TEMA 5

Sistemas en el nivel de transferencia entre registros (RT)

TEMA 5: Sistemas en el nivel de transferencia entre registros (RT)

RESUMEN:

En este tema se introducen los conceptos básicos de organización, descripción, análisis y síntesis de sistemas digitales en el nivel de transferencia a registros (RT), en los que un procesador es un ejemplo más. Estos sistemas se construyen utilizando la estrategia de dividirlos en dos bloques, uno el “Camino de datos” (Unidad de Procesamiento UP) y el otro la “Unidad de Control UC”. Veremos la misión de estos dos bloques y su interacción. Un aspecto importante es cómo capturar la especificación de estos sistemas de modo que sea fácil abordar su síntesis. Para ello, se introducirán métodos gráficos basados en las cartas ASM. Por último, se aborda el diseño de un computador sencillo.

El tema se complementa con la práctica P7, donde se describirá a nivel RT una realización en LogicWorks del computador sencillo CS1, para experimentar con él y analizar su funcionamiento.

TEMA 5: Sistemas en el nivel de transferencia entre registros (RT)

***OBJETIVOS* (Expresados como resultados de aprendizaje).**

Objetivos.

- Conocer la organización de los sistemas diseñados en el nivel de transferencia de registros, comprendiendo la misión del camino de datos y de la unidad de control, y su interacción.

- Deducir las operaciones de transferencia entre registros que puedan realizarse en un camino de datos dado.

Junto con la práctica P7, contribuye a:

- Descripción de las operaciones de un computador en el nivel de transferencia entre registros.

TEMA 5: Sistemas en el nivel de transferencia entre registros (RT)

5.1 Introducción y definiciones generales.

5.2 Unidad de procesamiento o camino de datos. Ejemplos de operaciones.

5.3 Unidad de Control. Ejemplos de generación de señales de control.

5.4 Ejemplo de un computador sencillo a nivel RT.

TEMA 5: Sistemas en el nivel de transferencia entre registros (RT)

Transparencias: Pedro Martín Smith.

BIBLIOGRAFÍA:

- **[DIA09]:** Estructura y Tecnología de Computadores. Teoría y problemas. **Autor/es:** Sergio Díaz Ruiz, María del Carmen Romero Ternero, Alberto J. Molina Cantero. **Más info:** *McGraw-Hill, 2009. y* **Apuntes de:** Sergio Díaz Ruiz.
- **[GRE86]:** Modern Logic Design. **Autor/es:** *Green, D.* **Más info:** *Addison Wesley, 1986.*
- **[GAJ97]:** Principios de Diseño Digital. **Autor/es:** *Gajski, D.* **Más info:** *Prentice Hall, 1997.*
- **[MAN05]:** Fundamentos de diseño lógico y de computadoras. 3ª edición. **Autor/es:** *M. Morris Mano; Charles R. Kime.* **Más info:** *Pearson Prentice Hall, 2005*
- **[CAPILANO]** Logic Works 4. CAPILANO COMPUTING SYSTEMS LTD**.** Addison Wesley.
- **[HAYES]:** Introducción al diseño lógico digital. **Autor/es:** *John P. Hayes.* **Más info:** *Addison-Wesley Iberoamericana, 1996.*

TEMA 5: Sistemas en el nivel de transferencia entre registros (RT)

5.1 Introducción y definiciones generales.

5.2 Unidad de procesamiento o camino de datos. Ejemplos de operaciones.

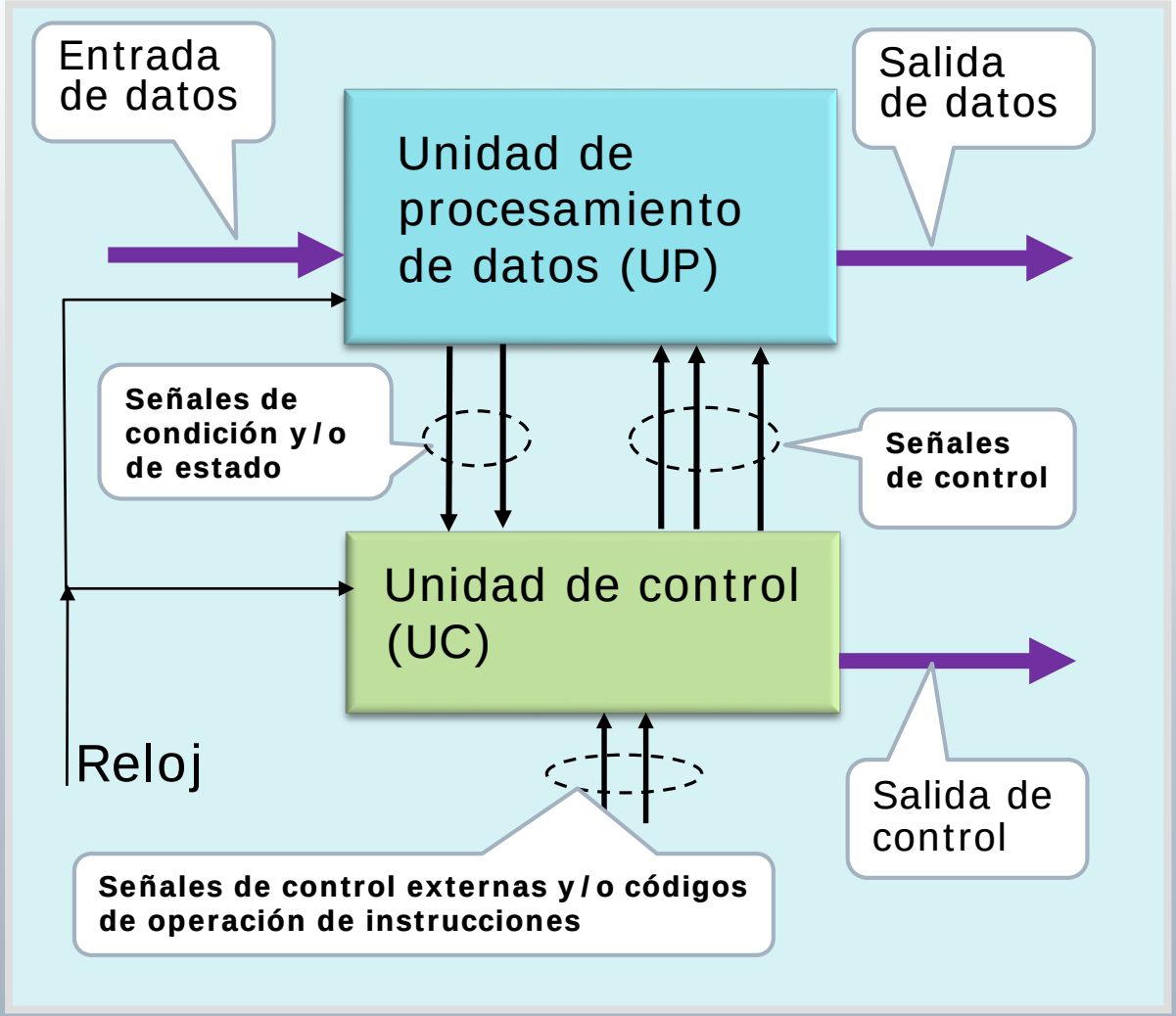
5.3 Unidad de Control. Ejemplos de generación de señales de control.

5.4 Ejemplo de un computador sencillo a nivel RT.

Niveles de complejidad en la descripción de un sistema digital.

NIVEL	COMPORTAMIENTO	COMPONENTES ESTRUCTURALES
Sistema	Algoritmos	Lenguaje máquina y ensamblador
Procesador	Instrucciones máquina	Procesadores, controladores, memorias, ASIC
Registro	Algoritmos Diagramas de flujo Cartas ASM	ALUs, MUXs, DEMUXs, registros, contadores, memorias
Puertas lógicas	Ecuaciones booleanas Tablas de estado	Puertas lógicas y biestables
Electrónico	Ecuaciones diferenciales Diagramas corriente-tensión	Transistores, resistencias, condensadores
Físico	Layout y modelos	Difusiones P,N, pistas de metal, polisilicio

Modelo de un sistema en el nivel de transferencia a registros (RT)



Operación elemental de Transferencia entre registros. Concepto de "Microoperación"

Una Microoperación es una operación elemental entre registros que se realiza en un ciclo de reloj,
Y consiste en:

1.1 Seleccionar las palabras almacenadas en un conjunto de registros o presentes en un terminal (bus) R1, R2, ..., Rk.

1.2 Transformar las palabras presentes en R1, R2, ..., Rk mediante circuitos combinacionales apropiados, tales como: operaciones lógicas entre palabras, sumadores, restadores, multiplicadores o ALUs, para producir uno o más resultados de salida. Se puede representar un resultado de la forma F(R1, R2, ..., Rk), donde F indica la función global o transformación realizada en este paso.

1.3. Almacenar o escribir los resultados en uno o más registros, entre los que se pueden incluir los registros originales R1, R2, ..., Rk.

LOS RESULTADOS SE EXPRESAN EN LA FORMA:

Resultado único

S←F(R1,R2,..RK);

Resultado Múltiple

S1←F1(R1,R2,..RK),
S2←F2(R1,R2,..RK),
.....
Sn←Fn(R1,R2,..RK);

Organización de un sistema de transferencia entre registros.

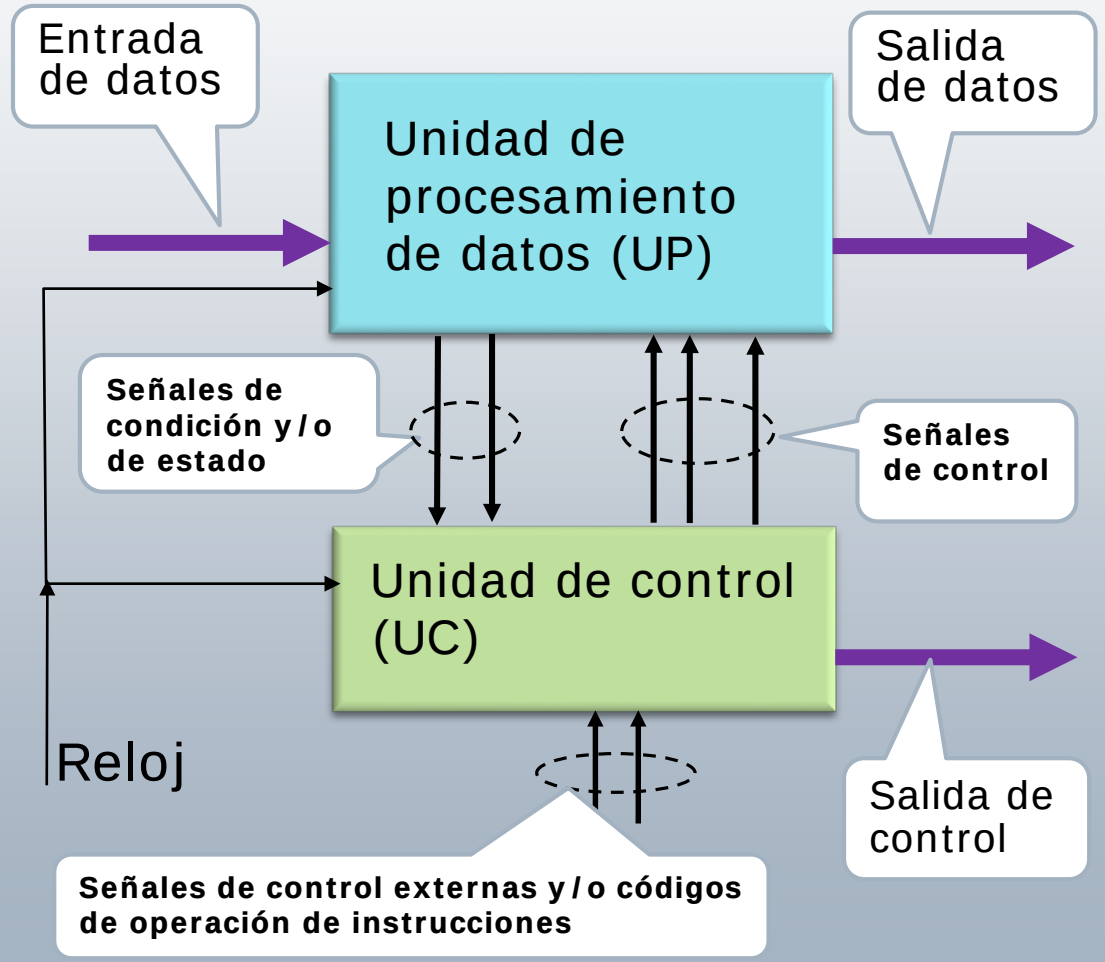
El Sistema RT ejecuta ordenes externas y/o códigos de operación (Macrooperaciones o instrucciones). Cada instrucción representa una “tarea” a realizar el sistema y cada “tarea” se consigue secuenciando durante un numero de pulsos de reloj operaciones elementales de transferencia de un ciclo máquina.

FUNCIONES DE LA UNIDAD DE PROCESAMIENTO

1) Almacenamiento de información.
2) Transformar información (Cálculo)
3) Conducir información.

FUNCIONES DE LA UNIDAD DE CONTROL

Generar en cada ciclo de reloj las señales de control apropiadas para secuenciar las operaciones elementales de la Unidad de Procesamiento, según la Instrucción a ejecutar.



TEMA 5: Sistemas en el nivel de transferencia entre registros (RT)

5.1 Introducción y definiciones generales.

5.2 Unidad de procesamiento o camino de datos. Ejemplos de operaciones.

5.3 Unidad de Control. Ejemplos de generación de señales de control.

5.4 Ejemplo de un computador sencillo a nivel RT.

El camino de datos

- Unidad del sistema donde se ejecutan las operaciones RT o microoperaciones

- Compuesto por
  - ✓ Componentes de almacenamiento
  - ✓ Circuitos de enrutamiento de datos
  - ✓ Unidades funcionales de procesamiento de datos

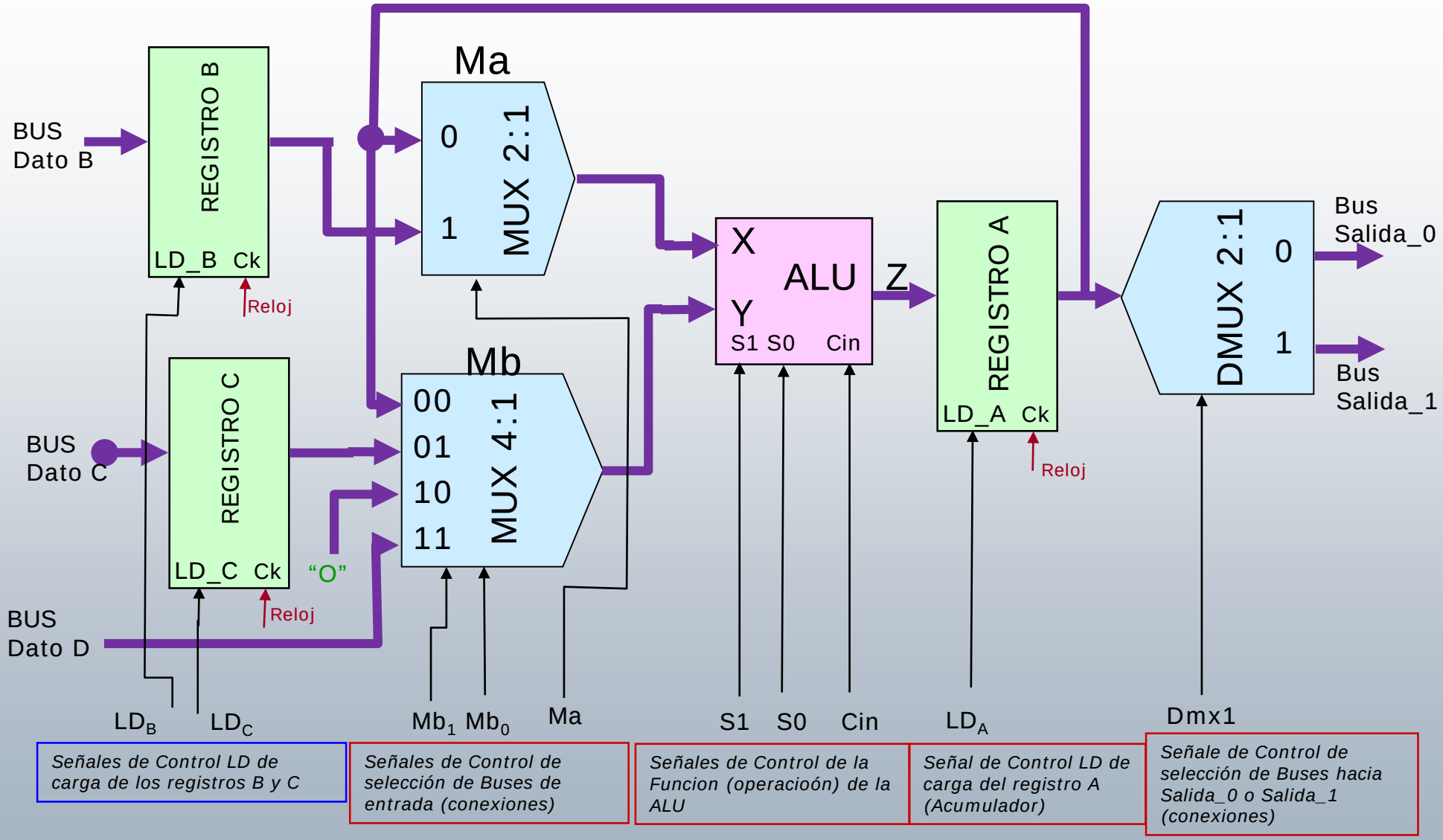
Componentes de almacenamiento.

- Registros
- Contadores
- Bancos de registros
- Memorias RAM, ROM, etc.

Módulos de procesamiento

- Circuitos lógicos
  - Puertas para palabras
  - Unidades lógicas
- Circuitos aritméticos
  - Sumadores.
  - Multiplicadores
  - Comparadores de magnitud
- Unidades aritmético-lógicas (ALUs)
- Desplazadores y rotadores

Ejemplo 1 de una unidad de procesamiento UP.



5.2. Unidad de procesamiento o camino de datos. Ejemplos de operaciones.

5.2.1 Módulos de enrutamiento (Enlaces y buses)

Modos de enlazar elementos de forma que sean posibles transferencias de información entre cualesquiera de ellos.

Componentes de control de los buses

- Demultiplexores/decodificadores
- Selectores de datos (multiplexores)
- Adaptadores tri-estado

Tipos de buses

BUSES DEDICADOS.

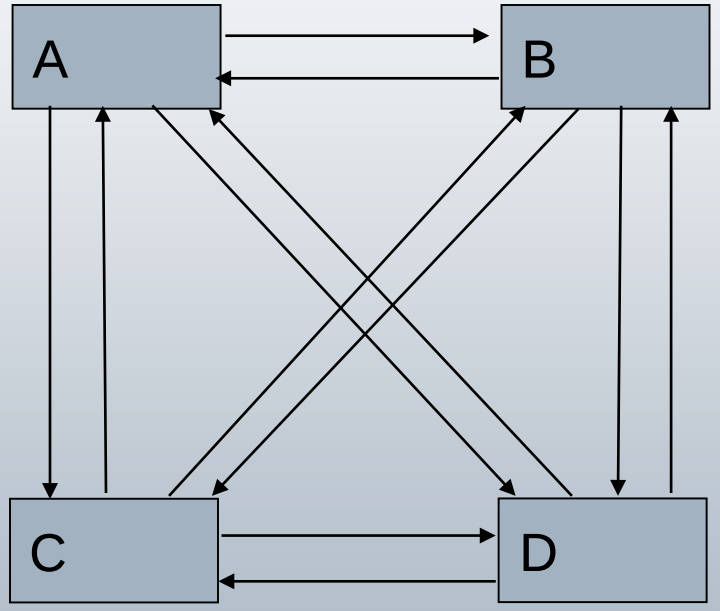
Una única fuente y destino para cada bus.

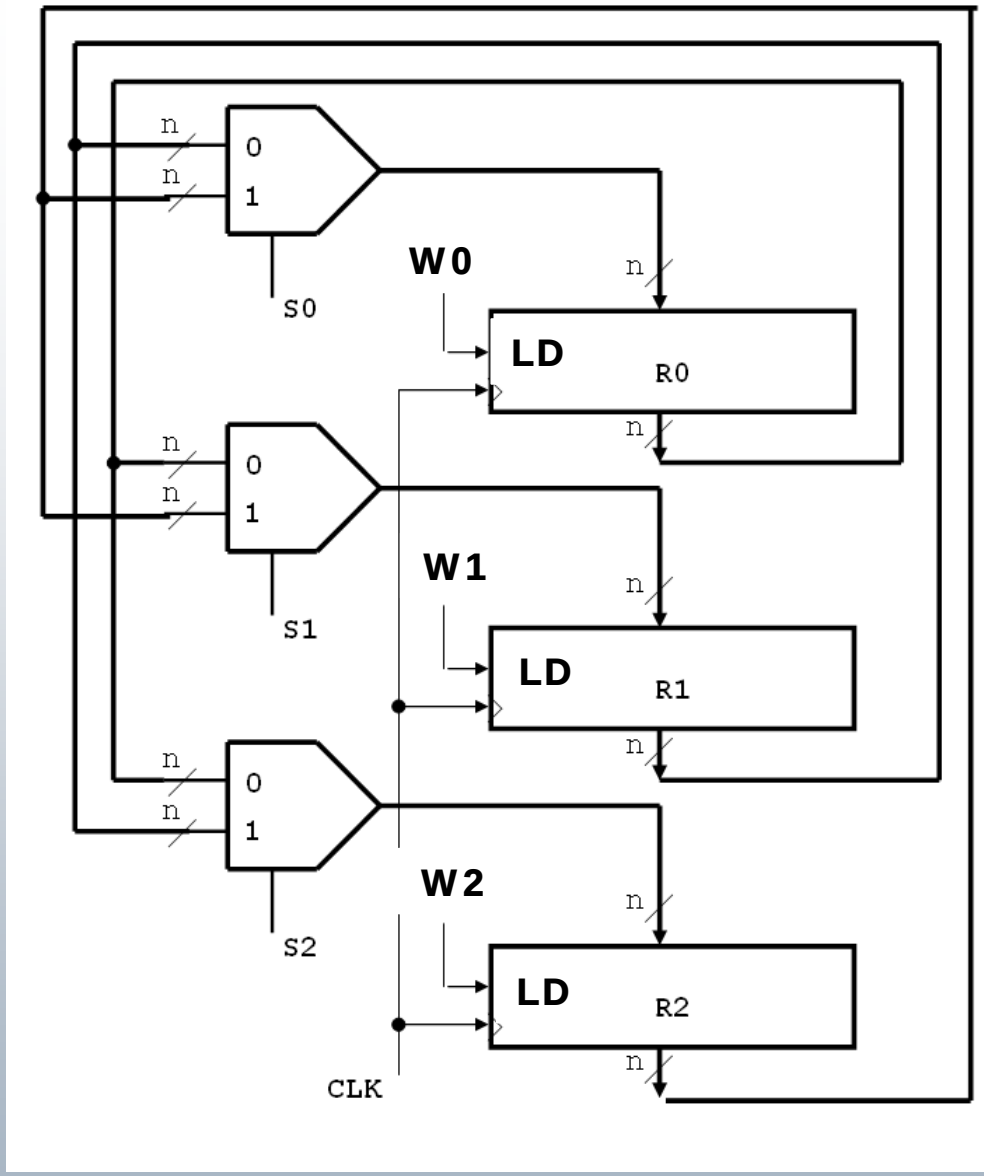
Desventajas:

1. Gran número de líneas de interconexión
2. Difícil ampliación.

Ventajas:

1. Posibilidad de transferencias simultaneas, implicando mayor velocidad.





EJERCICIO 1:

Con el esquema de transferencia basada en multiplexores con buses dedicados de la figura ¿qué operaciones de la siguiente tabla pueden efectuarse en un solo ciclo de reloj? Indicar los valores requeridos para las señales de control, en los casos en que proceda.

Operación RT	¿En un solo ciclo?	Señales de control					
		S0	S1	S2	W0	W1	W2
R0 ← R1							
R1 ← R2							
R2 ← R0							
R1 ← R2 R2 ← R0							
R0 ← R1 R1 ← R2 R2 ← R0							

NOTA: Basado en ejemplo propuesto en [MANO05], pág. 325

BUS COMPARTIDO (basado en adaptadores tri-estado)

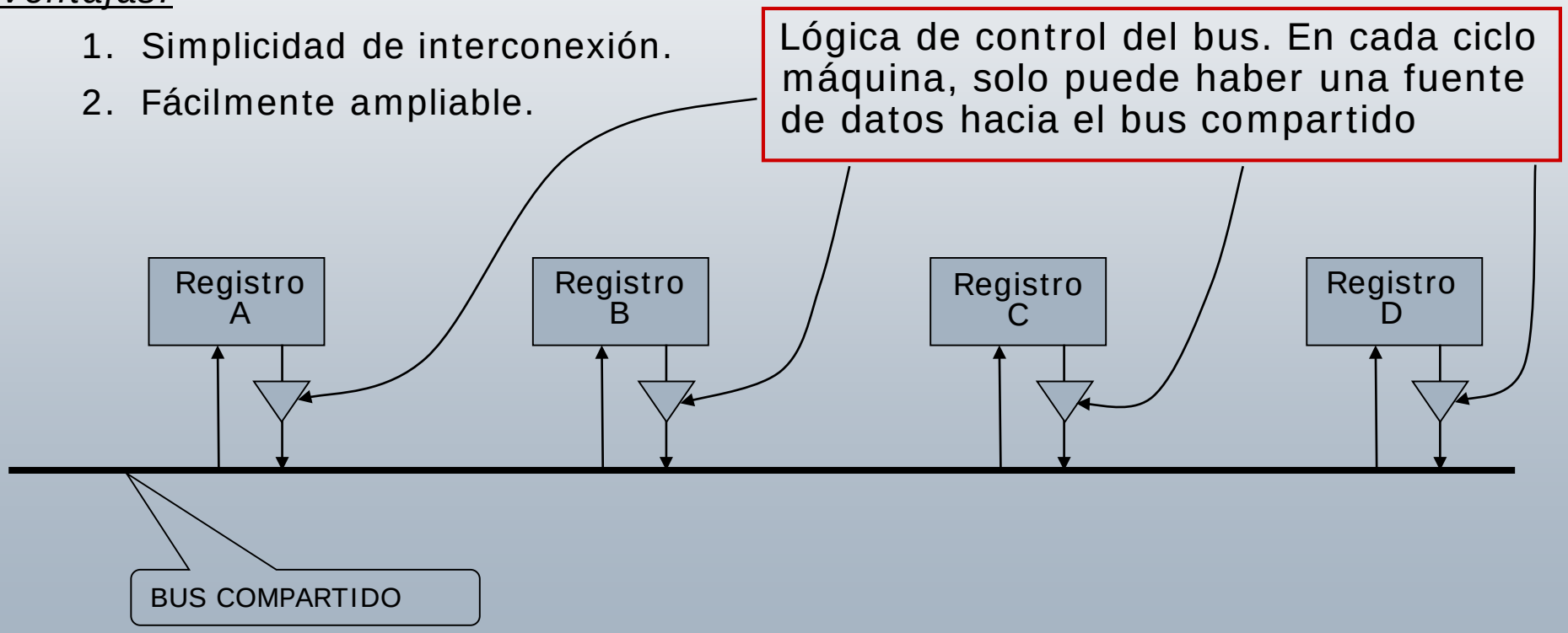
Bus enlazando diferentes fuentes y destinos.

Desventajas:

1. No son posibles transferencias simultaneas en donde se tengan más de una fuente.
2. Se requiere una lógica de control del bus más compleja.

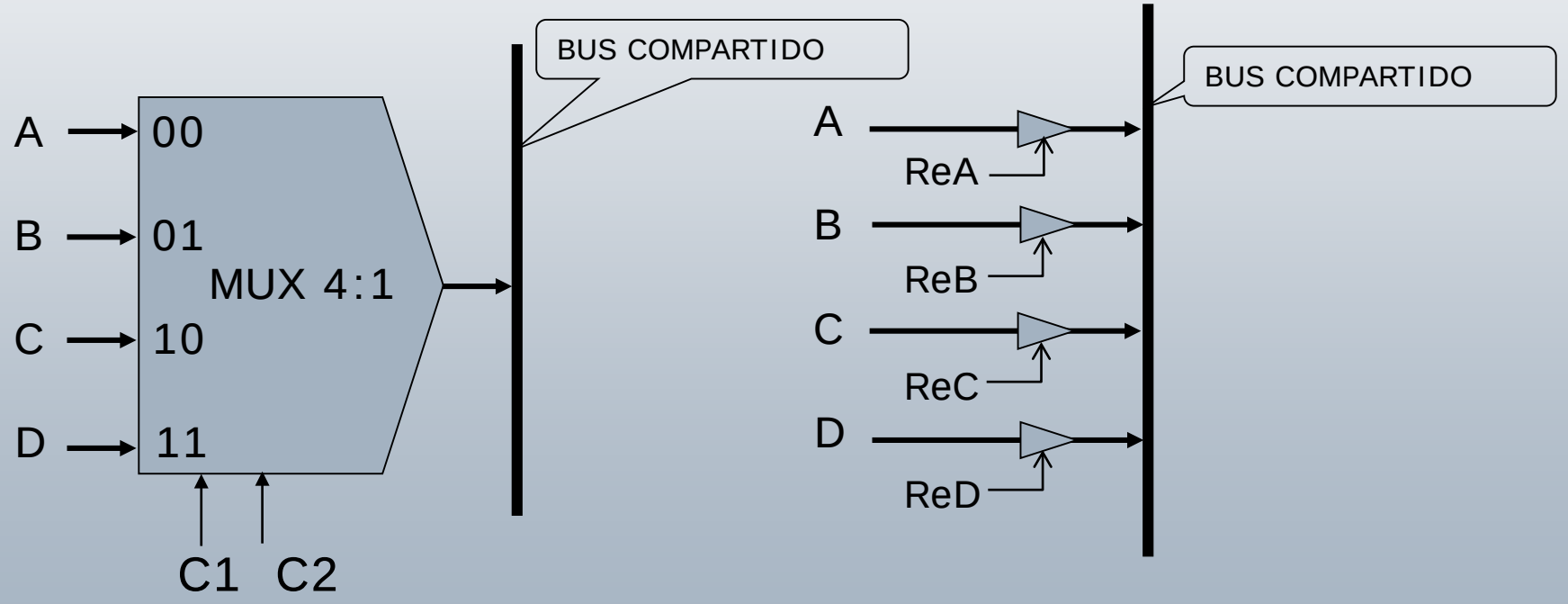
Ventajas:

1. Simplicidad de interconexión.
2. Fácilmente ampliable.



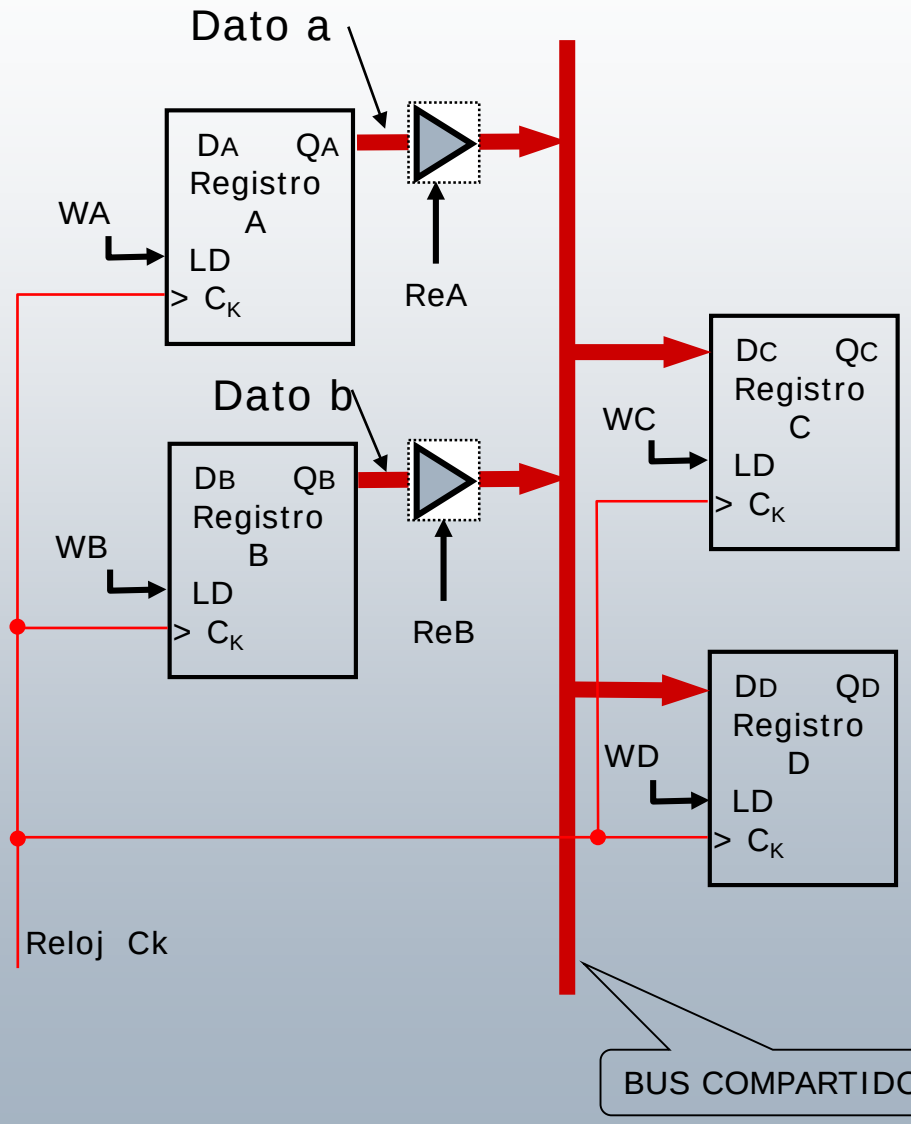
BUS COMPARTIDO: (Lógica de control mediante multiplexores o con conmutadores triestado

1. Multiplexores, Demultiplexores (como selectores de datos).
2. Conmutadores triestado (monodireccionales o bidireccionales).Puede que los componentes a conectar al bus ya incluyan la posibilidad de salidas ***triestado*** mediante una entrada de control “CHIP SELECT (Cs)” que sirve para la habilitación de salida (OE) que se asocia normalmente a una “LECTURA” de un bus o registro (***Read***), en al figura ***ReA, RaB, ReC, ReD***.

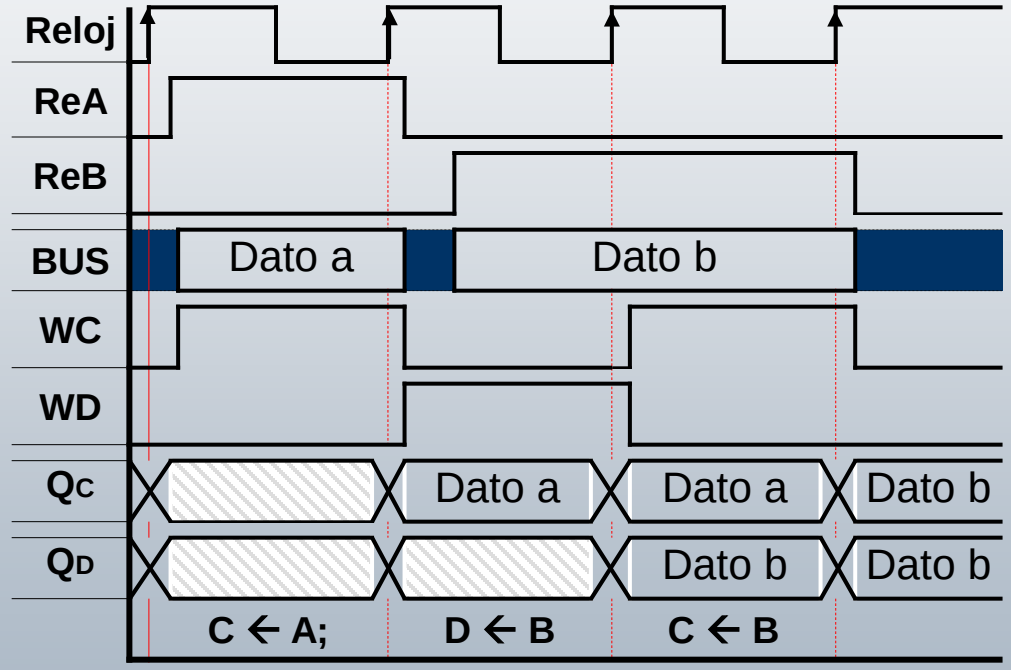


C1 C2 ← Señales de control del bus → ReA, ReB ReC, ReD

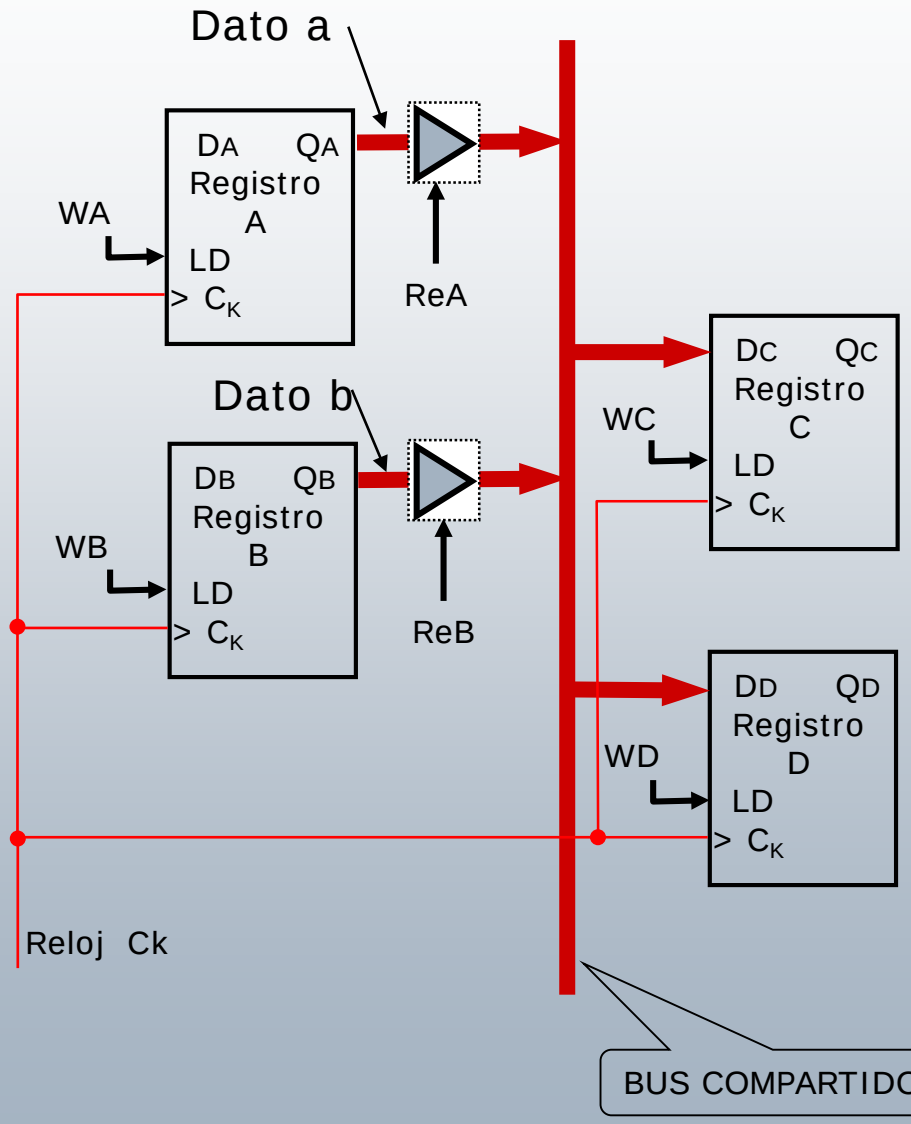
Ejemplo 2 de operaciones de transferencias a través de un bus compartido .



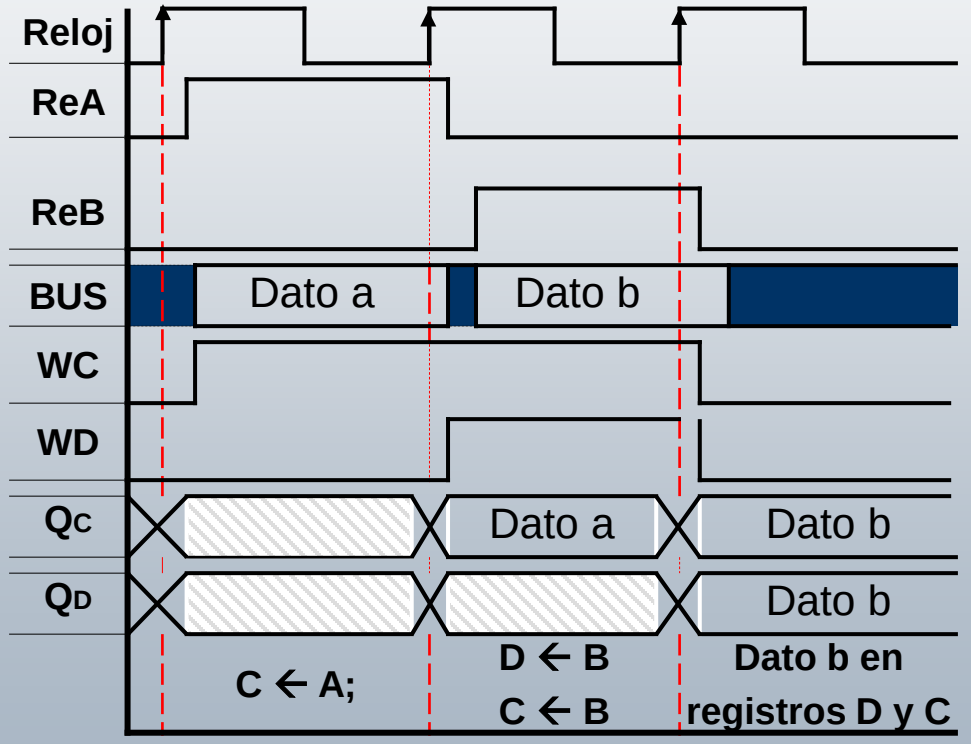
C ← A;
D ← B;
C ← B;

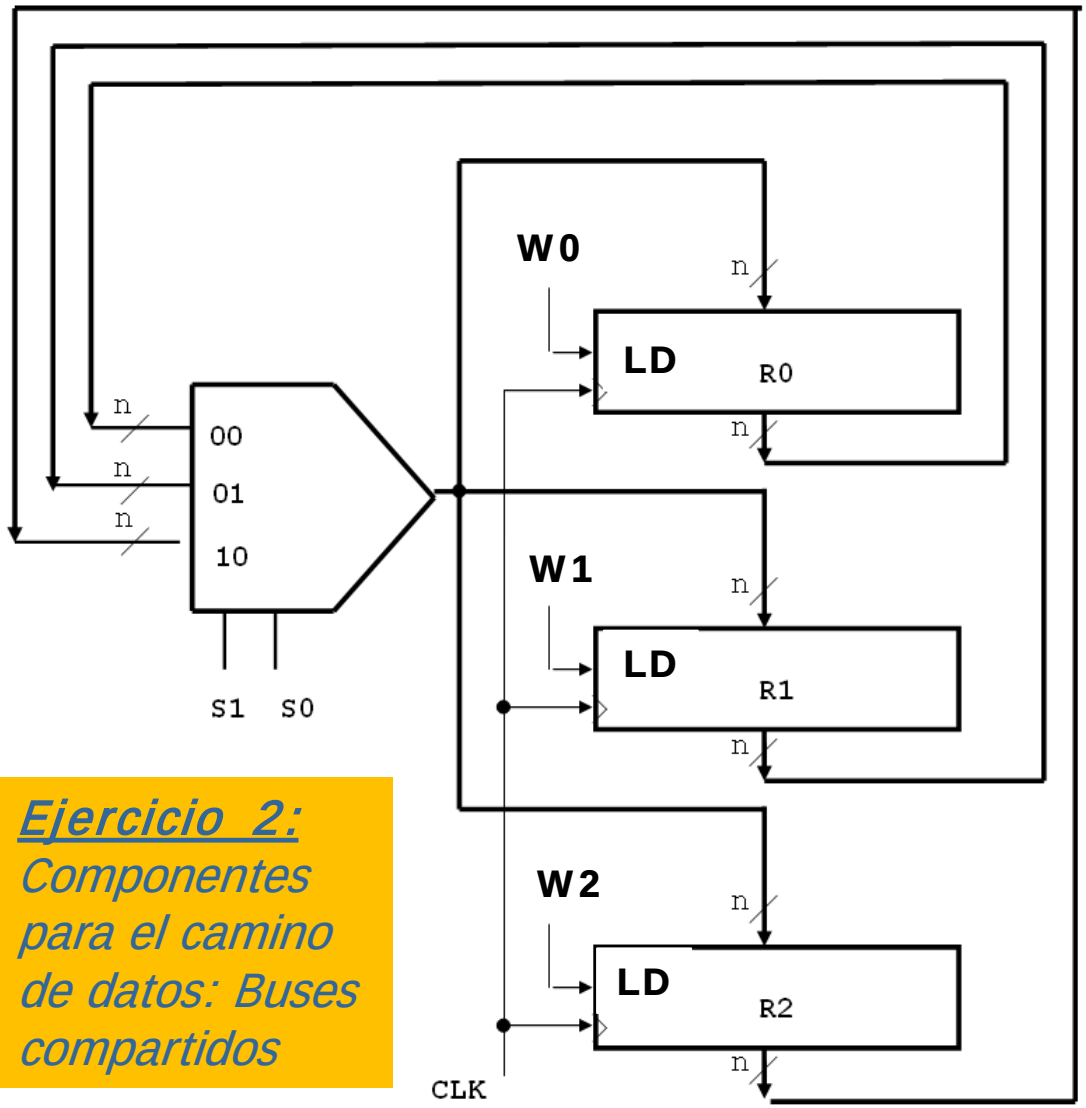


Ejemplo 2 de operaciones de transferencias a través de un bus compartido .



