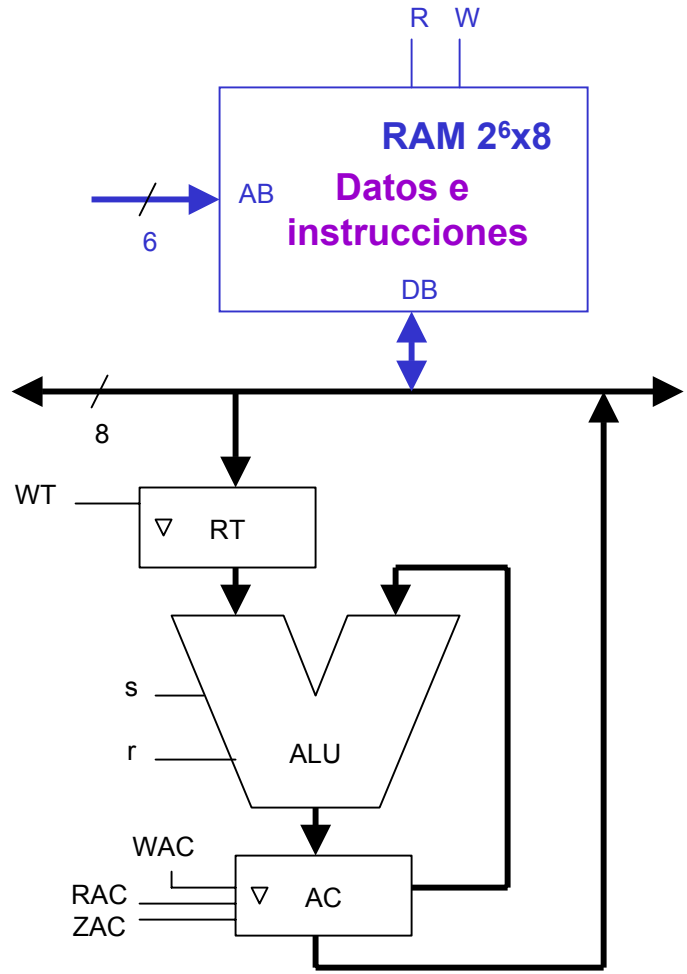
CS1: Unidad de datos

- **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*)**



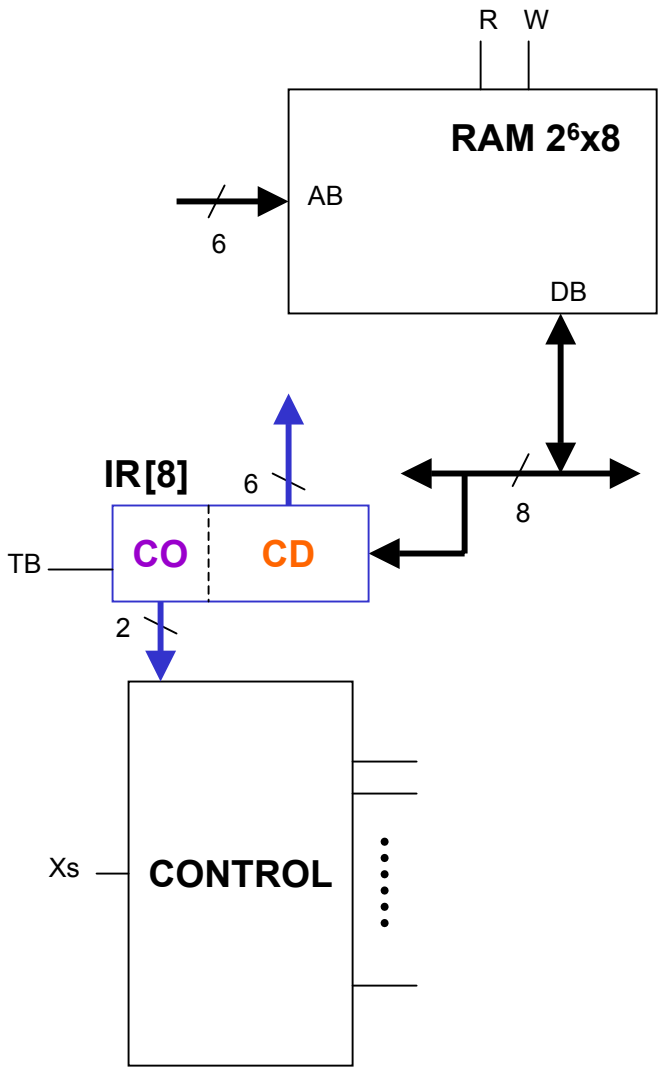
CS1: Unidad de datos (2)

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**
  - **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**

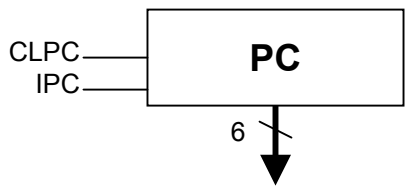
CS1

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



CS1: Unidad de datos (3)

