CENTRO UNIVERSITARIO
DE TECNOLOGÍA
Y ARTE DIGITAL

Arquitectura de ordenadores

5. Funcionamiento del computador

Ignacio Calles González ignacio.gonzalez@ext.live.u-tad.com
Tiago Manuel Louro Machado de Simas tiago.louro@u-tad.com
Francisco Javier García Algarra javier.algarra@u-tad.com
Carlos M. Vallez Fernández carlos.vallez@u-tad.com

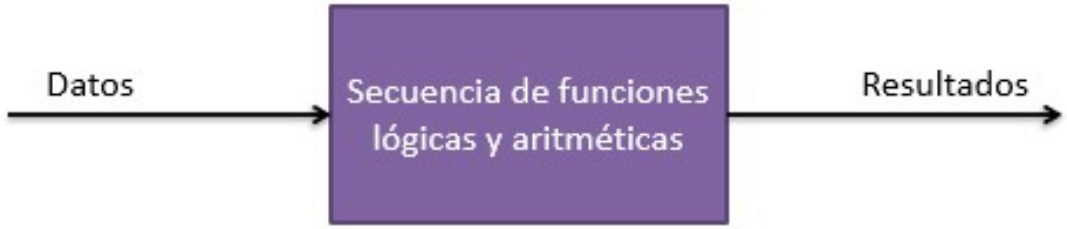
2023-2024

Índice

1. Introducción

Debemos recordar que inicialmente los precursores de los ordenadores eran máquinas destinadas a una tarea concreta. Su funcionalidad estaba establecida por las conexiones que tenían. Una función programada mediante circuitos electrónicos se denomina programa cableado (***hardwired program***). Sus características son:

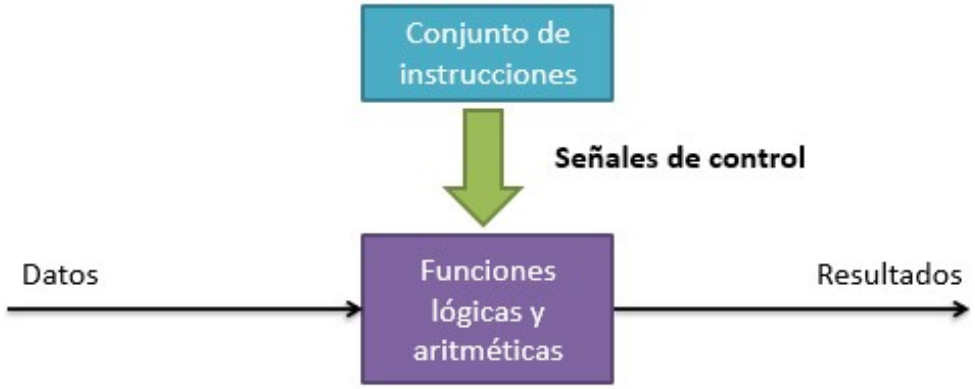
- La función que se desea representar y ejecutar se programa mediante la unión de componentes electrónicos formando un circuito.
- Para los **mismos datos** de entrada, el circuito siempre da los **mismos resultados**.
- Cuando el circuito está acabado, sólo puede ejecutar la función para la que ha sido creado. Ejecutar otra función implica rehacer el circuito electrónico.



1. Introducción

Posteriormente vimos que aparecieron los ordenadores programables. La **programación software** establece una distinción entre las funciones que se pueden ejecutar y el programa que se desea llevar a cabo:

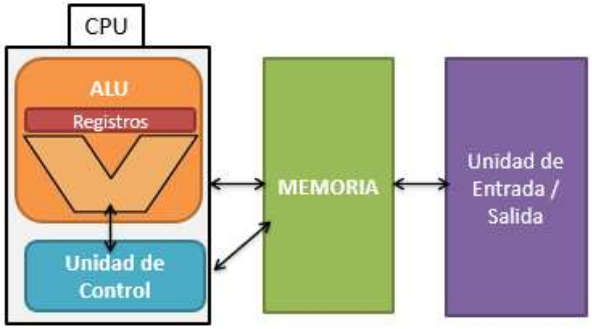
- Existe un conjunto fijo de funciones lógicas y aritméticas.
- Existe un conjunto de instrucciones que codifican el programa que se desea ejecutar.
- Existe un conjunto de señales de control que decodifican cada instrucción en el subconjunto de funciones lógicas y aritméticas que la pueden llevar a cabo.
- Para los **mismos datos** de entrada, los **resultados pueden variar**, ya que dependen del conjunto de instrucciones ejecutados sobre ellos.



1. Introducción

La **programación software** se basa en los principios definidos en la **arquitectura de Von Neumann**:

- Los datos y las instrucciones se almacenan en una **memoria** de lectura-escritura.
- Los contenidos de la memoria se identifican mediante una **dirección**, independientemente del contenido.
- La ejecución del programa se lleva a cabo ejecutando secuencialmente las **instrucciones** que lo componen.

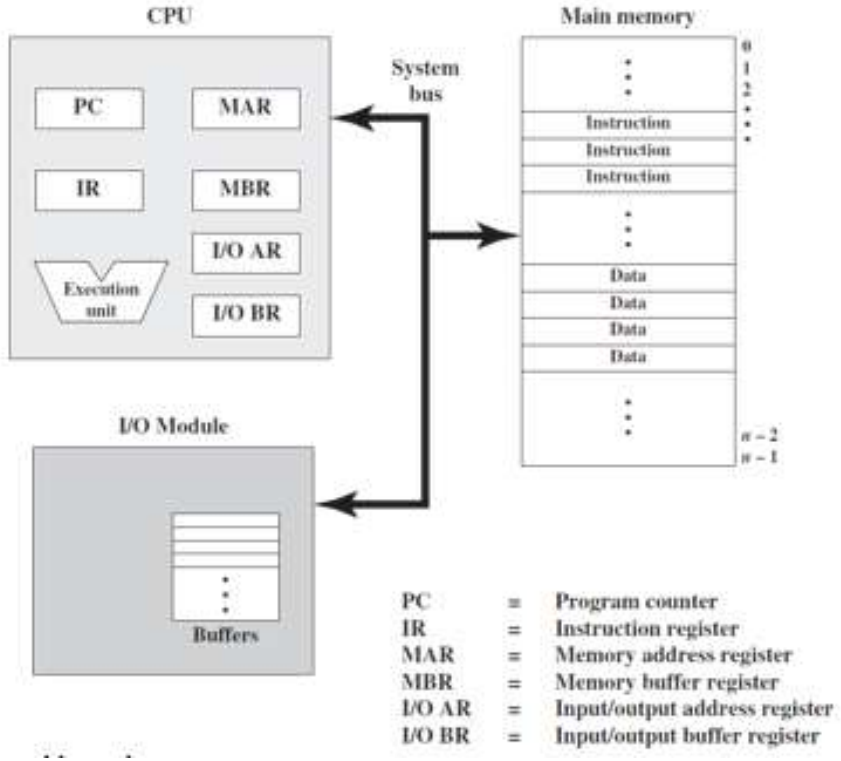


Índice

2. Estructura general de un ordenador programable

La **estructura general** de un computador programable por software es la siguiente:

- La **CPU intercambia datos con la memoria** mediante (registros):
  - **MAR**: Especifica la próxima dirección de lectura/escritura.
  - **MBR**: Almacena el dato leído/a escribir.
- Igualmente, **intercambia datos con la E/S** mediante:
  - **IOAR**: Especifica la dirección de un dispositivo de E/S.
  - **IOBR**: Almacena un dato a transferir a un dispositivo de E/S.
  - .

