IMPLEMENTACIÓN DE LA PILA EN UN PROCESADOR.

La pila es una zona de memoria que tiene varias funciones en el procesador:

1. Esta destinada a guardar las direcciones de retorno del contador de programa (PC[1]-Program Counter) después del llamado a subrutinas o funciones.
2. En ella se colocan los parámetros en las llamadas a funciones en los lenguajes de alto nivel.
3. Se guarda el estado del procesador cuando se ejecuta el sistema de interrupciones. Normalmente el contador de programa, el registro de estado y algunos registros de control del procesador.
4. Se usa como una zona de memoria para guardar datos y registros temporalmente por el programador.

Existen dos formas de implementar una pila en una arquitectura:

1. Implementación usando la memoria principal del sistema.

Esta es la forma de implementación en un procesador de propósito general como INTEL, Motorola, AMD o ARM. Aquí se toma una sección de la memoria principal (DRAM) del sistema para destinarla al uso de la pila, la ventaja con esta implementación es que podemos tener la pila de un gran tamaño (muchos niveles) puesto que generalmente los recursos en la memoria principal son grandes. La desventaja es que la memoria principal tiene un tiempo de acceso muy grande, por lo que el acceso a la pila en las operaciones de lectura (POP) y escritura (PUSH) es lento.

Los programas recursivos hacen un uso muy intenso de la pila al llamar a la misma función muchas veces así misma, lo cual provoca dos cosas:

1. Que se pueda acabar la pila rápidamente ocasionando una excepción en el procesador llamada “Stack overflow”.
2. Al tener muchos accesos a la memoria principal, la ejecución del programa recursivo siempre es mas lento que su versión iterativa.

2. Implementación en hardware.

Un diseño en hardware permite obtener un llamado a subrutinas con una mayor velocidad y desempeño que una pila implementada en memoria. En esta implementación cada una de las posiciones de la pila permite guardar la dirección de retorno del contador de programa en una llamada a subrutina. Cada posición de la pila se le conoce como un “nivel” de tal modo que si tenemos una pila de “n” niveles podremos anidar el llamado de funciones “n” veces.

La implementación en hardware se puede realizar de dos formas:

1. Implementación con múltiples contadores de programa. En esta implementación se usan “n” contadores de programa para una pila de “n” niveles. Esta implementación puede ser muy costosa en términos de recursos por los múltiples Flip-Flops que se usan cuando el número de niveles es alto. Esta implementación se muestra en la ilustración 1.

[1] El contador de programa (PC-Program Counter) es el registro que contiene la dirección de la siguiente instrucción a ejecutar por parte del procesador.

