Unidad 4

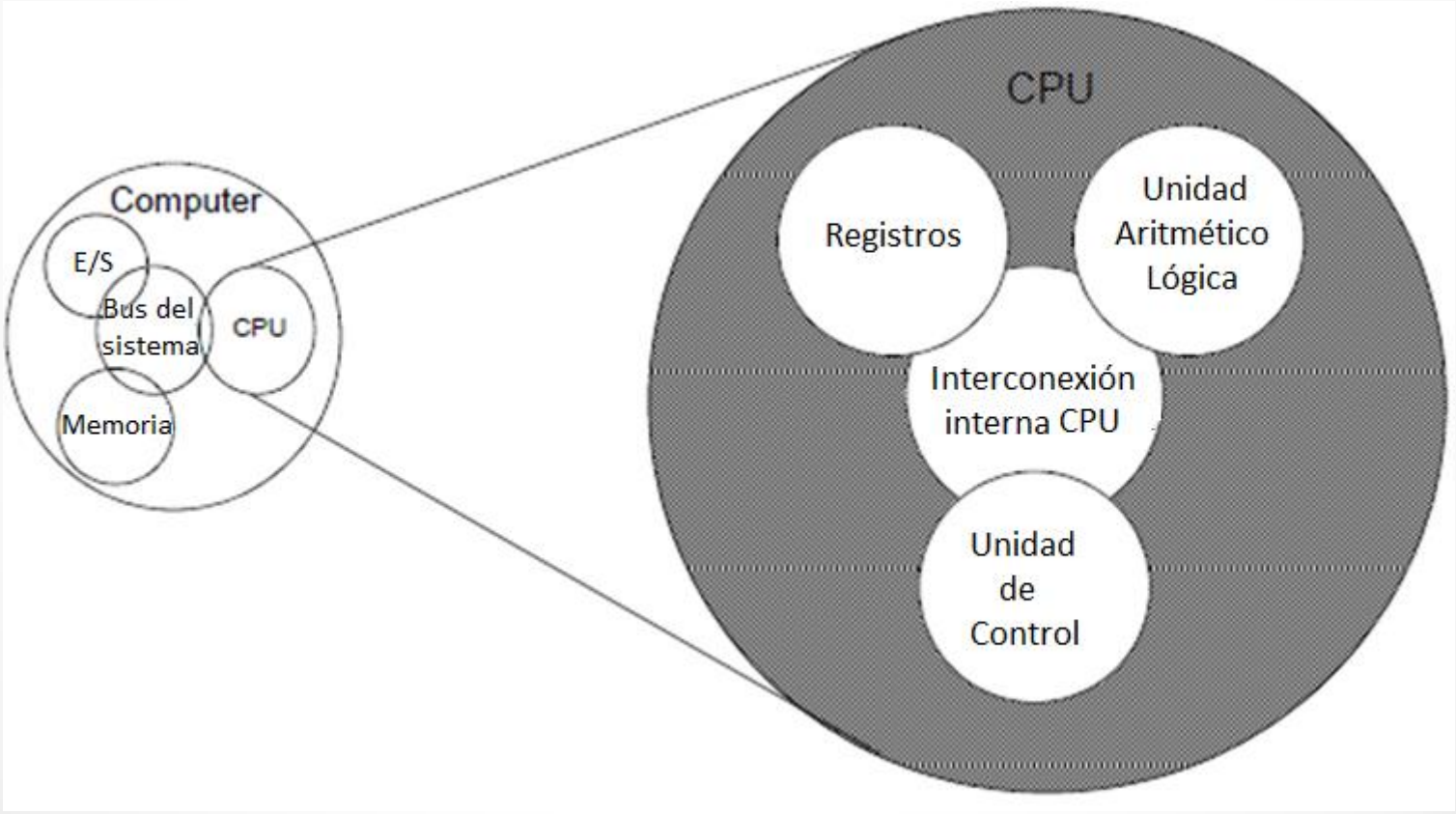
Arquitectura general y organización funcional de computadoras. Descripción de los distintos bloques (Memoria, ALU, Unidad de control y Unidad de E/S). Proceso de búsqueda y ejecución de las instrucciones. Interrupciones. Estructuras de interconexión. Buses.

Componentes de la Unidad Central de Proceso (CPU)

Una CPU está compuesta por:

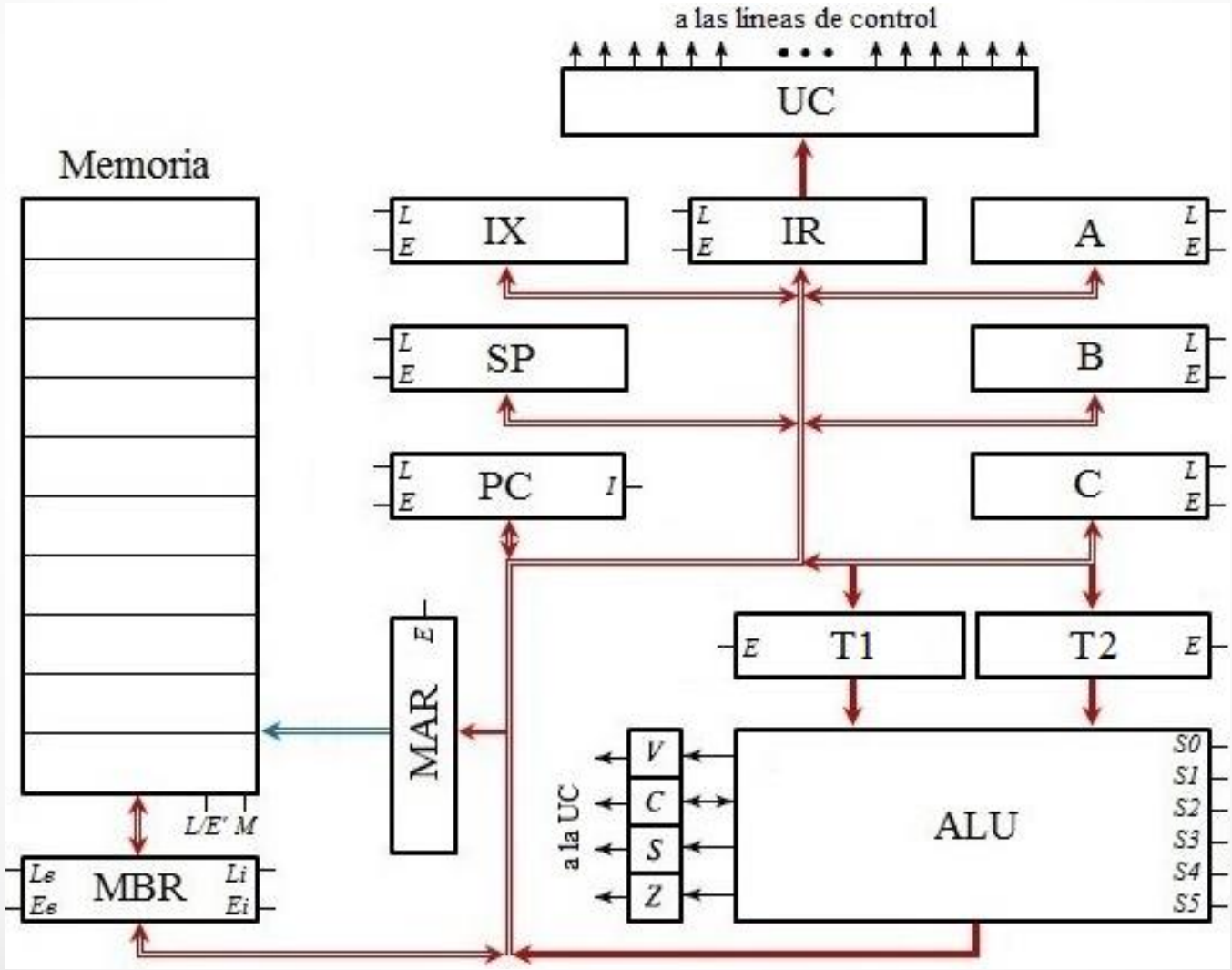
- Un conjunto de ***Registros*** de uso *general,* en el que se almacenan temporalmente los datos que están siendo procesados, y otros de uso *específico*, con información utilizada para la interrelación entre los componentes);
- Una ***Unidad Aritmético Lógica (ALU),*** en la que se realizan las operaciones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

Componentes de la Unidad Central de Proceso (CPU)



Arquitectura general

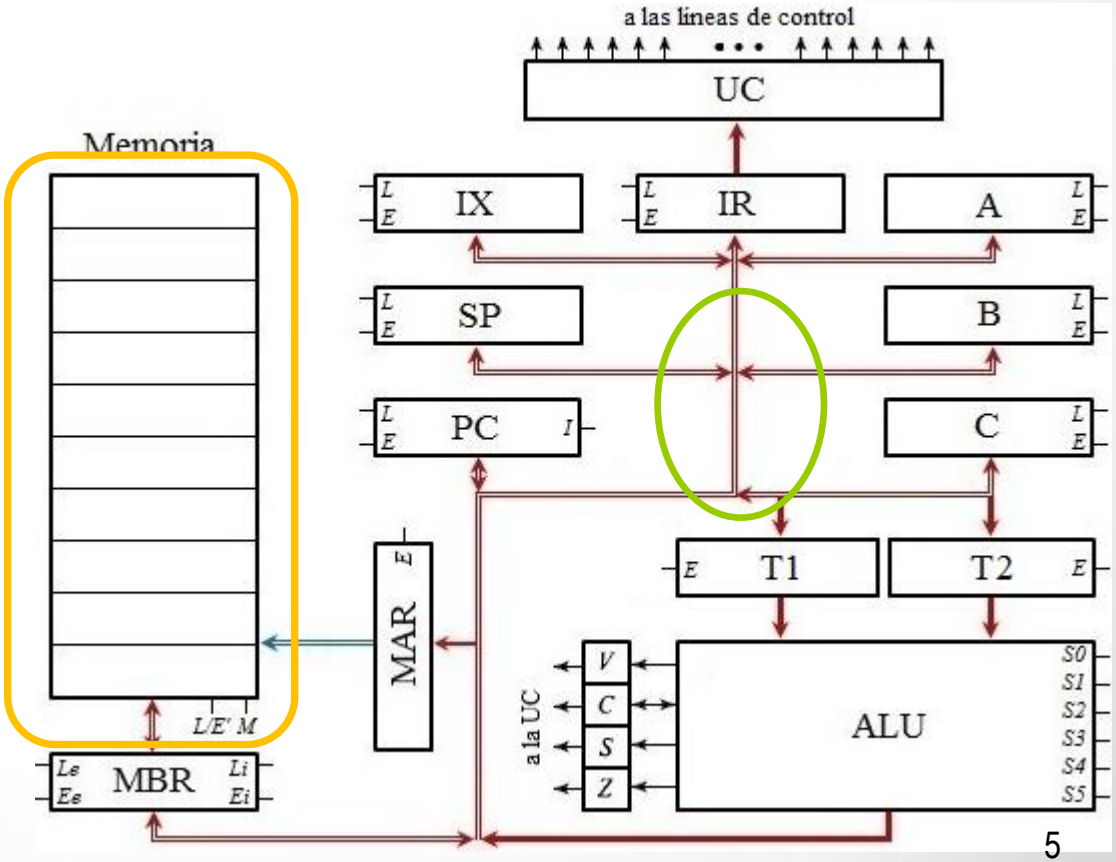
Una posible arquitectura para un computador simple es:



Arquitectura general

En el diagrama anterior, se visualizan:

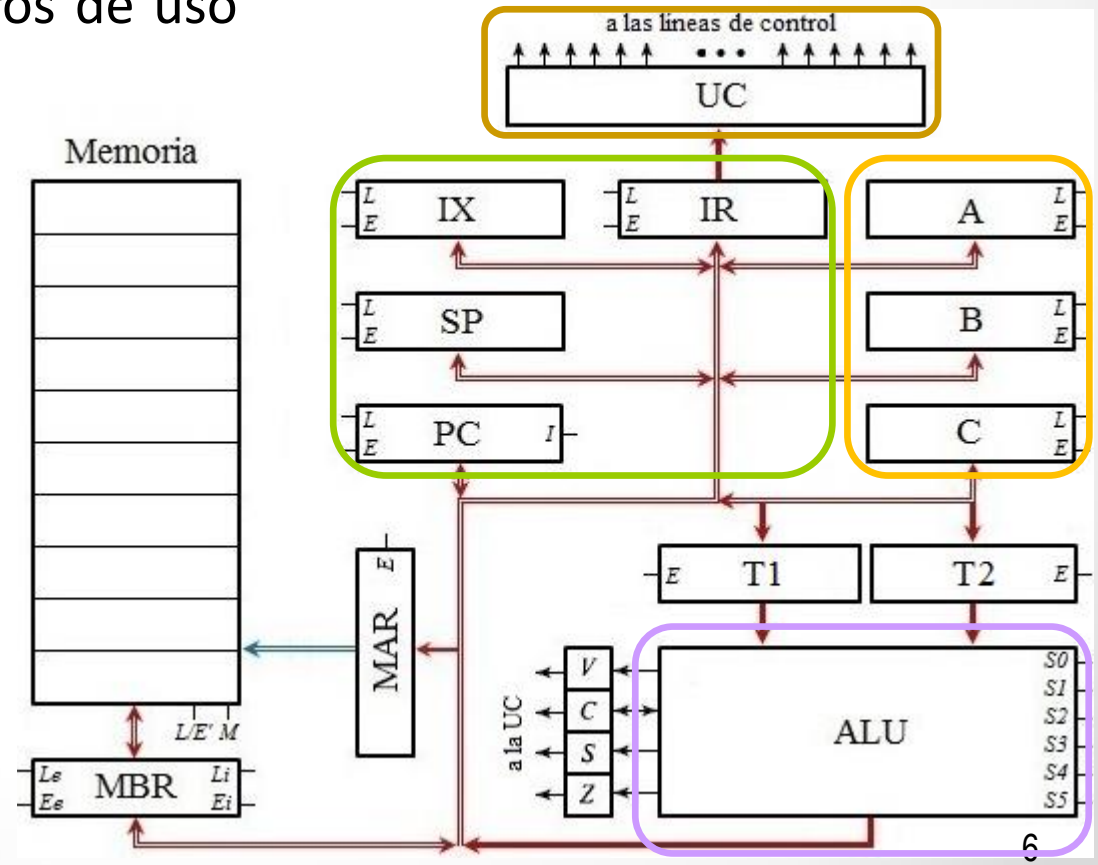
1. Estructuras de interconexión: buses de datos (en rojo), con líneas bidireccionales (trazo doble) y unidireccionales (trazo simple), y el bus de direcciones (unidireccional, en azul).
2. Memoria principal: representada como un array de *n* posiciones, con sus buses de datos y direcciones, y líneas de control: selección (*M*) y lectura/ escritura (L/E´).
3. La Unidad Central de Proceso (CPU), compuesta por los restantes elementos



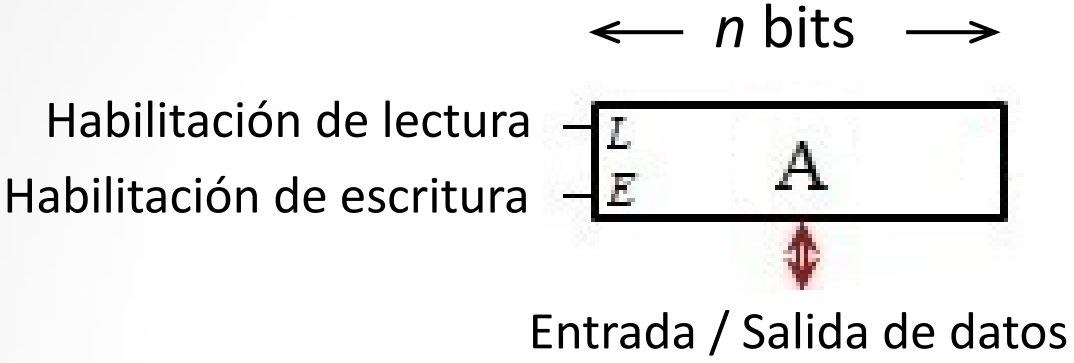
Arquitectura general

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 (A, B y C), y un conjunto de registros de uso específico (PC, SP, IX e IR), todos con sus entradas de control.
3. La Unidad de Control (UC), que recibe datos del IR y de los flags, y genera las salidas de control para todos los restantes elementos.



CPU: Registros



- Compuestos por *n* biestables (tipo D)
- Con una línea de datos bidireccional (entrada/salida)
- Con dos líneas de habilitación: lectura y escritura (o bien: habilitación del registro y lectura/escritura´)
- Pueden incluir otras funciones

CPU: Registros

La función de los registros de uso específico es:

- **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.
- **Registro de Instrucción** (IR, *Instruction Register*): contiene el valor binario de la instrucción en curso, para ser decodificado y secuenciado por la UC.
- **Puntero de pila** (SP, *Stack pointer*): apunta a la dirección tope de la pila en memoria (utilizada en la ejecución de subrutinas e interrupciones).
- **Registro Índice** (IX, *Index Registrer*): utilizado en operaciones que requieren indexación, como uso de tablas.

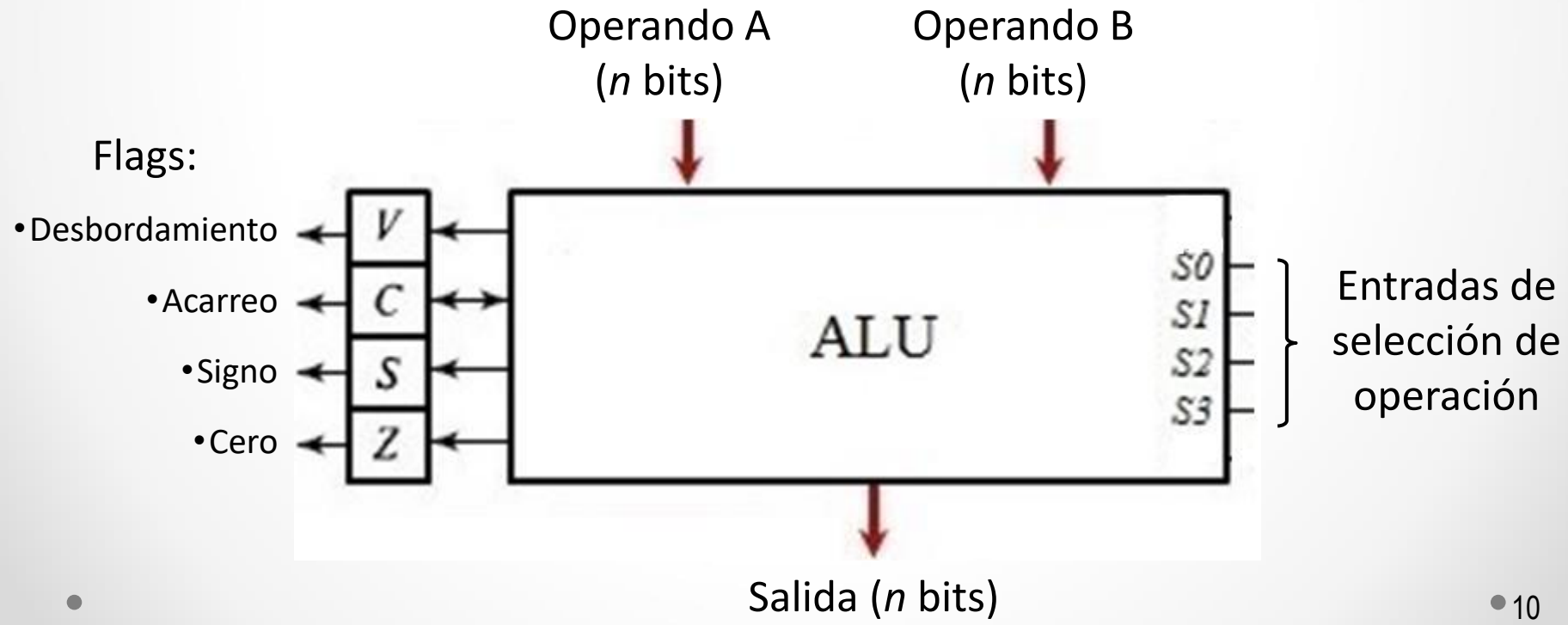
CPU: Registros

Además de estos, existen otros registros no visibles al programador:

- **Registro de Direcciones** (MAR, *Memory Adress Register*), que contiene la dirección actual de memoria;
- **Registro de Datos** (MBR, *Memory Buffer Register*) que almacena el dato a ser escrito en la memoria o el leído de ésta),
- **Registros temporales** (T1 y T2), que almacenan los datos a operar en la ALU.