C ← A;
D ← B, C ← B;





EJERCICIO 2:
Con el esquema de transferencia basada en multiplexores con bus compartido de la figura ¿qué operaciones de la siguiente tabla pueden efectuarse en un solo ciclo de reloj? Indicar los valores requeridos para las señales de control, en los casos en que proceda.

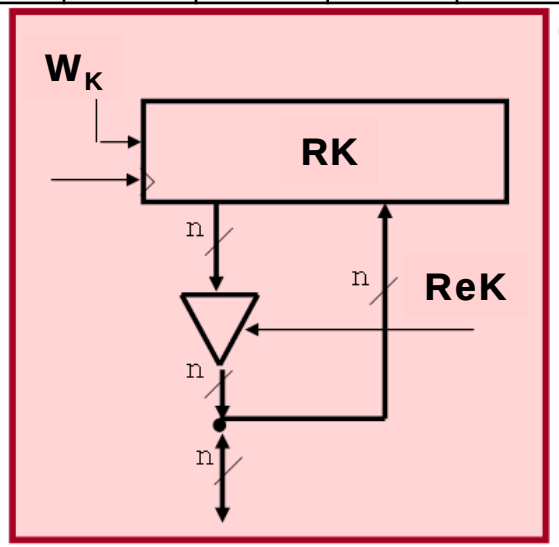
Operación RT	¿En un solo ciclo?	Señales de control				
		S1	S0	W0	W1	W2
R0 ← R1						
R1 ← R2						
R2 ← R0						
R1 ← R2 R2 ← R0						
R0 ← R1 R2 ← R1						

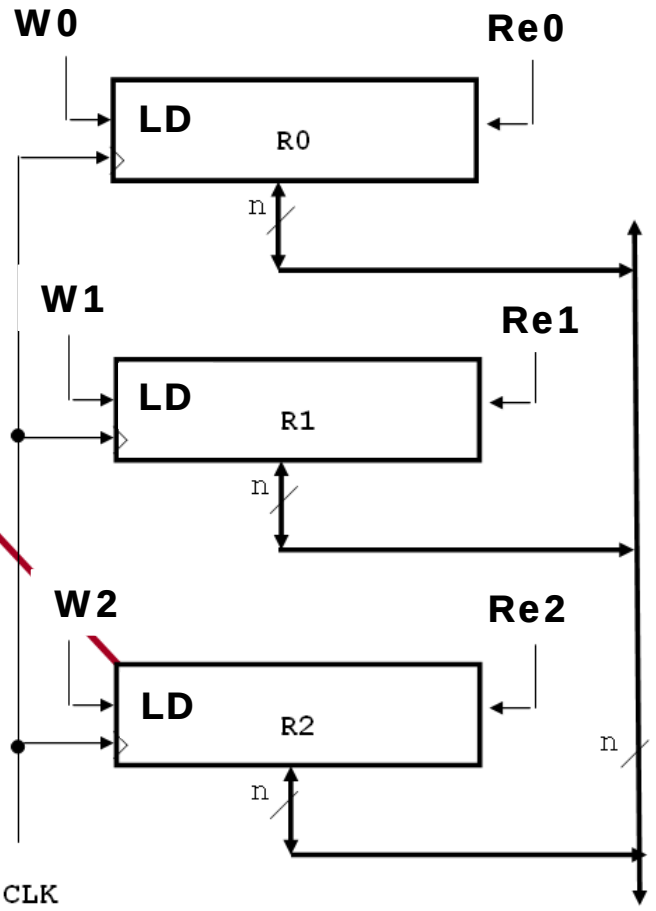
Ejercicio 2:
Componentes para el camino de datos: Buses compartidos

NOTA: Basado en ejemplo propuesto en [MANO05], pág. 325

EJERCICIO 3: Con el esquema de **transferencia basada en adaptadores triestado** de la figura ¿qué operaciones de la siguiente tabla pueden efectuarse en un solo ciclo de reloj? Indicar los valores requeridos para las señales de control, en los casos en que proceda.

Operación RT	¿En un solo ciclo?	Señales de control					
		Re0	Re1	Re2	W0	W1	W2
R0 ← R1							
R1 ← R2							
R2 ← R0							
R1 ← R2 R2 ← R0							
R1 ← R0 R2 ← R0							





NOTA: Basado en ejemplo propuesto en [MANO05], pág. 327

Ejercicio 3:
*Componentes para el camino de datos: **Buses (bidireccionales) compartidos***

5.2.2 Microoperaciones y palabra de control de un camino de datos. Tabla de microoperaciones.

- Se introduce el concepto de “ conjunto de microoperaciones de un camino de datos” como las operaciones RT básicas que éste puede hacer en un ciclo de reloj.

- Para averiguarlo es necesario tener en cuenta sus enlaces y la función de sus componentes descritos como operaciones de transferencia entre registros.

- Concepto de palabra de control, como las señales concretas de control de todos los componentes incluidos en el camino de datos y normalmente agrupados en campos según el componente afectado.
- Lo ilustramos con un ejemplo sencillo de camino de datos.

EJEMPLO 3 de CAMINO DE DATOS SENCILLO.

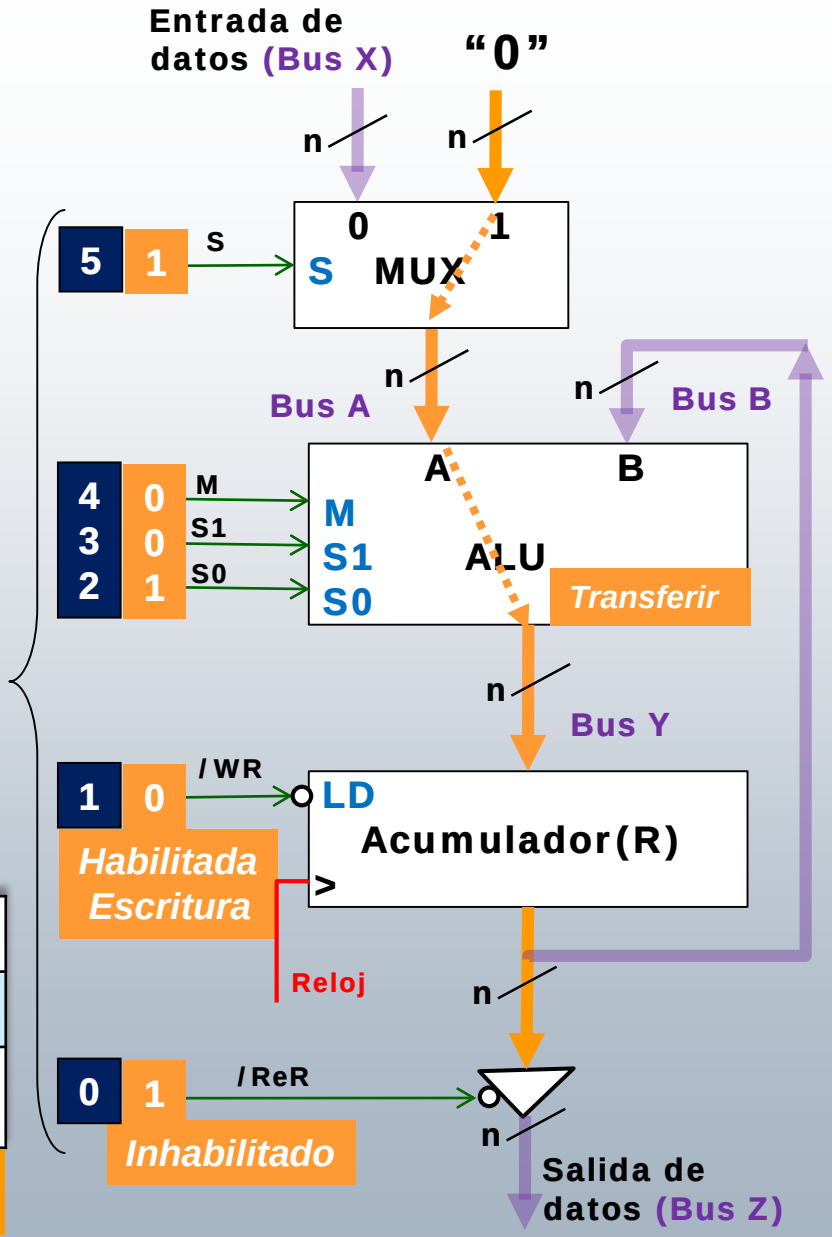
Se ilustran los conceptos de "Microoperaciones en un camino de datos y de "Palabra de control".

M	S_1	S_0	Operaciones de la ALU
0	0	0	Complementar A
0	0	1	Transferir A
0	1	0	NAND
0	1	1	XOR
1	0	0	Decrementar A
1	0	1	Sumar
1	1	0	Restar
1	1	1	Incrementar A

R ← '0'

Palabra de control					
5	4	3	2	1	0
Selección de entrada	Controles de la ALU			Escritura en acumulador R	Lectura del acumulador R
1	0	0	1	0	1

Entradas de control



NOTA: Basado en ejemplo propuesto en [GAJ97], pág. 311-313

EJEMPLO 3 de CAMINO DE DATOS SENCILLO.

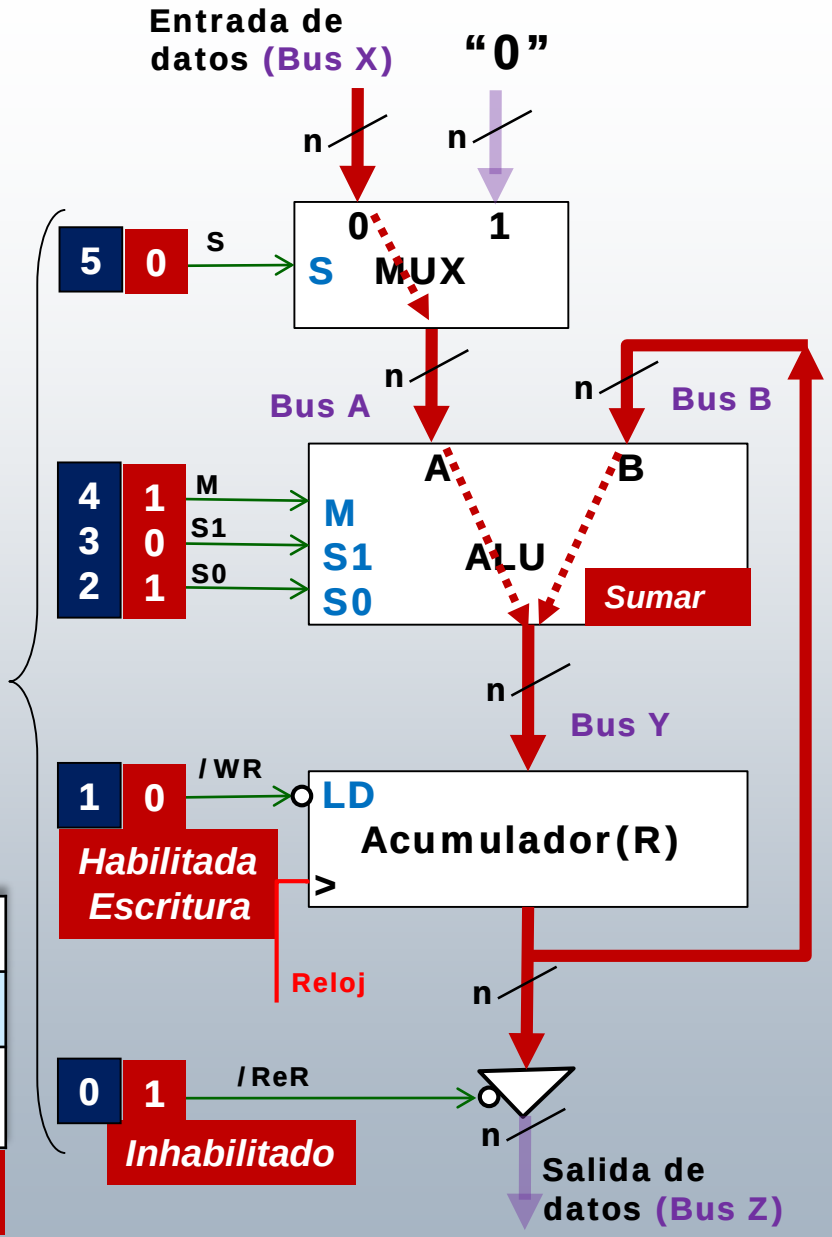
Se ilustran los conceptos de "Microoperaciones en un camino de datos y de "Palabra de control".

M	S_1	S_0	Operaciones de la ALU
0	0	0	Complementar A
0	0	1	Transferir A
0	1	0	NAND
0	1	1	XOR
1	0	0	Decrementar A
1	0	1	Sumar
1	1	0	Restar
1	1	1	Incrementar A

$R \leftarrow R + X$

Entradas de control

Palabra de control					
5	4	3	2	1	0
Selección de entrada	Controles de la ALU			Escritura en acumulador R	Lectura del acumulador R
0	1	0	1	0	1



NOTA: Basado en ejemplo propuesto en [GAJ97], pág. 311-313

EJEMPLO 3 de CAMINO DE DATOS SENCILLO.

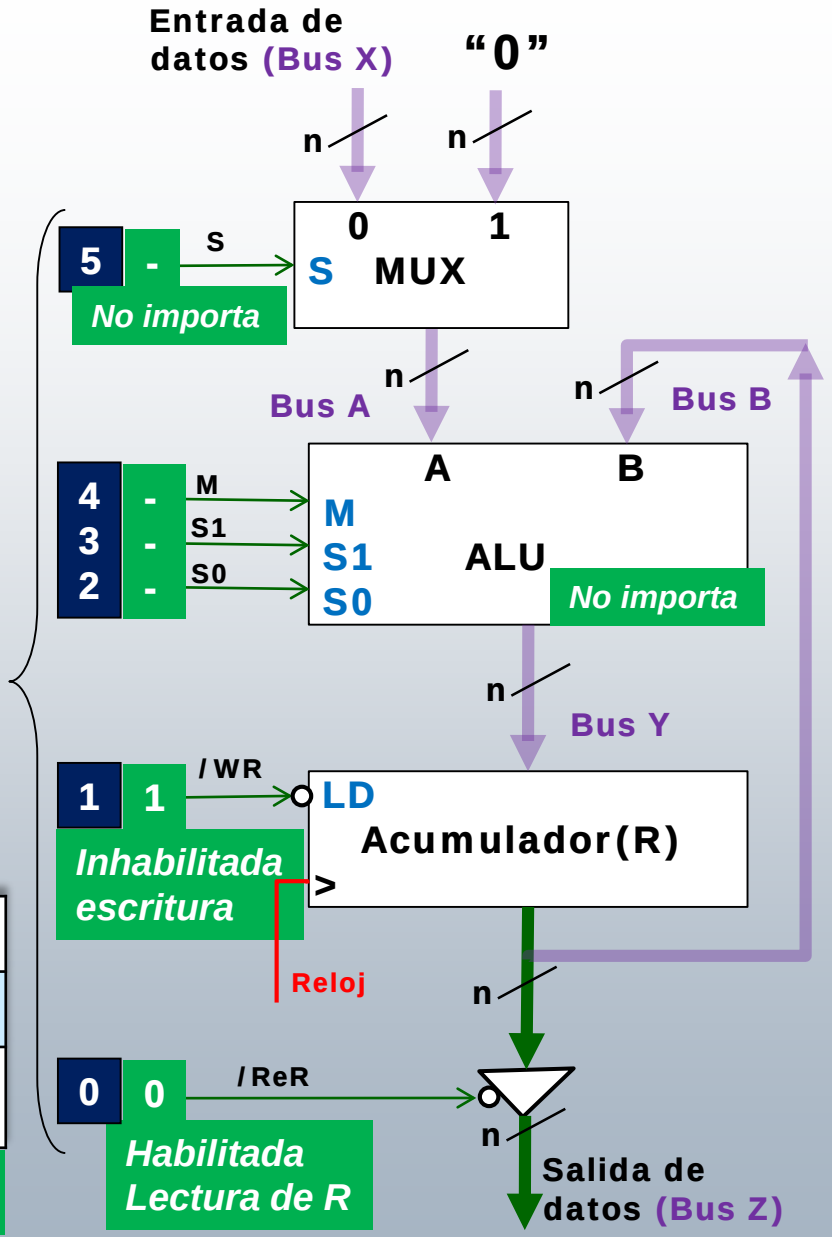
Se ilustran los conceptos de "Microoperaciones en un camino de datos y de "Palabra de control".

M	S_1	S_0	Operaciones de la ALU
0	0	0	Complementar A
0	0	1	Transferir A
0	1	0	NAND
0	1	1	XOR
1	0	0	Decrementar A
1	0	1	Sumar
1	1	0	Restar
1	1	1	Incrementar A

Entradas de control

$Z = R$

Palabra de control					
5	4	3	2	1	0
Selección de entrada	Controles de la ALU			Escritura en acumulador R	Lectura del acumulador R
-	-	-	-	1	0



NOTA: Basado en ejemplo propuesto en [GAJ97], pág. 311-313

EJEMPLO 3 de CAMINO DE DATOS SENCILLO.

Se ilustran los conceptos de “Microoperaciones en un camino de datos y de “Palabra de control”.

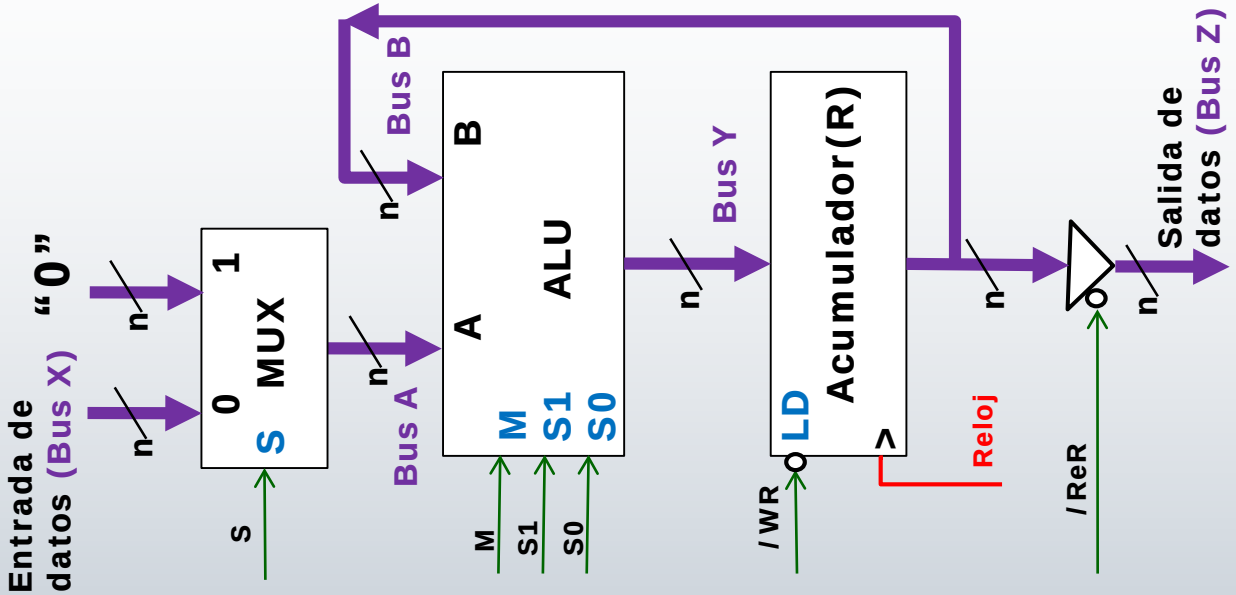
Para los ejemplos vistos anteriormente de microoperaciones RT y señales de control que deben activarse, se obtiene la siguiente tabla de microoperaciones:

Tabla de Microoperaciones RT - Palabras de control

Operaciones RT	PALABRA DE CONTROL (Señales de control)						
	Selección de entrada	Controles de la ALU			Escritura en acumulador R	Lectura del acumulador R	(En hexadecimal)
	S	M	S1	S0	/WR	/ReR	
R <--' 0'	1	0	0	1	0	1	25
R <-- R+ X	0	1	0	1	0	1	15
Z = R	x	x	x	x	1	0	02

NOTA: Basado en ejemplo propuesto en [GAJ97], pág. 311-313

Ejercicio 4: Deduzca otras microoperaciones que puede realizar la UP del Ejemplo 3 y complete la tabla que se indica.

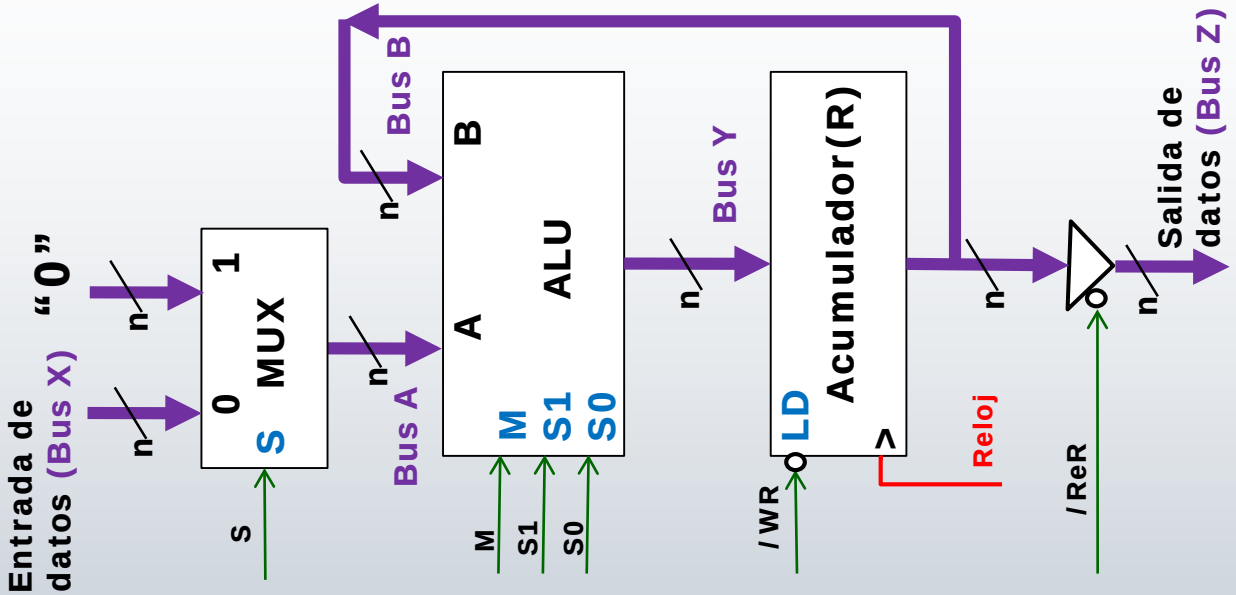


M	S_1	S_0	Operaciones de la ALU
0	0	0	Complementar A
0	0	1	Transferir A
0	1	0	NAND
0	1	1	XOR
1	0	0	Decrementar A
1	0	1	Sumar
1	1	0	Restar
1	1	1	Incrementar A

Operaciones RT	PALABRA DE CONTROL (Señales de control)						
	Selección de entrada	Controles de la ALU			Escritura en acumulador R	Lectura del acumulador R	(En hexadecimal)
	S	M	S1	S0	/WR	/ReR	

NOTA: Basado en ejemplo propuesto en [GAJ97], pág. 311-313

Una posible solución al Ejercicio 4 es:



M	S_1	S_0	Operaciones de la ALU
0	0	0	Complementar A
0	0	1	Transferir A
0	1	0	NAND
0	1	1	XOR
1	0	0	Decrementar A
1	0	1	Sumar
1	1	0	Restar
1	1	1	Incrementar A

Operaciones RT	PALABRA DE CONTROL (Señales de control)						
	Selección de entrada	Controles de la ALU			Escritura en acumulador R	Lectura del acumulador R	(En hexadecimal)
	S	M	S1	S0	/WR	/ReR	
R<-- /X	0	0	0	0	0	1	01
R<-- "00..01"	1	1	1	1	0	1	3D
R<--"11..1"	1	0	0	0	0	1	21
R<--X NAND R	0	0	1	0	0	1	09
R<--X - 1	0	1	0	0	0	1	11

NOTA: Basado en ejemplo propuesto en [GAJ97], pág. 311-313

TEMA 5: Sistemas en el nivel de transferencia entre registros (RT)

5.1 Introducción y definiciones generales.

5.2 Unidad de procesamiento o camino de datos. Ejemplos de operaciones.

5.3 Unidad de Control. Ejemplos de generación de señales de control.

5.4 Ejemplo de un computador sencillo a nivel RT.

5.3 Unidad de Control. Ejemplos de generación de señales de control.

- 5.3.1 Misión de la Unidad de Control: Motivación. Ejemplo 1 de sistema RT completo.
- 5.3.2 Modelo de máquina de estado algorítmica ASM (Estructura Tipo Mealy). Cartas ASM. Técnicas asociadas.
- 5.3.3 Implementación de la unidad de control: Técnica de un biestable por estado.

5.3.1 Misión de la Unidad de Control: Motivación. Ejemplo 1 de sistema RT completo.

La unidad de control es la encargada de secuenciar microoperaciones del camino de datos de acuerdo con una especificación (algorítmica) del sistema RT a realizar. Para motivar inicialmente, cómo puede realizarse esto, proponemos el siguiente ejemplo sencillo.

5.3.1.1 Ejemplo 4 de sistema RT completo (Sumador de 8 datos)

Utilizando el "EJEMPLO 3 DE CAMINO DE DATOS SENCILLO" de la sección anterior", realizar el sistema RT completo que consiste en implementar en hardware el siguiente algoritmo para sumar 8 datos que se reciban en ciclos de reloj consecutivos a través del Bus X

$R \leftarrow 0$

bucle:

desde i = 1 **a** 8

$R \leftarrow R + X$

fin de bucle

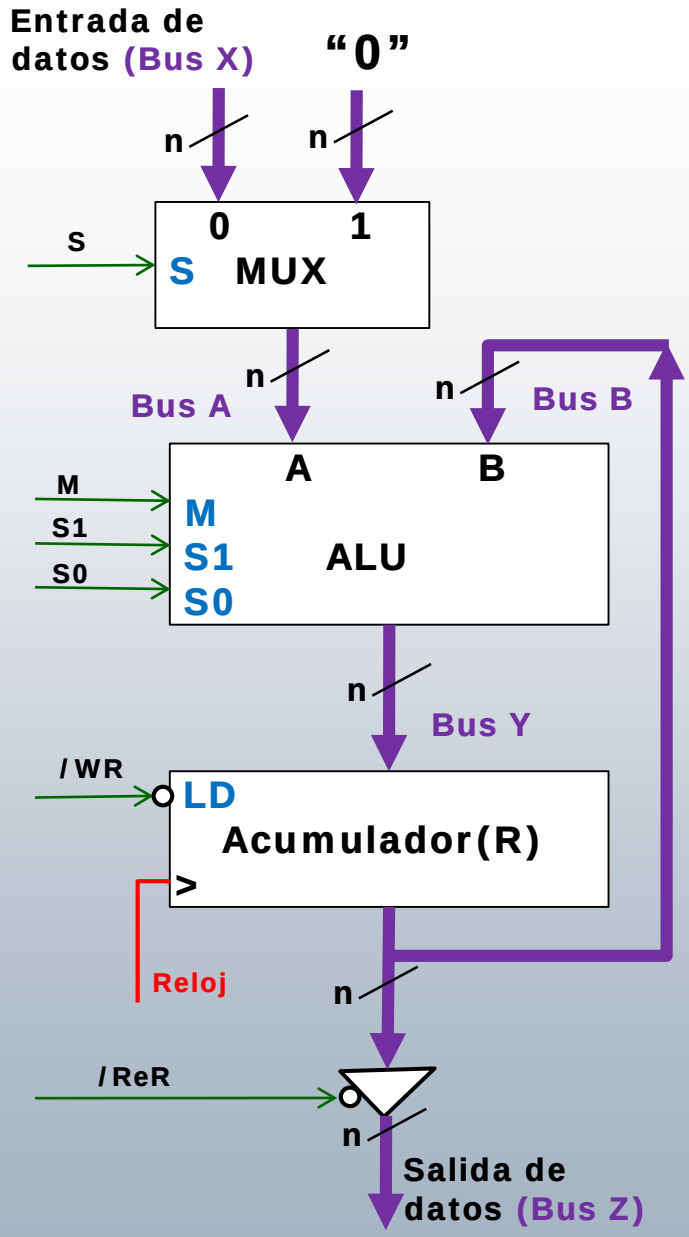
$Z = R$

NOTA: Basado en ejemplo propuesto en [GAJ97], pág. 311-313

5.3.1.1 Ejemplo 4 de sistema RT completo (Sumador de 8 datos)

El camino de datos referido, para el que se propone el ejemplo es el siguiente:

M	S_1	S_0	Operaciones de la ALU
0	0	0	Complementar A
0	0	1	Transferir A
0	1	0	NAND
0	1	1	XOR
1	0	0	Decrementar A
1	0	1	Sumar
1	1	0	Restar
1	1	1	Incrementar A



NOTA: Basado en ejemplo propuesto en [GAJ97], pág. 311-313

5.3.1.1 Ejemplo 4 de sistema RT completo (Sumador de 8 datos)

Operaciones RT	PALABRA DE CONTROL (Señales de control)						
	Selección de entrada	Controles de la ALU			Escritura en acumulador R	Lectura del acumulador R	(En hexadecimal)
	S	M	S1	S0	/WR	/ReR	
R <--' 0'	1	0	0	1	0	1	25
R <-- R+ X	0	1	0	1	0	1	15
Z = R	x	x	x	x	1	0	02

Secuencia de operaciones RT

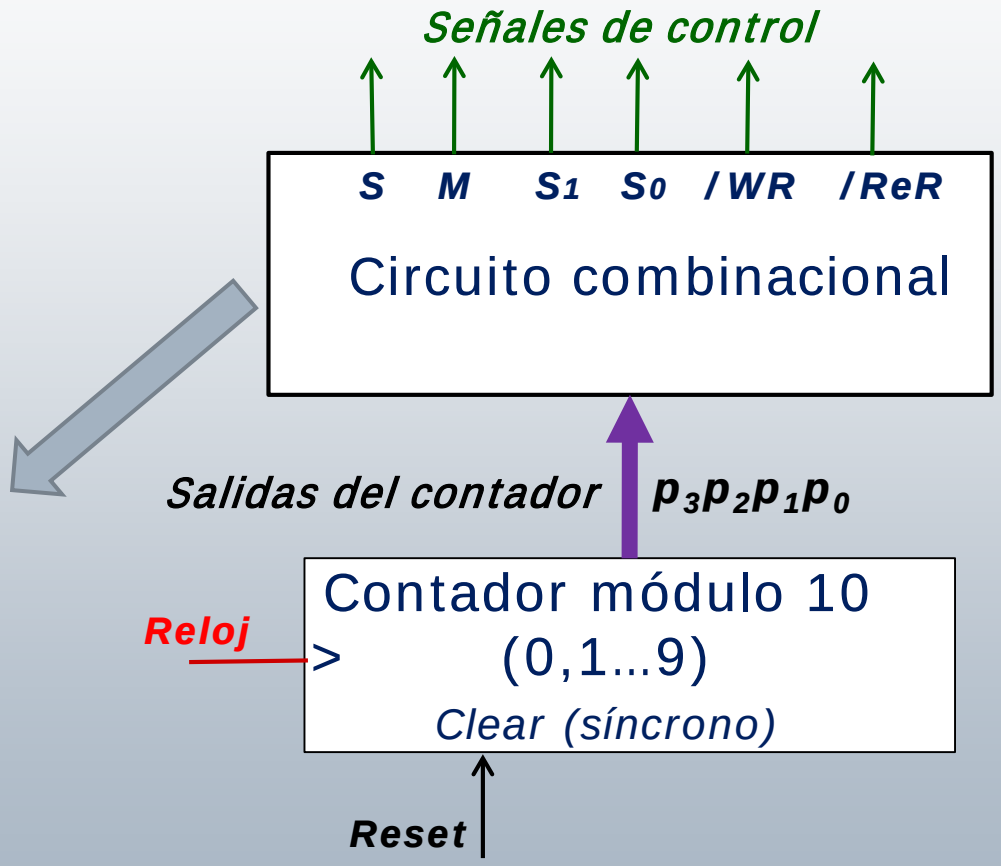
Operaciones RT	Secuencia de control
R <--' 0'	25
R <-- R+ X	15
R <-- R+ X	15
...	...
R <-- R+ X	15
Z = R	02

Repetir 8 veces

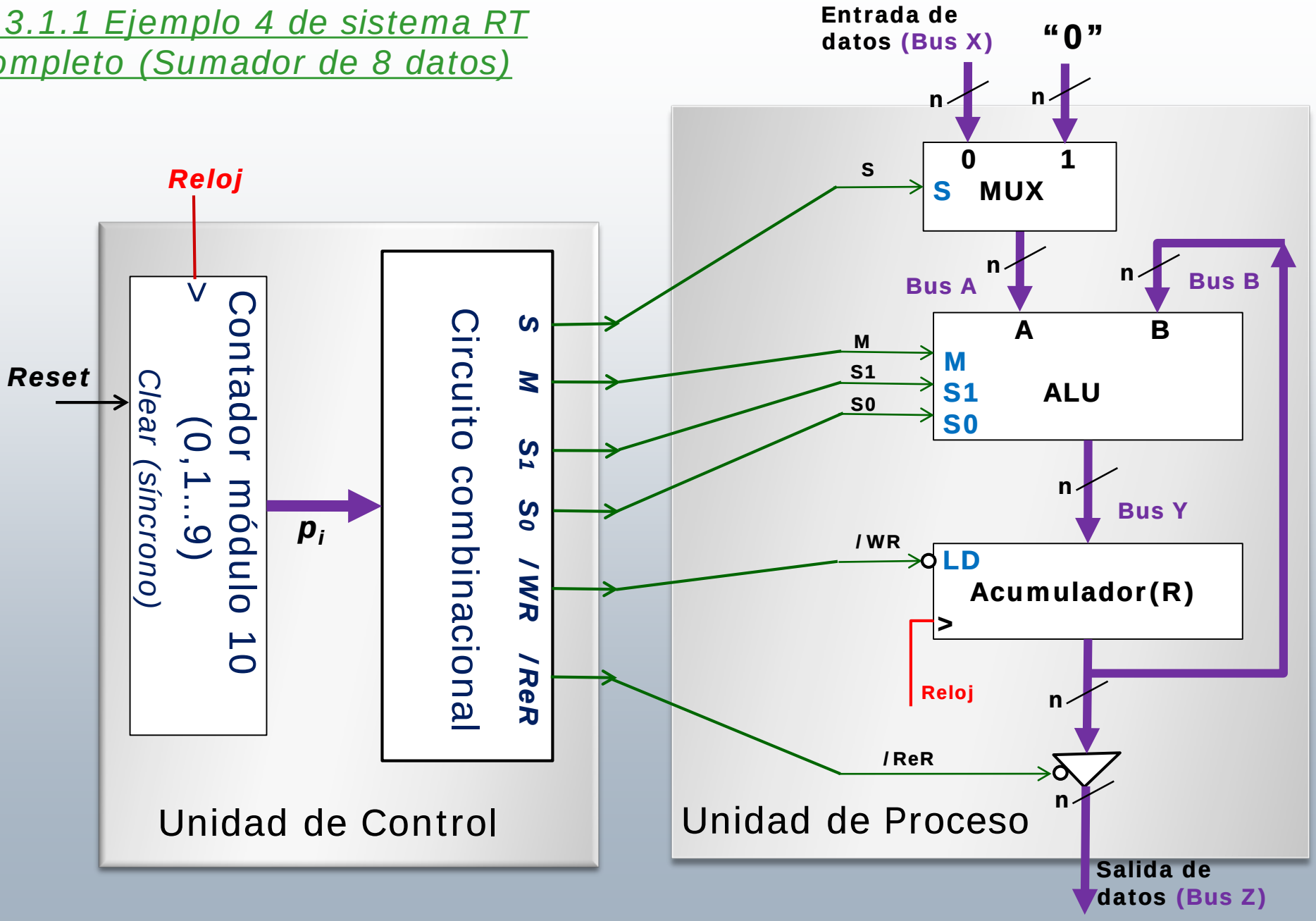
Para ejecutar el algoritmo RT en el camino de datos se precisa de un circuito secuencial que genere esta secuencia de control, denominado UNIDAD DE CONTROL

Unidad de control sencilla

Sec. de control	Salidas del contador	Salidas del convertidor
	$p_3p_2p_1p_0$	S, M, S_1, S_0, /WR, /ReR
25	0 0 0 0	1 0 0 1 0 1
15	0 0 0 1	0 1 0 1 0 1
15	0 0 1 0	0 1 0 1 0 1
...	...	...
15	1 0 0 0	0 1 0 1 0 1
02	1 0 0 1	0 0 0 0 1 0

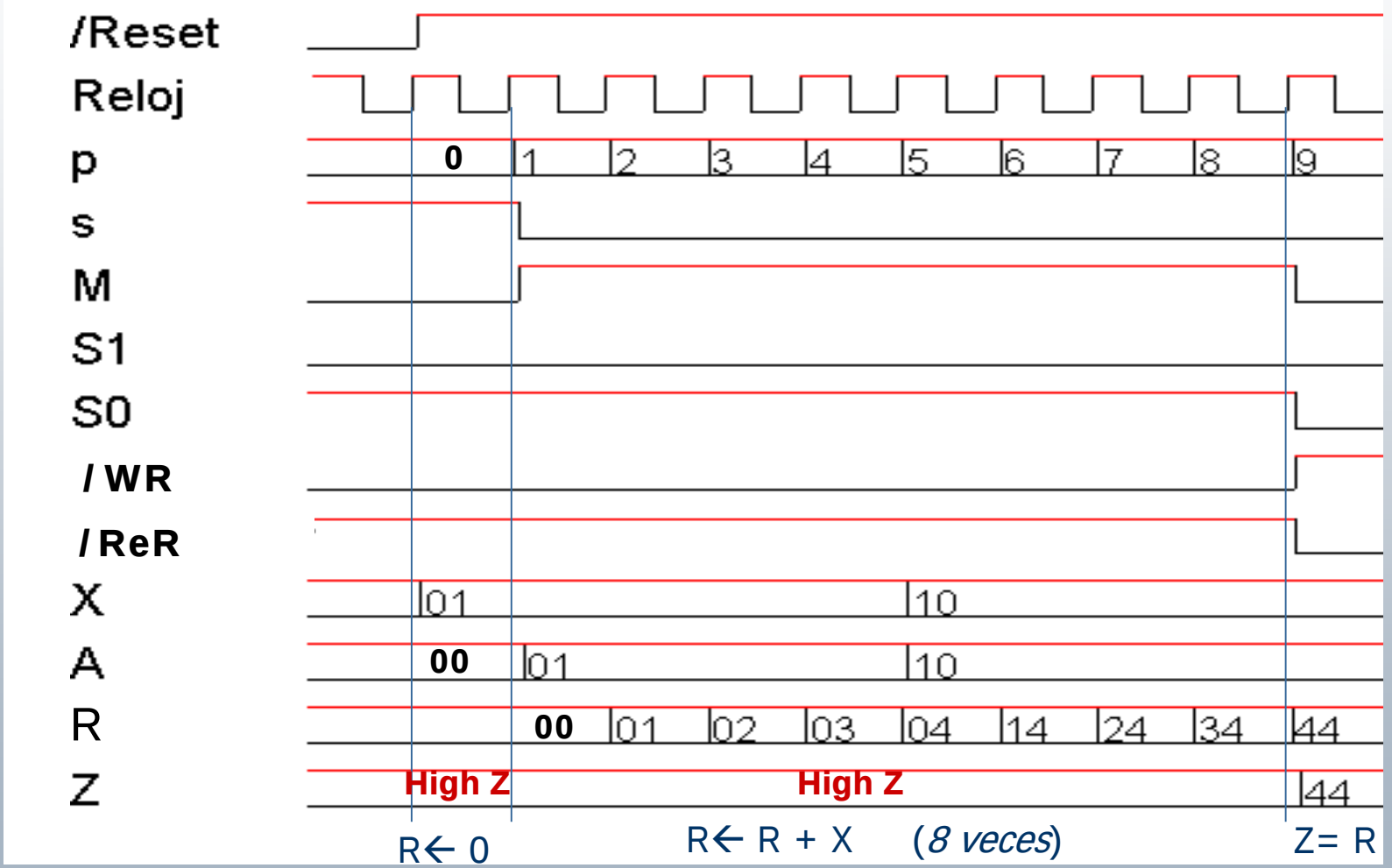


5.3.1.1 Ejemplo 4 de sistema RT completo (Sumador de 8 datos)



5.3.1.1 Ejemplo 4 de sistema RT completo (Sumador de 8 datos)

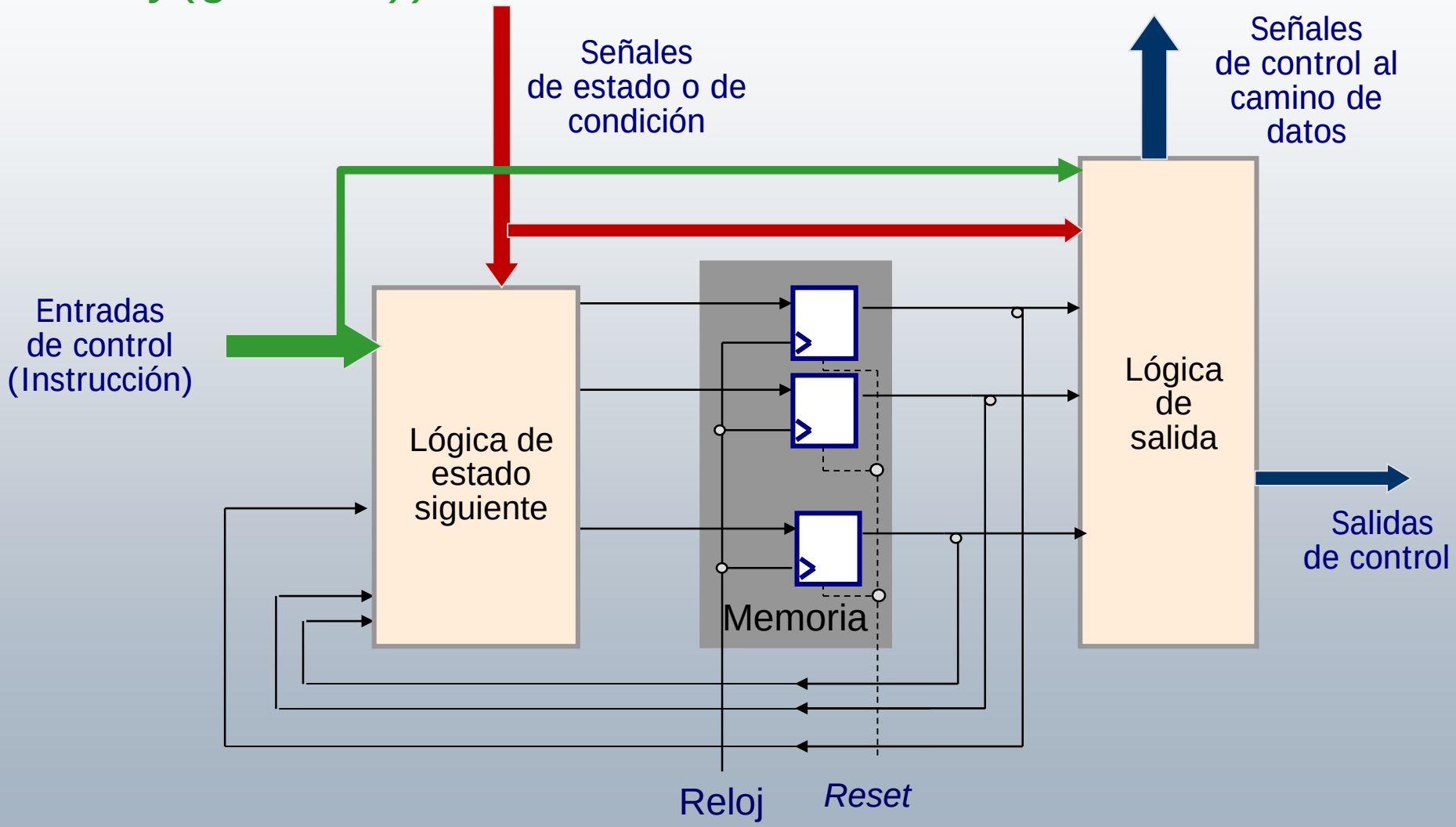
Cronograma.



5.3 Unidad de Control. Ejemplos de generación de señales de control.

- 5.3.1 Misión de la Unidad de Control: Motivación. Ejemplo 1 de sistema RT completo.
- 5.3.2 Modelo de máquina de estado algorítmica ASM (Estructura Tipo Mealy). Cartas ASM. Técnicas asociadas.
- 5.3.3 Implementación de la unidad de control: Técnica de un biestable por estado.

Modelo de máquina de estado algorítmica ASM para unidades de control cableadas (Estructura Tipo Mealy(general))



Cartas ASM. Técnicas asociadas.

- Una carta ASM es una descripción gráfica del desarrollo de instrucciones en secuencias de microoperaciones que un sistema RT debe ejecutar, de acuerdo con los requisitos y especificación del sistema.

- Se han desarrollado múltiples técnicas manuales basadas en cartas ASM para el diseño eficiente de FSMs y de sistemas RT, también denominados FSMD (*Finite State Machine with Data-path*).

