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

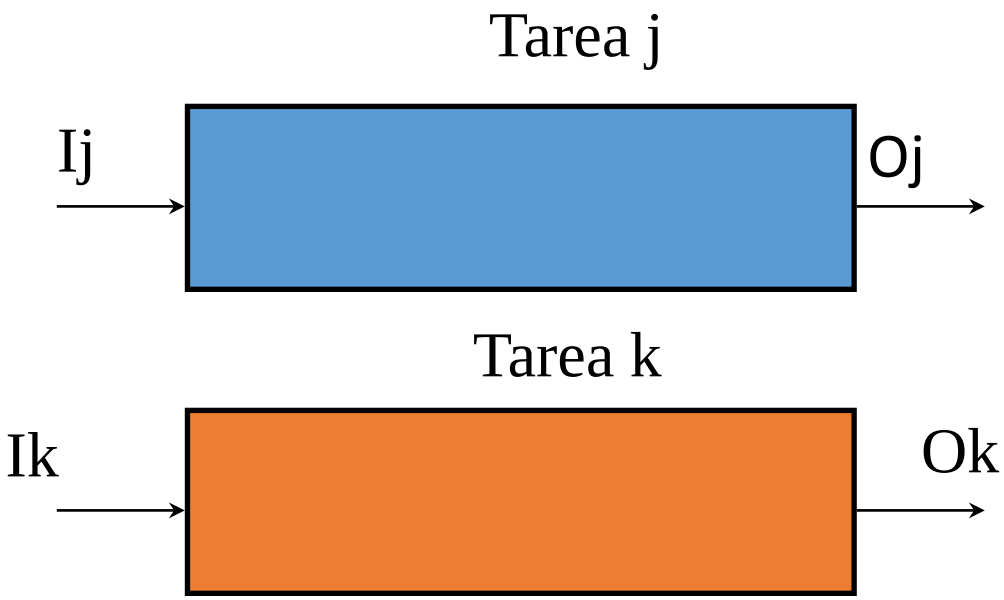
Conceptos Arquitectónicos de la computadora