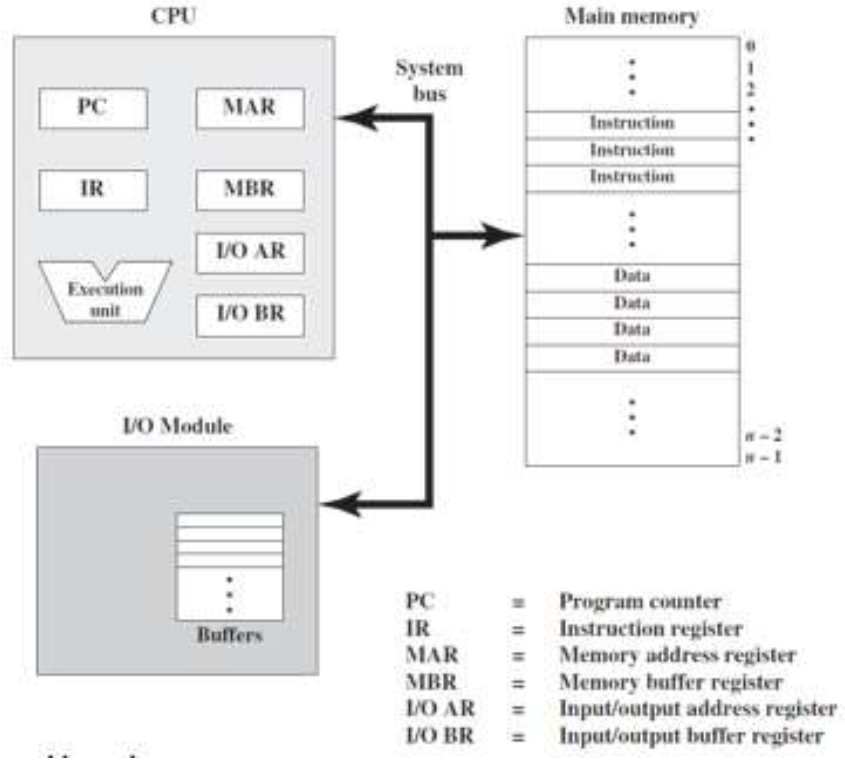


2. Estructura general de un ordenador programable

La **estructura general** de un computador programable por software es la siguiente:

- El **PC** contiene la **dirección de la siguiente instrucción a ejecutar**.
- EL **IR** contiene la **siguiente instrucción a ejecutar**.
- La **memoria** consiste en un conjunto de posiciones identificadas mediante **direcciones** enumeradas secuencialmente.
  - El contenido de cada posición es un valor en **binario**, que puede ser interpretado como una **instrucción** o como un **dato**.

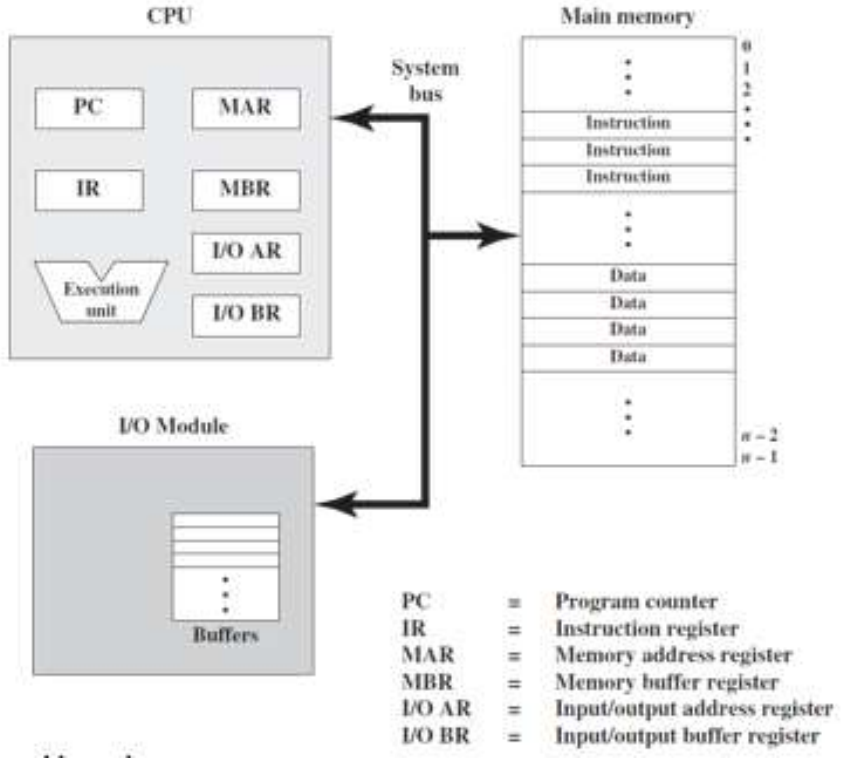
.



2. Estructura general de un ordenador programable

La **estructura general** de un computador programable por software es la siguiente:

- Un **módulo de E/S** transfiere datos desde un dispositivo externo a la CPU y a la memoria.
  - Contiene sus propios **registros** (*buffers*) para almacenar los datos a transferir.
- Las transferencias de datos se llevan a cabo a través del bus de sistema.



Índice

3 Ejecución de un programa

Un programa software se ejecuta mediante la **repetición** de un proceso denominado **ciclo de instrucción**. Este proceso consta de dos fases esenciales: el **ciclo de captación** y el **ciclo de ejecución**.

Comienza el ciclo de instrucción:

a. Comienza el ciclo de **captación**:

   I. **La CPU extrae de la memoria la instrucción** indicada en el contador de programa (PC). La instrucción **se almacena en el registro de instrucción** (IR).

   II. El valor del **PC se incrementa**.

3 Ejecución de un programa

b. Comienza el ciclo de **ejecución**:

   I. **La CPU interpreta la instrucción** para determinar qué acción debe llevar a cabo.

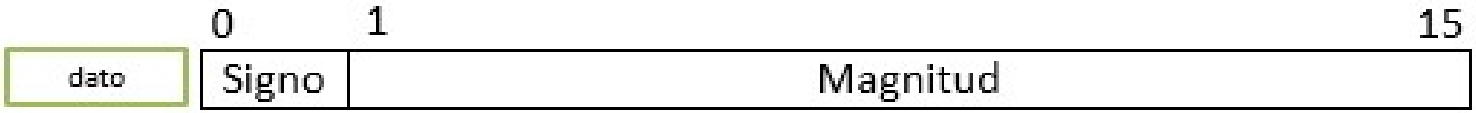
   II. **Se lleva a cabo la acción**, que puede ser:

   - Transferir datos entre CPU y memoria.
   - Transferir datos entre CPU y E/S.
   - Aplicar una operación lógica o aritmética sobre unos datos.
   - Alterar el valor del PC.

3 Ejecución de un programa

Ejemplo

Supóngase un computador que maneja datos de 16 bits de longitud. El formato de las instrucciones y de los datos que maneja ese computador es el siguiente:



3 Ejecución de un programa

Ejemplo

Expresándolo en hexadecimal:

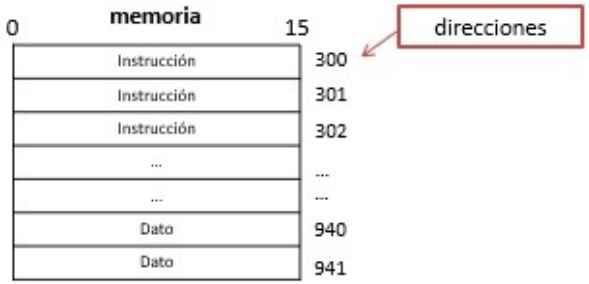
.

3 Ejecución de un programa

Ejemplo

En la siguiente imagen se detallan otras características del ejemplo:

La memoria se organiza en posiciones de 16 bits.