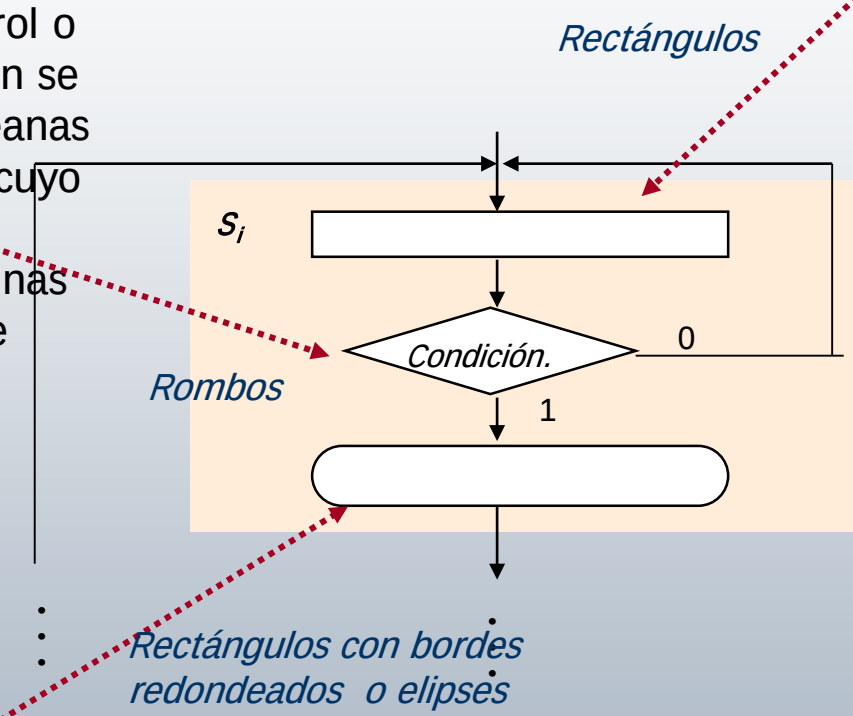
- Su origen y relevancia se remonta a la década de 1960 en los laboratorios Hewlett-Packard y descrita por primera vez por Clare en el 1973.

Cartas ASM. Técnicas asociadas. Elementos de las cartas ASM

Cajas de decisión:

Entradas externas de control o señales de estado (También se admiten expresiones booleanas entre dichas entradas) de cuyo valor puede depender

- la ejecución de algunas **acciones** en cajas de salida condicional.
- el estado siguiente (cajas de estado)



Cajas de estado:

Acciones que se ejecutan incondicionalmente cuando la máquina se encuentra en un estado (S_i)

Las **Acciones** consisten en:
1) Microoperaciones RT en la *Carta ASM de Procesado*

Eje. *A←B*

2) Activación de señales de control en la *Carta ASM de Control*

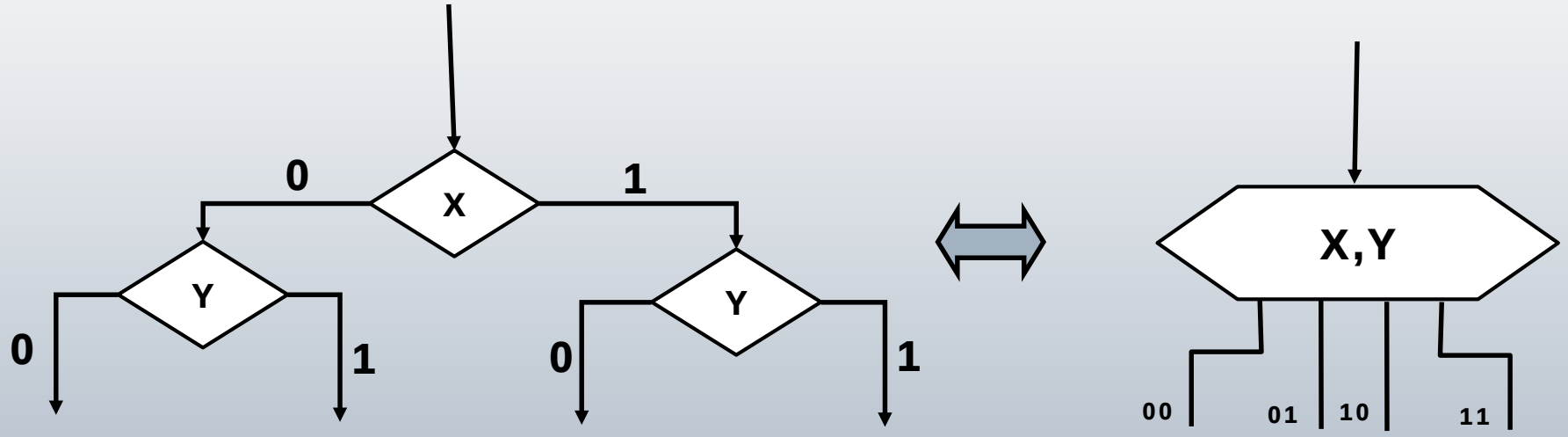
Eje. *LD_A (Escritura WA)*

Cajas de salida condicional:

Acciones que se ejecutan únicamente cuando la máquina se encuentra en el estado S_i y se verifica una condición

Cartas ASM. Técnicas asociadas.
Elementos de las cartas ASM

Cajas de decisión múltiples. Símbolo y equivalencia.



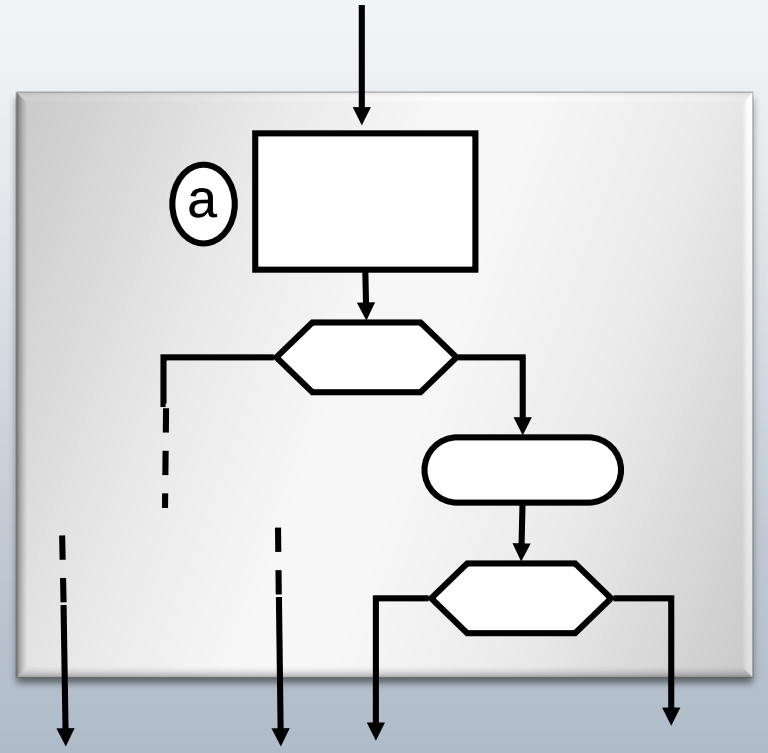
Cartas ASM. Técnicas asociadas. Elementos de las cartas ASM

Bloque ASM

• Un bloque ASM consta de una única caja de estado. La entrada al bloque ASM coincide con la entrada a la caja de estado.

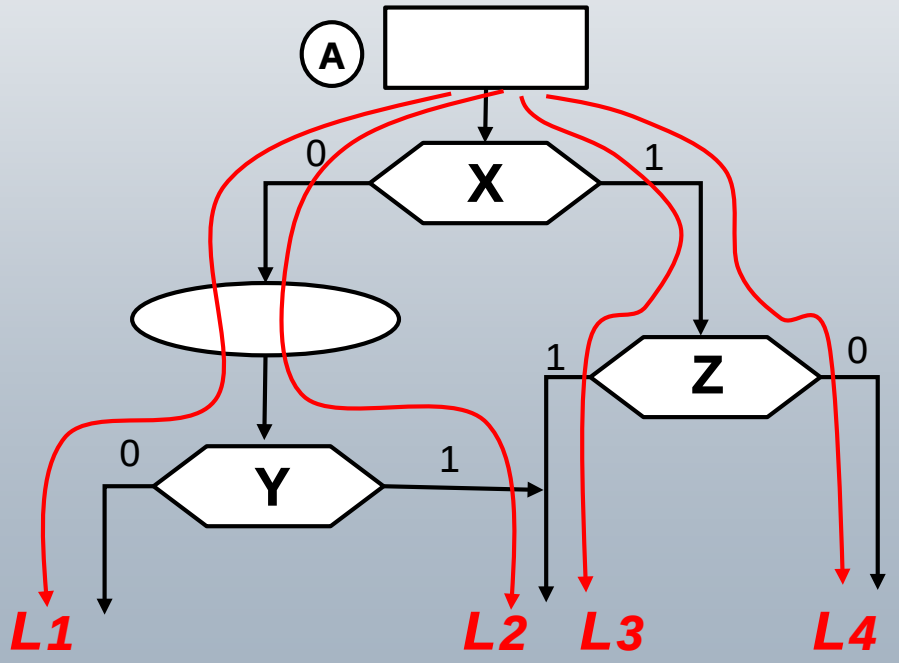
• Puede contener, opcionalmente, y en cualquier número: Cajas de decisión y Cajas de salida condicionales.

• Un bloque ASM se ejecuta en un ciclo de reloj.



Cartas ASM. Técnicas asociadas. Caminos de enlace (“link paths”)

Un "camino de enlace" (link paths) es un camino que enlaza dos cajas de estado. En cada instante la máquina de estado se encontrará en un único camino de enlace, el cual puede especificarse mediante una expresión booleana que toma el valor 1 si la máquina de estado se halla en dicho camino y el valor cero en otro caso.



$$L_1 = (Estado\,A) \cdot \overline{X} \cdot \overline{Y}$$

$$L_2 = (Estado\,A) \cdot \overline{X} \cdot Y$$

$$L_3 = (Estado\,A) \cdot X \cdot Z$$

$$L_4 = (Estado\,A) \cdot X \cdot \overline{Z}$$

Cartas ASM. Técnicas asociadas.
Reglas de construcción de Cartas ASM.

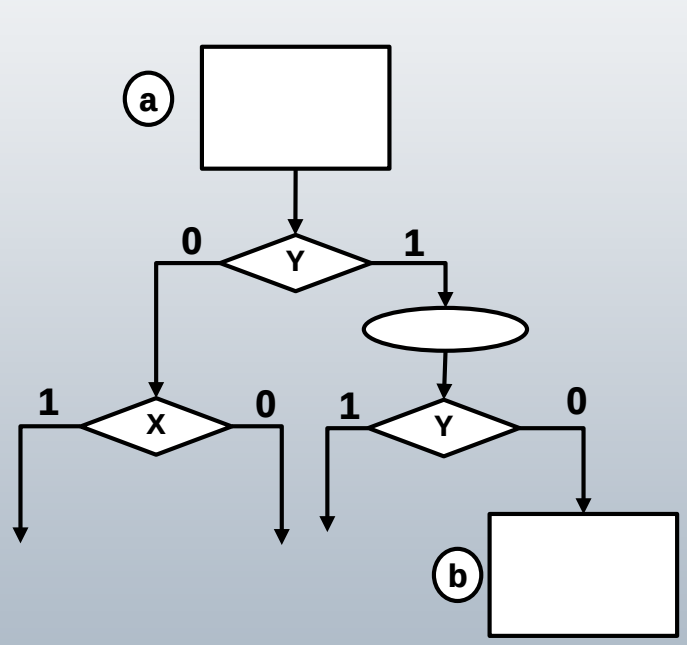
1) **Para cada estado y una condición particular de las entradas, debe haber un único camino de enlace activo.**

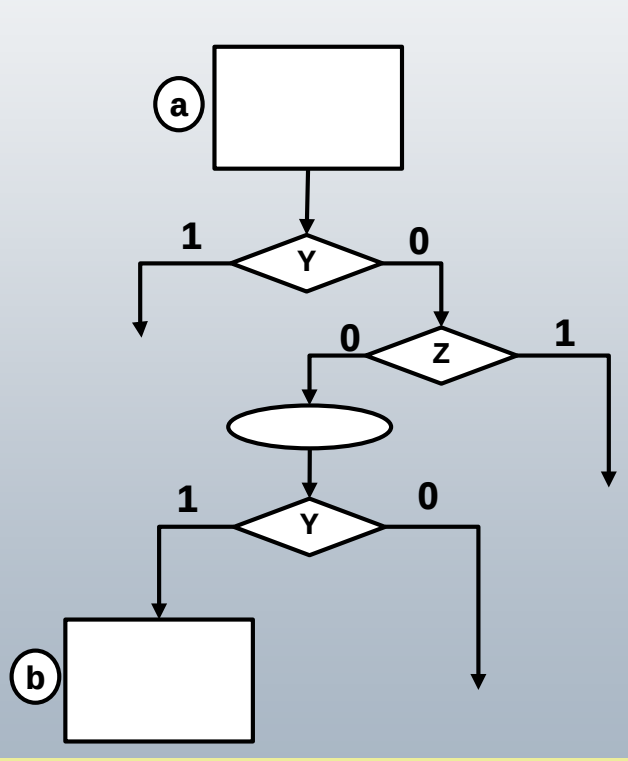
2) **Cada camino de enlace ha de terminar a la entrada de una caja de estado.**

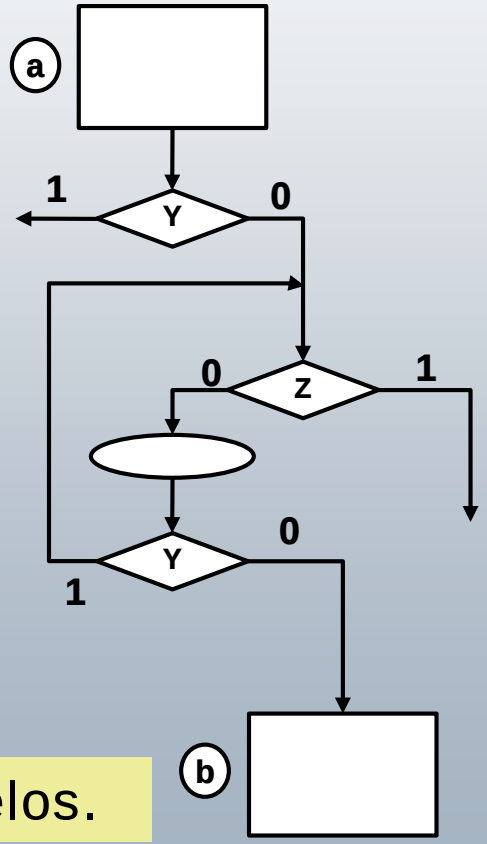
3) **Las cajas de decisión han de organizarse de modo que no den lugar a caminos de enlace "imposibles". Esto se detecta al dar "cero" la expresión booleana del camino de enlace.**

4) **La entrada a una "caja de salida condicional" solo puede hacerse a través de cajas de decisión.**

Ejemplos incorrectos de Cartas ASM







Ejercicio: Indique los errores cometidos y coméntelos.

Continuar con grupo D

Cartas ASM. Técnicas asociadas.

- A partir de las cartas ASM es posible especificar y diseñar tanto el flujo de datos como el flujo de control. Habitualmente:
  - **(1)** Se parte de un diseño de conjunto o arquitectura del sistema (Diseño preliminar), donde se identifican en bloques la unidad de procesamiento (UP) y de control (UC), pasando a especificar el comportamiento del sistema mediante una " **Carta ASM de procesado** ". Esta Carta describe la secuencia de microoperaciones RT que debe realizar el sistema en cada ciclo de reloj, compatibles con la ruta de datos que se está considerando y/o diseñando.
  - **(2)** A partir de (1) se obtiene la "**Carta ASM de control**", qué especifica las señales de control que han de activarse en cada ciclo de reloj.
  - **(3)** A partir de ésta última se puede diseñar la unidad de control de forma sistemática utilizando técnicas disponibles. Nosotros consideraremos la técnica de "un biestable por estado (uno caliente :Hot One) ".

Carta ASM de procesado. Ejemplo 5: “Cálculo del máximo”.

Ejemplo 5 : “Cálculo del máximo”:

Se presenta un ejemplo de obtención de una Carta ASM de procesado para un sistema RT que obtiene el máximo de 3 datos numéricos representados en binario sin signo.

El sistema dispone de una entrada de inicio de ejecución Xs. Antes de activar Xs, se introducen los tres datos utilizando tres teclados (un teclado para cada dato).

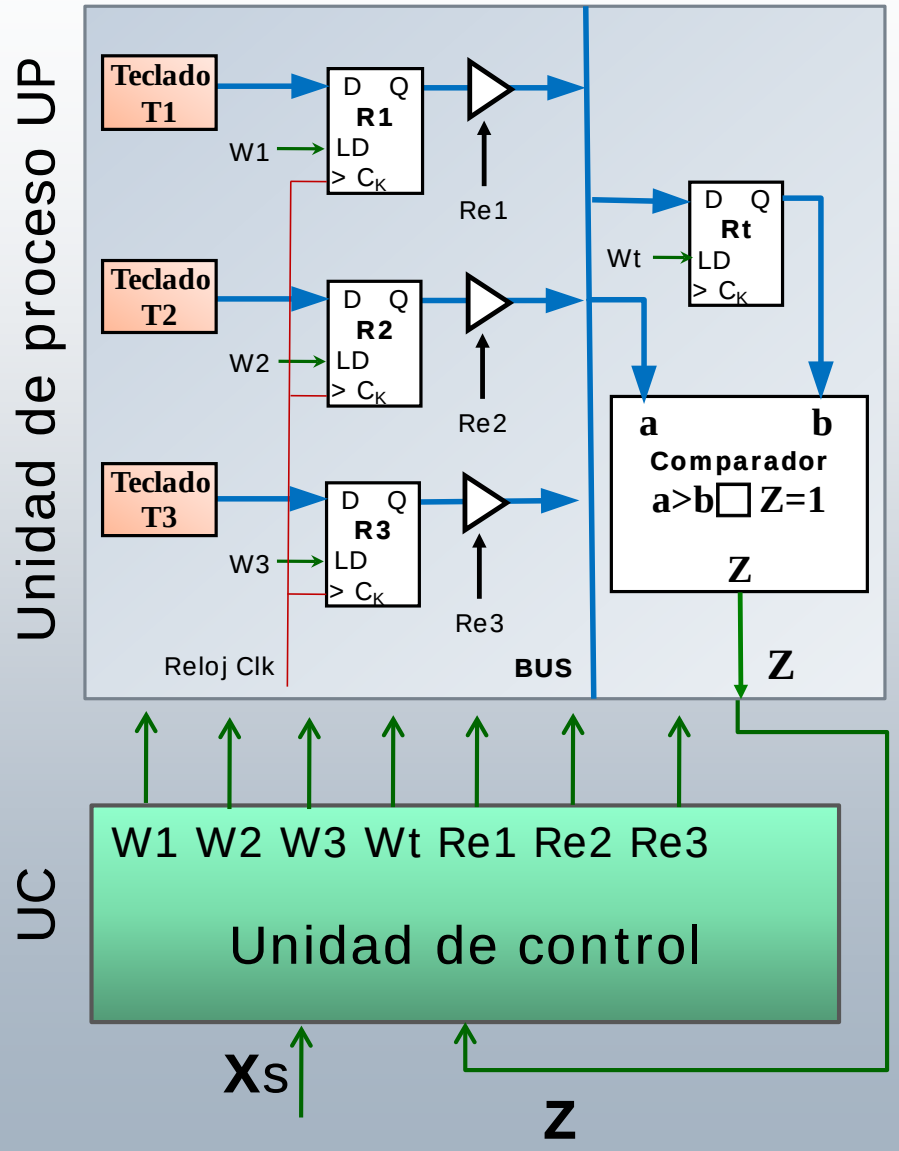
Después de activar Xs=1 (durante un ciclo de reloj) el sistema almacena los datos en tres registros: R1, R2 y R3, y pasa a obtener el valor máximo de ellos. El resultado debe almacenarse en un registro, que denominaremos Rt.

El sistema debe conservar el resultado en Rt hasta que se inicie un nuevo cálculo, actuando subre Xs.

Un posible algoritmo para el “Cálculo del máximo”:

1) Si Xs=0 entonces seguir en el paso 1)
2) Capturar datos de entrada:
   Ri← Ti (teclados con los datos)
2) Rt←R1
3) Si R2>Rt entonces:
   Rt←R2
4) Si R3>Rt entonces
   Rt←R3
5) Ir al paso 1.

Carta ASM de procesado. Ejemplo 5: “Cálculo del máximo”.



Posible organización global del sitema RT de “Cálculo del máximo”:

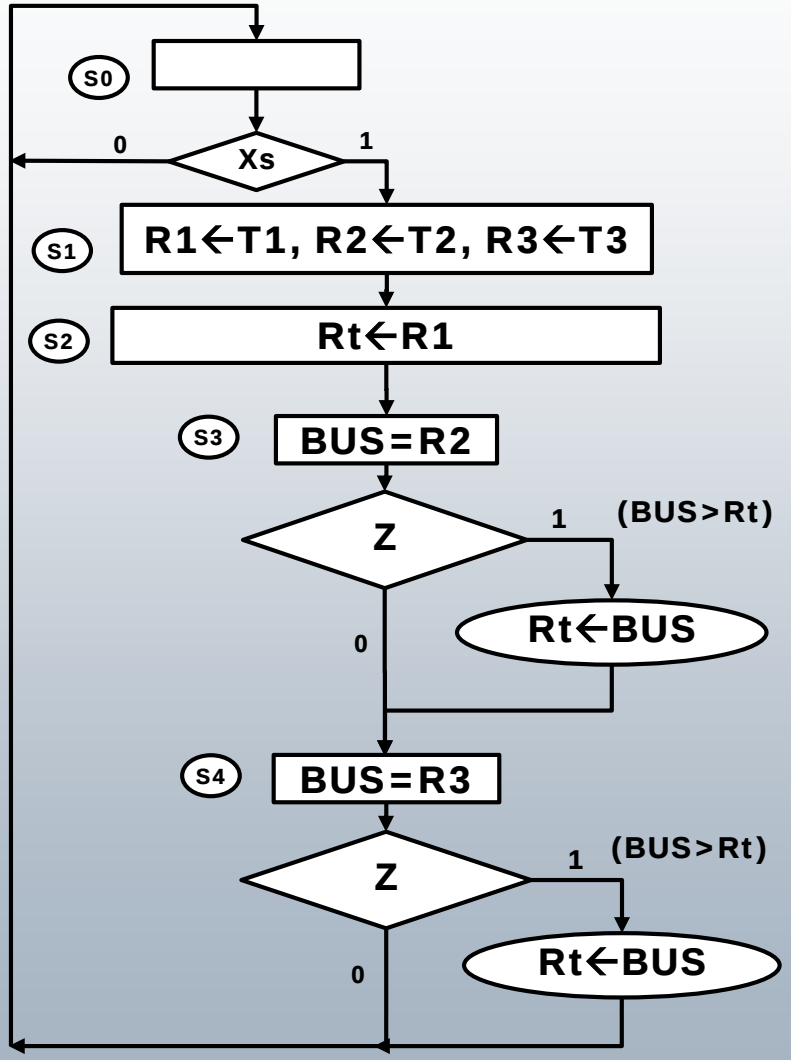
- El sistema cuenta con registros Ri y teclados Ti conectados en las entradas de esos registros, también, con el registro Rt para almacenar el resultado.

- Se dispone de un ***BUS compartido***, donde se podrán leer valores de un registro Ri dado, seleccionable.

- Las señales, como Xs y Z son señales de entrada a la unidad de control, según las cuales el sistema se bifurcará para ejecutar acciones diferentes.
  Xs : Señal externa para iniciar el algoritmo.
  Z: Señal de control, generada por un comparador de datos, disponibles en BUS y en Rt. Supondremos la señal Z definida por:

  Z=1 si BUS> Rt, Z=0 en caso contrario

Carta ASM de procesado. Ejemplo 5: “Cálculo del máximo”.

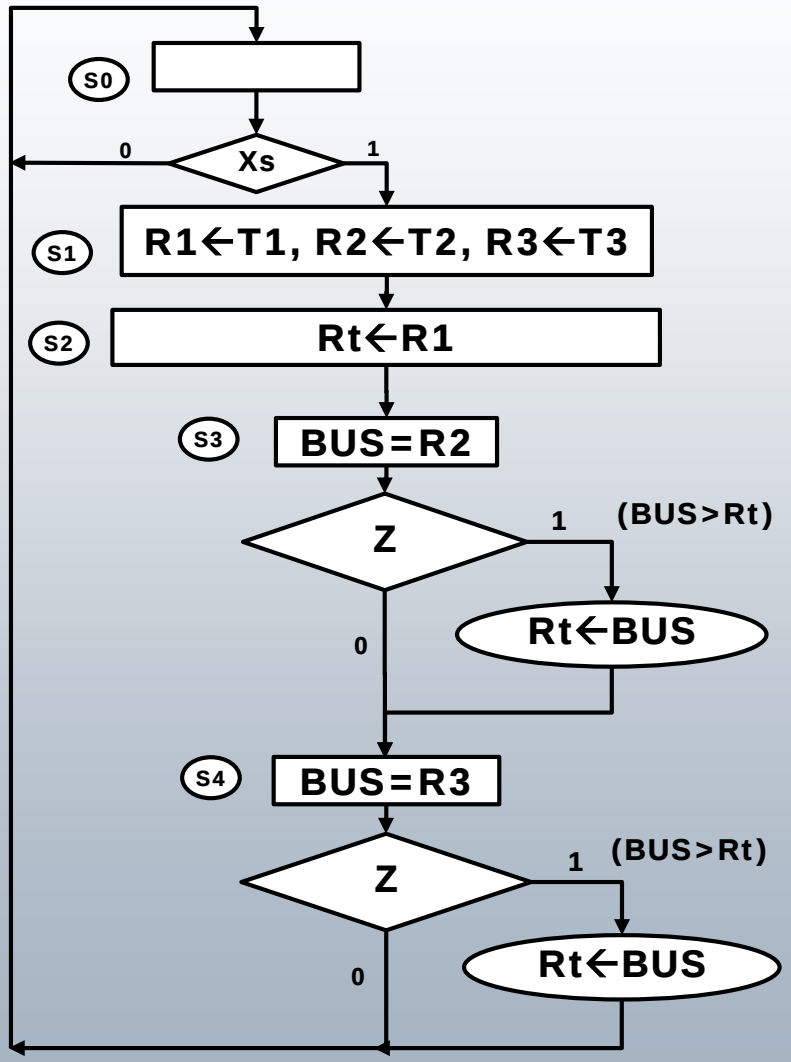


Observaciones importantes:

- Con ***BUS=Ri***, se indica una acción inmediata, independiente de la señal de reloj.

- Con ***Ri←X*** se indica una acción que se prepara en el estado especificado, pero que no se ejecuta hasta el siguiente flanco de reloj activo.

Carta ASM de procesado. Ejemplo 5: “Cálculo del máximo”.



Explicación de la Carta ASM de procesado para el algoritmo de “Cálculo del máximo”:

En estado (S0):

Si Xs=0 entonces, en el siguiente flanco activo de reloj, ir de nuevo al estado (S0), de lo contrario ir al estado (S1).

En estado (S1):

En el siguiente flanco activo de reloj, escribir datos de entrada en los registros Ri:

Ri← Ti (teclados con los datos),
e ir al estado (S2)

En estado (S2):

En el siguiente flanco activo de reloj,

Rt←R1 e ir al estado (S3).

En estado (S3):

Con BUS=R2, si Z=1 (es decir si BUS>Rt) entonces, en el siguiente flanco activo de reloj: Rt←BUS. Ir al estado (S4), se cumpla o no la condición.

En estado (S3):

Con BUS=R3, si Z=1 (es decir si BUS>Rt) entonces, en el siguiente flanco activo de reloj: Rt←BUS. Ir al estado (S4), se cumpla o no la condición.

Carta ASM de procesado. Ejemplo 5: “Cálculo del máximo”.

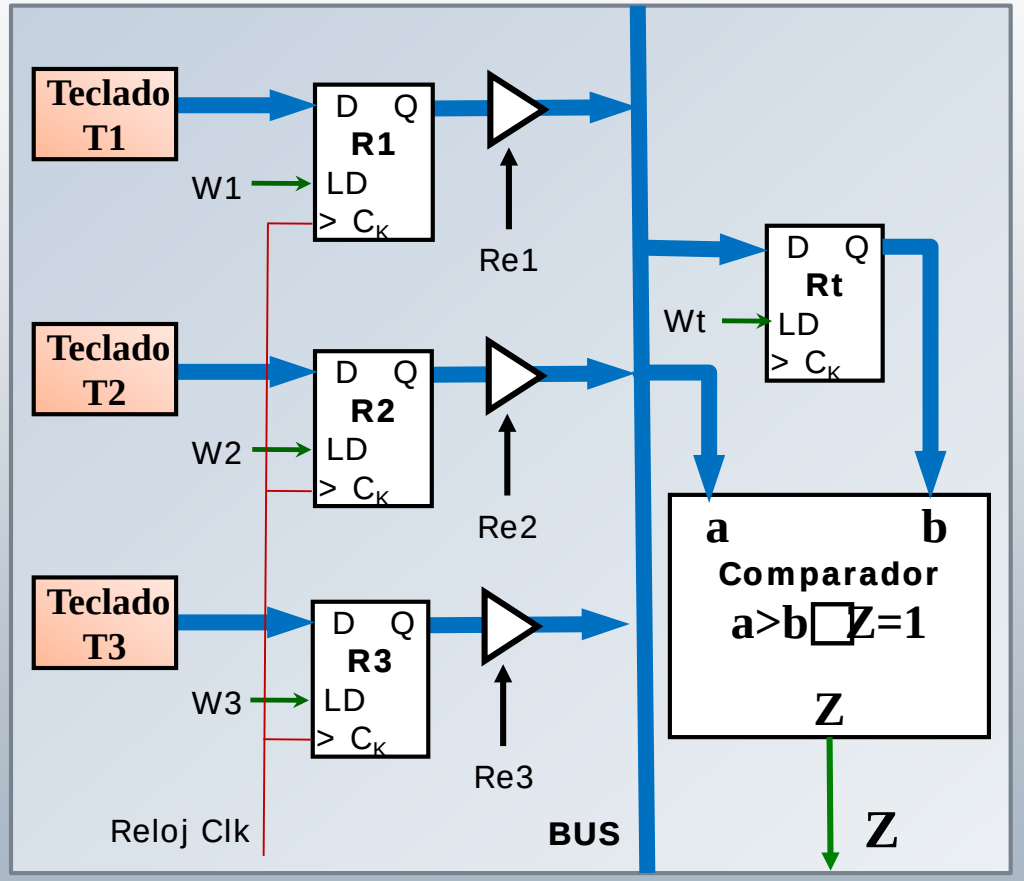
Para realizar la Carta ASM de control, tenemos que detallar las señales de control de los componentes de la Unidad de Proceso y las acciones que producen a nivel de las transferencias a registros en el sistema considerado.

En el ejemplo:

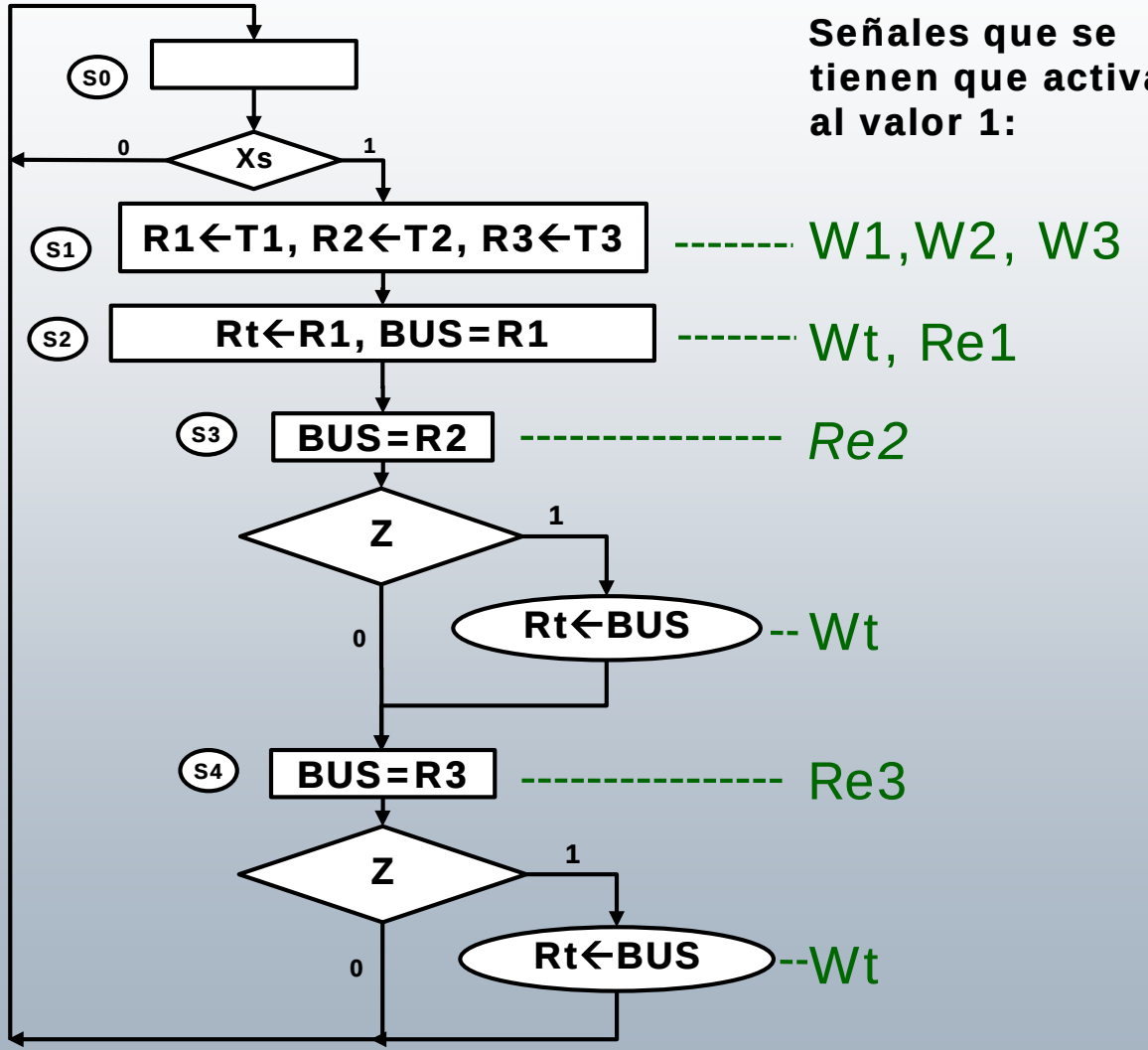
Wi	Rei	Registros Ri y elementos triestado.
0	0	Mantiene valor
0	**1**	**BUS=R**i (Acción inmediata). ***Lectura de Ri***
1	0	**Ri←Ti** (se ejecuta en el siguiente flanco activo de reloj). ***Escritura en Ri.***
1	1	No lo utilizaremos

Wt	Registro Temporal Rt
0	Mantiene valor
1	**RT ← Bus**

	Comparador
BUS>Rt	**Z=1**
BUS□Rt	**Z=0**



Carta ASM de procesado. Ejemplo 5: “Cálculo del máximo”.



Señales que se tienen que activar al valor 1:

W1,W2, W3

Wt, Re1

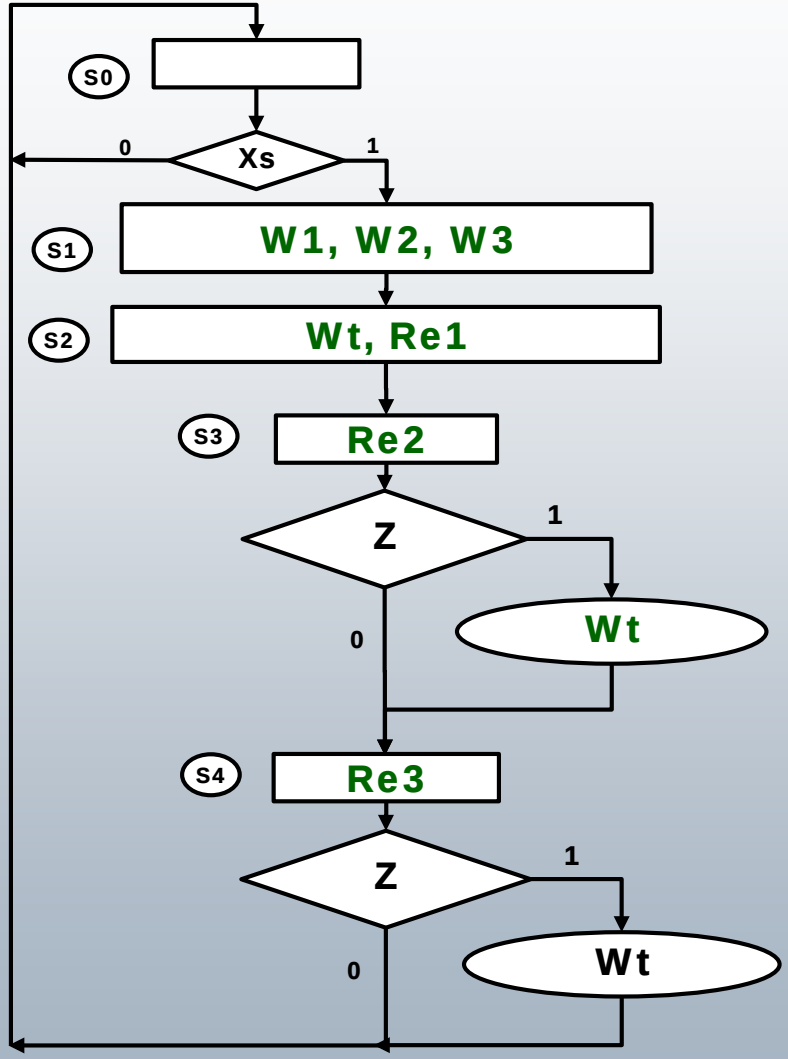
Re2

Wt

Re3

Wt

Carta ASM de procesado



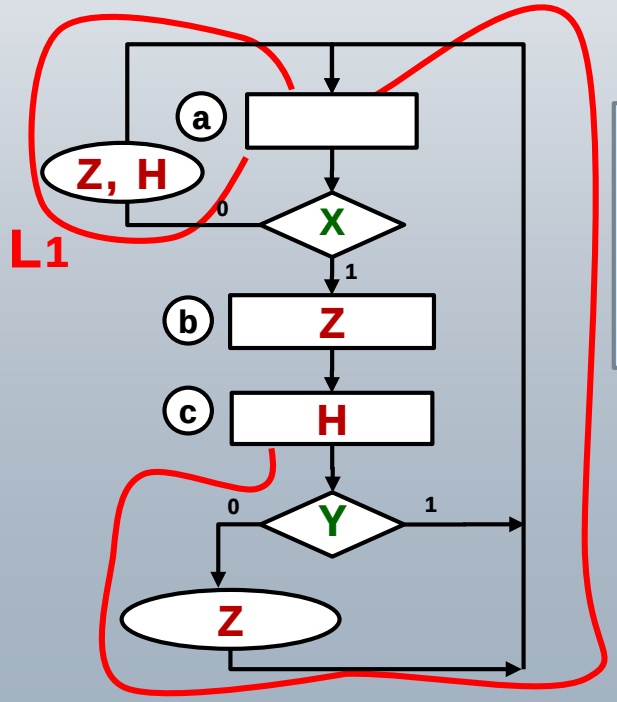
Carta ASM de control

5.3 Unidad de Control. Ejemplos de generación de señales de control.

- 5.3.1 Misión de la Unidad de Control: Motivación. Ejemplo 1 de sistema RT completo.
- 5.3.2 Modelo de máquina de estado algorítmica ASM (Estructura Tipo Mealy). Cartas ASM. Técnicas asociadas.
- 5.3.3 Implementación de la unidad de control: Técnica de un biestable por estado.

5.3.3 Implementación de la unidad de control: Técnica de “un biestable por estado”.

Las cartas ASM de control son básicamente diagramas de estado. De hecho pueden utilizarse para describir el comportamiento de los circuitos secuenciales. Las técnicas de implementación con estas cartas se basan en el concepto de “camino de enlace”, sus expresiones lógicas, y del hecho de que todas las funciones del sistema, tanto las de estado siguiente como las de salida, se pueden obtener como suma booleana de caminos de enlace donde dicha salida se especifica como activa, incluyendo los “términos de estado” correspondientes a cajas de estado donde se especifique la salida.



Ejemplo

$$Z=(b)+L_1+L_2=(b)+(a)\cdot\bar{X}+(c)\cdot\bar{Y}$$

$$H=(c)+L_1=(c)+(a)\cdot\bar{X}$$

Los términos de estado (a), (b), (c) dependen de la “asignación de estados” considerada.

5.3.3 Implementación de la unidad de control: Técnica de “un biestable por estado”.

Si los biestables son de tipo D y la asignación de estados es tal que se dedica “un único biestable activo” por cada estado, las funciones booleanas de estado siguiente se simplifican respecto a las variables de estado, hasta el punto de que los términos producto de estado se reducen a una sola variable de estado. Esto permite interpretar directamente los símbolos y señales de la carta ASM hacia la estructura de circuito que la implementa.

Asignación de “un biestable por estado”

	ABC
a	100
b	010
c	001

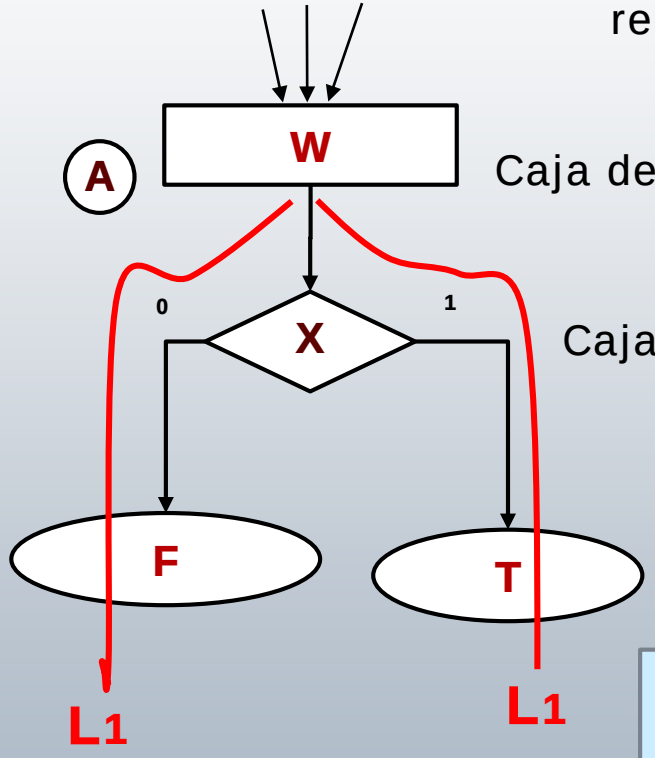
Ejemplo

$$Z=(b)+(a)\cdot\overline{X}+(c)\cdot\overline{Y}=B+A\cdot\overline{X}+C\cdot\overline{Y}$$

$$H=(c)+(a)\cdot\overline{X}=C+A\cdot\overline{X}$$

5.3.3 Implementación de la unidad de control: Técnica de “un biestable por estado”. ***Transformaciones.***

Múltiples entradas a una Caja de estado requieren una puerta OR

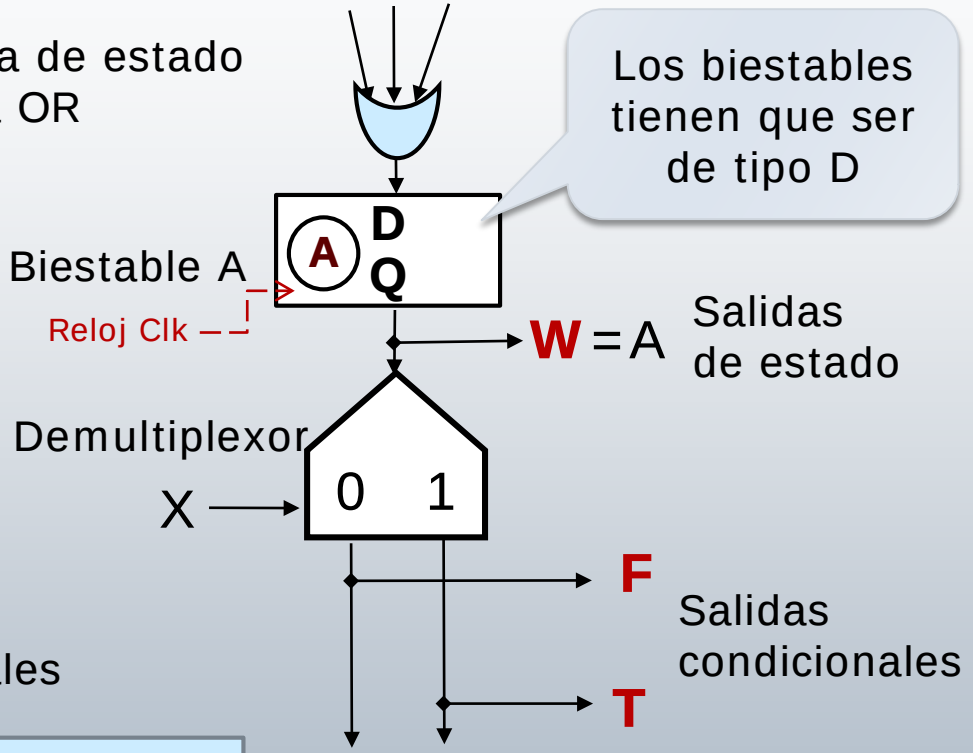


Cajas condicionales

$$W = (A)$$

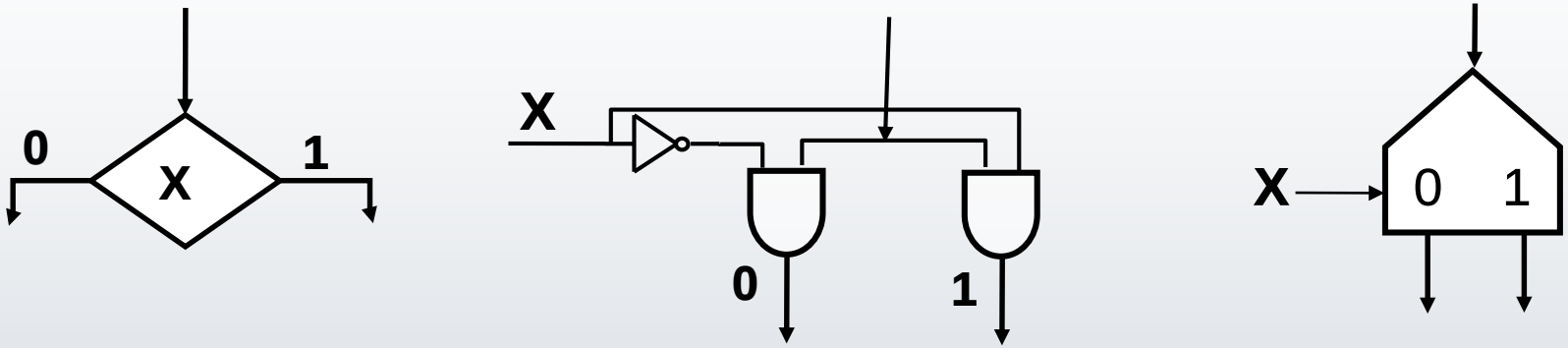
$$F = L_1 = (A) \cdot \overline{X}$$

$$T = L_2 = (A) \cdot X$$

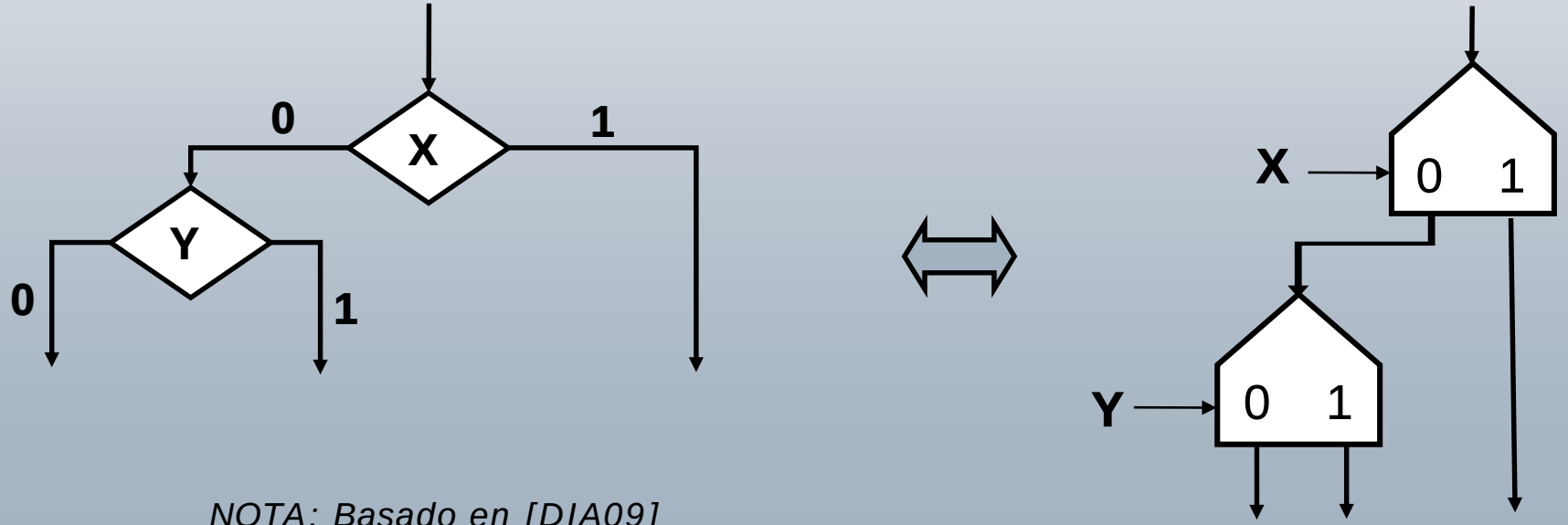


NOTA: Basado en [DIA09]

Caja de decisión simple.