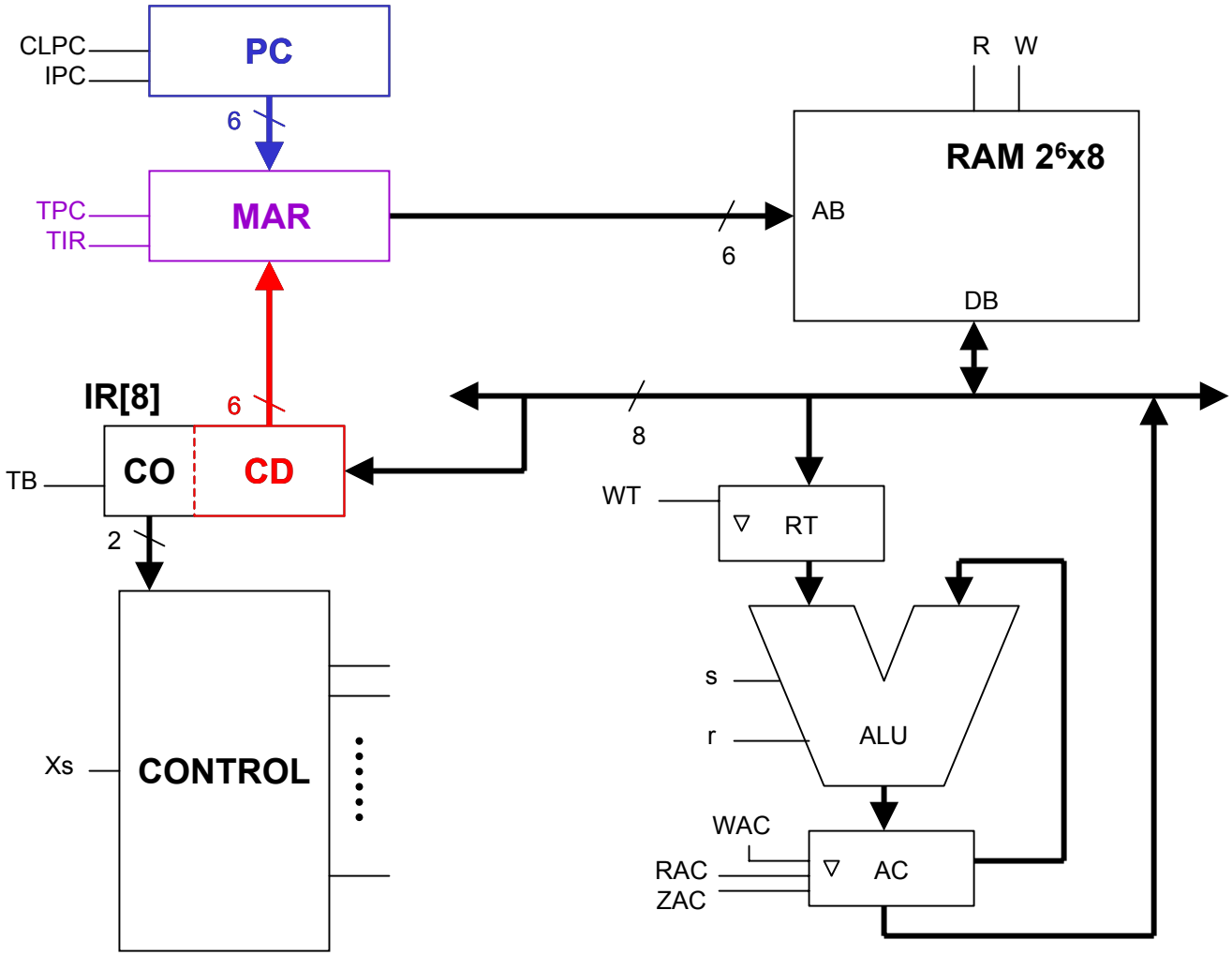
- **Una vez ejecutada la instrucción actual, hay que extraer la siguiente desde memoria y almacenarla en IR**
- **Para ello es necesario un registro puntero de instrucciones (PC, *Program Counter*):**
  - **almacena la dirección de memoria donde se encuentra la siguiente instrucción**
  - **una vez transferida 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**



CS1: Unidad de datos (4)

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

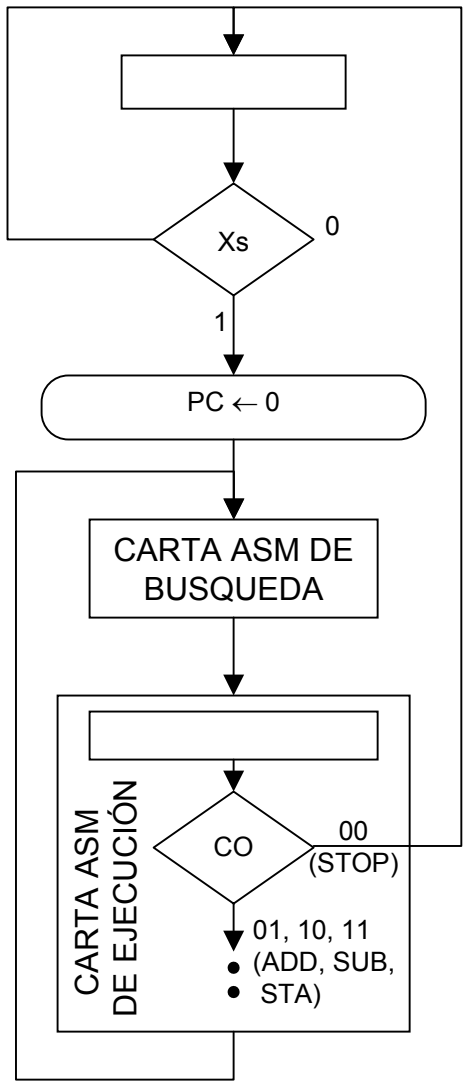
- **El campo CD del registro IR: para poder acceder al operando en memoria**
- **El registro PC: para poder indicar la dirección de la siguiente instrucción**



- **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)**

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:

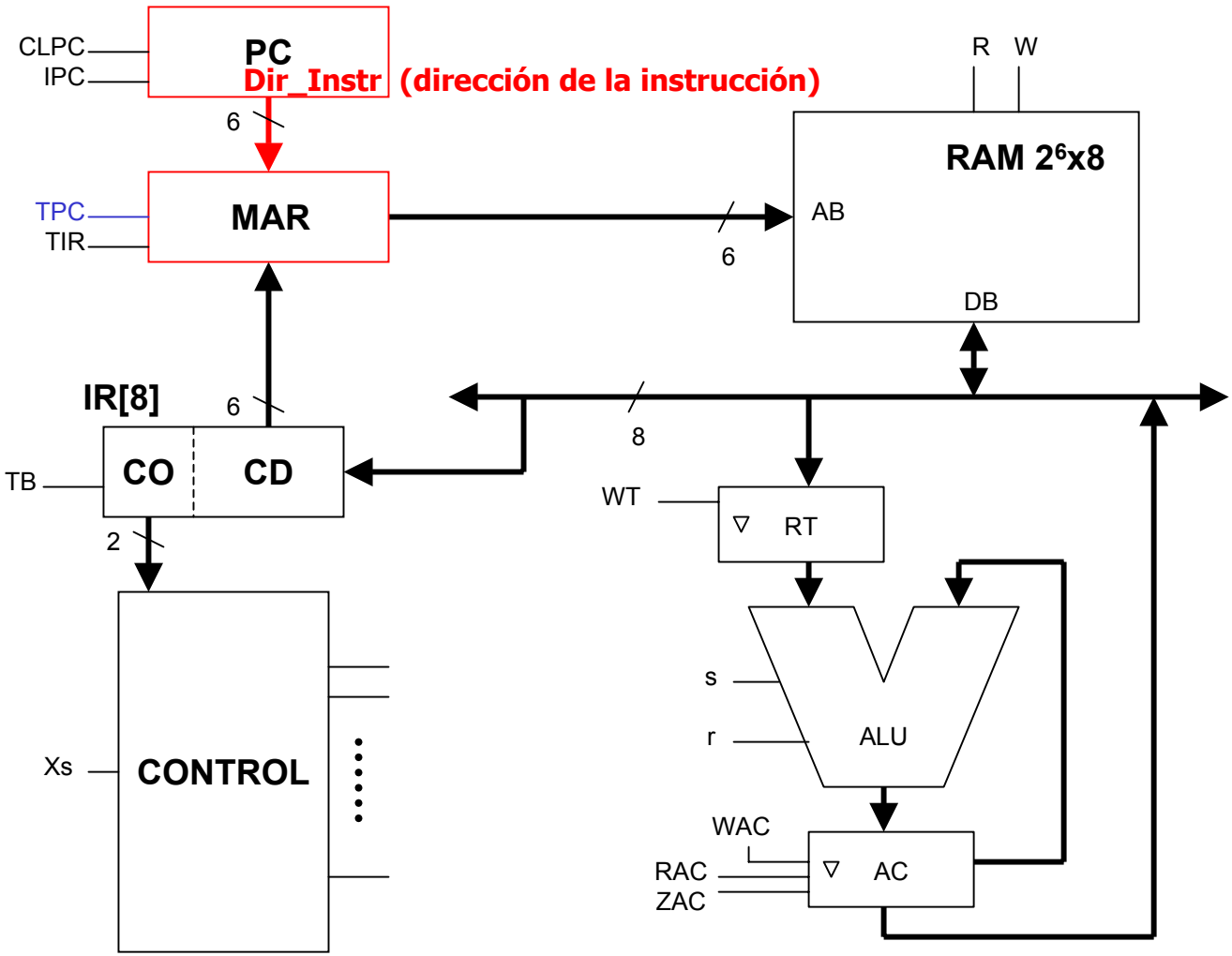
CO (IR_{7-6})	Registro IR	Mnemónico	Descripción nivel RT
00	0 0 x x x x x x	STOP	(Fin ejecución)
01	0 1 A_5 A_4 A_3 A_2 A_1 A_0	ADD A_{5-0}	AC ← AC + RAM(A_{5-0})
10	1 0 A_5 A_4 A_3 A_2 A_1 A_0	SUB A_{5-0}	AC ← AC - RAM(A_{5-0})
11	1 1 A_5 A_4 A_3 A_2 A_1 A_0	STA A_{5-0}	RAM(A_{5-0}) ← AC

CS1: Instrucciones. Fase de búsqueda (1)

