## PROBLEMAS. LA UNIDAD DE CONTROL.

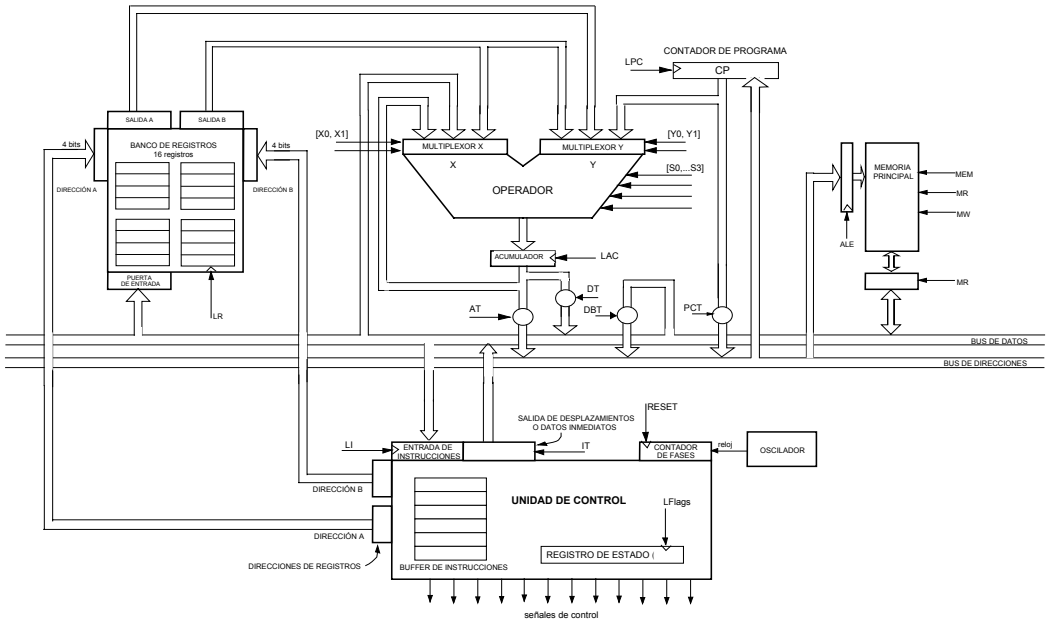
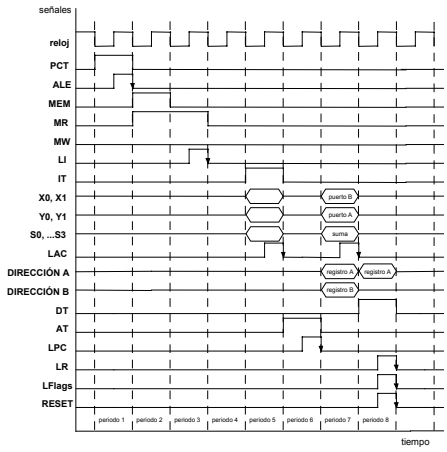



Figura 1.

Figura 2.




- 1. Sea el computador elemental de la figura 1. La instrucción ADD A, B tiene el cronograma de la figura 2 en dicho computador elemental.

  a) optimizar la temporización de señales para incrementar la velocidad de procesamiento.

  b) discutir qué cambios habría que introducir en el hardware para aumentar la velocidad de procesamiento.

- 2. Dibuja el cronograma correspondiente a la ejecución de la instrucción JZ ETIQUETA (saltar si cero a la posición ETIQUETA) en el computador de la figura 1.

- 3. Dibuja el cronograma correspondiente a SUB [DIRECCION], B.

