Tema 7. Ejercicio resuelto. 8 variantes de E/S síncrona

En este documento se explica en detalle un ejemplo/ejercicio-resuelto de un PPE sencillo sobre el que se varía el protocolo síncrono de E/S de datos/resultados (los circuitos secuenciales, emisor y receptor tienen la misma señal de reloj que el PPE a diseñar) y después la asíncrona (cuando la frecuencia y fase de la señal de reloj del emisor y del receptor son, en principio, distintas y el emisor no conoce nada sobre el reloj del receptor y viceversa).

Funcionalidad del Procesador de Propósito Específico.

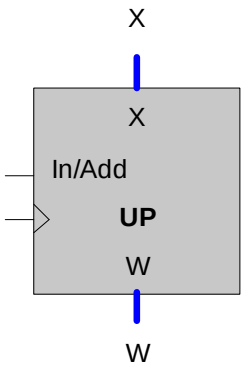
El cálculo que realizará el PPE que vamos a diseñar consiste en sumar cuatro números que llegan secuencialmente por el bus de entrada X y entregar el resultado en el bus de salida W (todos los buses son de n bits). Se considera que todos los datos y el resultado son números naturales representados en binario con n bits y no se tiene en cuenta si el resultado de las sumas sea o no representable en binario con n bits. El cálculo a realizar es:

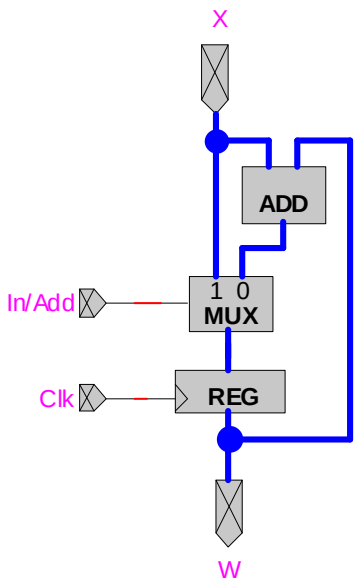
$$W = (X_0 + X_1 + X_2 + X_3) \bmod 2^n$$

Cada uno de los cuatro operandos estará presente durante un ciclo en el bus de entrada X y los cuatro operandos llegarán secuencialmente por el bus X, en ciclos consecutivos, uno detrás de otro.

Unidad de proceso (UP)

Una posible unidad de proceso para realizar este cálculo se muestra a continuación. Esta será la UP que usaremos en todas las variantes del ejercicio. En cada caso variará el diseño de la unidad de control, que se adaptará a diferentes enunciados sobre la forma de sincronizar la entrada de operandos y la salida de resultados.





La UP tiene un registro acumulador. Este registro se carga, al final de cada ciclo, con uno de dos posibles valores, según valga la señal In/Add (que genera la unidad de control) durante ese ciclo.

- Si In/Add vale 1 se cargará el valor presente en el bus de entrada X durante ese ciclo (acción Input).
- Si In/Alu vale 0 se cargará el valor resultante de sumar el contenido del registro acumulador más el valor presente en el bus de entrada X durante ese ciclo (acción Add).

Para sumar los cuatro operandos, la unidad de control habrá de generar un 1 en la señal In/Add durante el ciclo en que el primer operando, X_0, se encuentre en el bus X y un 0 en los siguiente tres ciclos, en los que se encontrará en el bus X cada uno de los operandos X_1, X_2 y X_3, a razón de uno por ciclo. En cada uno de estos tres ciclos se sumará el resultado parcial almacenado en el registro acumulador con el operando presente en la entrada X y se cargará, al final del ciclo, el resultado de la suma en el registro acumulador. Un cronograma simplificado nos puede ayudar a entender esto para una secuencia concreta de cuatro valores a sumar que llegan por X a partir del ciclo, por ejemplo, 2 (las celdas con x o XX indican valores de la señal o del bus que no conocemos o que no nos importan para lo que queremos visualizar):

Ciclo	0	1	2	3	4	5	6	7	8	9	10	11	12
X (hexa)	XX	XX	3A	F4	18	63	XX	XX	XX	XX	XX	XX	XX
In/Add	x	x	1	0	0	0	x	x	x	x	x	x	x
W (hexa)	XX	XX	XX	3A	2E	36	99	XX	XX	XX	XX	XX	XX