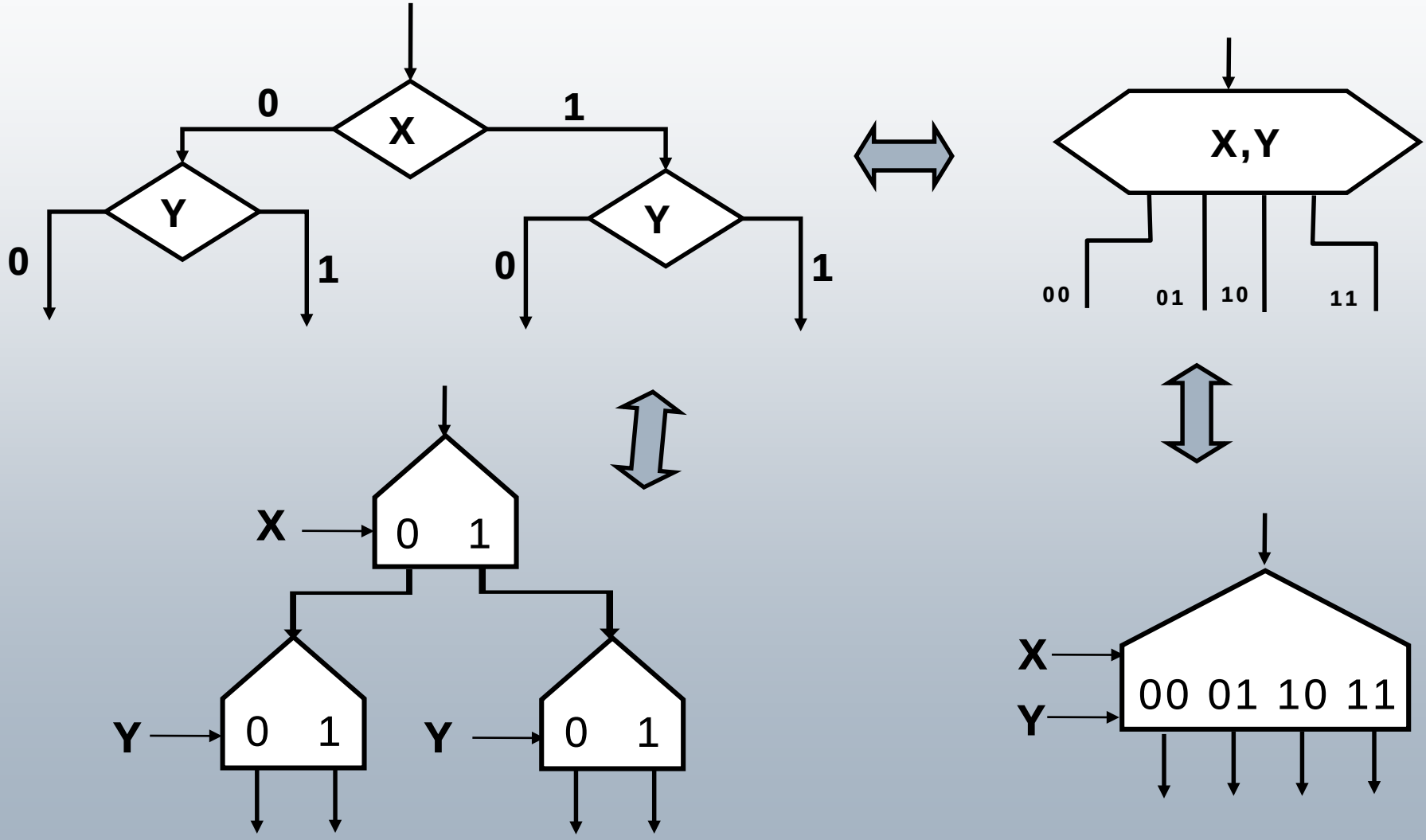


Caja de decisión múltiple.



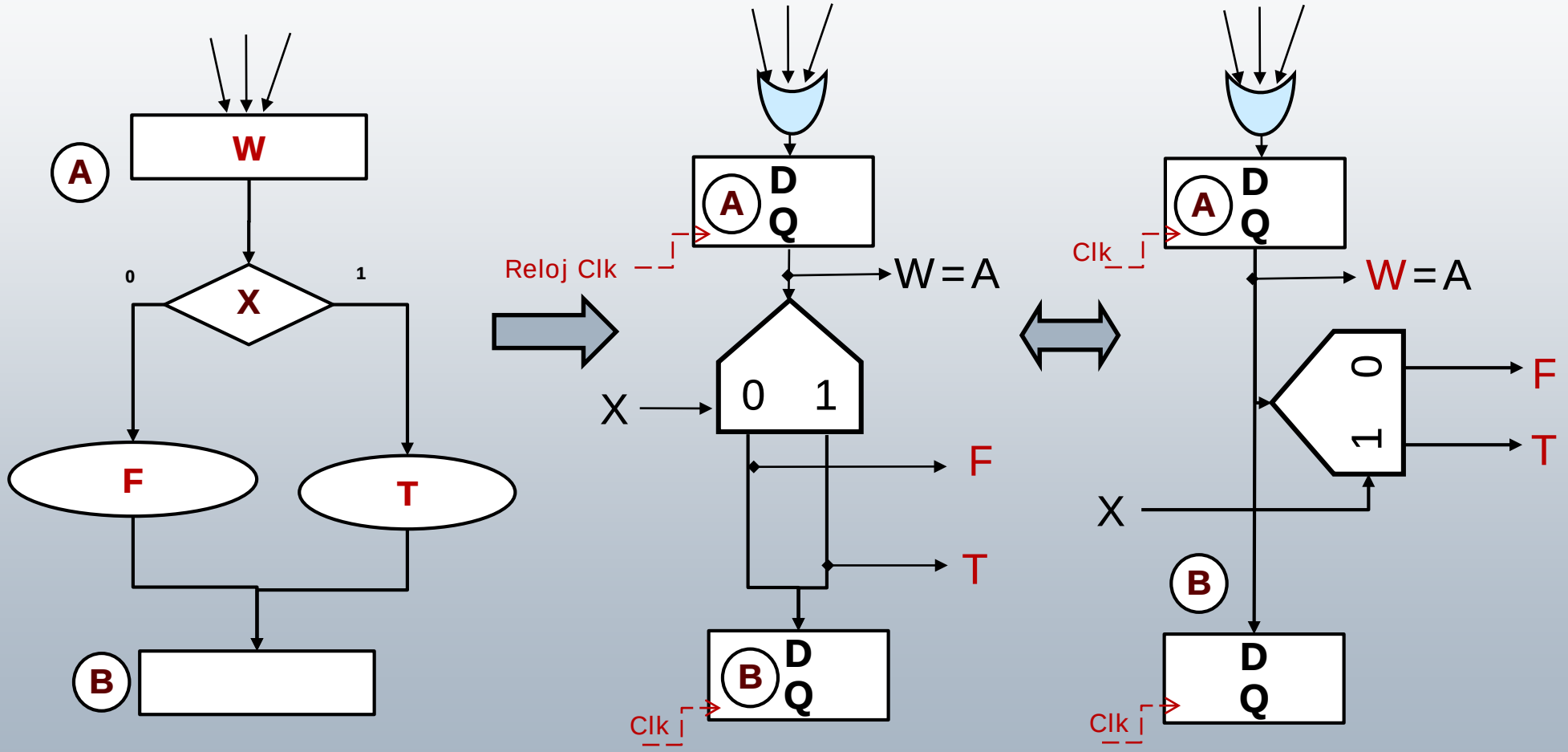
NOTA: Basado en [DIA09]

Cajas de decisión múltiples equivalentes.



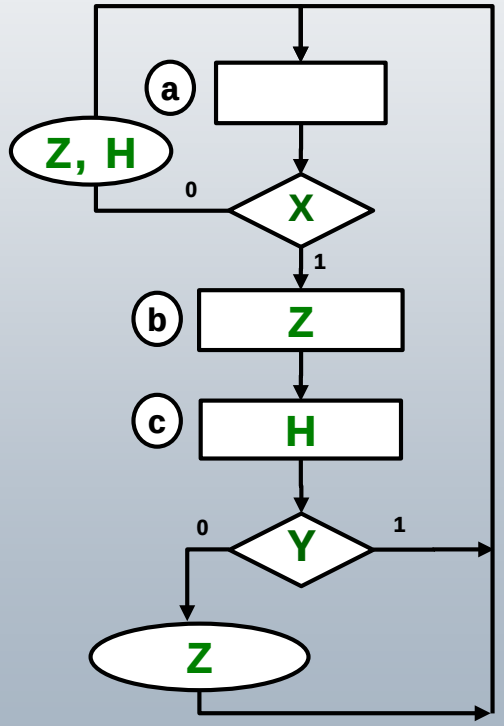
NOTA: Basado en [DIA09]

5.3.3 Implementación de la unidad de control: Técnica de "un biestable por estado". ***Transformaciones equivalentes.***



NOTA: Basado en [DIA09]

5.3.3 Implementación de la unidad de control: Técnica de "un biestable por estado". ***Ejemplo de síntesis.***

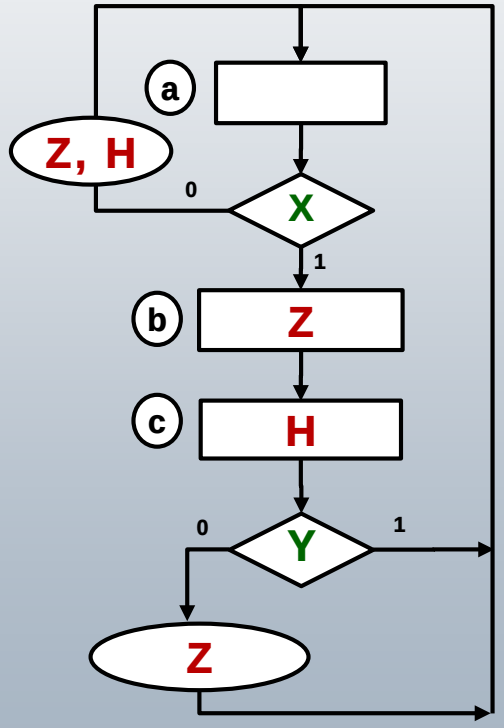


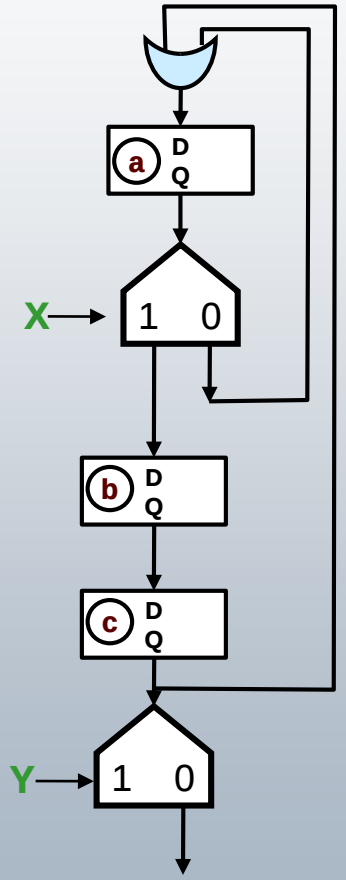
Ejemplo:

Obtener mediante la técnica de "un biestable por estado" la unidad de control (circuito secuencial) cuyo comportamiento viene especificado mediante la Carta ASM de control de la figura de la izquierda.

5.3.3 Implementación de la unidad de control: Técnica de "un biestable por estado". ***Ejemplo de síntesis.***

Ejemplo

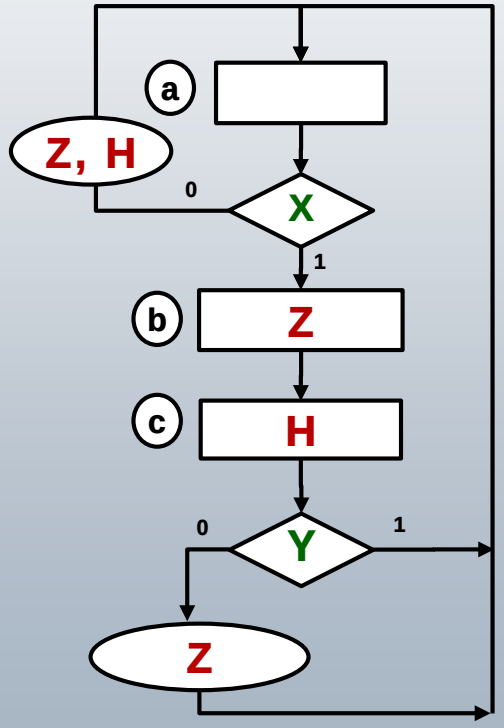


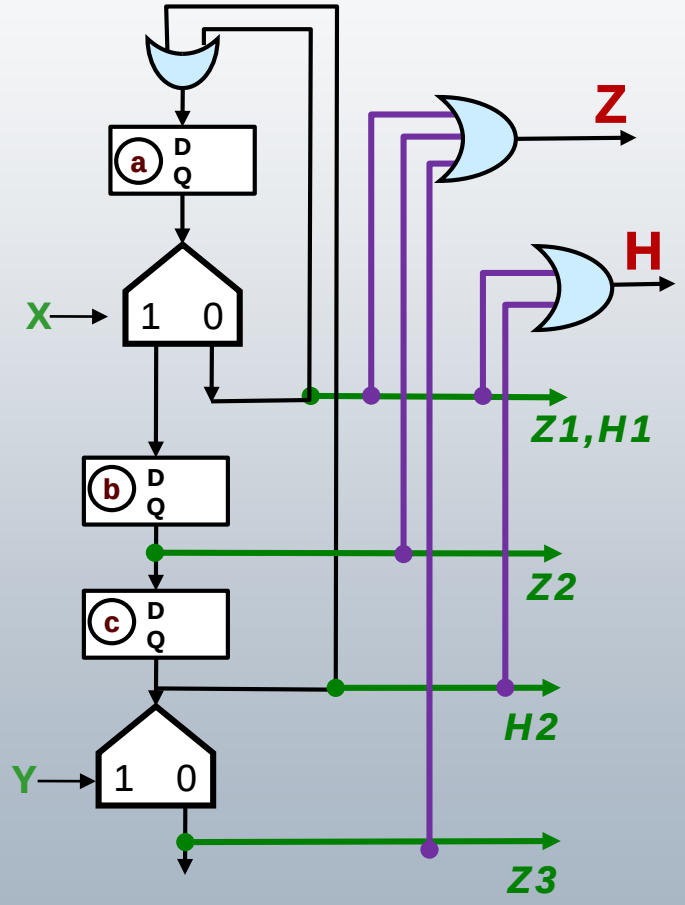


1) En este primer paso, realizamos las conexiones entre componentes de circuitos, asociados a los símbolos de la Carta ASM. Cada caja de estado se corresponde con un biestable. Cada caja condicional con un DEMUX, y cada camino de enlace de la carta ASM con una conexión entre componentes. Si varias flechas de la Carta ASM se dirigen hacia la entrada de una misma caja de estado, hay que poner una puerta OR en la entrada del biestable correspondiente a dicha caja de estado. Además de esto, aplicaríamos las transformaciones equivalentes que considerásemos más apropiadas en el diseño.

5.3.3 Implementación de la unidad de control: Técnica de "un biestable por estado". ***Las salidas.***

Ejemplo





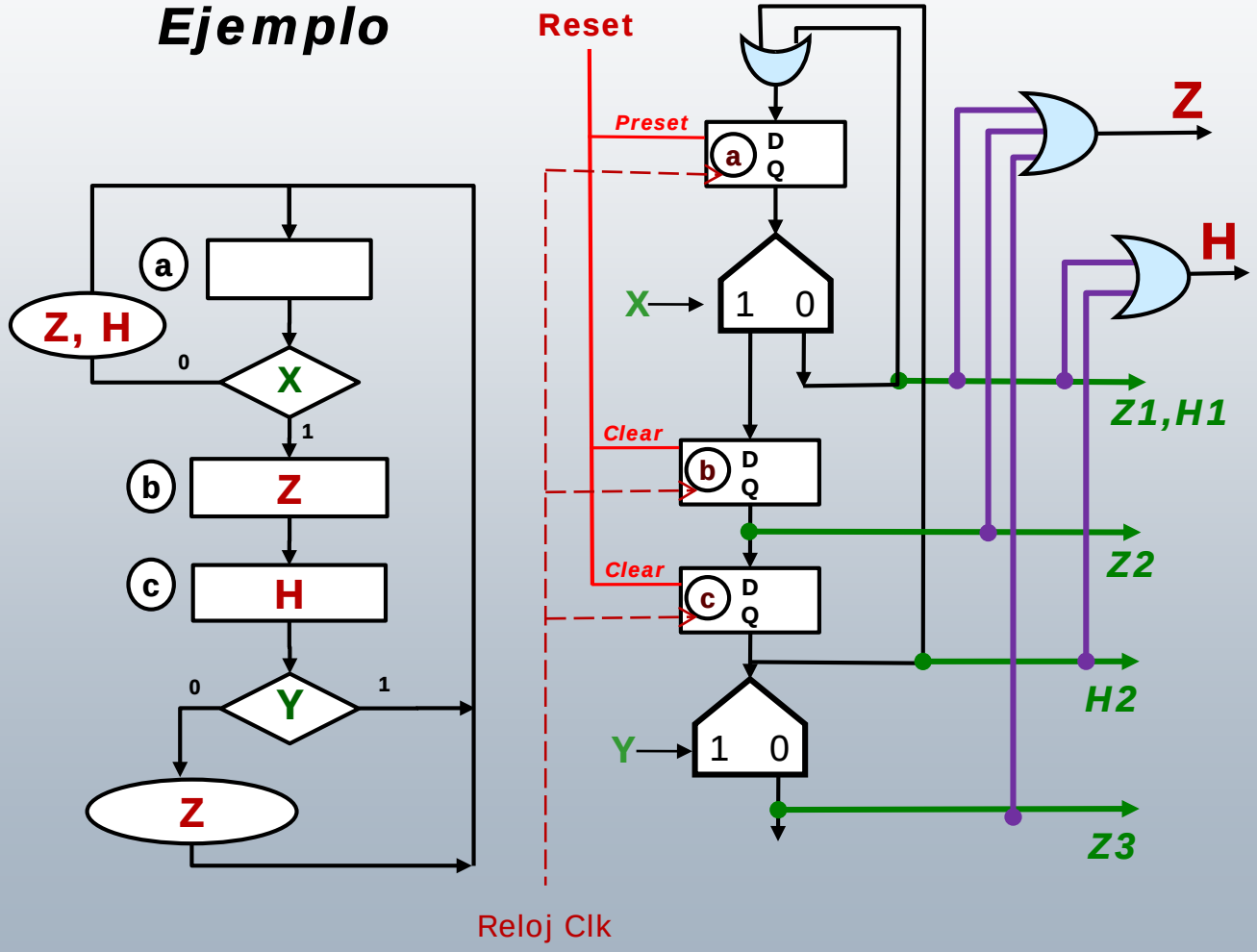
Si llamamos ***Zi***, a las salidas que activan a **Z** y ***Hi*** las que activan a **H**, entonces la salida **Z** es la OR de las salidas ***Zi***. Igualmente, la salida **H** se obtiene como la OR de las ***Hi***:

$$Z = Z1 + Z2 + Z3$$
$$H = H1 + H2$$

5.3.3 Implementación de la unidad de control: Técnica de "un biestable por estado". ***Inicialización y señal de reloj.***

CONTINUAR Grupo A

Ejemplo



La señal ***Reset*** se utilizará para inicializar el sistema de modo asíncrono al estado inicial. En el ejemplo este estado sería: "**abc=100**", es decir se inicializa el biestable "**a**" al valor "**1**" (activando su "***preset***") y el resto de los biestables al valor "**0**" (activando sus "***clear***")

TEMA 5. Sistemas en el nivel de transferencia entre registros (RT)

5.1 Introducción y definiciones generales.

5.2 Unidad de procesamiento o camino de datos. Ejemplos de operaciones.

5.3 Unidad de Control. Ejemplos de generación de señales de control.

5.4 Ejemplo de un computador sencillo a nivel RT.

TEMA 5.
5.4 Ejemplo de un computador sencillo a nivel RT.

5.4.1 Introducción.

5.4.2 Fases de Diseño RT.

5.4.3 Planificación de CS1.

5.4.4 Diseño Preliminar de CS1.

5.4.5 Diseño detallado de CS1.

a) Diseño detallado de la UP.

b) Diseño detallado de la UC.

NOTA: Basado en [DIA09]

5.4 Ejemplo de un computador sencillo

5.4.1 Introducción.

- En este apartado se aplican los conocimientos adquiridos en las secciones anteriores del Tema 5 para diseñar un computador sencillo (CS1), desde el punto de vista de un sistema digital a nivel de transferencia entre registros (RT).
- El computador sencillo CS1 ha sido ideado y descrito en el libro:

  [DIA09]: Estructura y Tecnología de Computadores. Teoría y problemas. **Autor/es:** Sergio Díaz Ruiz, María del Carmen Romero Ternero, Alberto J. Molina Cantero. **Más info:** *McGraw-Hill, 2009.*
- En las transparencias se han adaptado la descripción y diseño de CS1 a la asignatura de Tecnología y Organización de Computadores. Recomendamos al estudiante utilizar como uno de los libros básicos de texto para este Tema 5 el libro **[DIA09]**, sobre diseño RT y computadores sencillos.

5.4 Ejemplo de un computador sencillo

5.4.1 Introducción.

- En los ejemplos vistos anteriormente en el Tema 5, como el sumador de 8 datos y la calculadora, corresponden a sistemas digitales de *uso específico*, en donde el/los algoritmos que se ejecutan están implementados en el propio hardware del sistema.

- En este apartado ilustramos el diseño RT de un sistema digital de *uso general*, como es un computador. Esto es, *un sistema digital que implementa y secuencia microoperaciones de transferencia para capturar, interpretar y ejecutar secuencias de instrucciones almacenadas en memoria RAM (programa)*. Ahora, el programa a ejecutar puede cambiarse, simplemente cargando otra secuencia distinta de instrucciones en la RAM, sin necesidad de modificaciones del Hardware.

5.4 Ejemplo de un computador sencillo

5.4.2 Fases de Diseño RT.

- **Planificación**: Especificar el sistema a realizar, características que debe cumplir, y decidir el tipo de diseño (“descendente” o “ascendente”). También, determinar la tecnología de los componentes a utilizar.

- **Diseño Preliminar**: Especificar el diseño de conjunto o arquitectura del sistema, es decir, identificar las unidades de procesamiento (UP) y de control (UC). En el caso de utilizar técnicas ASM, especificar el comportamiento del sistema mediante una Carta ASM de procesado, esto es, la secuencia de microoperaciones RT que debe realizar dicho sistema.

- **Diseño detallado** (Diseño lógico): Realizar la UP utilizando los tipos de componentes apropiados. Definir y/o identificar las señales de control que requiere el sistema y realizar la unidad de control.

- **Diseño físico**: Construir, depurar y documentar una versión prototipo del sistema.

5.4 Ejemplo de un computador sencillo

5.4.3 Planificación de CS1.

Queremos realizar un diseño de tipo “descendente (de arriba hacia abajo), de un computador sencillo CS1, cuyas características son:

Especificaciones del CS1 :

Se trata de un computador sencillo CS1 con una arquitectura Von Neumman que tiene una Memoria RAM capaz de almacenar $2^6=64$ datos/instrucciones de 8 bits (bus de direcciones de 6 bits y bus de datos de 8 bits), una ALU que permita sumar o restar y cinco registros:

RT registro temporal de 8 bits

AC registro acumulador, de 8 bits

MAR registro de direcciones (6 bits) hacia la memoria principal.

PC Contador de programa que almacena la dirección (6 bits) de la siguiente instrucción a ejecutar.

IR Registro de instrucciones de 8 bits.

5.4 Ejemplo de un computador sencillo

5.4.3 Planificación de CS1.

Se desea que el registro IR del computador esté formado por un campo de Código de Operación (CO) de 2 bits y un campo para indicar la dirección de memoria principal (6 bits) donde se encuentra el operando, de acuerdo con el tipo de instrucción a ejecutar.

En resumen, el computador debe diseñarse para implementar el siguiente conjunto de instrucciones:

Registro IR		Nemónico	Descripción RT
CO	Dirección del operando		
00	X X X X X X	STOP	Fin ejecución
01	$A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	ADD A_{5-0}	AC ← AC + M(A_{5-0})
10	$A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	SUB A_{5-0}	AC ← AC - M(A_{5-0})
11	$A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	STA A_{5-0}	M(A_{5-0}) ← AC

NOTA: Basado en [DIA09]

5.4 Ejemplo de un computador sencillo

5.4.4 Diseño Preliminar de CS1.

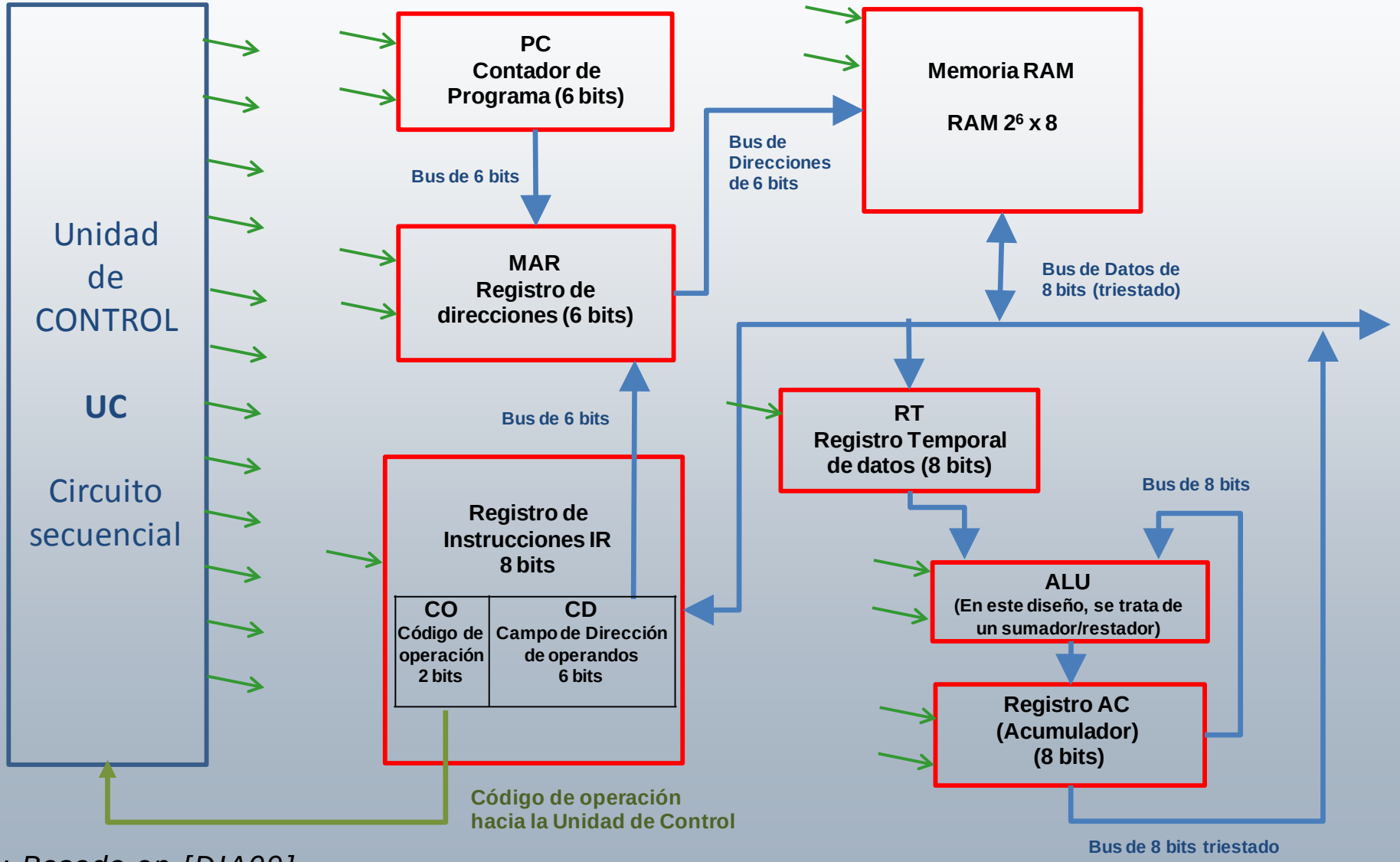
Especificar el diseño de conjunto del sistema, identificando las unidades de procesamiento (UP) y de control (UC) y sus arquitecturas. En el caso de utilizar Cartas ASM, obtener la especificación del comportamiento del sistema mediante una Carta ASM de procesado, esto es, la secuencia de microoperaciones RT que debe realizar dicho sistema.

- En este paso, hay que tener en cuenta que la división entre UC y UP del sistema es conceptual y no física, siendo el diseñador el que decide la división más conveniente.

- Para facilitar el diseño del computador sencillo, consideraremos que la UP estará formada por los componentes que: almacenan, transforman y conducen información. Respecto a la UC, ésta consistirá en un circuito secuencial tipo Mealy.

5.4.4 Diseño Preliminar de CS1.

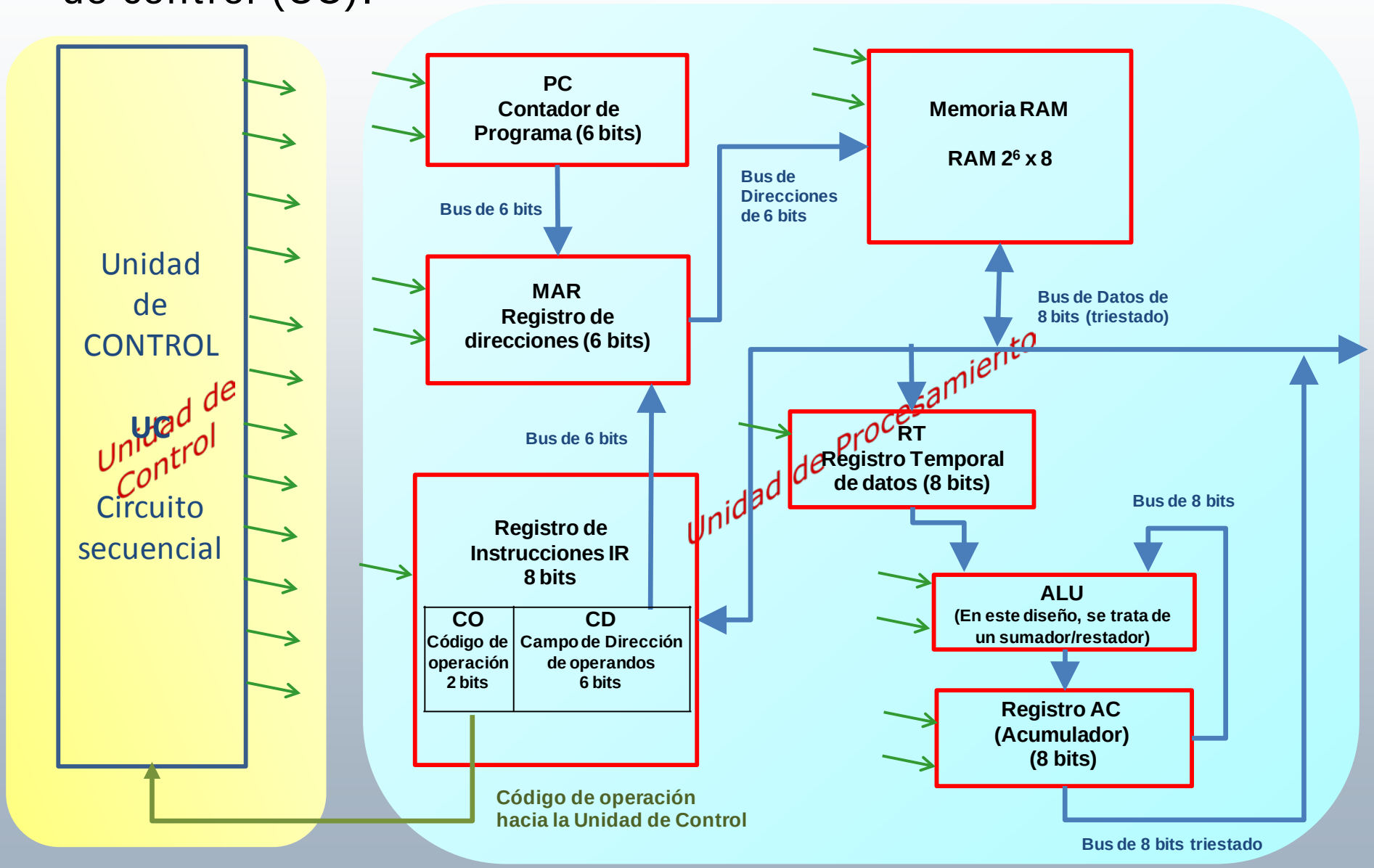
Arquitectura de conjunto: unidad de procesamiento (UP) y de control (UC).



NOTA: Basado en [DIA09]

5.4.4 Diseño Preliminar de CS1.

Arquitectura de conjunto: unidad de procesamiento (UP) y de control (UC).



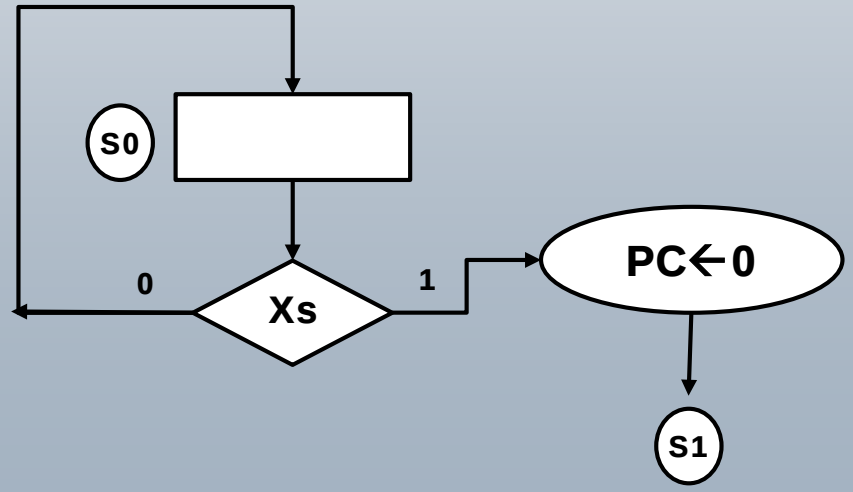
5.4.4 Diseño Preliminar de CS1.

Obtención de la Carta ASM de procesado, que especifique las secuencias de microoperaciones RT requeridas en el sistema: a) Inicialización, b) Fase de captación de Instrucciones, c) Fase de ejecución de Instrucciones.

a) Inicialización.

Al encender el computador sencillo **CS1,** consideraremos que éste se inicializará a un estado ***S0,*** donde esperará a que se active un "pulso" al valor "1" ,mediante un pulsador externo ***Xs,*** para que CS1 comience la ejecución de un programa.

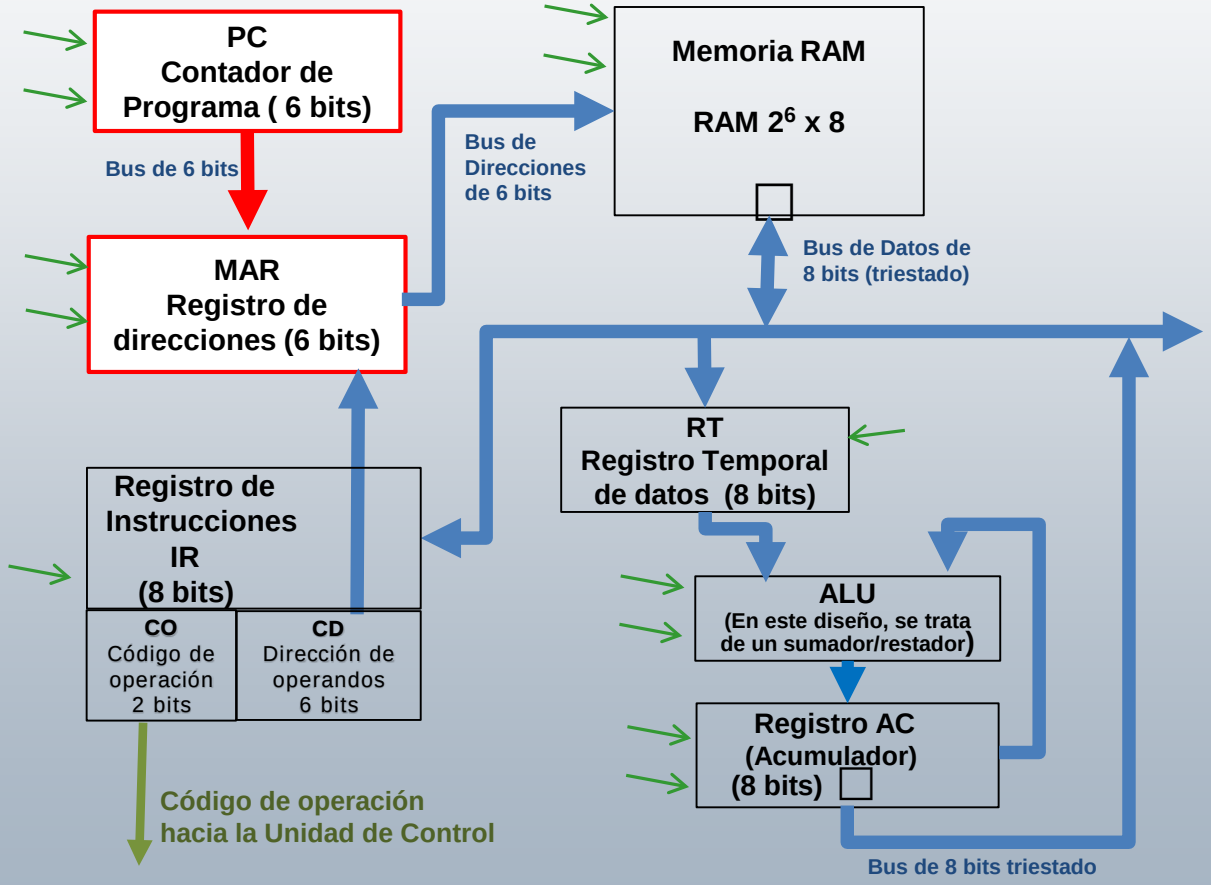
Nota:La duración ***Tp*** del pulso ***Xs=1*** debe cumplir que**:** ***Tc<Tp<3Tc*** donde ***Tc*** es el periodo de un ciclo de reloj.

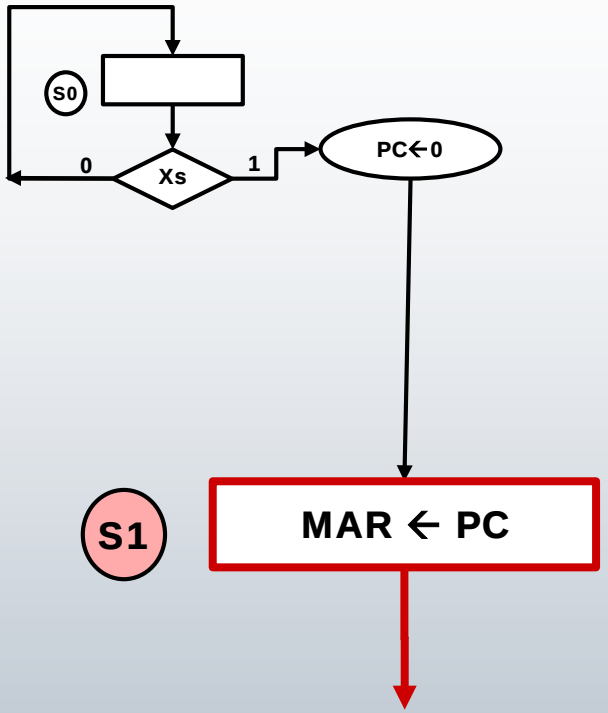


Obtención de la Carta ASM de procesado.

b) Fase de Captación de Instrucción.

Ciclo 1

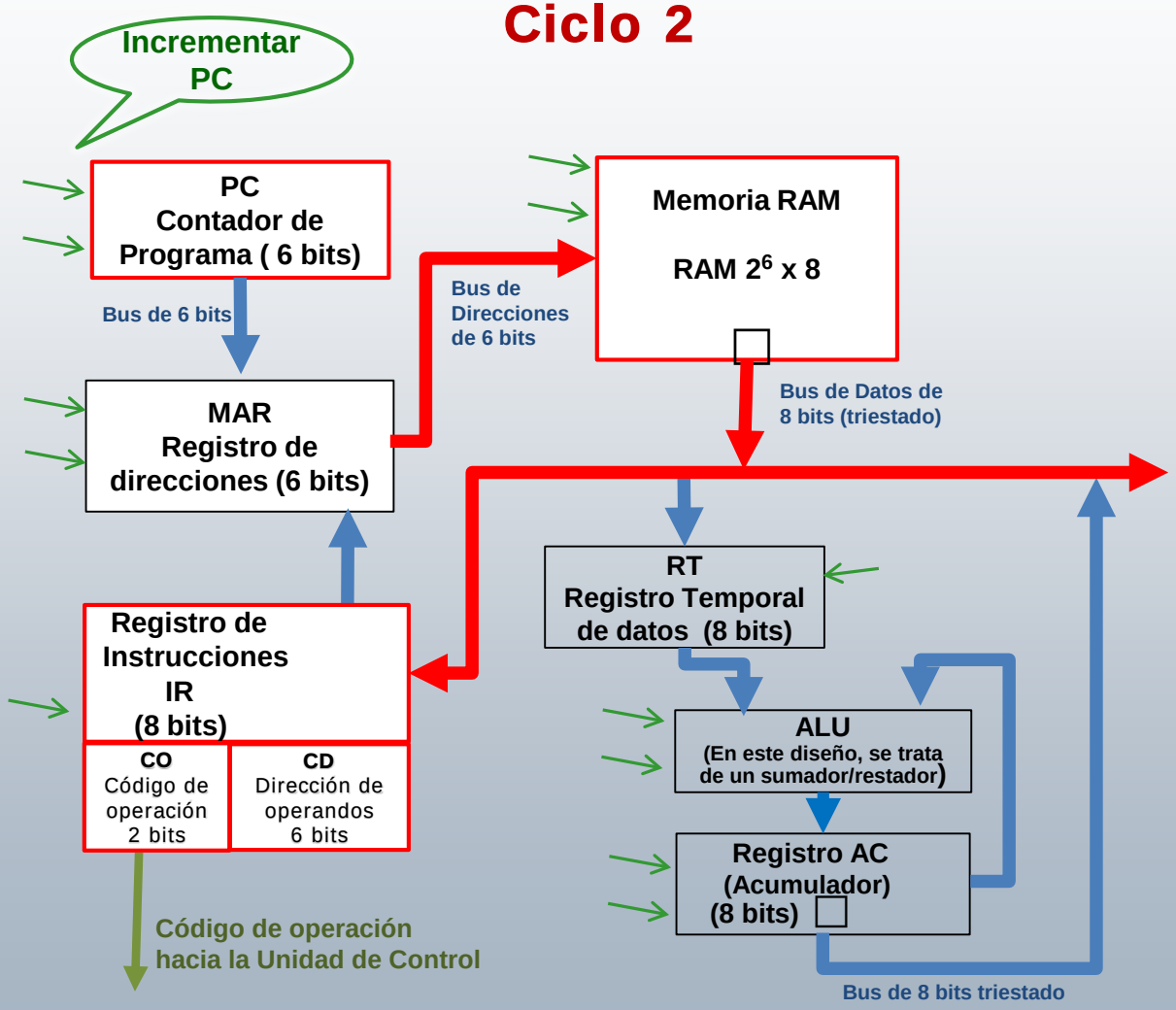


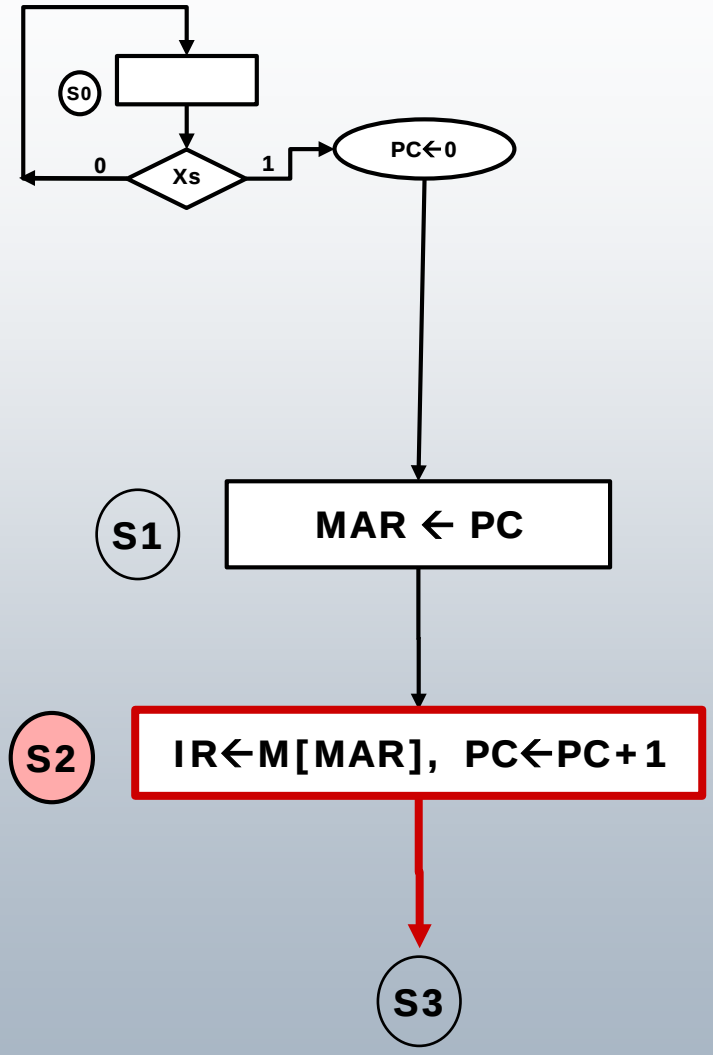


Obtención de la Carta ASM de procesado.

b) Fase de Captación de Instrucción.