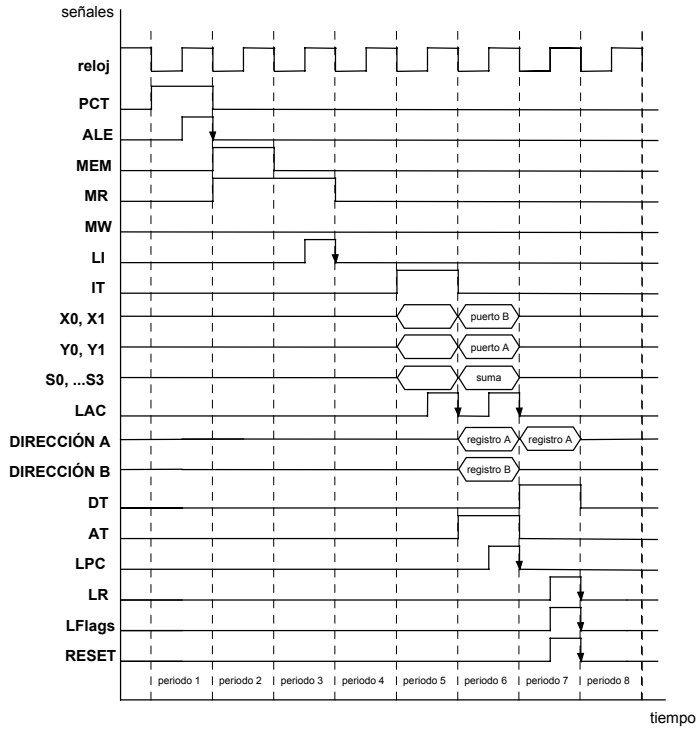
- 4. Diseñar una unidad de direccionamiento para el computador de la figura 1 que mejore su rendimiento.

- 5. Diseñar un computador elemental que cumpla las siguientes condiciones:
  - es capaz de ejecutar 4 instrucciones.
  - tiene una memoria de 64 palabras de 8 bits.
  - la ALU trabaja con operandos de 8 bits.
  - la ALU almacena la salida en un registro ACUMULADOR.
  - un operando de la ALU siempre es el ACUMULADOR.
  - el otro operando de la ALU proviene de la memoria.
  - el contador de programa (CP) solo se puede incrementar en la unidad.

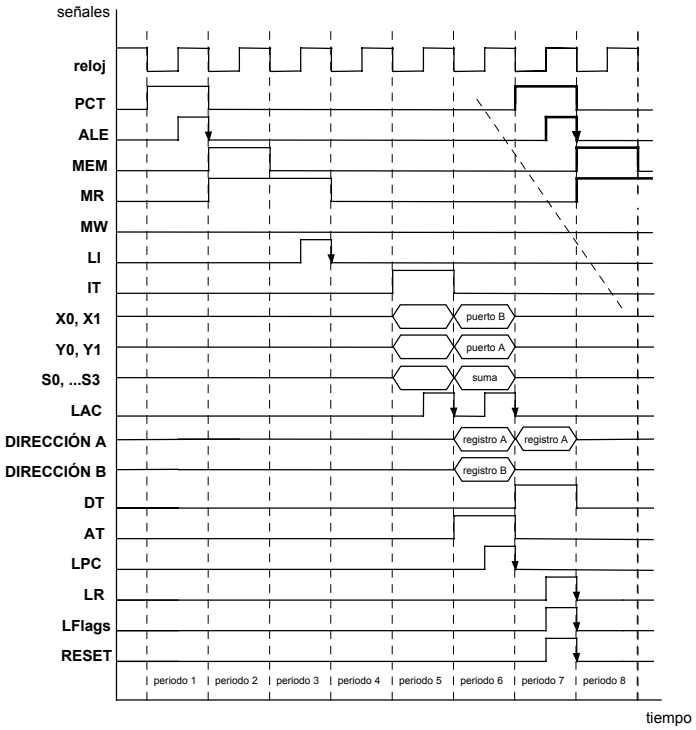
- **SOLUCIÓN A LOS PROBLEMAS.**

- **1**. a) sin cambiar el hardware:

1. ya que no hay conflicto entre las señales de control, el periodo 7 se puede superponer al 6:



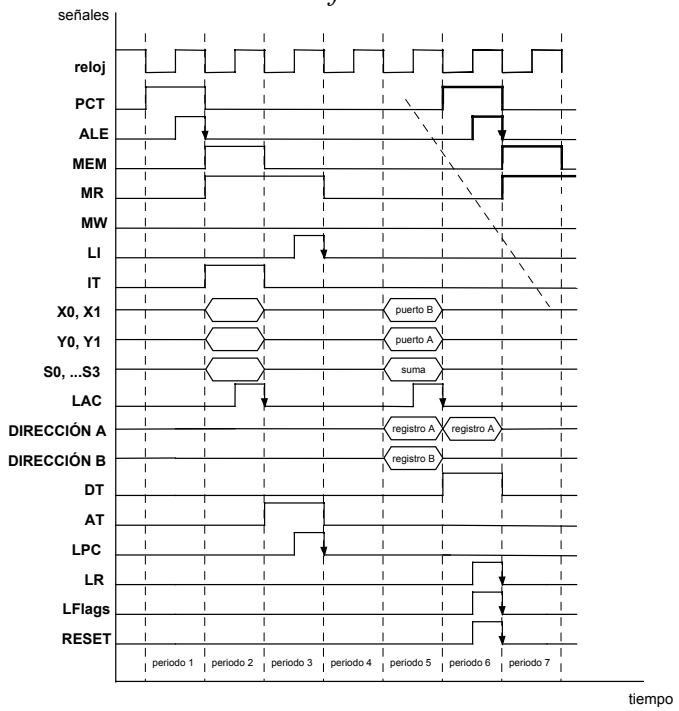

2. en el periodo 7 se puede comenzar una nueva fase de *fetch* ya que no se usa el bus de direcciones (en el periodo 6 se está usando todavía).




3. suponiendo que: - bus de datos y direcciones es de 16 bits.

- memoria organizada en palabras de 16 bits.
- tamaño estandar de instrucciones = 16 bits.

El incremento del contador de programa (CP) se puede realizar simultaneamente con la fase de *fetch*.

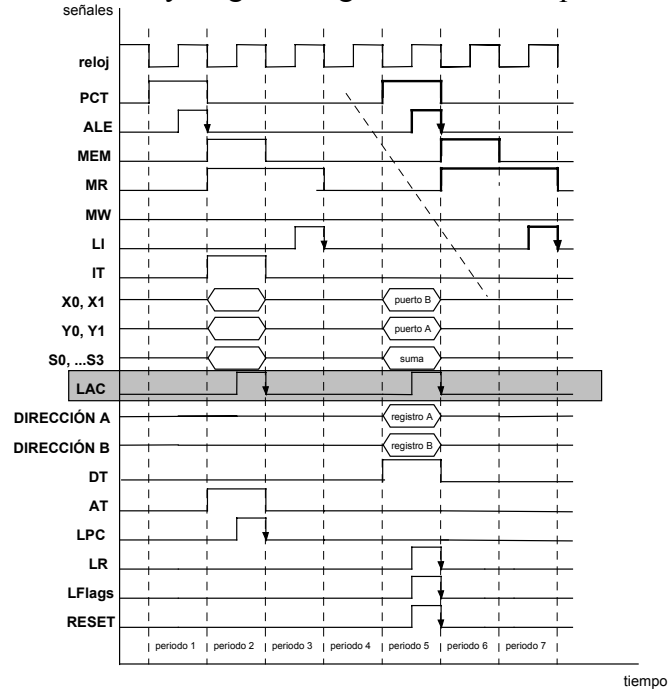



La fase de *fetch* de la siguiente instrucción no se puede superponer más ya que en el periodo 6 se está usando el bus de datos (habría conflicto entre las señales DT y MR).

b) cambiando el hardware:

Eliminando el ACUMULADOR a la salida de la ALU consigo:

- realizar la actualización del CP en un solo periodo.
- realizar la suma y carga del registro en un solo periodo.