Tiene un PC, un IR, y un único registro para almacenar datos llamado AC (acumulador).

Las tres primeras operaciones realizables por el procesador son:

Codop	Codop (hexadecimal)	Instrucción
0001	1	Cargar AC desde memoria
0010	2	Guardar AC en memoria
0101	5	Sumar al AC un dato de la memoria

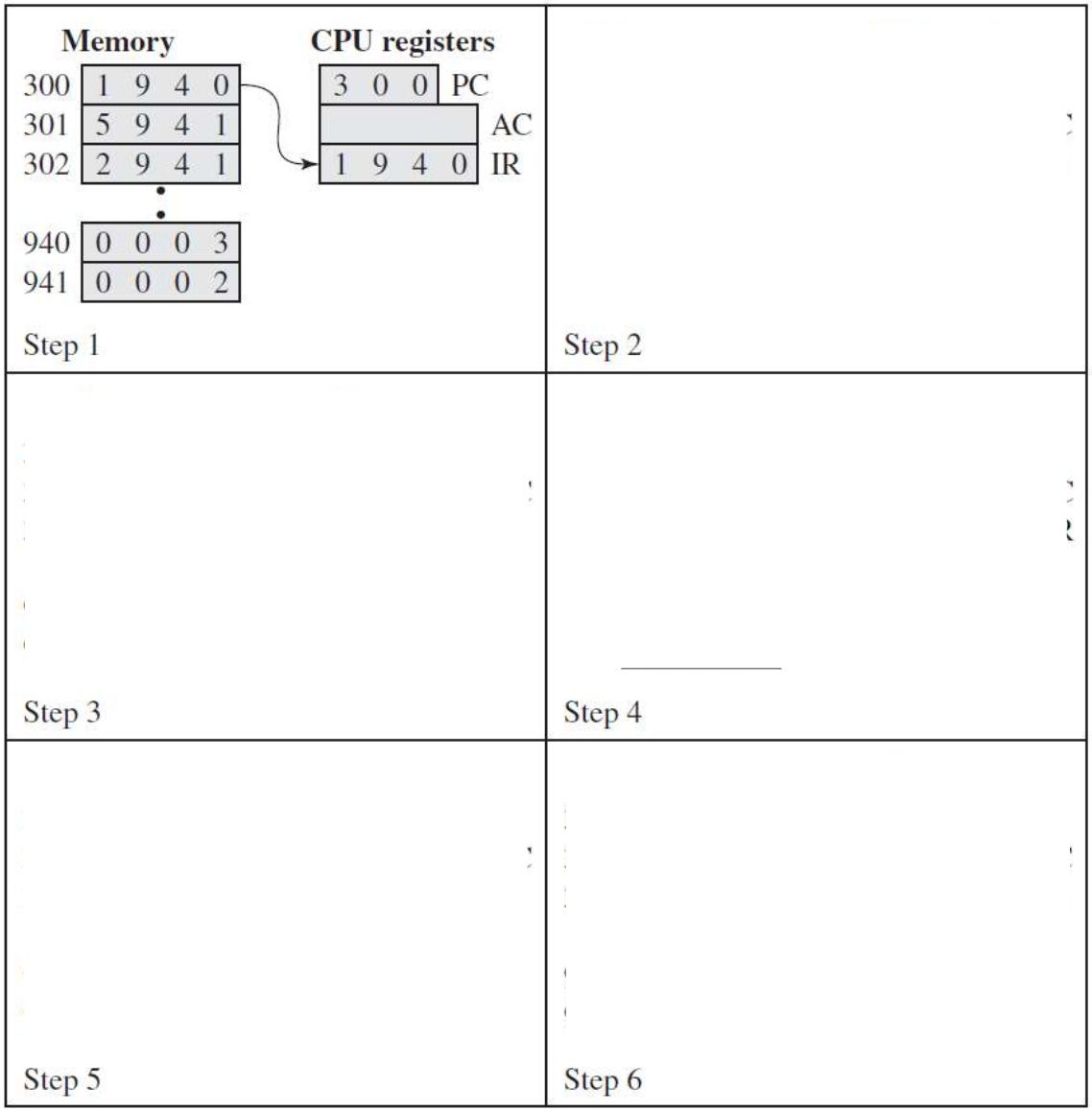
Se desea ejecutar el siguiente programa:

1940
5941
2941

Índice

4 Ejemplo de ejecución

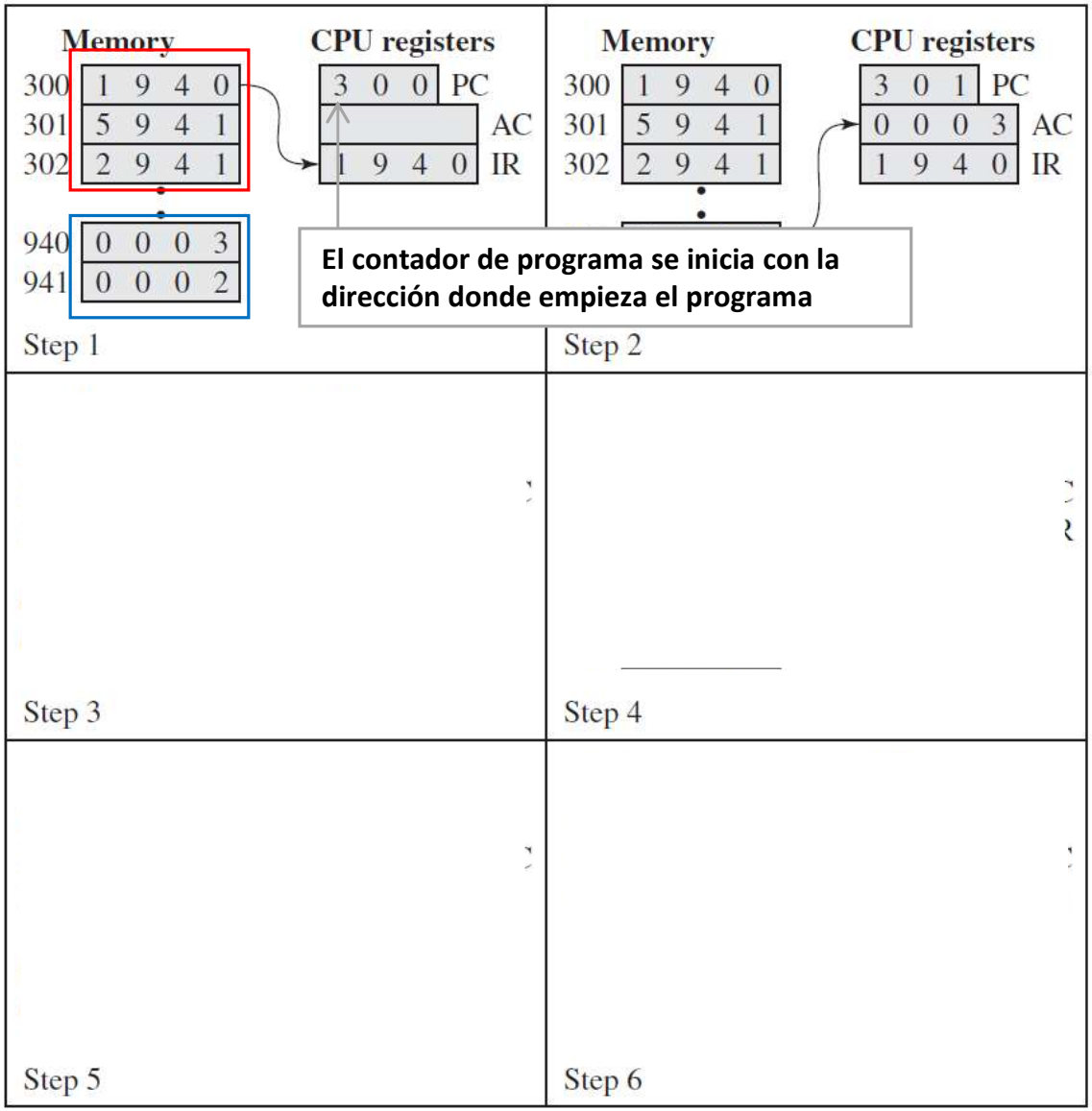
En las siguientes diapositivas vemos paso a paso cómo van cambiando los registros del ordenador



