



SISTEMAS OPERATIVOS

3004610 - 1

German Sánchez Torres, I.S., M.Sc., Ph.D.

Profesor, Facultad de Ingeniería - Programa de Sistemas

Universidad del Magdalena, Santa Marta.

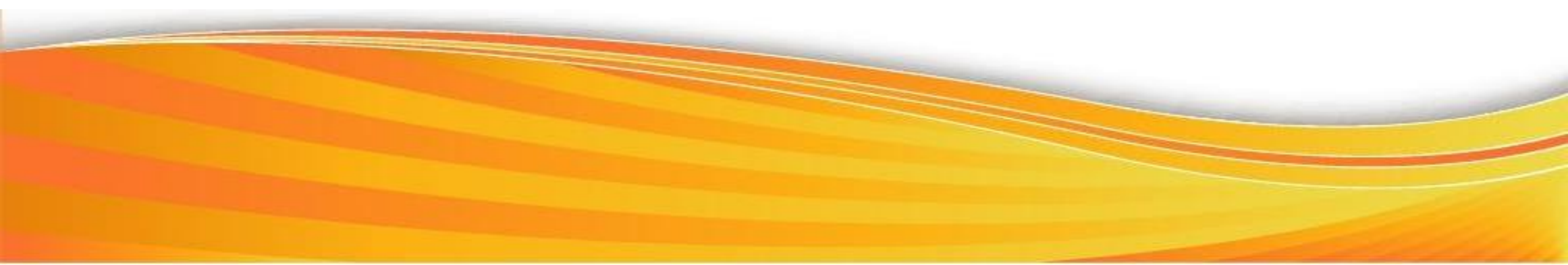
Phone: +57 (5) 4214079 Ext 1138 - 301-683 6593

Edificio Docente, Cub 3D401.

Email: sanchez.gt@gmail.com - gsanchez@unimagdalena.edu.co



Conceptos Arquitectónicos de la computadora



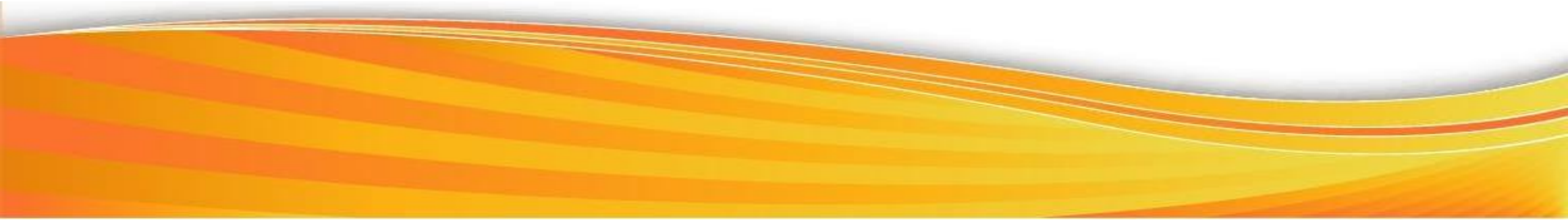
Conceptos Arquitectónicos de la computadora

Contenido

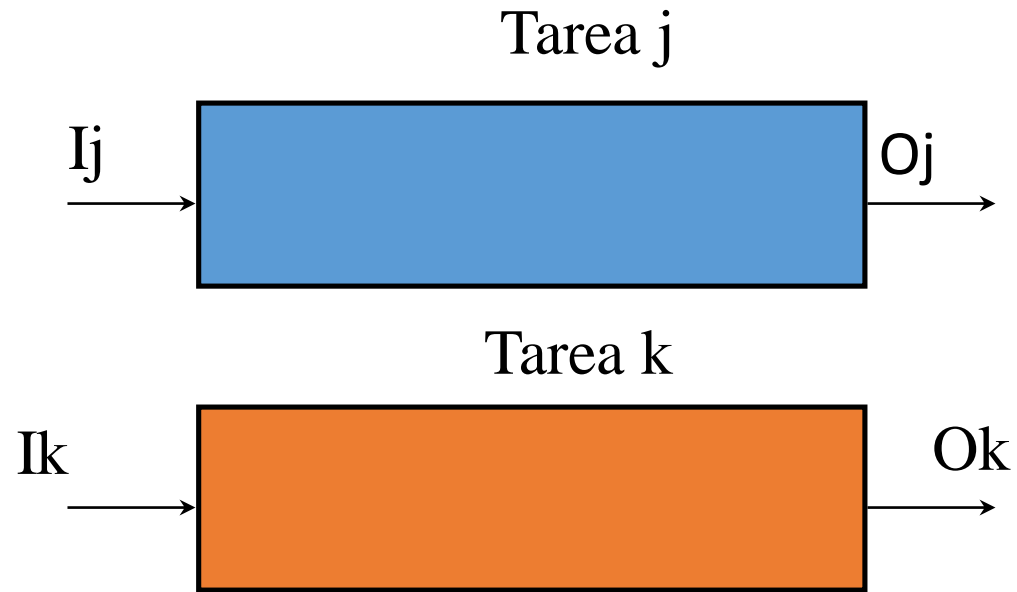
1. Estructura y funcionamiento del ordenador.
2. Modelos de programación.
3. Interrupciones.
4. Reloj.
5. Jerarquía de memoria.