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

1. MAR ← PC

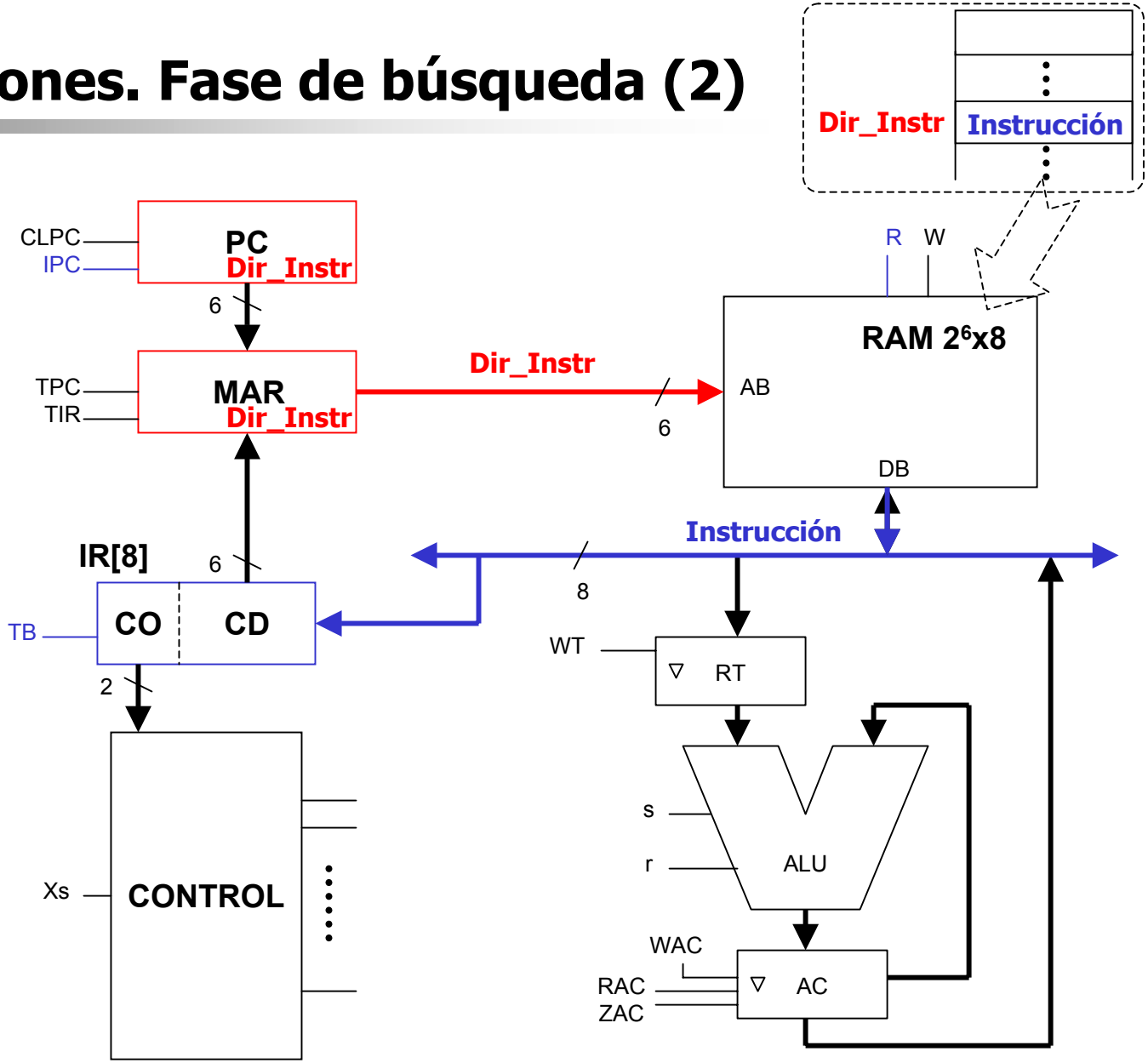
2. IR ← RAM;
   PC ← PC + 1



CS1: Instrucciones. Fase de búsqueda (2)

1. MAR ← PC

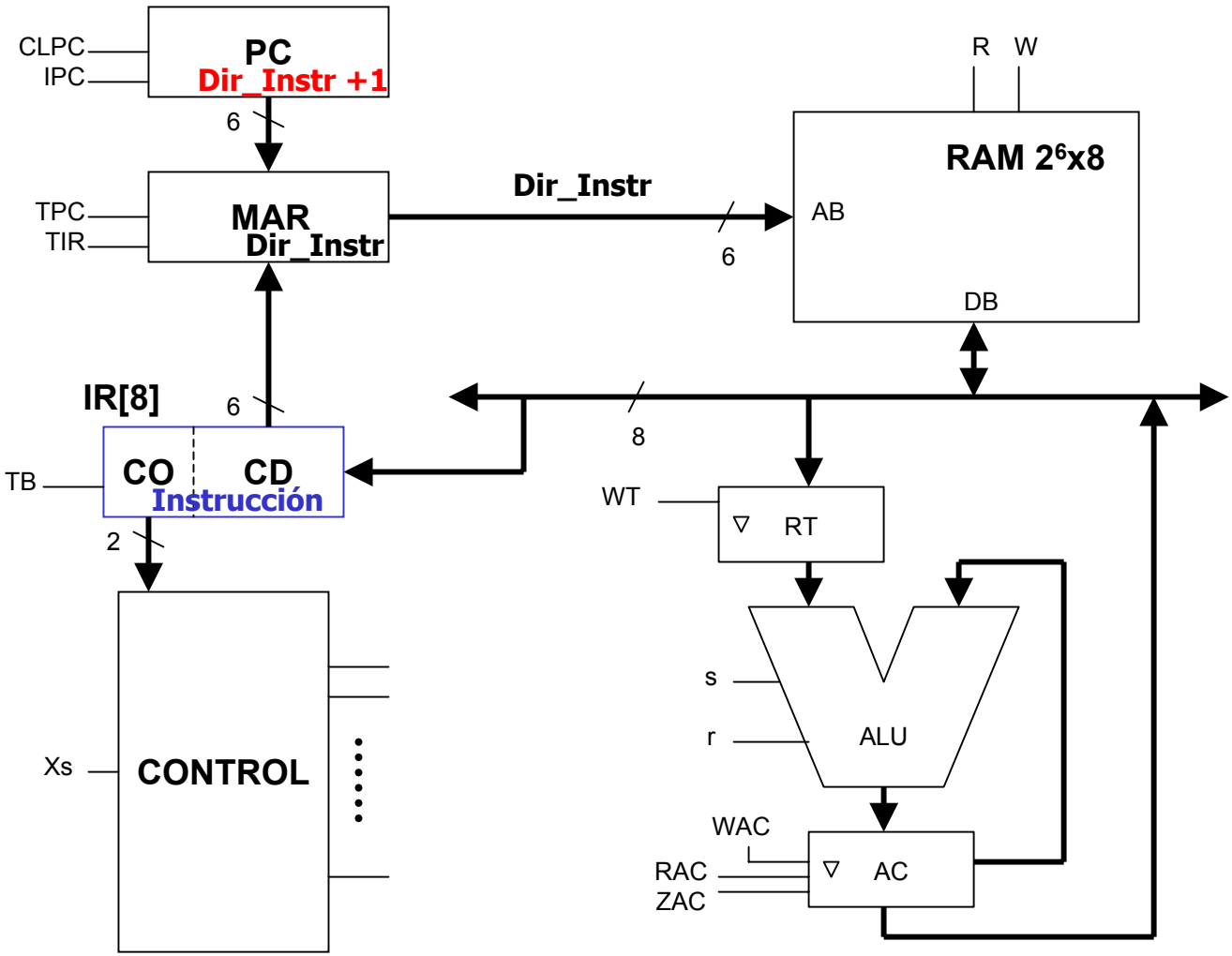
2. IR ← RAM;
   PC ← PC + 1



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**



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

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

ADD:

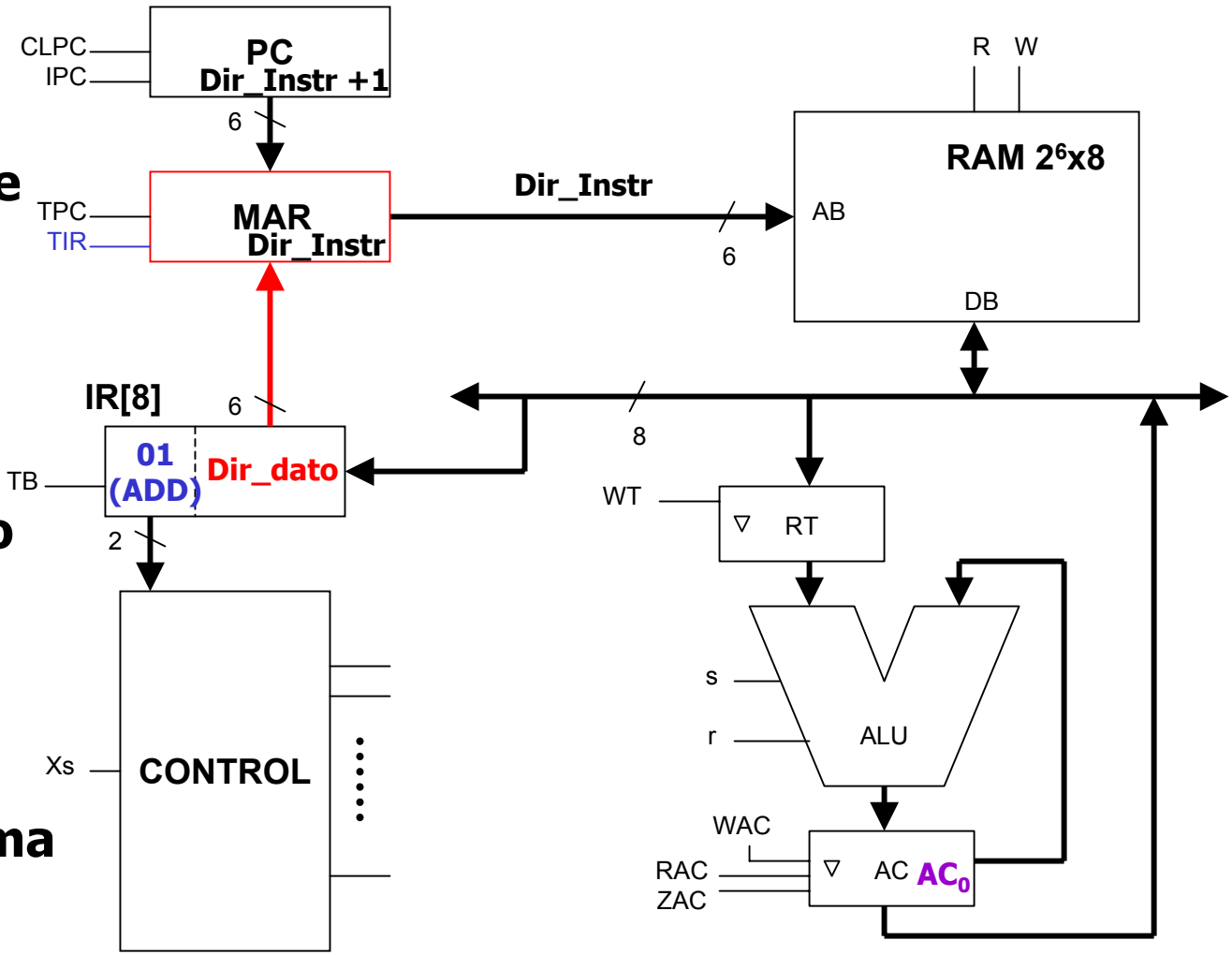
AC ← AC + RAM(Dir_dato)

1º Extraer el dato (2 clk):

1. MAR ← IR
2. RT ← RAM

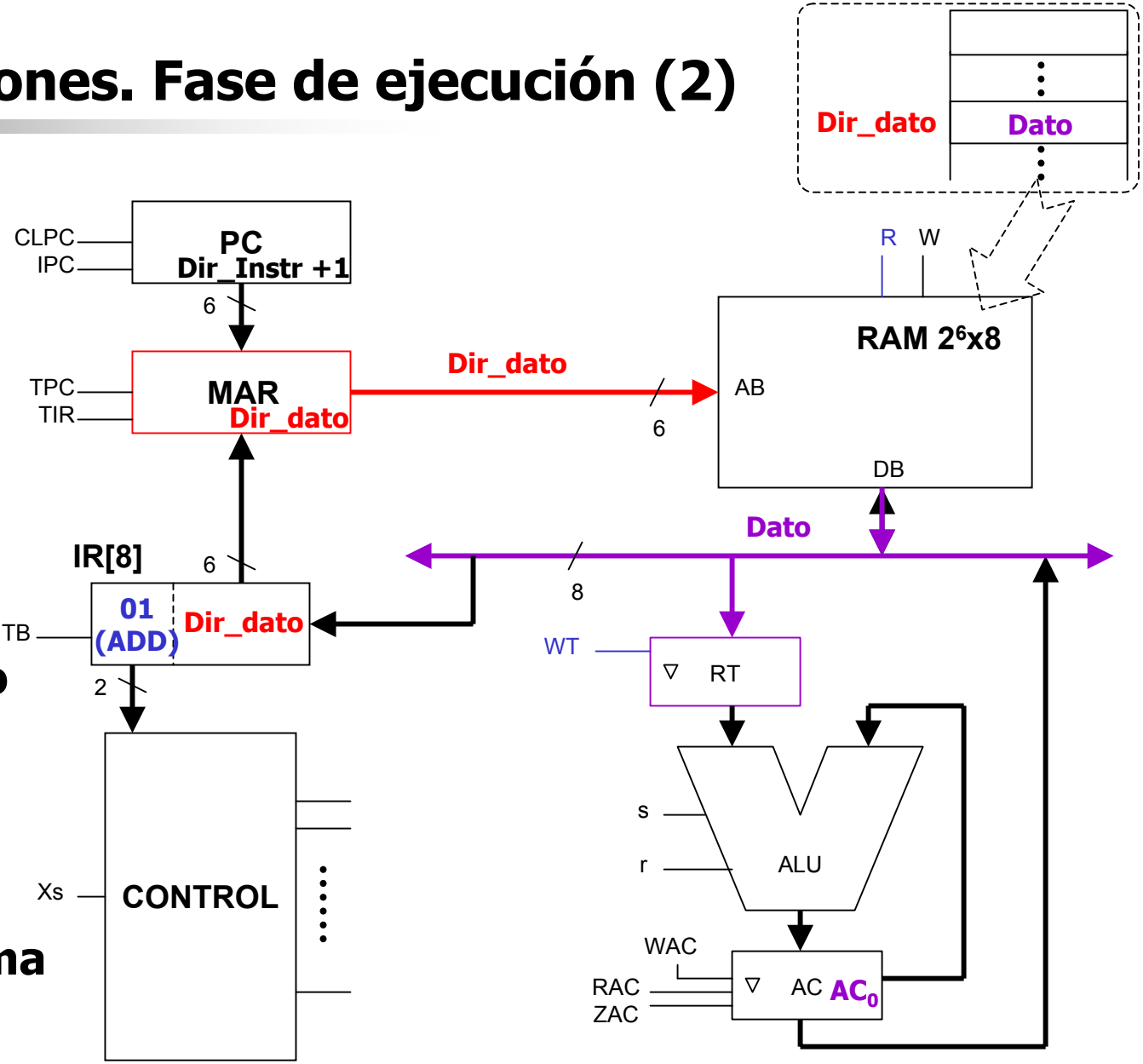
2º Realizar la suma (1 clk):