CPU: Unidad Aritmética Lógica

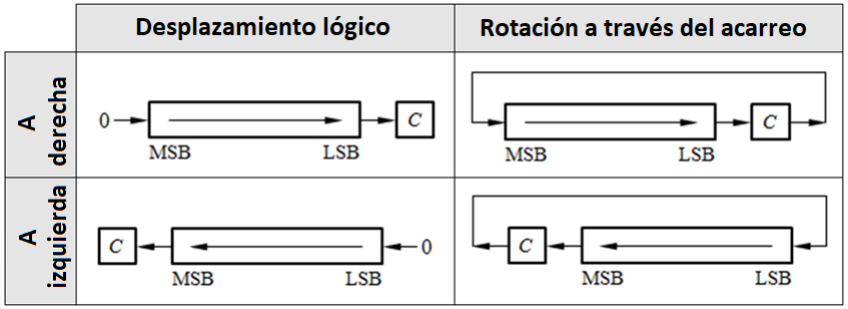
Esencialmente, la ALU es un circuito *combinacional* que realiza operaciones aritméticas y lógicas entre dos operandos A y B (o sobre uno de ellos), de acuerdo a la combinación de valores presentes en sus entradas de selección, presentado el resultado en su salida:



CPU: Unidad Aritmética Lógica

S_3	S_2	S_1	S_0	Micro operación	Descripción
0	0	0	0	A+1	Incremento de A
0	0	0	1	A-1	Decremento de A
0	0	1	0	A+B	Suma aritmética de A y B
0	0	1	1	A+B+C_i	Suma aritmética de A y B y el acarreo de entrada
0	1	0	0	A+B'+C_i	Suma aritmética de A y el complemento de B y el acarreo de entrada
0	1	0	1	A+B'+1	Suma aritmética de A y el complemento de B más 1 (A menos B)
0	1	1	0	A'	Complemento de A
0	1	1	1	A or B	Suma lógica de A y B
1	0	0	0	A and B	Producto lógico de A y B
1	0	0	1	A oexc B	Or exclusiva de A y B
1	0	1	0	LSL A	Desplazamiento aritmético a la izquierda de A (A x 2)
1	0	1	1	LSR A	Desplazamiento aritmético a la derecha de A (A / 2)
1	1	0	0	ROL A	Rotación a la izquierda a través del acarreo de A
1	1	0	1	ROR A	Rotación a la derecha a través del acarreo de A
1	1	1	0	CLR A	Puesta a cero de A
1	1	1	1	SET A	Puesta a uno de A

Operaciones de la ALU (de acuerdo a los valores presentes en las entradas S_i):



Banderas (flags) de condición

En las operaciones condicionales, es necesario que el proceso vaya a uno u otro estado en función del resultado de la última operación de la ALU. Las más comunes son:

- Una salida ***C*** que es uno si el acarreo de salida es uno, y cero en caso contrario,
- Una salida ***S*** que es cero si el valor de la ALU es positivo, y uno si es negativo.
- Una salida ***Z*** que es uno si el valor de la ALU es cero, y uno en caso contrario.
- Una salida ***V*** que es uno si el resultado de la última operación realizada produce un desbordamiento o sobrecapacidad (para ocho bits, esto es si el resultado es mayor que 255 o menor que -256). En caso contrario, su valor es cero.

Las salidas descriptas se implementan con lógica combinacional y se almacenan en biestables tipo D (deben memorizarse para utilizarse en siguientes microoperaciones)

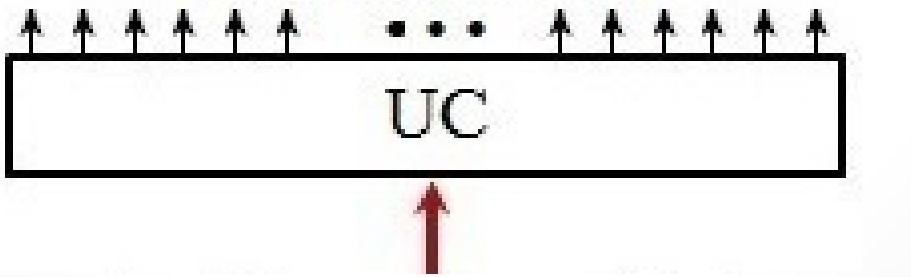
CPU: Unidad de Control

Esencialmente, la UC es un circuito secuencial, que genera:

1º) La secuencia de señales de control para obtener de la memoria la instrucción a ejecutarse;

2º) Recibe en su entrada el código de la operación que corresponde a la instrucción a ejecutarse, y genera una secuencia de señales de control propia para esa instrucción:

Salidas: líneas de control (de todos los restantes elementos)



Entradas codificadas: Código de Operación (desde el Registro de Instrucción)

Funcionamiento del computador

Sobre la estructura descripta, el computador ejecuta operaciones complejas mediante un encadenamiento lógico (*programa*) de un repertorio finito de operaciones simples, denominadas ***instrucciones***.

Una instrucción es, formalmente, un código binario que se *interpreta* por parte del procesador y da lugar a una serie predeterminada de operaciones elementales o ***microoperaciones***.

A su vez, una microoperación es una transferencia entre registros o entre registros y 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.

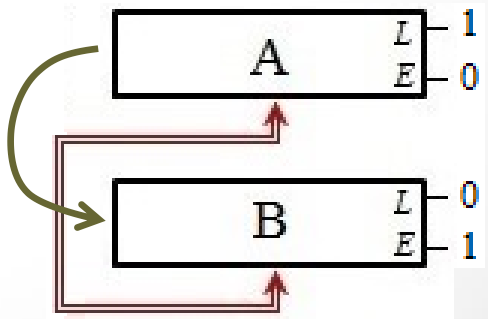
Funcionamiento del computador

Las microoperaciones se describen mediante la notación de Nivel de Transferencia de Registro (RTL, *Register Transfer Level*), que utiliza un conjunto de expresiones y sentencias particular. Así, por ejemplo, la transferencia de datos de un registro a otro será:

$$B \leftarrow A$$

y significa que el contenido del registro A (fuente) se copia en el B (destino). El contenido del registro fuente no cambia

A nivel circuital, la transferencia implica la activación de la señal de lectura de A y de escritura de B durante un pulso de reloj.



Ejecución de un programa

El siguiente ejemplo muestra la ejecución de un fragmento de programa con tres instrucciones:

- Cargar en el registro B el contenido de la posición de memoria 940_{16}
- Sumar el contenido de la posición de memoria 941_{16} al registro B y guardar el resultado en B
- Almacenar el resultado almacenado en el registro B en la posición de memoria 941_{16}

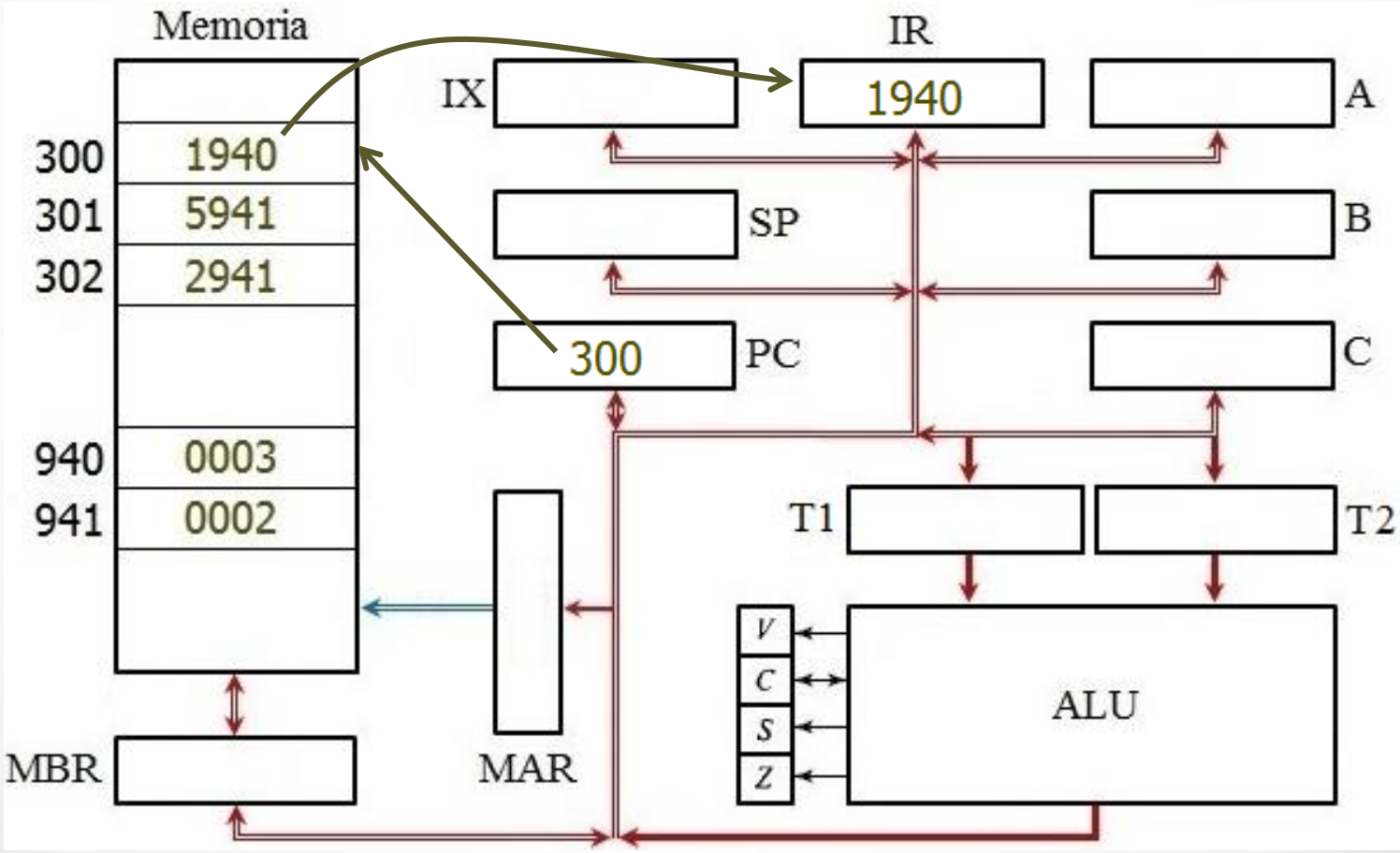
Se considera que cada posición de memoria almacena 16 bits. Los primeros 4 bits indican la operación a realizar, y los siguientes 12 bits indican una dirección de memoria.