Contenido

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

Estructura y Funcionamiento Del Ordenador

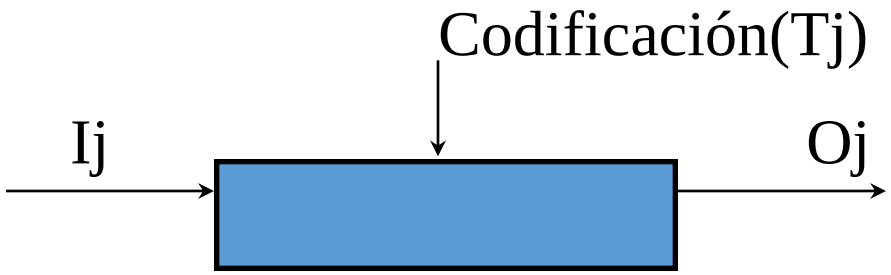
Computación digital personalizada



Circuitos diferentes para cada tarea

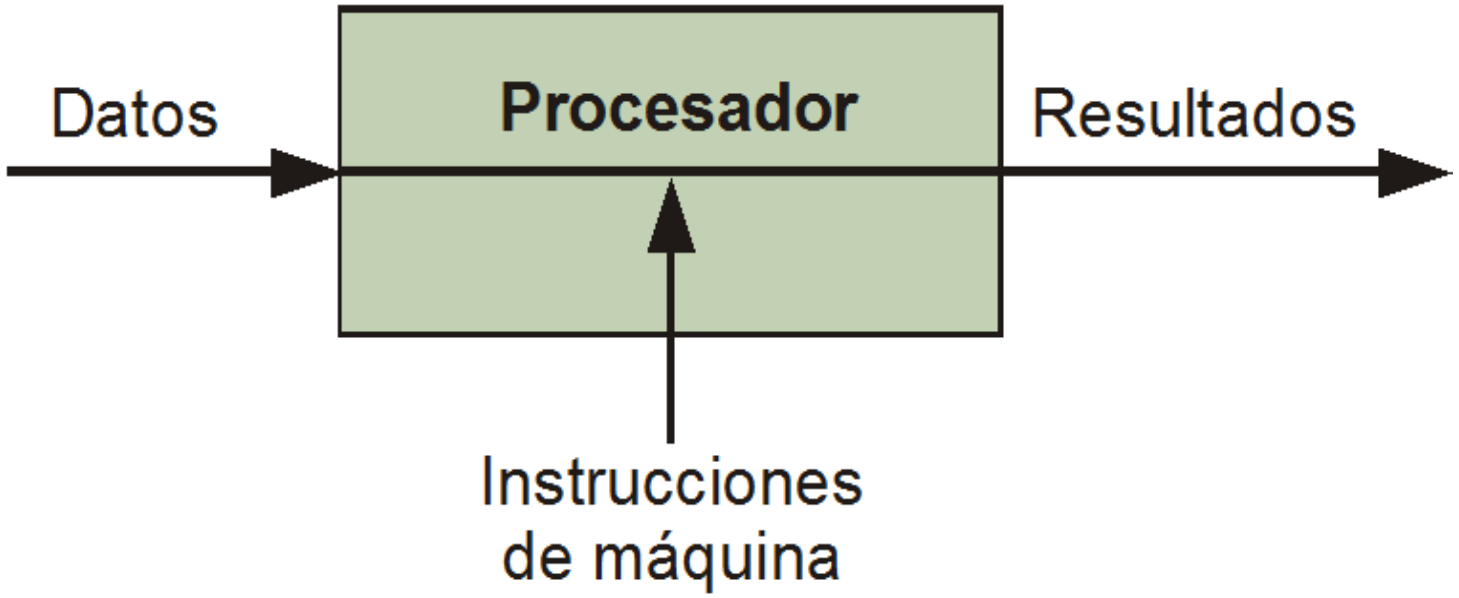
Computación de programas almacenados

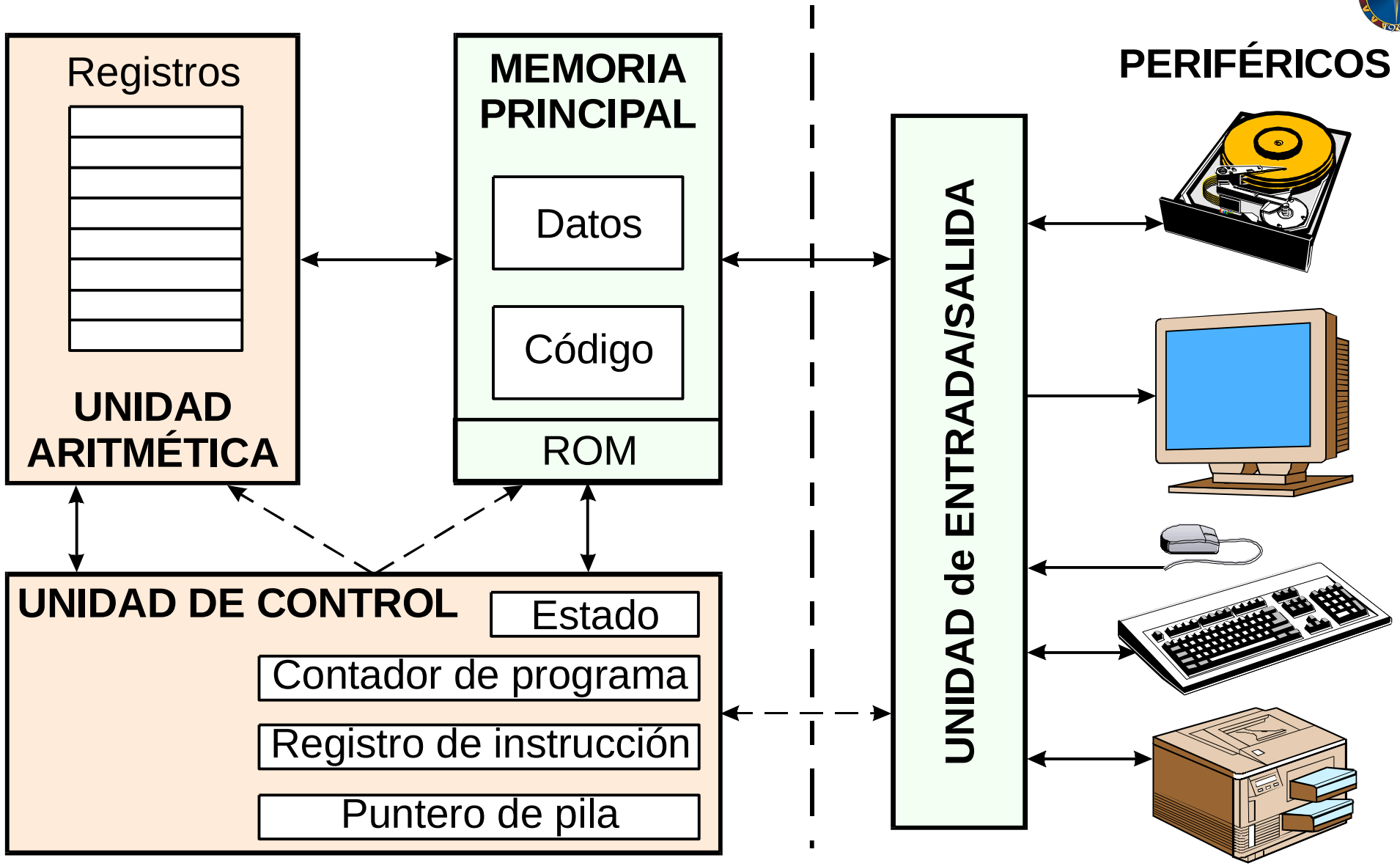
- *Von Neumann* mostró la existencia de una maquina universal (hardware) que puede ser personalizada para realizar diferentes tareas mediante entradas de control
- El *Software* es la codificación de la tarea para controlar la maquina
- Única unidad de almacenamiento dado que instrucciones y datos no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