3. AC ← AC + RT



Nota: antes de comenzar, AC = AC_0

CS1: Instrucciones. Fase de ejecución (2)



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)

ADD:

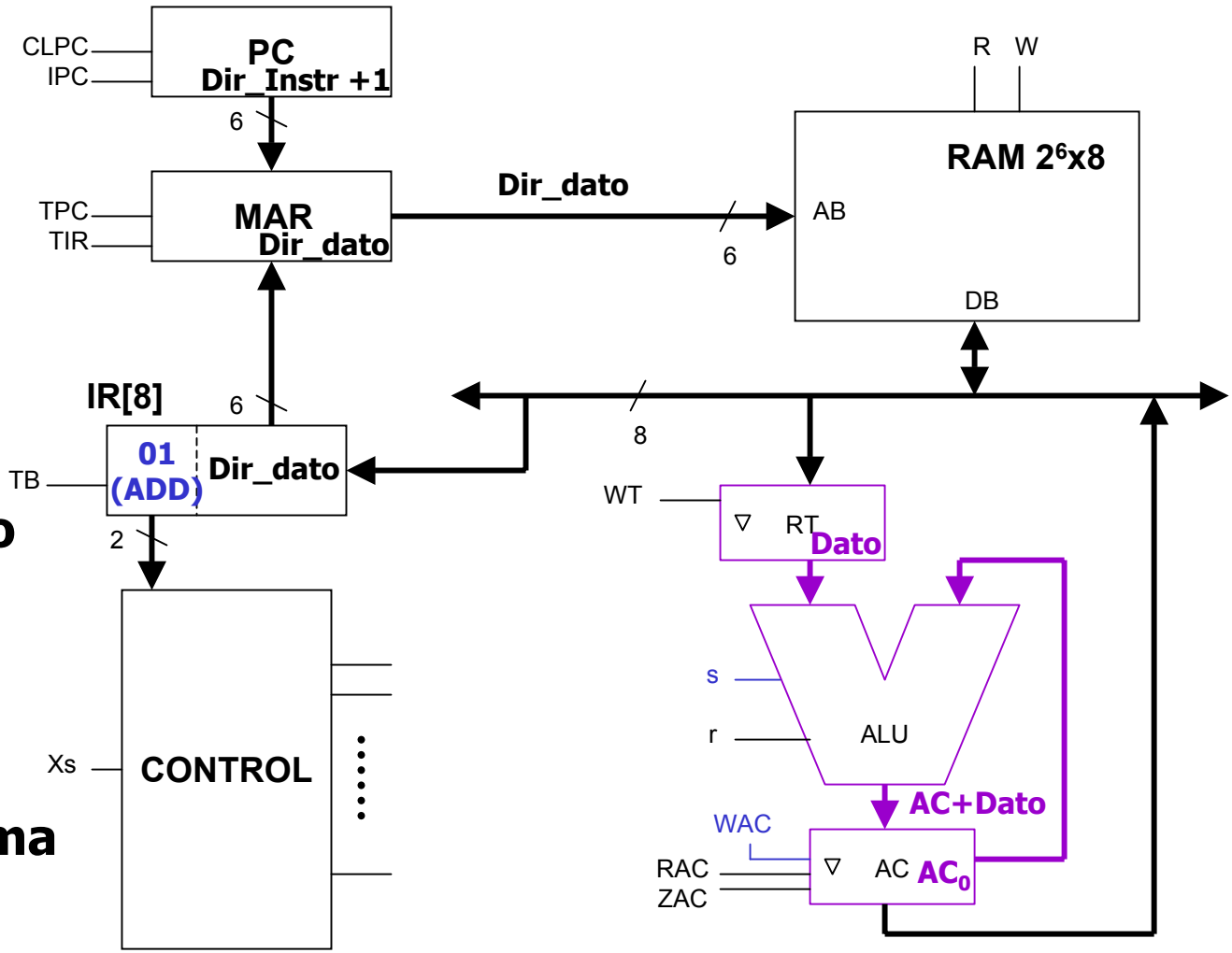
AC ← AC + RAM(Dir_dato)

1º Extraer el dato (2 clk):

1. MAR ← IR
2. RT ← RAM

2º Realizar la suma (1 clk):