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

Ejecución de un programa

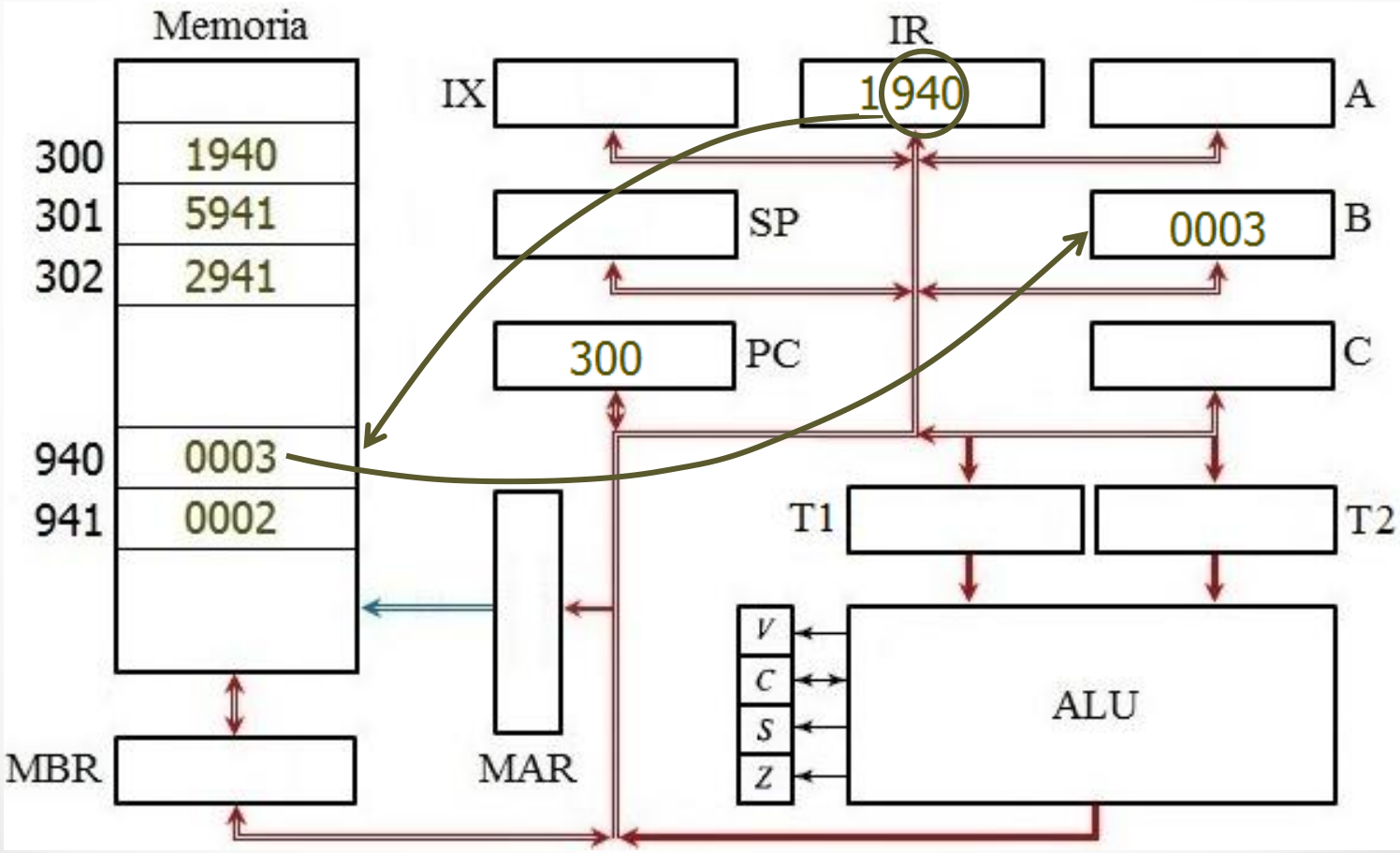
El contador de programa (PC) contiene 300_{16} como la dirección de la primera instrucción.

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



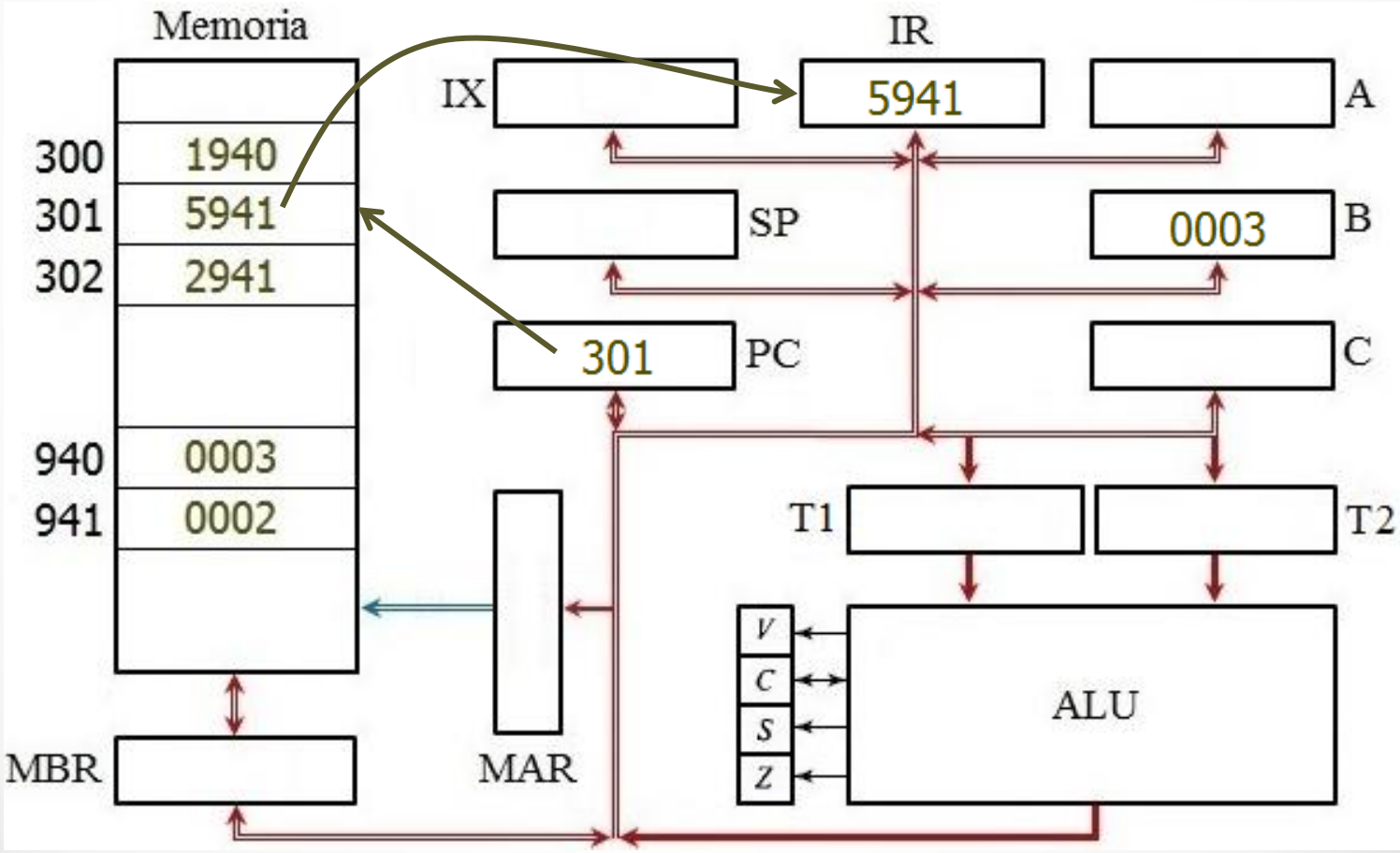
Ejecución de un programa

Los primeros 4 bits en IR (1_{16}) indican que el registro B 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}



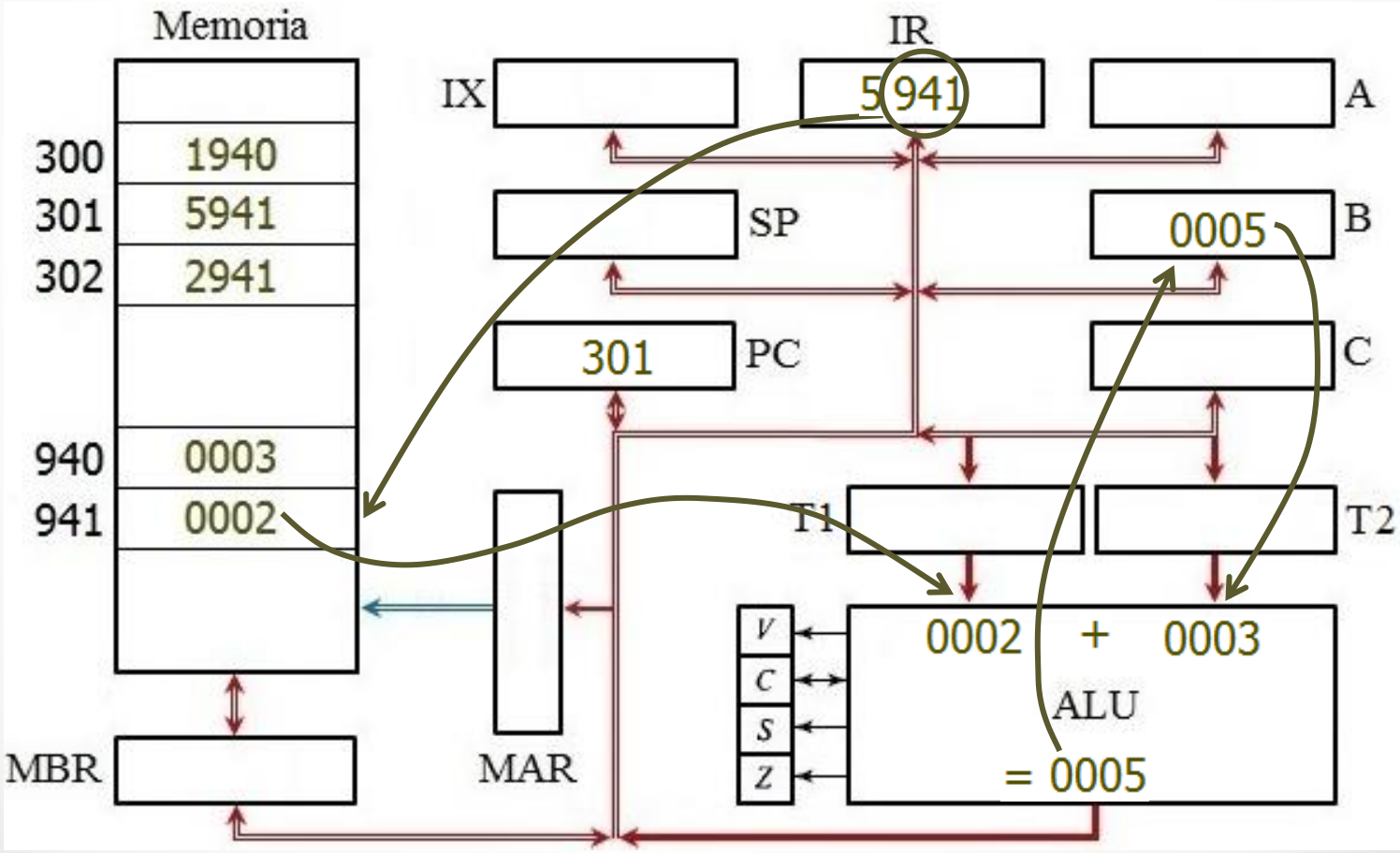
Ejecución de un programa

Se incrementa el contador de programa y se busca la siguiente instrucción en la dirección 301_{16}