Único circuito

Estructura general a alto nivel





- ❑Monoprocesador
- ❑Multiprocesador
- ❑Multicomputador

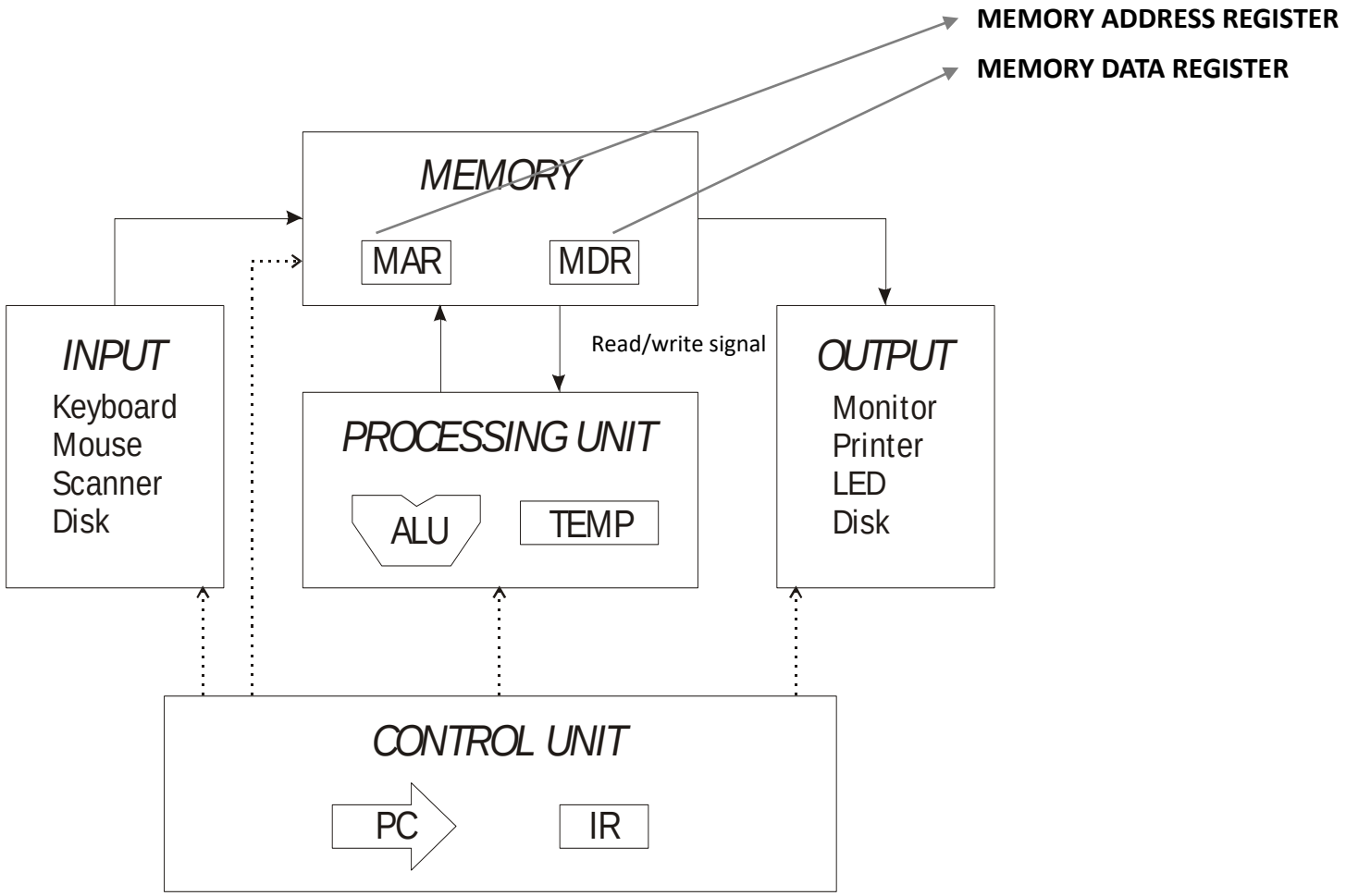
Modelo De Von Neumann

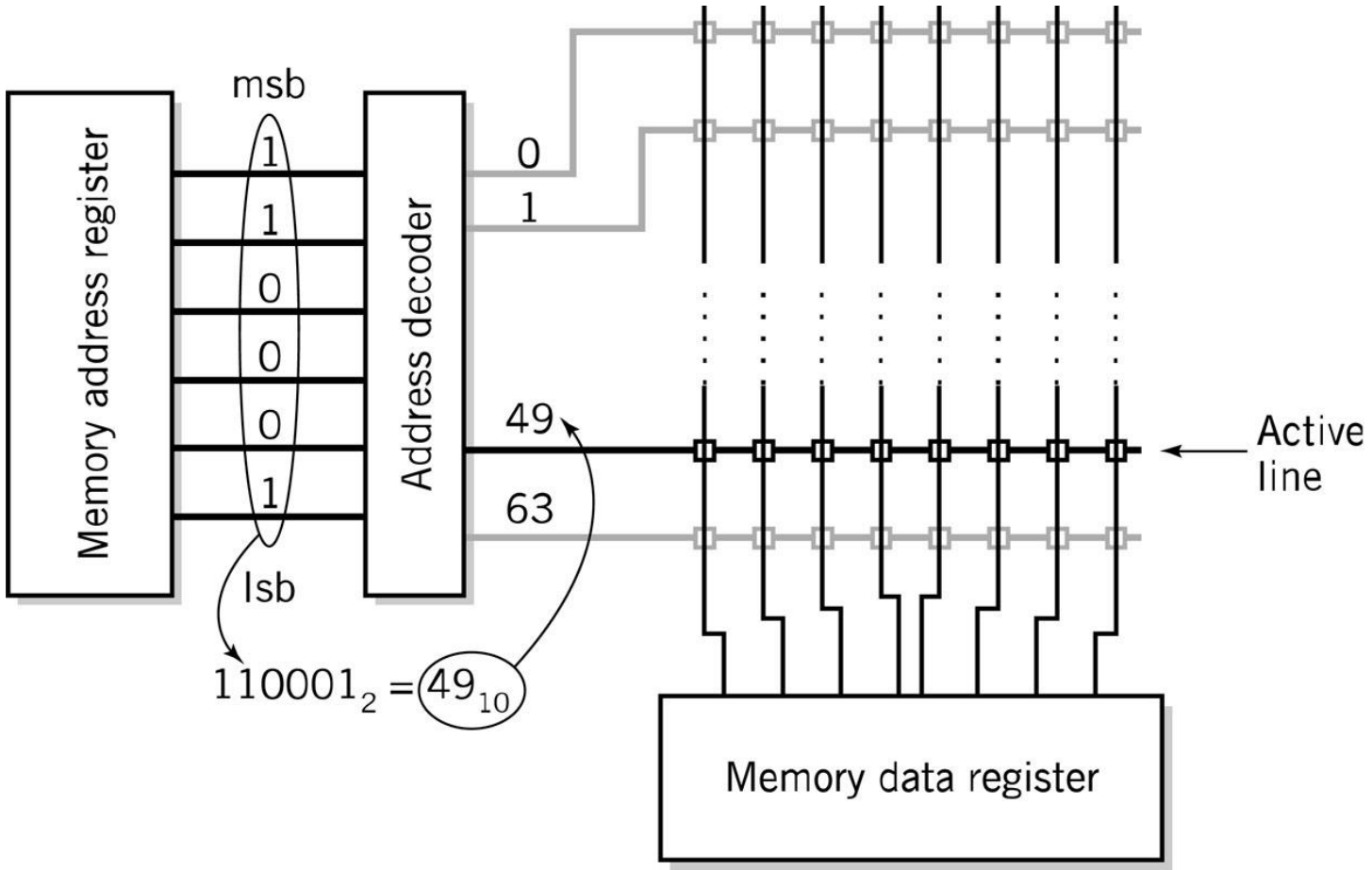
Funciones de la Unidad de control:

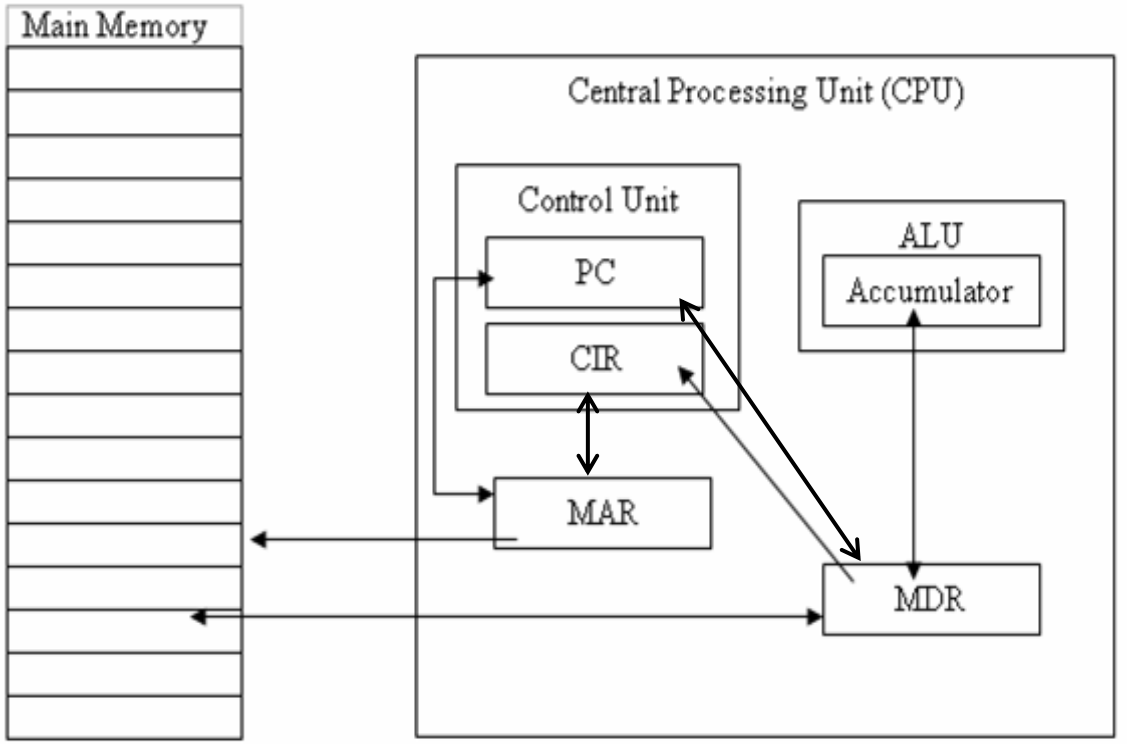
- Lectura instrucciones máquina.
- Interpretación de las instrucciones.
- Lectura datos de memoria.
- Ejecución instrucciones.
- Almacenamiento de resultados.