Estructura y Funcionamiento Del Ordenador



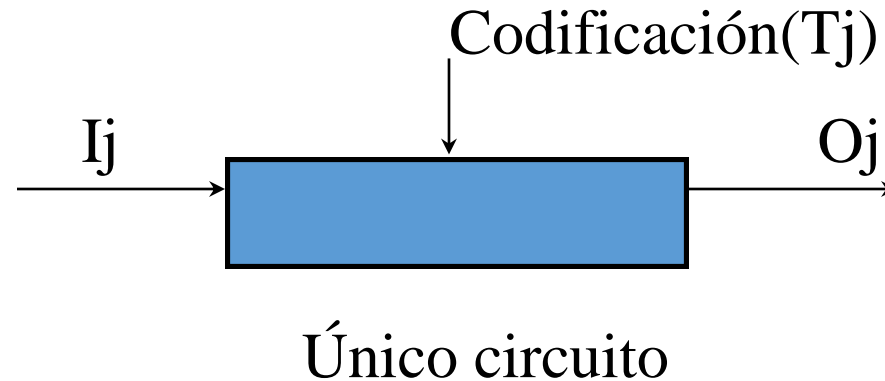
Computación digital personalizada



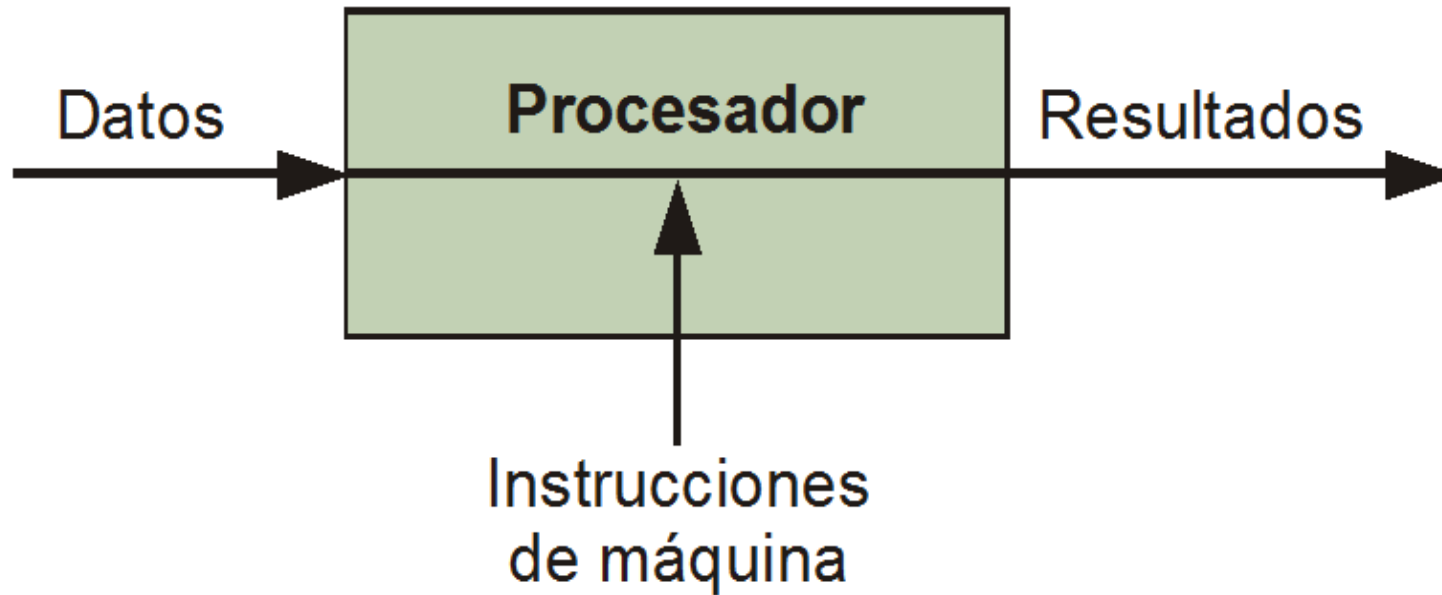
Circuitos diferentes para cada tarea

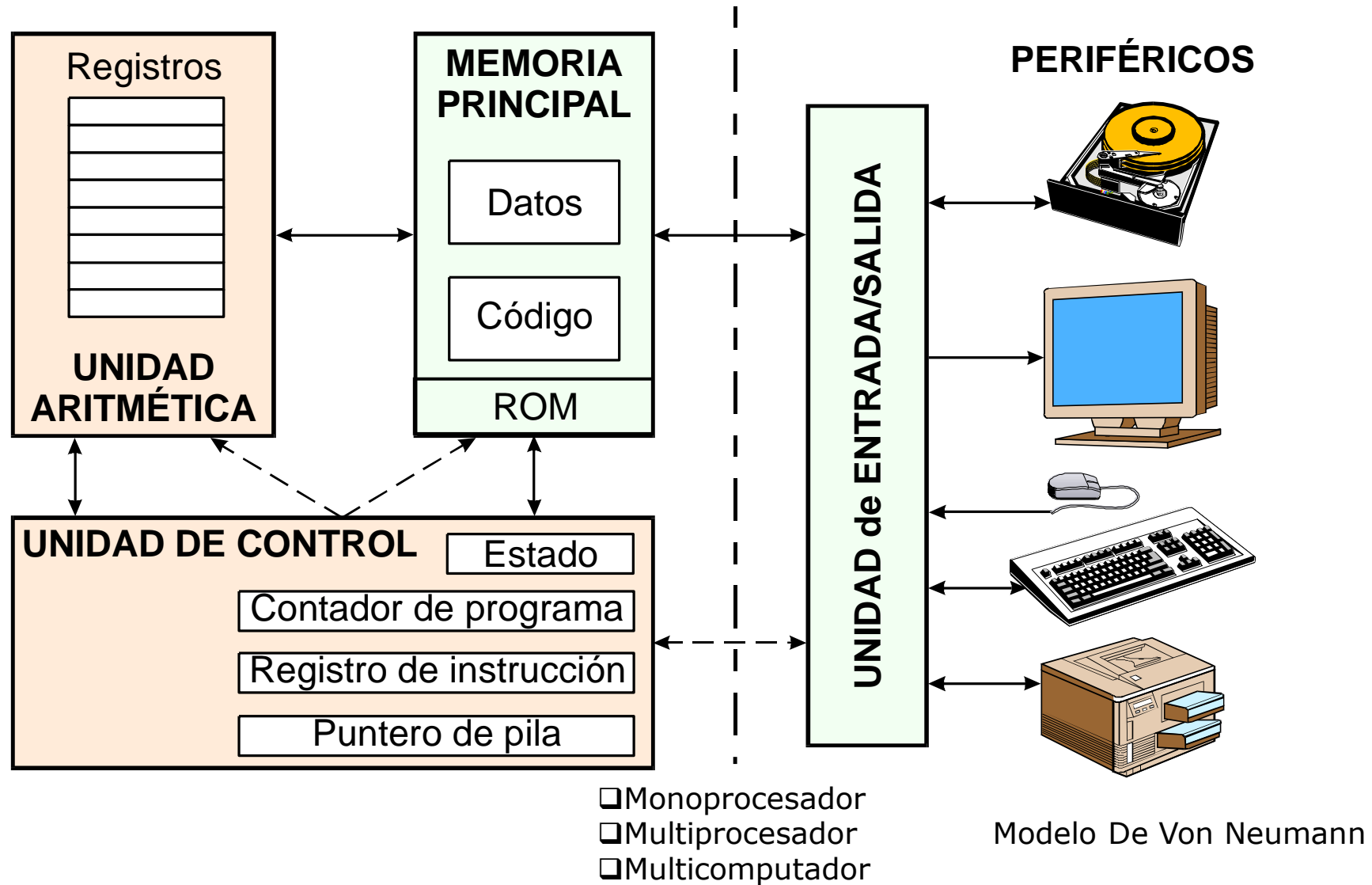
Computación de programas almacenados

- *Von Neumann* mostró la existencia de una maquina universal (hardware) que puede ser personalizada para realizar diferentes tareas mediante entradas de control
- El *Software* es la codificación de la tarea para controlar la maquina
- Única unidad de almacenamiento dado que instrucciones y datos no presentan diferencias



Estructura general a alto nivel



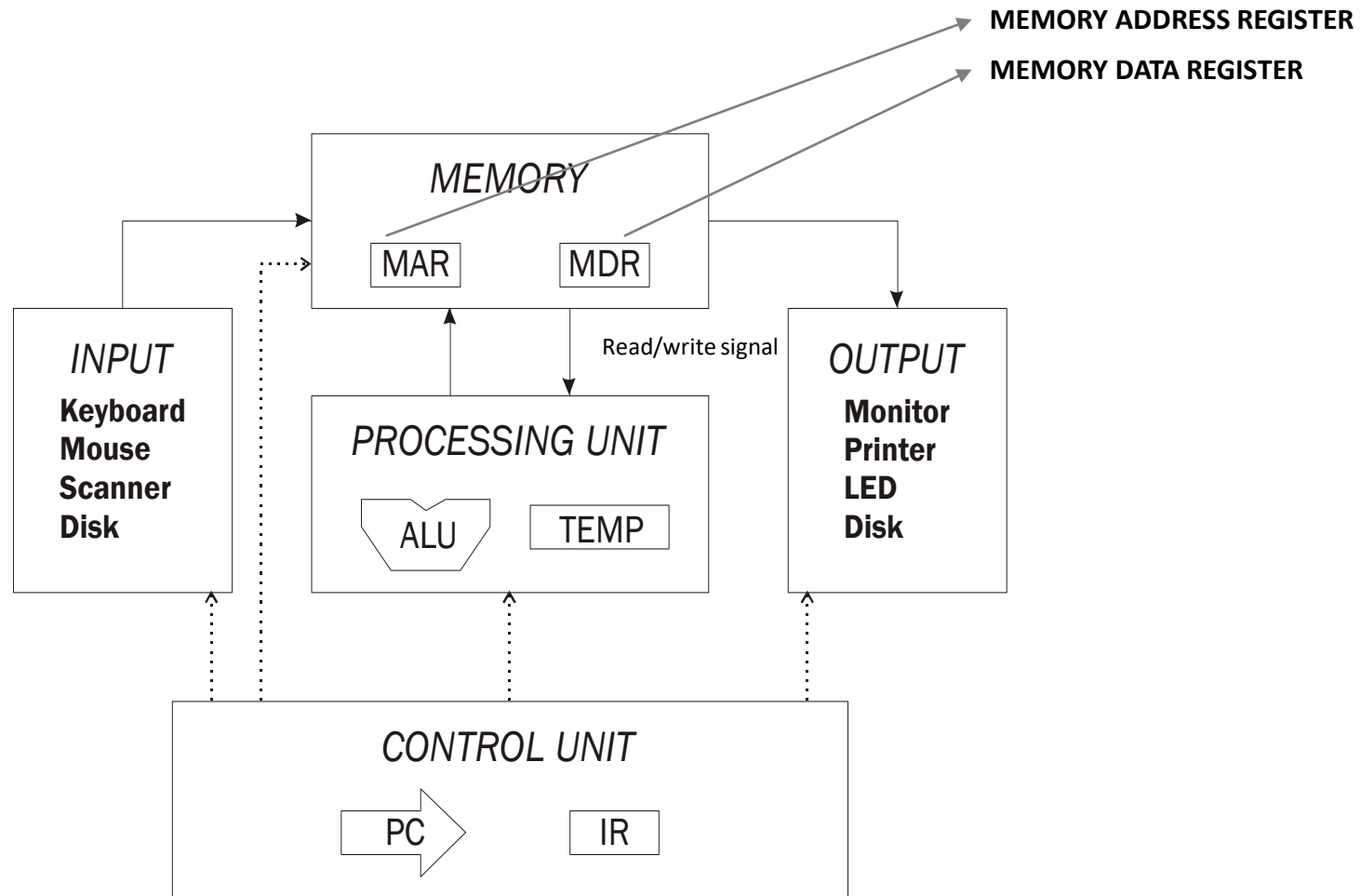


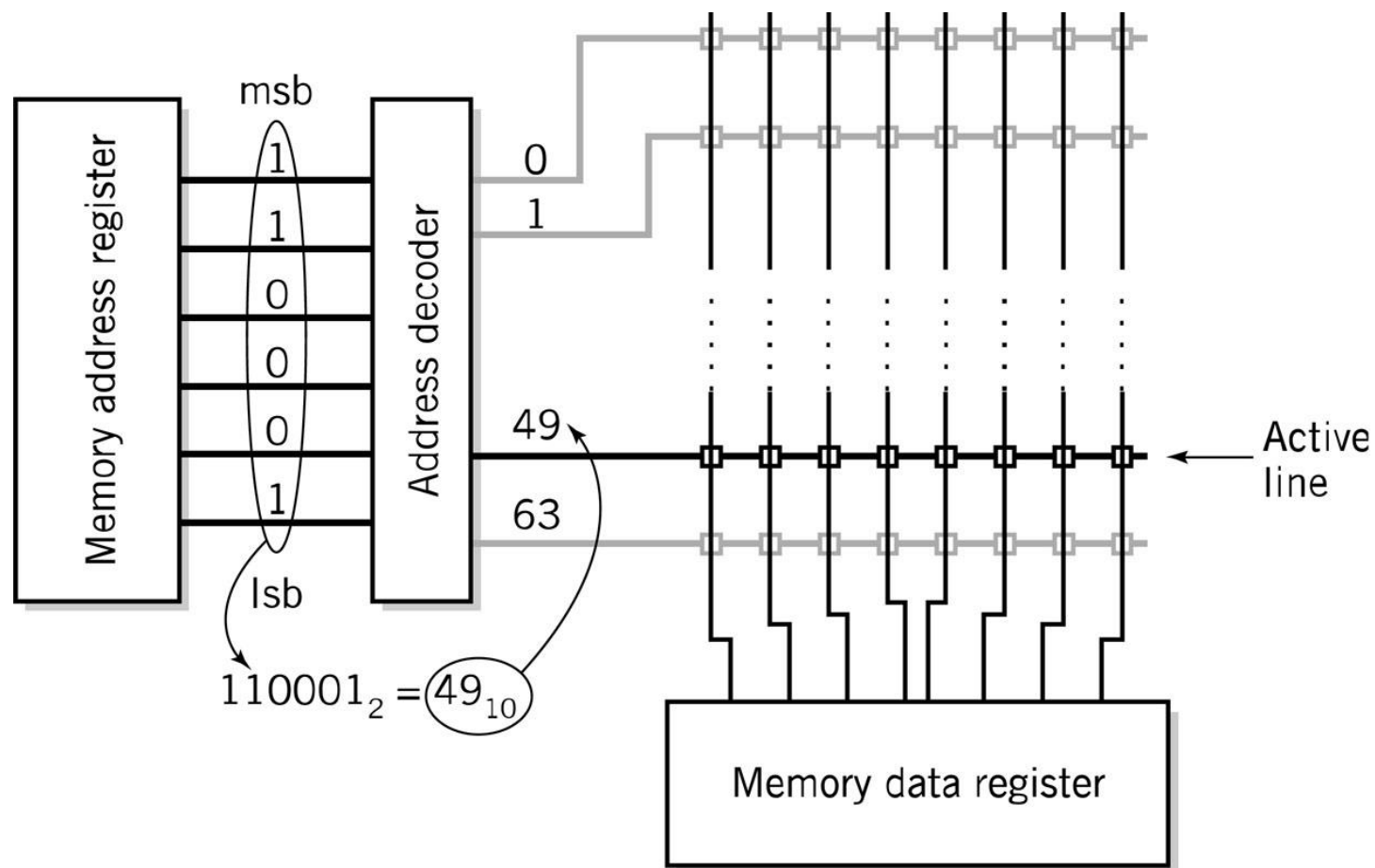
Funciones de la Unidad de control:

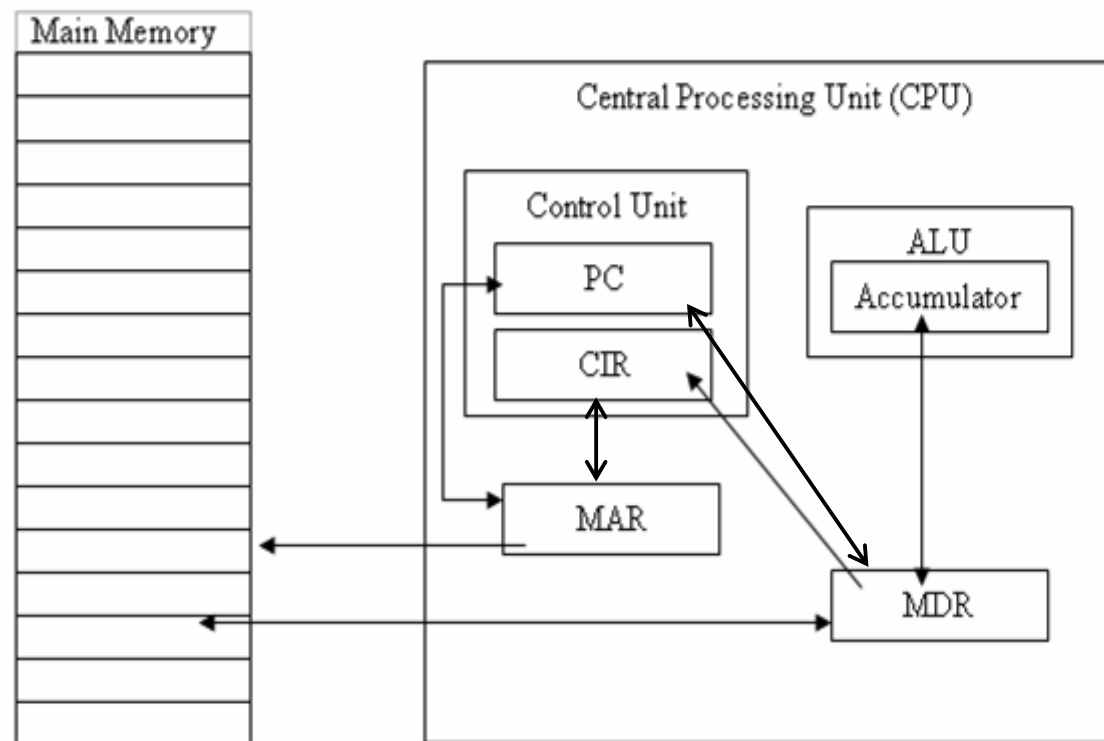
- Lectura instrucciones máquina.
- Interpretación de las instrucciones.
- Lectura datos de memoria.
- Ejecución instrucciones.
- Almacenamiento de resultados.