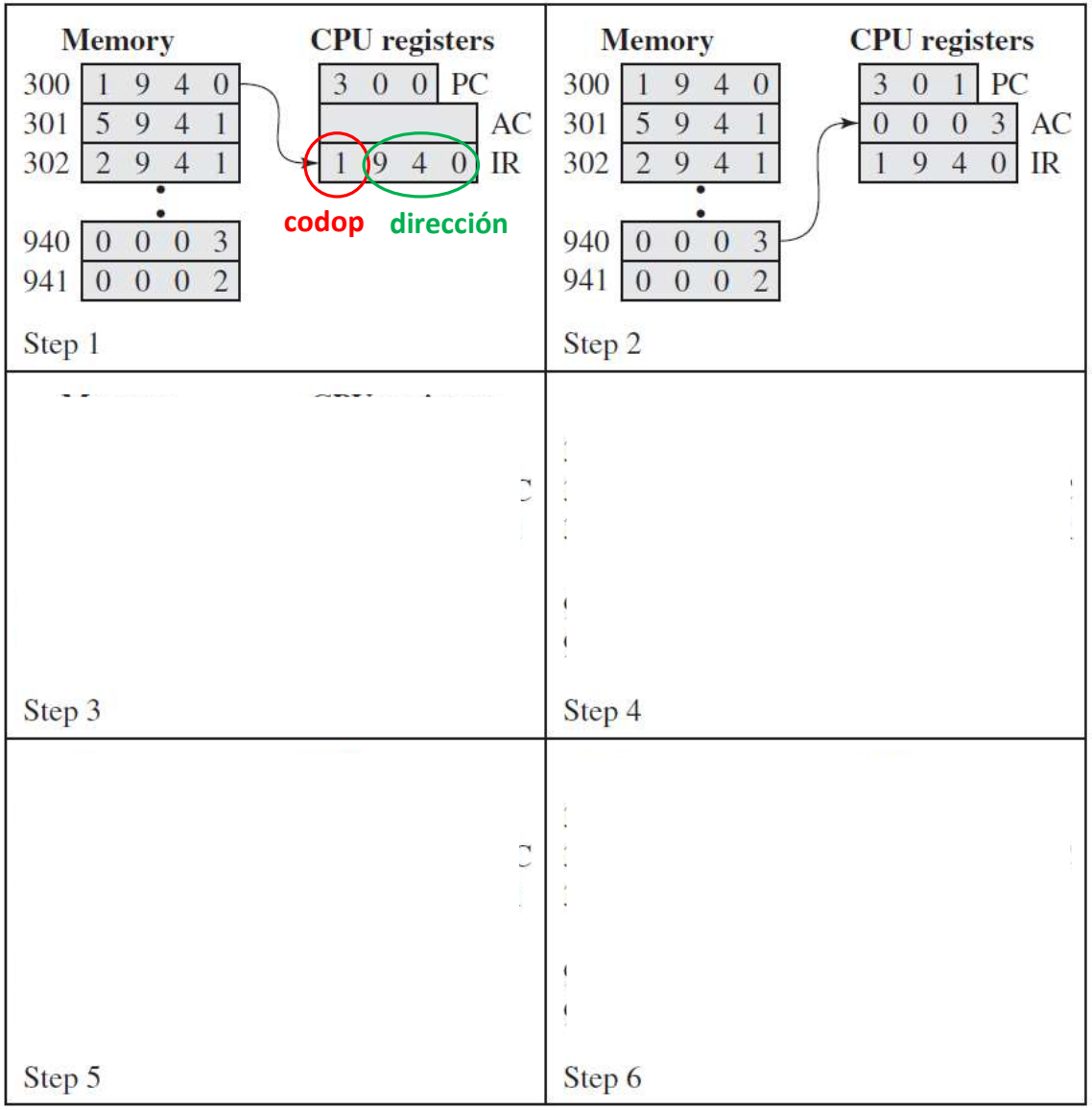
Codop	Codop (hexadecimal)	Instrucción
0001	1	Cargar AC desde memoria
0010	2	Guardar AC en memoria
0101	5	Sumar al AC un dato de la memoria