Ejecución de un programa

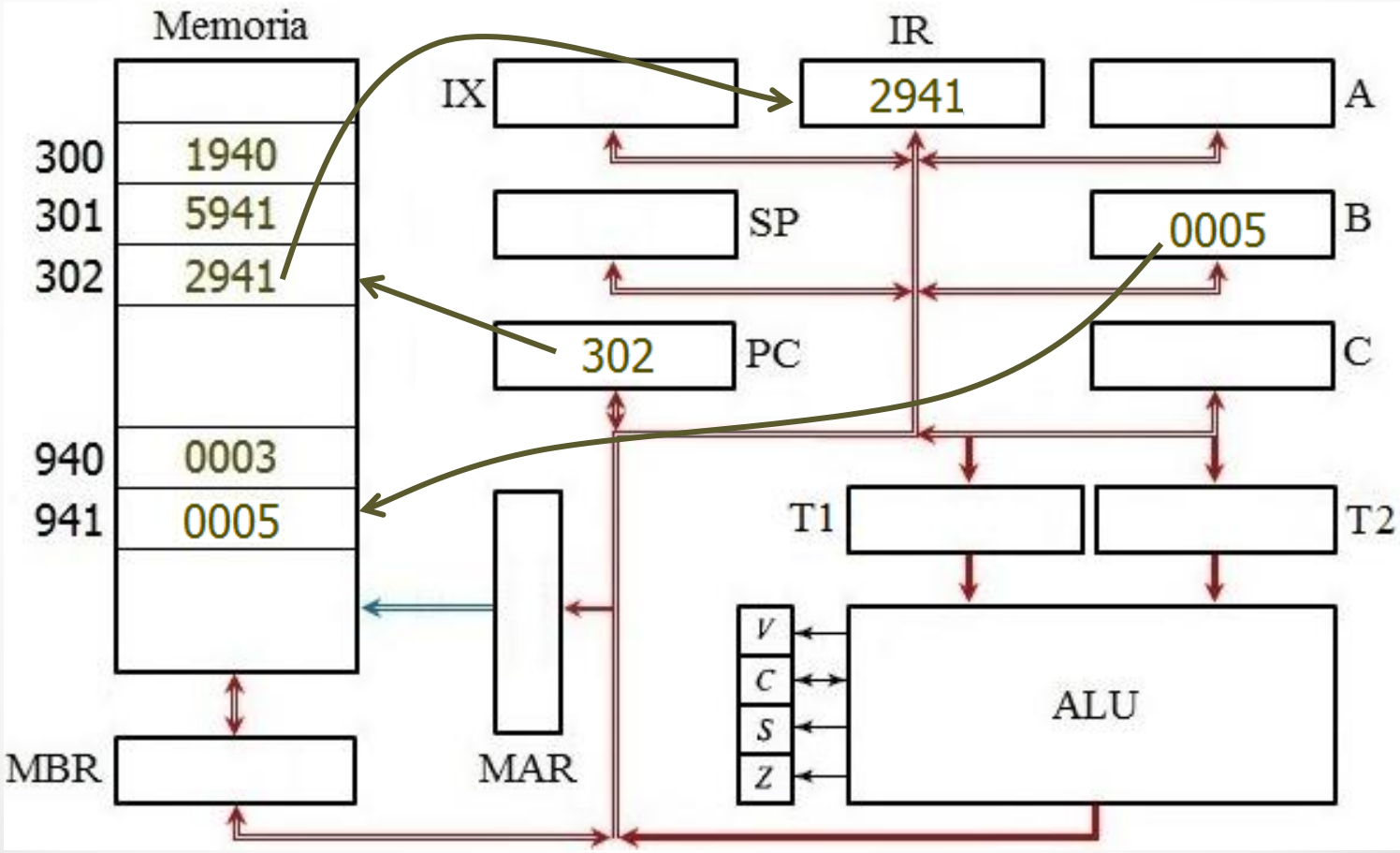
El 5_{16} en IR indica que se debe sumar el contenido de una dirección de memoria especificada -en este caso 941_{16} -, con el contenido del registro B y almacenar el resultado en el registro B



Ejecución de un programa

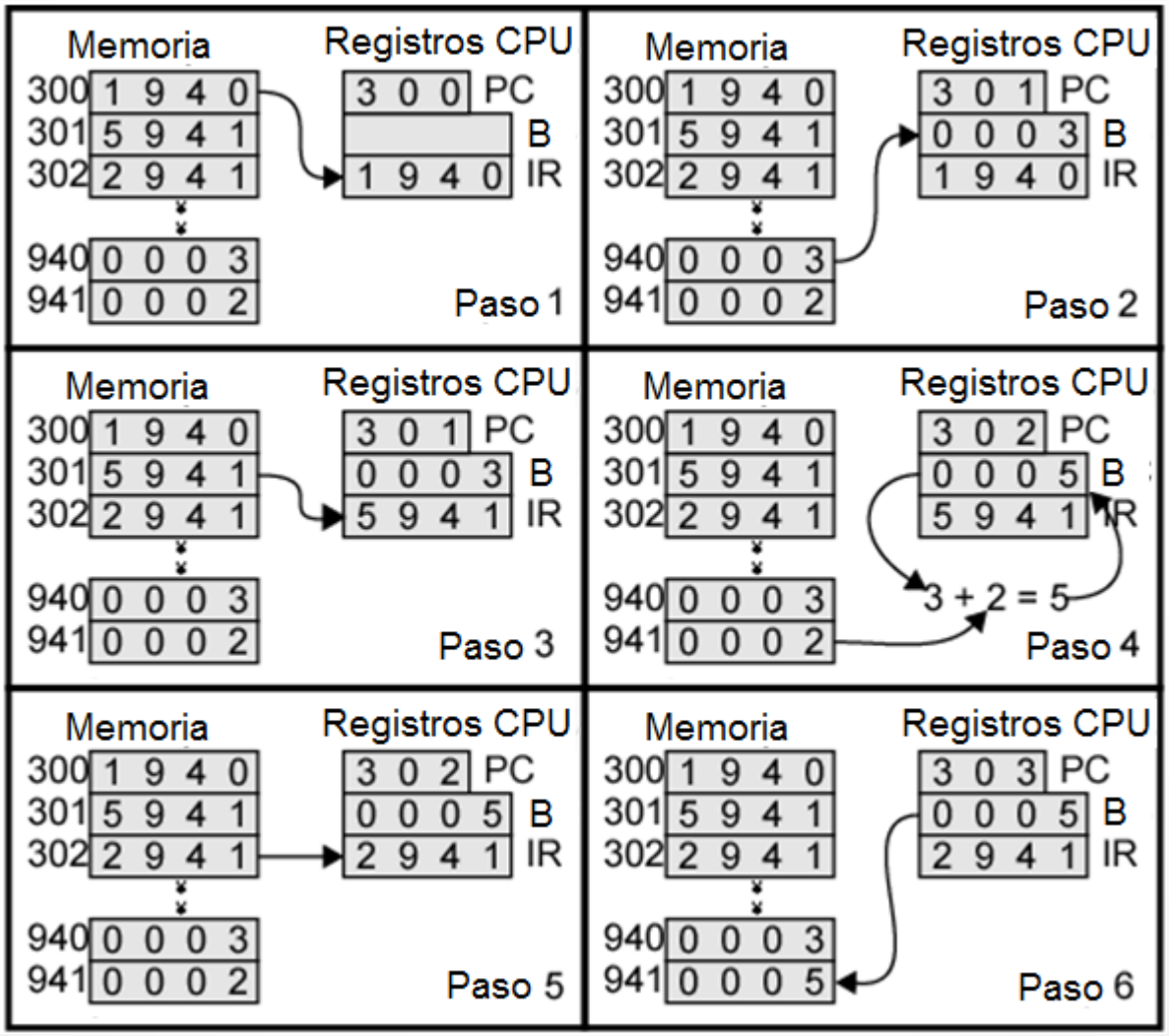
Se incrementa el PC y se busca la siguiente instrucción en 302_{16}

El 2_{16} en IR indica que se debe almacenar el contenido del registro B en una dirección especificada -en este caso 941_{16} -



Ejecución de un programa

Los pasos vistos se pueden resumir en el siguiente cuadro:



Ciclo de instrucción

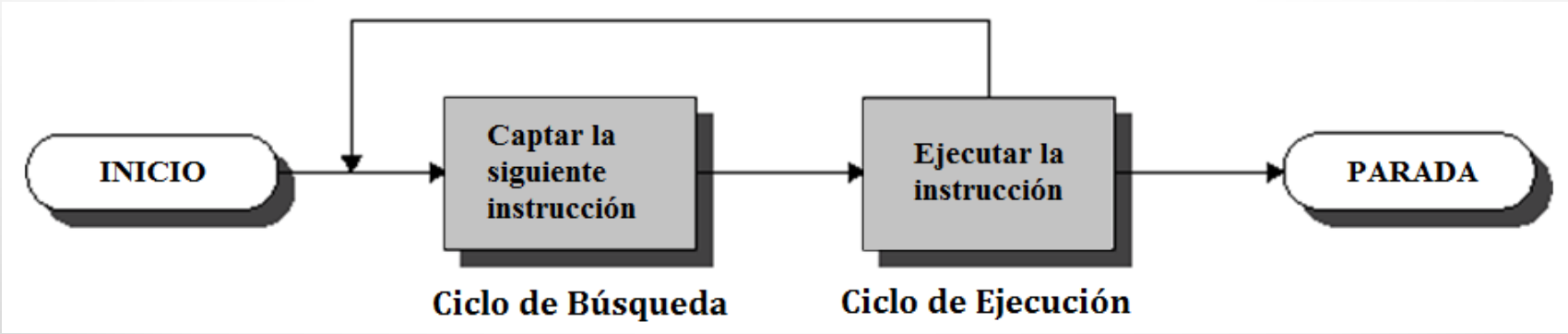
En el ejemplo dado, se representaron genéricamente los movimientos de datos, prescindiendo de las microoperaciones y registros intermedios utilizados.

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

- Trae la instrucción desde la memoria
- Realiza el procesamiento correspondiente a la instrucción específica.

