionado por el MUX.
* **Cero**: Obtiene un Bit de 0.
* **Negativo**: La salida sin el último bit (bit de signo).
* **Acarreo**: En caso en que se haga operaciones aritméticas y en las mismas haya un residuo.
* **Desbordamiento**: Esto se da en caso de que falle alguna operación.

**Explicación de la relación con otros módulos.**

Este módulo se encuentra dentro del procesador y se relaciona con la memoria de registros de donde obtiene los datos en ciertos caso, y con la memoria de datos ya que envía el resultado lógico/aritmético realizado y si es necesario guarda el mismo en memoria de datos o memoria de registros.

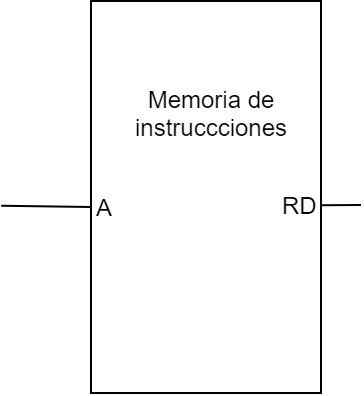
# **Explicación de funcionamiento.**

# Para explicar el funcionamiento de la ALU se muestra una tabla con las diferentes opciones que presenta con los cuatro bits de control.

|  |  |  |
| --- | --- | --- |
| **Control** | **Operación** | **Descripción** |
| 0000 | suma | A + B |
| 0001 | resta | A - B |
| 0010 | and | A & B |
| 0011 | negación | ~B |
| 0100 | or | A | B |
| 0101 | xor | A ⊕ B |
| 0110 | corrimiento izq (<<) | B << Alog2N |
| 0111 | corrimiento derecho (>>) | B >> Alog2N |
| 1000 | corrimiento aritmético (>>>) | B >>> Alog2N |
| ... |  |  |
| default |  | 0 |

# 

# Memoria de Instrucciones



**Objetivo del módulo.**

Proporcionar al procesador las instrucciones del programa de juego del bichito saltarin

**Entradas.**

* **A:** Dirección de memoria proveniente desde el registro del PC del procesador que indica la posición de la siguiente instrucción que se va a ejecutar.

**Salidas.**

* **RD:** Instrucción accesada por la entrada A. Es la siguiente instrucción a ejecutar.

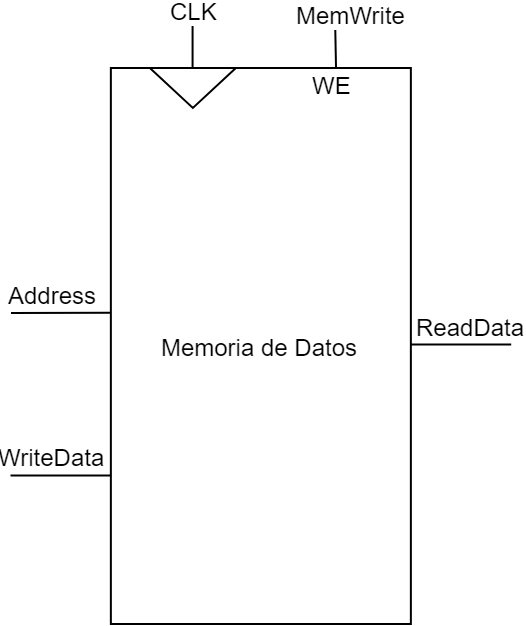
**Explicación de la relación con otros módulos.**

* **Procesador:** Esta memoria proporciona al procesador las instrucciones del programa que se debe ejecutar

**Explicación de funcionamiento.**

Memoria de tipo ROM que contiene las instrucciones del programa que se va a ejecutar, no permite escrituras.

# Memoria de Datos

****

**Objetivo del módulo.**

Sistema de almacenamiento de datos, permitiendo lectura y escritura.