Instrucciones

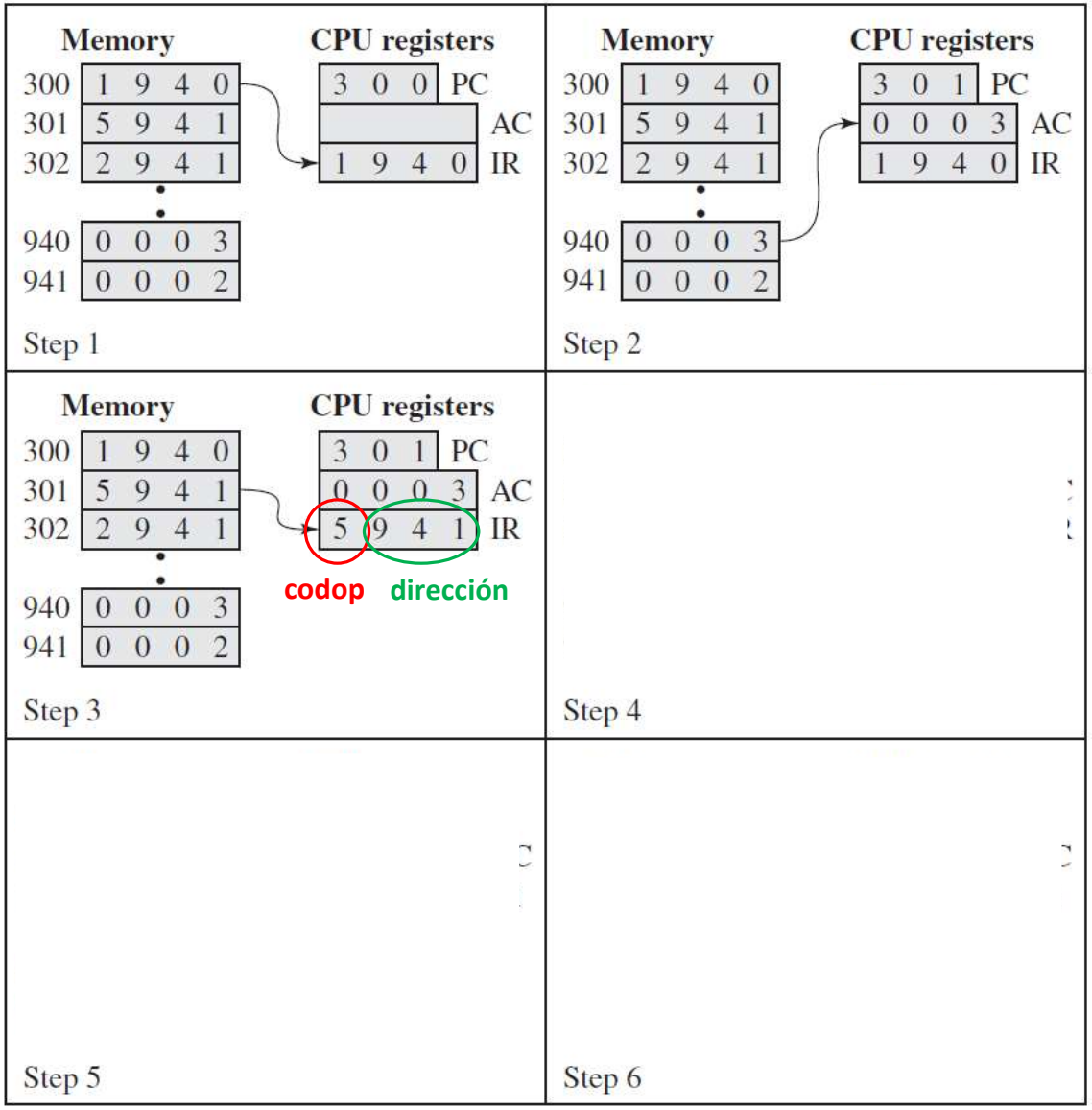
Datos



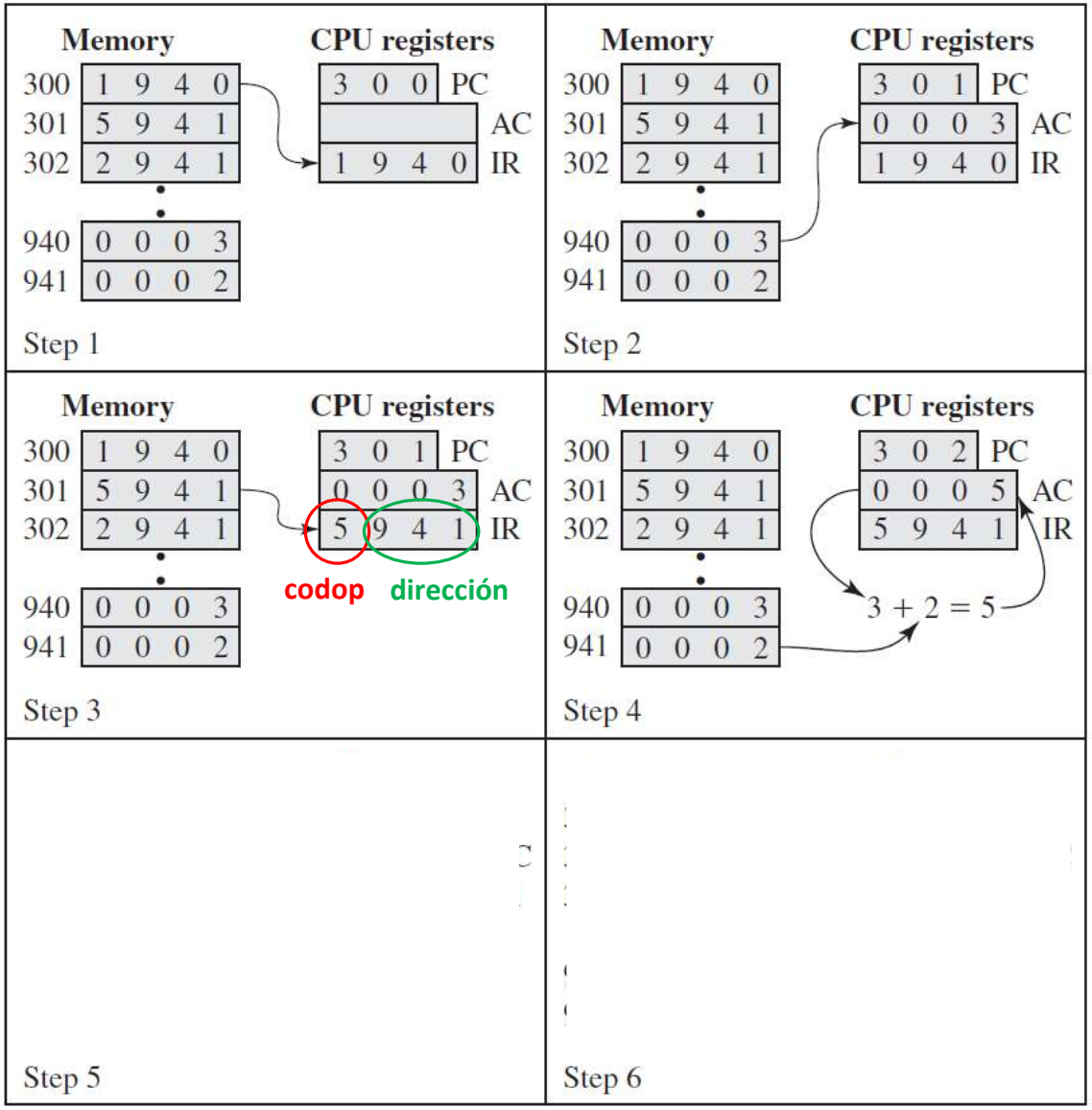
Codop	Codop (hexadecimal)	Instrucción
0001	1	Cargar AC desde memoria
0010	2	Guardar AC en memoria
0101	5	Sumar al AC un dato de la memoria



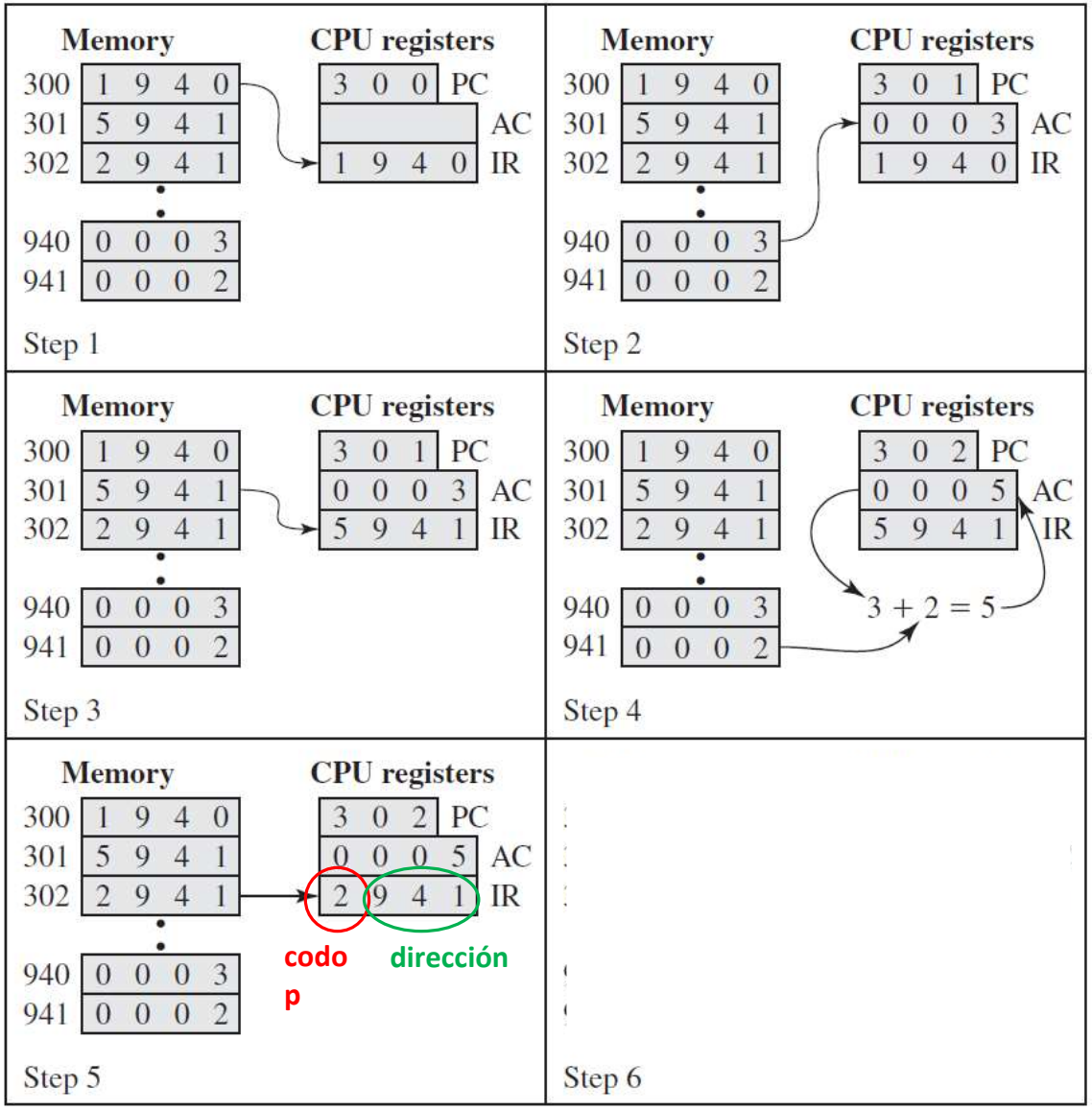
Codop	Codop (hexadecimal)	Instrucción
0001	1	Cargar AC desde memoria
0010	2	Guardar AC en memoria
0101	5	Sumar al AC un dato de la memoria