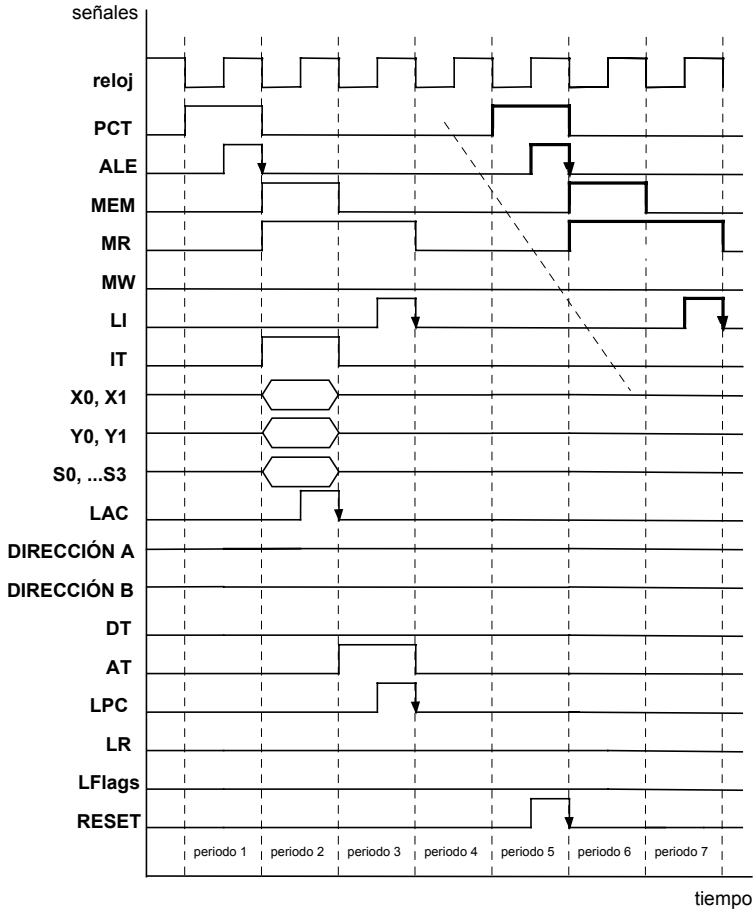


- **2**. Hacemos la suposición propuesta en el problema anterior en el punto 3:

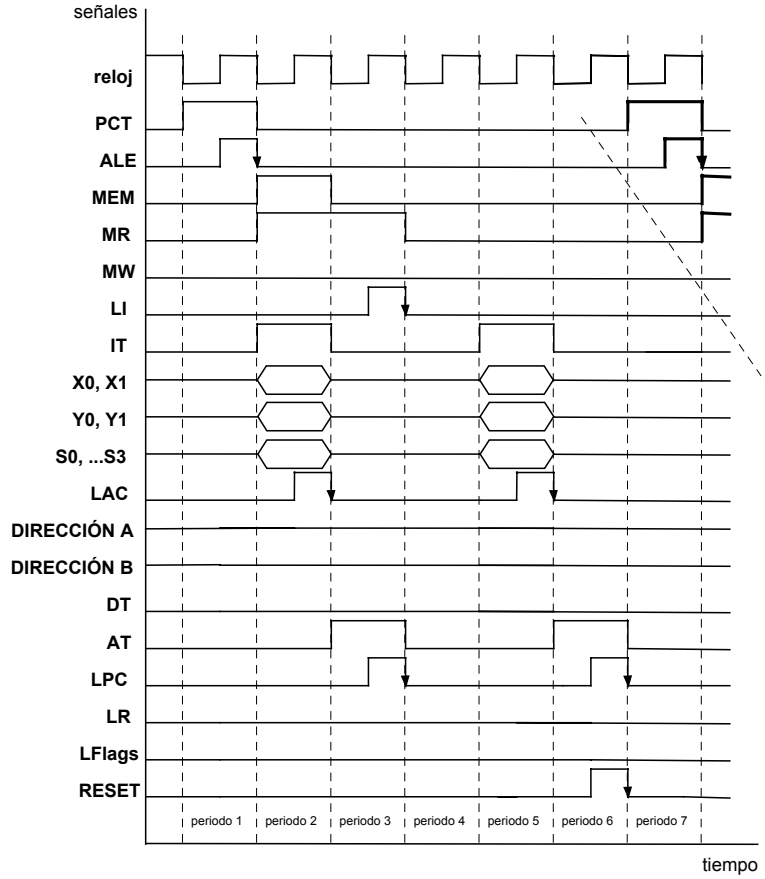
  - bus de datos y direcciones es de 16 bits.
  - memoria organizada en palabras de 16 bits.
  - tamaño estandar de instrucciones = 16 bits.

De forma que el incremento del contador de programa (CP) se pueda realizar simultaneamente con la fase de *fetch*.

La instrucción de salto condicional tiene dos cronogramas dependiendo de que se cumpla o no la condición. La comprobación de cumplimiento de la condición se hace durante la fase de decodificación de la instrucción. En esta fase la UNIDAD DE CONTROL compara la condición con el registro de estado y determina el estado de un bit. Este bit formará parte normalmente del código de operación de la instrucción de salto. El estado de este bit puede dar lugar a dos códigos de operación diferentes y por tanto a dos cronogramas distintos. Estos cronogramas son los dados a continuación:




**NO SALTA**




**SALTA**

Observese como en ambos casos se realiza una primera actualización del CP que apunta a la siguiente instrucción. Si se ha de realizar el salto se produce una reactualización del CP. Parece lógico que se haga una sola actualización del CP después de evaluar el cumplimiento de la condición pero se puede demostrar que es mejor apuntar en primera instancia a la instrucción consecutiva...