Registros asociados a la Unidad de control:

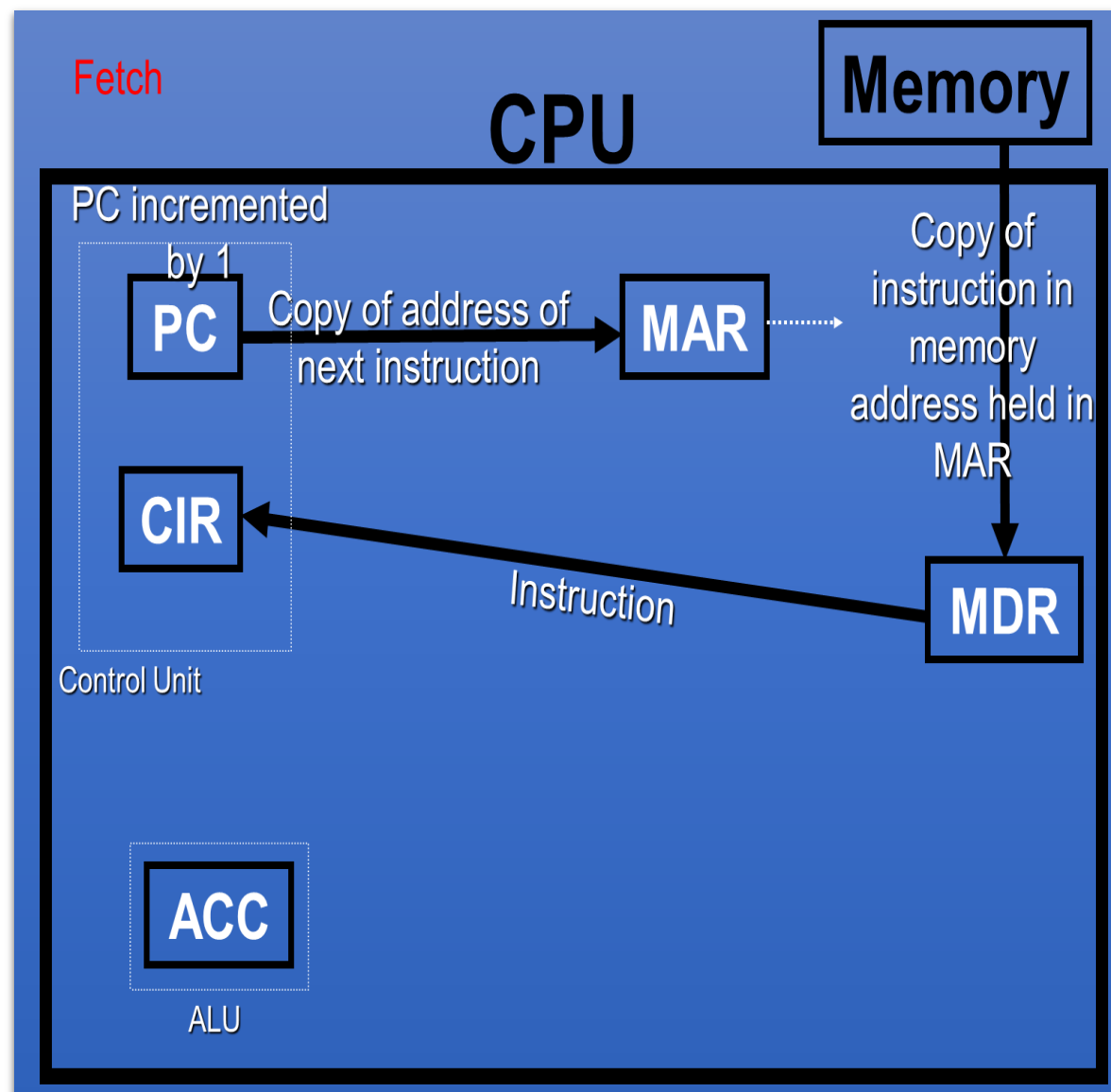
- Contador de programa (PC: program counter): dirección de la siguiente instrucción a ejecutar.
- Puntero de pila (SP: stack pointer).
- Registro de instrucción (IR: instrucción register): instrucción a ejecutar.
- Registro de estado (SR: status register): información producida en la ejecución de las instrucciones más recientes (bits de estado aritmético, bits de interrupción, nivel de ejecución, etc.....).

