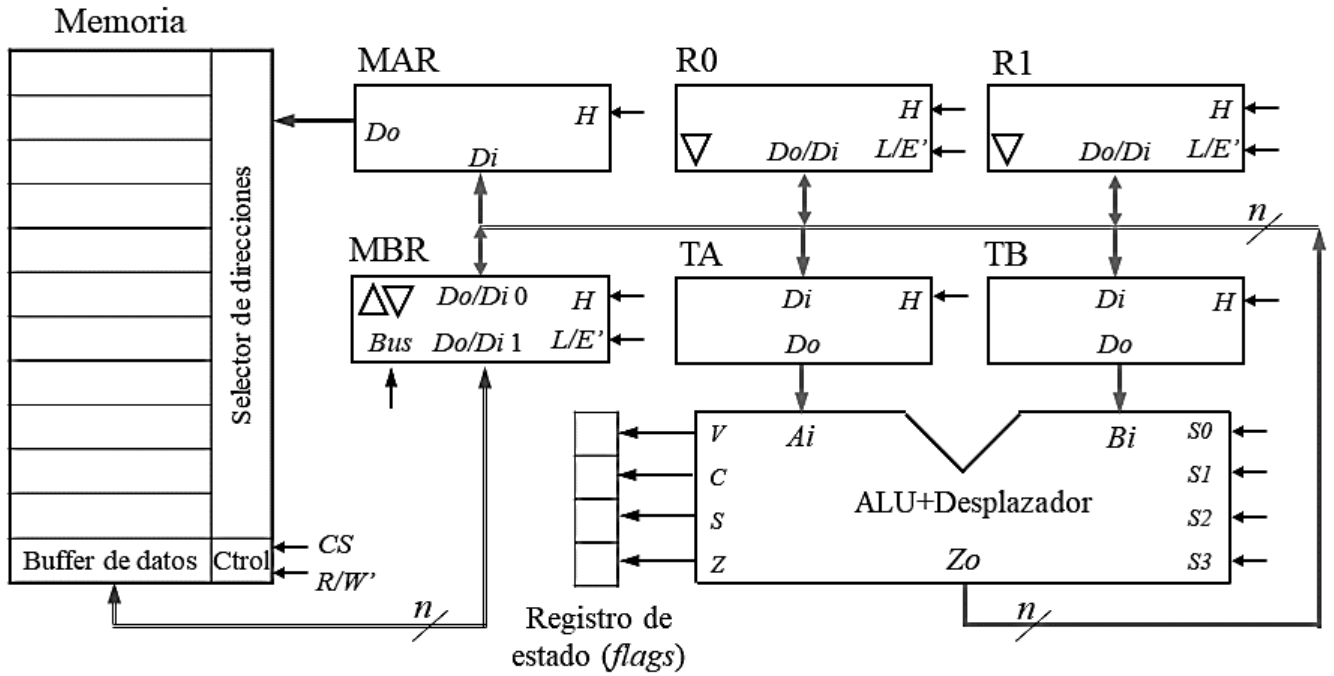
Computador simple

TEMAS:

- Secuenciales cableados → Sistemas programables
- Componentes de un computador básico
- Arquitectura de un computador simple
- Cómo se ejecuta un programa en el computador simple
- Subrutinas

Unidad procesadora (repaso)

El sistema procesador basado en una ALU y registros presentado anteriormente realiza una microoperación específica que se define a través de una palabra de control:

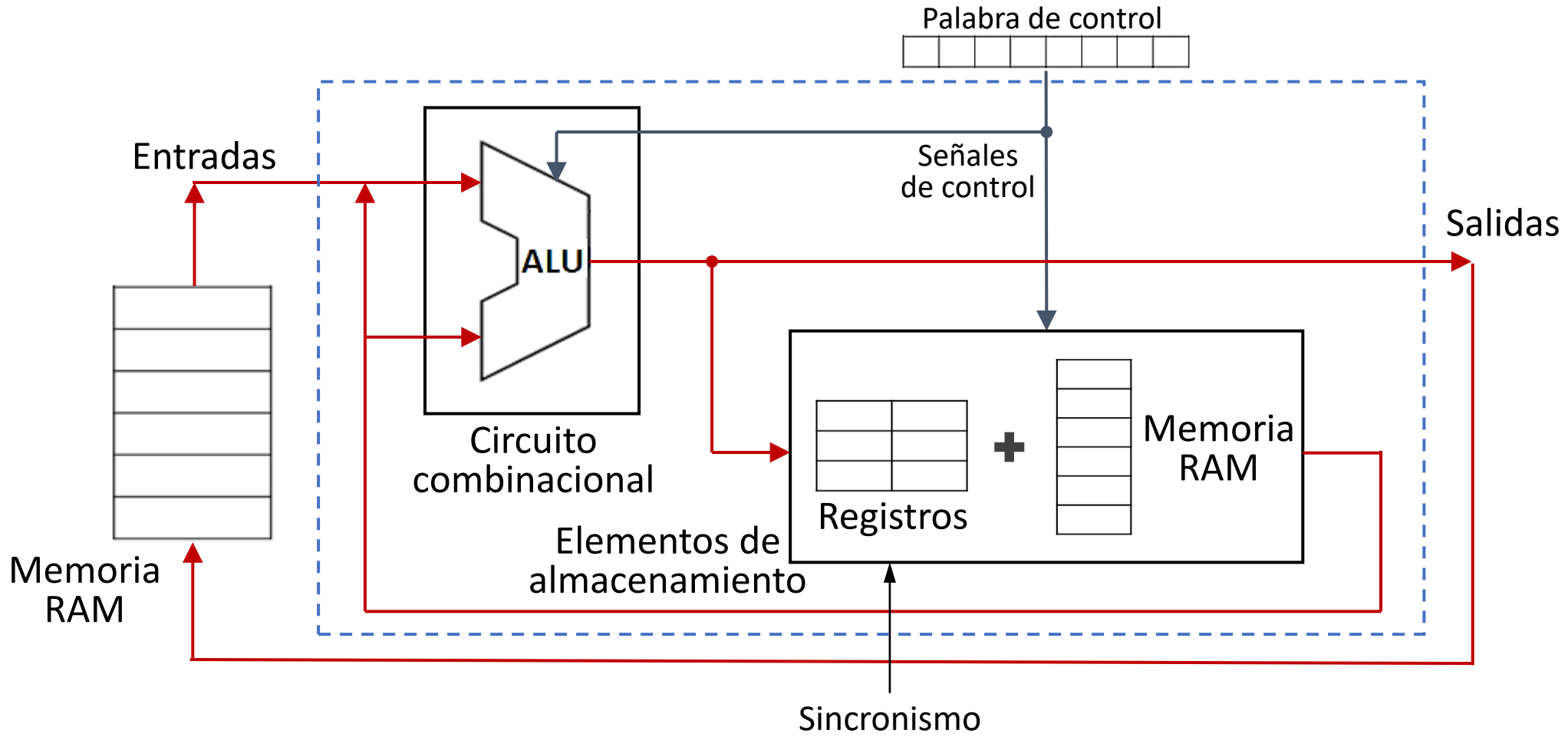


CS	R/W´	H	L/E´	Bus	H	H	L/E´	H	L/E´	H	H	S3	S2	S1	S0
MEM		MBR			MAR	R0		R1		TA	TB	SO			

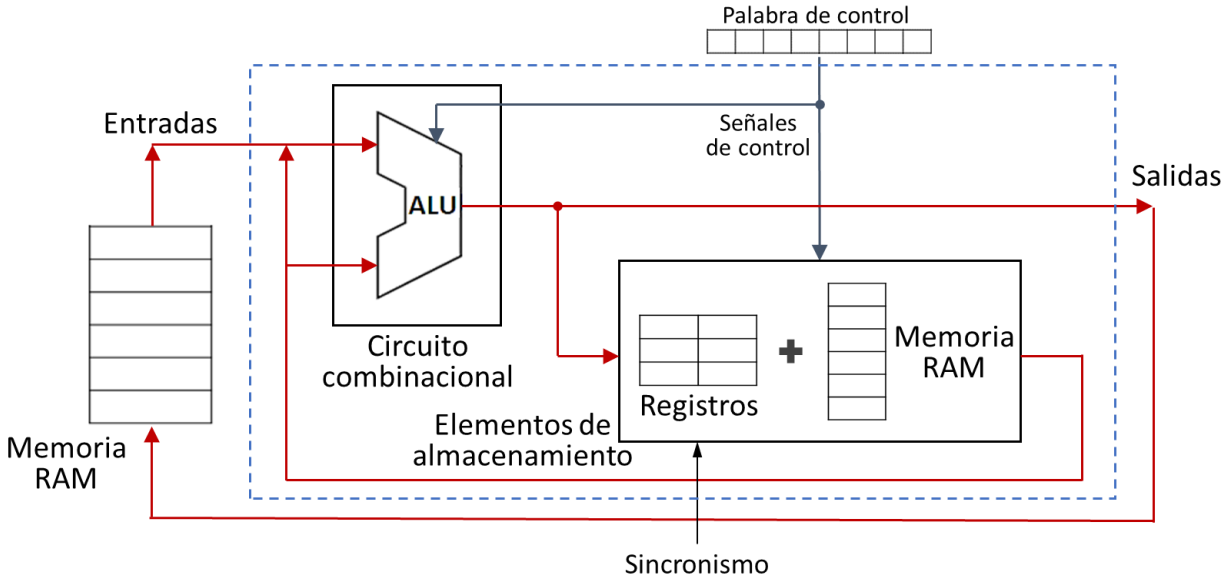
Una microoperación es una transferencia entre registros o entre el MBR y la memoria, u operación aritmética, lógica o de desplazamiento efectuada sobre los datos en la ALU, y que tiene la particularidad de efectuarse en un solo ciclo de reloj)

Unidad procesadora – Modelo estructural

El circuito anterior puede ser visto como un esquema estructural:



Unidad procesadora – Modelo estructural



Las líneas rojas en la figura y los elementos conectados a ellas en el diagrama constituyen la **ruta** o el **camino de datos**.

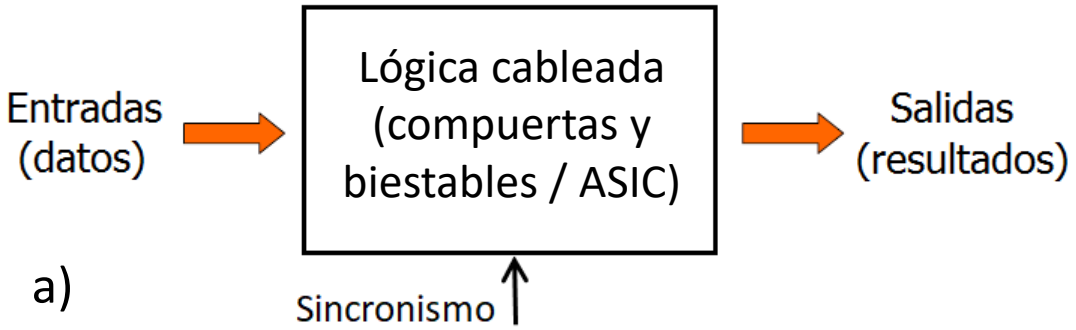
El movimiento y la operación de los datos sobre esa ruta dependen de las líneas que salen de la Palabra de control.

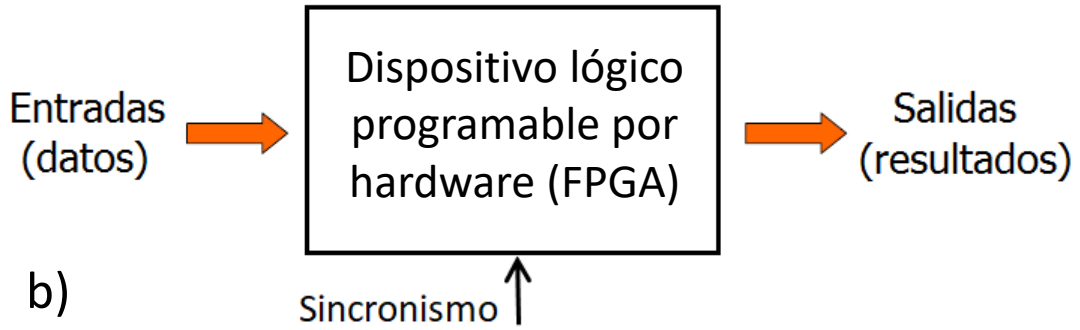
En el caso de que se trate de una operación sencilla, puede realizarse mediante una sola microoperación; en caso contrario, es necesario realizar una secuencia de microoperaciones.

Resta definir quien genera el contenido (los bits) de la Palabra de control, y cómo se secuencian las microoperaciones necesarias para implementar una operación compleja.