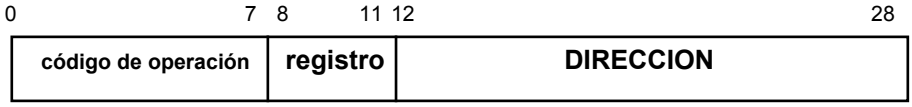
Las instrucciones de salto condicional siempre se dan por parejas, es decir, tenemos una condición y su contraria siempre. Por ejemplo, tenemos *saltar si cero* y *saltar si no cero*, etc. El buen programador utilizará la instrucción de salto de forma que se cumpla con **mayor probabilidad la condición que no produce salto**. De esta forma se incrementa la velocidad total de procesamiento.

Dada la forma de trabajar de estas instrucciones, los desplazamientos que llevan incluidos en su formato han descontado la longitud de la propia instrucción ya que se asume en la primera actualización del CP.

• **3**. La instrucción SUB [DIRECCION], B representa una resta donde uno de los operandos está en memoria. El modo de direccionamiento es directo absoluto:

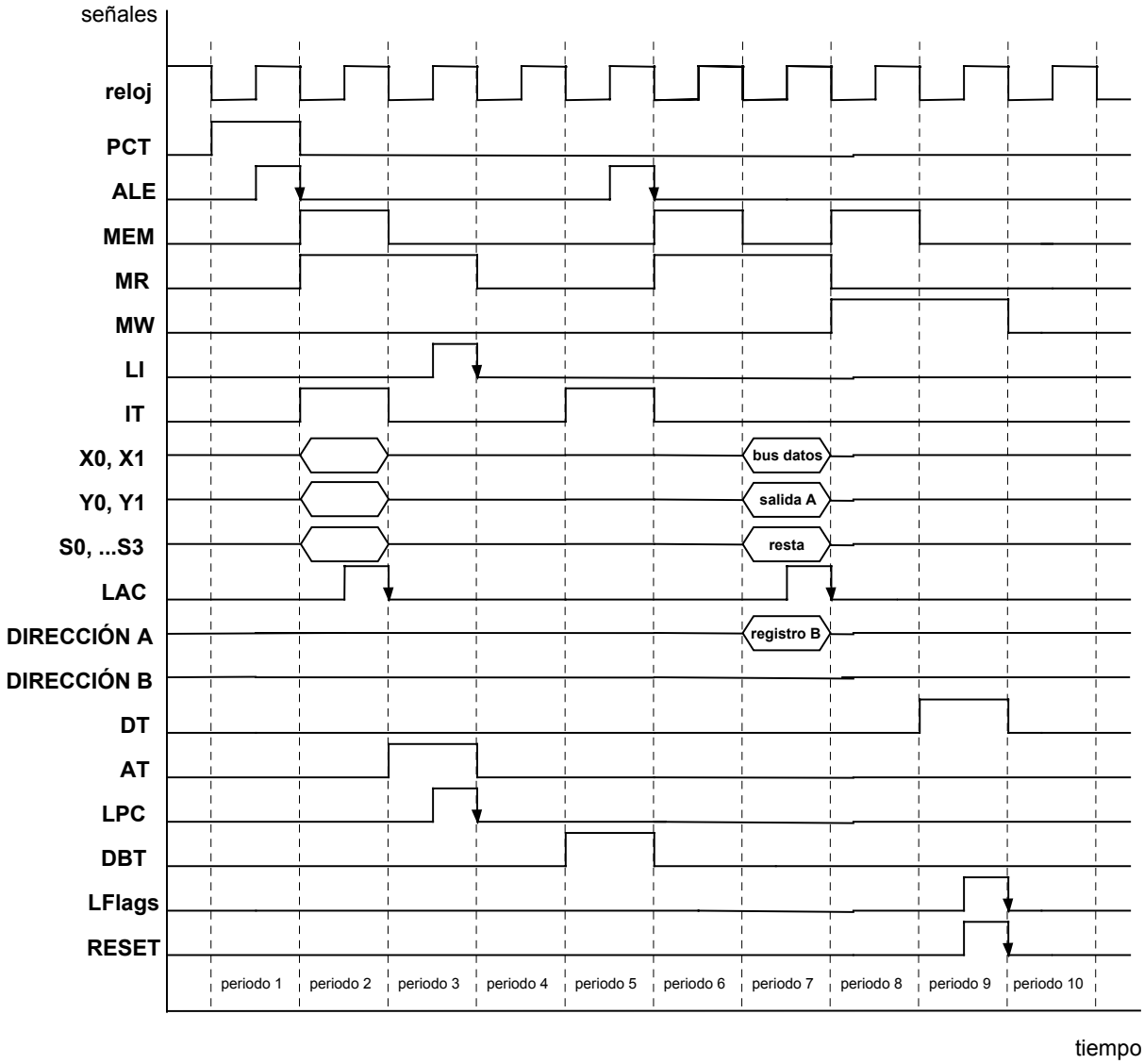
[DIRECCION] - B → [DIRECCION]