Registros asociados a la Unidad de control:

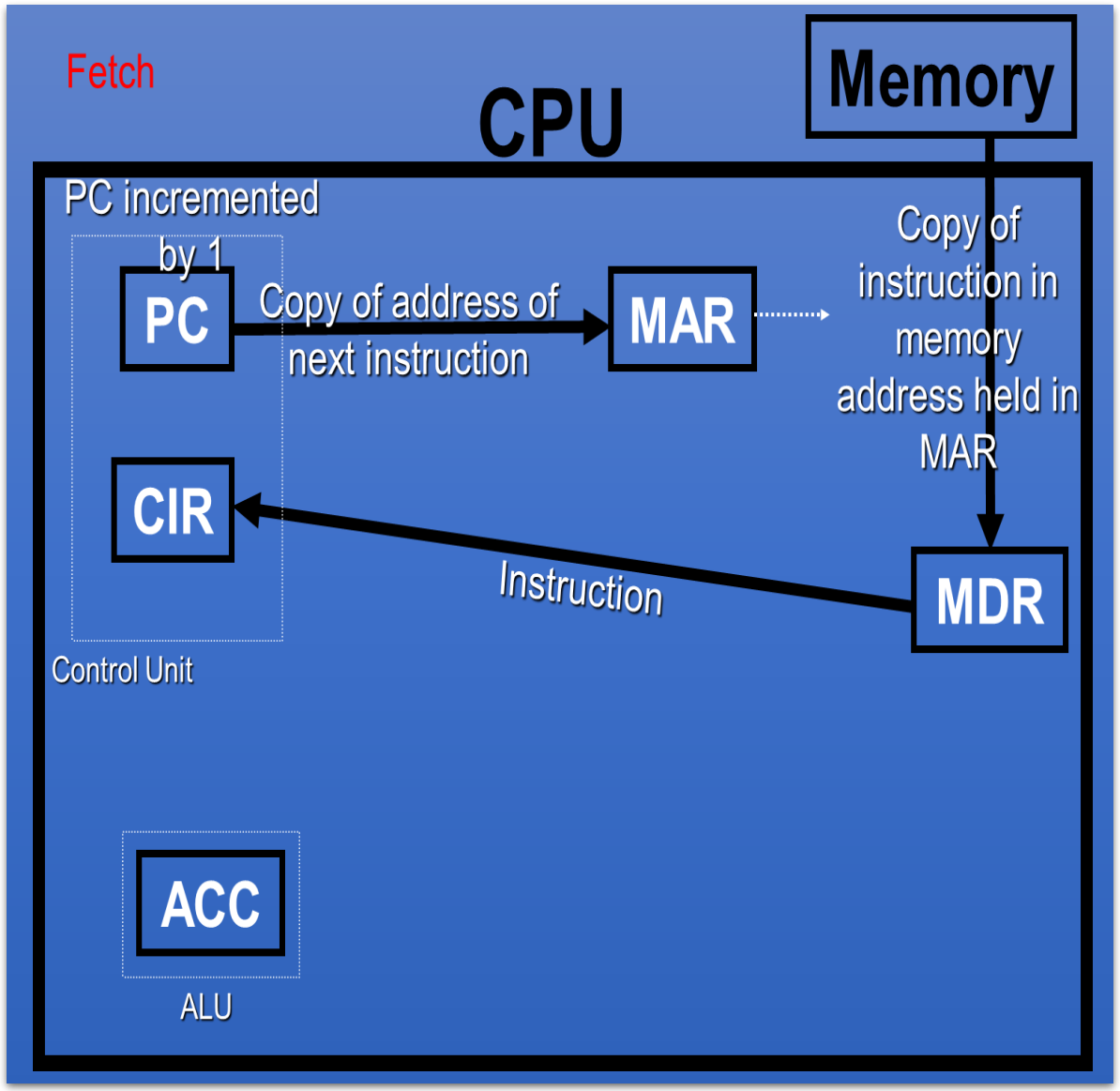
- Contador de programa (PC: program counter): dirección de la siguiente instrucción a ejecutar.
- Puntero de pila (SP: stack pointer).
- Registro de instrucción (IR: instrucción register):instrucción a ejecutar.
- Registro de estado (SR: status register): información producida en la ejecución de las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

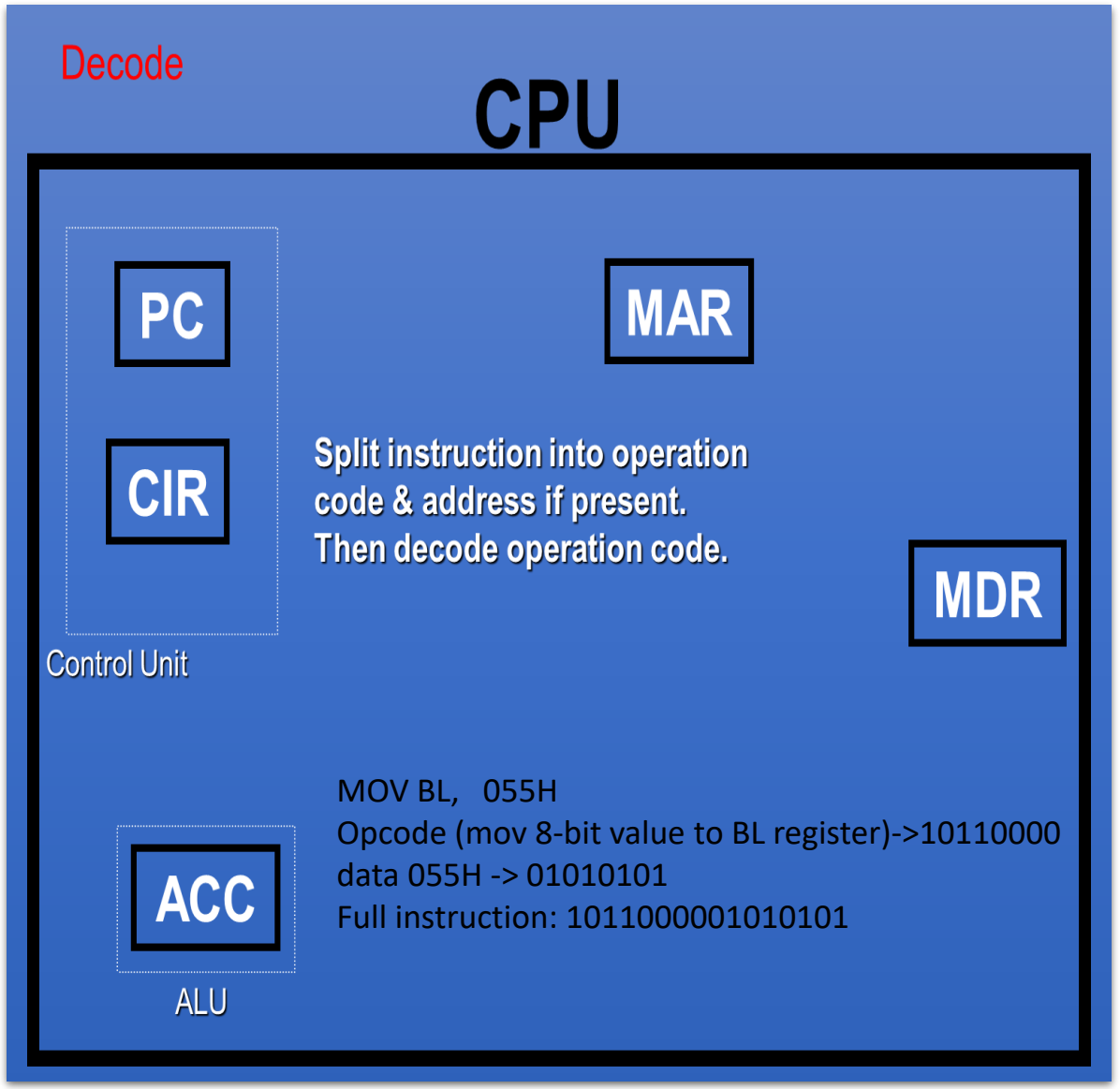

MEMORY ADDRESS REGISTER
MEMORY DATA REGISTER
MEMORY
MAR
MDR
INPUT
Keyboard
Mouse
Scanner
Disk
Read/write signal
OUTPUT
Monitor
Printer
LED
Disk
PROCESSING UNIT
ALU
TEMP
CONTROL UNIT
PC
IR