Unidad de control

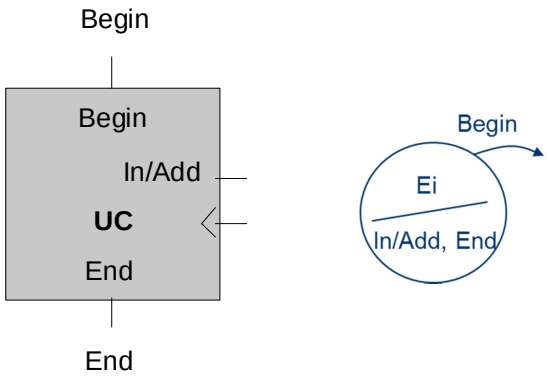
A la unidad de control (UC) del PPE le llega del exterior una señal (de un bit) denominada Begin y de la UC sale hacia el exterior una señal denominada End.

- Begin: Se usa para sincronizar la entrada de datos por el bus X (y determinar cuándo hay que comenzar un nuevo cálculo: la suma de 4 nuevos números).
- End: Se usa para sincronizar la salida del resultado por W (e indicar que termino el cálculo en curso).

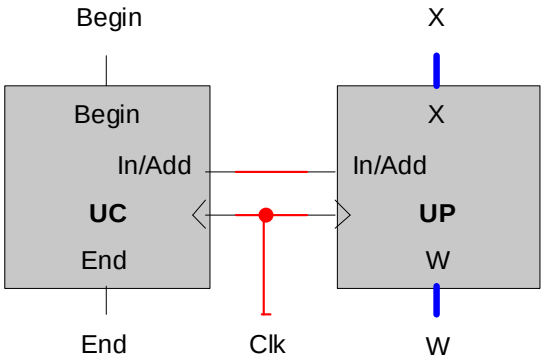
A continuación definimos la sincronización de la entrada de operandos y la salida de resultados para diferentes casos de sincronización y dibujamos el grafo de la UC para cada uno de ellos. Seguiremos la siguiente notación:

- Begin(c) y End(c) indican el valor binario presente en las señales Begin y End en el ciclo genérico c.
- $X(c)_u$ y $W(c)_u$ indican el número natural representado en los n bits del bus X y W en el ciclo c.

Por lo tanto, en todos estos casos las entradas y salidas del bloque UC y la leyenda del grafo son las siguientes:



Así, el esquema en dos bloques del PPE es el siguiente:



Caso 1a

Enunciado:

```
If (Begin(c) == 1) {
   W(c+5)u = ( X(c+1)u + X(c+2)u + X(c+3)u + X(c+4)u) mod 2n ;
   End(c+5) = 1;
}
```

Además:

- La señal End valdrá 0 el resto de ciclos.
- Se ignorará el valor de la señal Begin durante los ciclos c+1 al c+5 ambos incluidos.

Resultado:

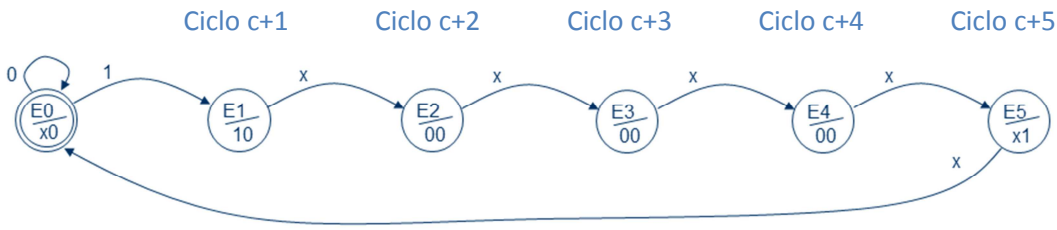
Mientras Begin vale 0 la UC se mantiene en el estado inicial, E0, esperando a que Begin valga 1. Si estando en el estado E0 la entrada Begin vale 1, diremos que estamos, de acuerdo con el enunciado, en el ciclo c. En este caso, al ciclo siguiente (ciclo c+1), la UC estará en el estado E1. La entrada presente en X durante este ciclo c+1 se debe guardar al final de este ciclo en el registro acumulador (ya que se trata del primer operando X_0. Para ello, la UC debe generar el valor 1 en la señal In/Add durante el ciclo c+1 (salida In/Add = 1 en el estado E1). En el ciclo

c+2, c+3 y c+4 se sumará cada uno de los tres operandos con el contenido del registro acumulador, por lo que en estos tres ciclos (Estados E2, E3 y E4) la señal In/Add debe valer 0.