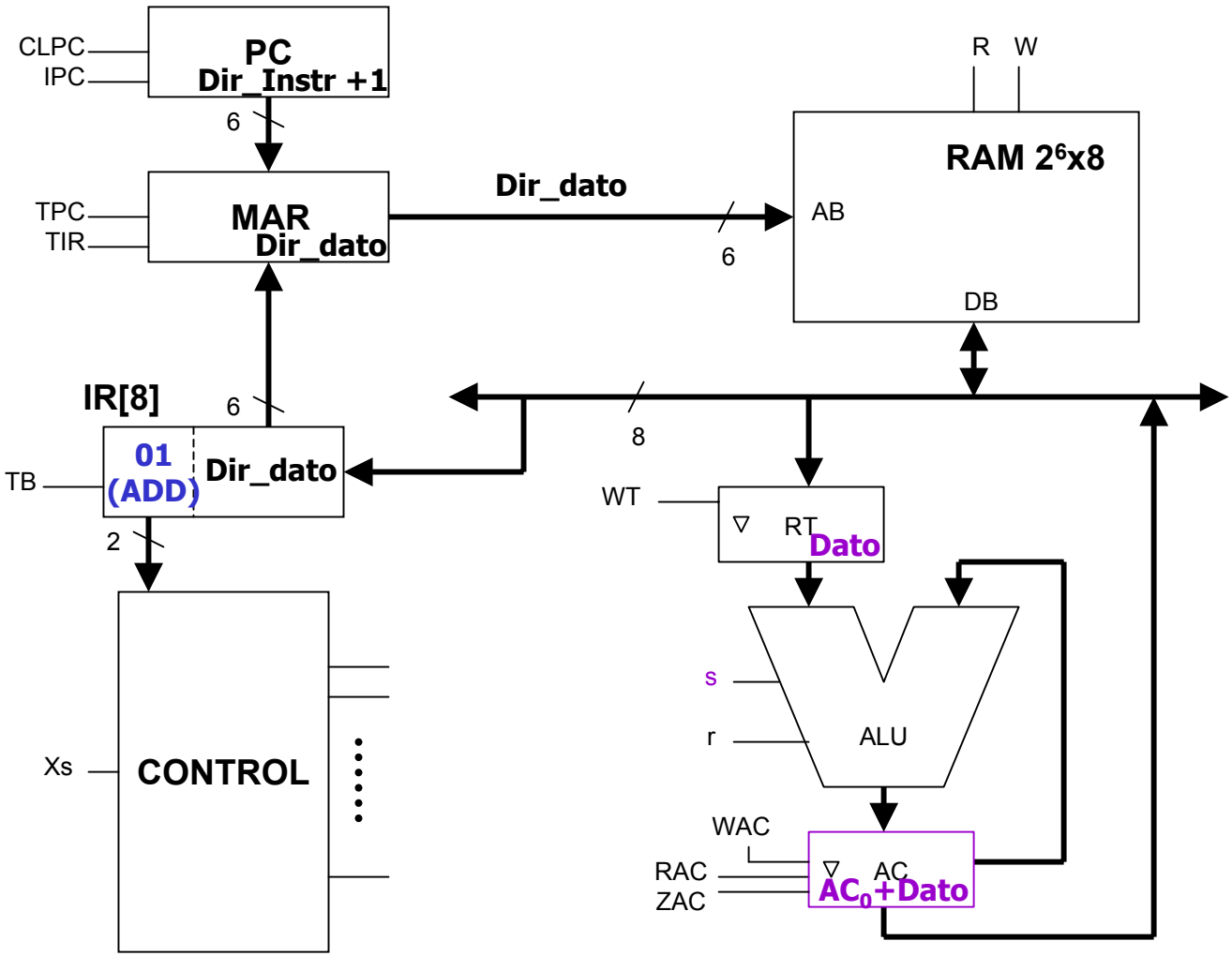
3. AC ← AC + RT



CS1: Instrucciones. Fase de ejecución (fin)

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).