El formato de la instrucción podría ser:



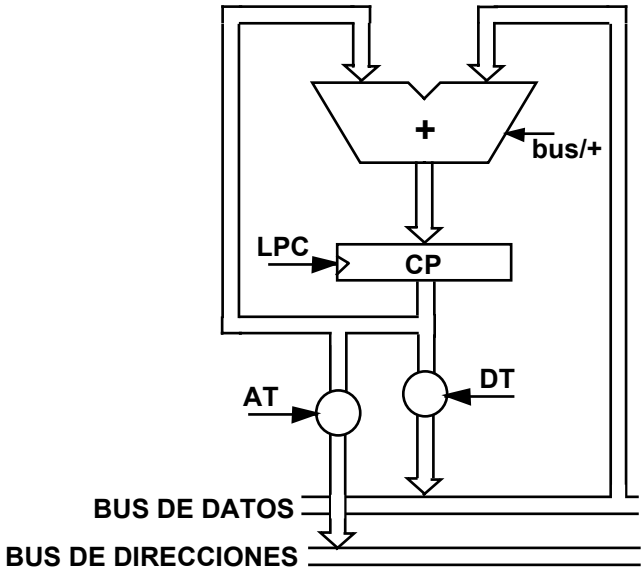

Las operaciones elementales a realizar son:
- cargar instrucción
- actualizar CP
- decodificación de la instrucción
- leer operandos (direccionar el de memoria y pasar a la ALU ambos)
- restar
- guardar resultado

| | | |
|---|---|---|
| **direccionar el operando de memoria**: | - la UC pone DIRECCION en el bus de datos<br>- se transfiere el contenido del bus de datos al de direcciones<br>- se carga el registro de direcciones | IT<br><br>DBT<br>ALE<br>(desactivar) |
| **realizar resta**:<br>• volcar contenido de memoria en bus datos<br><br>• pasar operandos a ALU<br><br><br><br><br>• restar | <br><br>- ciclo de memoria de lectura<br><br>- selección en X del operando del bus de datos<br>- la UC genera DIRECCIÓN A para apuntar al registro B<br>- selección de operando Y y operación<br><br>- carga del acumulador | <br><br>MEM<br>MR<br><br>X0, X1<br><br>4 bits<br>Y0, Y1 y S0, ...S3<br><br>LAC<br>(desactivar) |
| **guardar resultado**:<br>• direccionar memoria<br><br><br>• escribir en memoria | <br>- no requiere operación elemental (registro de dirección de memoria permanece cargado).<br>- transferancia al bus de datos<br>- ciclo de memoria de escritura | <br><br><br>DT<br>MEM<br>MW |
| **actualizar estado**: | | LFlags |
| **poner a 0 el contador de fases**: | | <br>RESET |




- periodo 4: decodificación.
- periodo 5: carga de dirección.
- periodo 6 y 7: volcado de memoria, selección de operandos y operación y carga del acumulador con el resultado.
- periodo 8: escritura del resultado en memoria.

• **4**. La mejora del rendimiento pasa por diseñar una unidad de direccionamiento que cuente con un operador de suma propio. Uno de los posibles diseños sería el de la figura:




La señal ***bus/***+ controla el operador de forma que realice sumas o deje pasar a la salida el contenido del bus de datos (operando de la derecha). De esta forma podemos actualizar el CP con un valor del bus de datos o con el contenido anterior del CP más un desplazamiento.

- **5**. El diseño debe ajustarse a las siguientes especificaciones:

- 4 instrucciones ⇒ código de operación de 2 bits.
- 64 palabras de memoria ⇒ dirección de 6 bits (bus de direcciones).
- operandos de 8 bits ⇒ bus de datos de 8 bits.
  - no se pueden sumar desplazamientos al CP solo se puede incrementar en la unidad.
  - cada instrucción tiene un campo de código de operación (2 bits) y un operando de 6 bits.
- no hay banco de registros.