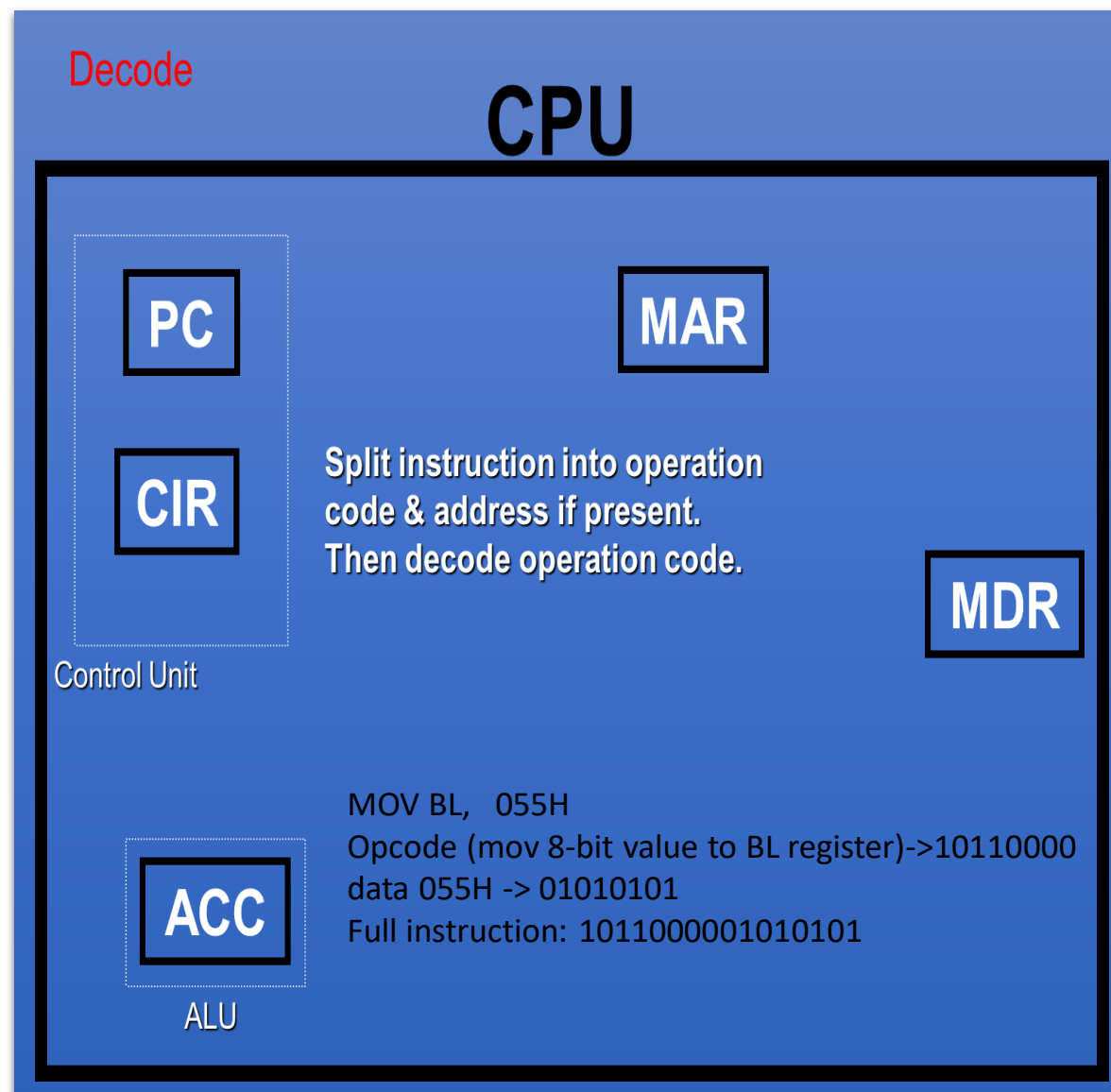


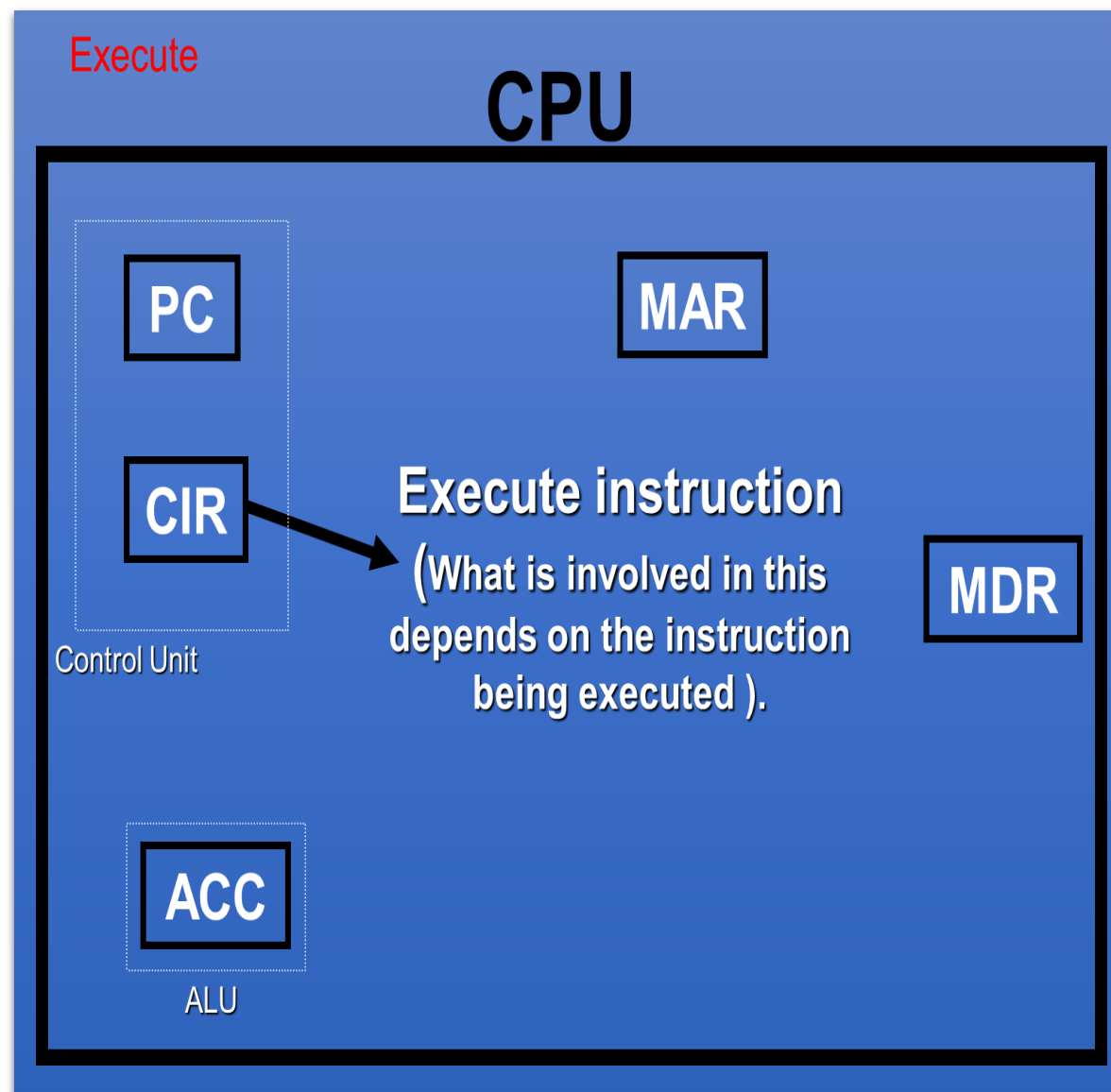




Funciona con el ciclo de: Leer-decodificar-Ejecutar







Instrucciones a Ejecutar

Jump

Input / Load (number directly)

Input / Load (from memory)

Add (a number directly)

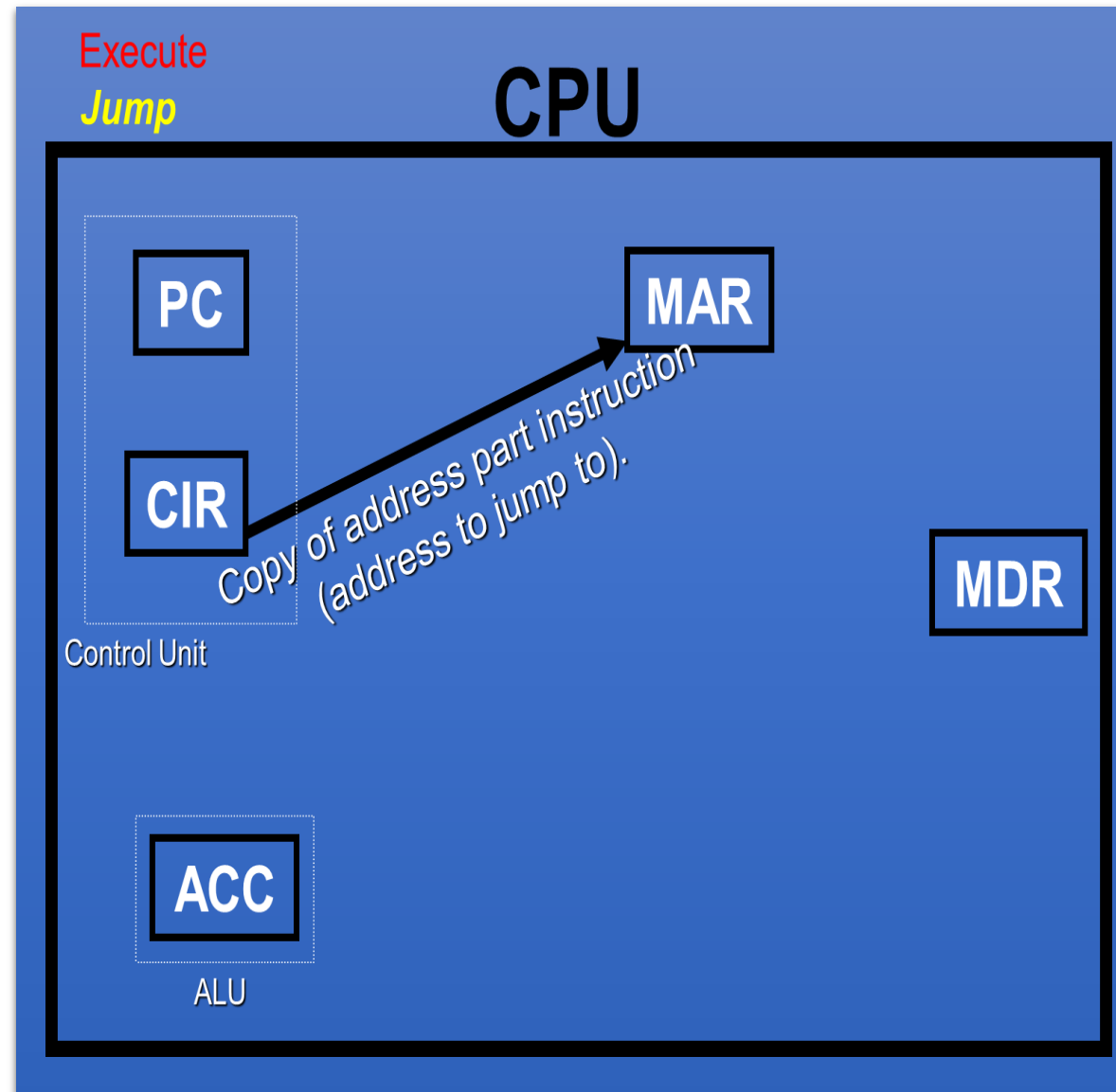
Add (a number from memory)

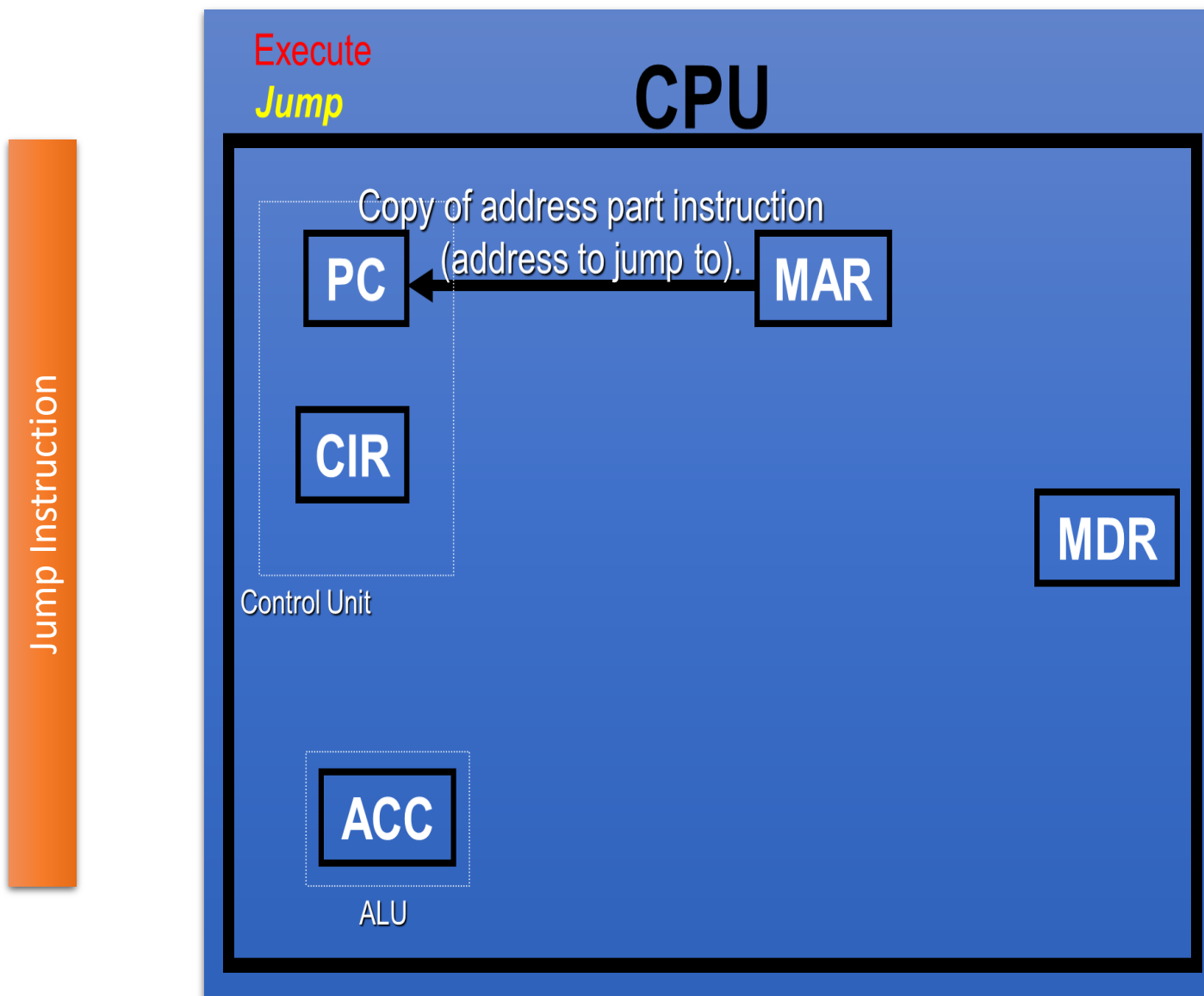
Store

Output (directly from accumulator)

Output (from memory)