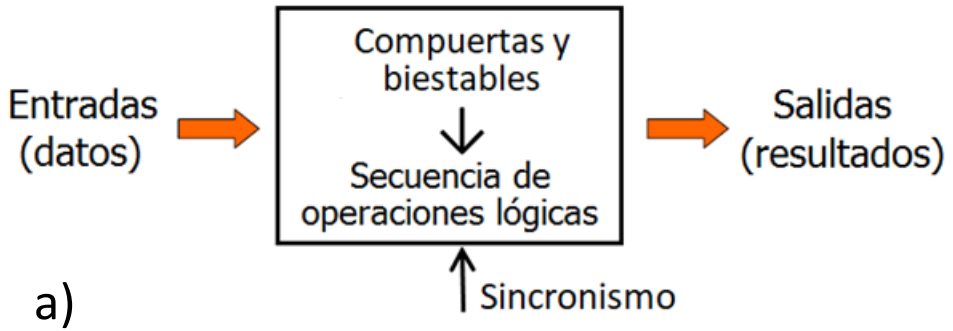
Procesamiento de datos digitales

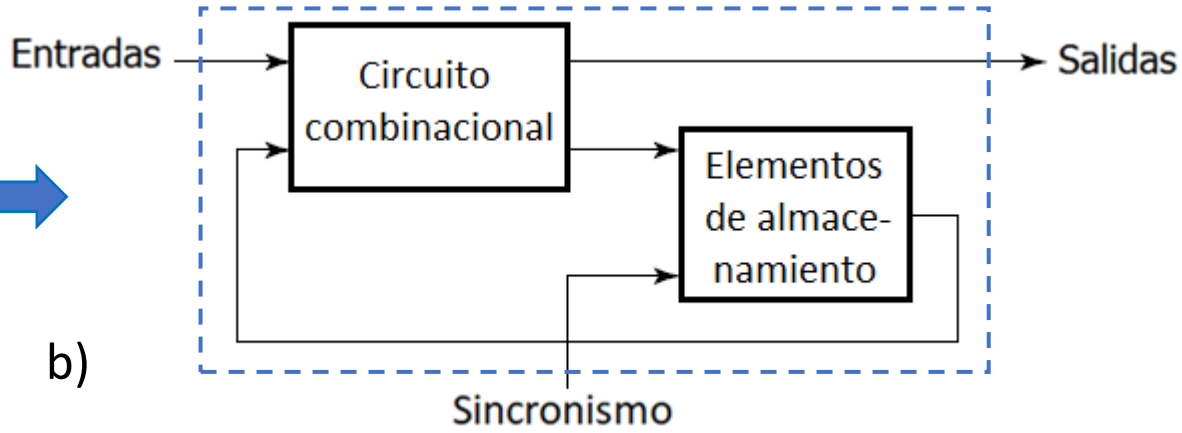
Cualquier problema que implique producir resultados a partir de datos de entrada que puedan ser procesados digitalmente puede ser resuelto mediante soluciones específicas :





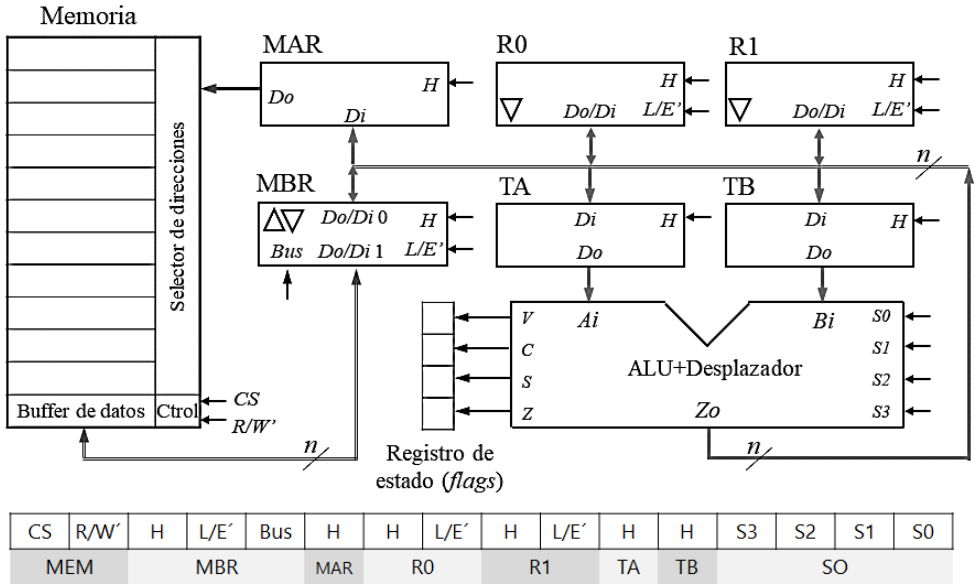
Un circuito secuencial responde a los siguientes esquemas: a) funcional y b) estructural:





Unidad procesadora

El sistema procesador basado en una ALU y registros presentado anteriormente también puede ser asimilado a esquemas funcionales y estructurales similares a un sistema secuencial:



Este esquema es la base para la idea de los **sistemas programables**

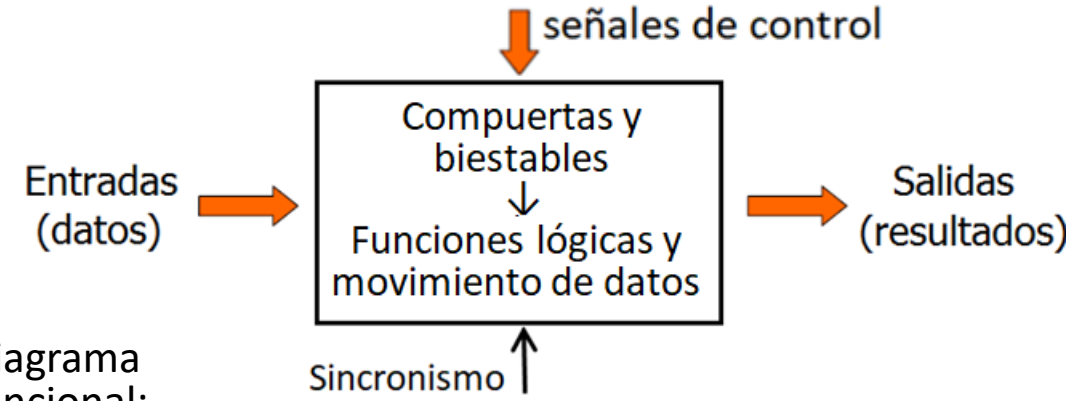
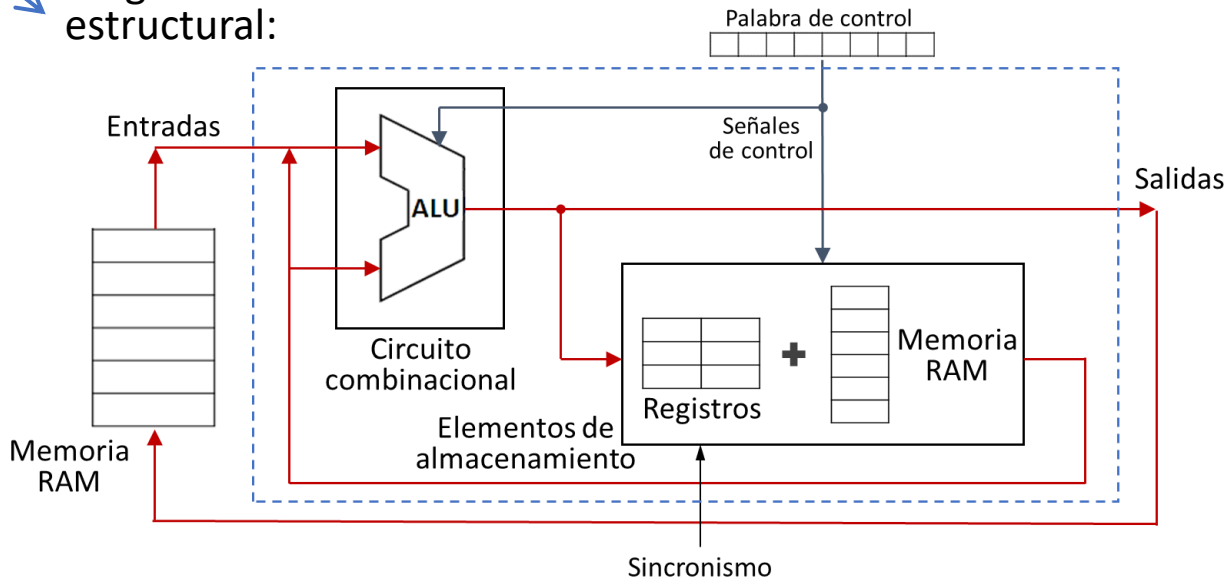


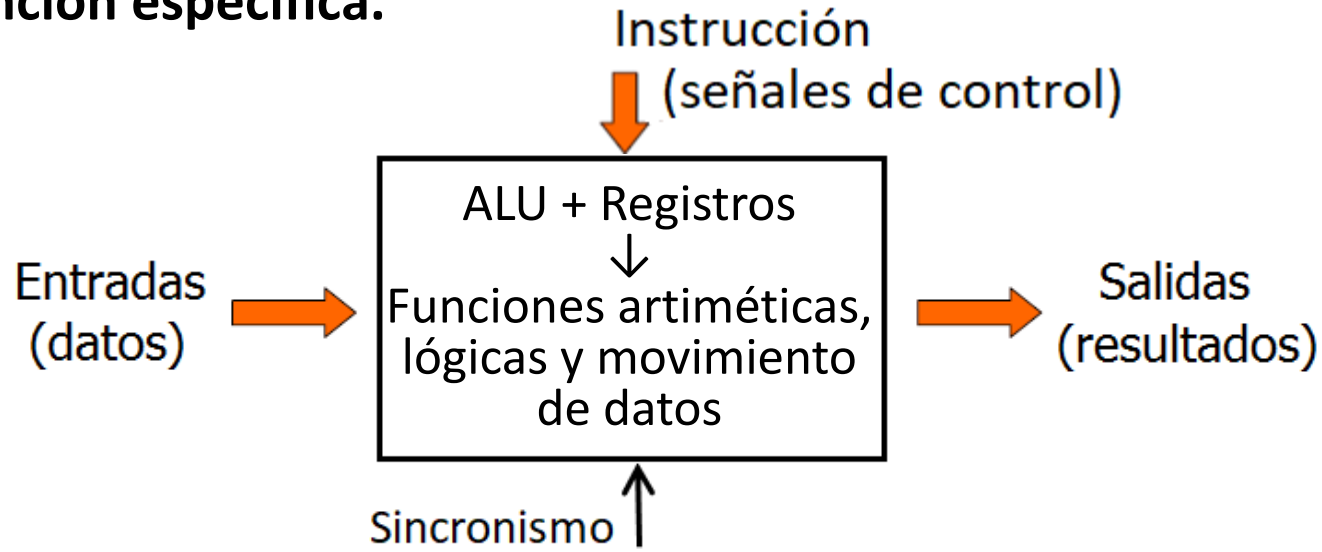
Diagrama funcional:

Diagrama estructural:



Diseños programables por software

Una filosofía de diseño alternativa a las soluciones cableadas o programables por hardware consiste en un circuito integrado por un número determinado de *unidades funcionales* y *registros*, capaces de realizar un **conjunto limitado y predeterminado de operaciones lógicas y de movimiento de datos dentro del circuito**, pero que a diferencia de los anteriores, **no realiza ninguna función específica.**



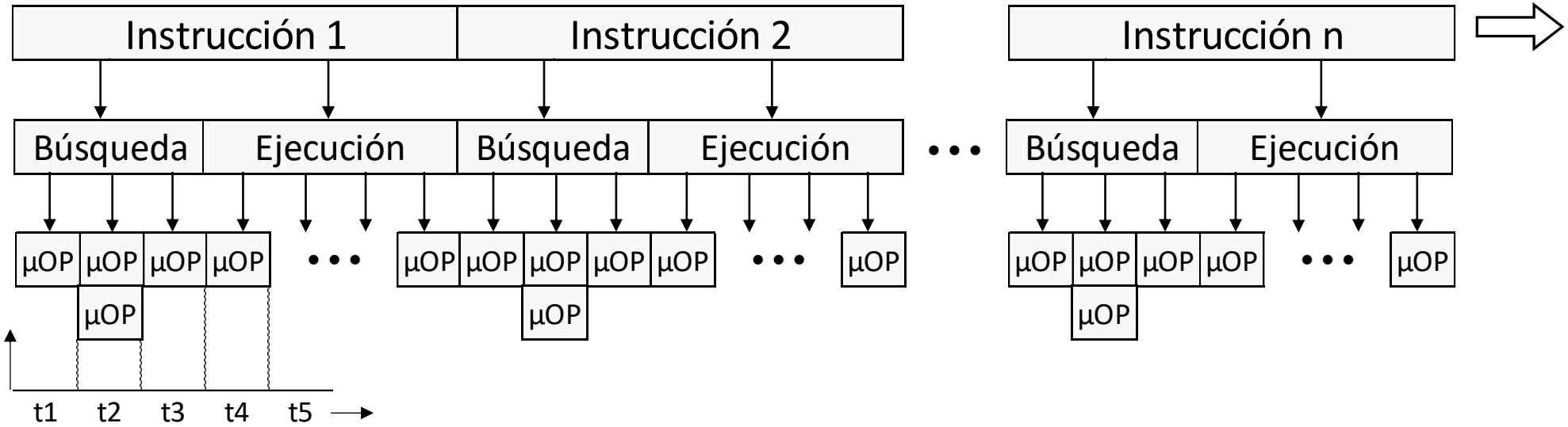
Sin embargo, el movimiento de datos dentro de la estructura y la operación específica que el circuito realiza en cada paso (determinado por la señal de sincronismo) están definidos por un conjunto de **señales de control** generadas por una ***instrucción***.

Diseños programables por software

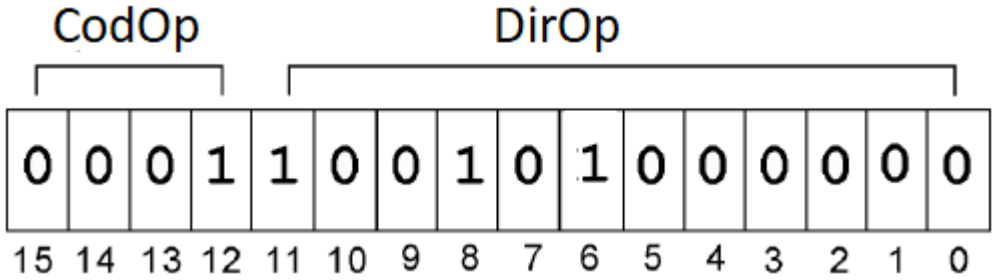
- Las operaciones posibles del circuito están determinadas por un conjunto finito y predefinido de instrucciones, denominado **repertorio de instrucciones máquina.**
- Una instrucción máquina es, formalmente, un código binario que se *interpreta* y *ejecuta* por parte del procesador mediante una secuencia de **microoperaciones.**
- Un conjunto de instrucciones máquina pueden combinarse libremente para, mediante algoritmos, crear un **programa**, que se almacena en una memoria (≡ ***software***)
- Cada instrucción está estructurada en **campos**, que deben contener la información de:
  - La *operación específica* que debe realizarse (codificada en un *código de operación*).
  - La *localización* en la ruta de datos *del o los operandos* fuente y de destino.
  - La *dirección* en la memoria de la siguiente instrucción del programa.
- Una **unidad de control** del sistema debe proveer la secuencia de microoperaciones que realicen la búsqueda en la memoria de cada instrucción, la decodificación de su código de operación, y la generación de las microoperaciones sobre el camino de datos que ejecuten la operación indicada.

Diseños programables por software

- Ejemplo de programa



- Ejemplo de estructura de una instrucción simple:



CodOp representa el *código de operación*: lo que la instrucción tiene que hacer.