En el estado E0 no importa el valor de In/Add, pues lo que se cargue al final del ciclo en el registro no se usa para nada: si en E0 Begin vale 1, al ciclo siguiente (en E1) se cargará el primer operando en el registro, chafando el valor cargado en E0.

Por último, durante el ciclo siguiente de sumar el último operando se puede leer el resultado de la suma de los cuatro números en el bus de salida W (ciclo c+5, estado E5), por lo que durante este ciclo la salida End valdrá 1.

En el grafo se etiquetado cada nodo E1, …, E5 con el número de ciclo en el que se encuentra la UC si denominamos c el ciclo en el que, estando en el estado E0, Begin vale 1. Esto se ha añadido para ayudar a entender el grafo, son comentarios que no forman parte del grafo.



Caso 1b

Enunciado:

La sincronización es como en el caso anterior excepto que ahora End vale 1 en el ciclo c+4. De esta forma el protocolo de comunicación es el mismo tanto para la entrada como para la salida:

- Begin a 1 valida el primer operando, que está presente en X al ciclo siguiente y
- End a 1 valida el resultado, que está presente en W al ciclo siguiente.

El enunciado completo de este caso es (a continuación se usa negrita para detallar los cambios sobre el enunciado del caso anterior y esto se hará en todos los casos que siguen):

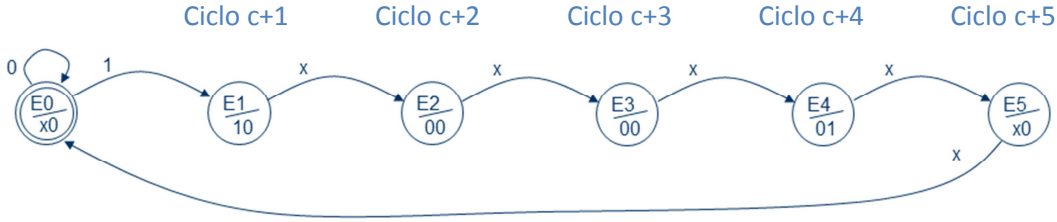
```
If (Begin(c) == 1) {
   W(c+5)u = ( X(c+1)u + X(c+2)u + X(c+3)u + X(c+4)u) mod 2n ;
   End(c+4) = 1;
}
```

Además:

- La señal End valdrá 0 el resto de ciclos.
- Se ignorará el valor de la señal Begin durante los ciclos c+1 al c+5 ambos incluidos.

Resultado:

El grafo es igual al del caso anterior excepto que End vale 1 un ciclo antes, en el estado E4, en vez de hacerlo en E5.



Caso 1c

Enunciado:

La sincronización es como en el caso anterior excepto que ahora Begin no se debe ignorar en el ciclo c+5. Así, con este enunciado, se pueden sumar dos secuencias de 4 operandos con un ciclo menos de separación entre ambas secuencias que en el caso anterior (antes tenía que haber al menos dos ciclos de separación y ahora es suficiente con uno) .

El enunciado completo es:

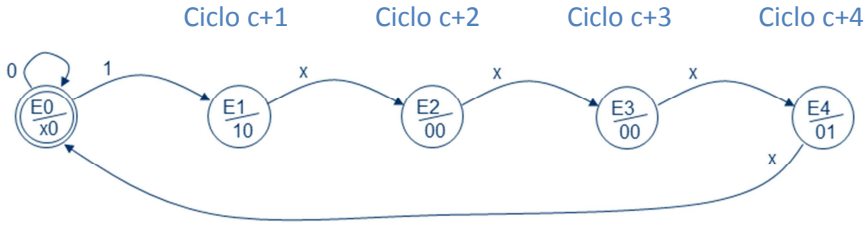
```
If (Begin(c) == 1) {
   W(c+5)u = ( X(c+1)u + X(c+2)u + X(c+3)u + X(c+4)u) mod 2n ;
   End(c+4) = 1;
}
```

Además:

- La señal End valdrá 0 el resto de ciclos.
- Se ignorará el valor de la señal Begin durante los ciclos c+1 al **c+4** ambos incluidos.