Jump Instruction



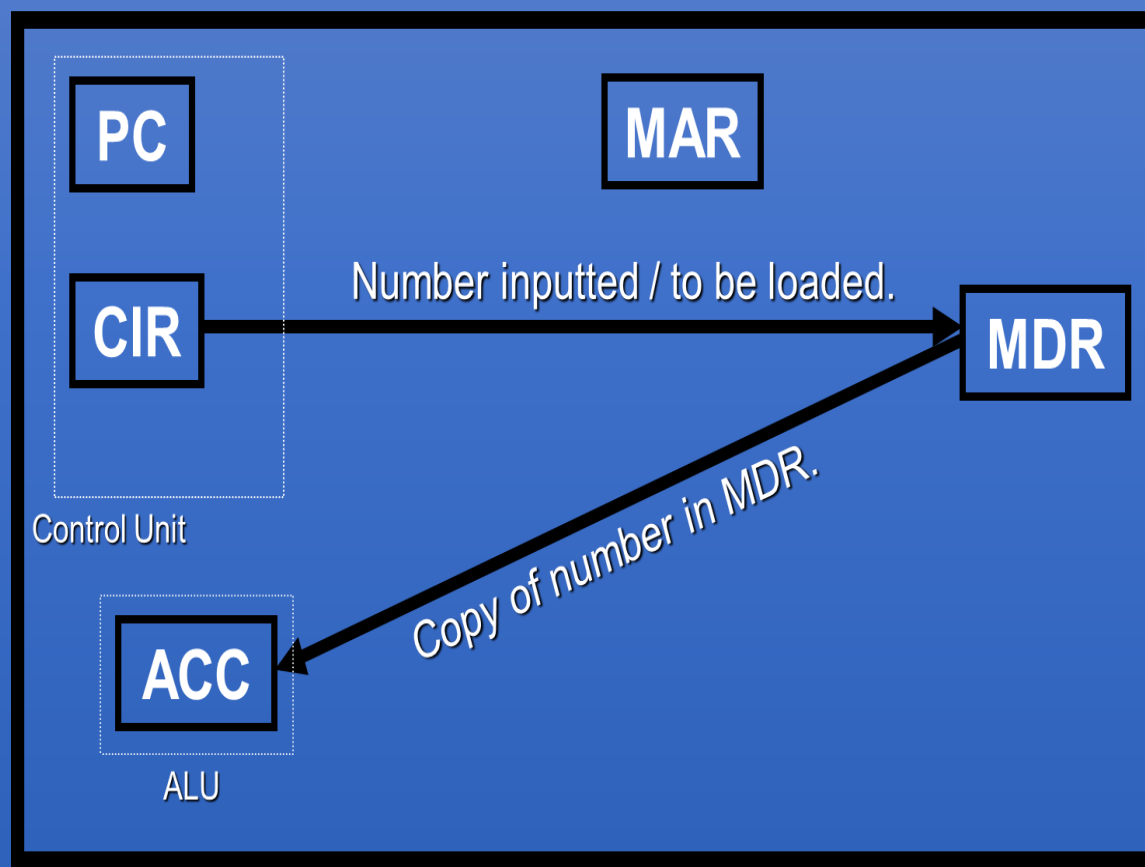


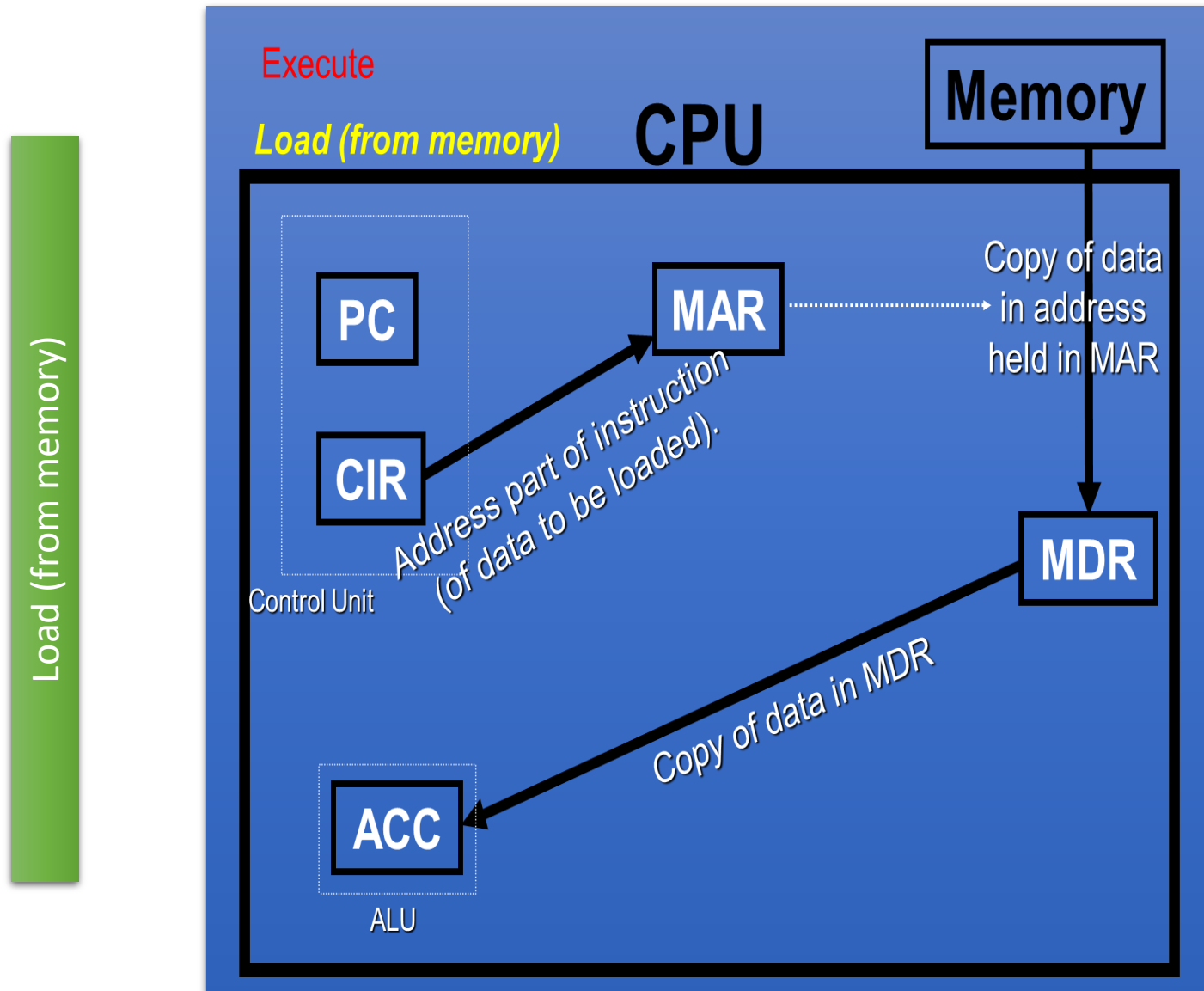
Input/Load (number directly)

Execute

Input / Load (number directly) into accumulator

CPU



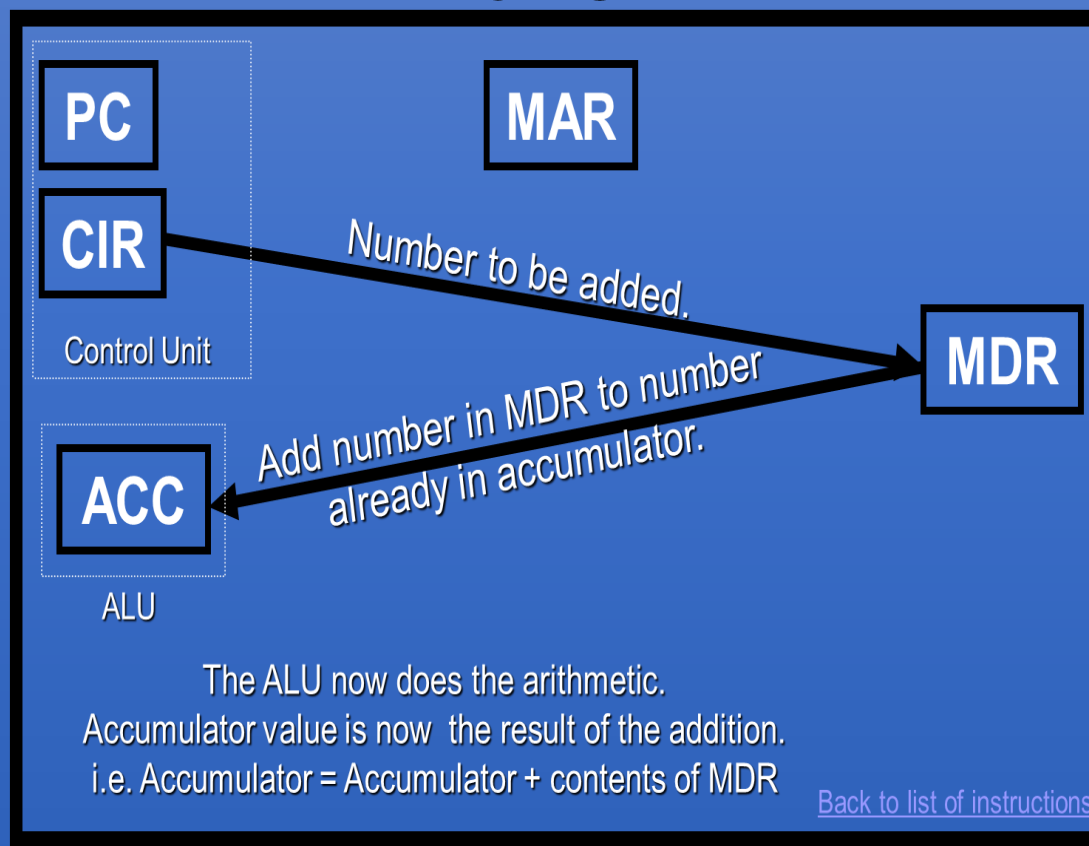


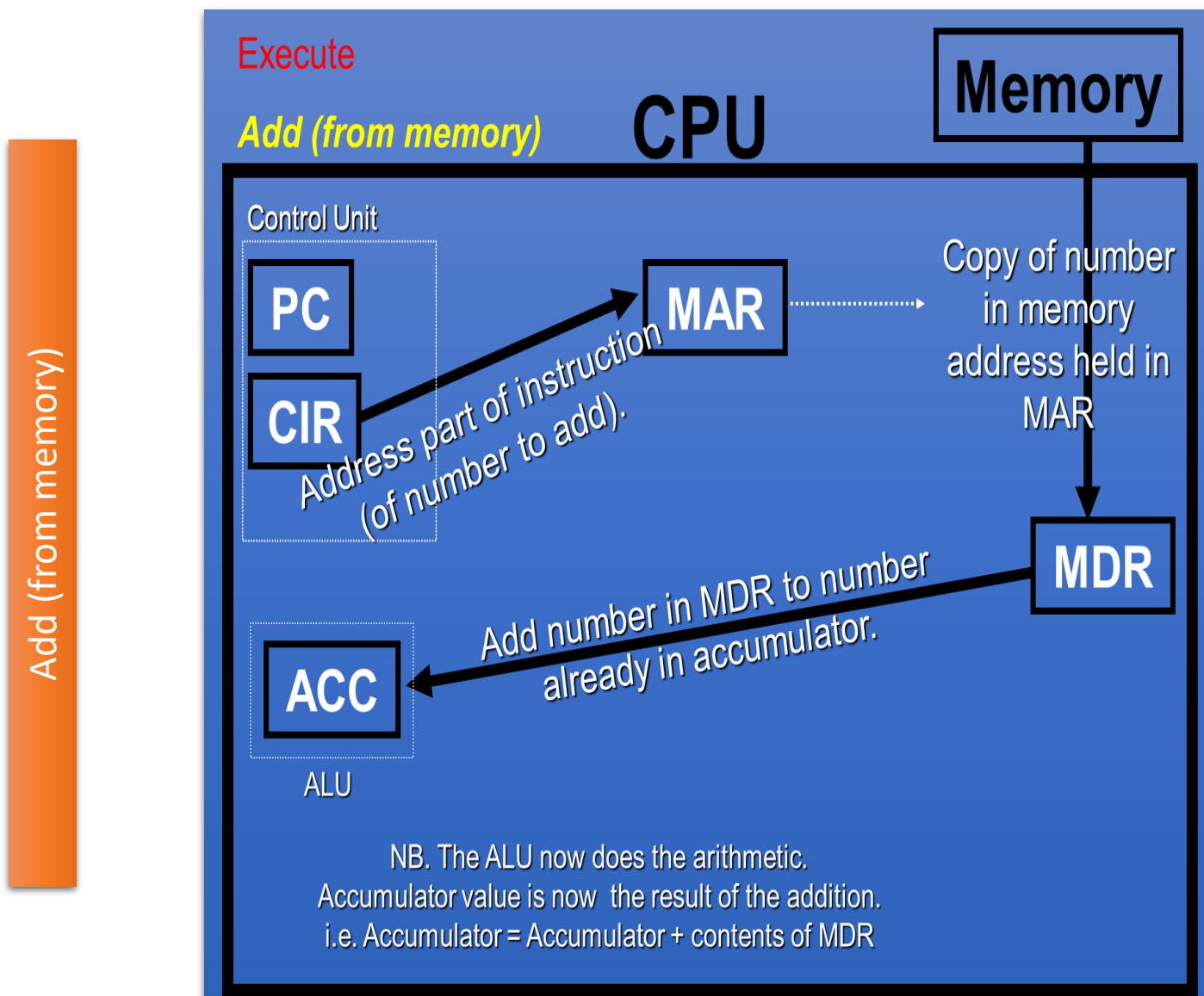
Add (a number directly)