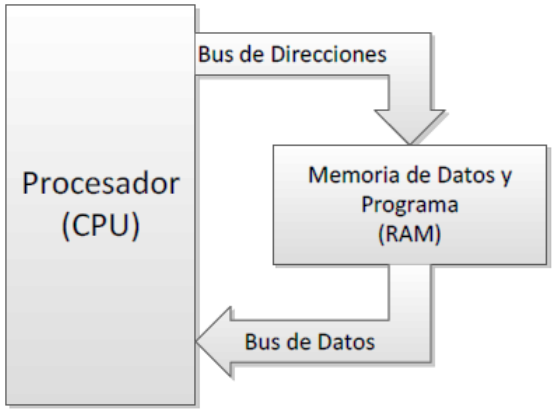
DirOp es la *dirección del operando*: donde está el dato que hay que procesar

Modelos básicos de un sistema programable

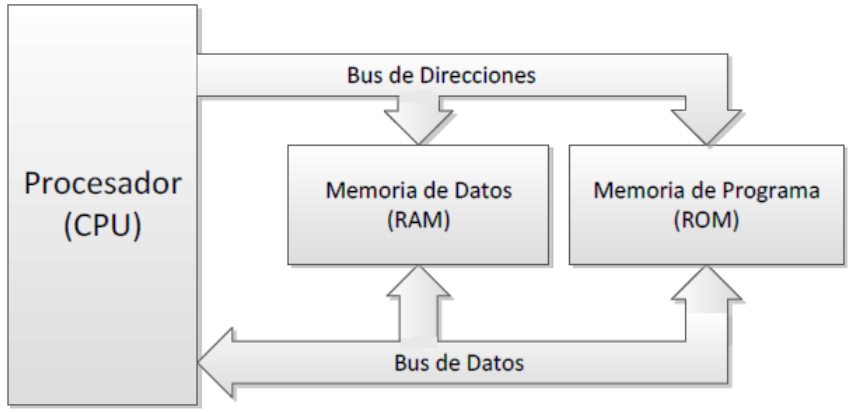
- Debido a que tanto las instrucciones como los datos se almacenan en la memora, la **unidad de control** debe determinar la manera de diferenciarla, de acuerdo a la organización:

Modelo *Von Neumann*:
(a) las instrucciones y los datos comparten la memoria, o
(b) si están separadas comparten las líneas de comunicación con el procesador.

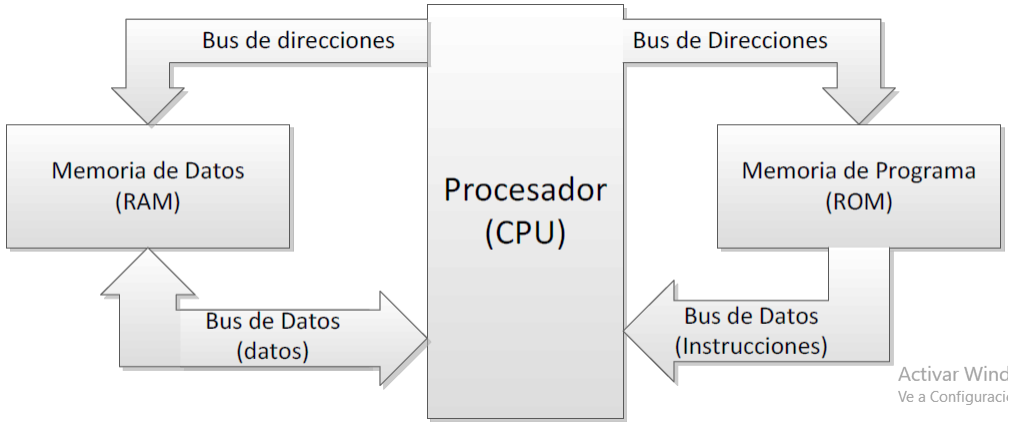
a)



b)

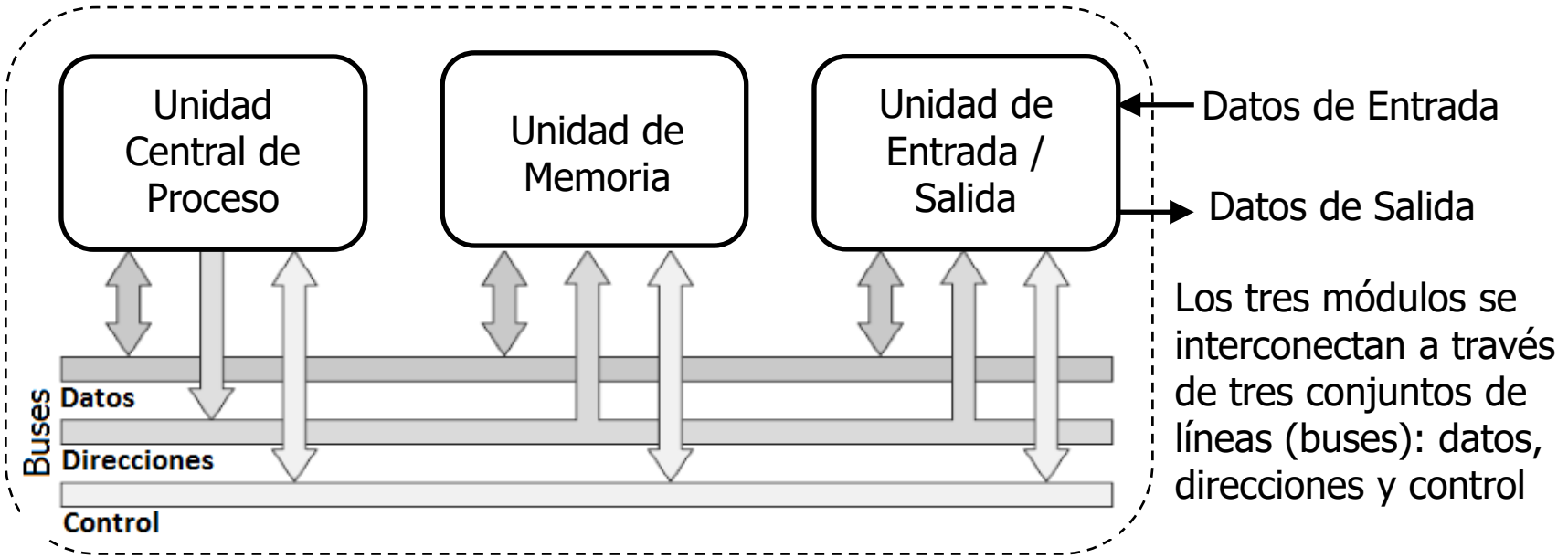


Modelo *Harvard*:
el procesador tiene memorias y líneas de comunicación separadas para instrucciones y datos.

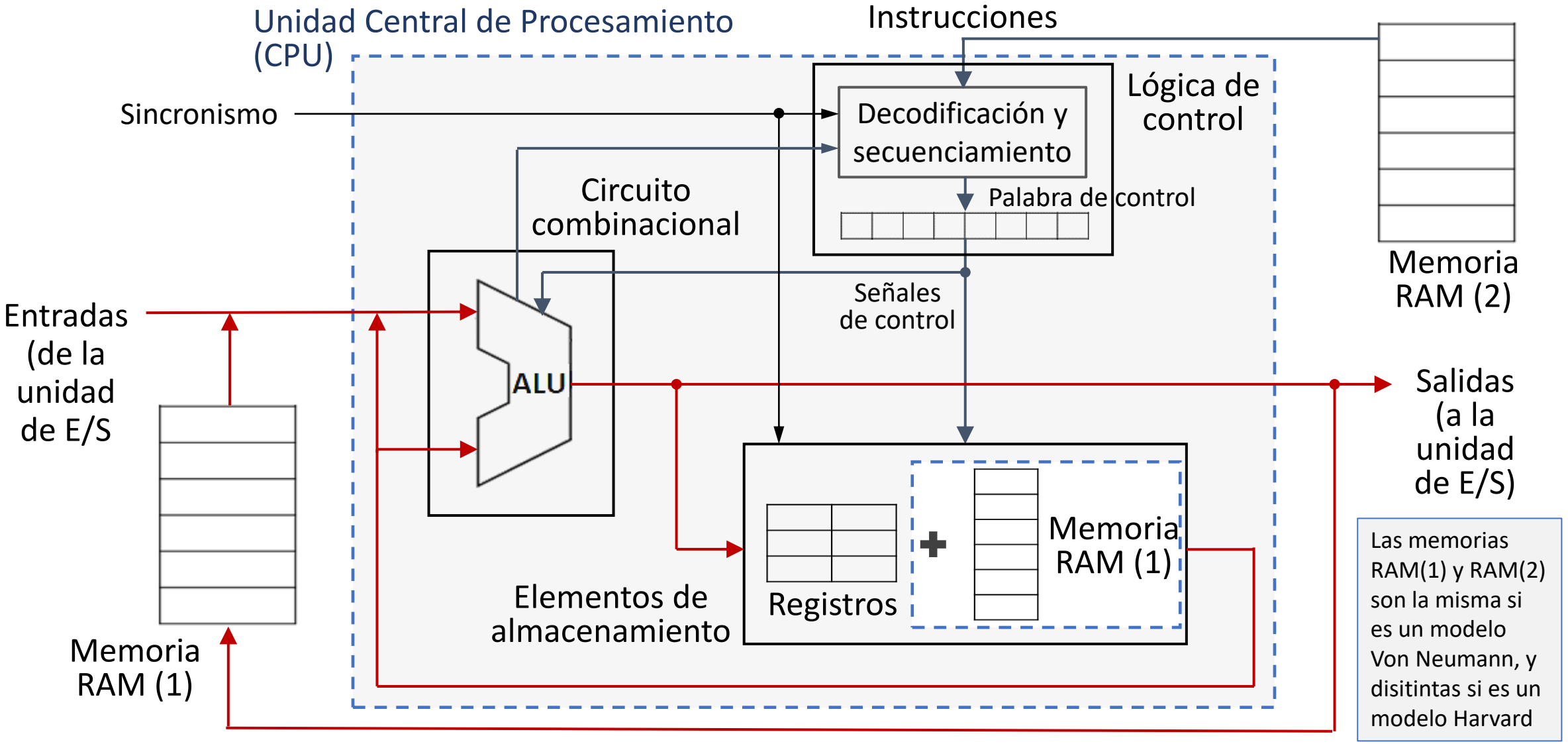


Componentes del computador

- Un módulo de **Entrada / Salida**, por el que ingresan y egresan los datos;
- Un módulo de **Memoria**, en el que se almacenan datos de entrada y resultados procesados, así como las instrucciones del programa;
- Una **Unidad Central de Proceso (CPU)**, en la que se interpretan las instrucciones del programa, se realizan las operaciones lógicas y matemáticas sobre los datos provenientes de los módulos anteriores y se envían a estos los resultados de ese procesamiento.

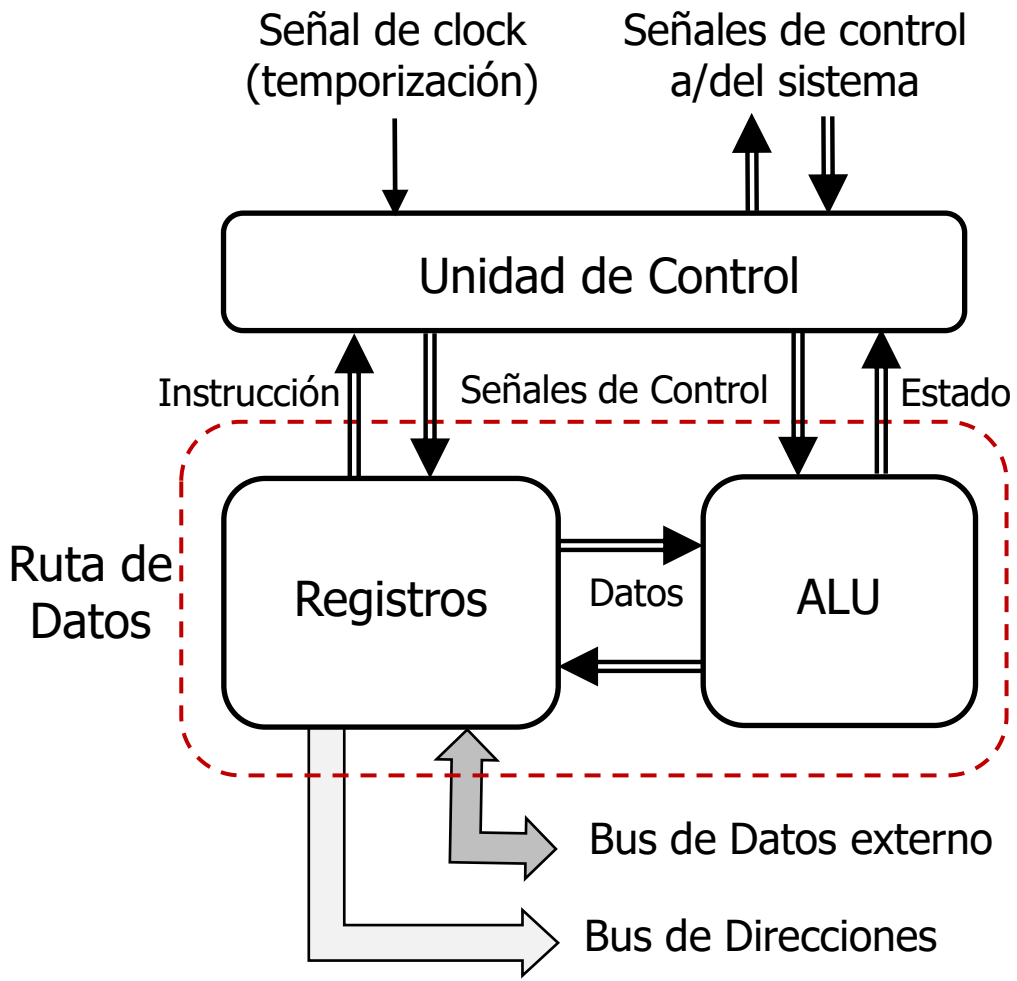


Computador simple



Las memorias RAM(1) y RAM(2) son la misma si es un modelo Von Neumann, y disitintas si es un modelo Harvard