Ciclo 2





Obtención de la Carta ASM de procesado.

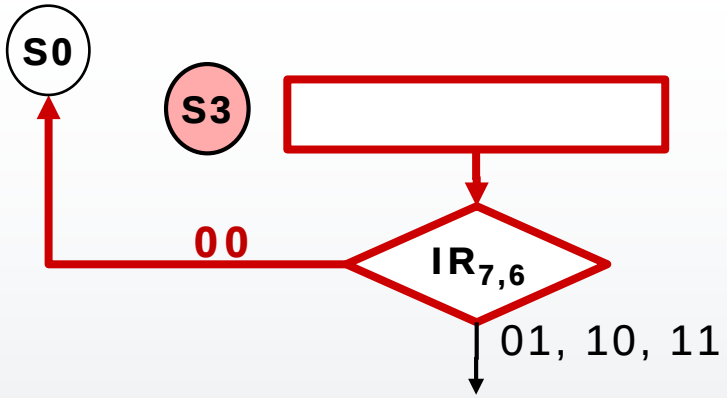
c) Fase de ejecución de Instrucciones.

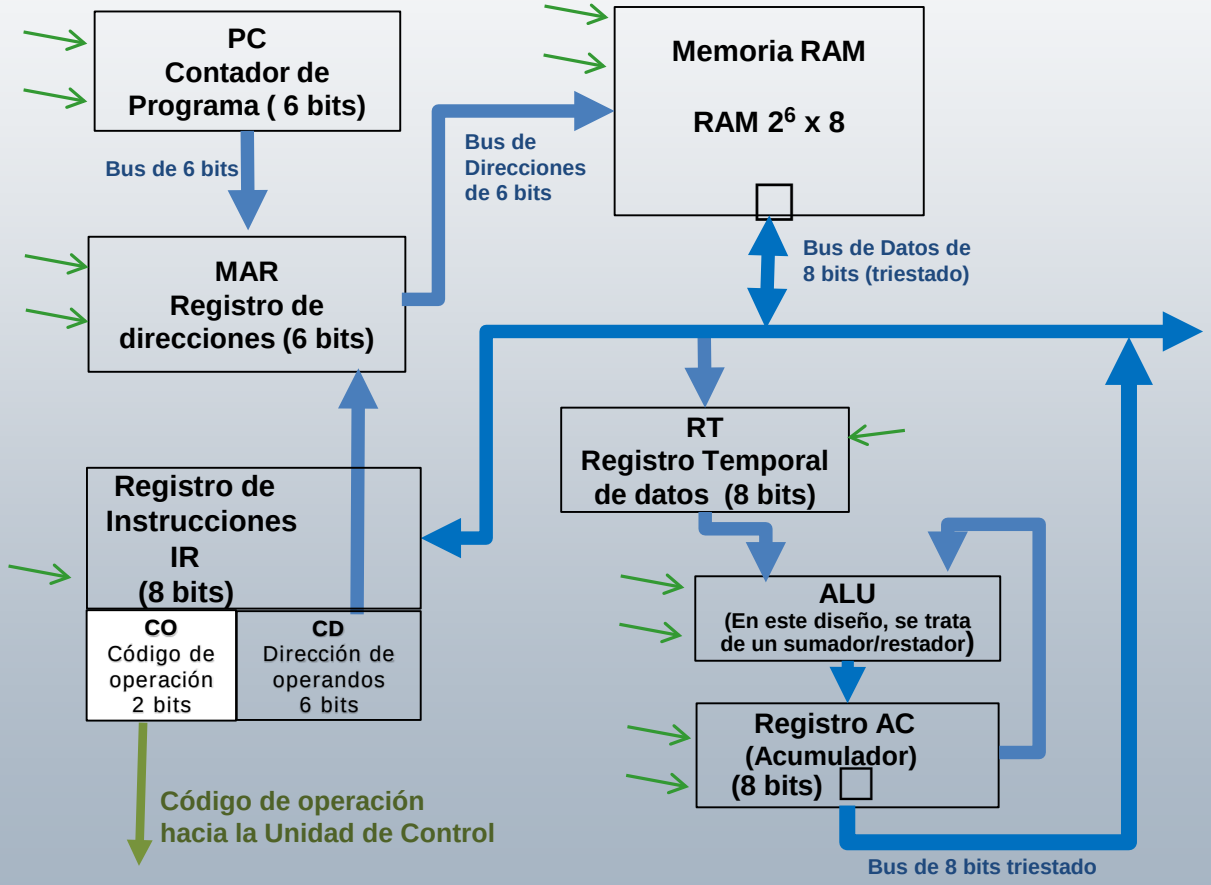
Fase de Ejecución: STOP

CO $IR_{7,6}$	Dirección Operando	Nemónico	Descripción de la Instrucción
00	X X X X X X	STOP	Fin Ejecución

Fase de Ejecución: STOP (Ciclo 3)

CO $IR_{7,6}$	Dirección Operando	Nemónico	Descripción de la Instrucción
00	X X X X X X	STOP	Fin Ejecución





Obtención de la Carta ASM de procesado.

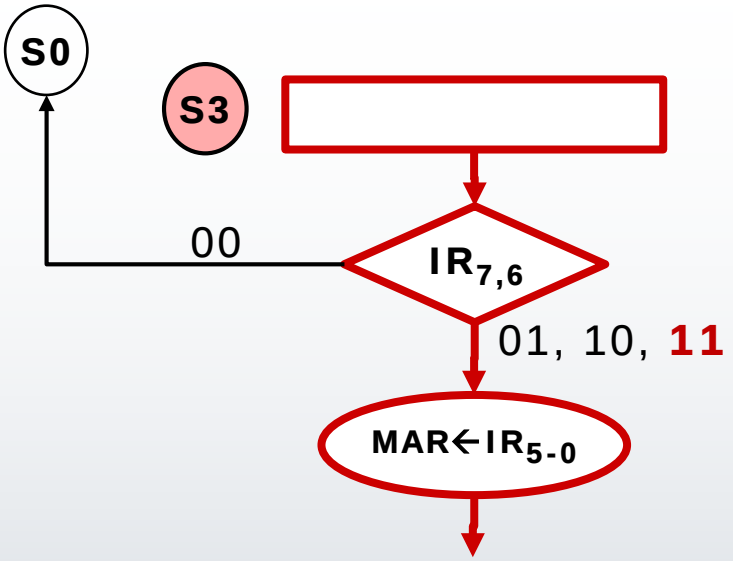
c) Fase de ejecución de Instrucciones.

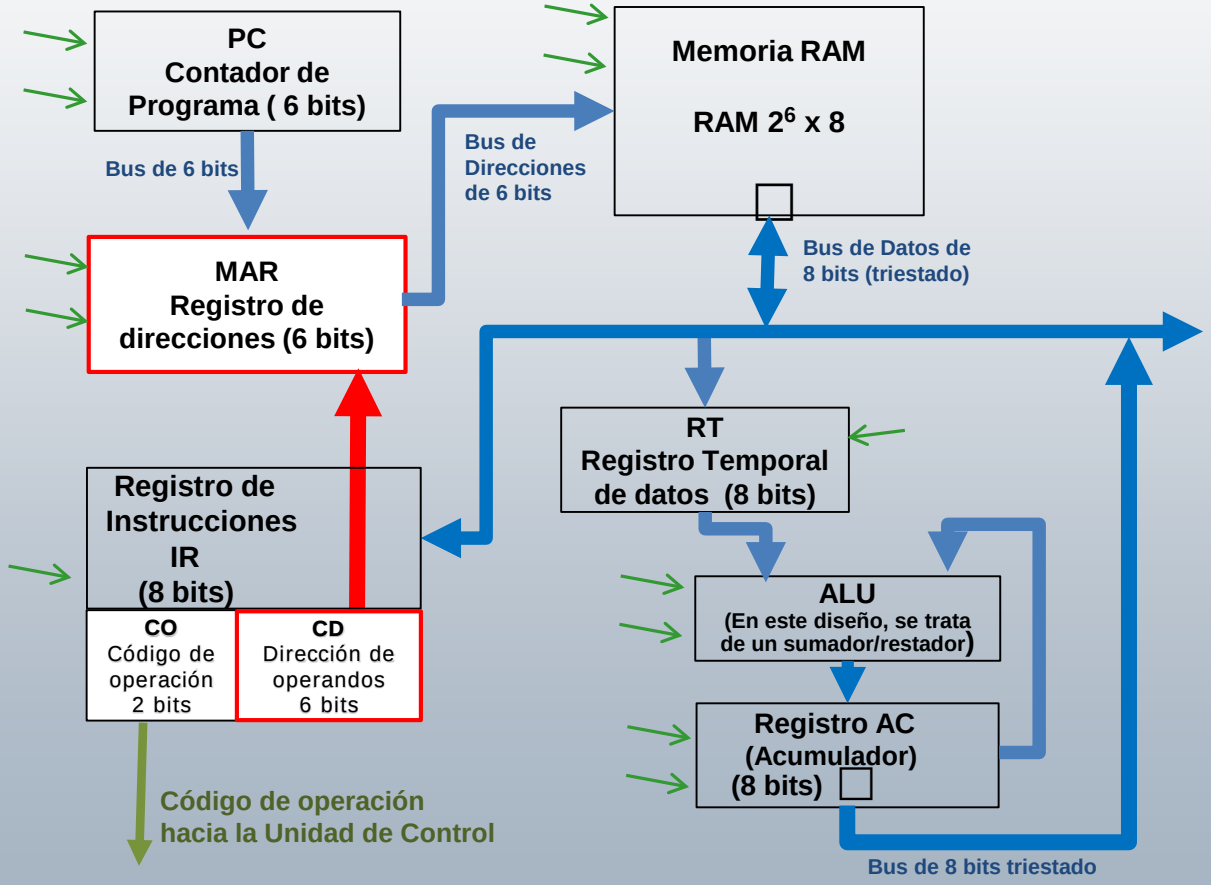
Fase de Ejecución: STA

CO $IR_{7,6}$	Dirección Operando	Nemónico	Descripción de la Instrucción
11	$A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	STA $A_{5\text{-}0}$	$M[A_{5\text{-}0}] \leftarrow AC$

Fase de Ejecución: STA (Ciclo 3)

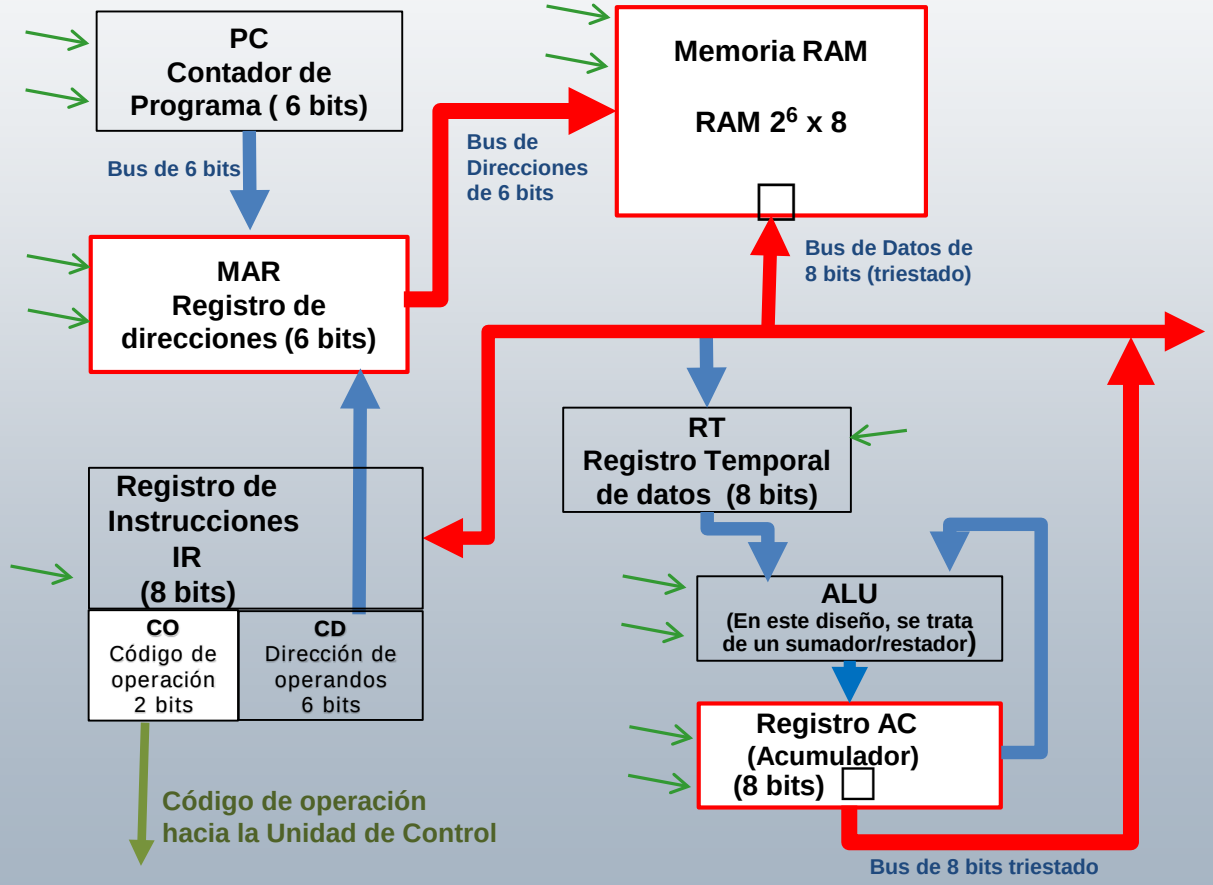
CO $IR_{7,6}$	Dirección Operando	Nemónico	Descripción de la Instrucción
11	$A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	STA $A_{5\text{-}0}$	$M[A_{5\text{-}0}] \leftarrow AC$

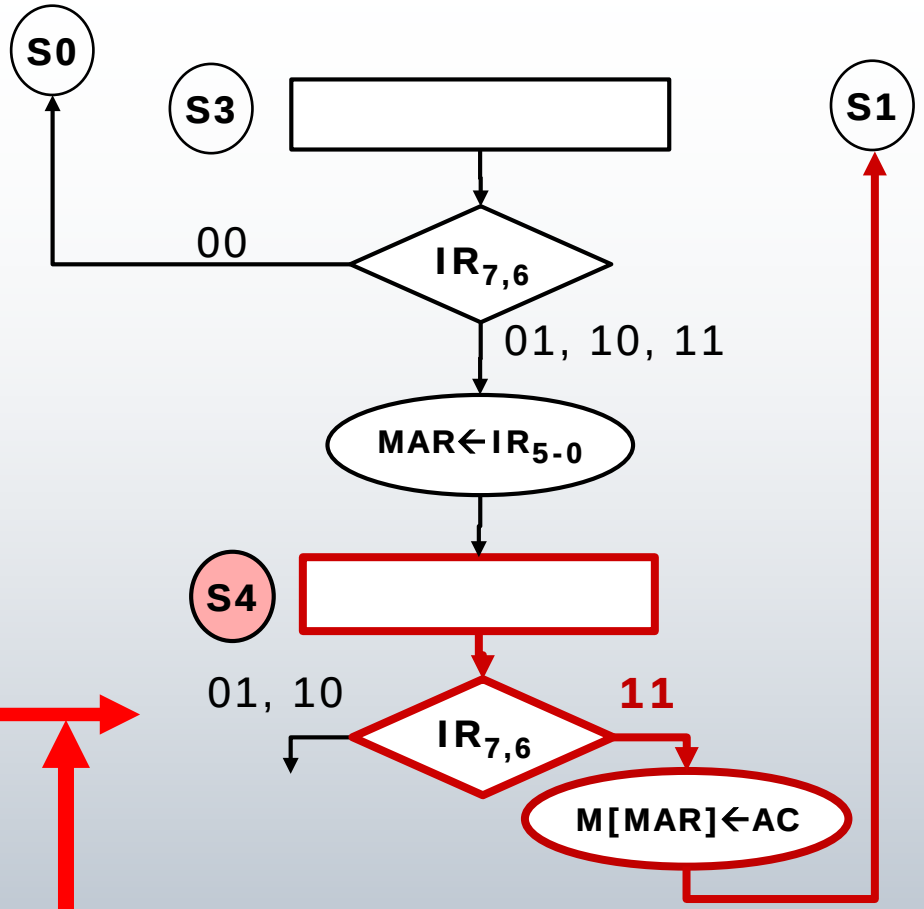




Fase de Ejecución: STA (Ciclo 4)

CO $IR_{7,6}$	Dirección Operando	Nemónico	Descripción de la Instrucción
11	$A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	STA $A_{5\text{-}0}$	$M[A_{5\text{-}0}] \leftarrow AC$





Obtención de la Carta ASM de procesado.

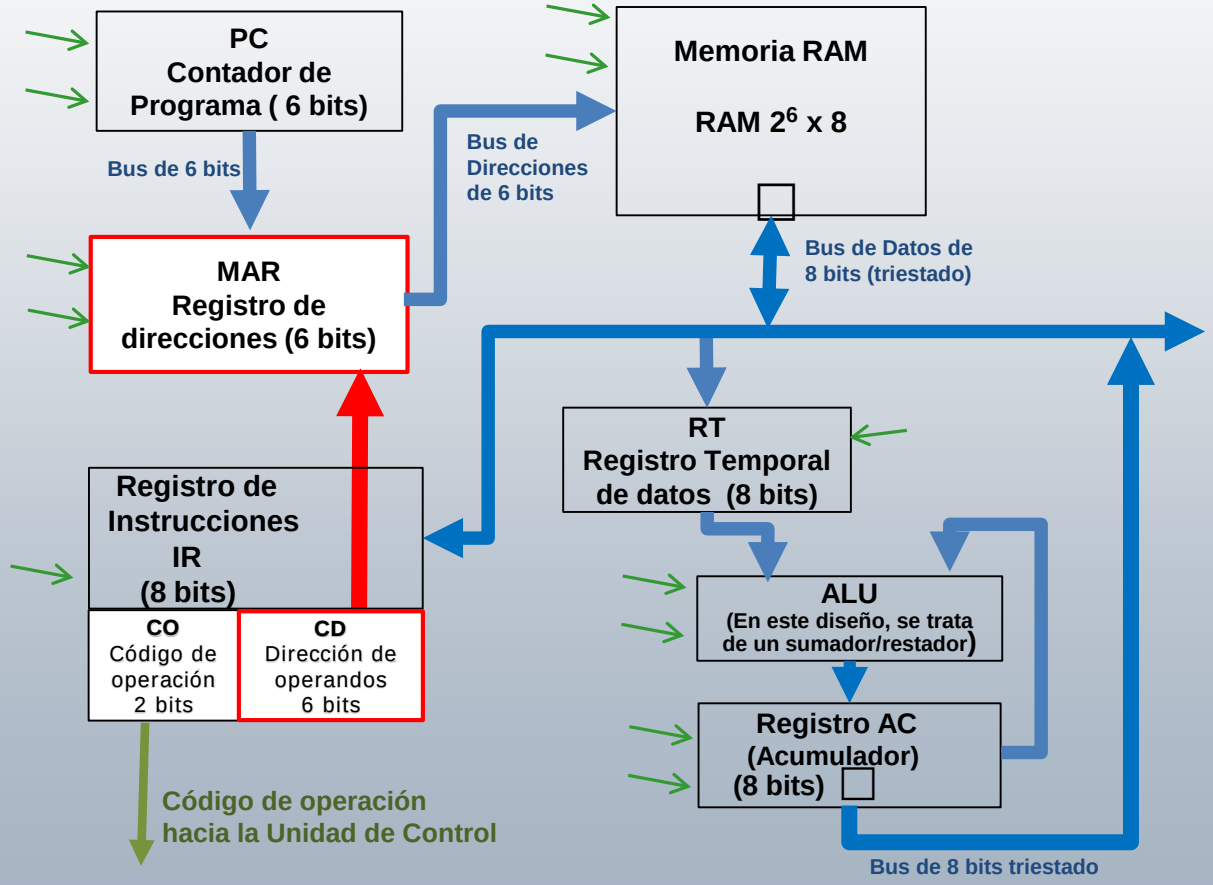
c) Fase de ejecución de Instrucciones.

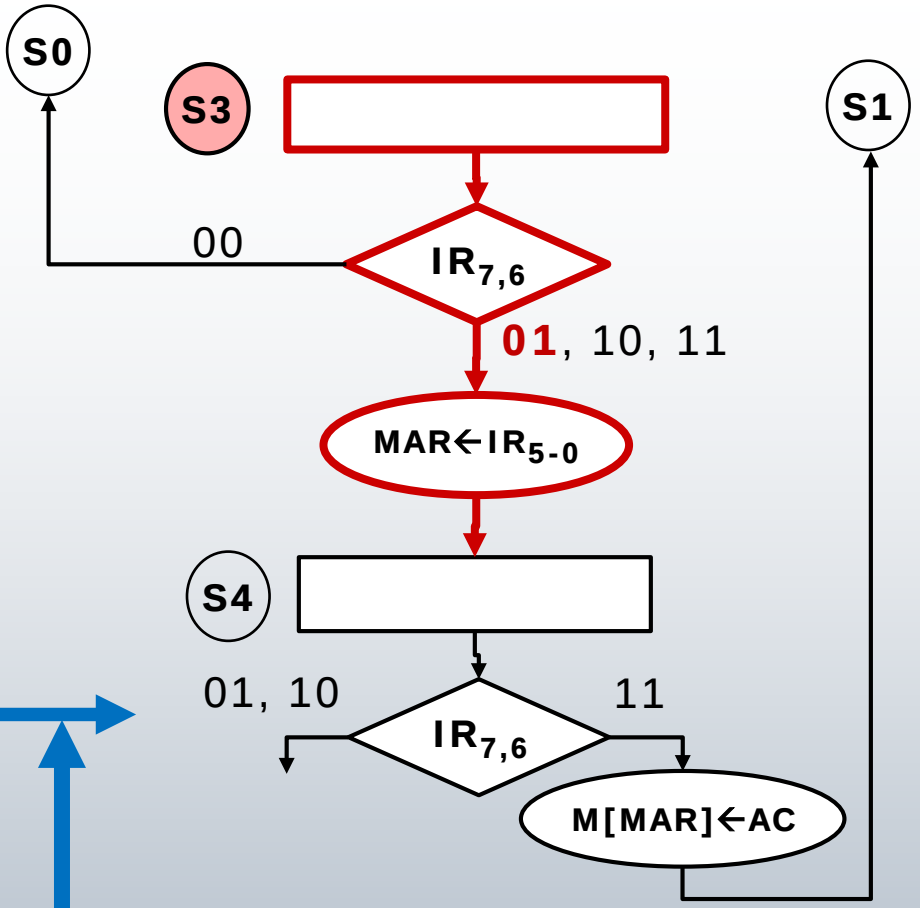
Fase de Ejecución: ADD

CO $IR_{7,6}$	Dirección Operando	Nemónico	Descripción de la Instrucción
01	$A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	$ADD A_{5-0}$	$AC \leftarrow AC + M[A_{5-0}]$

Fase de Ejecución: ADD (Ciclo 3)

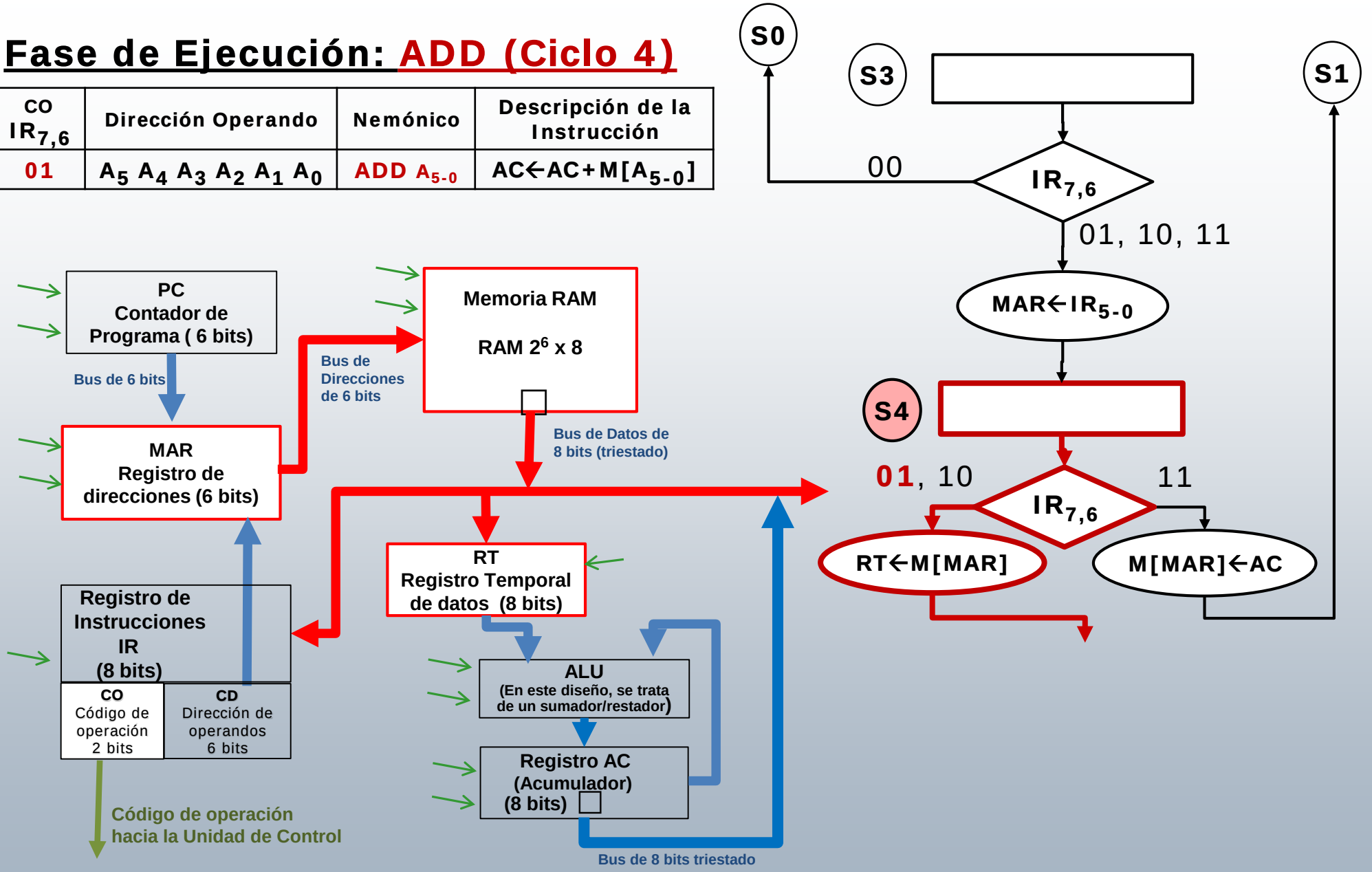
CO $IR_{7,6}$	Dirección Operando	Nemónico	Descripción de la Instrucción
01	$A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	ADD A_{5-0}	AC←AC+M[A_{5-0}]





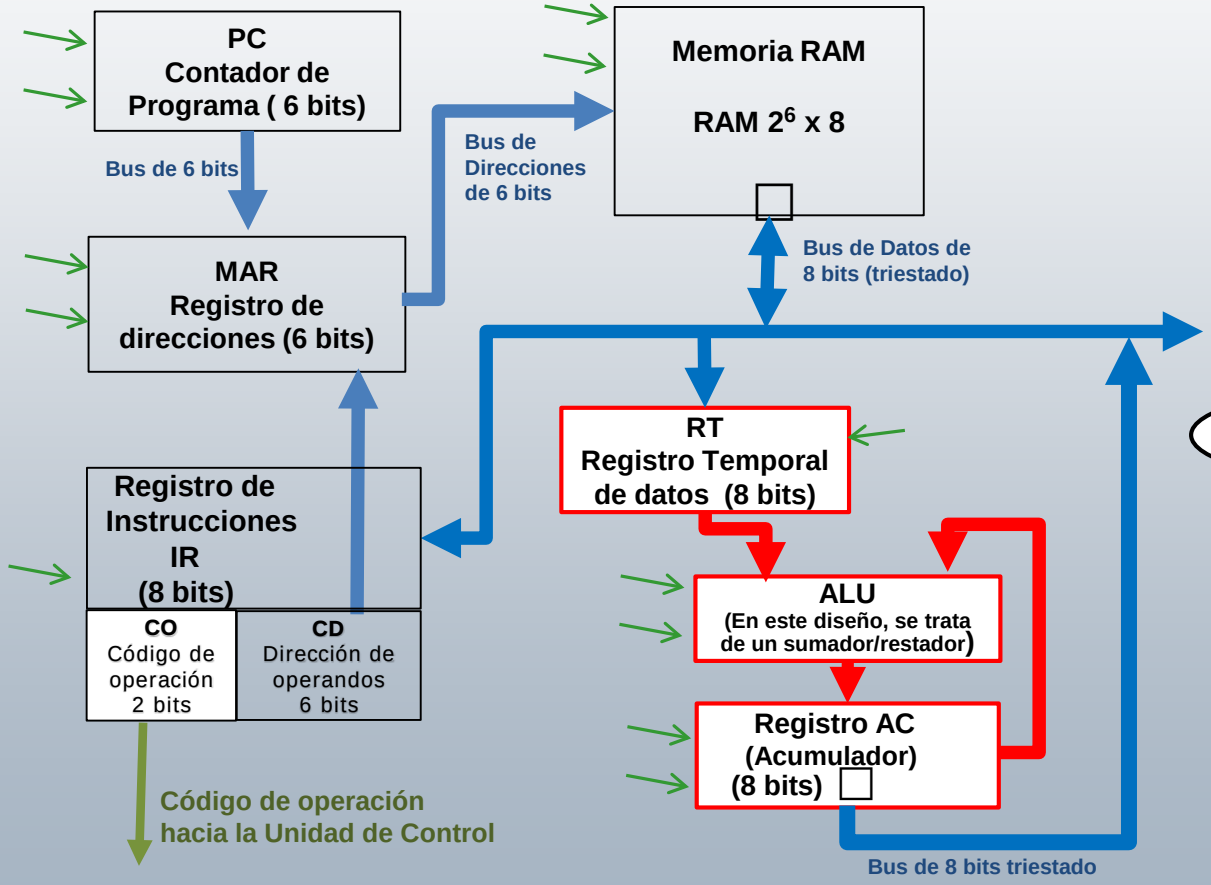
Fase de Ejecución: ADD (Ciclo 4)

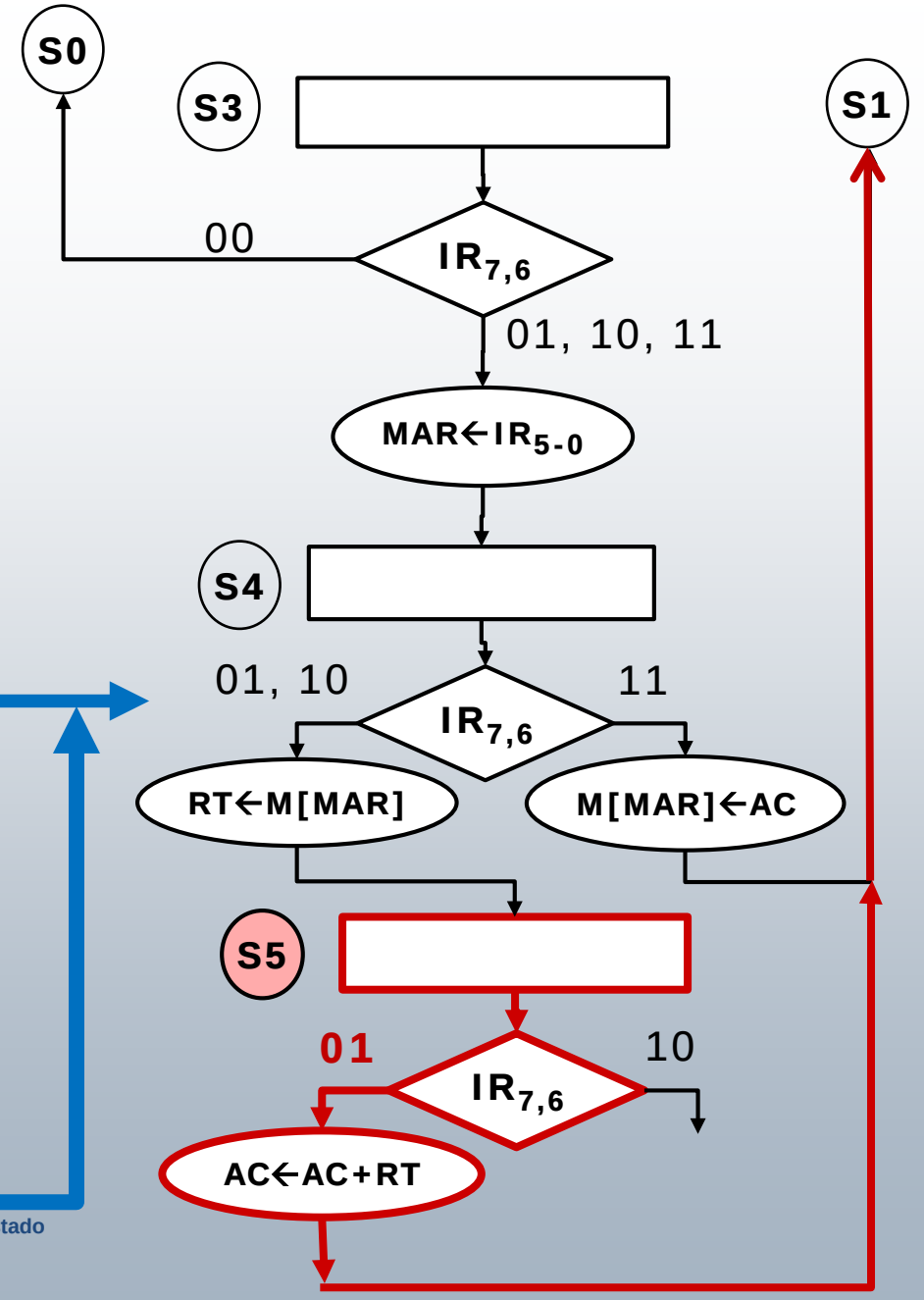
CO $IR_{7,6}$	Dirección Operando	Nemónico	Descripción de la Instrucción
01	$A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	ADD A_{5-0}	AC←AC+M[A_{5-0}]



Fase de Ejecución: ADD (Ciclo 5)

CO $IR_{7,6}$	Dirección Operando	Nemónico	Descripción de la Instrucción
01	$A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	ADD A_{5-0}	AC←AC+M[A_{5-0}]





Obtención de la Carta ASM de procesado.

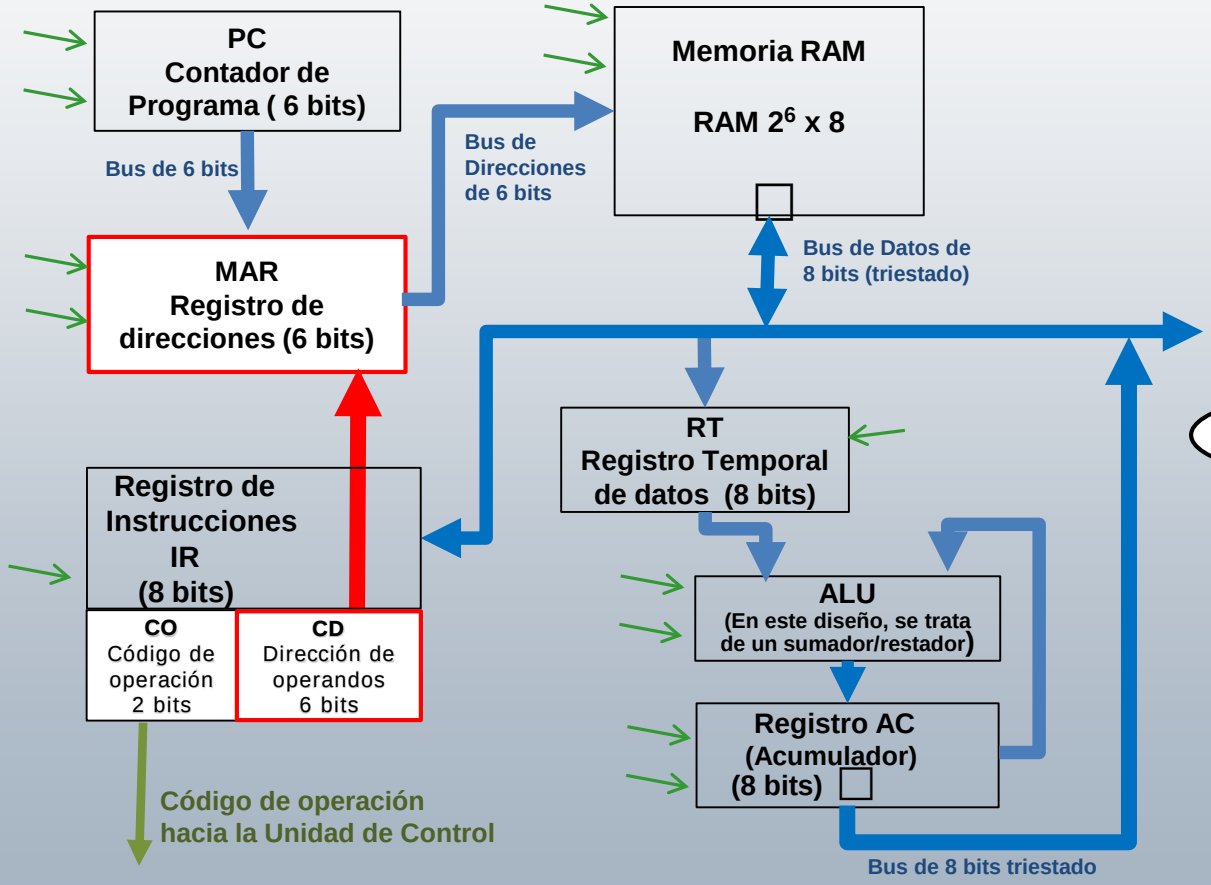
c) Fase de ejecución de Instrucciones.