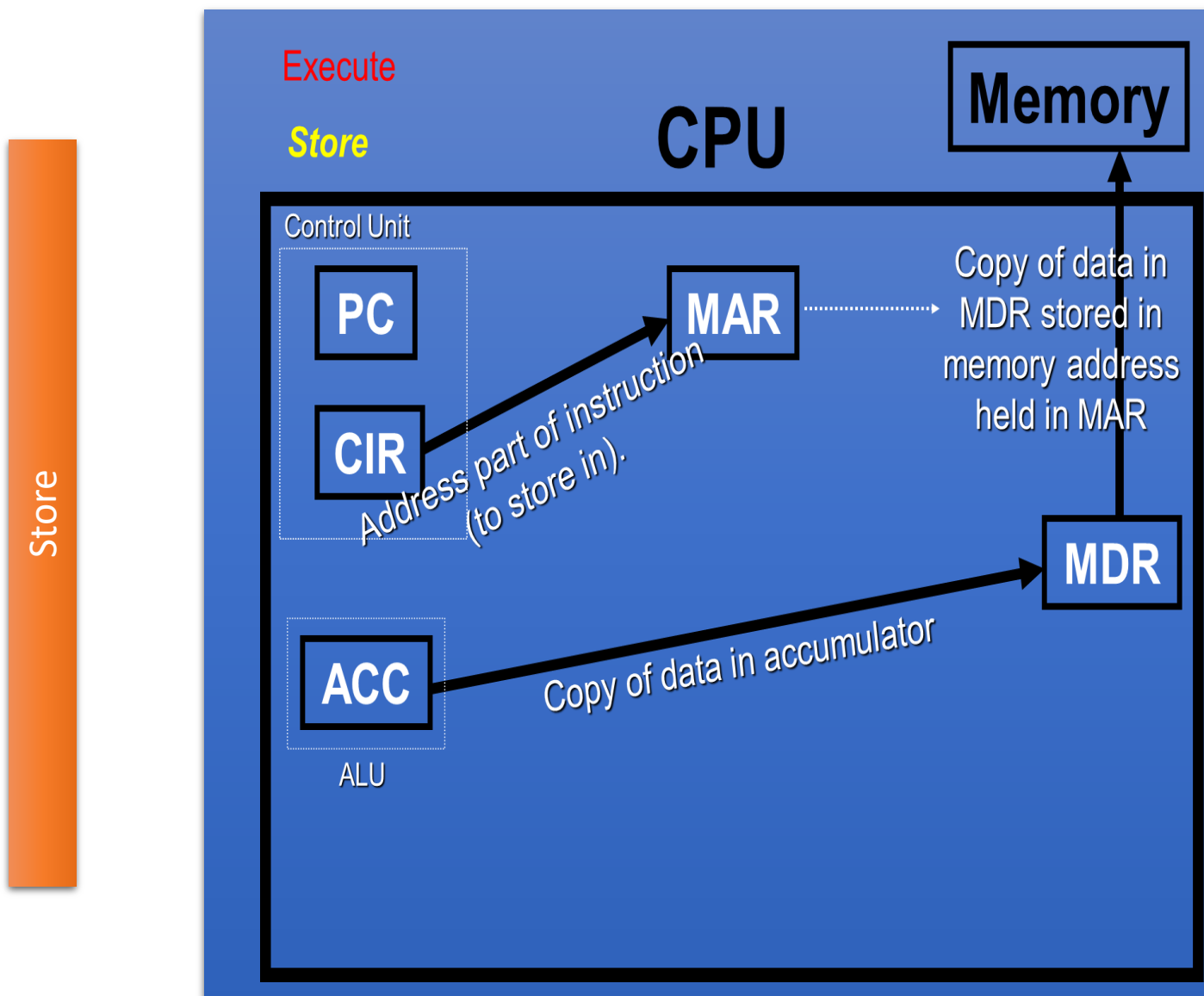
Execute

Add (a number directly)