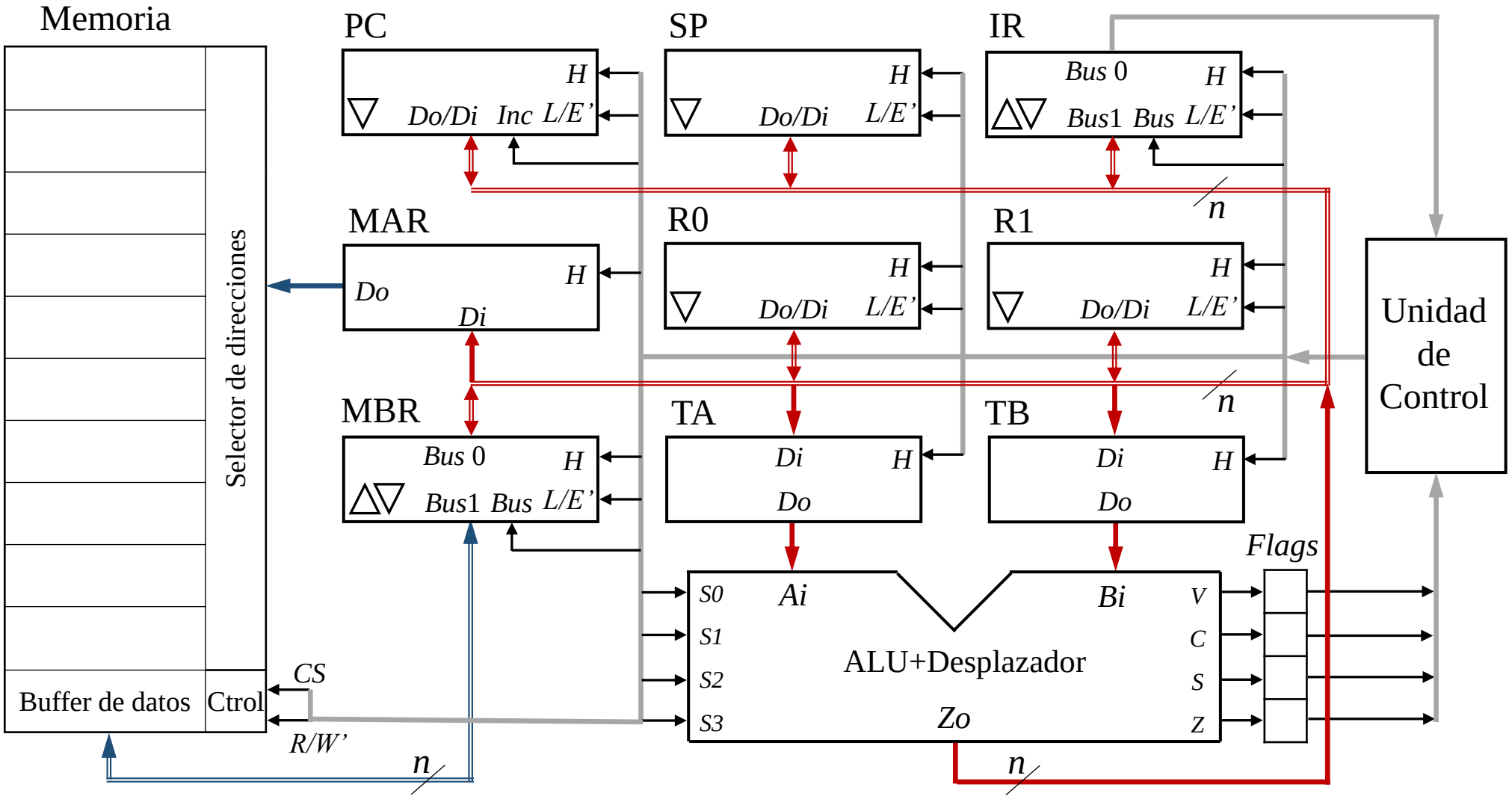
Unidad Central de Proceso (CPU)



La CPU está compuesta por:

- La Ruta de Datos, integrada por:
  - Un conjunto de **Registros** de uso *general,* en el que se almacenan temporalmente los datos que están siendo procesados, y otros de uso *específico*, con información utilizada para la interrelación entre los componentes;
  - Una **Unidad Aritmético Lógica (ALU),** en la que se realizan las operaciones entre los datos;
- La **Unidad de Control**, que decodifica las instrucciones del programa y en combinación con un circuito generador de señales de tiempo, genera las señales de control para los componentes anteriores.

Computador simple



TA; TB = Registros temporales de la ALU

R0; R1 = Registros de uso general

MAR = Registro de direcciones de memoria

MBR = Registro separador de datos

PC = Contador de Programa

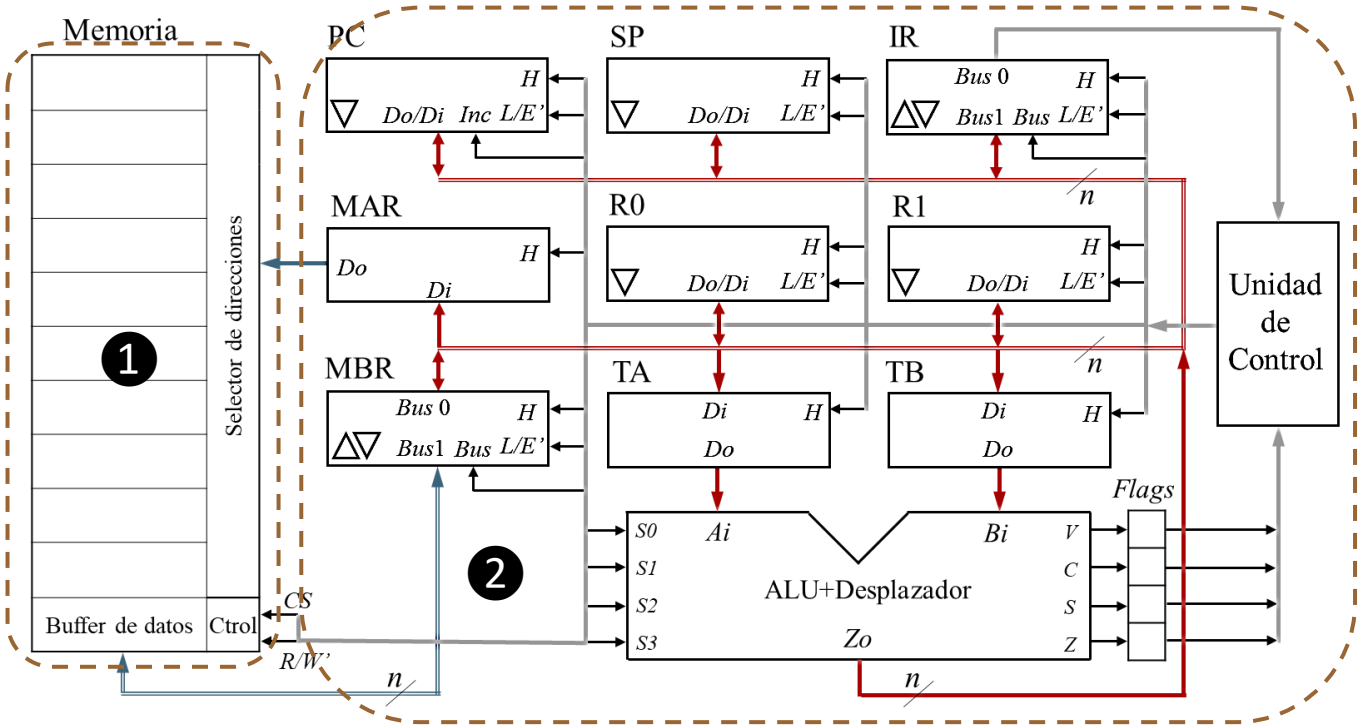
SP = Puntero de pila

IR = Registro de instrucción

Computador simple

En el esquema, se visualizan:

1. Memoria principal: representada como un array de palabras, con sus buses de datos y direcciones, y líneas de control: selección (*CS*) y lectura/escritura (*R/W´*).
2. Unidad central de proceso (CPU): constituida por todos los bloques restantes



Las estructuras de interconexión representadas son: los buses de direcciones y de datos entre la CPU y la memoria (líneas azules); los buses internos de datos de la CPU (líneas rojas), unidireccionales (una flecha) y bidireccionales (dos flechas); y el bus de líneas de control (líneas grises). No están representadas las líneas de sincronismo (clock) ni de alimentación.

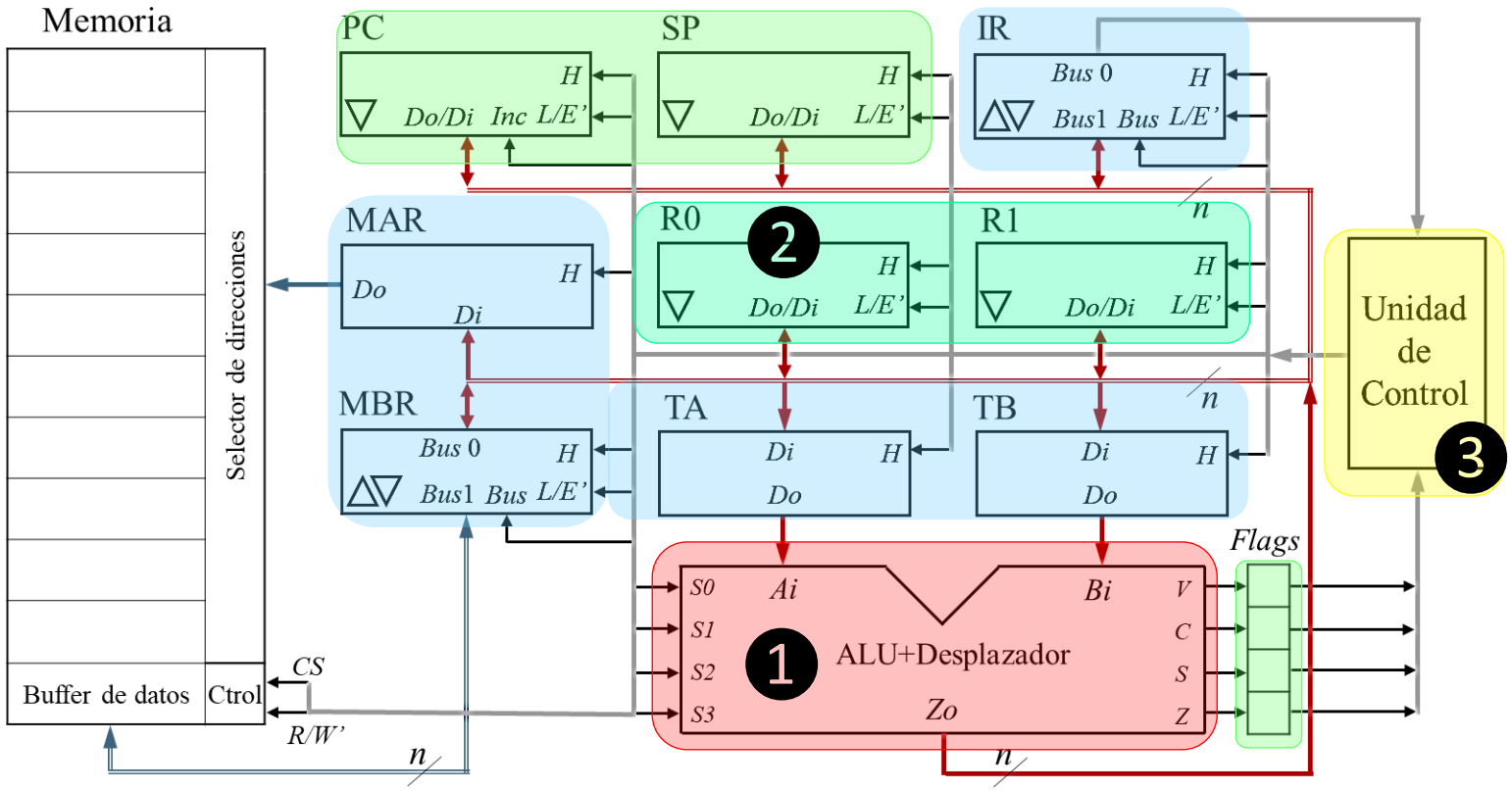
Computador simple

La CPU está compuesta por:

1. La Unidad Aritmético Lógica (ALU), con dos conjuntos de entradas de datos y un conjunto de salida, entradas de selección de operación, y las salidas a los bits de condición o flags.

2. Un conjunto de registros:
   - De uso general (R0 y R1)
   - De uso específico, utilizables por el programador: (PC, SP, Flags).
   - De uso específico, no visibles al programador: (IR, MBR, MAR, T1 y T2)

3. La Unidad de Control (UC), que recibe datos del IR y de los flags, y genera las señales de control para todos los restantes elementos.



CPU: Registros de uso específico

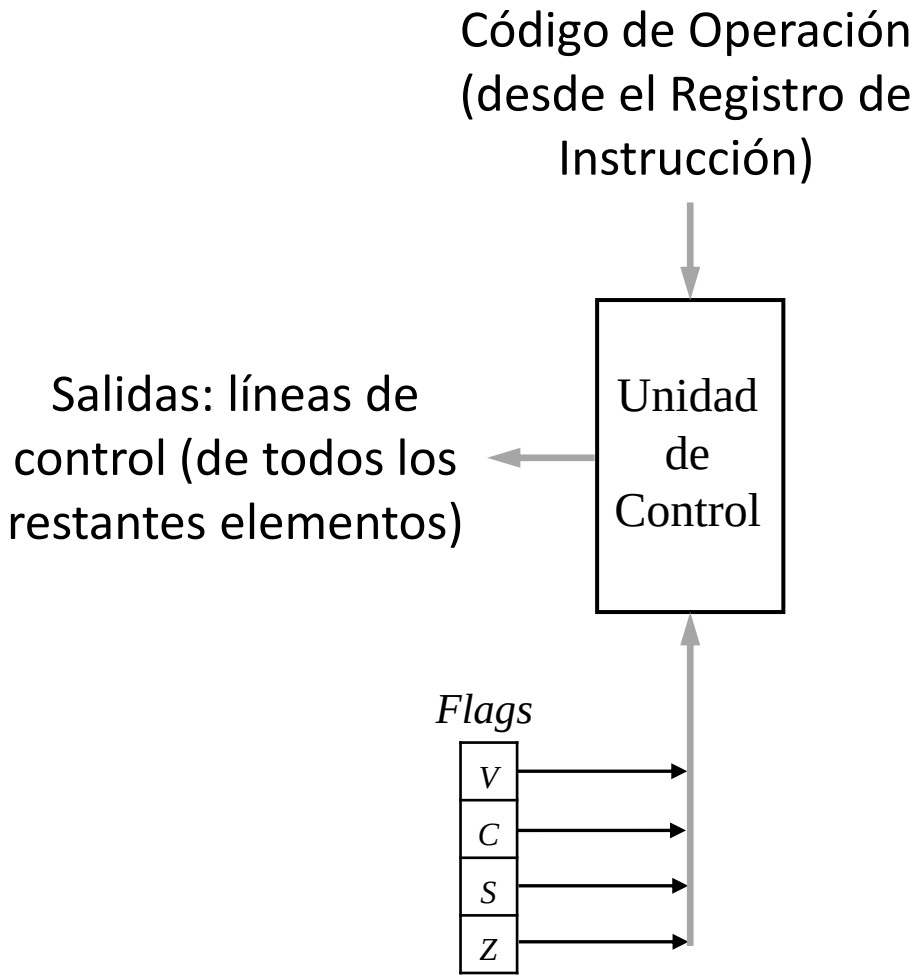
	NOMBRE	FUNCIÓN
VISIBLES	**Contador de Programa** (PC, *Program Counter*)	Contiene la dirección de memoria de la instrucción en curso y se incrementa o modifica apuntando a la dirección de la próxima.
VISIBLES	**Puntero de pila** (SP, *Stack pointer*)	Apunta a la dirección tope de la pila en memoria (utilizada en la ejecución de subrutinas e interrupciones).
NO VISIBLES	**Registro de Instrucción** (IR, *Instruction Register*)	Contiene el valor binario de la instrucción en curso, para ser decodificado y secuenciado por la Unidad de Control.
NO VISIBLES	**Registro de Direcciones** (MAR, *Memory Adress Register*)	Contiene la dirección actual de memoria.
NO VISIBLES	**Registro de Datos** (MBR, *Memory Buffer Register*)	Almacena el dato a ser escrito en la memoria o el leído de ésta.
NO VISIBLES	**Registros temporales** (T1 y T2)	Almacenan los datos a operar en la ALU.