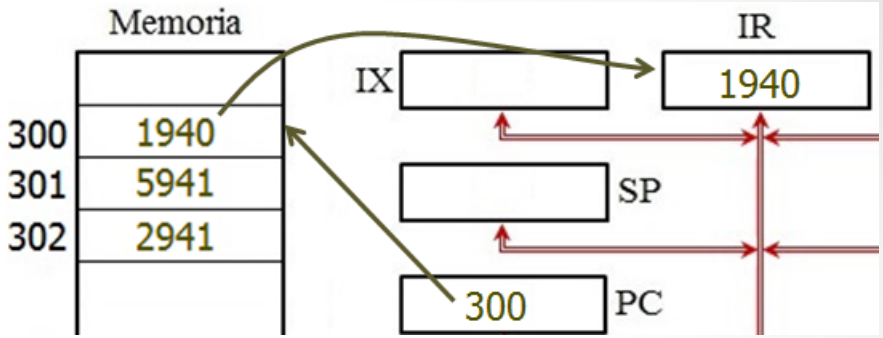
Esto permite subdividir cada instrucción en al menos dos partes básicas:

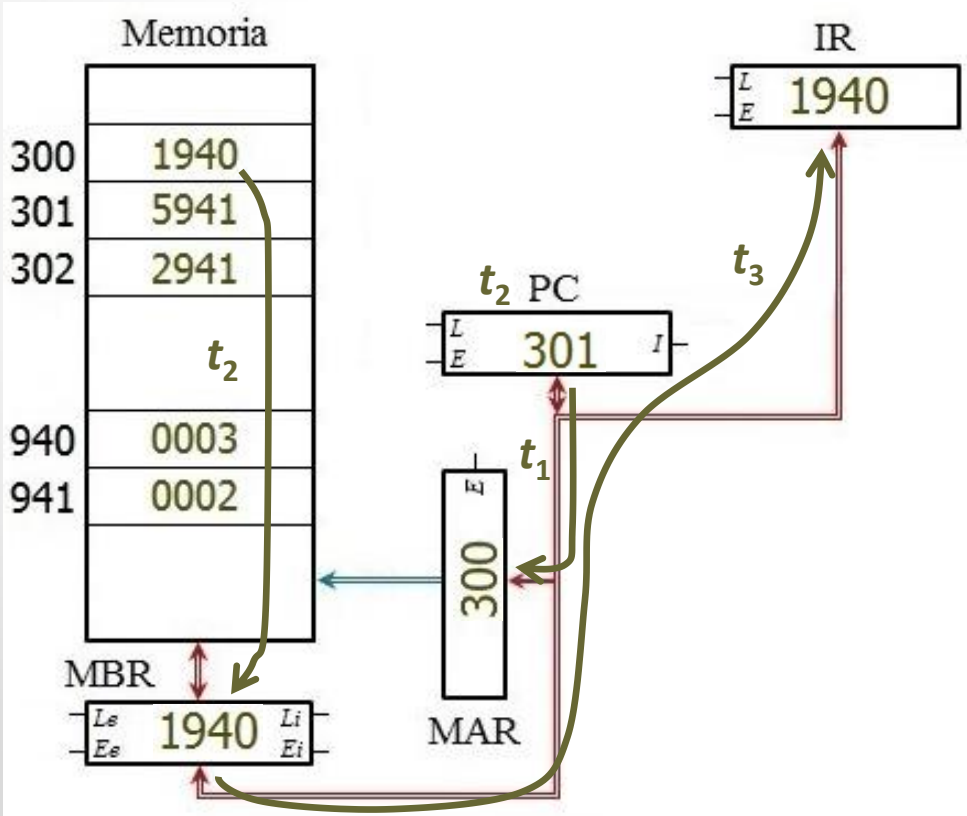
1. Un ciclo de *búsqueda* o *captación* (*fetch*).
2. Un ciclo de *ejecución*.



Ciclo de búsqueda

El ciclo de búsqueda común a todas las instrucciones se corresponde al proceso de carga de la instrucción en el IR:





A nivel de microoperación, se describe por:

- t_1: MAR ← PC
- t_2: MBR ← (memoria)
  PC ← PC +1
- t_3: IR ← MBR

donde cada t_x se corresponde a un ciclo de reloj.

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 ← mem[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 ← (memoria) y PC ← PC+1 se ejecutan en el mismo ciclo de reloj. También podría ser:

- t1: MAR ← (PC)
- t2: MBR ← (memoria)
- 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←(memoria)
2. No se produzcan conflictos:
   - ✓ No se debe leer ni escribir el mismo registro al mismo tiempo.
   - ✓ MBR ← (memoria) & 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:

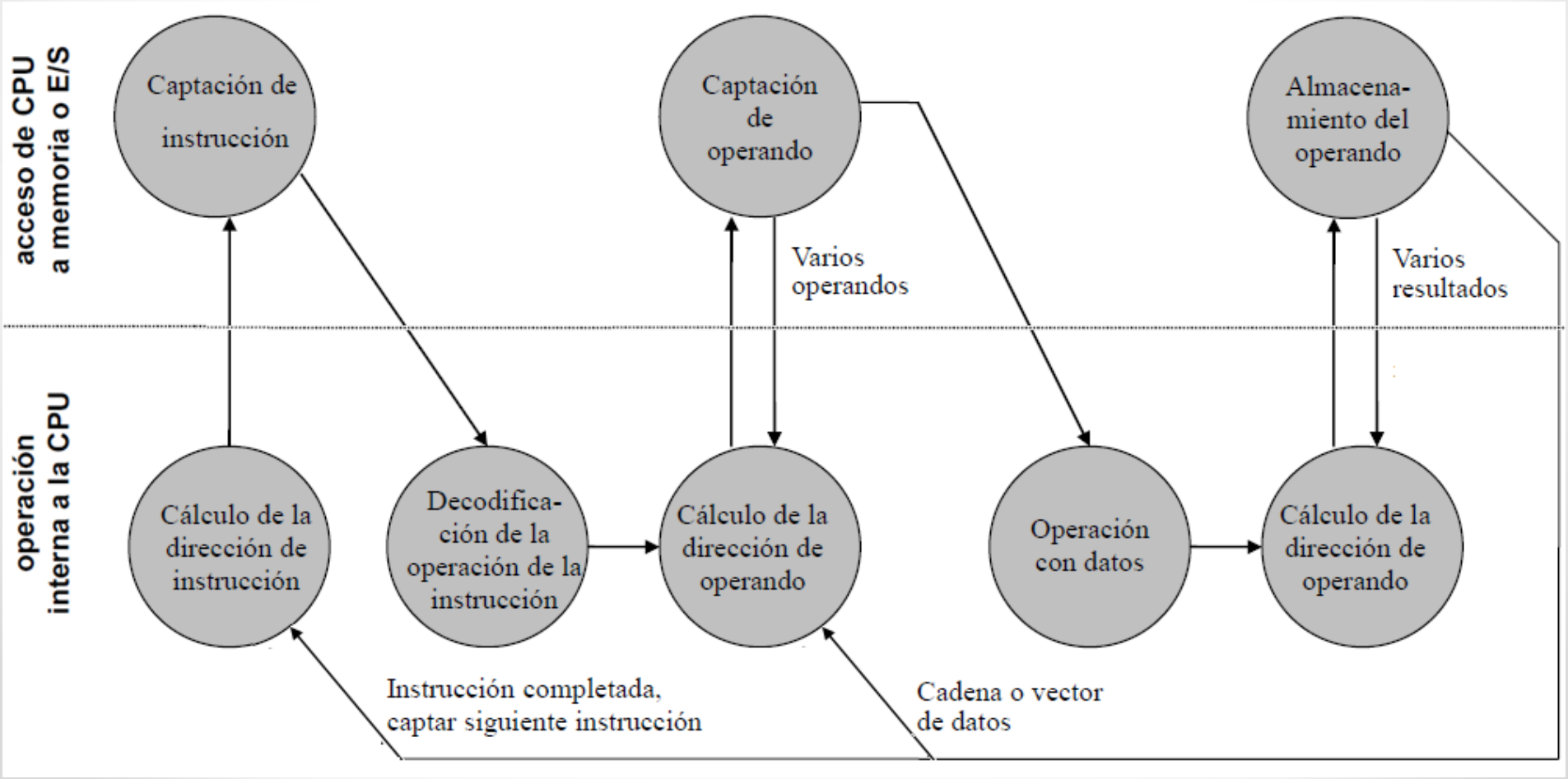
- t_1: MAR ← ($IR_{dirección}$) La dirección de memoria se carga al MAR
- t_2: MBR ← mem[MAR] El contenido de la memoria se copia en MBR
- t_3: B ← MBR El contenido de MBR se copia en B

2. La suma del dato contenido en el registro B con el proveniente de la memoria:

- t_1: MAR ← ($IR_{dirección}$) La dirección de memoria se carga al MAR
- t_2: MBR ← mem[MAR] El contenido de la memoria se copia en MBR
  T2 ← B El contenido de B se copia a T2
- t_3: T1 ← MBR El contenido de MBR se copia a T1
- t_4: B ← T1 + T2 La ALU realiza la suma y el resultado se guarda en B

Ciclo de instrucción detallado

Una visión más detallada del ciclo de instrucción puede darse con la forma de un diagrama de estados. Para una determinada instrucción, algunos pueden no estar y otros recorrerse en más de una ocasión.



Ciclo de instrucción detallado

1. **Cálculo de la dirección de la instrucción**: determina la dirección de la siguiente instrucción a ejecutarse
2. **Captación de la instrucción**: lee la instrucción de su posición de memoria a la CPU.
3. **Decodificación de la instrucción**: determina el tipo de operación a realizar y los operandos que se usarán.
4. **Cálculo dirección del operando**: si la operación implica la referencia a un operando en la memoria o E/S, determina la dirección.
5. **Captación del operando**: busca el operando en la memoria o E/S.
6. **Operación con datos**: ejecuta la instrucción.
7. **Cálculo de la dirección del operando**.
8. **Almacenamiento del operando**: escribe el resultado en la memoria o E/S

Los estados 2,5 y 8 implican interacción entre la CPU y la memoria o la E/S; el resto son internos a la CPU.

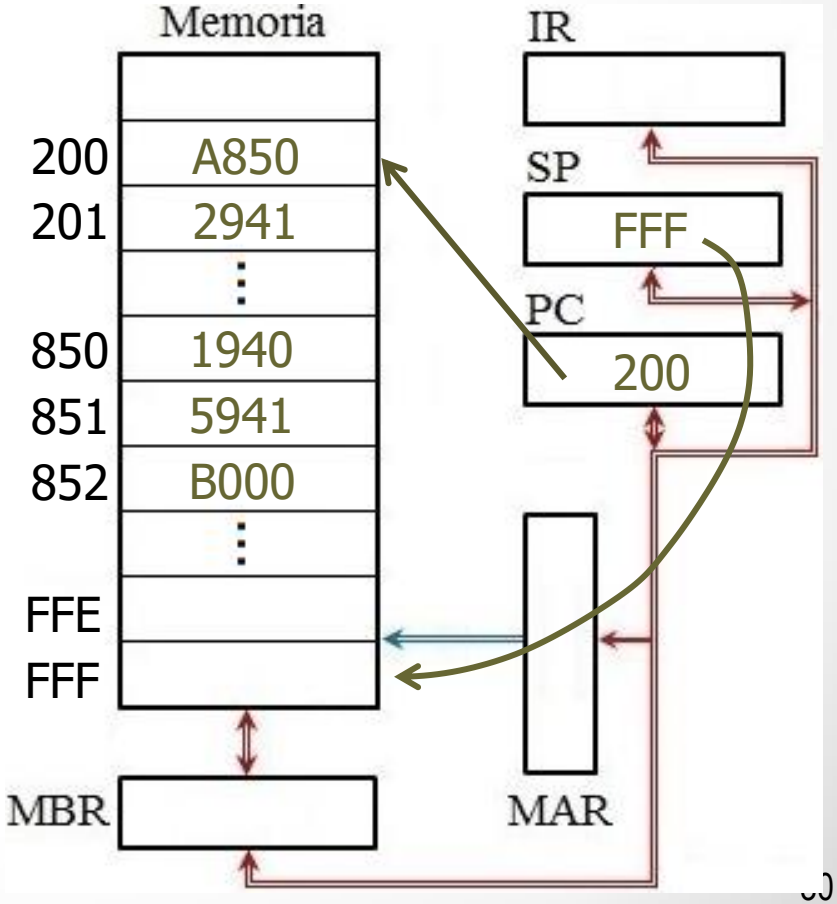
Subrutinas

Las rutinas o procedimientos 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”*).