Resultado:

El grafo es igual al del caso anterior excepto que desaparece el estado E5 anterior y ahora es del E4 del que se pasa al E0.



Caso 1d

Enunciado:

La sincronización es como en el caso anterior excepto que ahora Begin no se debe ignorar en el ciclo c+4. Con este enunciado se pueden realizar dos cálculos consecutivos (cada cálculo suma cuatro números) sin que haya ningún ciclo de separación entre el último operando de un cálculo y el primero del siguiente (un ciclo menos de separación que en el caso anterior).

El enunciado completo es:

```
If (Begin(c) == 1) {
```

$W(c+5)_u = (X(c+1)_u + X(c+2)_u + X(c+3)_u + X(c+4)_u) \bmod 2^n$;

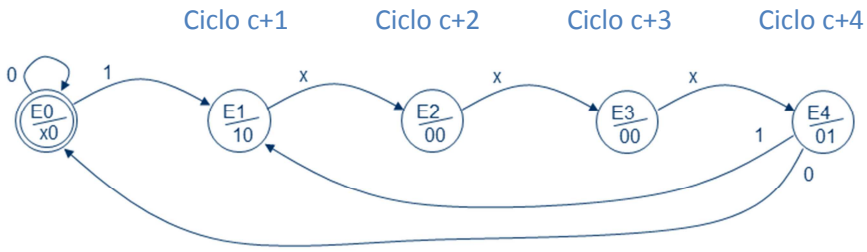
```
   End(c+4) = 1;
}
```

Además:

- La señal End valdrá 0 el resto de ciclos.
- Se ignorará el valor de la señal Begin durante los ciclos c+1 al **c+3** ambos incluidos.

Resultado:

El grafo es igual al del caso anterior excepto que ahora del estado E4 salen dos arcos con destinos diferentes según Begin valga 0 (se pasa al nodo E0) o 1 (se pasa a E1).



Caso 1e

Enunciado:

La sincronización es como en el caso anterior excepto que ahora Begin no se debe ignorar en ningún ciclo. Si durante el cálculo de la suma de cuatro números (en uno de los ciclos c+1, al c+3, ambos incluidos) Begin vale 1, se debe abortar el cálculo en curso y se debe iniciar un nuevo cálculo con la secuencia que comenzará a llegar por X al ciclo siguiente. El enunciado completo:

```
If (Begin(c) == 1) {
```

$W(c+5)_u = (X(c+1)_u + X(c+2)_u + X(c+3)_u + X(c+4)_u) \bmod 2^n$;

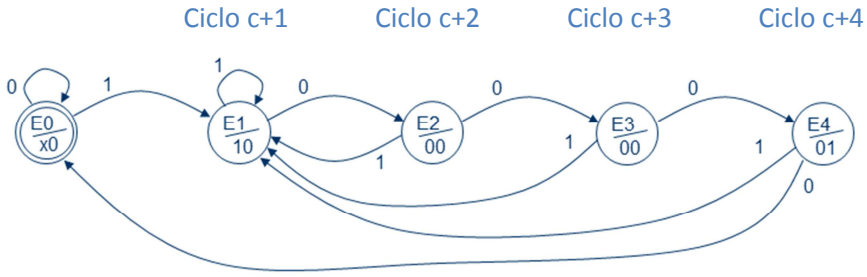
```
   End(c+4) = 1;
}
```

Además:

- La señal End valdrá 0 el resto de ciclos.
- **No se ignorará el valor de la señal Begin durante ningún ciclo. Si Begin vale 1 durante el cálculo de la suma de cuatro números (ciclos c+1 al c+3 ambos incluidos) se abortará el cálculo en curso y se comenzara el nuevo cálculo con el primer dato que llegará al ciclo siguiente.**

Resultado:

El grafo es igual al del caso anterior excepto que ahora la etiqueta de los arcos que antes tenían x ahora es 0 y que de los estados E1, E2 y E3 sale un arco con etiqueta 1 hacia el estado E1.



Caso 2a

Enunciado:

La sincronización se diferencia de todos los casos anteriores en que ahora el protocolo de comunicación es diferente a los anteriores (y es el mismo tanto para la entrada como para la salida):

- Begin a 1 valida el primer operando, que está presente en X en ese mismo ciclo.
- End a 1 valida el resultado, que está presente en W en ese mismo ciclo.