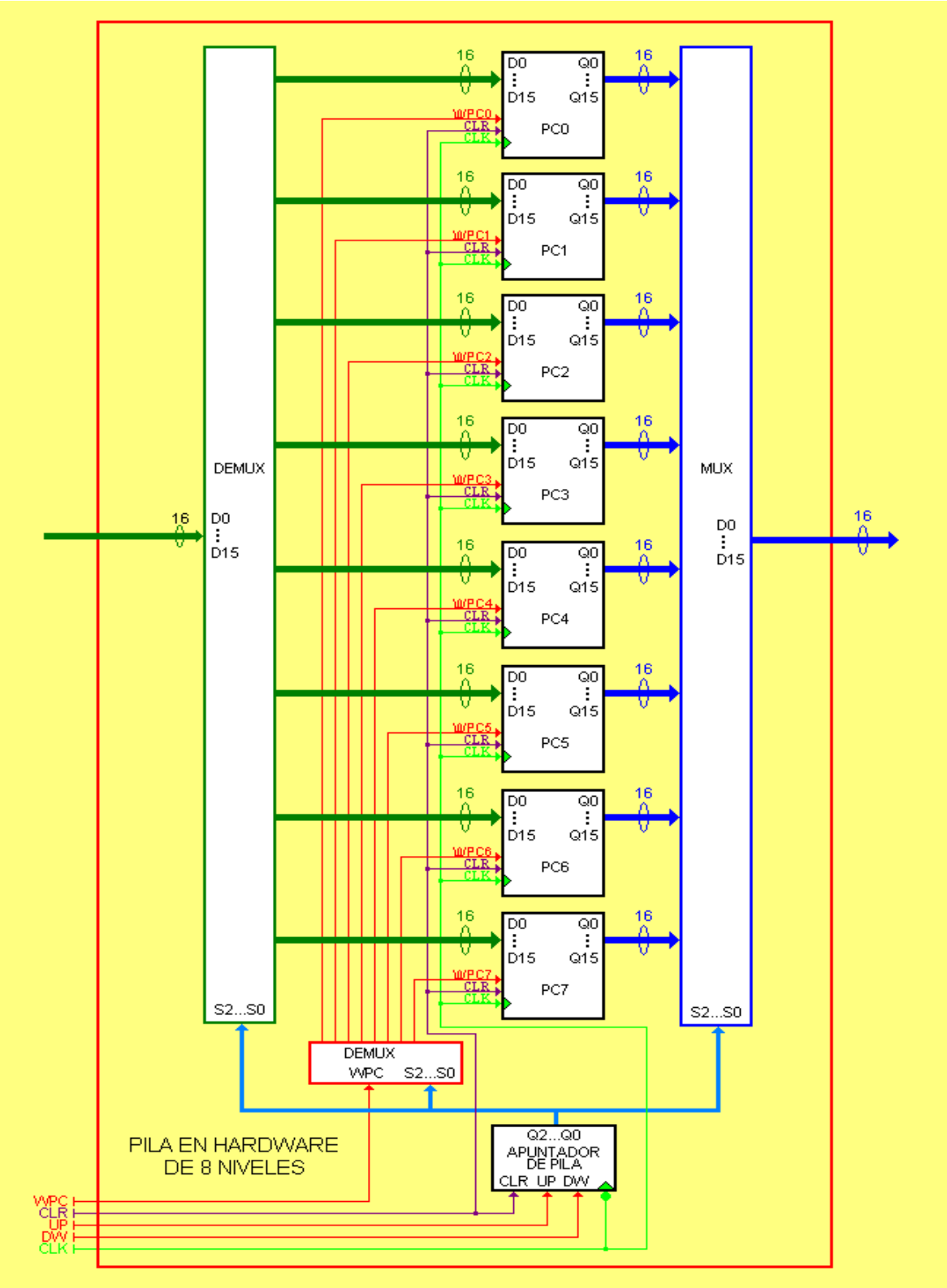

Ilustración 1: Arquitectura de la pila en hardware con múltiples contadores.

2. Implementación con memoria RAM de un puerto. En esta implementación se usa solo un contador de programa para la ejecución de instrucciones en el procesador. Los “n” niveles de la pila se implementan usando memoria RAM de un puerto. Esta implementación es más eficiente en términos de recursos porque se puede usar el recurso dedicado de memoria RAM distribuida que nos ofrece las LUT’s de los Slices de un FPGA.