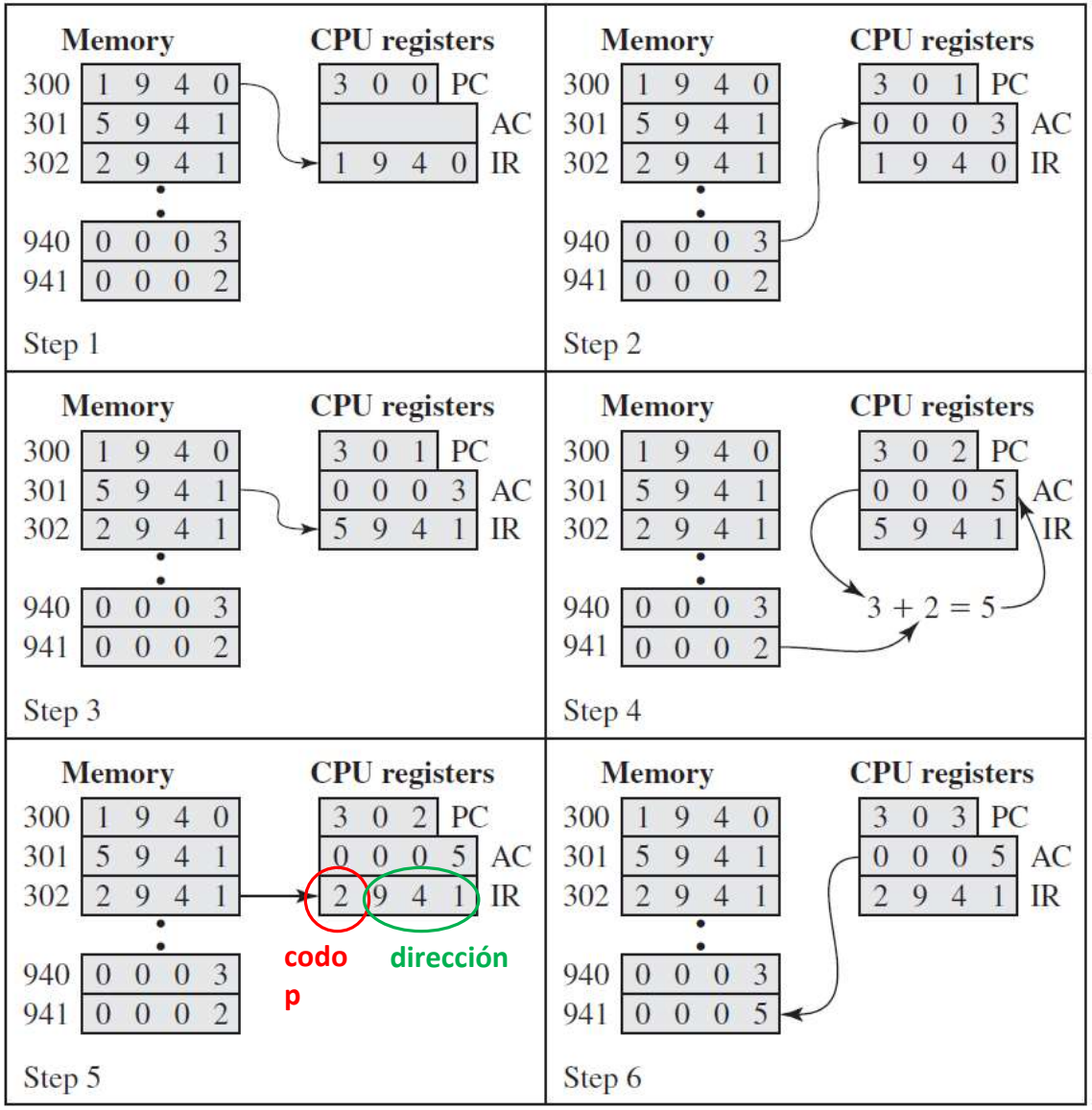
Codop	Codop (hexadecimal)	Instrucción
0001	1	Cargar AC desde memoria
0010	2	Guardar AC en memoria
0101	5	Sumar al AC un dato de la memoria



Codop	Codop (hexadecimal)	Instrucción
0001	1	Cargar AC desde memoria
0010	2	Guardar AC en memoria
0101	5	Sumar al AC un dato de la memoria



Codop	Codop (hexadecimal)	Instrucción
0001	1	Cargar AC desde memoria
0010	2	Guardar AC en memoria
0101	5	Sumar al AC un dato de la memoria



Codop	Codop (hexadecimal)	Instrucción
0001	1	Cargar AC desde memoria
0010	2	Guardar AC en memoria
0101	5	Sumar al AC un dato de la memoria