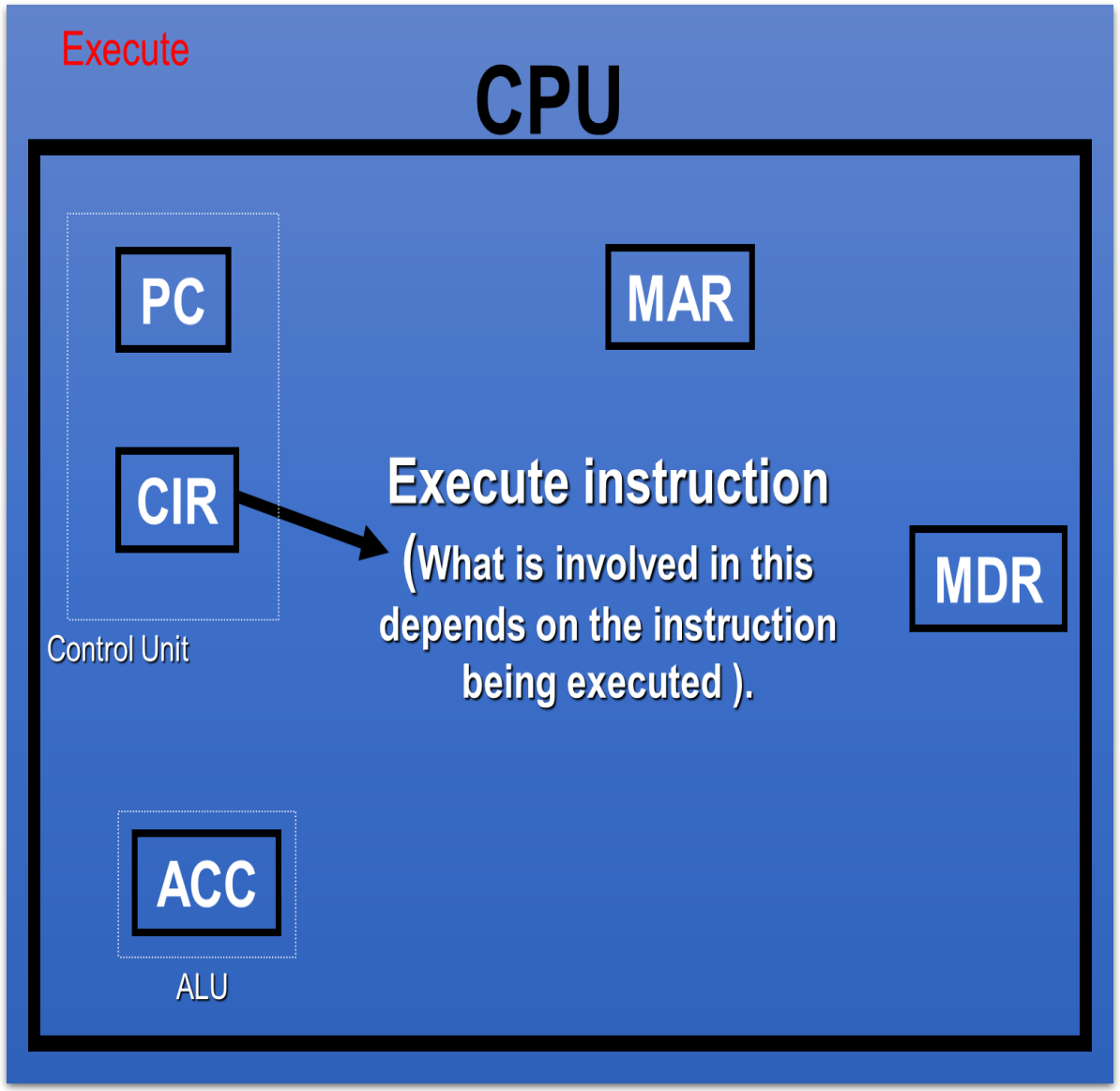

Memory address register
msb
1
1
0
0
0
1
lsb
Address decoder
0
1
49
63
Active line
$110001_2 = 49_{10}$
Memory data register



Funciona con el ciclo de: Leer-decodificar-Ejecutar


Fetch
CPU
Memory
PC incremented by 1
PC
Copy of address of next instruction
MAR
Copy of instruction in memory address held in MAR
CIR
Instruction
MDR
Control Unit
ACC
ALU


Decode
CPU
PC
MAR
CIR
Split instruction into operation code & address if present.
Then decode operation code.
MDR
Control Unit
ACC
ALU
MOV BL, 055H
Opcode (mov 8-bit value to BL register)->10110000
data 055H -> 01010101
Full instruction: 1011000001010101


Execute
CPU
PC
MAR
CIR
Execute instruction
(What is involved in this depends on the instruction being executed).
MDR
Control Unit
ACC
ALU

Instrucciones a Ejecutar

Jump

Input / Load (number directly)

Input / Load (from memory)

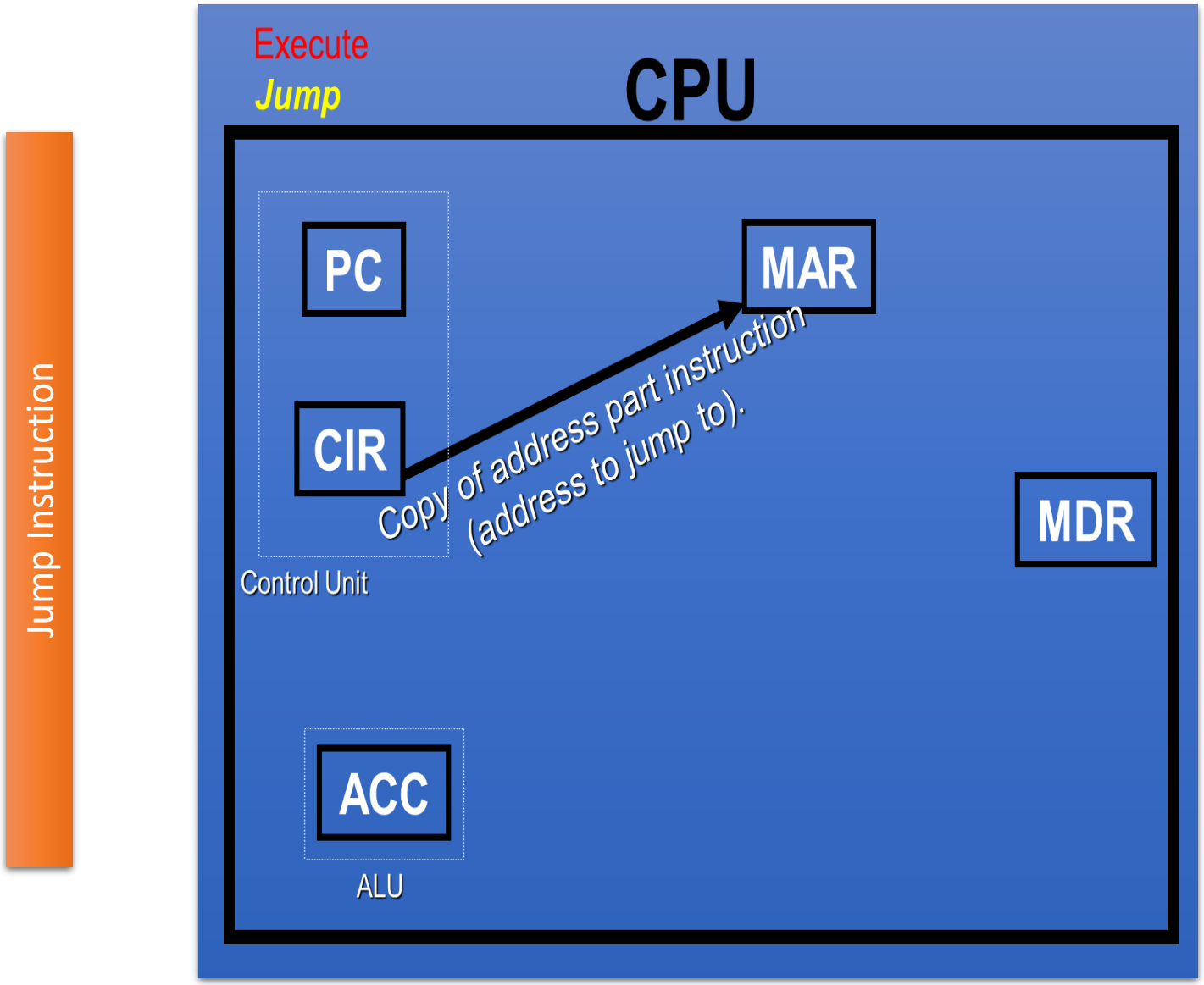
Add (a number directly)