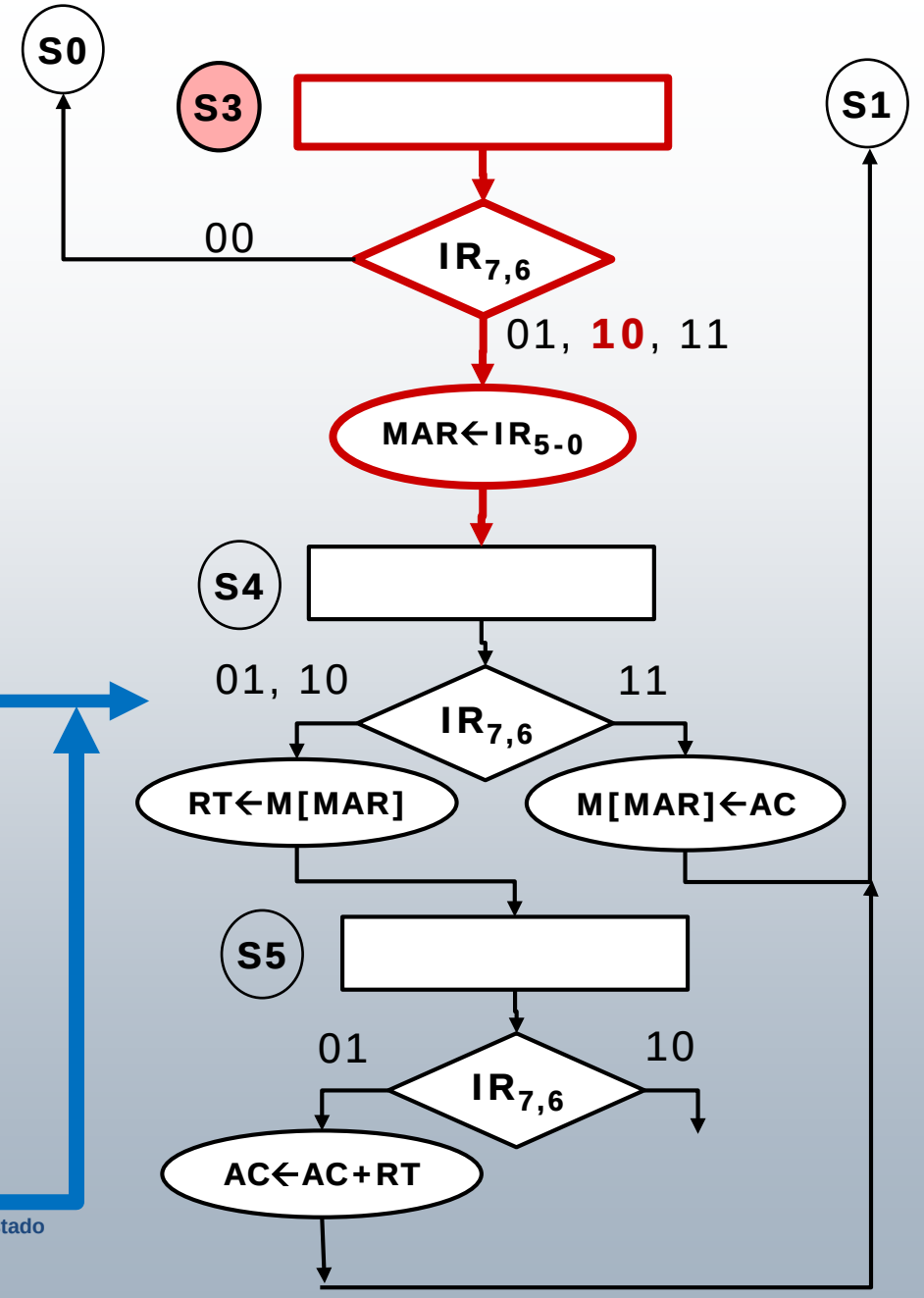
Fase de Ejecución: SUB

CO $IR_{7,6}$	Dirección Operando	Nemónico	Descripción de la Instrucción
10	$A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	SUB $A_{5\text{-}0}$	$AC \leftarrow AC\text{-}M[A_{5\text{-}0}]$

Fase de Ejecución: SUB (Ciclo 3)

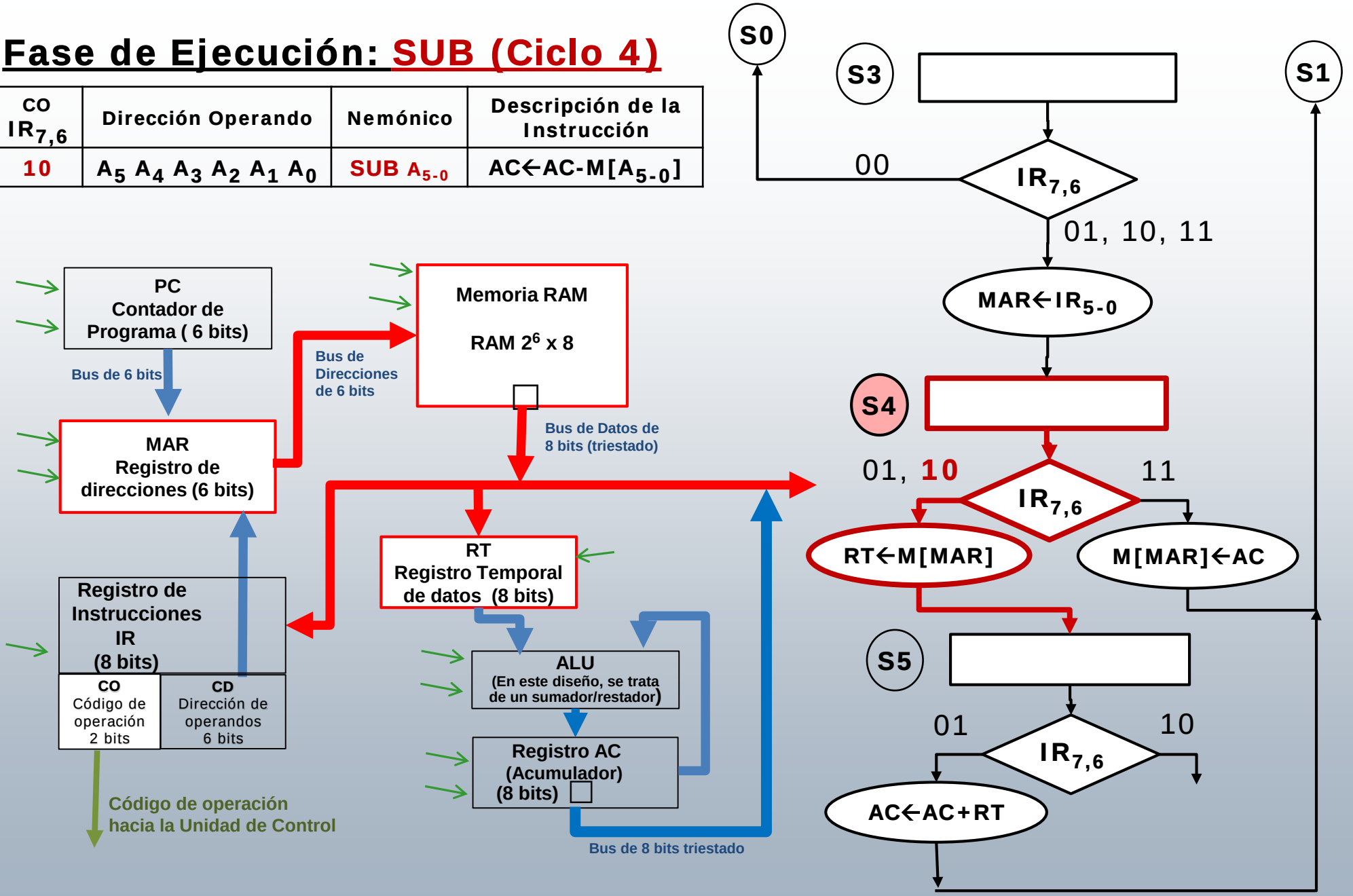
CO $IR_{7,6}$	Dirección Operando	Nemónico	Descripción de la Instrucción
10	$A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	SUB $A_{5\text{-}0}$	AC←AC-M[$A_{5\text{-}0}$]





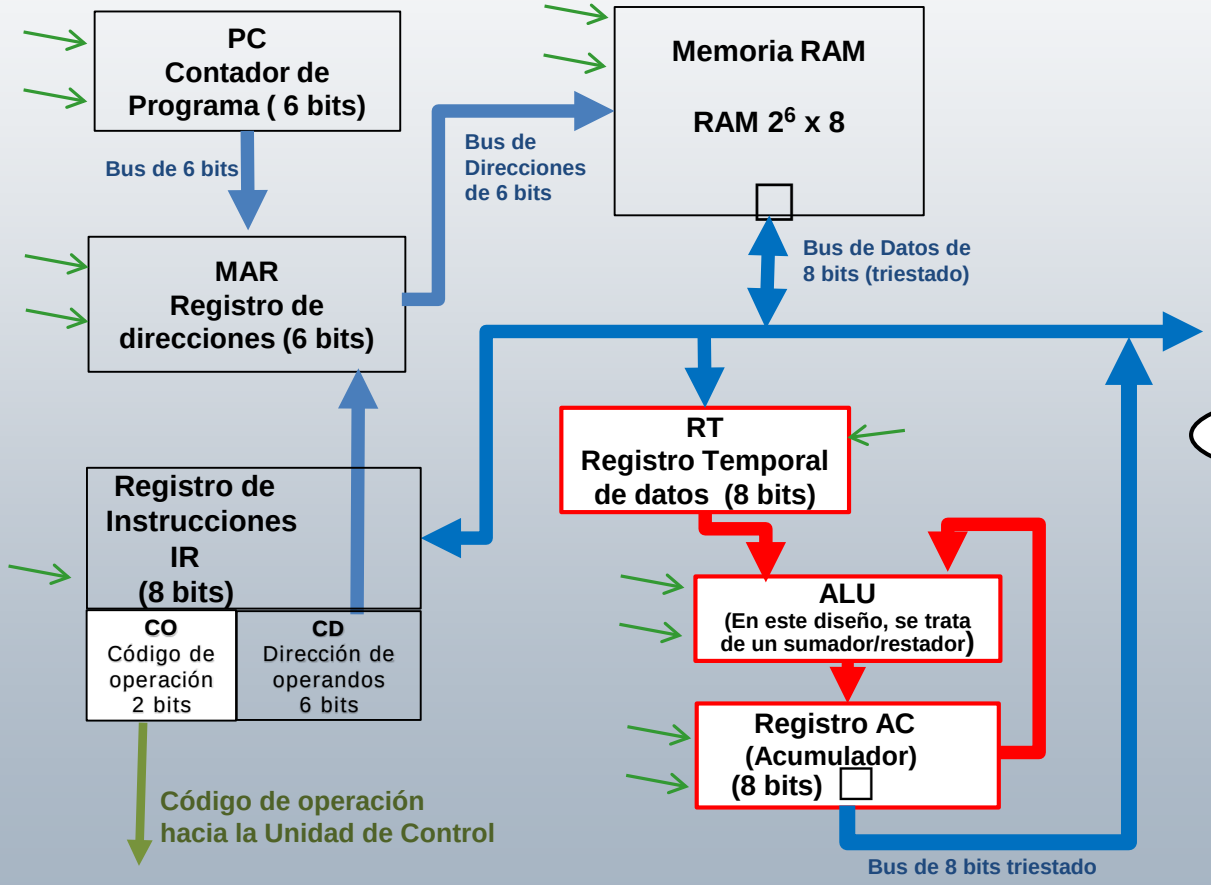
Fase de Ejecución: SUB (Ciclo 4)

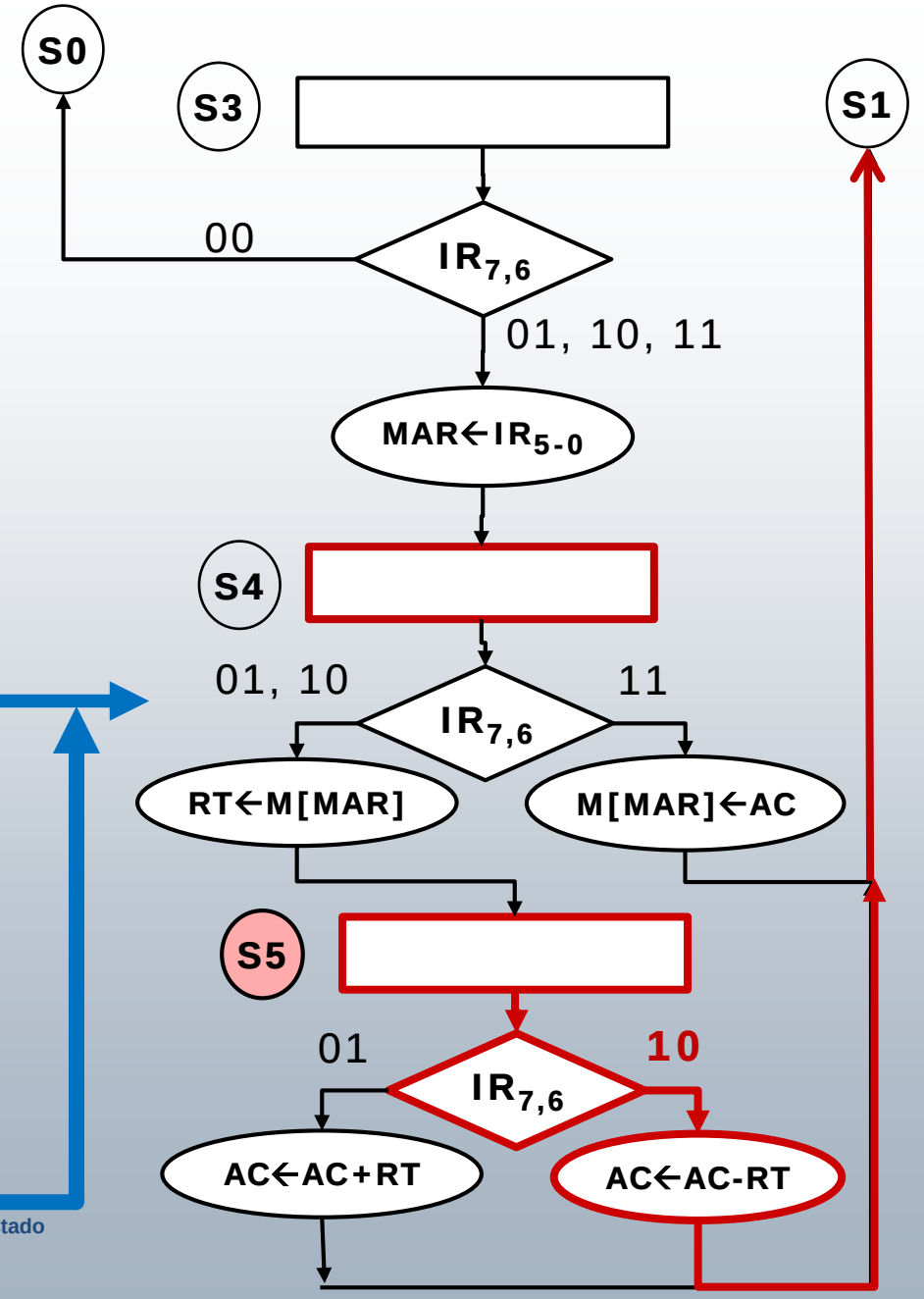
CO $IR_{7,6}$	Dirección Operando	Nemónico	Descripción de la Instrucción
10	$A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	SUB $A_{5\text{-}0}$	AC←AC-M[$A_{5\text{-}0}$]



Fase de Ejecución: SUB (Ciclo 5)

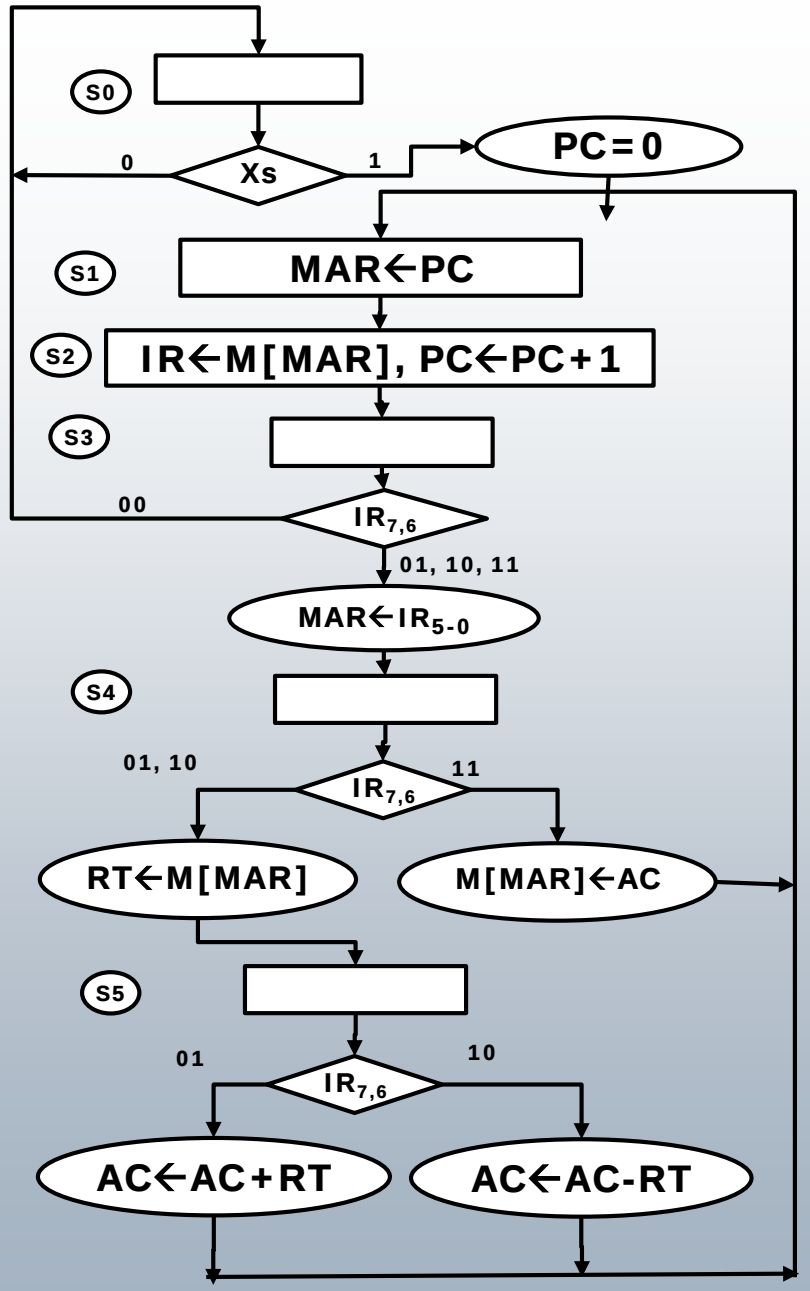
CO $IR_{7,6}$	Dirección Operando	Nemónico	Descripción de la Instrucción
10	$A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	SUB $A_{5\text{-}0}$	AC←AC-M[$A_{5\text{-}0}$]





Carta ASM de procesado de CS1 y observaciones.

- En los pasos anteriores hemos obtenido una Carta ASM de procesado (*Fig. de la derecha*) que cumple con los requerimientos de diseño del computador sencillo CS1. No obstante, hemos de indicar que no es la única solución, ya que podríamos haber obtenido otras cartas ASM de procesado equivalentes respecto a la funcionalidad de CS1.
- Otra observación es que en este diseño "descendente", no partimos de unos componentes de la UP previamente diseñados, sino de bloques funcionales que en etapas posteriores se irán detallando, para finalmente construirlos.

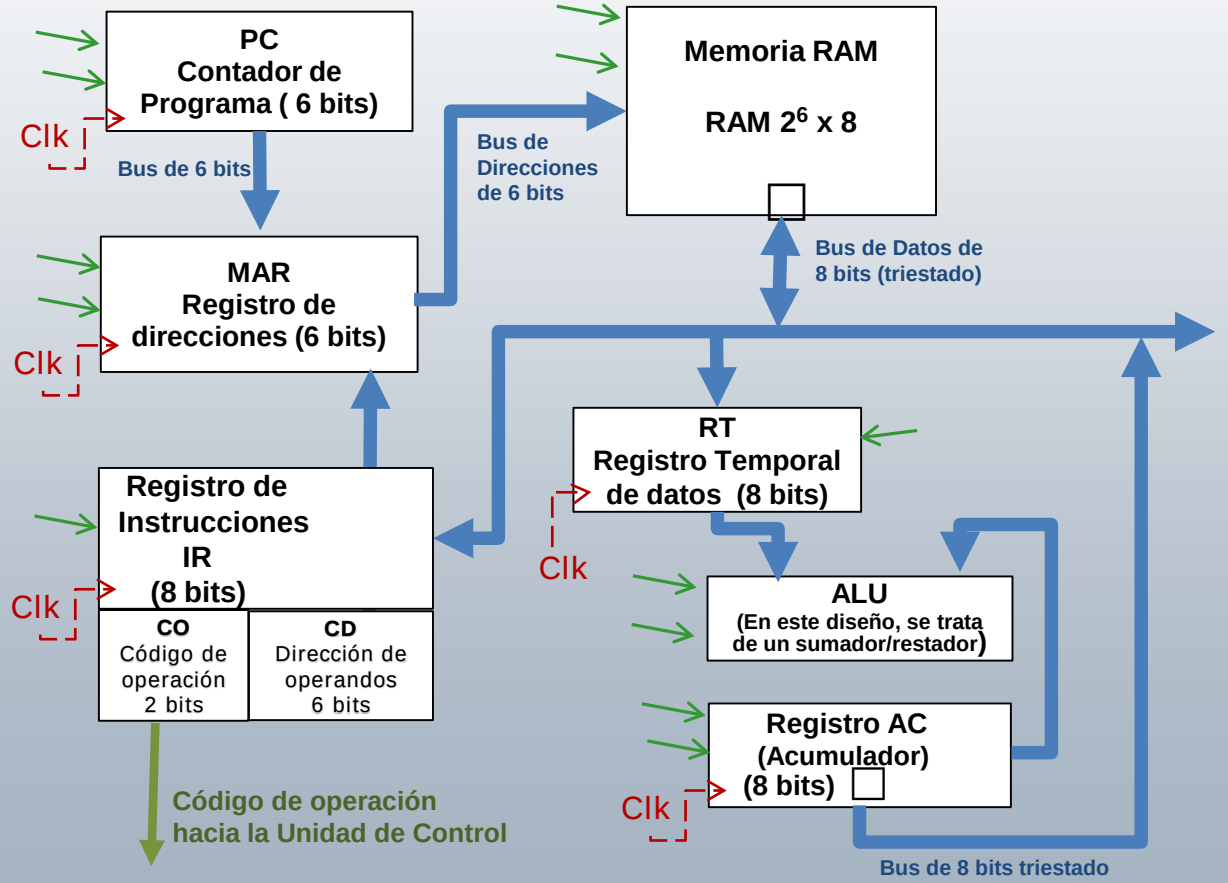


5.4.5 Diseño detallado de CS1. Diseño de la UP.

Se detalla el comportamiento de los componentes de la UP, sus señales de control y las palabras de control para realizar las microoperaciones.

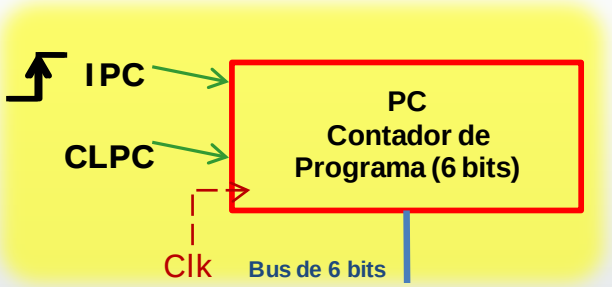
- **Unidad de procesamiento**

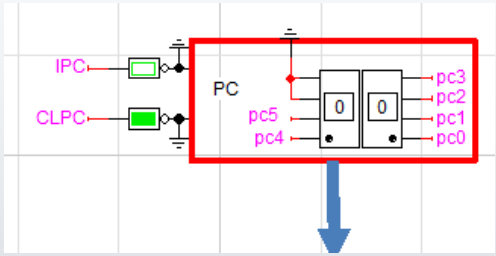
  Desde un diseño a nivel RT, conviene definir a la UP como la unidad que incluye los componentes: contadores, registros, ALU y memoria, así como los buses necesarios para realizar las microoperaciones de transferencias necesarias en el sistema computador.



5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

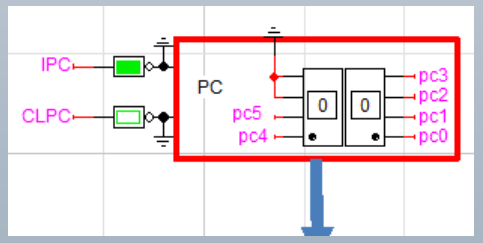
- **Contador de programa PC.**

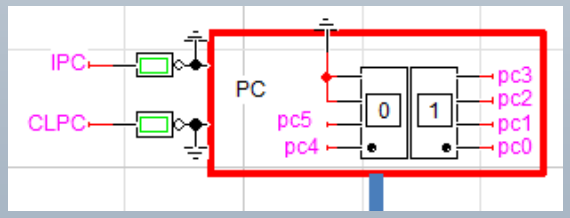




CLPC	IPC	Contador de programa PC
0	0	Mantiene valor
0	1	PC←PC+1 en el siguiente flanco de reloj.
1	0	PC=0 (Asíncrona)
1	1	Prohibido

Ejemplo: Antes del flanco → Después del flanco

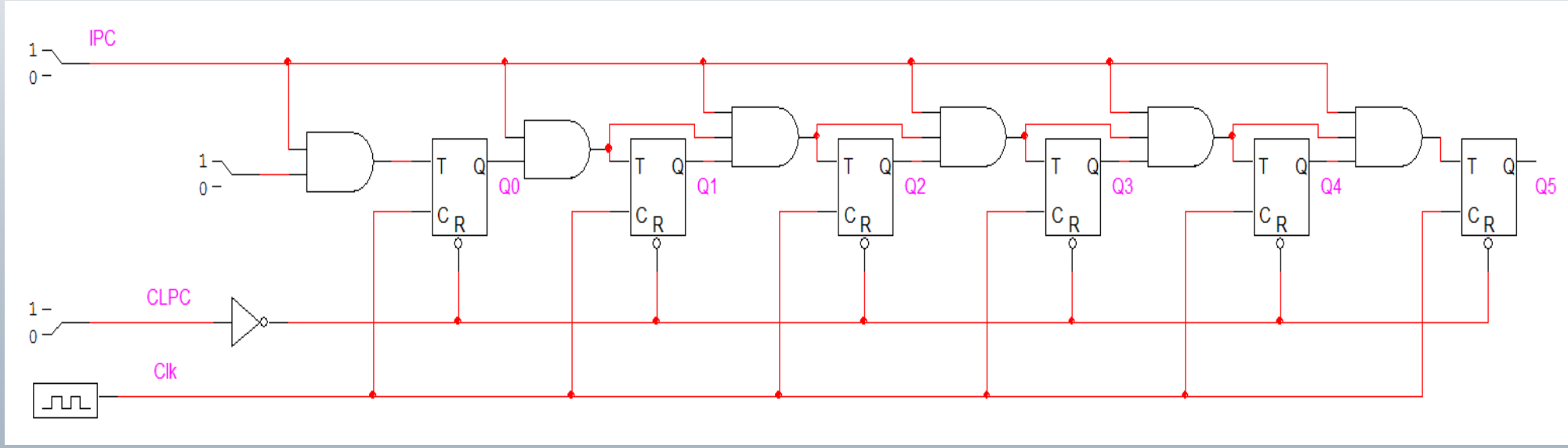




5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

- **Contador de programa PC.**

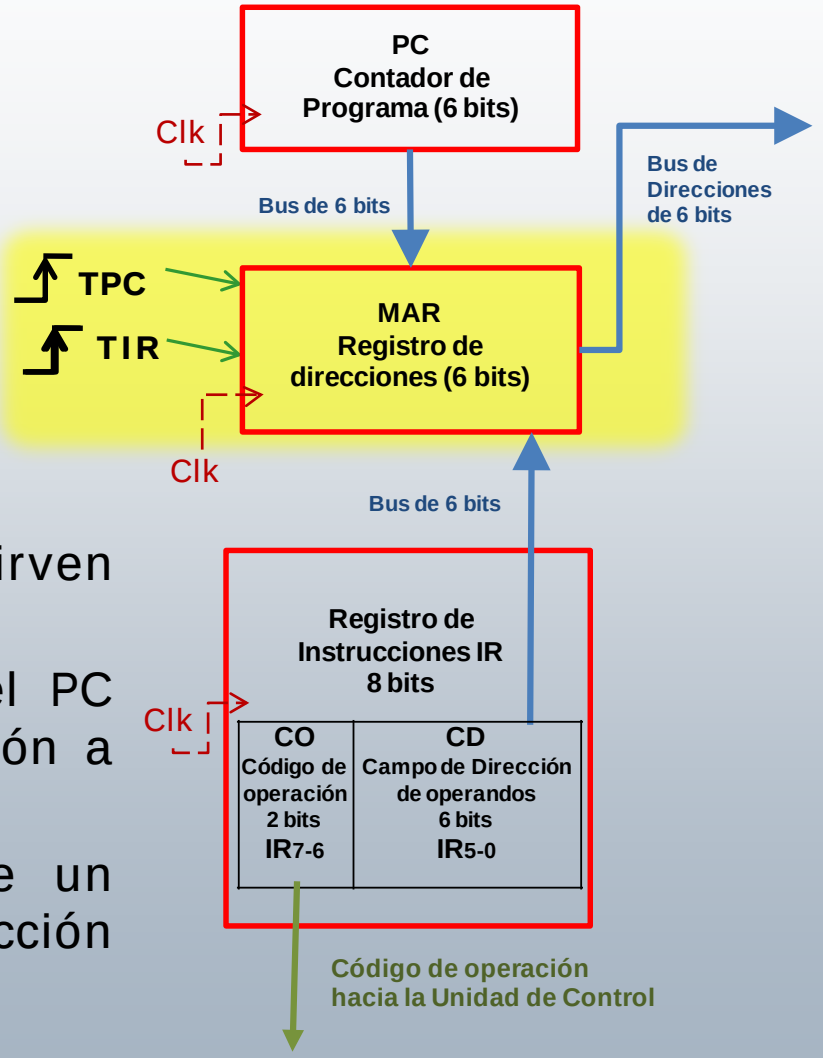
CLPC	IPC	Contador de programa PC
0	0	Mantiene valor
0	1	PC←PC+1 en el siguiente flanco de reloj.
1	0	PC=0 (Asíncrona)
1	1	Prohibido



5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

- **MAR: Registro de direcciones.**

TIR	TPC	Registro de direcciones MAR
0	0	Mantiene valor
0	1	MAR←PC
1	0	MAR←IR_{5-0} (Campo CD)
1	1	Prohibido



Las señales de control de MAR, sirven para:

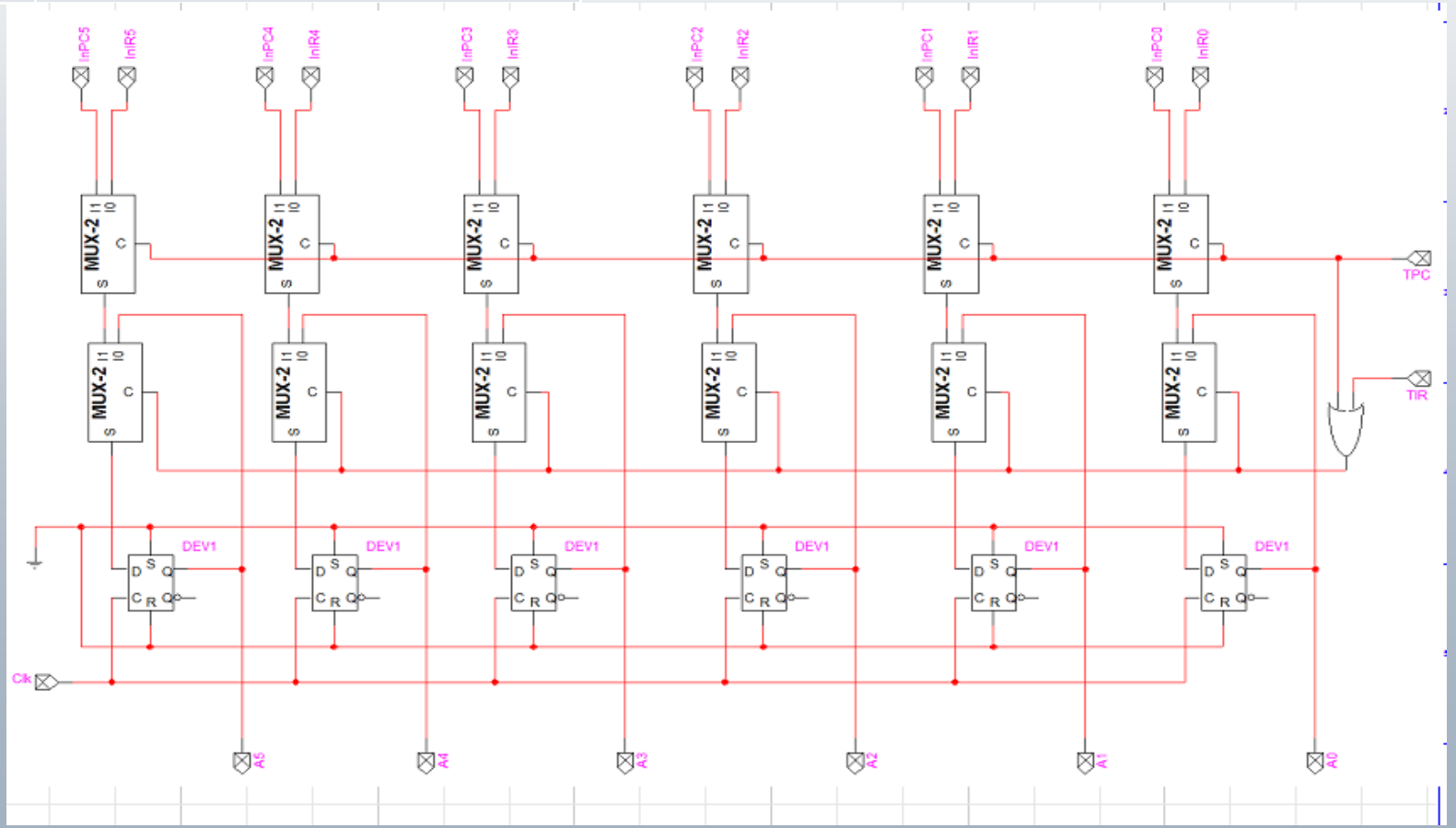
a) Transferir a MAR el contenido del PC (dirección de memoria de la instrucción a ejecutar)

b) Transferir a MAR la dirección de un operando desde el registro de instrucción IR (campo CD).

5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

- **MAR: Registro de direcciones.**

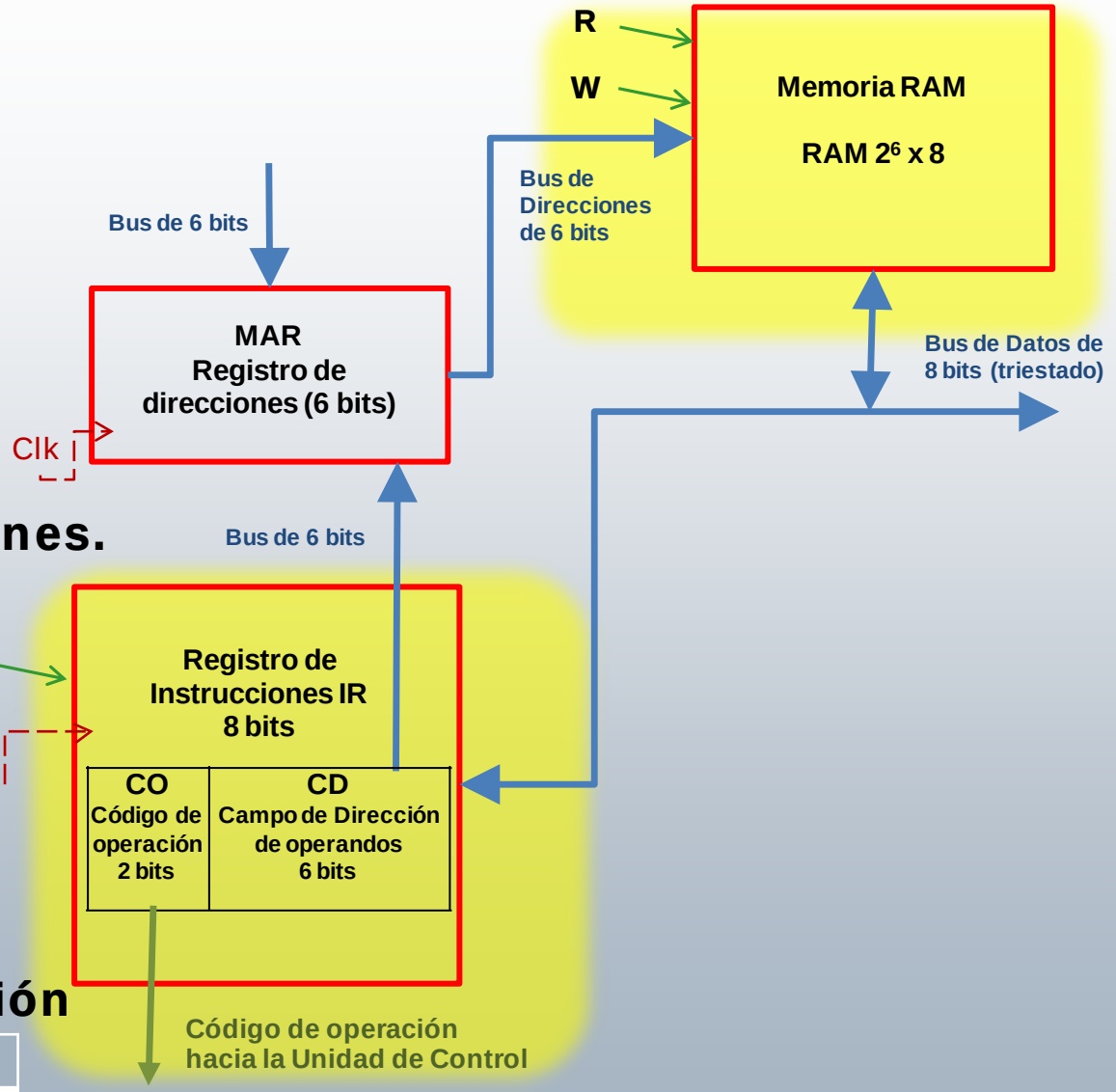
TIR	TPC	Registro de direcciones MAR
0	0	Mantiene valor
0	1	MAR←PC
1	0	MAR←IR_{5-0} (Campo CD)
1	1	Prohibido



5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

Memoria Principal MP.

R	W	Menoría Principal MP
0	0	Bus Datos interno de la RAM en alta impedancia.
0	1	M[MAR] ← Bus Datos (Escritura)
1	0	Bus Datos ← M[MAR] (Lectura)
1	1	Prohibido



IR: Registro de Instrucciones.

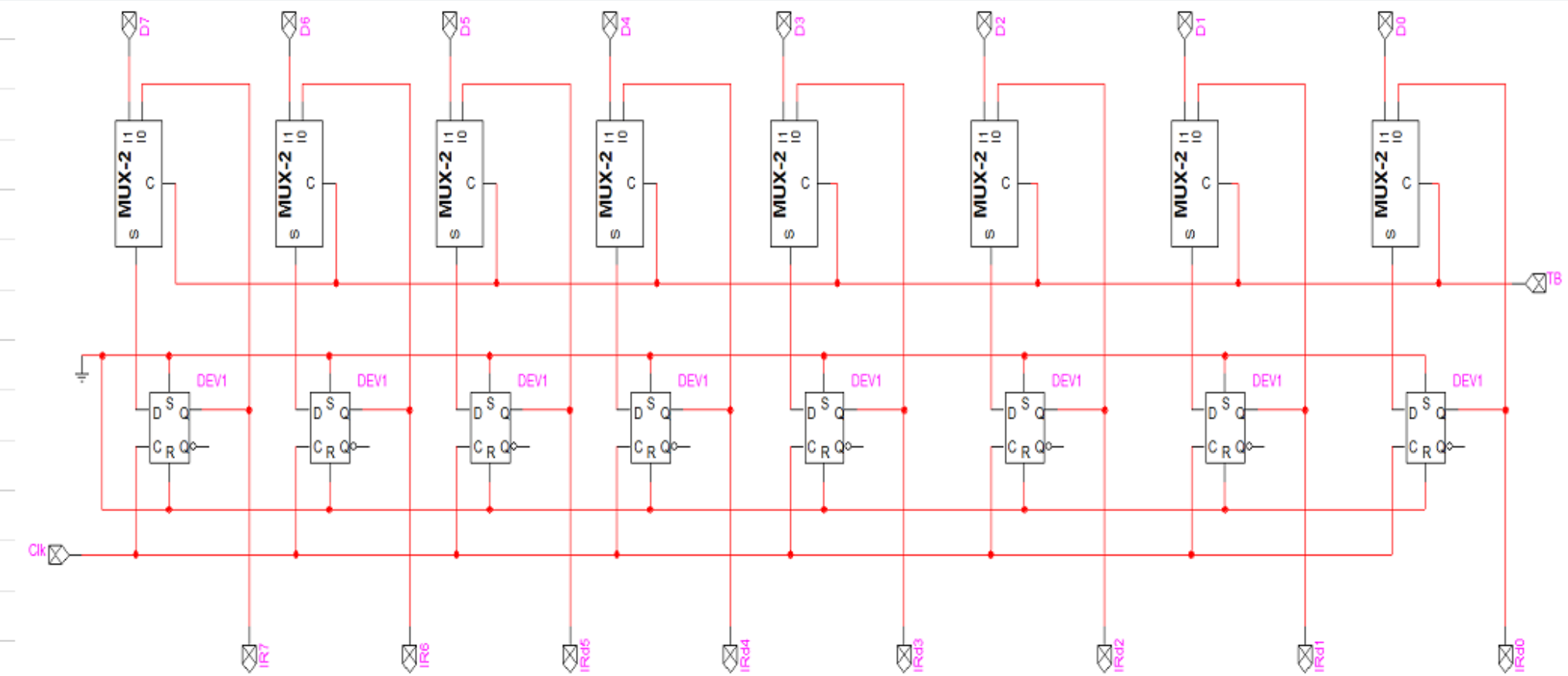
TB	Registro de Instrucciones IR
0	Mantiene valor
1	IR ← Bus Datos.

Microoperación RT necesaria en la fase de captación de una instrucción

TB	R	W	Microoperación RT
1	1	0	IR ← M[MAR]

5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

TB	Registro de Instrucciones IR
0	Mantiene valor
1	IR ← Bus Datos.



Registro temporal RT.

WT	Registro Temporal RT
0	Mantiene valor
1	RT ← Bus Datos.

Registro Acumulador AC.

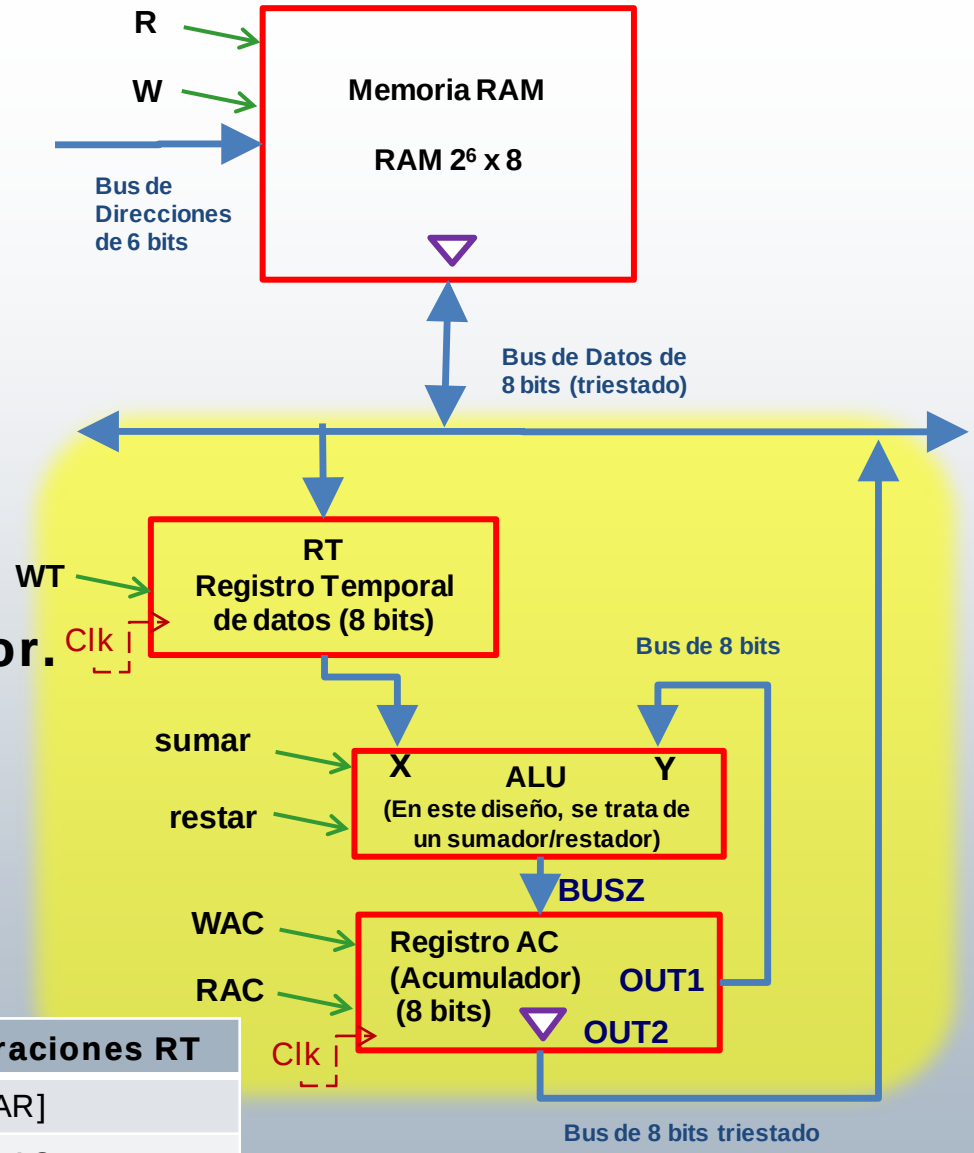
RAC	WAC	Acumulador AC
0	0	Mantiene valor, OUT2 en alta impedancia
0	1	AC ← BUSZ, OUT2 en alta impedancia
1	0	OUT2 = [AC]
1	1	Prohibido Siempre OUT1 = [AC]

ALU. Basta con un sumador restador.

sumar	restar	Acumulador AC
0	0	No importa
0	1	Z = Y - X Resta
1	0	Z = Y + X Suma
1	1	Prohibido

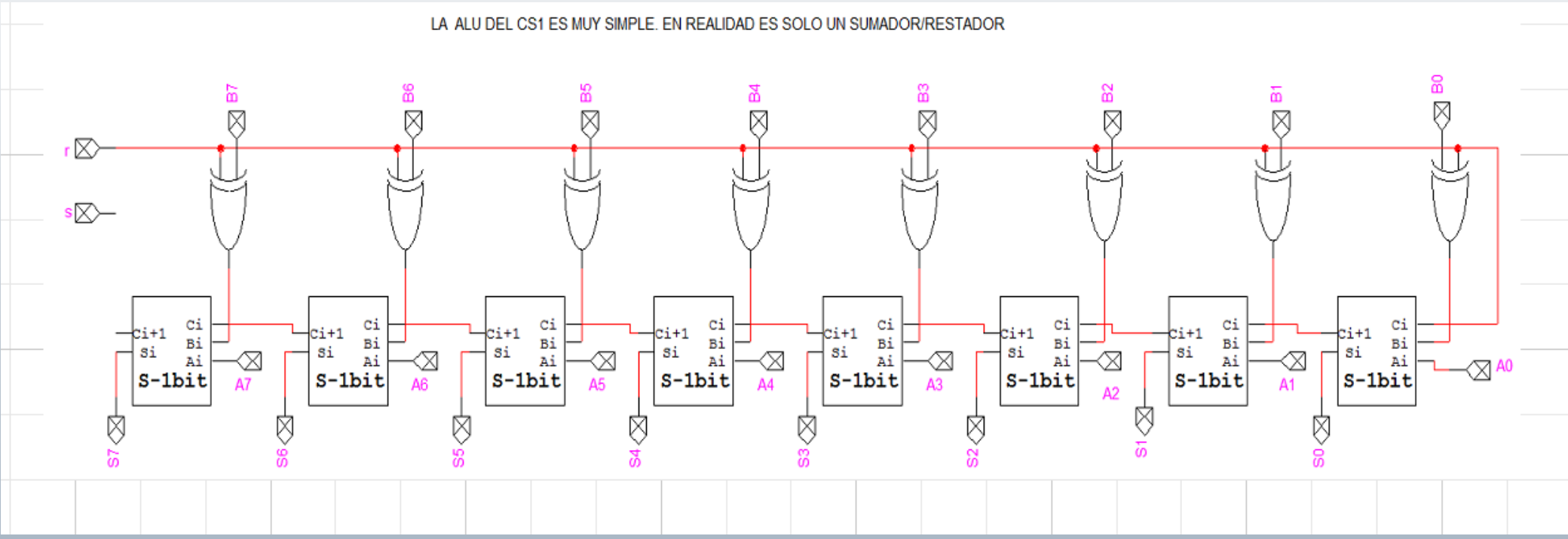
Microoperaciones RT

WT	WAC	RAC	sumar	restar	R	W	Microoperaciones RT
1	0	0	--	--	1	0	RT ← M[MAR]
0	0	1	--	--	0	1	M[MAR] ← AC
0	1	0	1	0	0	0	AC ← AC+RT
0	1	0	0	1	0	0	AC ← AC-RT



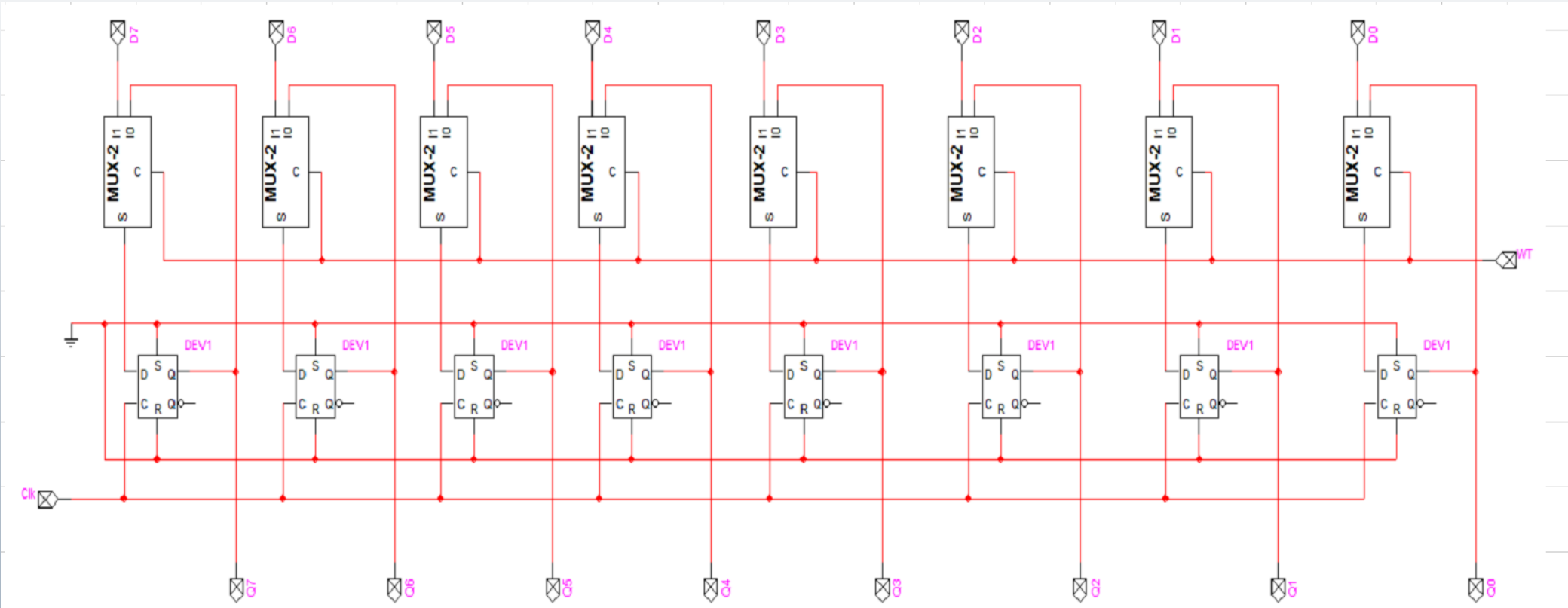
5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

sumar	restar	Acumulador AC
0	0	No importa
0	1	Z= Y - X Resta
1	0	Z= Y + X Suma
1	1	Prohibido



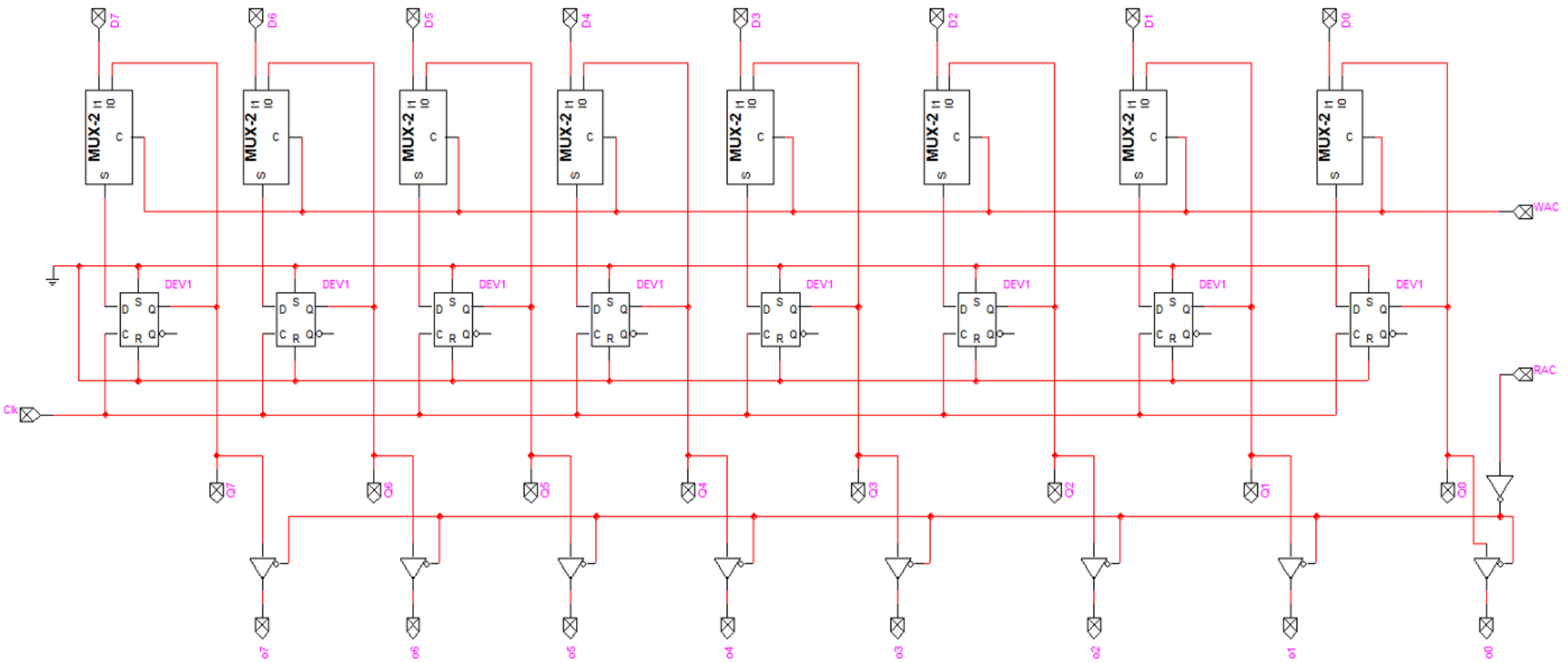
5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

WT	Registro Temporal RT
0	Mantiene valor
1	RT ← Bus Datos.