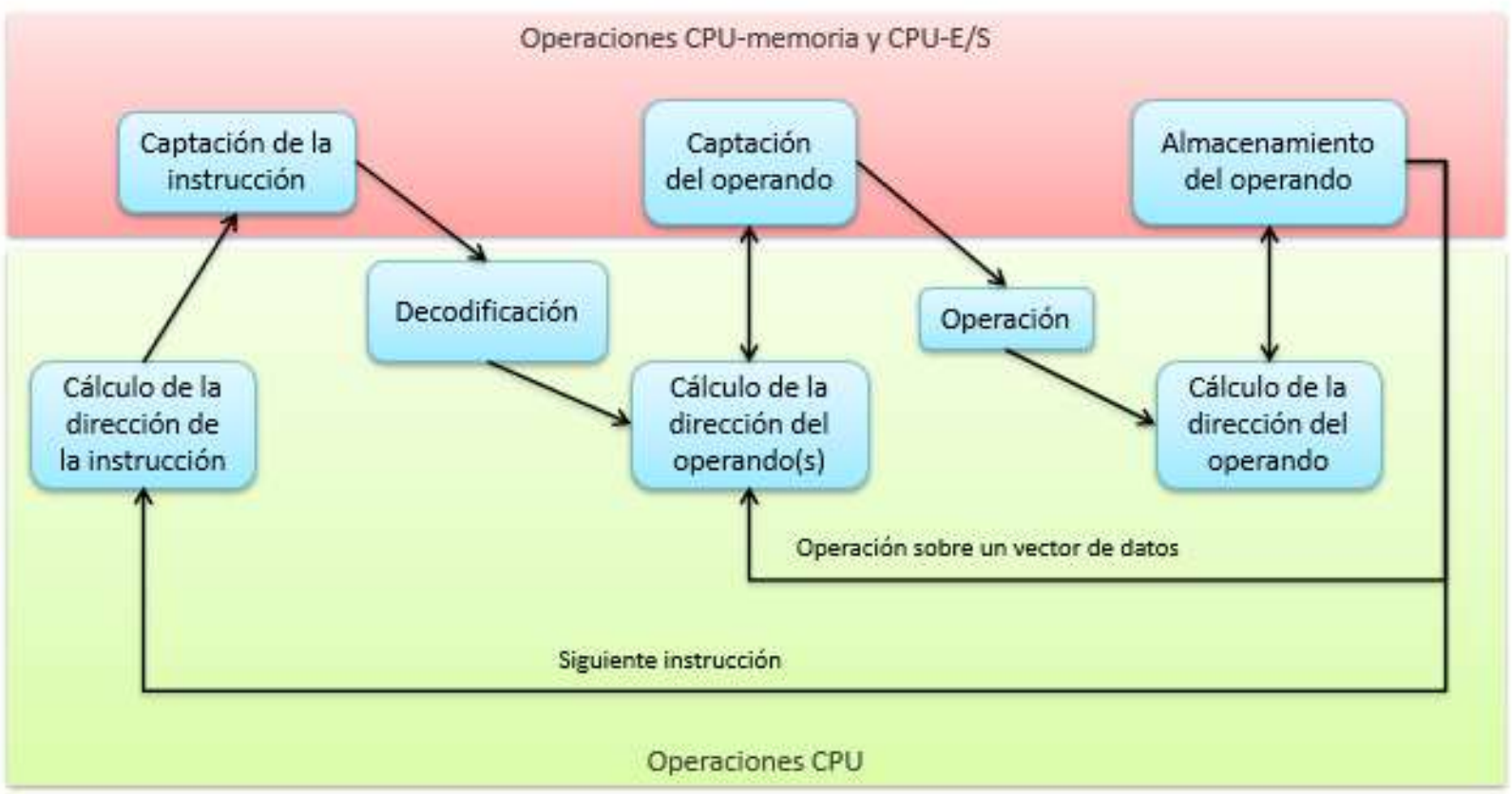
Índice

5. Diagrama de estados de un ciclo de instrucción

En el ejemplo anterior hemos visto cómo se van ejecutando cada una de las instrucciones. El siguiente diagrama resume lo que se conoce como ciclo de instrucción. Muestra de forma esquemática los diferentes estados por los que va pasando la ejecución de una instrucción.

5. Diagrama de estados de un ciclo de instrucción

Diagrama de estados de un **ciclo de instrucción**:



Índice

6. Ciclo de interrupción

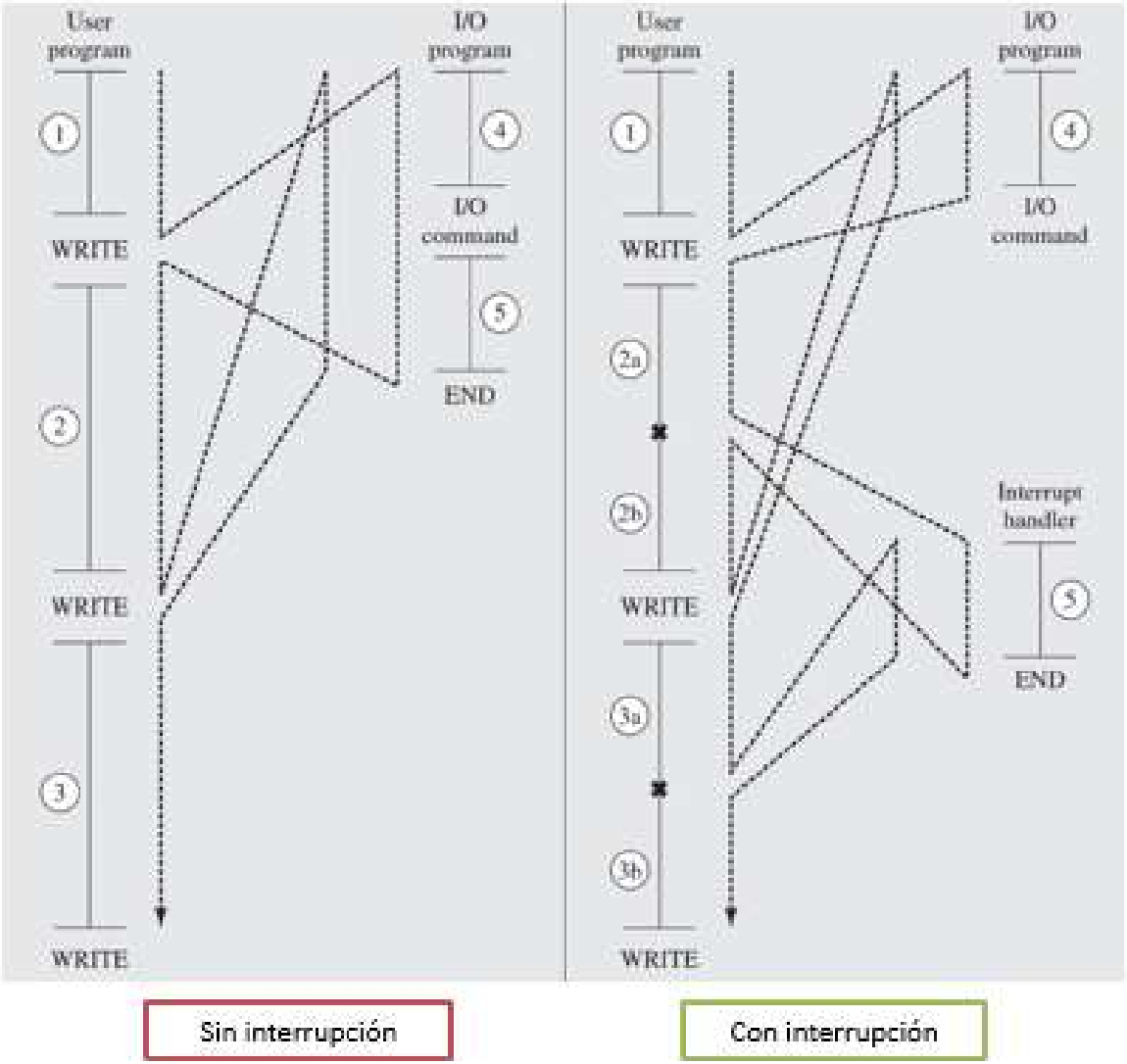
- Existe un ciclo adicional dentro del ciclo de instrucción: **el ciclo de interrupción**.
- Una **interrupción** es un mecanismo que interrumpe el funcionamiento normal de la CPU para que ésta gestione una incidencia: operación de E/S, temporización, fallo de hardware, fallo de programa.
- Las más comunes son las interrupciones de E/S.
- Los dispositivos de E/S suelen operar más lentamente que la CPU.
- Se debe evitar que la CPU tenga que quedarse esperando a que la E/S acabe de operar para seguir trabajando, ya que esto es ineficiente.

6. Ciclo de interrupción

- Gracias a las interrupciones, la CPU puede seguir ejecutando instrucciones mientras la E/S realiza su trabajo.
- Gracias a las interrupciones, se mejora la eficiencia de la CPU.
- El tema de interrupciones es algo que se abordará en detalle en la asignatura de Sistemas operativos.