Este es el tipo de protocolo síncrono que usamos más asiduamente en otros ejemplos y ejercicios. El enunciado completo es:

```
If (Begin(c) == 1) {
   W(c+4)u = ( X(c)u + X(c+1)u + X(c+2)u + X(c+3)u) mod 2^n ;
   End(c+4) = 1;
}
```

Además:

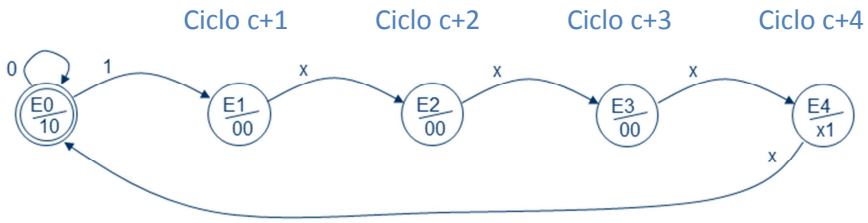
- La señal End valdrá 0 el resto de ciclos.
- Se ignorará el valor de la señal Begin durante los ciclos c+1 al **c+4** ambos incluidos.

Resultado:

En el estado inicial, E0, se espera a que Begin valga 1 para pasar al estado siguiente, E1. La UC se encuentra en el estado E1 en el ciclo c+1 (ya que en el ciclo anterior, que de acuerdo con el enunciado denominamos ciclo c, la UC estaba en el estado E0 y Begin valía 1). Así, en el ciclo c+1 en el bus de entrada X se encuentra el segundo operando de la suma: el primer operando se encontraba en X en el ciclo anterior, c. Por ello, la UC no puede esperar a poner la señal In/Add a 1 en el estado E1 para cargar el primer operando de la suma en el registro, porque éste operando ya no está en X en este ciclo: tiene que hacerlo en el ciclo anterior, en el estado E0.

Como estamos diseñando una UC de Moore, la salida en un estado, en concreto en el E0, tiene que ser la misma independientemente del valor de la entrada Begin. Por ello, en E0 se debe cargar el valor del bus X en el registro acumulador: por si Begin vale 1 en este ciclo, para que al pasar al estado E1 ya se haya cargado (al final del ciclo c) el primer operando en el registro. Esto se consigue haciendo que la señal In/Add valga 1 en el estado E0. Si resulta que estando en E0 Begin vale 0, el dato que se carga en el registro al final del ciclo no es un dato valido para iniciar una secuencia de cuatro sumas, pero eso no tiene importancia ya que al ciclo siguiente estaremos en el mismo estado E0 y se volverá a cargar otro dato en el registro (chafando el anterior). Esto se repite hasta que Begin valga 1, que al ciclo siguiente la UC estará en el E1 y se sumará el segundo operando, que en este ciclo c+1 está en X, con el primero que ya está en el registro (se cargó al final del ciclo anterior, estado E0). Hacemos énfasis en esto porque es motivo de error habitual.

Por último, End vale 1 durante el ciclo en que se muestra el resultado por la salida W, que es el ciclo c+4, al ciclo siguiente de sumar el cuarto operando: en el estado E4.



Caso 2b

Enunciado:

La sincronización es como en el caso anterior excepto que ahora Begin no se debe ignorar en el ciclo c+4, con lo que con este enunciado se pueden sumar dos conjuntos de 4 datos con un ciclo menos de separación entre ambas secuencias de operandos que en el caso anterior. Esto es, ahora pueden enlazarse dos cálculos sin ningún ciclo de separación entre el último operando de una secuencia y el primero de la siguiente. En este sentido este caso es tan eficiente como el caso 1d.

El enunciado completo es:

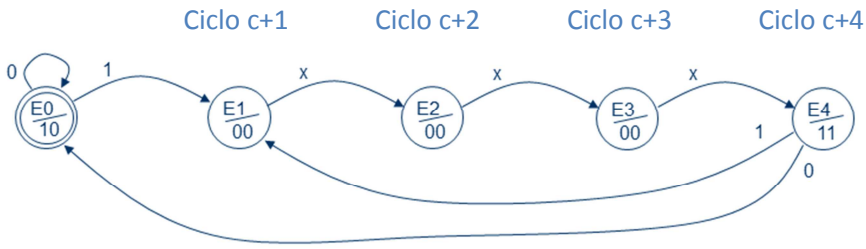
```
If (Begin(c) == 1) {
   W(c+4)u = ( X(c)u + X(c+1)u + X(c+2)u + X(c+3)u) mod 2^n ;
   End(c+4) = 1;
}
```

Además:

- La señal End valdrá 0 el resto de ciclos.
- Se ignorará el valor de la señal Begin durante los ciclos c+1 al **c+3** ambos incluidos.