5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

RAC	WAC	Acumulador AC
0	0	Mantiene valor, OUT2 en alta impedancia
0	1	AC ← BUSZ, OUT2 en alta impedancia
1	0	OUT2 = [AC]
1	1	Prohibido
		Siempre OUT1 = [AC]



5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

Microoperaciones RT.

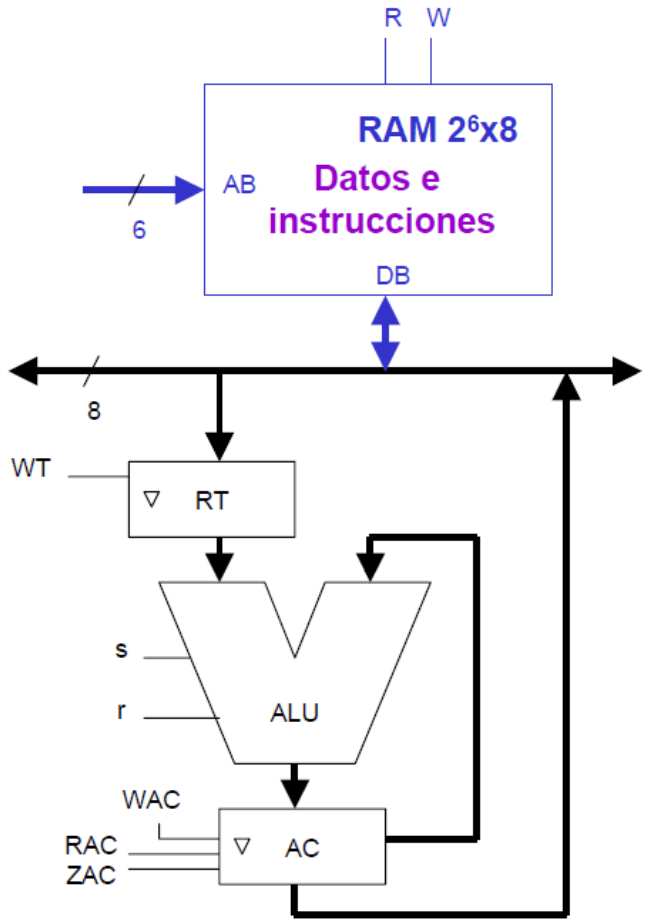
- En la tabla se resumen microoperaciones de transferencia que la UP puede satisfacer, y que son suficientes para realizar el conjunto de las cuatro instrucciones del computador simple.
- En la fila (*) se muestra una transferencia múltiple de dos transferencias simples que pueden realizarse en paralelo en un mismo ciclo máquina.
- En color amarillo se indican las señales de control a activar al valor “1” para ejecutar cada microoperación

Microoperaciones RT	**Señales de control**											
	PC		**MAR**		**IR**	**RT**	**AC**		ALU		**Memoria Principal**	
	CLPC	IPC	TIR	TPC	TB	WT	WAC	RAC	sumar	restar	R	W
PC=0	**1**	**0**	0	0	0	0	0	0	-	-	0	0
(*) PC←PC+1 IR←M[MAR]	**0**	**1**	0	0	**1**	0	0	0	-	-	**1**	0
MAR←PC	0	0	0	**1**	0	0	0	0	-	-	0	0
MAR←IR$_{5-0}$	0	0	**1**	0	0	0	0	0	-	-	0	0
RT←M[MAR]	0	0	0	0	0	**1**	0	0	-	-	**1**	0
M[MAR]←AC	0	0	0	0	0	0	0	**1**	-	-	0	**1**
AC←AC+RT	0	0	0	0	0	0	**1**	0	**1**	0	0	0
AC←AC-RT	0	0	0	0	0	0	**1**	0	0	**1**	0	0

5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

CS1: Unidad de datos

- **Aumentar la cantidad de datos a manejar:**
  - **incrementar el número de registros**
  - **incluir una memoria RAM** ← CS1
- **Ejecutar las instrucciones de forma autónoma:**
  - **Almacenar las instrucciones en una memoria:**
    - **en la memoria donde se almacenan los datos: arquitectura *Von Neumann*** ← CS1
    - **en otra memoria, específica para instrucciones: arquitectura *Harvard***
  - **La instrucción a ejecutar se extrae de la memoria y se almacena temporalmente en el *registro de instrucción* (IR, *Instruction Register*)**



NOTA: Basado en [DIA09]

5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

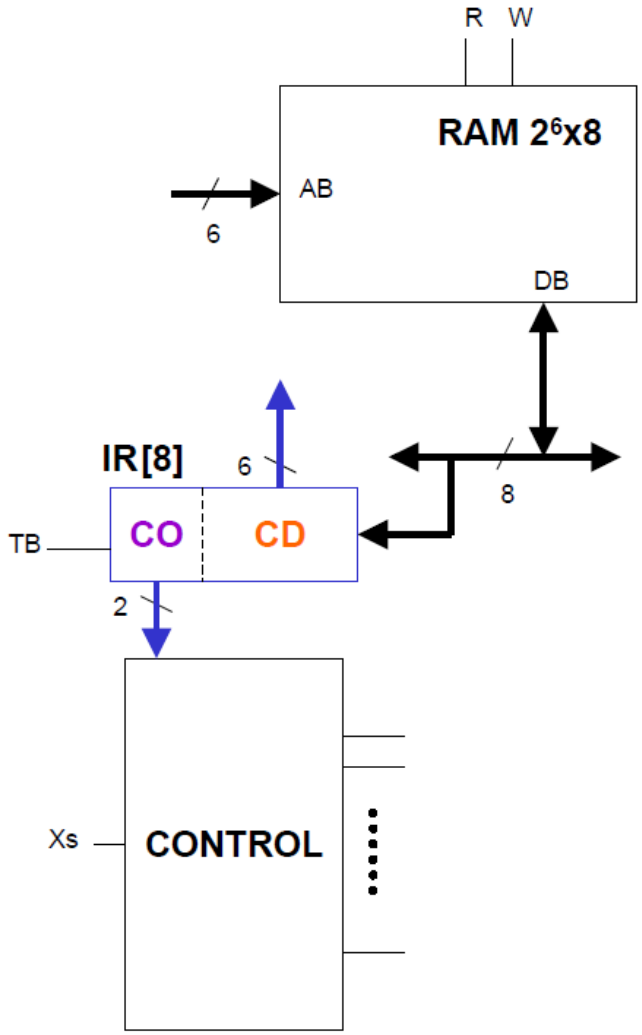
CS1: Unidad de datos (2)

NOTA: Basado en [DIA09]

Cada instrucción tiene dos partes:

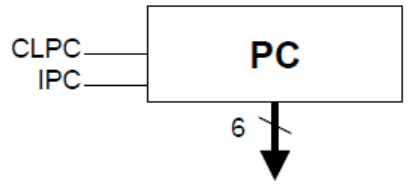
- **Código de operación (CO): tipo de operación a realizar**
- **Operandos: datos con los que la instrucción va a trabajar:**
  - **A veces la instrucción no necesita operandos; otras veces, los operandos están implícitos y no hay que incluirlos en la instrucción**
  - **CS1** **En el CS1, las instrucciones incluyen un campo de dirección de operando (CD) donde se indica la dirección de la RAM donde se encuentra el operando**

CS1 • **|IR|=8bits; |CD|=6bits ⇒ |CO|=2bits ⇒ 4 tipos de instrucciones**



CS1: Unidad de datos (3)

- **Una vez ejecutada la instrucción actual, hay que extraer la siguiente desde memoria y almacenarla en IR**
- **Para ello es necesario un registro puntero de instrucciones (PC, *Program Counter*):**
  - **almacena la dirección de memoria donde se encuentra la siguiente instrucción**
  - **una vez transferida la nueva instrucción a IR, hay que incrementar PC (IPC) para apuntar a la siguiente**
  - **inicialmente, PC vale 0 (CLPC), por lo que la primera instrucción siempre estará en la dirección 0 de la RAM**



NOTA: Basado en [DIA09]

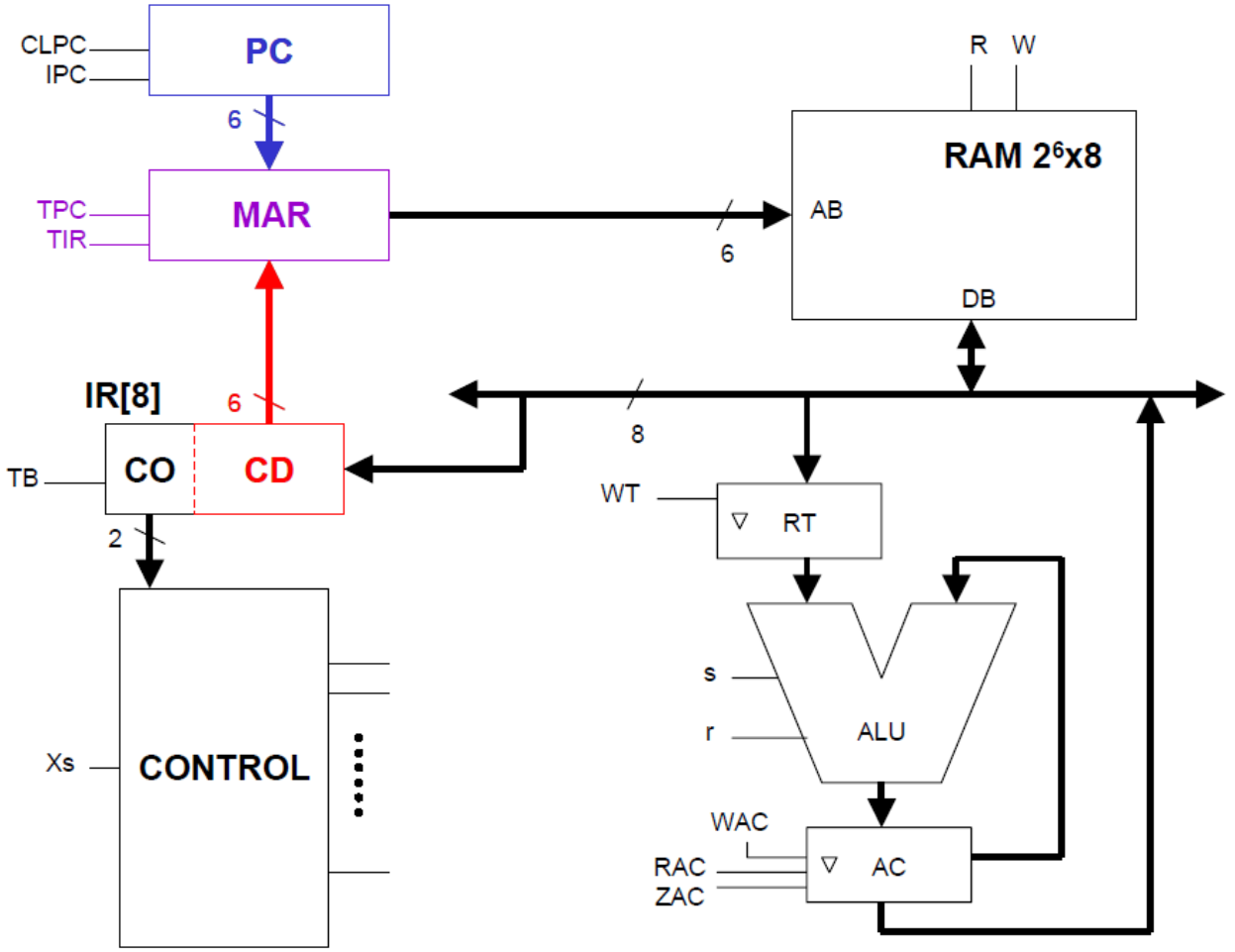
5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

CS1: Unidad de datos (4)

NOTA: Basado en [DIA09]

El bus de direcciones de la RAM debe ser accedido por:

- **El campo CD del registro IR: para poder acceder al operando en memoria**
- **El registro PC: para poder indicar la dirección de la siguiente instrucción**



- **Se incluye el registro de direcciones de memoria (MAR, *Memory Address Register*) para almacenar temporalmente la dirección de la instrucción (TPC) o del dato (TIR)**

5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

CS1: Instrucciones

1. **Fase de búsqueda (*fetch*):** la unidad de control se encarga de emitir las microoperaciones que permiten cargar en IR la instrucción apuntada por PC;
2. **Fase de ejecución (*exec*):** la unidad de control se encarga de emitir las microoperaciones necesarias (entre otras, acceder a los operandos) en función del código de operación de la instrucción.

Las instrucciones del CS1 son:

NOTA: Basado en [DIA09]

CO (IR_{7-6})	Registro IR	Mnemónico	Descripción nivel RT
00	0 0 x x x x x x	STOP	(Fin ejecución)
01	0 1 A_5 A_4 A_3 A_2 A_1 A_0	ADD A_{5-0}	AC ← AC + RAM(A_{5-0})
10	1 0 A_5 A_4 A_3 A_2 A_1 A_0	SUB A_{5-0}	AC ← AC − RAM(A_{5-0})
11	1 1 A_5 A_4 A_3 A_2 A_1 A_0	STA A_{5-0}	RAM(A_{5-0}) ← AC

5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

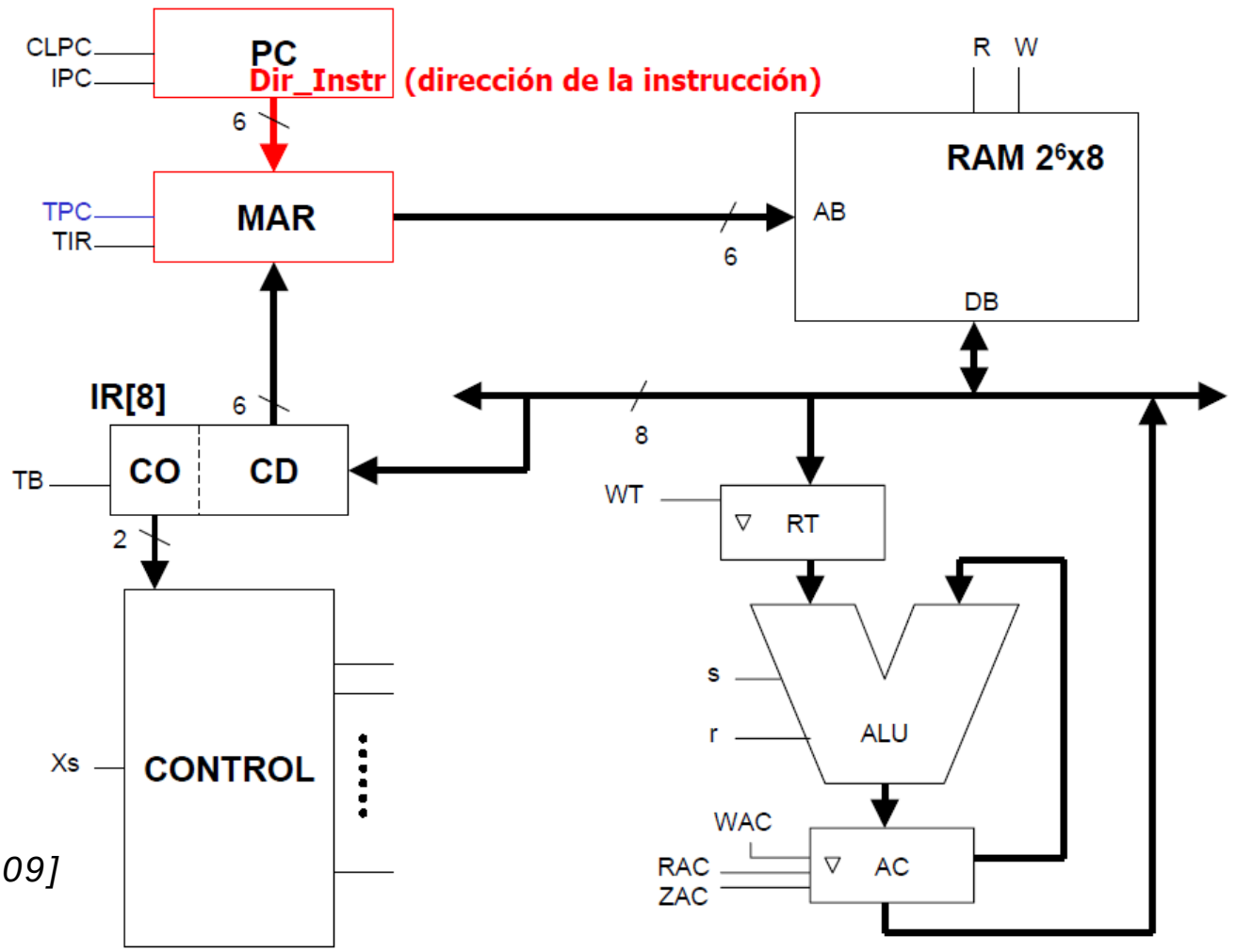
CS1: Instrucciones. Fase de búsqueda (1)

En el CS1, para cargar la siguiente instrucción en IR, necesitamos dos ciclos de reloj:

1. MAR ← PC

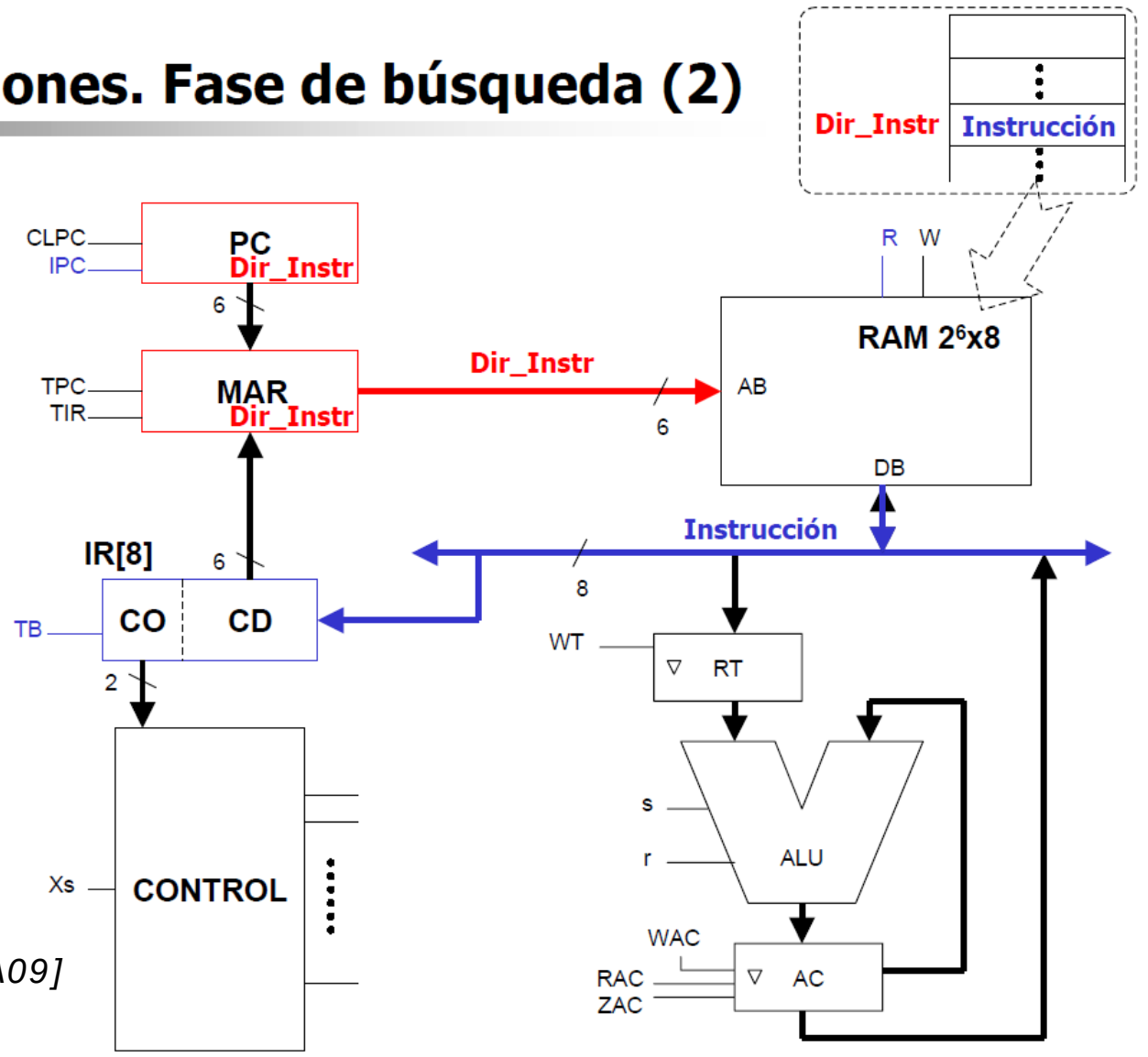
2. IR ← RAM;
   PC ← PC + 1

NOTA: Basado en [DIA09]



5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

CS1: Instrucciones. Fase de búsqueda (2)



1. MAR ← PC

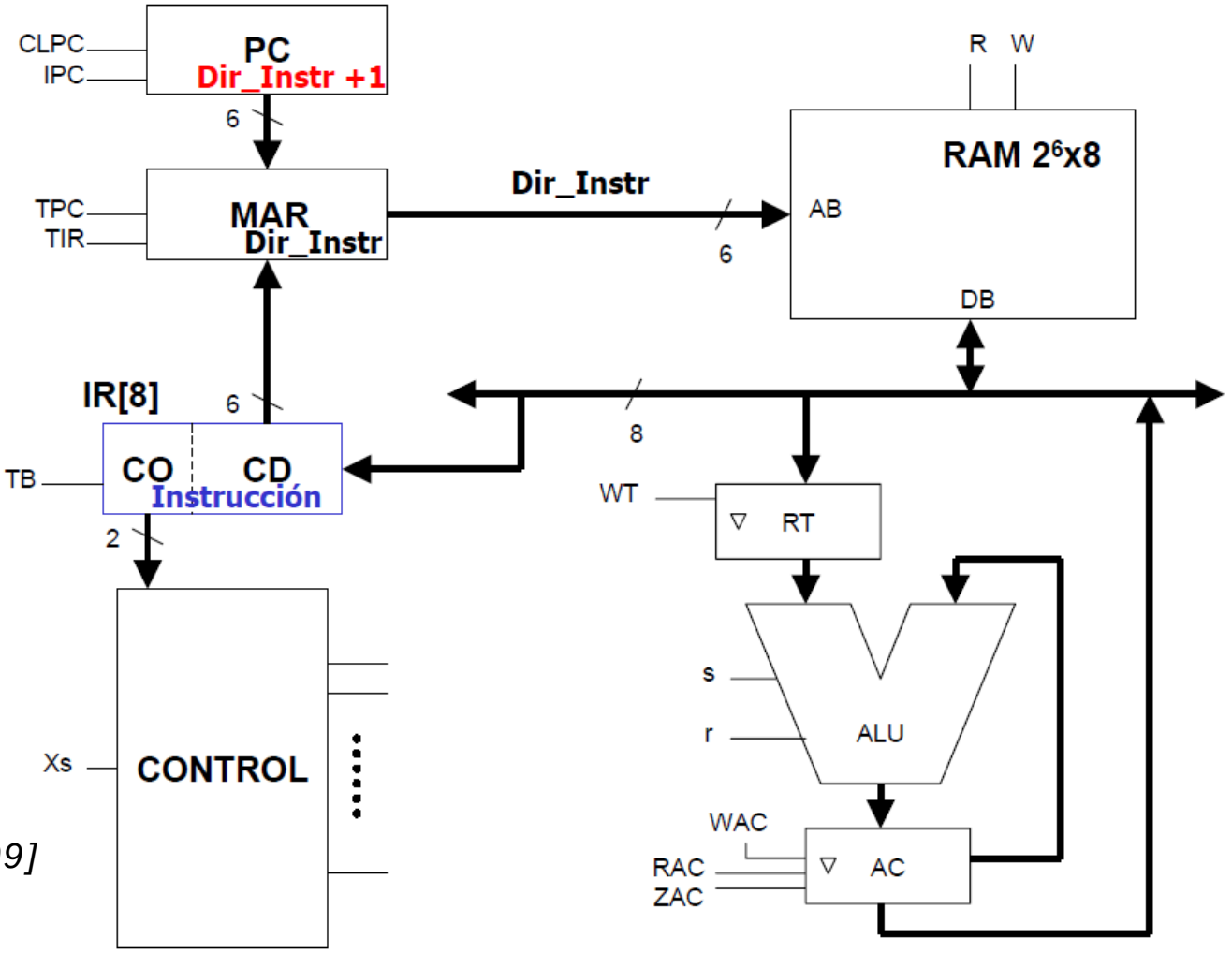
2. IR ← RAM;
   PC ← PC + 1

NOTA: Basado en [DIA09]

CS1: Instrucciones. Fase de búsqueda (fin)

Tras la fase de búsqueda:

- **la instrucción a ejecutar está almacenada en IR**
- **PC contiene la dirección de la siguiente instrucción**



NOTA: Basado en [DIA09]

5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

CS1: Instrucciones. Fase de ejecución [ADD] (1)

NOTA: Basado en [DIA09]

La fase de ejecución comienza donde terminó la fase de búsqueda.

ADD:

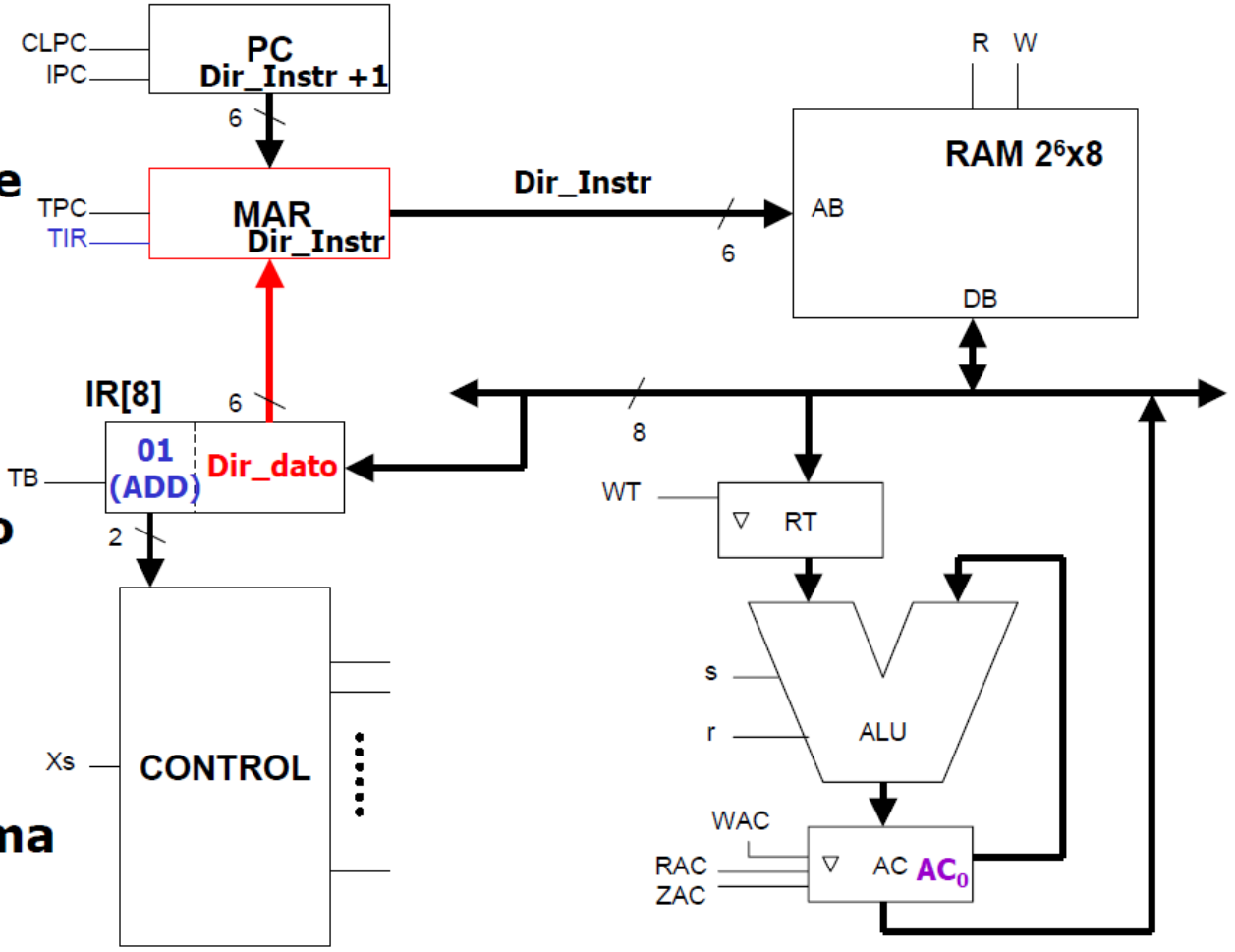
AC ← AC + RAM(Dir_dato)

1º Extraer el dato (2 clk):

1. MAR ← IR
2. RT ← RAM

2º Realizar la suma (1 clk):

3. AC ← AC + RT

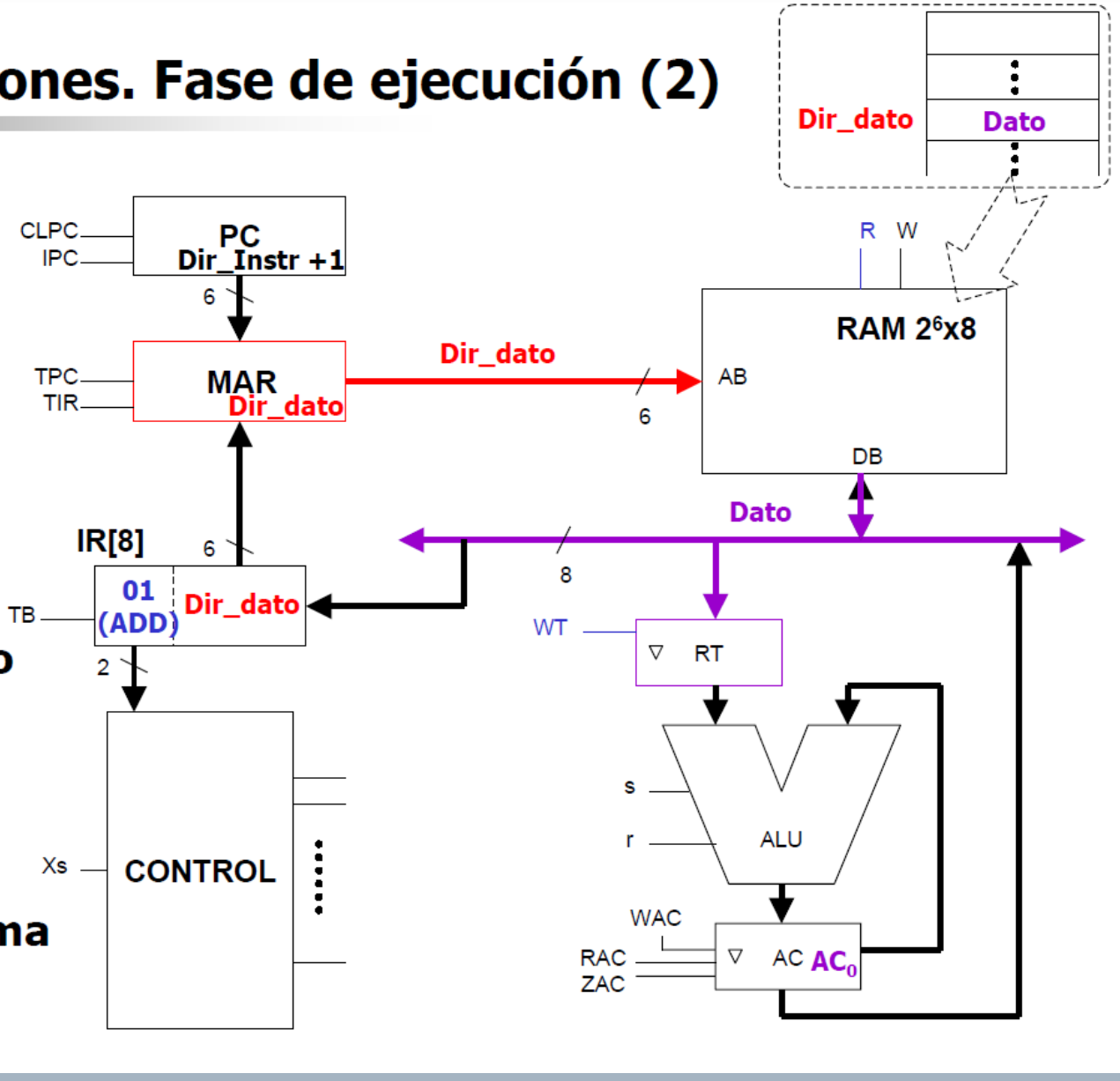


Nota: antes de comenzar, AC = AC_0

5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

CS1: Instrucciones. Fase de ejecución (2)

NOTA: Basado en [DIA09]



ADD:

AC ← AC + RAM(Dir_dato)

1º Extraer el dato (2 clk):

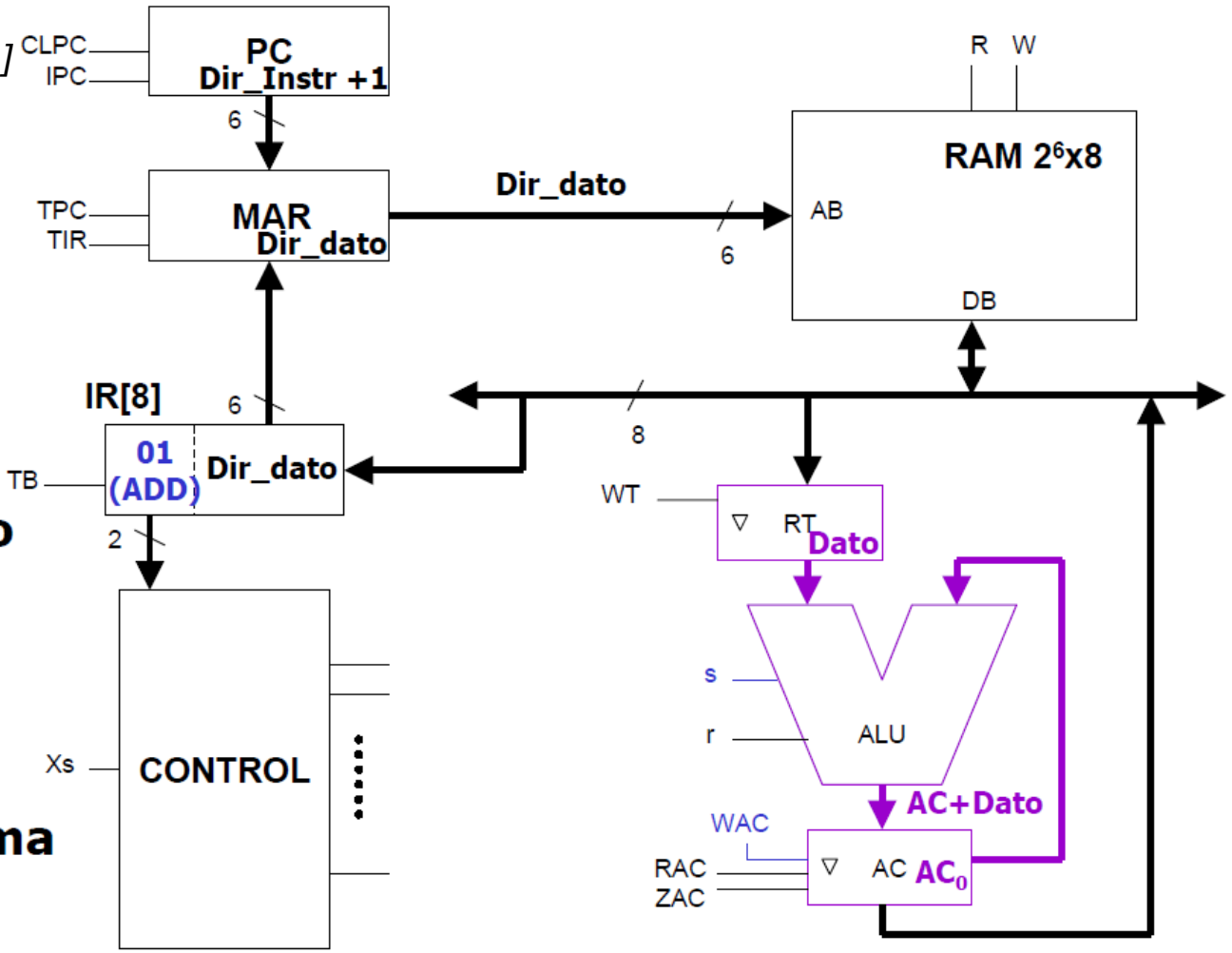
1. MAR ← IR
2. RT ← RAM

2º Realizar la suma (1 clk):

3. AC ← AC + RT

CS1: Instrucciones. Fase de ejecución (3)

NOTA: Basado en [DIA09]



ADD:

AC ← AC + RAM(Dir_dato)

1º Extraer el dato (2 clk):

1. MAR ← IR
2. RT ← RAM

2º Realizar la suma (1 clk):

3. AC ← AC + RT

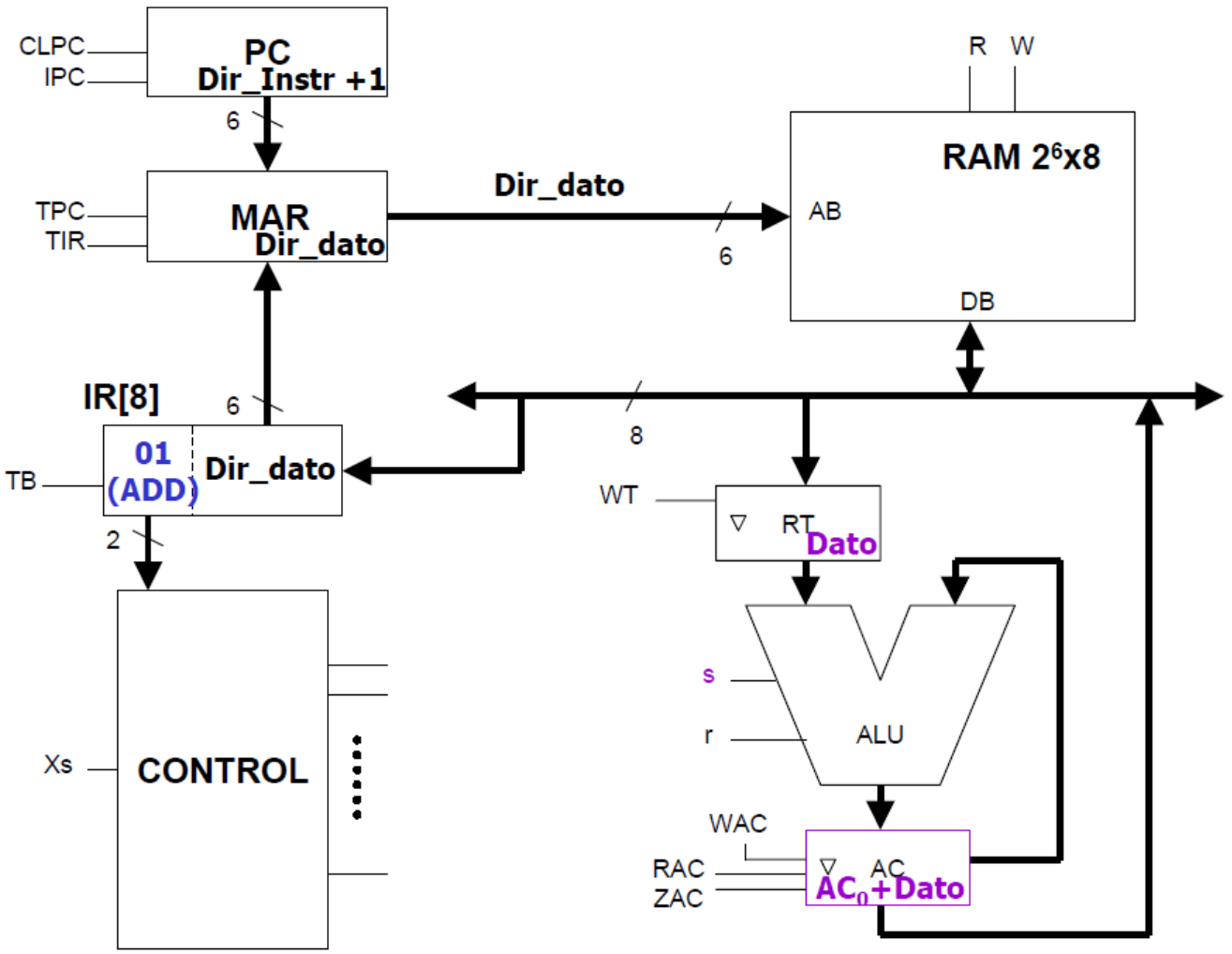
5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

CS1: Instrucciones. Fase de ejecución (fin)

NOTA: Basado en [DIA09]

Al terminar la fase de ejecución de ADD, el registro AC queda actualizado con el resultado de la suma.

El sistema está listo para buscar y ejecutar la siguiente instrucción (apuntada por PC).