La condición de visible (al programador) significa que puede ser accedido desde el repertorio de instrucciones del procesador.

CPU: Unidad de Control

La Unidad de Control es el bloque que:

- Genera la secuencia de palabras de control para obtener de la memoria la instrucción a ejecutarse;
- Recibe en su entrada el código de la operación que corresponde a la instrucción a ejecutarse, y genera la secuencia de palabras de control particular para ejecutar esa instrucción.
- También tiene como entradas los *flags* de la ALU, a partir de los cuales genera las palabras de control necesarias para realizar los saltos en el programa determinados por las instrucciones condicionales.



Ejecución de un programa

En el ejemplo a visualizar, cada posición de memoria tiene 16 bits.

- Los primeros 4 bits indican la operación a realizar (CodOp)

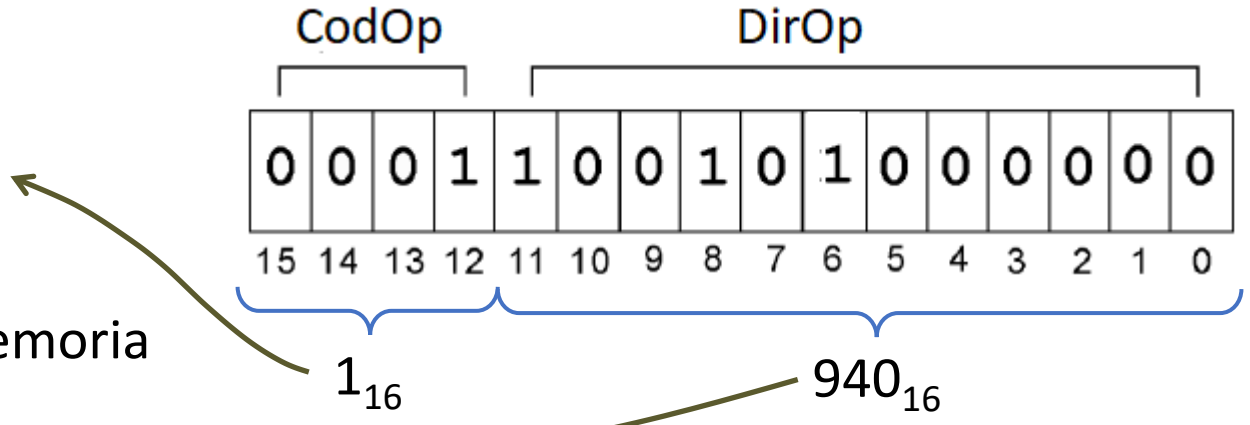
  → Por lo tanto, se pueden realizar $2^4 = 16$ instrucciones distintas

- Los siguientes 12 bits indican una dirección de memoria donde se encuentra el operando (DirOp).

  → Por lo tanto, se pueden direccionar $2^{12} = 4.096$ (4K) palabras de memoria

Las instrucciones a utilizar son:

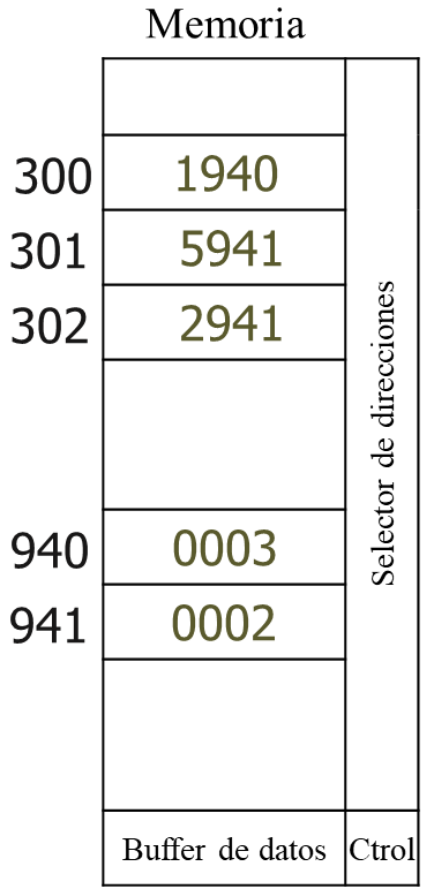
- $0001_2 = 1_{10}$ = cargar R1 desde la memoria
- $0010_2 = 2_{10}$ = almacenar R1 en memoria
- $0101_2 = 5_{10}$ = sumar R1 con un dato en memoria



Ejecución de un programa

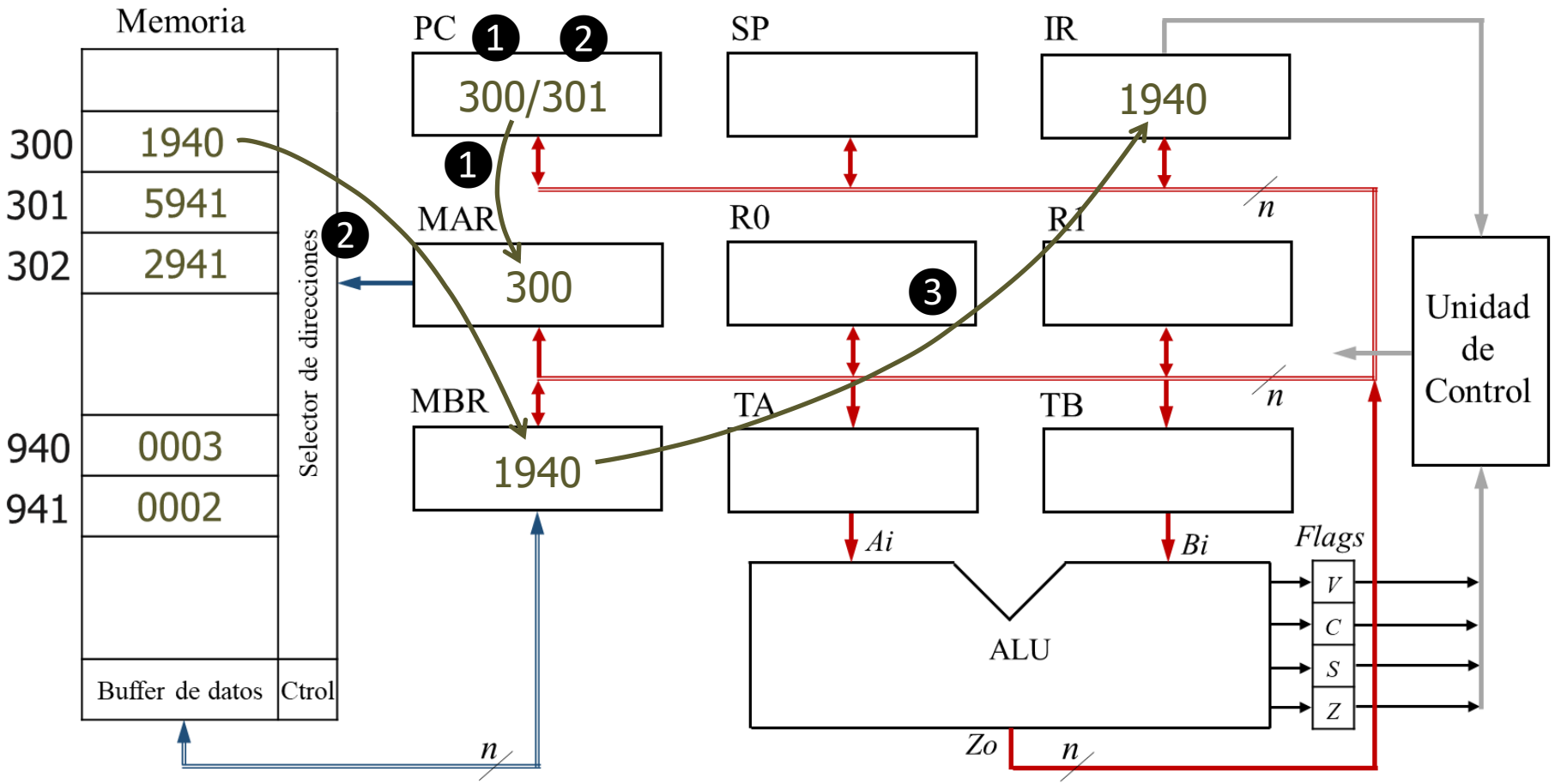
El siguiente ejemplo es la ejecución de un fragmento de programa con tres instrucciones, cargadas desde la dirección de memoria 300_{16} :

1. Cargar en el registro R1 (1_{16}) el contenido de la posición de memoria 940_{16}
2. Sumar el contenido de la posición de memoria 941_{16} al registro R1 y guardar el resultado en el mismo registro (5_{16}).
3. Almacenar el resultado guardado en el registro R1 (2_{16}) en la posición de memoria 941_{16}



Ejecución de un programa

- Instrucción 1: Cargar en el registro R1 el contenido de la posición de memoria 940_{16}



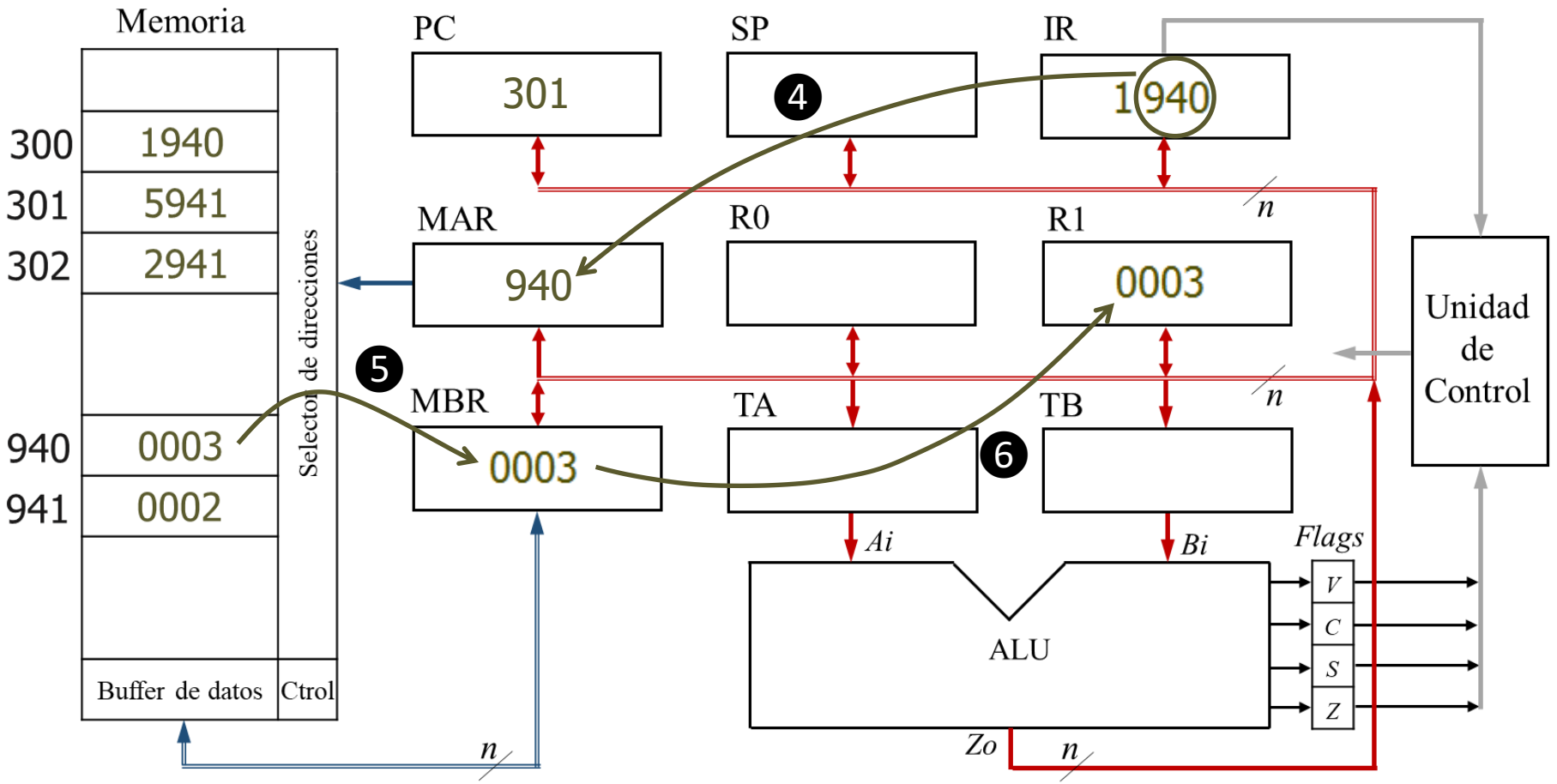
El contador de programa (PC) contiene 300_{16} como la dirección de la primera instrucción.

- t_1: MAR ← PC
- t_2: MBR ← M[MAR]
  PC ← PC +1
- t_3: IR ← MBR

El contenido de esta dirección se carga en el registro de instrucción (IR).