**Entradas.**

* **Address:** Dirección de memoria donde se van a realizar la operación de Lectura o Escritura.
* **MemWrite:** Señal que habilita la escritura en la memoria. También llamada WriteEnable.
* **WriteData:** Dato que se va a escribir en memoria.
* **CLK:** Clock.

**Salidas.**

* **ReadData:** Dato leído para una dirección de memoria proporcionada en la entrada Address.

**Explicación de la relación con otros módulos.**

* **Procesador:** Esta memoria proporciona un mecanismo de almacenamiento y lectura de datos que pueden ser operados por el procesador.

**Explicación de funcionamiento.**

Memoria de tipo RAM que contiene datos almacenados como resultado de instrucciones ejecutadas por el procesador. Permite lectura y escritura.

# Sprite

****

**Objetivo del módulo.**

El objetivo del módulo es pintar en pantalla los objetos que representan al bichito, los obstáculos y algunos elementos del fondo en la pantalla. En este caso en memoria se trabaja con tres módulos diferentes en los cuales, pero los mismos tienen las mismas entradas y salidas. Lo que cambia en sí es la memoria que contiene.

**Entradas.**

* **Pos\_X:** La posición X, en la cual se está barriendo la pantalla en cada momento, representa que se está mostrando en pantalla en un momento específico.
* **Pos\_Y:** La posición Y de barrido en la pantalla, representa la posición en pantalla en ese momento específico.
* **Curr\_Y:** El movimiento en Y, en caso de que sea el bichito, y X cuando para el movimiento del obstaculos. Este valor de entrada viene del procesador, el cual siempre mantiene los mismos guardados. En caso del obstáculo el X va disminuyendo de forma constante, representando el movimiento del obstáculo al hacia el bichito.

**Salidas.**

* **R,G,B:** Estos tres valores de salida son todos de 7 bits, cada uno representa el color que se quiere pintar, el mismo se envía al controlador de la VGA. Representan los valores del color que se desea pintar.
* **Visible:** Es un bit que define si el sprite se muestra o no en pantalla.

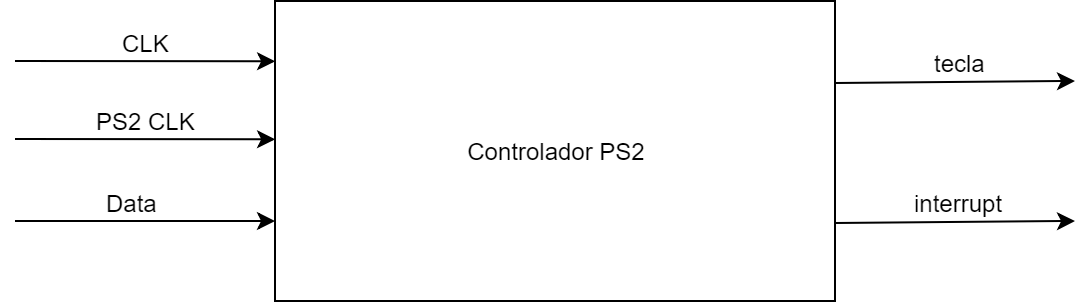
**Explicación de la relación con otros módulos.**

Los sprite se relacionan con dos módulos, el procesador ARM y el controlador VGA. El procesador como se mencionó anteriormente mantiene en sus registros los valores en Y del bichito en cada momento, y el X respectivo de cada obstáculo. Se suma o se resta estas posiciones dependiendo de la acción que realiza el teclado mediante la conexión de ps2, estas sumas y restas son realizadas por el procesador. Y el controlador VGA le envía constantemente la posición en la que está pintando en este momento al controlador, para que el mismo sprite le envíe si el elemento es visible o no y los respectivos colores que debe pintar.