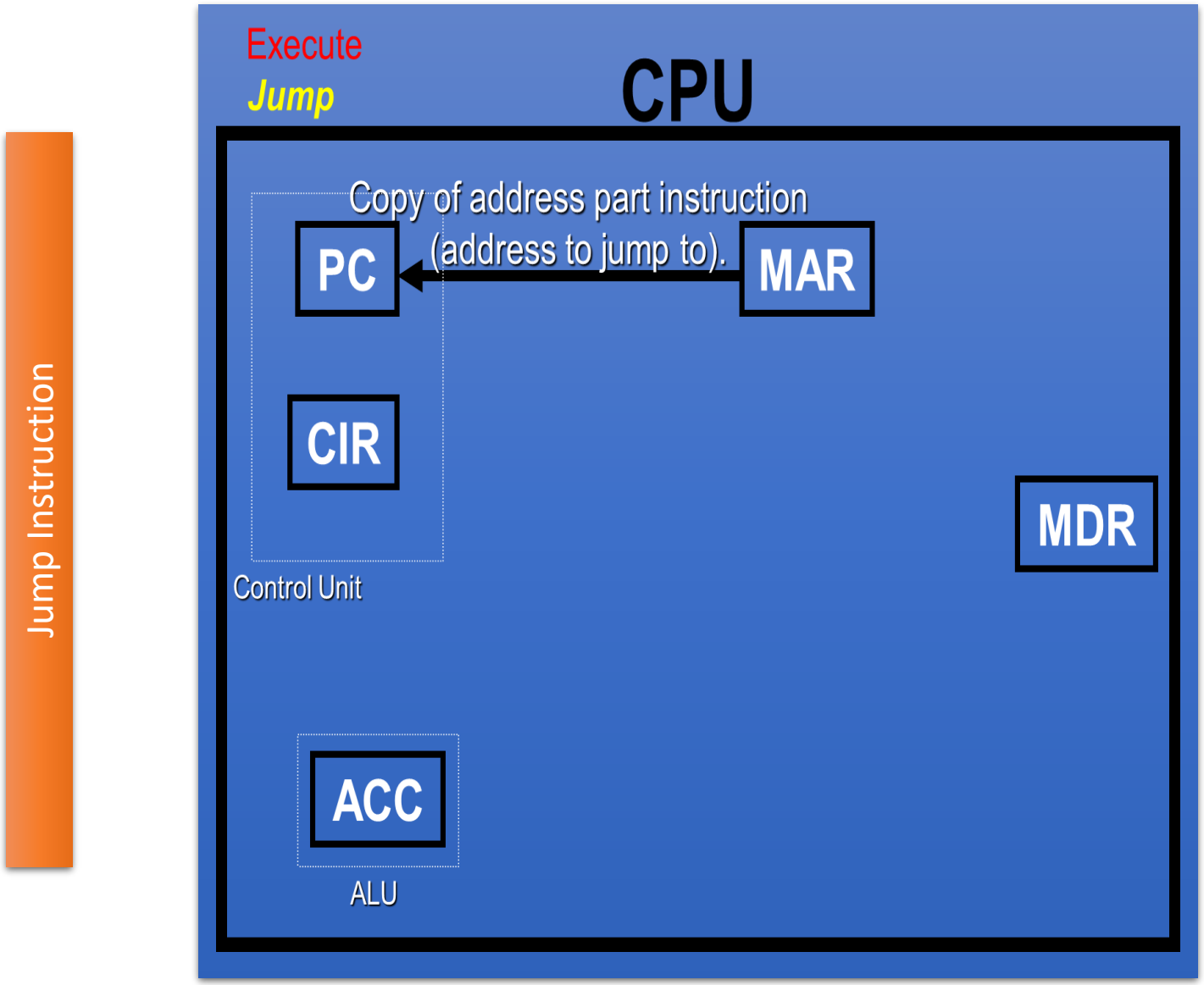
Add (a number from memory)