Ejecución de un programa

- Instrucción 1: Los primeros 4 bits en IR (1_{16}) indican que el registro R1 se cargará con un dato proveniente de la dirección especificada en los restantes 12 bits de la instrucción. En este caso tal dirección es 940_{16}, y el dato, 3_{16}

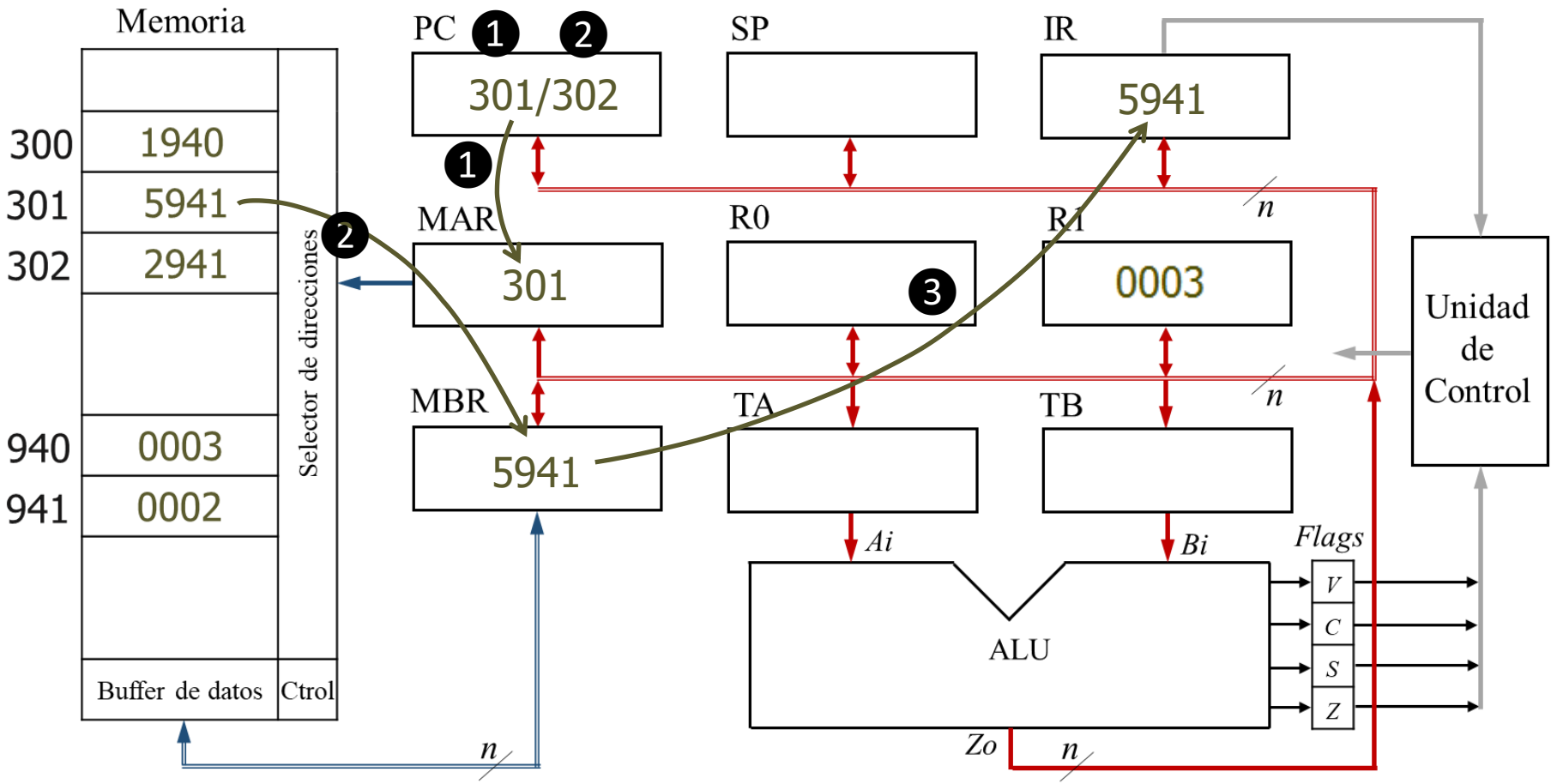


La secuencia de microoperaciones será:

- t_4: MAR ← IR(11:0)
- t_5: MBR ← M[MAR]
- t_6: R1 ← MBR

Ejecución de un programa

- Instrucción 2: Sumar el contenido de la posición de memoria 941_{16} al registro R1 y guardar el resultado en R1.



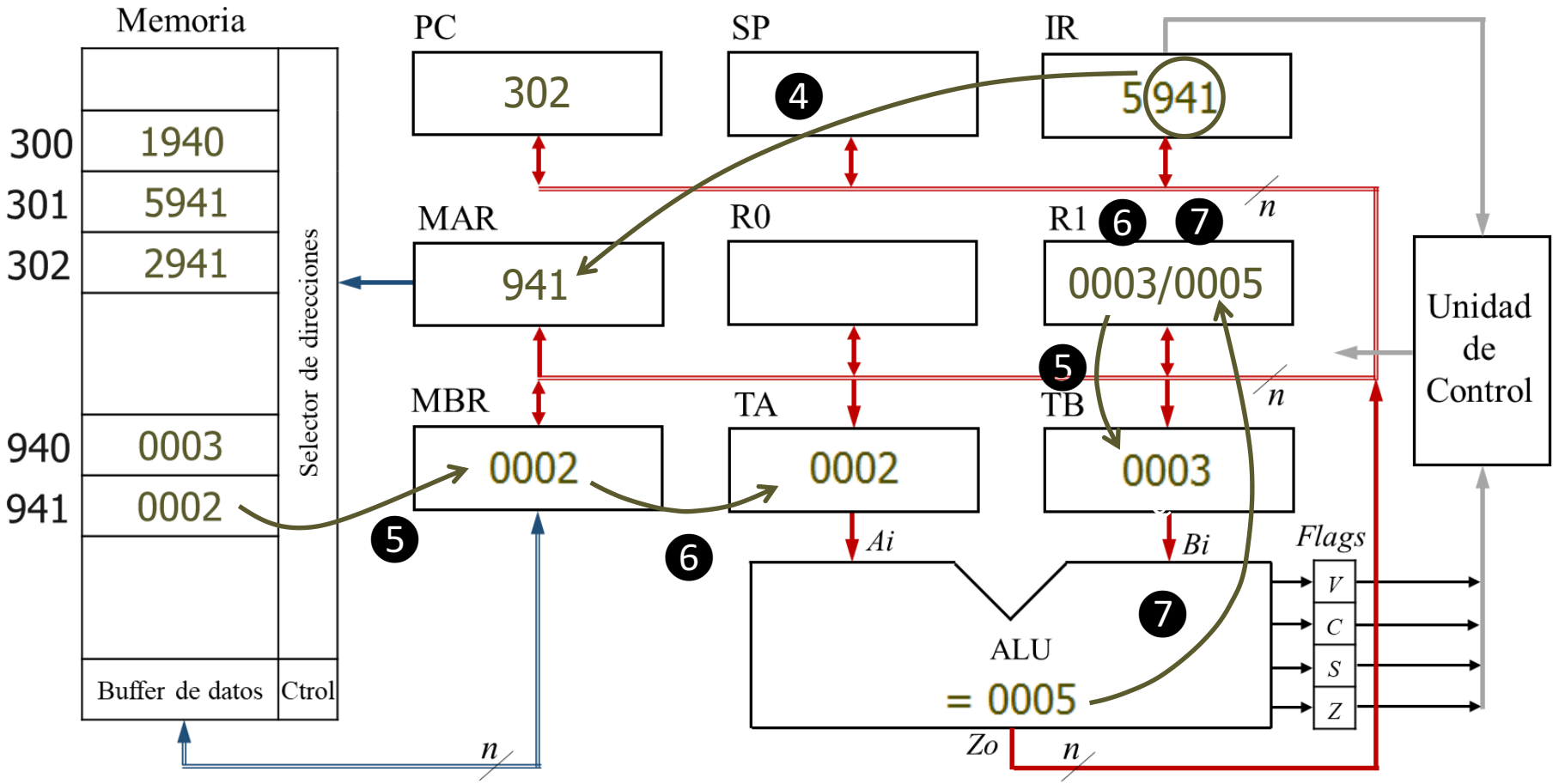
El contador de programa (PC) contiene 301_{16} (dirección de la segunda instrucción).

- t_1: MAR ← PC
- t_2: MBR ← M[MAR]
  PC ← PC +1
- t_3: IR ← MBR

El contenido de esta dirección se vuelve a cargar en el registro de instrucción (IR).

Ejecución de un programa

- Instrucción 2: Los primeros 4 bits en IR (5_{16}) El 5_{16} en IR indica que se debe sumar el contenido de una dirección de memoria especificada -en este caso 941_{16} -, con el contenido del registro R1 y almacenar el resultado en el mismo registro.

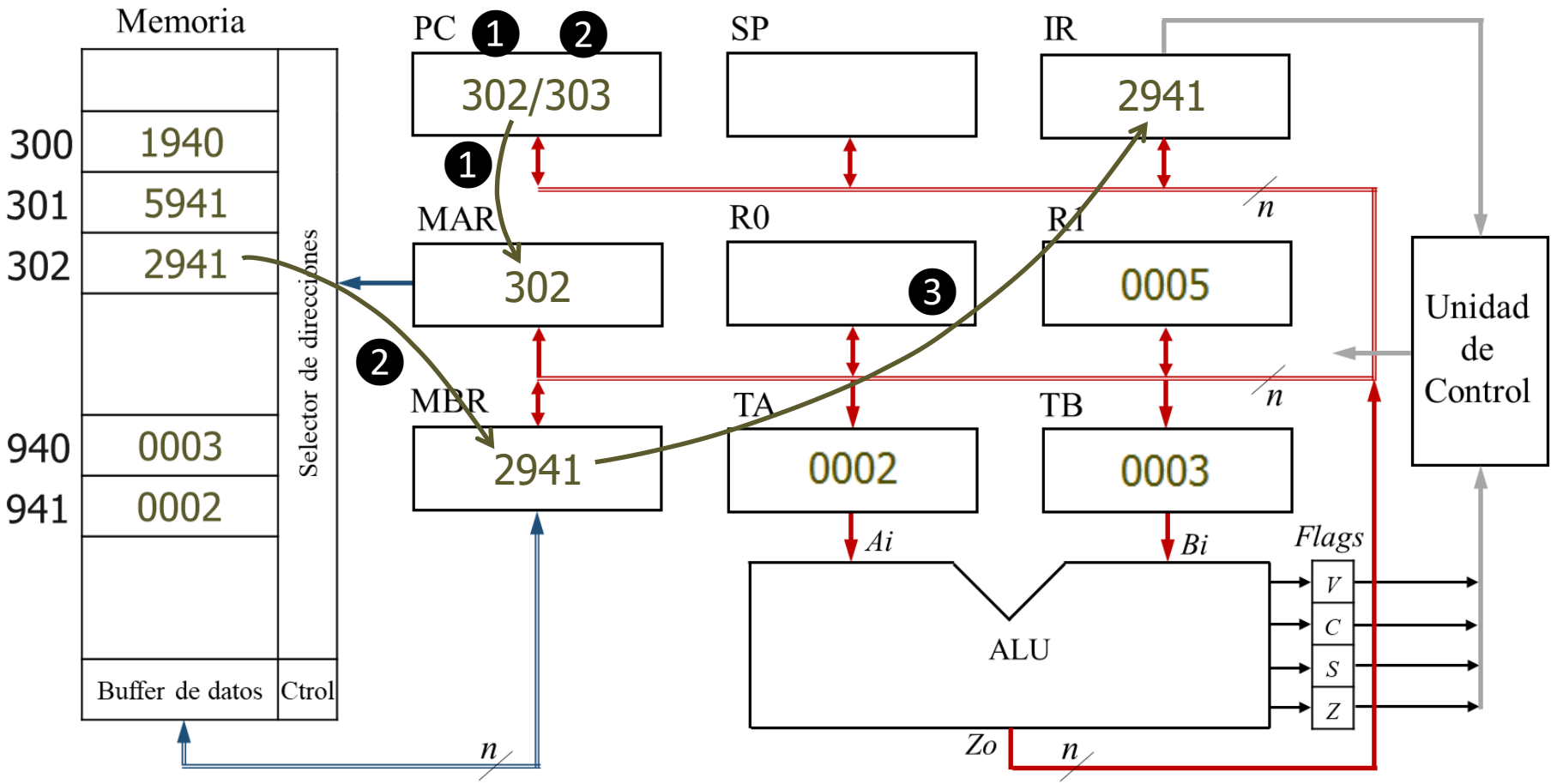


- t_4: MAR ← IR(11:0)
- t_5: MBR ← M[MAR],

  TB ← R1
- t_6: TA ← MBR
- t_7: R1 ← TA + TB

Ejecución de un programa

- Instrucción 3: Almacenar el resultado guardado en el registro R1 en la posición de memoria 941_{16}.



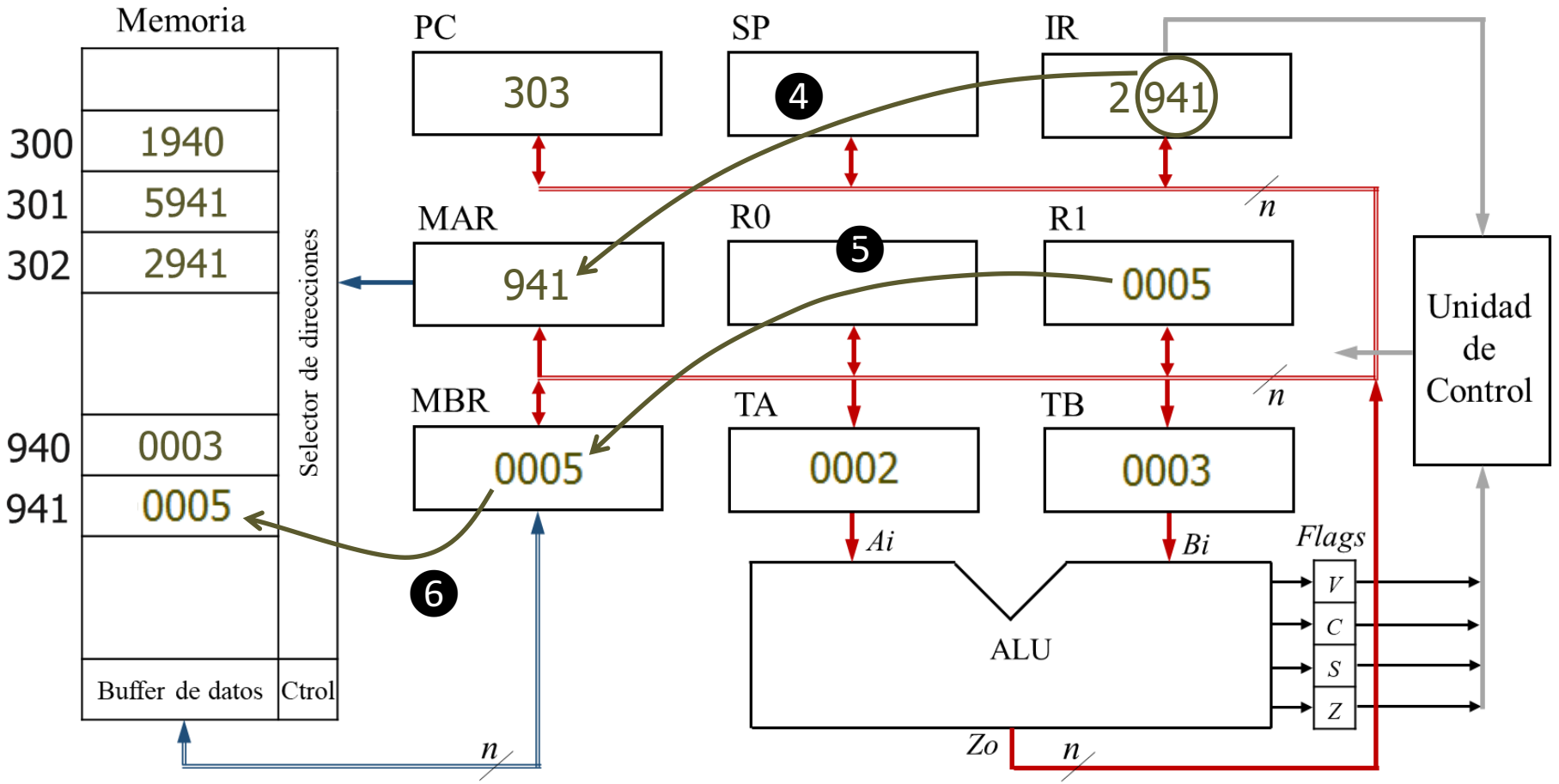
El contador de programa (PC) contiene 302_{16} (dirección de la segunda instrucción).