**Explicación de funcionamiento.**

Como se mencionó en la relación entre módulos, el funcionamiento del módulo del sprite es decidir si se pinta o no el sprite en pantalla, y cual sprite se debe pintar. Para esto trabaja en conjunto con el procesador y el controlador VGA. Pero en si la función principal es decidir dependiendo en la posición en la que está barriendo en pantalla si se debe pintar el sprite.

# Controlador PS/2



**Objetivo del módulo.**

El módulo es un controlador de teclado, el cual se sirve como interfaz entre el teclado y el juego del bichito saltarín.

**Entradas.**

* **CLK:** Se refiere al reloj de la FPGA.
* **PS2 CLK:** Es el reloj proveniente desde el teclado.
* **Data:** Son 11 bits que genera el teclado al presionar una tecla, dentro de los cuales se encuentra el código único de la tecla presionada.

**Salidas.**

* **tecla:** Código único de 8 bits de la tecla presionada, para este proyecto solo es de interés las teclas arriba y abajo. Este dato es almacenado en un registro.
* **interrupt:** Señal que genera una interrupción en el procesador para indicarle que se presionó una tecla de interés y ejecute el código de la interrupción. Cada vez que existe un interrupt, el procesador lee el registro donde se guardó la tecla y dependiendo de la tecla ejecutará ciertas instrucciones.

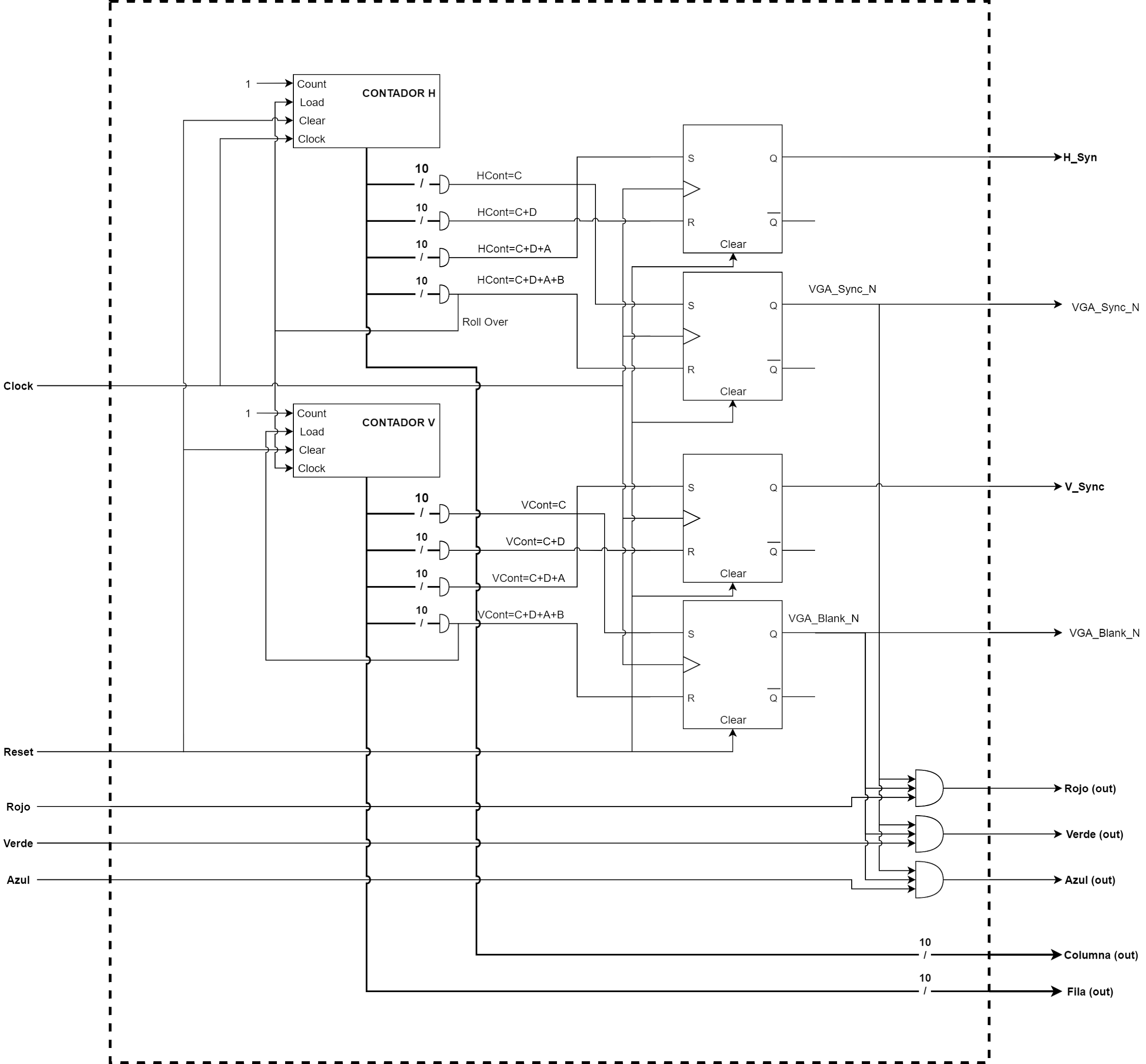
**Explicación de la relación con otros módulos.**