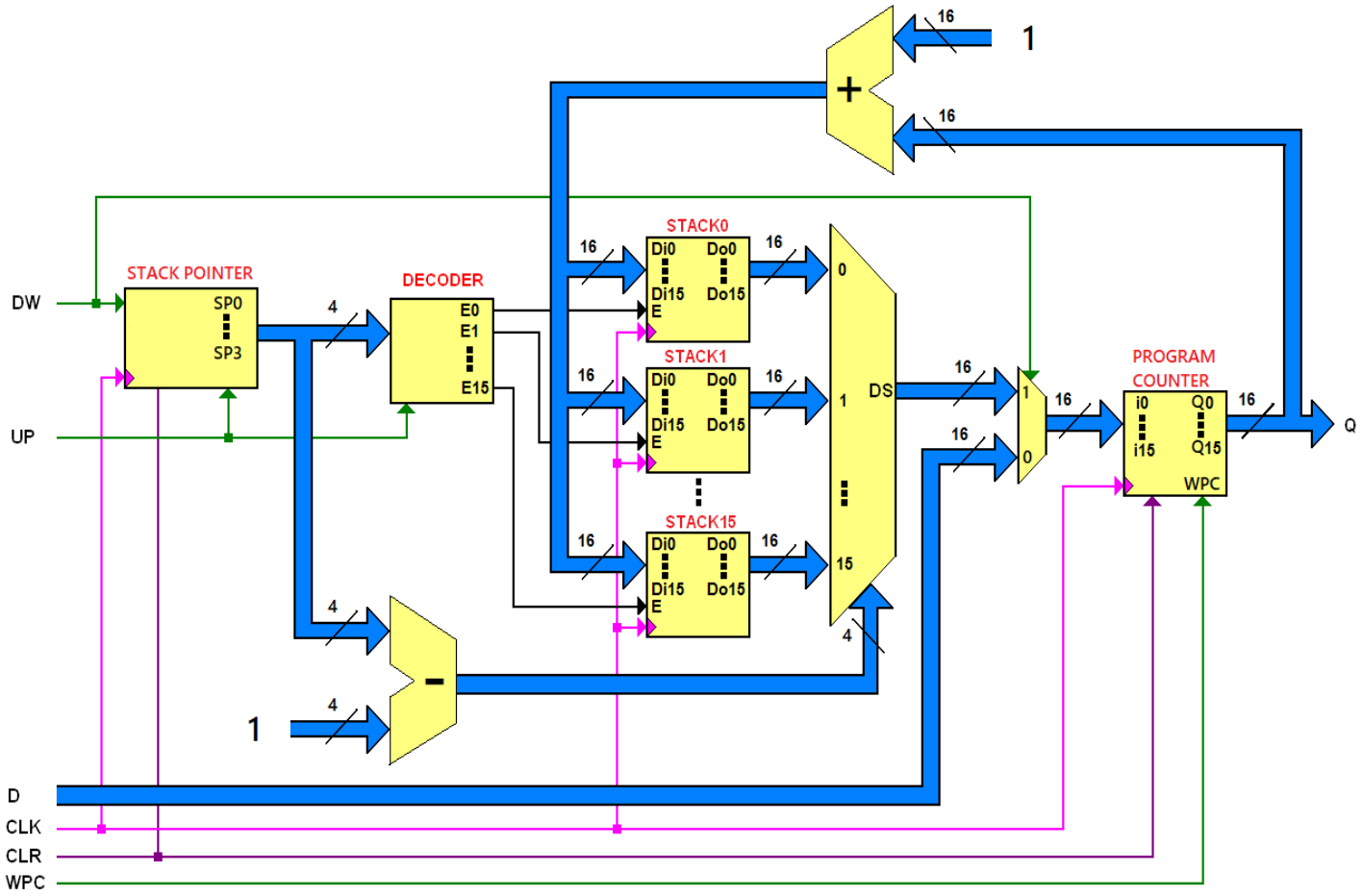

Ilustración 2: Arquitectura de la pila en hardware con memoria RAM de un solo puerto.

En ambas implementaciones la ejecución de una instrucción CALL y RET se realiza en tan solo un ciclo de reloj.

El bloque de la pila en el microprocesador ESCOMIPS se implementa en hardware con memoria RAM de un puerto, en la ilustración 3 se observan las señales de control que maneja la pila, estas son:

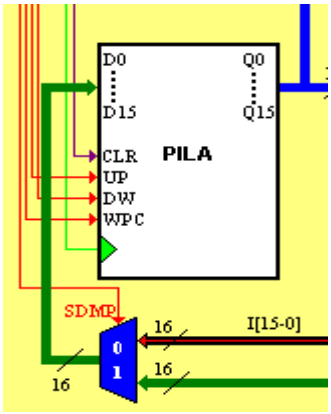

Ilustración 3 Pila en hardware.