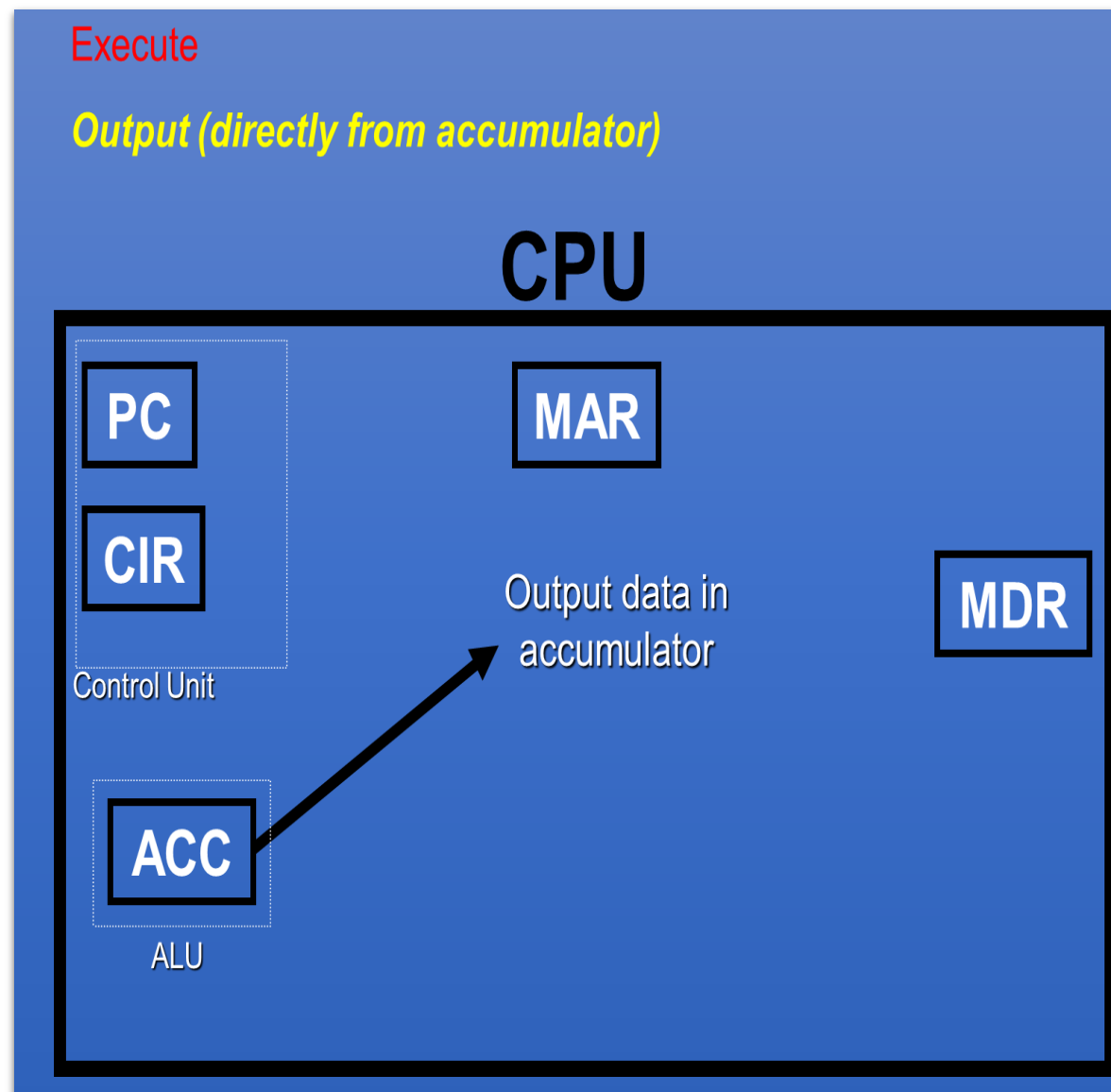
CPU



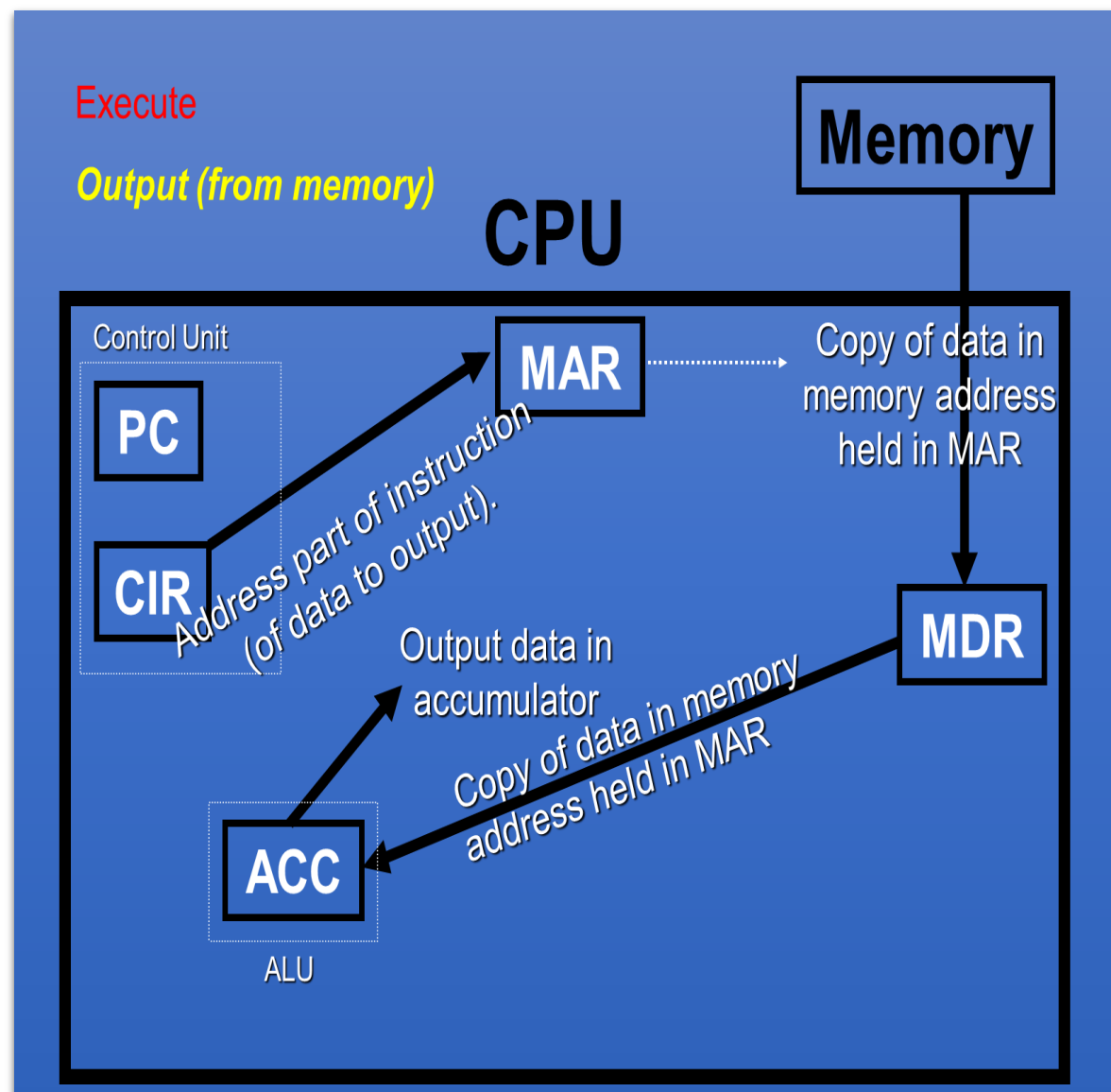


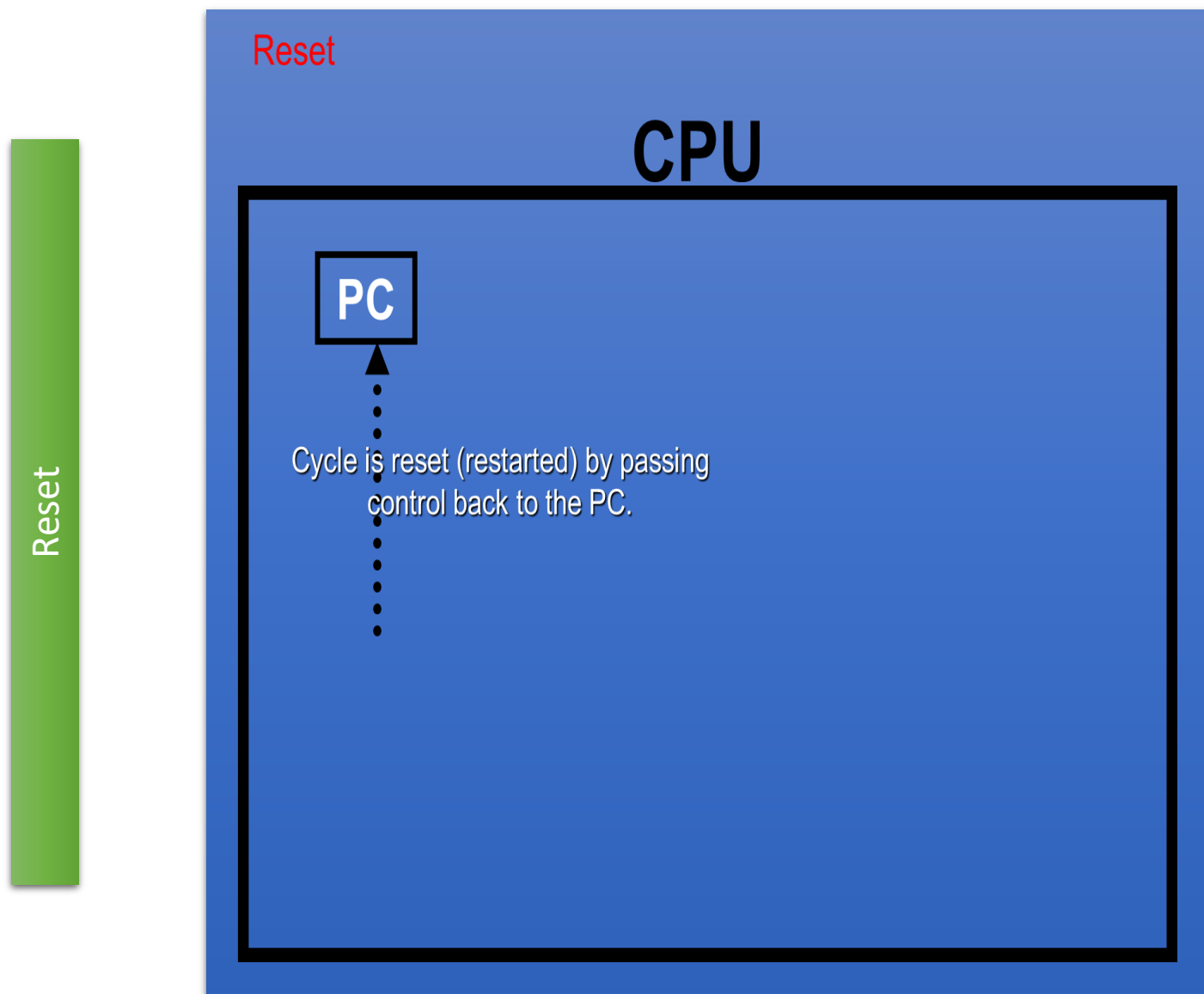


Output (directly from accumulator)



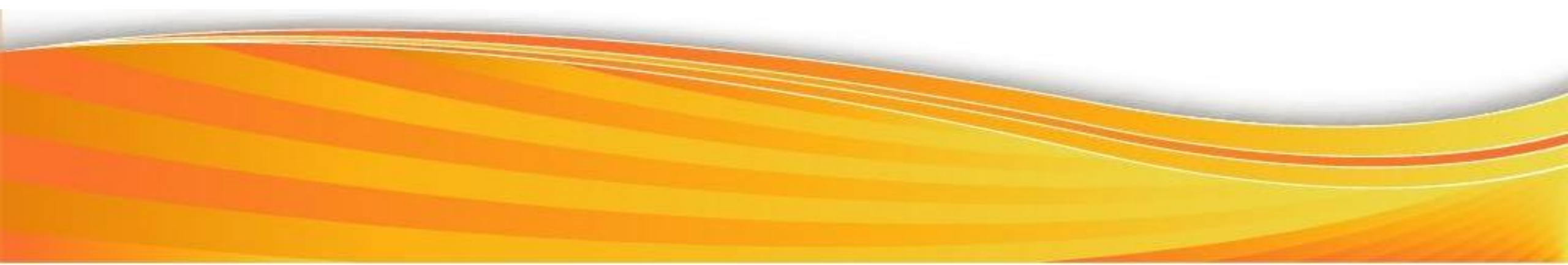
Output (from memory)







Modelos De Programación

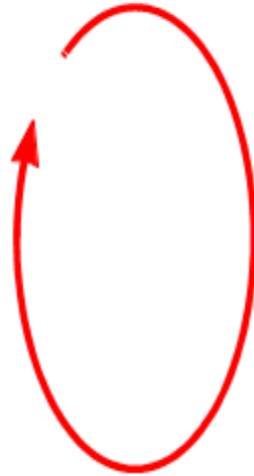


El modelo de programación, a bajo nivel, se basa en:

1. Elementos de almacenamiento disponibles en la ejecución de las instrucciones máquina:

- Registros generales.
- Contador de programa.
- Puntero de la pila.
- Registro de estado.
- Memoria principal.
- Mapa de E/S.

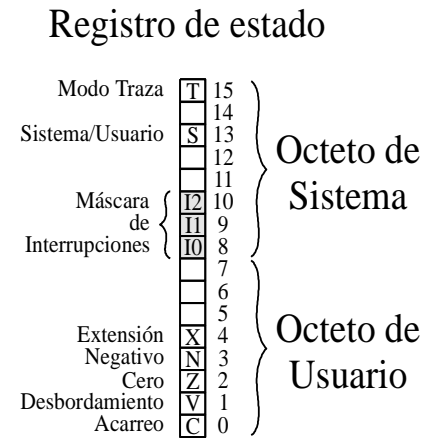
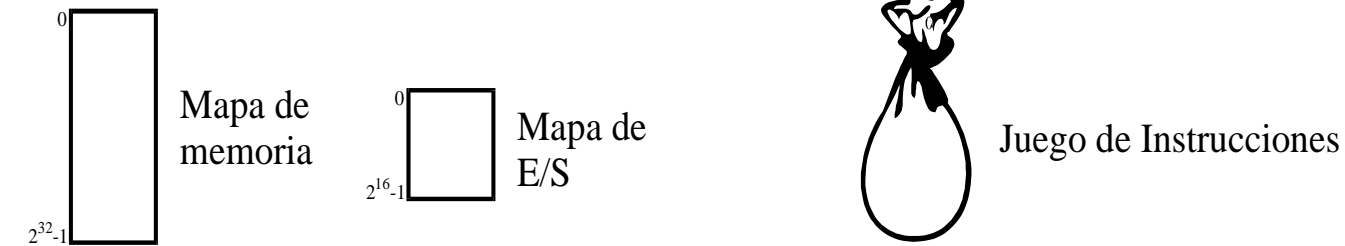
2. ***Juego de instrucciones***: operaciones que puede realizar el ordenador (diferentes modos de direccionamiento).
3. ***Secuencia de funcionamiento***: modo en que se van ejecutando las instrucciones máquina. Consta de una secuencia sencilla que se repite a altísima velocidad; se compone de las siguientes tareas (repetidas en un bucle infinito):

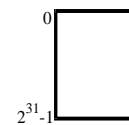


- a) Lectura de la instrucción apuntada por PC**
- b) Incremento del PC**
- c) Ejecución de la instrucción**
 - **secuencia lineal: ejecuta instrucciones consecutivas**
 - **bucle infinito**

4. Niveles de ejecución: generalmente los SSOO disponen de varios niveles de ejecución:

- nivel usuario: algunos recursos no están disponibles, para evitar que algún programa de algún usuario pueda interferir en la ejecución de otros programas o hacer uso de recursos que no le pertenecen (archivos, por ejemplo).
- nivel de núcleo: no hay restricciones.