* **Procesador:** Genera una interrupción para que el procesador sepa que se presionó una tecla de interés y ejecute el código de la interrupción, en este caso cambiar la posición del bichito saltarín.
* **Sprites:** De manera indirecta, por medio del procesador, es el responsable de cambiar la posición donde se debe pintar en pantalla el bichito.

**Explicación de funcionamiento.**

Interfaz utilizada para detectar cuando se presiona una tecla en el teclado de tipo PS2, solamente se da importancia a las teclas de arriba y abajo, utilizadas para hacer saltar al bichito. Este módulo también genera las señales para avisar a otros módulos cuando una tecla ha sido presionada

# Módulo Controlador de Video



**Objetivo**

Este es el controlador de video VGA, el cual generará las salidas H\_Sync, V\_Sync, Rojo, Verde y Azul, además de la posición del pixel (Fila, Columna) necesarias para pintar en pantalla un color.

**Entradas**

* Clock: Es el reloj que reciben ciertos componentes, para este caso es de 25 MHz.
* Reset: Señal que pone en estado inicial el módulo
* Rojo: Color rojo (8 bits) proveniente del generador aleatorio.
* Verde: Color verde (8 bits) proveniente del generador aleatorio.
* Azul: Color azul (8 bits) proveniente del generador aleatorio.

**Salidas**

* H\_Sync: Utilizada para controlar el tiempo de la tasa de escaneo, determina el tiempo que toma escanear un “row”.
* V\_Sync: Utilizada para controlar el tiempo de la tasa de escaneo, determina el tiempo que toma escanear la pantalla entera.
* VGA\_Sync\_N: Cuando está en bajo fuerza al DAC verde a un nivel de sincronización especial por debajo del nivel de negro normal.
* VGA\_Blank\_N: Cuando está en bajo fuerza a las tres salidas DAC a su nivel de negro.
* Rojo: Salida de color rojo que se conecta al VGA
* Verde: Salida de color verde que se conecta al VGA
* Azul: Salida de color azul que se conecta al VGA
* Columna: Número de la columna para un píxel
* Fila: Número de fila para un píxel.