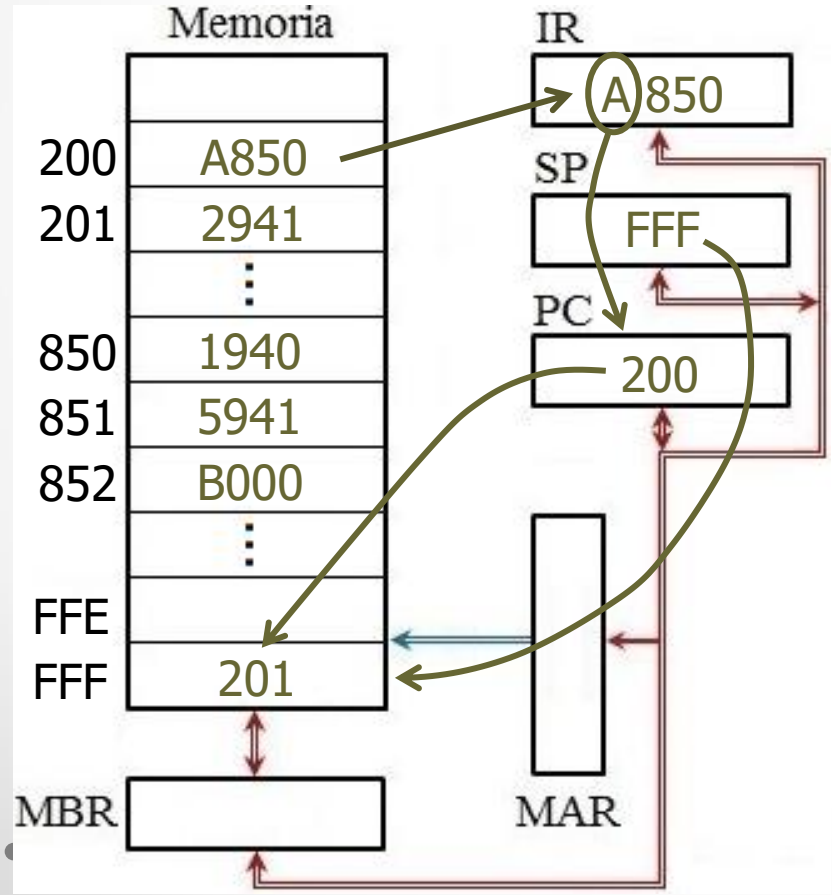
Para este ejemplo, se supone un programa principal que se está ejecutando en la posición 200_{16}.

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.

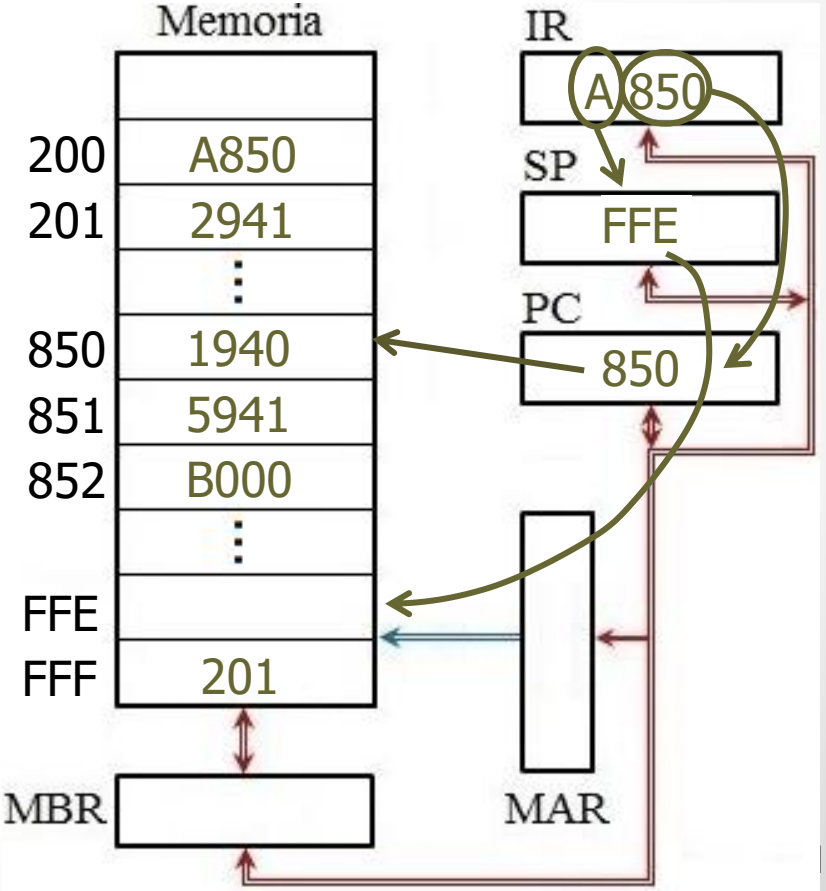


Subrutinas

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



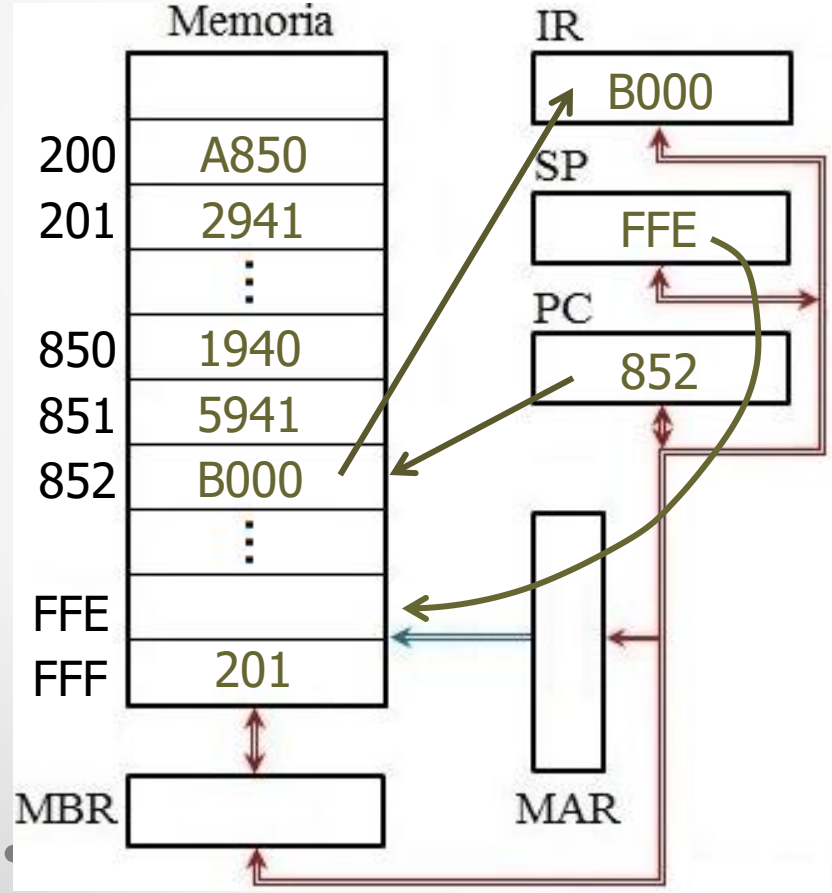
Se decrementa el SP, y el PC se carga con la dirección de la subrutina a ejecutar:

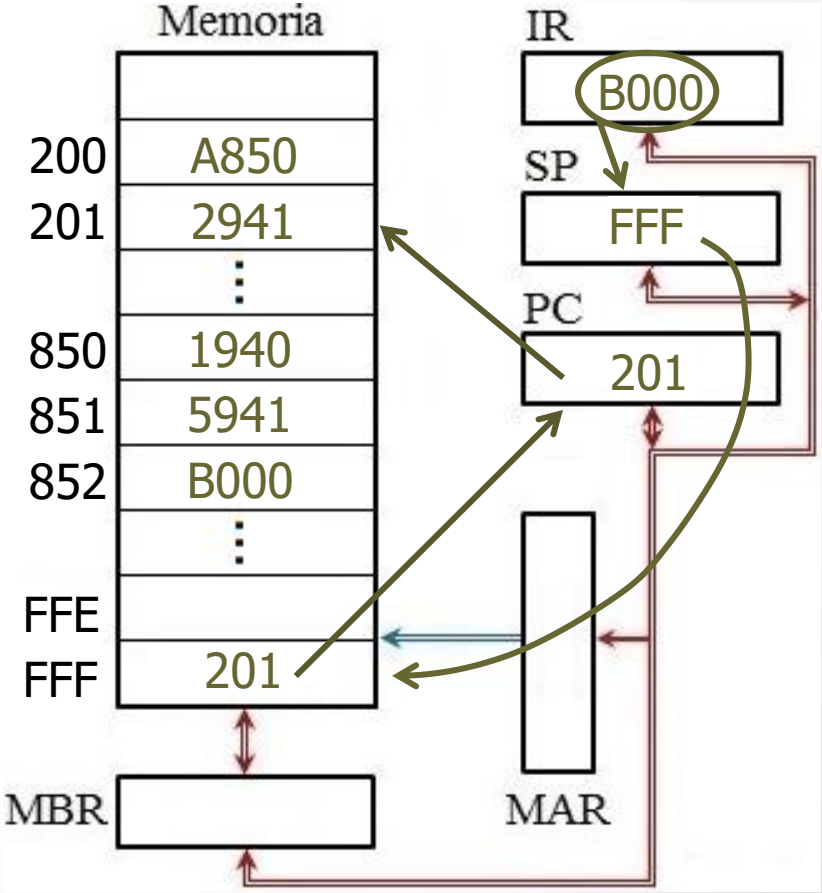


Subrutinas

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:





Interrupciones

Como se vio hasta aquí, un programa consiste en la ejecución secuencial de sus instrucciones. Sin embargo, existe un mecanismo que permite alterar el procesamiento normal de la CPU, denominado ***interrupción***.