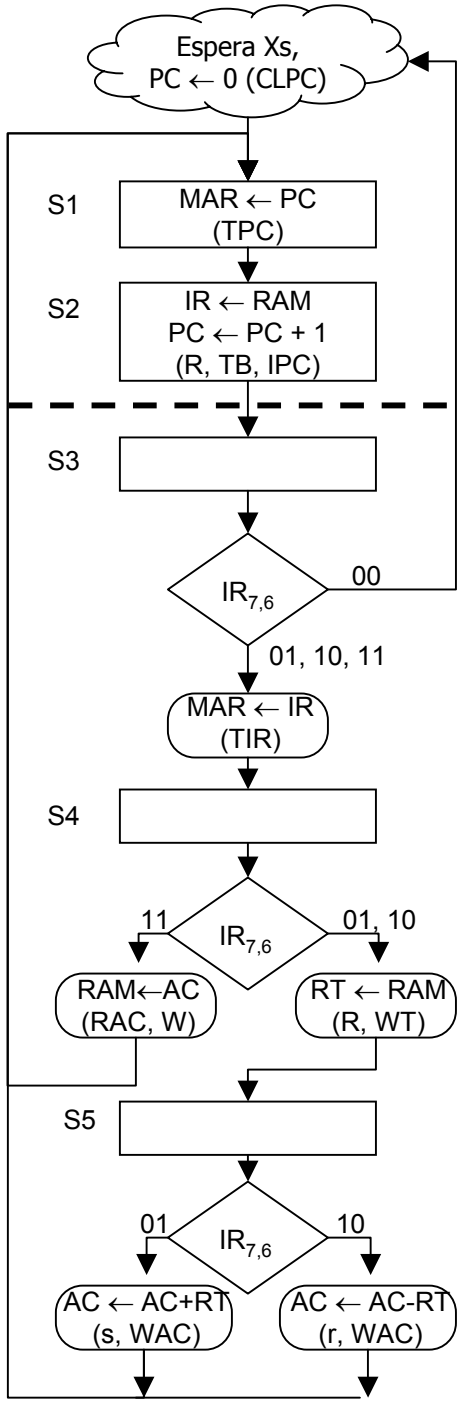
CS1: Controlador. Carta ASM

Búsqueda

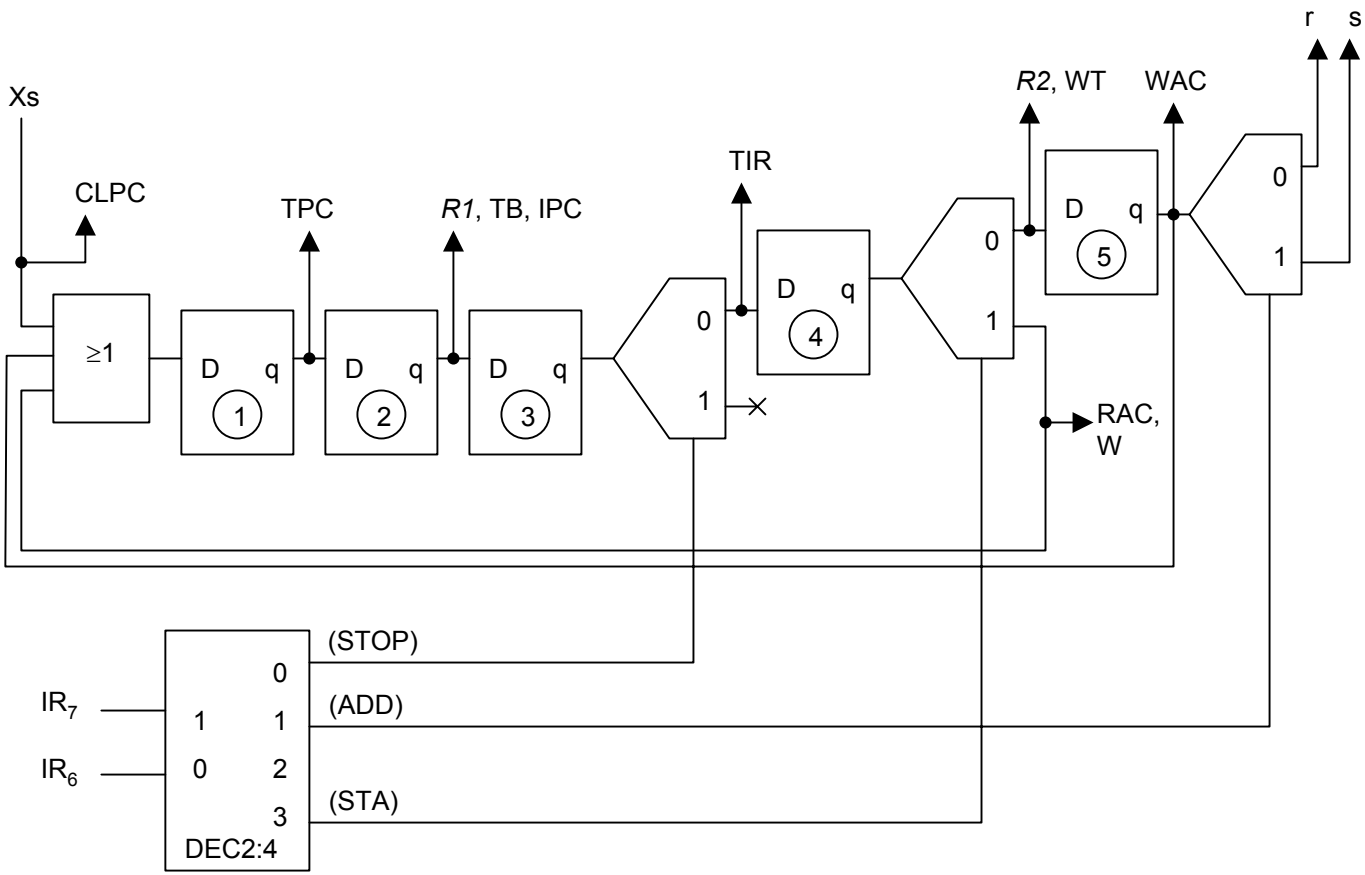
1	MAR ← PC (TPC)
2	IR ← RAM; PC ← PC + 1 (R, TB, IPC)

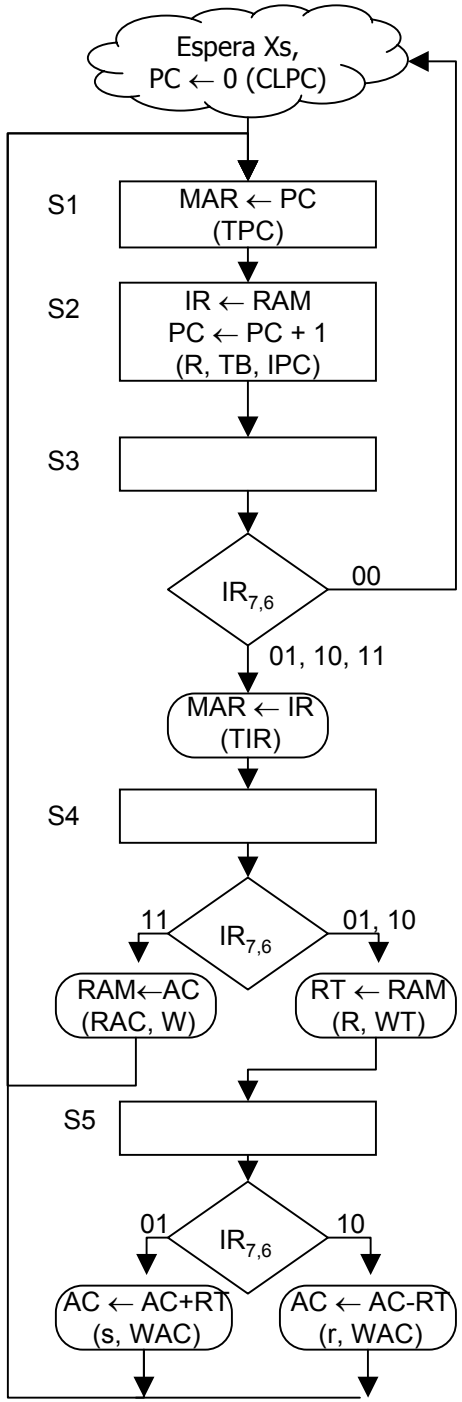
Ejecución

	STOP **$IR_{7,6}$=00**	**ADD A** **$IR_{7,6}$=01**	**SUB A** **$IR_{7,6}$=10**	**STA A** **$IR_{7,6}$=11**
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)	-



CS1: Controlador. Implementación 1 b/e





CS1: Ejemplo de programa

DIR	CONTENIDO RAM		RT
00	1 1 1 0 0 0 0 0	STA $20	RAM($20) ← AC
01	1 0 1 0 0 0 0 0	SUB $20	AC ← AC – RAM($20)
02	0 1 1 1 1 1 1 0	ADD $3E	AC ← AC + RAM($3E)
03	0 1 1 1 1 1 0 1	ADD $3D	AC ← AC + RAM($3D)
04	1 0 1 1 1 1 0 0	SUB $3C	AC ← AC – RAM($3C)
05	1 1 1 1 1 1 1 1	STA $3F	RAM($3F) ← AC
06	0 0 0 0 0 0 0 0	STOP	
...	...	...	
20		Uso transitorio	
...	...	...	
3C		Sustraendo: R	
3D		Sumando 2: S2	
3E		Sumando 1: S1	
3F		Resultado	

⋯➤ AC = 0
⋯➤ AC = S1
⋯➤ AC = S1 + S2
⋯➤ AC = S1 + S2 – R
⋯➤ **Resultado = S1 + S2 - R**

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