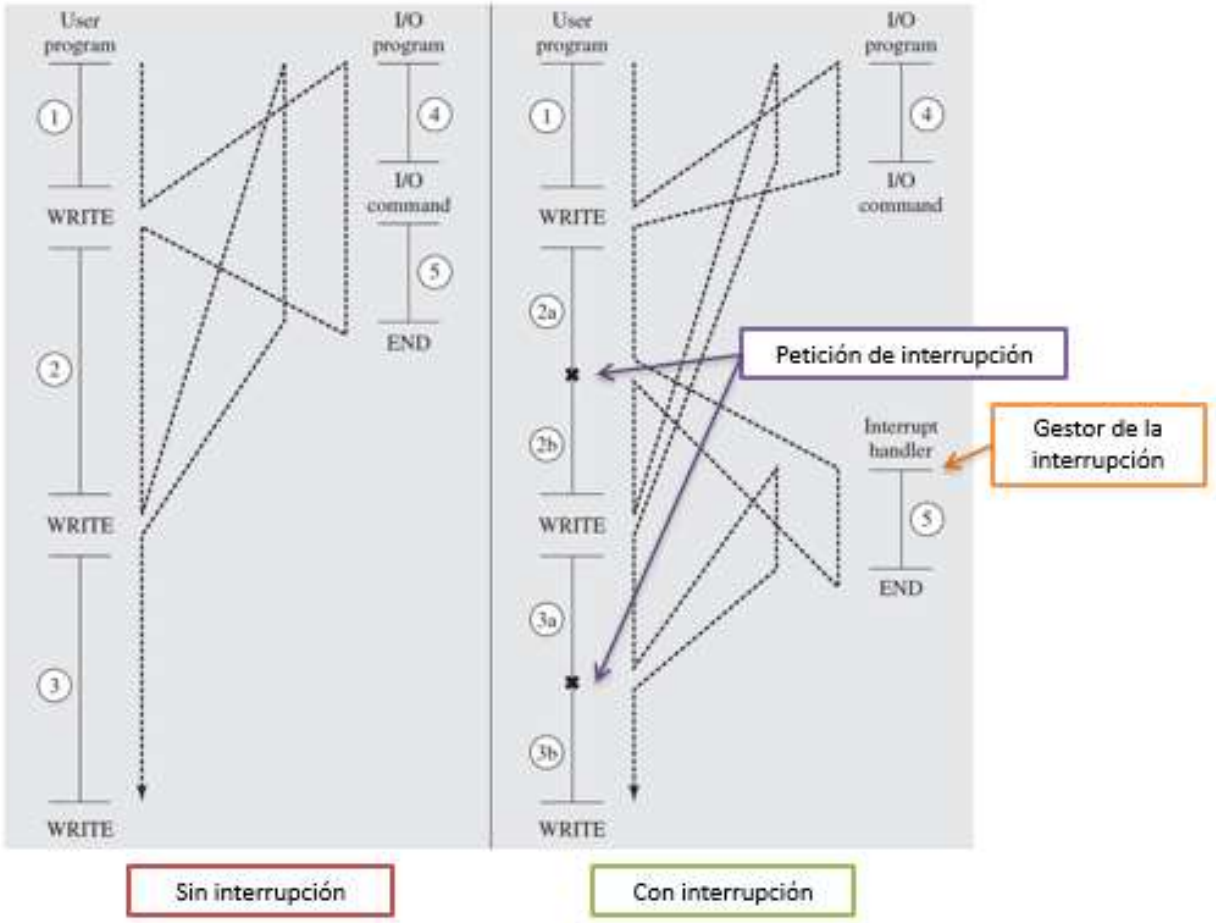
Vamos a comprobar en las siguientes imágenes cómo se mejora la eficiencia de la CPU con interrupciones:

6. Ciclo de interrupción



6. Ciclo de interrupción

Vemos en la siguiente imagen claramente cómo con interrupciones la CPU puede dedicarse a otras tareas evitando caer en una ejecución puramente secuencial.



6. Ciclo de interrupción

- Para gestionar el ciclo de interrupción, cada vez que la CPU acaba el ciclo de ejecución, comprueba si hay interrupciones pendientes. Si las hay, realiza el ciclo de interrupción. Si no, pasa al ciclo de instrucción siguiente.

  Comienza el ciclo de instrucción:

  a. Ciclo de **captación**.

  b. Ciclo de **ejecución**.

  c. Si hay interrupción pendiente:

     i. Comienza el ciclo de **interrupción**.

     ii. Suspende la ejecución del programa en curso y almacena su contexto.

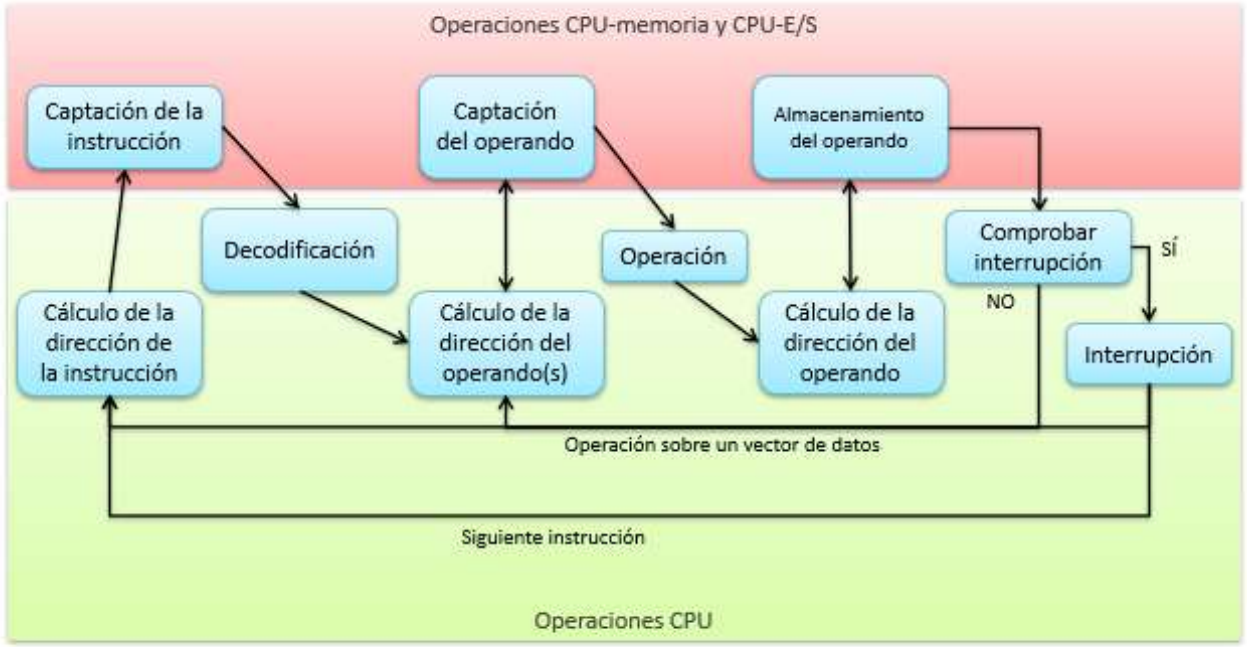
     iii. Mueve el contador de programa (PC) al comienzo del programa de gestión de la interrupción.

  d. En otro caso, pasa al siguiente ciclo de instrucción.

6. Ciclo de interrupción

Si incluimos la gestión de las interrupciones en nuestro anterior diagrama de estados nos quedaría algo como:

Diagrama de estados de un **ciclo de instrucción con interrupciones**:



Índice

7. Simulación de la ejecución de un programa

En este apartado haremos uso de un simulador web para entender cómo funciona un programa dentro del ordenador. Para ello basta con acceder a la web: https://peterhigginson.co.uk/lmc/

7. Simulación de la ejecución de un programa

Emplearemos el siguiente código ensamblador como ejemplo:

```
INP
STA 99
INP
ADD 99
OUT
HLT
```

Los pasos que ejecutaremos son:

Submit
RUN
introducir un numero hex,
otro num hex
y en OUPUT aparecerá la suma de los dígitos.