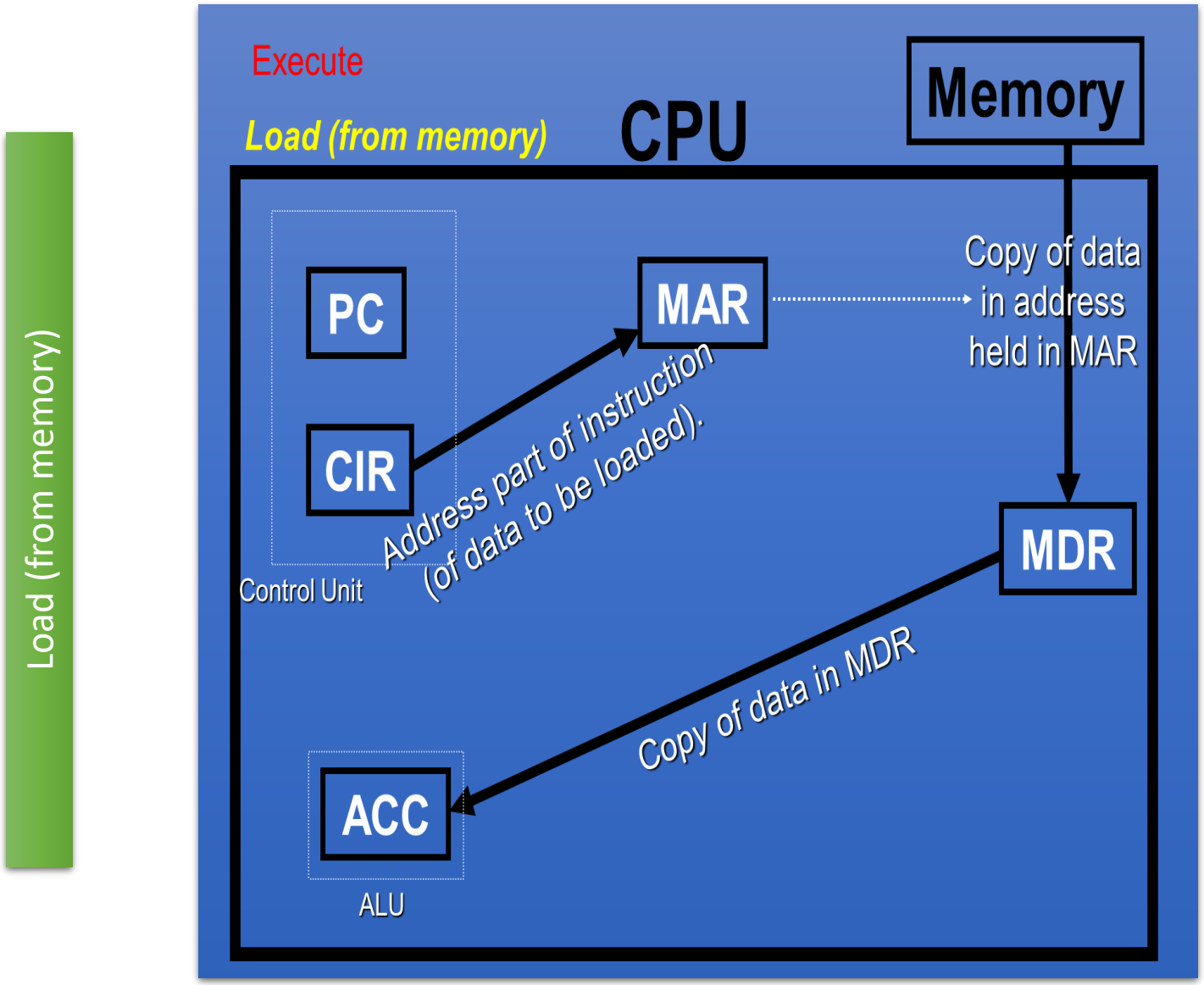
Store

Output (directly from accumulator)

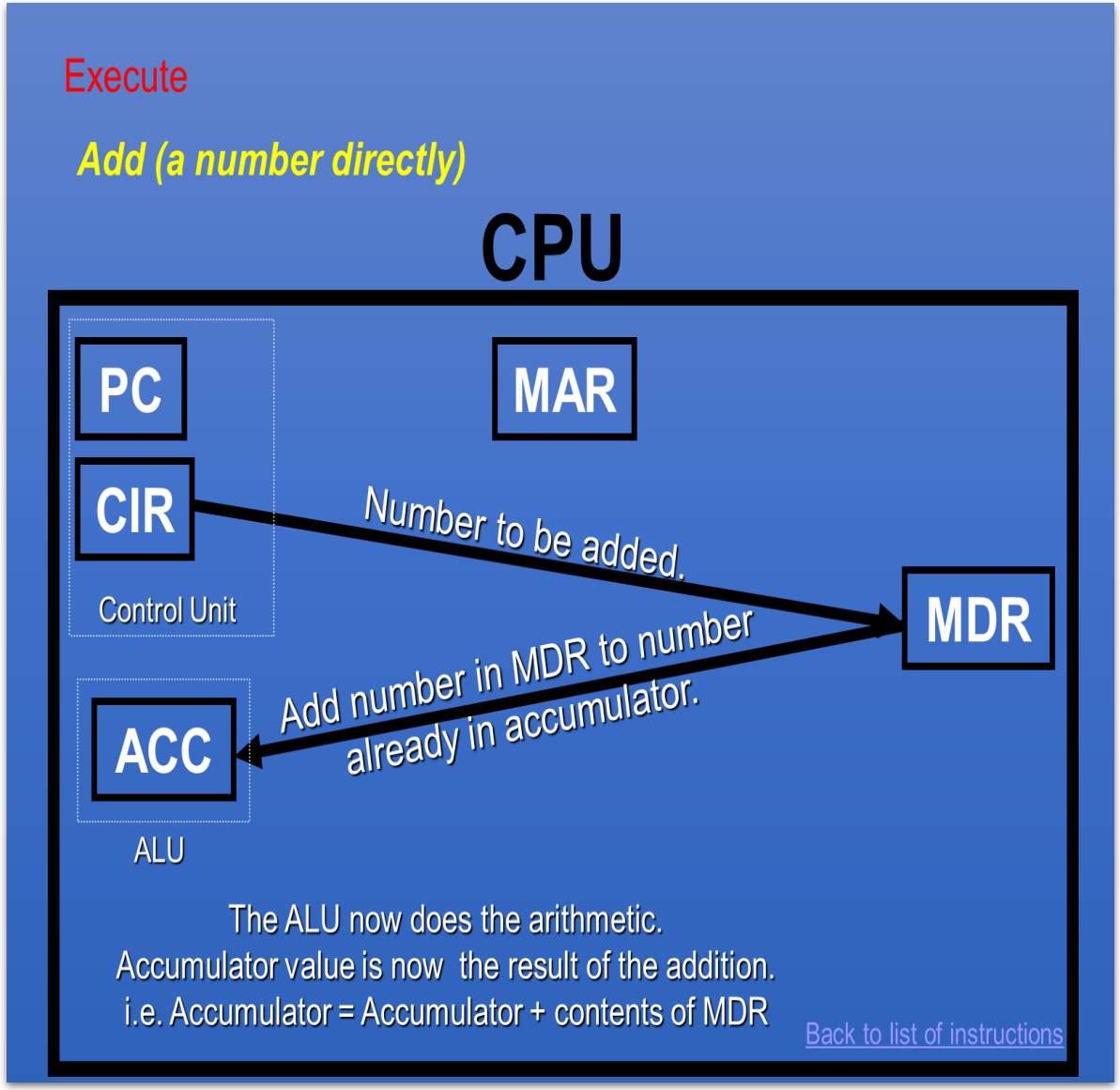
Output (from memory)


Execute
Jump
CPU
PC
MAR
CIR
Copy of address part instruction
(address to jump to).
MDR
Control Unit
ACC
ALU
Jump Instruction