5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

CS1: MICROOPERACIONES

Búsqueda	1	MAR ← PC (TPC)
	2	IR ← RAM; PC ← PC + 1 (R, TB, IPC)

NOTA: Basado en [DIA09]

		STOP $IR_{7,6}$=00	ADD A $IR_{7,6}$=01	SUB A $IR_{7,6}$=10	STA A $IR_{7,6}$=11
Ejecución	3	NOP	MAR ← IR (TIR)		
	4	-	RT ← RAM (R, WT)		RAM ← AC (RAC, W)
	5	-	AC ← AC + RT (s, WAC)	AC ← AC - RT (r, WAC)	-

5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

Microoperaciones RT.

- En la tabla se resumen microoperaciones de transferencia que la UP puede satisfacer, y que son suficientes para realizar el conjunto de las cuatro instrucciones del computador simple.
- En la fila (*) se muestra una transferencia múltiple de dos transferencias simples que pueden realizarse en paralelo en un mismo ciclo máquina.
- En color amarillo se indican las señales de control a activar al valor “1” para ejecutar cada microoperación

Microoperaciones RT	**Señales de control**											
	PC		**MAR**		**IR**	**RT**	**AC**		ALU		**Memoria Principal**	
	CLPC	IPC	TIR	TPC	TB	WT	WAC	RAC	sumar	restar	R	W
PC=0	**1**	**0**	0	0	0	0	0	0	-	-	0	0
(*) PC←PC+1 IR←M[MAR]	**0**	**1**	0	0	**1**	0	0	0	-	-	**1**	0
MAR←PC	0	0	0	**1**	0	0	0	0	-	-	0	0
MAR←IR$_{5-0}$	0	0	**1**	0	0	0	0	0	-	-	0	0
RT←M[MAR]	0	0	0	0	0	**1**	0	0	-	-	**1**	0
M[MAR]←AC	0	0	0	0	0	0	0	**1**	-	-	0	**1**
AC←AC+RT	0	0	0	0	0	0	**1**	0	**1**	0	0	0
AC←AC-RT	0	0	0	0	0	0	**1**	0	0	**1**	0	0

5.4.5 Diseño detallado de CS1. a) Diseño de la UP.

CS1: MICROOPERACIONES

Búsqueda	1	MAR ← PC (TPC)
	2	IR ← RAM; PC ← PC + 1 (R, TB, IPC)

NOTA: Basado en [DIA09]

		STOP $IR_{7,6}$=00	ADD A $IR_{7,6}$=01	SUB A $IR_{7,6}$=10	STA A $IR_{7,6}$=11
Ejecución	3	NOP	MAR ← IR (TIR)		
	4	-	RT ← RAM (R, WT)		RAM ← AC (RAC, W)
	5	-	AC ← AC + RT (s, WAC)	AC ← AC - RT (r, WAC)	-

Microoperaciones RT	Señales de control											
	PC		MAR		IR	RT	AC		ALU		Memoria Principal	
	CLPC	IPC	TIR	TPC	TB	WT	WAC	RAC	sumar	restar	R	W
PC=0	**1**	**0**	0	0	0	0	0	0	-	-	0	0
(*) PC←PC+1 IR←M[MAR]	**0**	**1**	0	0	**1**	0	0	0	-	-	**1**	0
MAR←PC	0	0	0	**1**	0	0	0	0	-	-	0	0
MAR←IR_{5-0}	0	0	**1**	0	0	0	0	0	-	-	0	0
RT←M[MAR]	0	0	0	0	0	**1**	0	0	-	-	**1**	0
M[MAR]←AC	0	0	0	0	0	0	0	**1**	-	-	0	**1**
AC←AC+RT	0	0	0	0	0	0	**1**	0	**1**	0	0	0
AC←AC-RT	0	0	0	0	0	0	**1**	0	0	**1**	0	0

5.4.5 Diseño detallado de CS1. b) Diseño de la UC.

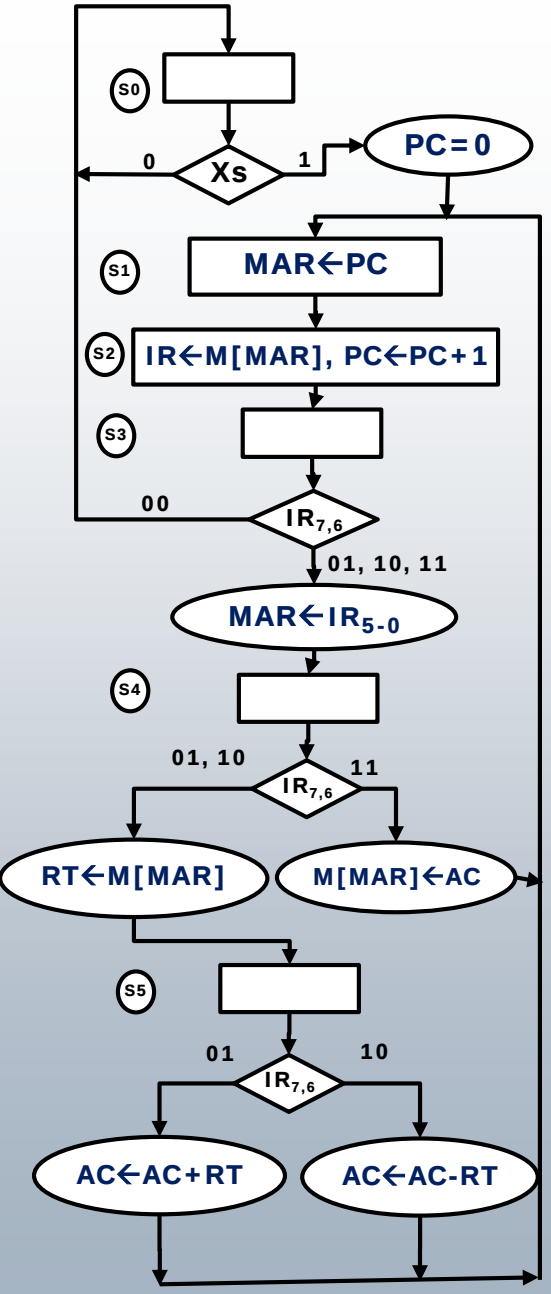
- ***Primer paso:***

  A partir de las “palabras de control” de las microoperaciones de la UP, se deducen las señales de control que han de “activarse” a valor 1, al objeto de ***trasladar la carta ASM de procesado a la carta ASM de control.***

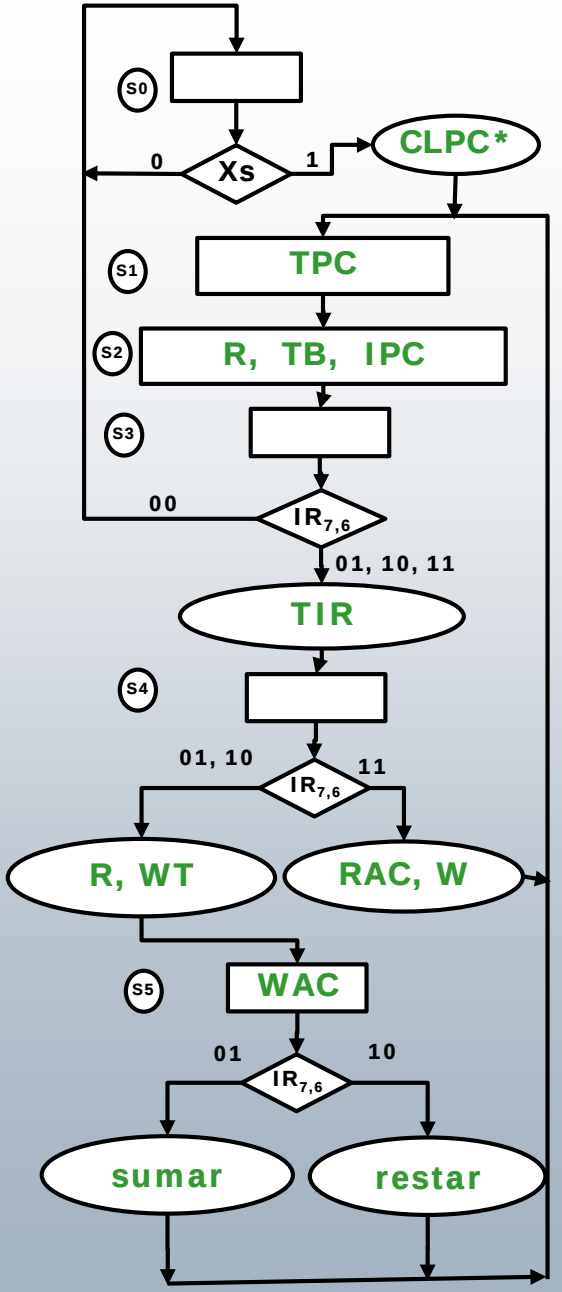
- ***Segundo paso:***

  A partir de la Carta ASM de Control, diseñaremos la unidad de Control. Existen numerosas alternativas. En este tema, por cuestiones didácticas nos centramos en la técnica “ de un biestable por estado” (o “uno caliente (Hot one)”) vista en apartados anteriores del Tema 5.

Desde Carta ASM de Procesado se pasa a Carta ASM de Control.



Micro-operaciones RT	Señales de control											
	PC		MAR		IR	RT	AC		ALU		Memoria Principal	
	CLPC	IPC	TIR	TPC	TB	WT	WAC	RAC	suma	restar	R	W
PC=0	**1**	**0**	0	0	0	0	0	0	-	-	0	0
PC←PC+1 IR←M[MAR]	**0**	**1**	0	0	**1**	0	0	0	-	-	**1**	0
MAR←PC	0	0	0	**1**	0	0	0	0	-	-	0	0
MAR←IR5-0	0	0	**1**	0	0	0	0	0	-	-	0	0
RT←M[MAR]	0	0	0	0	0	**1**	0	0	-	-	**1**	0
M[MAR]←AC	0	0	0	0	0	0	0	**1**	-	-	0	**1**
AC←AC+RT	0	0	0	0	0	0	**1**	0	**1**	0	0	0
AC←AC-RT	0	0	0	0	0	0	**1**	0	0	**1**	0	0



5.4.5 Diseño detallado de CS1. b) Diseño de la UC.

- ***Primer paso:***

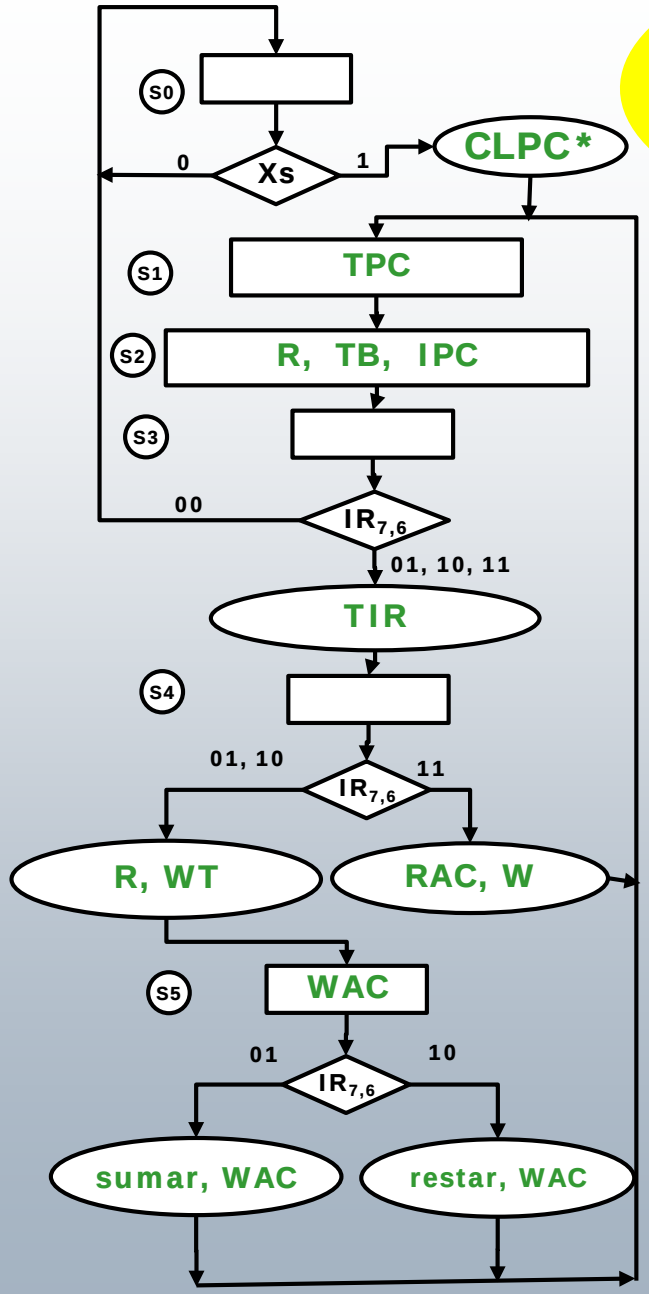
  A partir de las “palabras de control” de las microoperaciones de la UP, se deducen las señales de control que han de “activarse” a valor 1, al objeto de *trasladar la carta ASM de procesado a la carta ASM de control.*

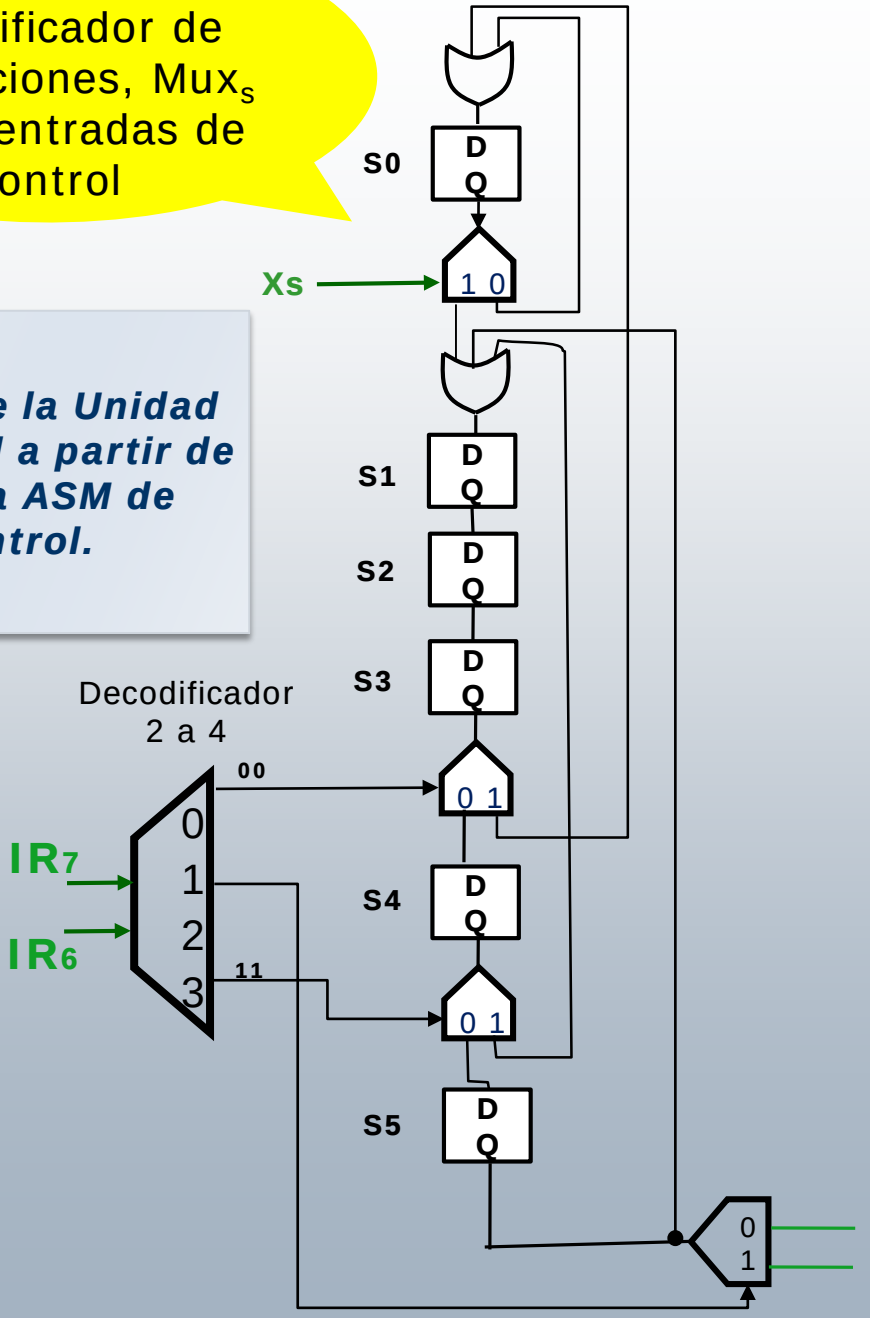
- ***Segundo paso:***

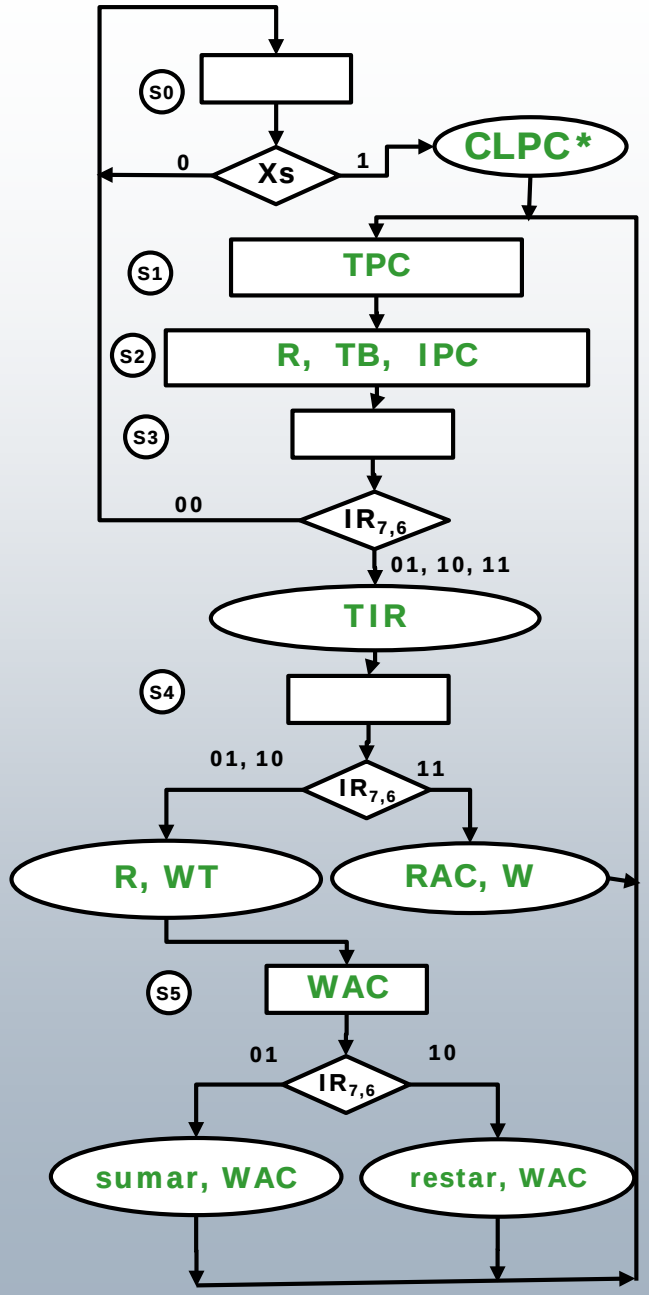
  A partir de la Carta ASM de Control, diseñaremos la unidad de Control. Existen numerosas alternativas. En este tema, por cuestiones didácticas ***nos centramos en la técnica “ de un biestable por estado”*** (o “uno caliente (Hot one)”) vista en apartados anteriores del Tema 5.

Biestables, enlaces, decodificador de instrucciones, Mux$_S$ de las entradas de control

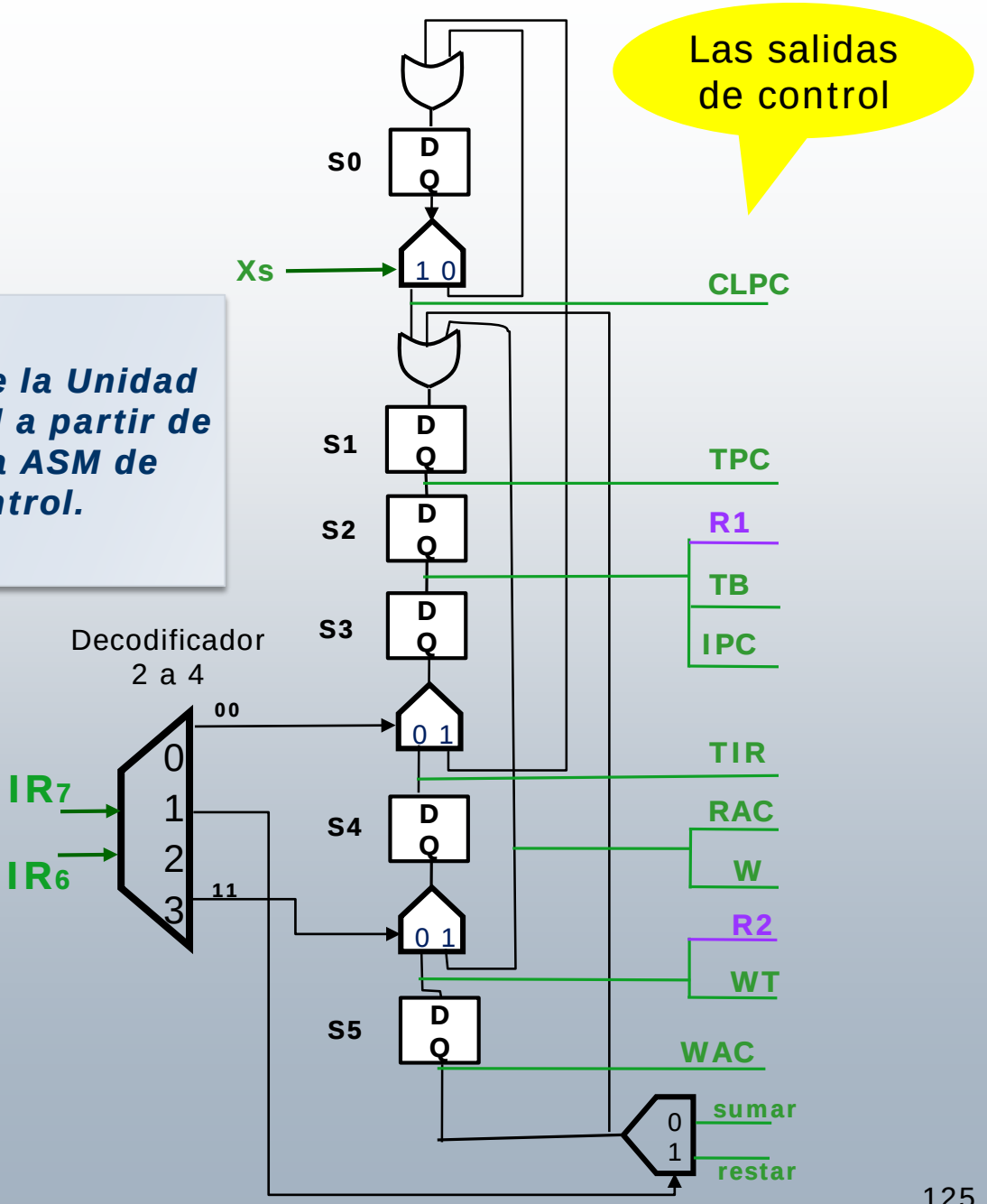
Diseño de la Unidad de Control a partir de la Carta ASM de Control.

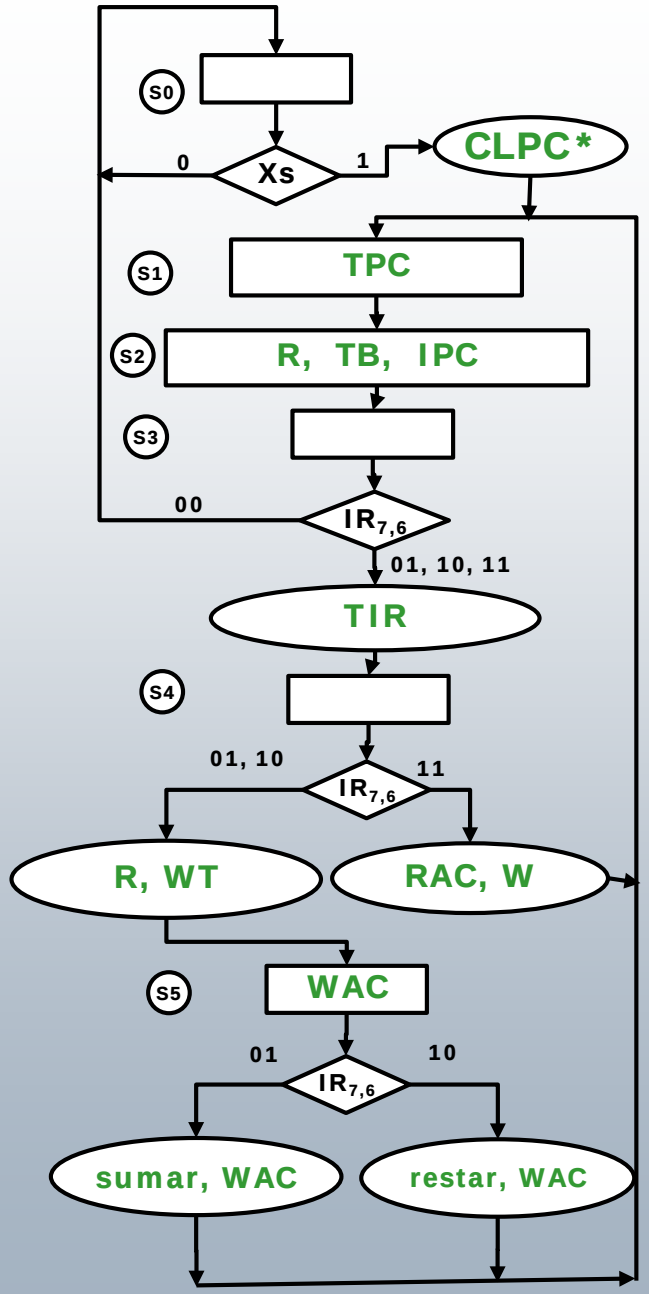




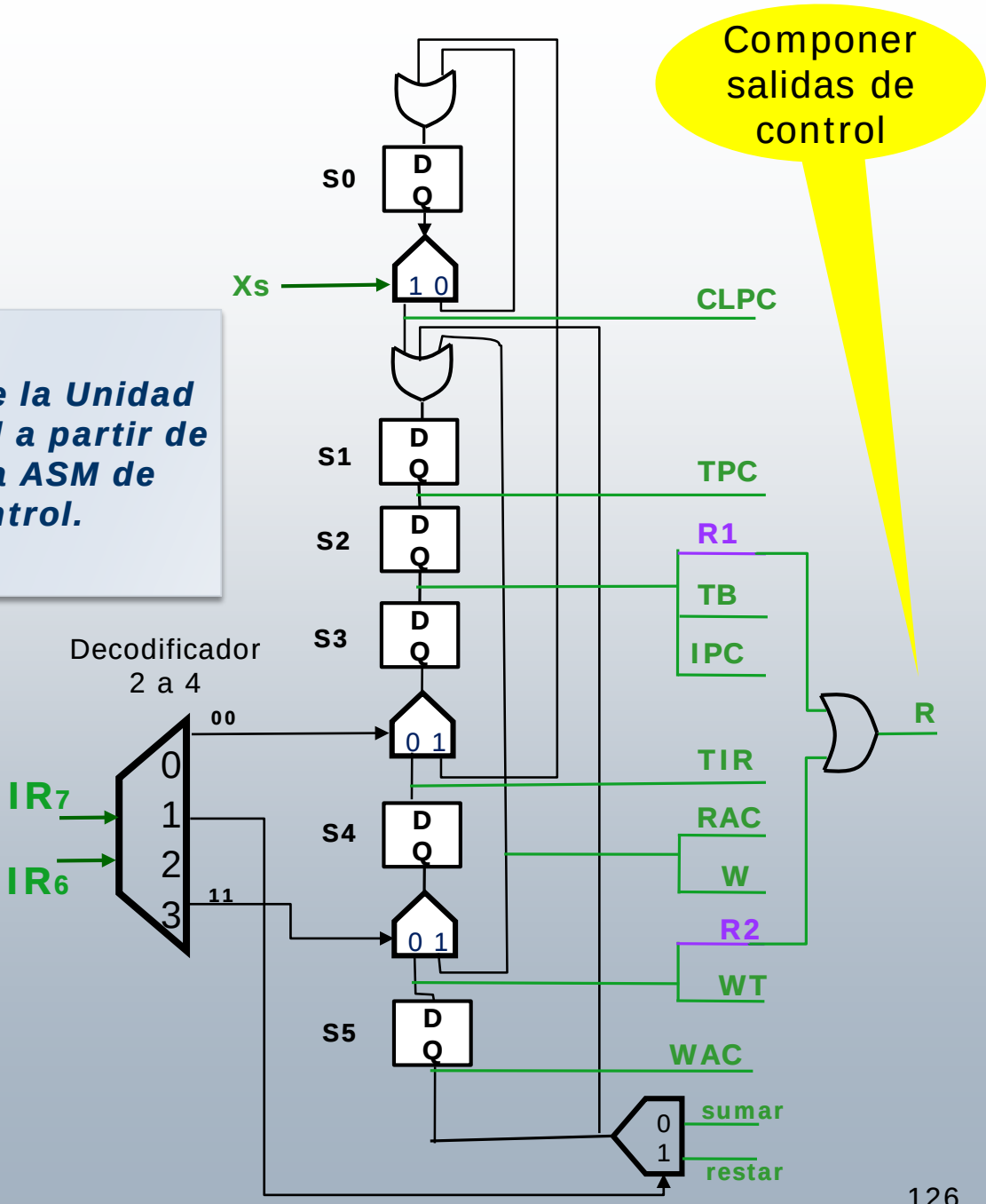


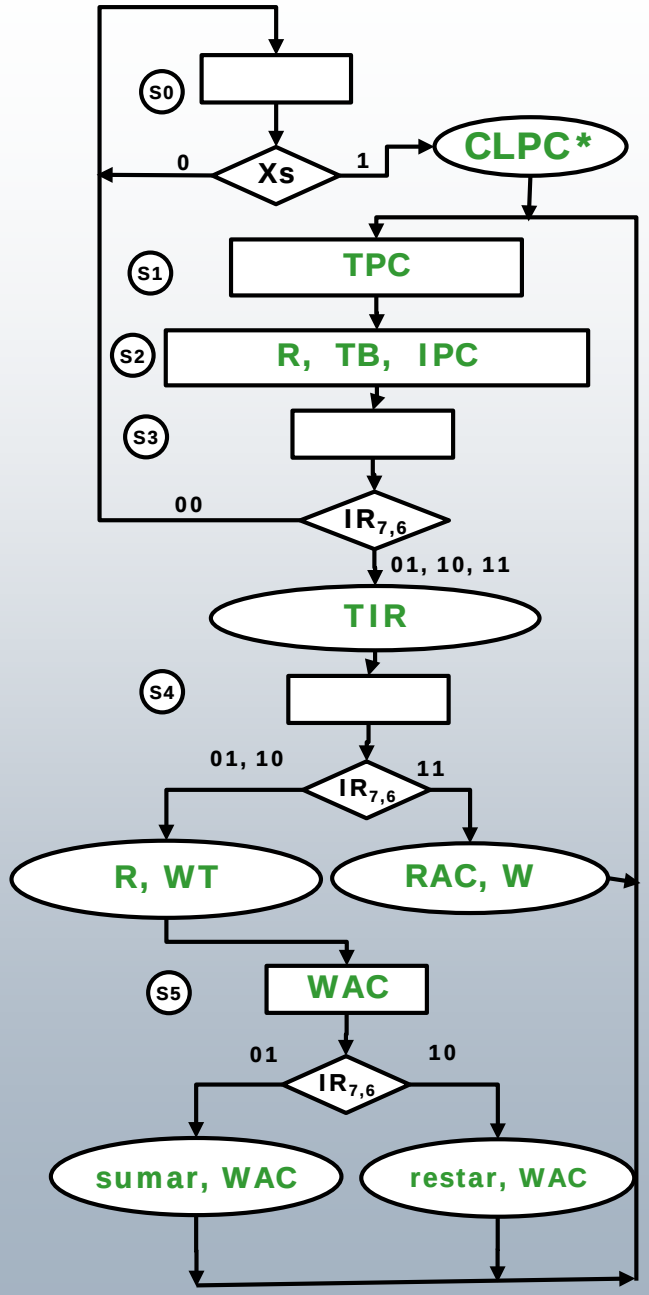
Diseño de la Unidad de Control a partir de la Carta ASM de Control.



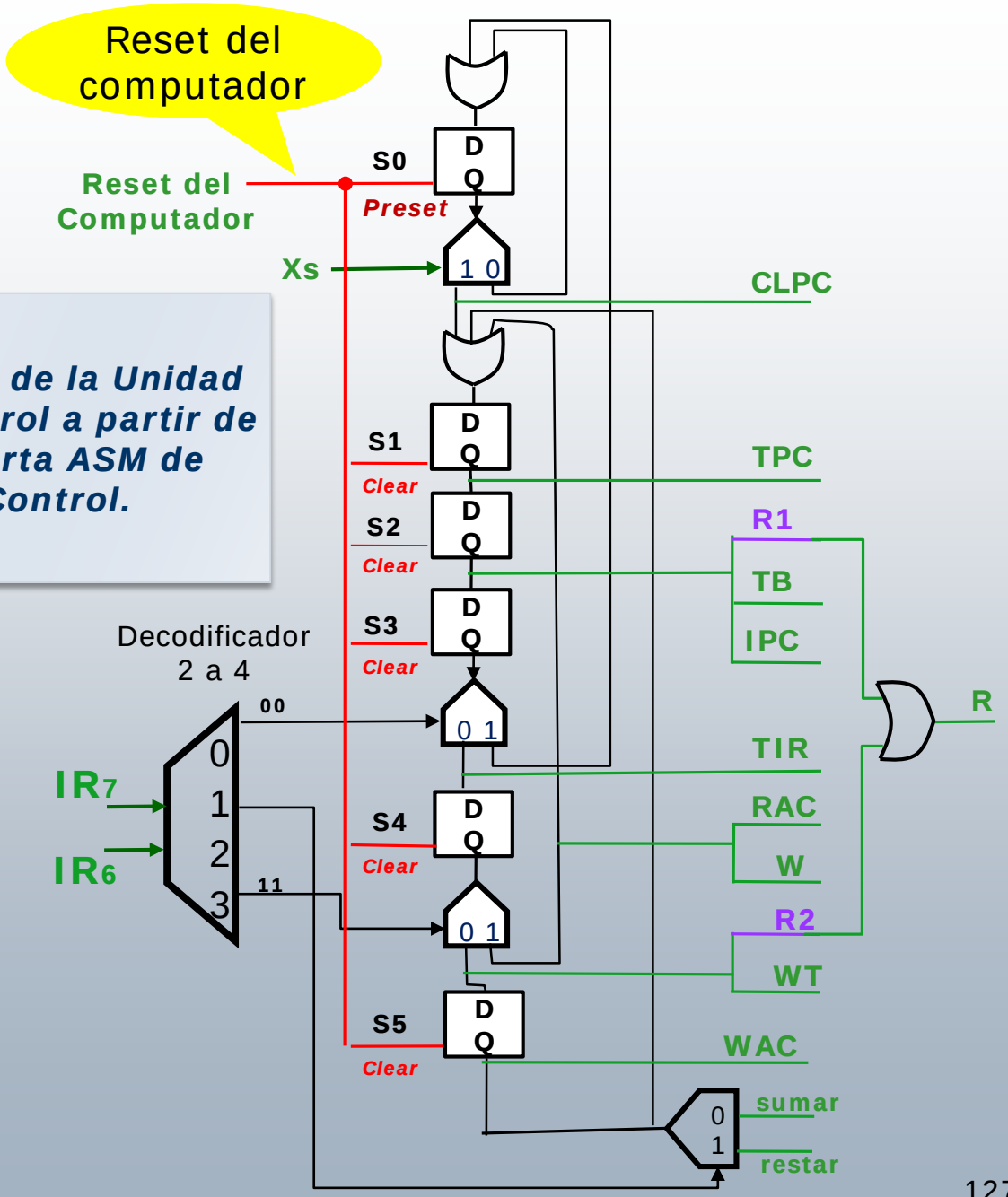


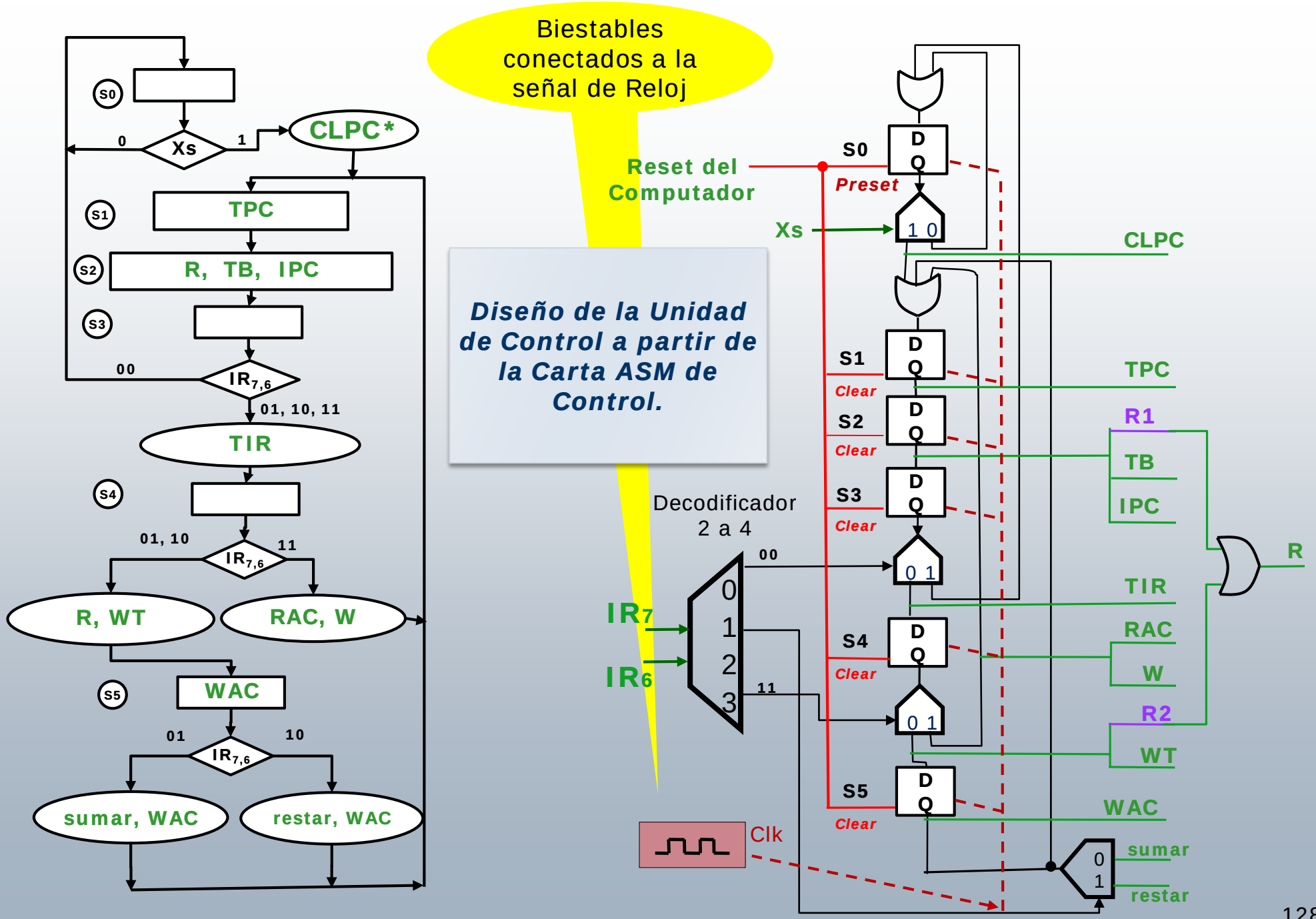
Diseño de la Unidad de Control a partir de la Carta ASM de Control.

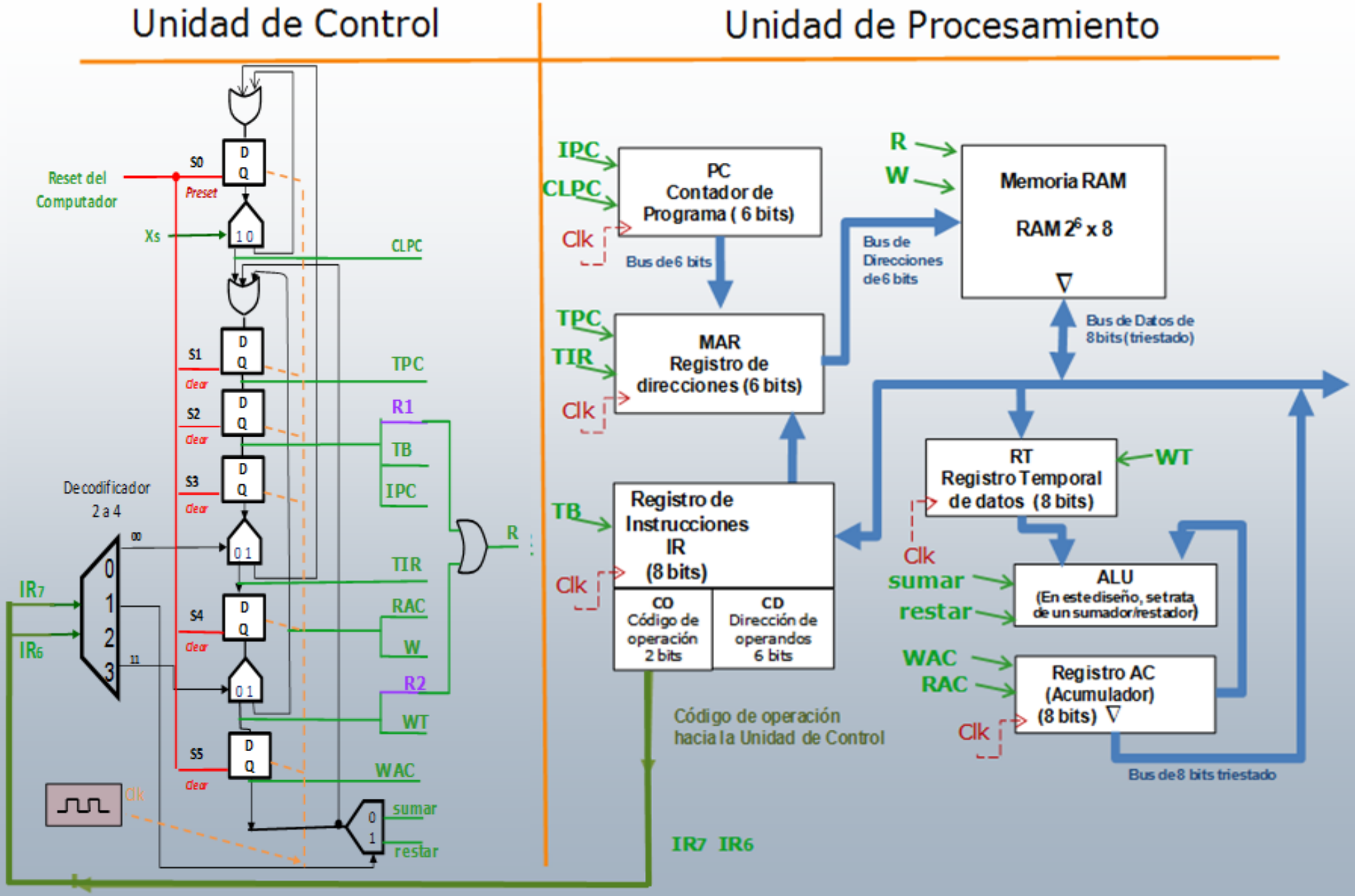




Diseño de la Unidad de Control a partir de la Carta ASM de Control.




Biestables conectados a la señal de Reloj
Diseño de la Unidad de Control a partir de la Carta ASM de Control.
S0
Xs
0
1
CLPC*
S1
TPC
S2
R, TB, IPC
S3
00
IR7,6
01, 10, 11
TIR
S4
01, 10
11
R, WT
RAC, W
S5
WAC
01
10
sumar, WAC
restar, WAC
Reset del Computador
Preset
Clear
CLPC
TPC
R1
TB
IPC
R
TIR
RAC
W
R2
WT
WAC
sumar
restar
Decodificador 2 a 4
IR7
IR6
Clk
D
Q


Unidad de Control
Unidad de Procesamiento
Reset del Computador
S0
Preset
Xs
CLPC
S1
Clear
TPC
S2
Clear
R1
TB
S3
Clear
IPC
Decodificador 2 a 4
00
01
TIR
IR7
IR6
S4
Clear
RAC
W
11
01
R2
WT
S5
Clear
WAC
Clk
sumar
restar
R
IPC
CLPC
Clk
PC Contador de Programa (6 bits)
Bus de6 bits
TPC
TIR
Clk
MAR Registro de direcciones (6 bits)
TB
Clk
Registro de Instrucciones IR (8 bits)
CO Código de operación 2 bits
CD Dirección de operandos 6 bits
Código de operación hacia la Unidad de Control
IR7 IR6
R
W
Memoria RAM
RAM 2^6 x 8
Bus de Direcciones de6 bits
Bus de Datos de 8bits (triestado)
RT Registro Temporal de datos (8 bits)
WT
Clk
sumar
restar
ALU
(En este diseño, se trata de un sumador/restador)
WAC
RAC
Registro AC (Acumulador) (8 bits)
Clk
Bus de8 bits triestado

- Este computador CS1 ha sido realizado en LogicWorks_ Versión 4.1.
- En la práctica 8 ilustraremos su funcionamiento.
- El estudiante podrá realizar programas y cargarlos manualmente en la memoria RAM, para comprobar su ejecución y realizar cronogramas de funcionamiento.
- En la versión estudiante de LogicWorks será posible ver imágenes del diseño de los componentes que utiliza a nivel de puertas lógicas y biestables, invitando al estudiante interesado a su realización y comprobación.

FIN

PREGUNTAS