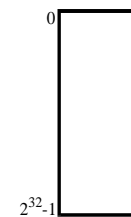


Mapa de memoria

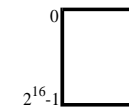


Juego de Instrucciones

Modelo de programación de usuario



Mapa de memoria



Mapa de E/S

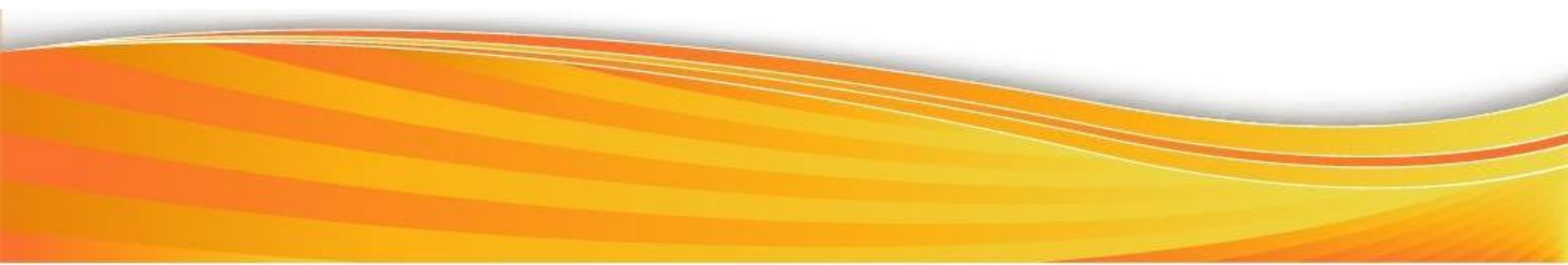


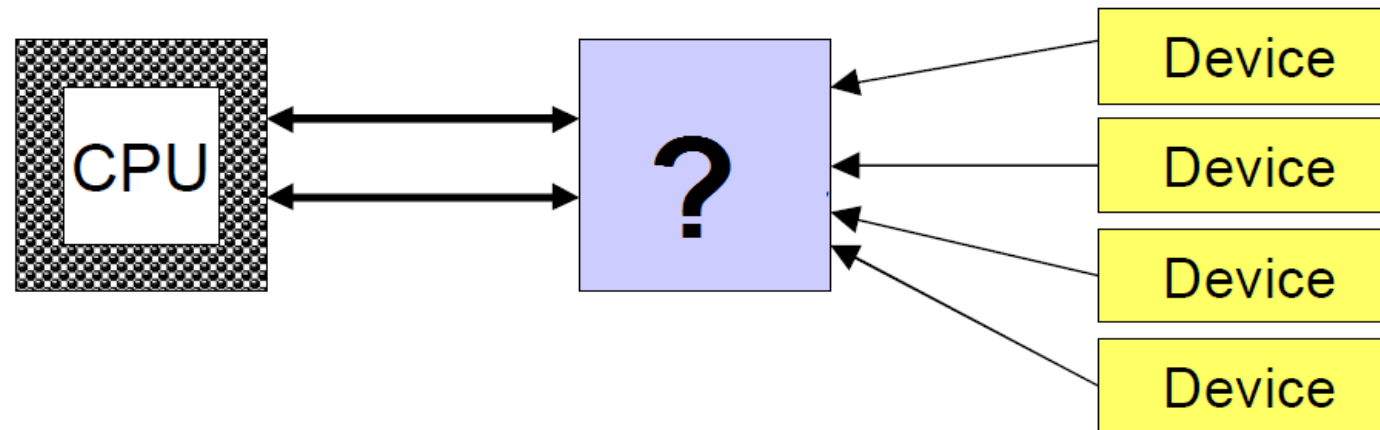
Juego de Instrucciones

Modelo de programación de núcleo



Interrupciones





Can multiple devices interrupt processor?

How does the processor recognize which device is causing the interrupt?

How does it know which ISR(*Interrupt Service Routine*) to execute?

Can interrupts interrupt ISRs?

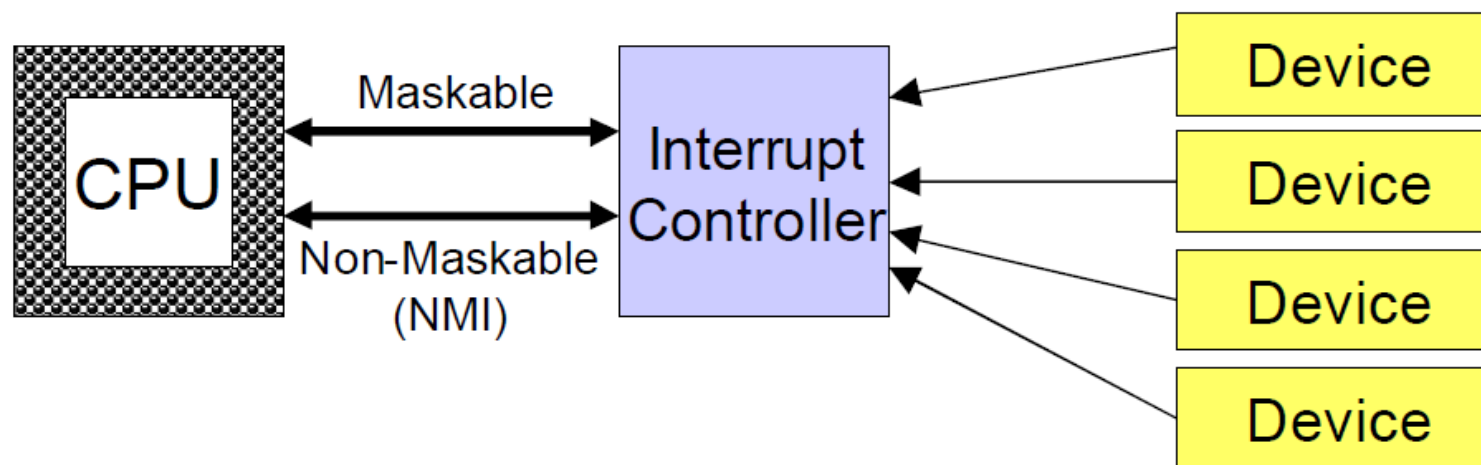
How do we handle simultaneous interrupt requests?

Simple solution

Poll every possible interrupting device and check if status bit indicates something is to be done and then service it

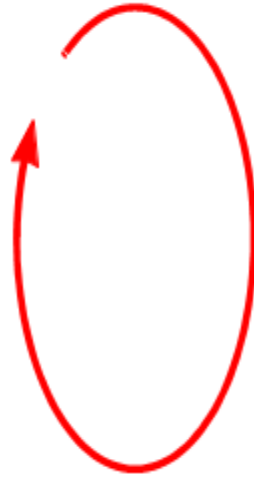
Advantage: simple

Disadvantage: slow



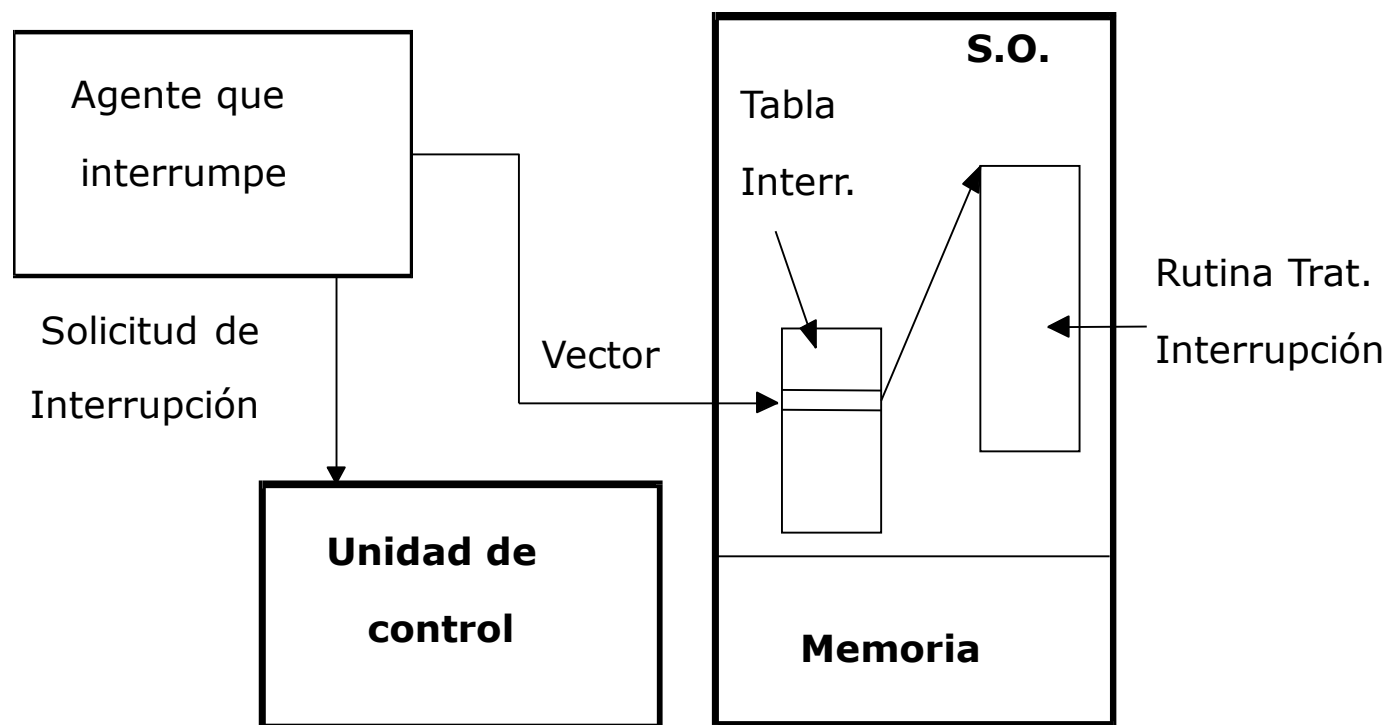
- ❑ Una interrupción se solicita activando una señal que llega a la unidad de control. El elemento que necesita ser atendido y genera la señal se denomina **agente generador**.
- ❑ Ante la solicitud (si esta interrupción está habilitada) se realiza un ciclo de aceptación de interrupción por parte de la Unidad Central. Este ciclo se produce al terminar la ejecución de la instrucción en curso, y consiste en:

- Guardar algunos registros del procesador (PC y registro de estado).
 - Eleva el nivel de ejecución del procesador (nivel de núcleo).
 - Salta al SO, cargando un nuevo valor en PC.
-
- ❑ El SO dispone de rutinas de tratamiento de las interrupciones.



1. Lectura de la instrucción apuntada por PC
2. Incremento del PC
3. Ejecución de la instrucción
4. Si (Existen interrupciones){
5. *//Atender Interrupción*
6. Guardar registros
7. Elevar nivel de ejecución
8. PC = dirección rutina de tratamiento
9. }

- ❑ Mecanismo usual para determinar la dirección de salto:
 1. El agente que interrumpe suministra un vector, que identifica la dirección del programa que desea que le atienda (programa de tratamiento de la interrupción).
 2. La UC toma dicha dirección y la carga en PC, con lo que comienza a ejecutarse el programa de tratamiento.



- ❑ Causas de generación de interrupciones:
 1. Excepciones de programas por problemas de ejecución (desbordamiento, operaciones aritméticas no permitidas, etc.).
 2. Interrupciones de reloj.
 3. Interrupciones de E/S.
 4. Excepciones del HW; por ejemplo, error de paridad en la memoria.
 5. Instrucciones de TRAP.

- ❑ Los ordenadores disponen de una instrucción RETI (instrucción para retorno de interrupción). Esta instrucción restituye los registros de estado y PC, para volver al punto de ejecución donde se generó la interrupción.
- ❑ Hay diferentes señales que solicitan interrupciones, con diferentes prioridades. Algunas de las señales se pueden inhibir explícitamente (por lo que no serán atendidas).



Reloj

Tres visiones del reloj

Oscilador que gobierna las fases de las instrucciones de máquina: 1000 MHZ (frecuencia).

Interrupción periódica. La señal de oscilador se divide para generar interrupciones cada cierto periodo de tiempo. Objetivo: conseguir que el SO pase a ejecutar de forma sistemática cada cierto intervalo de tiempo, evitando así que algún programa acapare los recursos del sistema.

Contador → fecha y hora

- Contabiliza unidades de tiempo (p.e. segundos) desde un instante (p.e. 0 h del 1 de enero de 1990).
- Esta cuenta la puede hacer:
HW especial con batería
SO.

Esquema general con las tres visiones del reloj