Resultado:

El grafo es igual al del caso anterior excepto que ahora del estado E4 salen dos arcos con destinos diferentes según Begin valga 0 (se pasa al nodo E0) o 1 (se pasa a E1) y que la salida In/Add en E4 tiene que valer 1 ya que este nodo puede hacer las veces de E0 si Begin vale 1 (porque en este caso de E4 se pasa a E2).



Caso 2c

Enunciado:

En este caso se cambia el protocolo de la salida del resultado (que ya no es coherente con el de entrada) con el objetivo de poder tener un estado menos en el grafo que el caso 2a. Para ello, End valdrá 1 un ciclo antes de que la salida W muestre el resultado de las cuatro sumas.

Este caso, aunque tiene un estado menos que los dos anteriores, es igual de eficiente que el anterior, como vemos a continuación, ya que se puede iniciar una suma de cuatro números cada cuatro ciclos o lo que es lo mismo, entre dos cálculos consecutivos puede no haber ningún ciclo de separación entre el último operando de una secuencia y el primero de la siguiente. El inconveniente de este caso es que el protocolo de salida no es el mismo que el de la entrada:

- Begin a 1 valida el primer operando, que está presente en X en ese mismo ciclo.
- End a 1 valida el resultado, que está presente en W al ciclo siguiente.

El enunciado completo de este caso es:

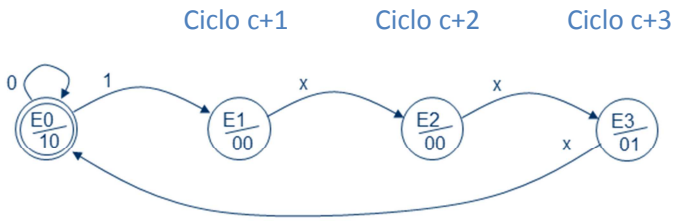
```
If (Begin(c) == 1) {
   W(c+4)u = ( X(c)u + X(c+1)u + X(c+2)u + X(c+3)u) mod 2n ;
   End(c+3) = 1;
}
```

Además:

- La señal End valdrá 0 el resto de ciclos
- Se ignorará el valor de la señal Begin durante los ciclos c+1 al **c+3** ambos incluidos.

Resultado:

El grafo es igual al del 2a pero sin el estado E4. Ahora End se pone a 1 en el estado E3.



Conclusiones

Hay muchos posibles protocolos síncronos para realizar la entrada de operandos y la salida de resultados de un PPE y para encadenar secuencias de cálculos. De todas las posibilidades que hemos visto, de entre las que no se aborta un cálculo para comenzar otro, usamos frecuentemente el caso 2b, ya que es uno de los más eficientes y usa el mismo protocolo para las entradas que para las salidas.

Otras posibilidades. Ejercicio propuesto

En este documento se han limitado el caso de abortar el cálculo actual, para iniciar uno nuevo cuando Begin valga 1 una vez comenzado un cálculo, a la sincronización concreta del ejemplo 1e. Se podría considerar el caso de abortar a cualquier otra sincronización de E/S pero para ello se requiere modificar la UP (que en este documento ha sido la misma para todos los casos), además de la UC. A continuación se propone un ejercicio en este sentido:

Dibujad la UP y el grafo de la UC para un PPE con la misma funcionalidad que el de este documento y con la sincronización del estilo del ejemplo 2b (en el ciclo en que Begin vale 1 el primer operando está en la entrada X y en el ciclo en que el resultado está en W la salida End debe valer 1) y que aborte el cálculo actual e inicie uno nuevo cada vez que Begin valga 1: En cualquier ciclo en el que Begin valga 1 se debe considerar que el valor presente en la entrada X en es el primer operando de un nuevo cálculo. Si el procesador estaba calculando se abortará el cálculo actual para iniciar el nuevo cálculo considerando que la entrada X en este ciclo en el que Begin vale 1 es el primer operando. Además, debéis escribir en C el protocolo de entrada/salida, que según la UP que uséis será igual o parecido al del ejemplo 2b.