- t_1: MAR ← PC
- t_2: MBR ← M[MAR]
  PC ← PC +1
- t_3: IR ← MBR

El contenido de esta dirección se vuelve a cargar en el registro de instrucción (IR).

Ejecución de un programa

- Instrucción 3: Los primeros 4 bits en IR (2_{16}) indican que se debe almacenar el contenido del registro R1 en una dirección especificada en los restantes 12 bits de la instrucción -en este caso 941_{16} -.



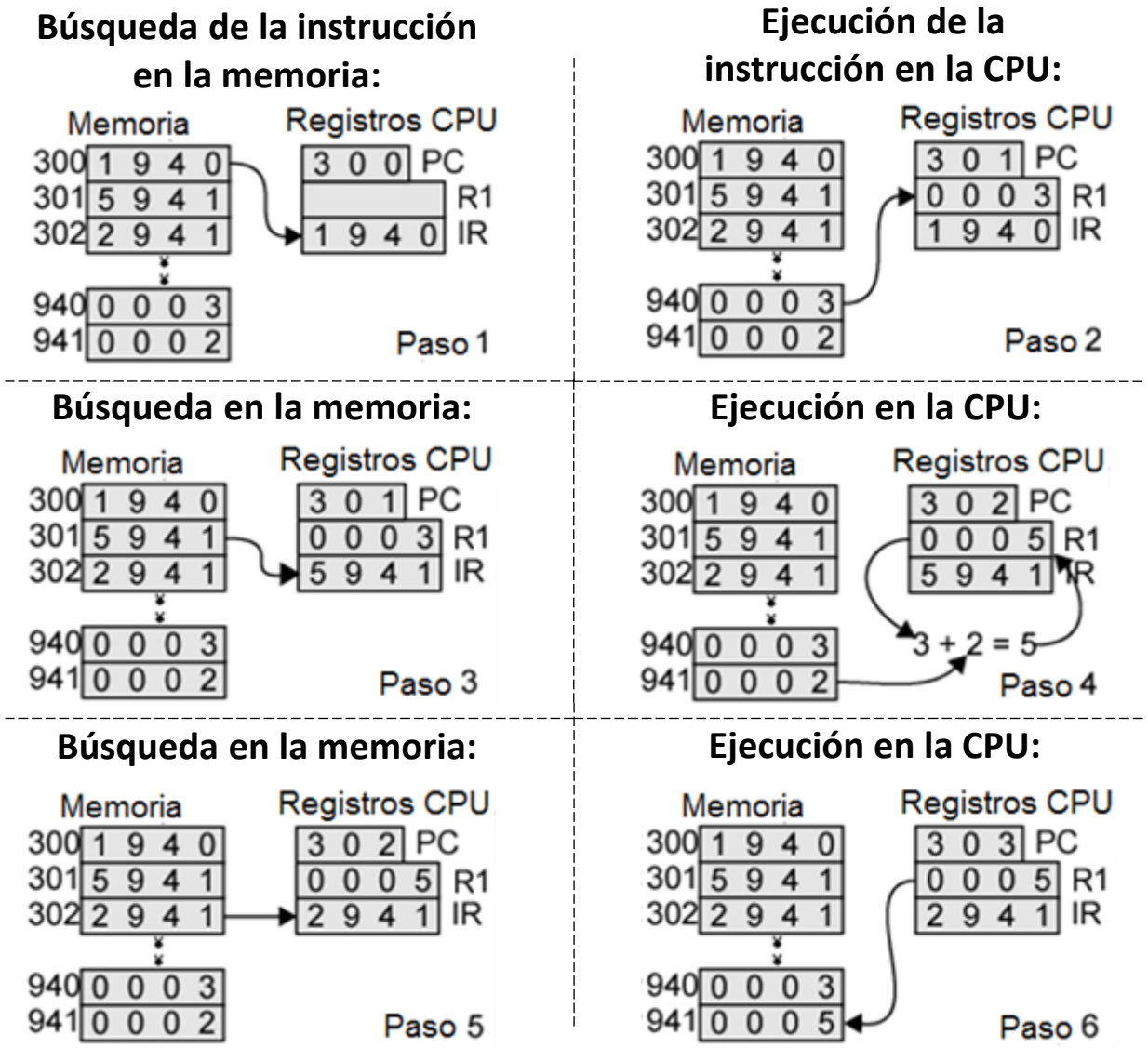
La secuencia de microoperaciones será:

- t_4: MAR ← IR(11:0)
- t_5: MBR ← R1
- t_6: M[MAR] ← MBR

Ejecución de un programa

La ejecución del programa vista desde la interacción de la memoria con los registros (prescindiendo de las microoperaciones y registros intermedios utilizados) es:

- Instrucción 1
- Instrucción 2
- Instrucción 3



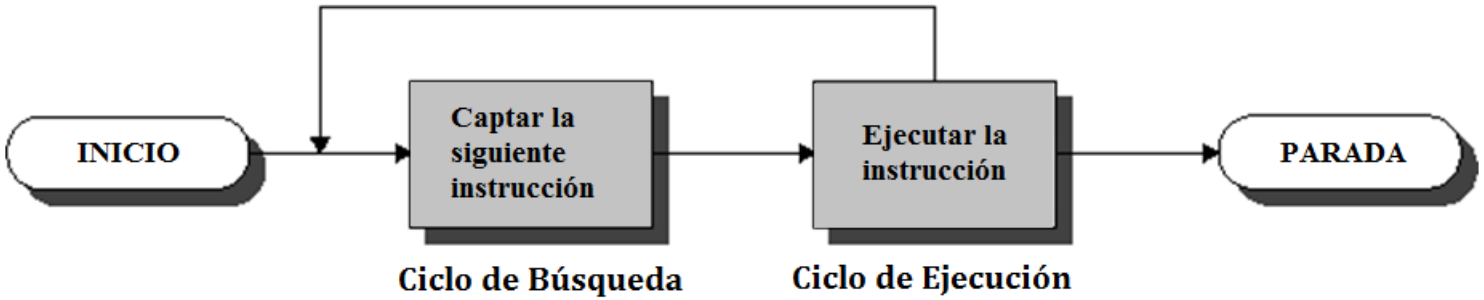
Ciclo de instrucción

El ejemplo permite visualizar que para la ejecución del programa, la CPU realiza una secuencia regular, que se repite hasta finalizar el mismo, denominada *ciclo de instrucción*:

- Traer la instrucción desde la memoria
- Realizar el procesamiento correspondiente a la instrucción específica.

Esto permite subdividir cada instrucción en al menos dos partes básicas:

1. Un ciclo de *búsqueda* o *captación* (*fetch*).
2. Un ciclo de *ejecución*.



Ciclo de búsqueda

Microinstrucción	Comentario
• t_1: MAR ← PC	La dirección de la instrucción está en el PC. Se copia al MAR (se coloca en el bus de direcciones).
• t_2: MBR ← M[MAR],	La Unidad de Control genera un comando READ. El resultado (un dato desde la memoria) aparece en el bus de datos. El dato se copia al MBR desde el bus de datos.
PC ← PC + 1	El PC se incrementa en 1 (en paralelo con la búsqueda del dato desde la memoria).
• t_3: IR ← MBR	El dato (instrucción) se mueve del MBR al IR. El MBR queda libre para nuevas búsquedas de datos.

Ciclo de búsqueda

Como se describió, el ciclo de búsqueda contiene cuatro microoperaciones. Sin embargo, requiere tres ciclos de reloj. Esto se debe a que MBR ← M[MAR] y PC ← PC+1 se ejecutan en el mismo ciclo de reloj. También podría ser:

- t1: MAR ← PC
- t2: MBR ← M[MAR]
- t3: PC ← PC +1, IR ← MBR

En general, y siempre que sea posible, es deseable agrupar micro operaciones en un mismo ciclo. Para ello se debe verificar que:

1. Se siga la secuencia apropiada
   - ✓ MAR ← PC debe preceder a MBR ← M[MAR]
2. No se produzcan conflictos:
   - ✓ No se debe leer ni escribir el mismo registro al mismo tiempo.
   - ✓ MBR ← M[MAR] & IR ← MBR no deben hacerse en el mismo ciclo.

Ciclo de ejecución

A continuación del ciclo de búsqueda, sigue el de ejecución. En este, la secuencia de microoperaciones y la cantidad de ciclos de reloj utilizados *depende de cada instrucción*. Por ejemplo, para los casos vistos:

1. La carga del dato contenido en memoria al registro B:
   - t_1: MAR ← IR(11:0) La dirección de memoria se carga al MAR
   - t_2: MBR ← M[MAR] El contenido de la memoria se copia en MBR
   - t_3: R1 ← MBR El contenido de MBR se copia en R1

2. La suma del dato contenido en el registro R1 con el proveniente de la memoria:
   - t_1: MAR ← IR(11:0) La dirección de memoria se carga al MAR
   - t_2: MBR ← M[MAR] El contenido de la memoria se copia en MBR
     T2 ← R1 El contenido de R1 se copia a T2
   - t_3: T1 ← MBR El contenido de MBR se copia a T1
   - t_4: R1 ← T1 + T2 La ALU hace la suma y guarda el resultado en R1

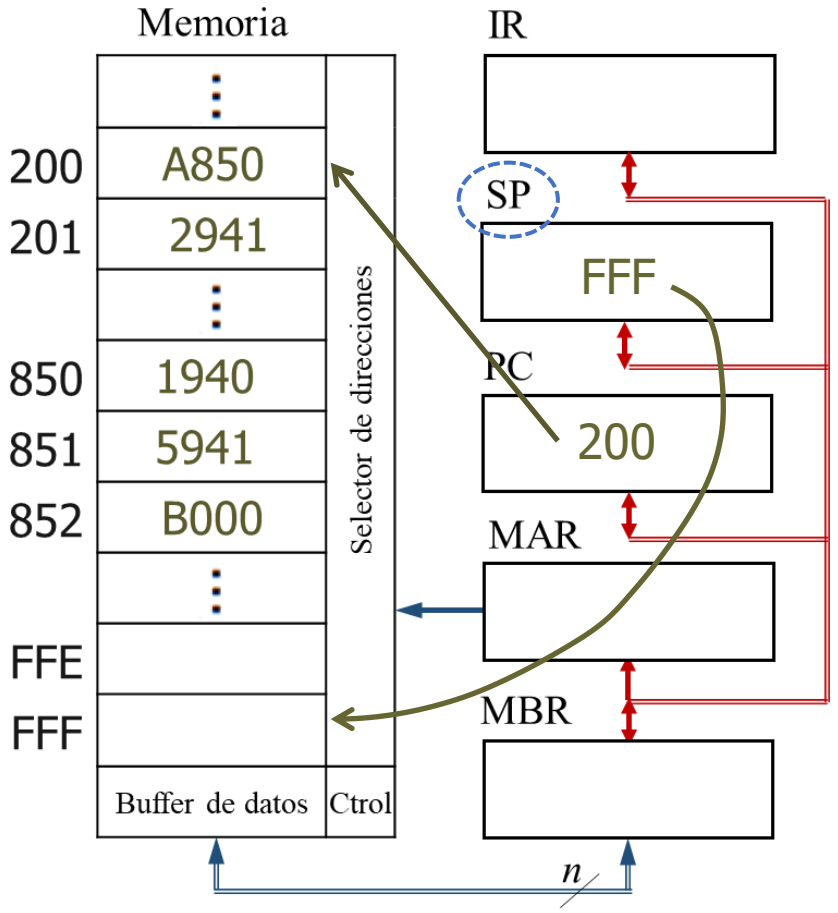
Subrutinas

Las subrutinas son “programas dentro de programas”. Para su operación, la mayoría de los procesadores operan con una pila (*stack*) en memoria, y el registro específico SP (puntero de pila).

El contenido del SP apunta al tope de la pila, una estructura lógica en memoria del tipo LIFO (*“last in – first out”*).

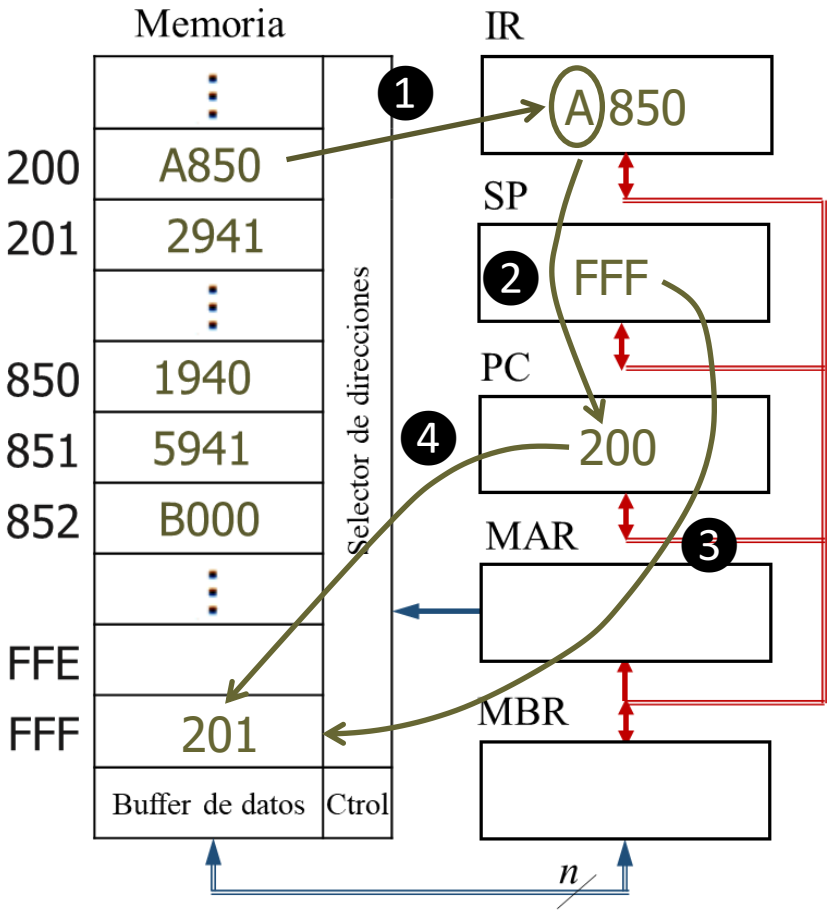
Para este ejemplo, se supone un programa principal que se está ejecutando en la posición 200_{16}.