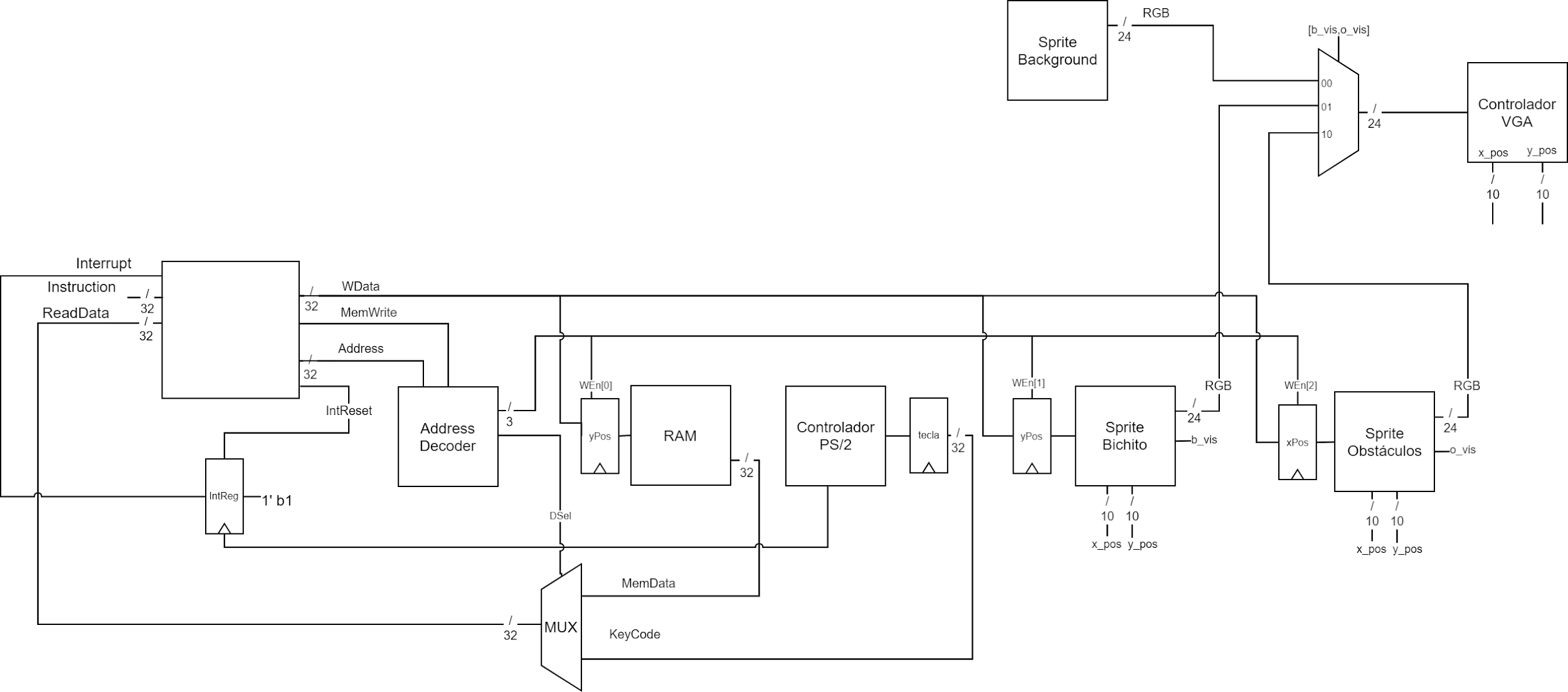
**Explicación de la relación con otros módulos**

La relación que tiene el controlador de la VGA con los otros módulos solo viene relacionado con los módulos de cada sprite ya que los mismos son los que se pintan en pantalla, ellos brindan los colores y la lógica para saber si se encuentran visibles o no. Los mismos vienen desde un mux que dependiendo a la visibilidad decide que sprite pintar.

**Explicación de funcionamiento**

Utilizando una frecuencia de 25 MHz para el contador horizontal, se van generando las señales para pintar cada píxel, el recorrido se hace por filas, de tal manera que cuando una fila acaba, esta señal sirve de clock para el contador vertical y de reinicio para el mismo contador horizontal. Estos contadores horizontal y vertical generan las señales de H\_Sync, V\_Sync, VGA\_Sync\_N y VGA\_Blank\_N.

Quinto Nivel



Procesador