➢**Bus de datos (D0,...,D15).** Este es un bus de entrada de 16 bits, aquí se pone la dirección a donde tiene que saltar el procesador cuando se llaman a funciones o se ejecutan brincos condicionales o incondicionales en un programa. Esta dirección se guarda en el PC actual cuando ejecutamos las instrucciones de brinco condicional e incondicional (BEQI, BNEI, BLTI, BGTI, BLETI, BGETI, B). Cuando se ejecuta la instrucción CALL la dirección colocada en este bus se debe guardar en el siguiente PC apuntado por el registro apuntador de pila (SP - Stack Poniter) o tope de la pila.

➢**Bus de salida (Q0,...,Q15).** Este es un bus de salida de 16 bits, donde sale la dirección del PC apuntado por el registro SP hacia la memoria de programa para poder leer la siguiente instrucción a ejecutar por el procesador.

➢**CLK**. Señal de reloj que sincroniza las acciones del contador de programa y el apuntador de pila.

➢**CLR**. Señal de clear o reset. Esta señal se ejecuta de forma asíncrona, es decir, se ejecuta en el instante que la unidad de control la coloca en 1. Permite inicializar a cero el contador de programa y el apuntador de pila.

➢**UP**. Señal que permite guardar la dirección de la siguiente instrucción a ejecutar después de haber llamado a una subrutina. Esta dirección se guarda en la memoria RAM de un solo puerto. Al mismo tiempo se carga en el contador de programa la dirección de la subrutina a ejecutar y se incrementa el apuntador de pila al siguiente nivel disponible. Esta señal se activa cada vez que se ejecuta una instrucción CALL, y se ejecuta de forma síncrona, es decir, se ejecuta en el momento que llega una flanco de subida de la señal de reloj.

➢**DW**. Señal que permite restaurar la dirección de la siguiente instrucción a ejecutar después de haber llamado a una subrutina. Esta dirección se obtiene de la memoria RAM de un solo puerto y se carga en el contador de programa. Al mismo tiempo se decrementa el apuntador de pila un nivel. Esta señal se activa cada vez que se ejecuta una instrucción RET. Esta señal se ejecuta de forma síncrona, es decir, se ejecuta en el momento que llega una flanco de subida de la señal de reloj.

➢**WPC**. Esta señal indica cuando queremos cargar una dirección a través del bus de datos (D0...D15) en el contador de programa. Esta señal se ejecuta de forma síncrona, es decir, se ejecuta en el momento que llega una flanco de subida de la señal de reloj.

Para entender mejor el funcionamiento de la pila observemos la ilustración 3. En esta ilustración se observan 8 contadores de 16 bits etiquetados como PC0, PC1, …, PC7. Estos contadores se seleccionan con otro contador etiquetado como “Apuntador de Pila” o “Stack Pointer (SP)”, cuando SP = 000, el PC0 es el contador que esta usando el procesador para la ejecución de instrucciones. Si mandamos llamar a una subrutina a través de la instrucción CALL, la unidad de control del procesador coloca la señal UP=1, con esto lo que hacemos es incrementar el SP en uno (SP=001), seleccionando así el contador PC1. Además, se coloca la señal WPC=1 lo que permite cargar la dirección colocada en el bus de datos (D0...D15) en el PC1. Cuando se ejecuta la instrucción RET, la unidad de control del procesador coloca la señal DW (Down) = 1, decrementando así el SP y regresando al PC0. **Hay que notar que el Apuntador de pila se activa con la señal de reloj en flanco de bajada. Esto se hace para poder sincronizar las acciones en la ejecución de la instrucción CALL ya que en el flanco de bajada el Apuntador de Pila se actualiza, seleccionando otro registro PC y en la flanco de subida se carga la dirección de salto en el registro PC seleccionado.**