En los primeros cuatro bits de la instrucción aparece el valor $1010_2 = A_{16}$, que es el código de llamada a la subrutina ubicada en la posición de memoria indicada por los doce bits siguientes.

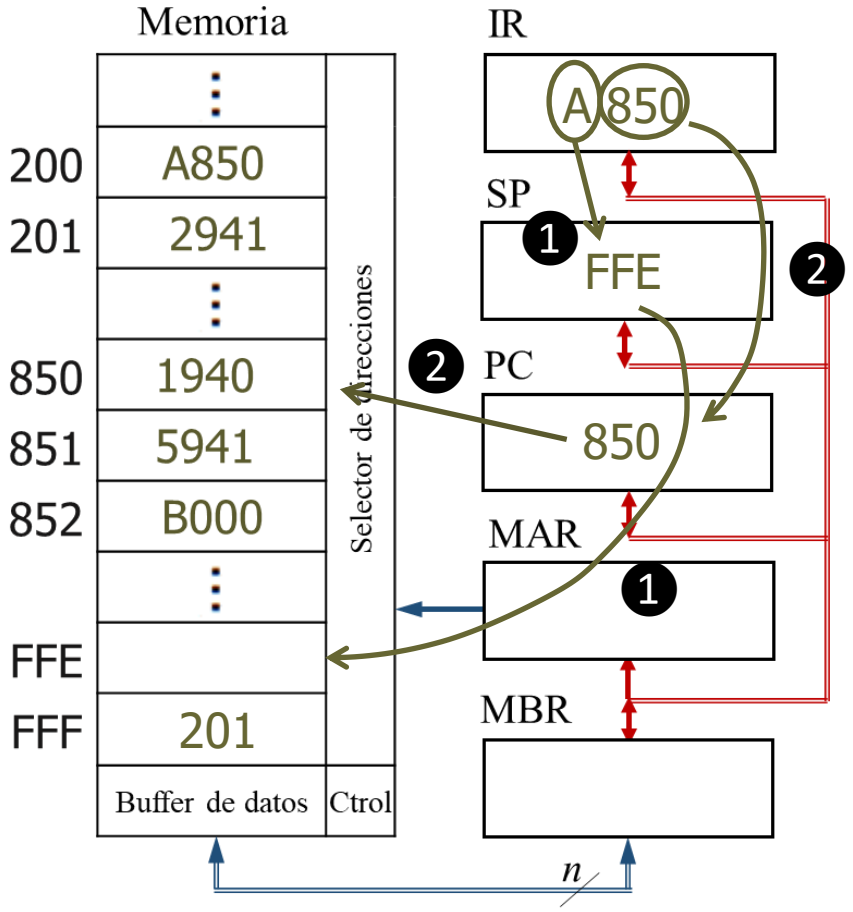


Subrutinas

Al ejecutarse el llamado, el PC anterior se incrementa (PC ← PC+1) y se salva en la pila, en la dirección apuntada por el SP.

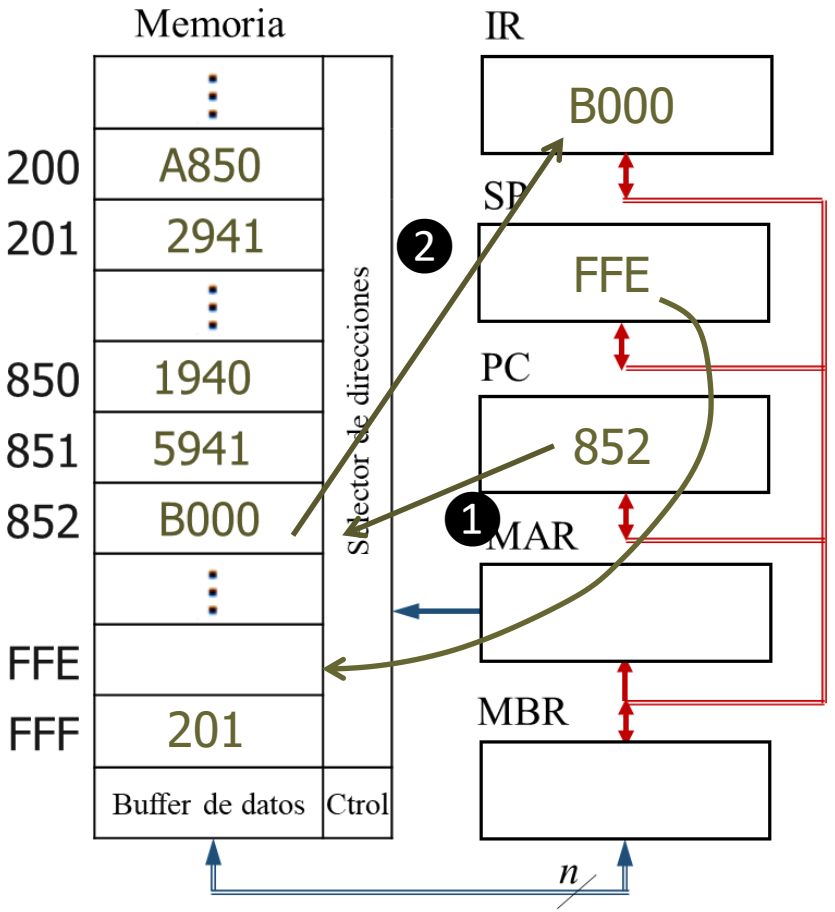


Se decrementa el SP, y el PC se carga con la dirección de la subrutina a ejecutar:

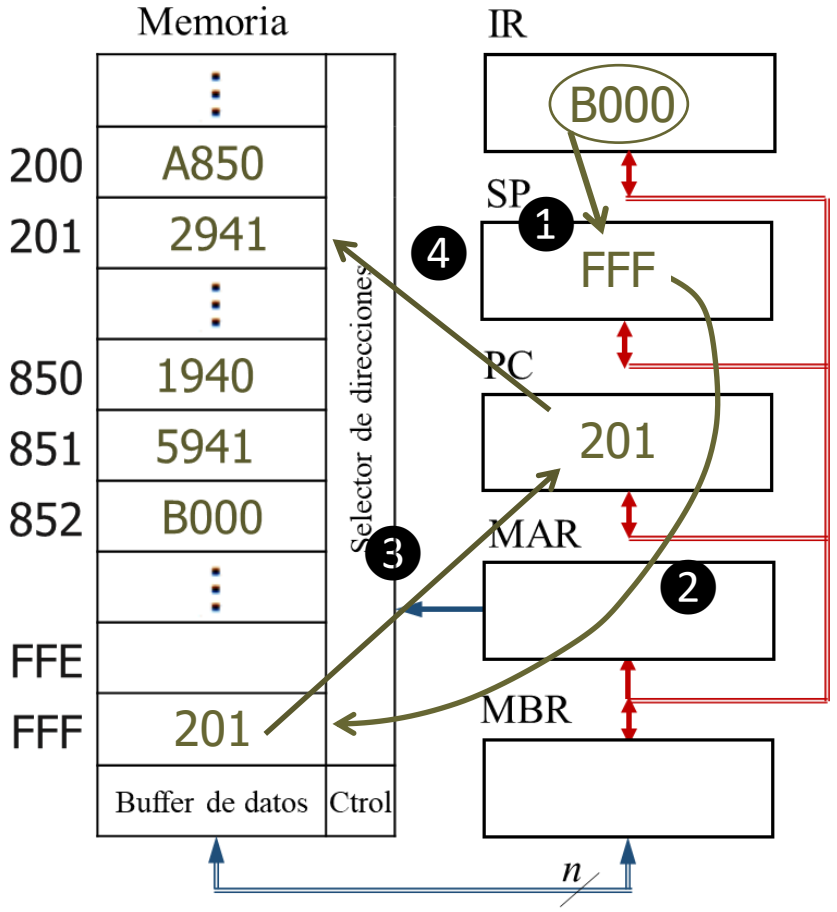


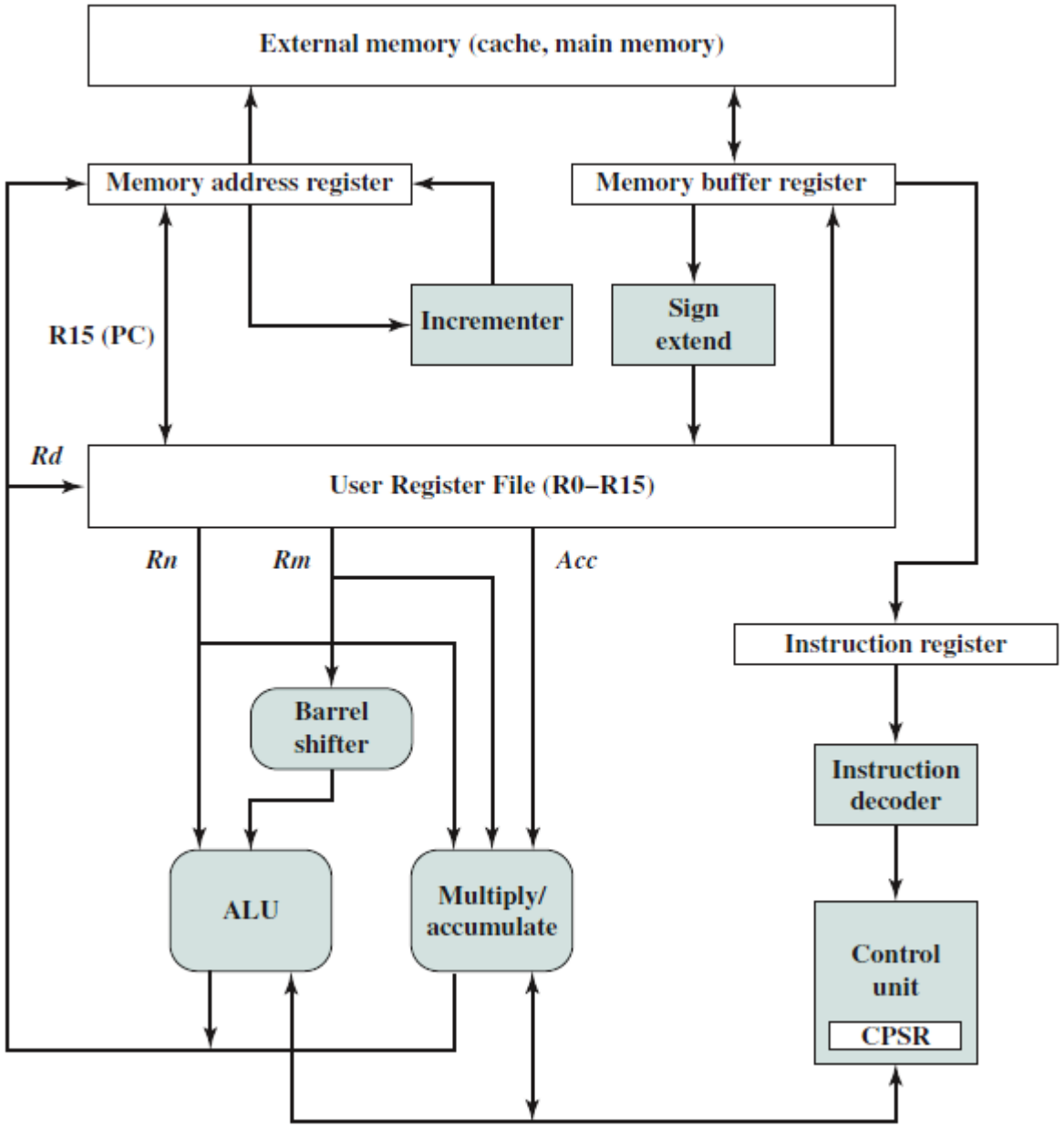
Subrutinas

En el final de la subrutina, está la instrucción de retorno ($B000_{16}$).



El SP se incrementa (SP ← SP+1), se restaura el PC anterior, y prosigue la rutina original:





Arquitectura básica ARM

(Modo Usuario)

Las instrucciones de procesamiento de datos ARM tienen dos registros de origen, Rn y Rm, y un único registro de resultado o destino, Rd. Los valores del registro fuente ingresan a la ALU o a una unidad multiplicadora separada que utiliza un registro adicional para acumular resultados parciales. El procesador ARM también incluye una unidad de hardware que puede cambiar o rotar el valor Rm antes de que ingrese a la ALU (*barrel shifter*)

Son visibles dieciséis registros de 32 bits numerados y uno o dos registros de estado del programa, para un total de 17 o 18 registros visibles por software.
Registros de uso general: R0 a R12
R13: puntero de pila (SP).
R14: registro de enlace (LR) y se utiliza para contener direcciones de retorno de subrutina
R15: contador de programa (PC).

Repaso conceptual

1. ¿Qué es una microinstrucción?
2. En un sistema procesador, ¿qué es la ruta (o camino) de datos?
3. ¿Qué es una instrucción máquina y qué información debe contener?
4. Describa las funciones de los tres bloques componentes de un sistema computador, y enumere los buses mediante los que se interconectan.
5. ¿Cuáles son los principales registros de uso específico de una CPU y cuál es la función de cada uno?
6. ¿Cuáles son las funciones de la Unidad de Control y cuáles son sus entradas y salidas?
7. ¿Cuáles son los ciclos básicos que constituyen una instrucción y qué características tienen?
8. ¿Cuál es el concepto de subrutina y cuales son las instrucciones específicas que utilizan?

Lecturas recomendadas

- Stallings, Williams - Organización y Arquitectura de Computadoras - 5º Ed. - Prentice Hall. Año 2000. *– Capítulo 3 (3.1 y 3.2)*