Las interrupciones pueden originarse en varias fuentes:

- Como resultado de una ejecución de una instrucción

  Ejemplo: desbordamiento aritmético ("*overflow*"), división por cero

- Por un temporizador interno del procesador.

  Ejemplo: funciones regulares realizadas por el Sistema Operativo.

- Por una operación de E/S.

  Ejemplo: para indicar la finalización normal de una operación.

- Por un fallo de hardware.

  Ejemplo: error de paridad en la memoria, pérdida de energía.

Interrupciones

Además de las mencionadas muchos procesadores tienen instrucciones explícitas que afectan al procesador de la misma manera que las interrupciones por hardware, generalmente usadas para hacer llamadas a funciones del Sistema Operativo.

Salvo las de este tipo, las interrupciones son eventos asincrónicos a los programas que ejecuta la CPU. Su atención implica dejar de ejecutar estos para pasar a ejecutar otros procesos que atienda las causas que las originan. Esto puede causar demoras inaceptables en los programas.

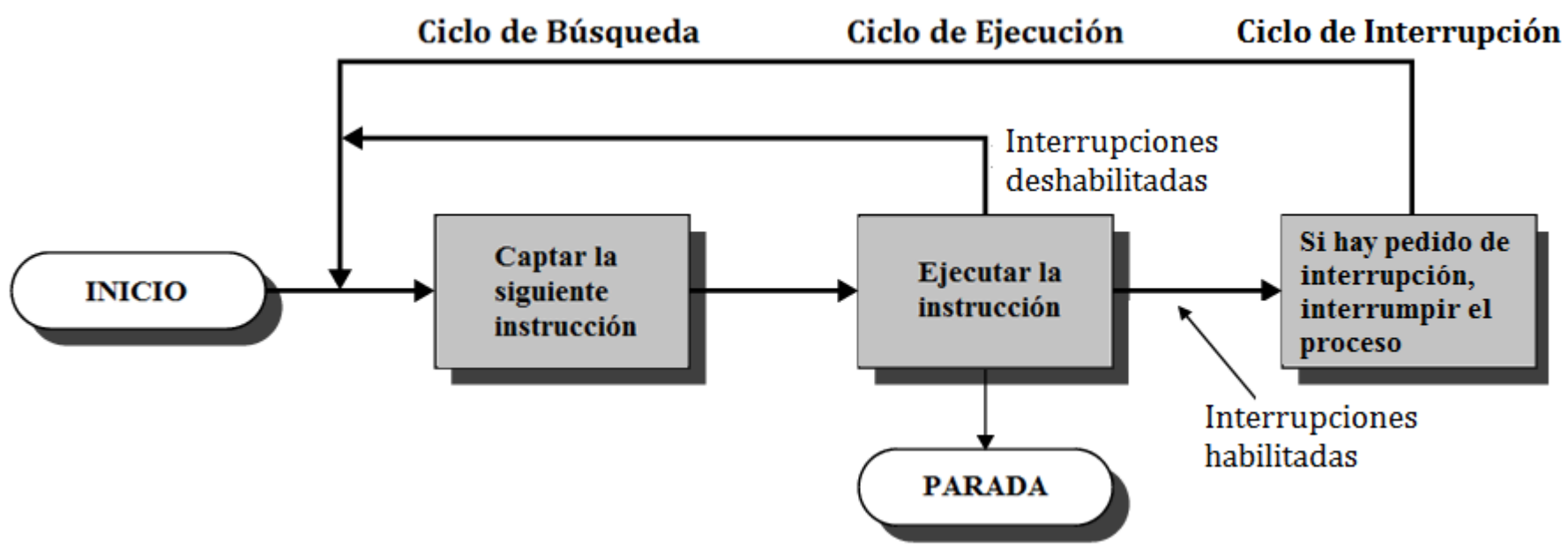
Por lo tanto, en sistemas con múltiples fuentes de interrupciones, estas deben ser jerarquizadas. Un primer paso para esto es separarlas en:

- Enmascarables: pueden ser ignoradas. Esto se logra mediante instrucciones específicas.
- No enmascarables: las que no pueden ignorarse, por indicar eventos peligrosos o de alta prioridad.

Ciclo de instrucción con interrupciones

Es necesario modificar el ciclo de instrucción básico, añadiendo un *ciclo de interrupción* a continuación del de ejecución:

- Se comprueba si se ha solicitado alguna interrupción, indicada por una señal (*flag*) de pedido de interrupción.
- Si no existe un flag activo o el pedido corresponde a una interrupción enmascarada, se capta la siguiente instrucción.
- Si hay algún pedido pendiente, se pasa a un *gestor de interrupción.*



Gestor de interrupciones

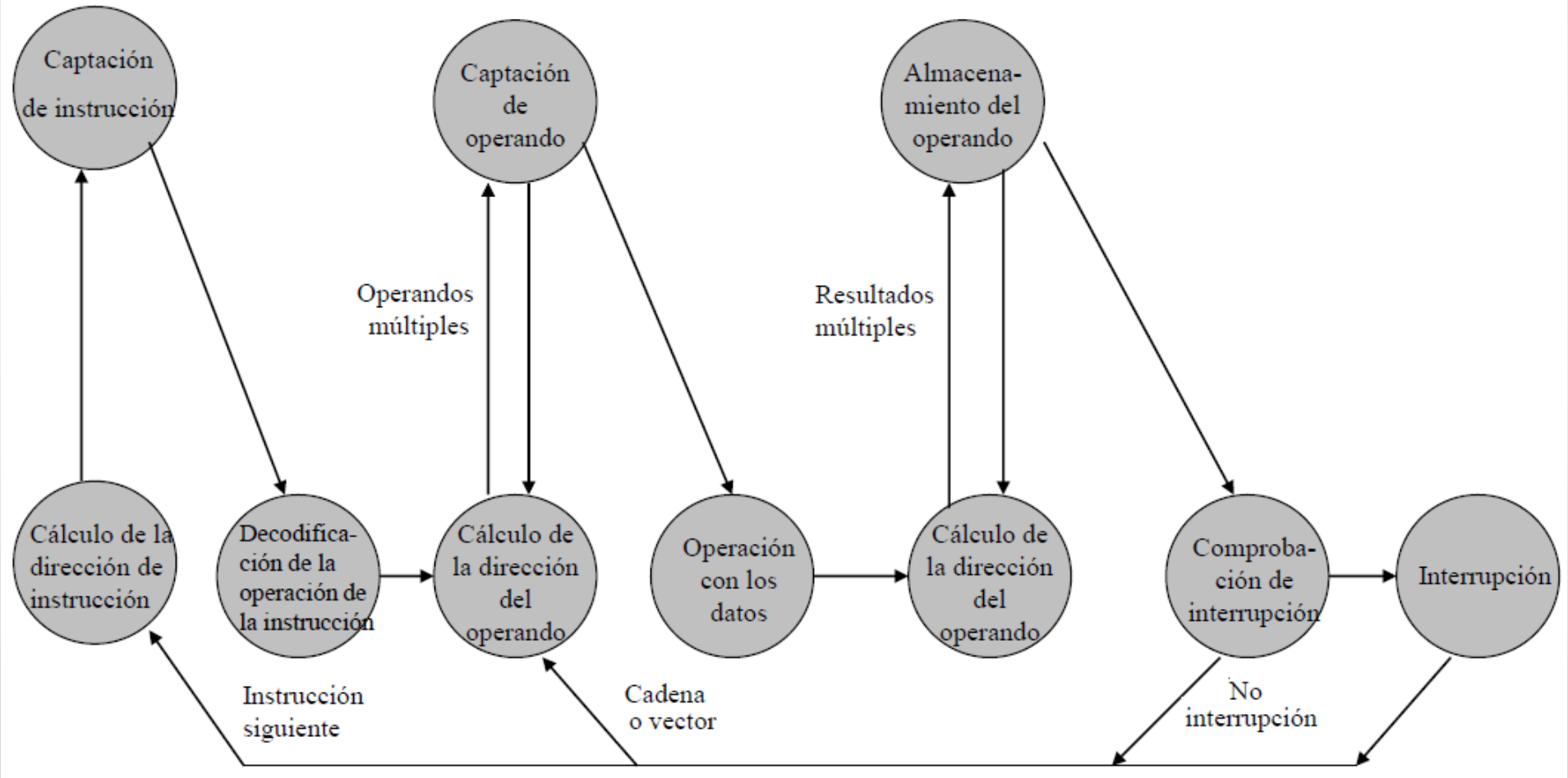
El gestor de interrupción debe realizar las siguientes acciones:

- Se suspende la ejecución del programa en curso.
- Se guarda el contexto (próxima instrucción a ejecutar y el estado del procesador).
- Se carga el PC con la dirección de comienzo de una rutina de gestión de interrupción. Se inhiben otras interrupciones.
- Finalizada la rutina de gestión, el procesador retoma la ejecución del programa del usuario en el punto de interrupción.

Este proceso es similar al explicado para la atención de las subrutinas, con la diferencia de que, además de guardarse en la pila el contenido del Contador de Programa, se guardan (y restauran), los contenidos de los registros y los flags del procesador con los contenidos que poseen al momento de abandonar el cauce del programa.

Diagrama de estados con interrupciones

En el ciclo de instrucción presentado como diagrama de estados, el ciclo de interrupción se inserta de la siguiente manera:



Manejo de múltiples interrupciones

Desde el punto de vista del proceso de la CPU:

- Mientras procesa una interrupción, el procesador ignorará otras interrupciones (el gestor las inhabilita).
- Las interrupciones permanecen pendientes y se comprueban después de que se ha procesado la primera interrupción.
- Son manejadas en secuencia a medida que ocurren
- Las interrupciones de baja prioridad pueden ser interrumpidas por interrupciones de mayor prioridad.
- Cuando se ha procesado una interrupción de mayor prioridad, el procesador vuelve a la interrupción anterior.

Manejo de múltiples interrupciones

Desde el punto de vista del hardware necesario para atender múltiples interrupciones, hay dos alternativas:

1. Cada dispositivo que puede provocar interrupción tiene una entrada física de interrupción conectada a la CPU.
   - Es muy sencillo, pero muy caro.
2. Línea de interrupción única
   Una sola entrada física de pedido de interrupción a la que están conectados todos los dispositivos. Existen dos posibilidades:
   - "Preguntar" a cada dispositivo si ha producido el pedido de interrupción (técnica *Polling*/encuesta).
   - El dispositivo que quiere interrumpir además de la señal de pedido de interrupción, debe colocar en el bus de datos un identificador (vector). A su vez, éste puede ser generado por el periférico directamente o bien por un Controlador de Interrupciones (PIC).

Referencias

- Stallings, Williams - Organización y Arquitectura de Computadoras - 5º Ed. - Prentice Hall. Año 2000.

  → *Capítulo 3*