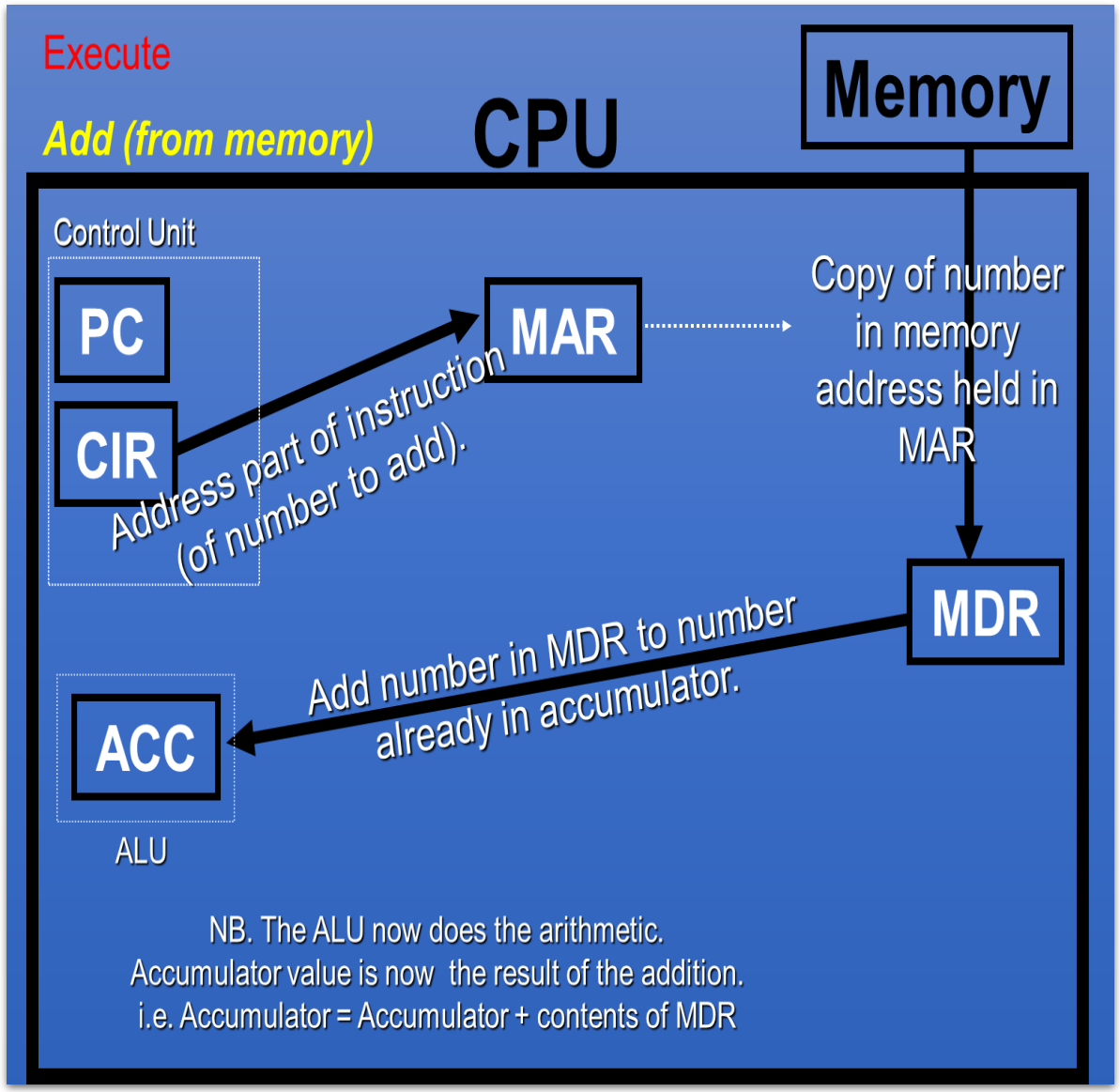

Execute
Jump
CPU
Copy of address part instruction
(address to jump to).
PC
MAR
CIR
MDR
Control Unit
ACC
ALU
Jump Instruction

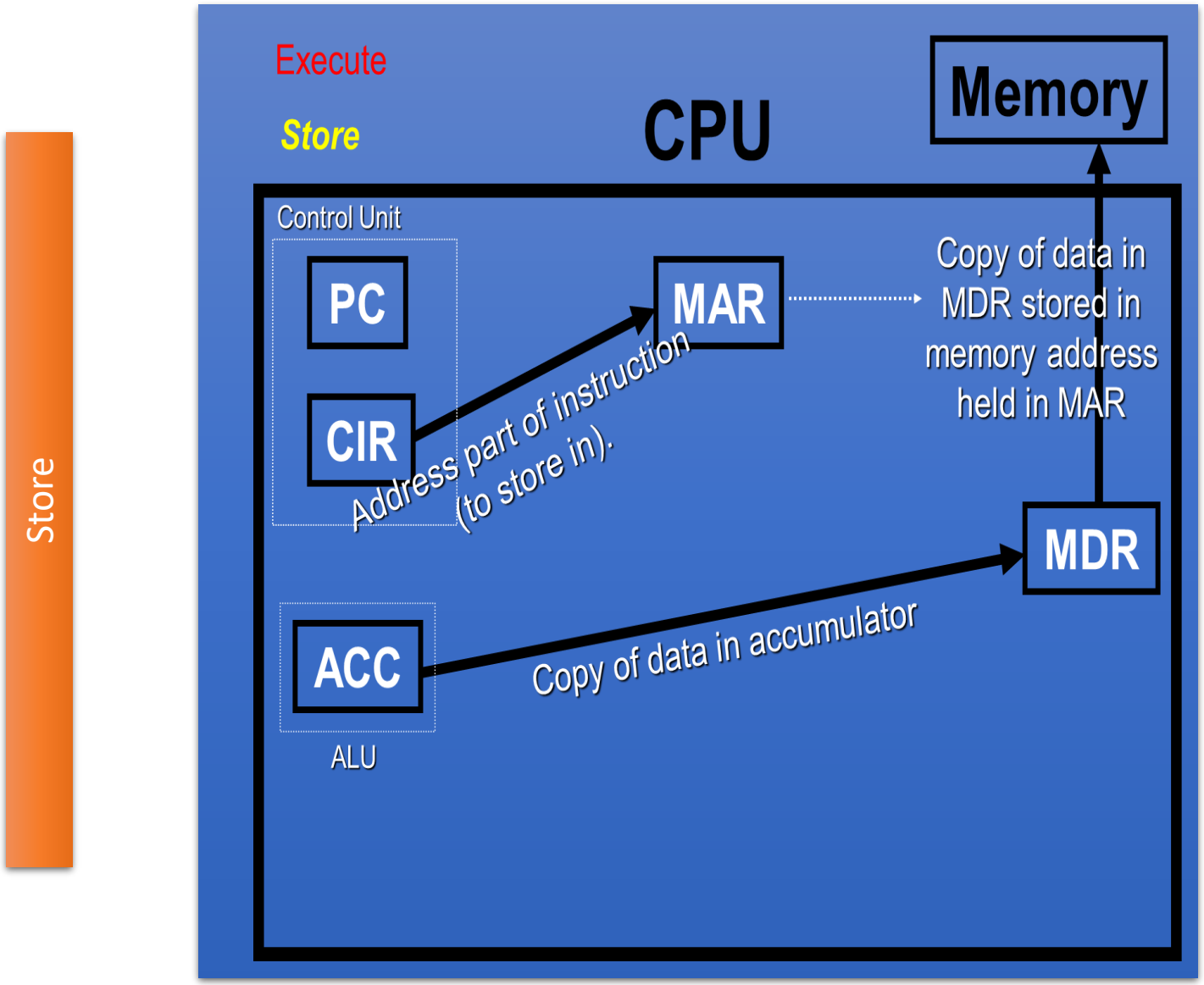

Load (from memory)
Execute
Load (from memory)
CPU
Memory
PC
MAR
CIR
Copy of data in address held in MAR
Address part of instruction (of data to be loaded).
Control Unit
MDR
Copy of data in MDR
ACC
ALU

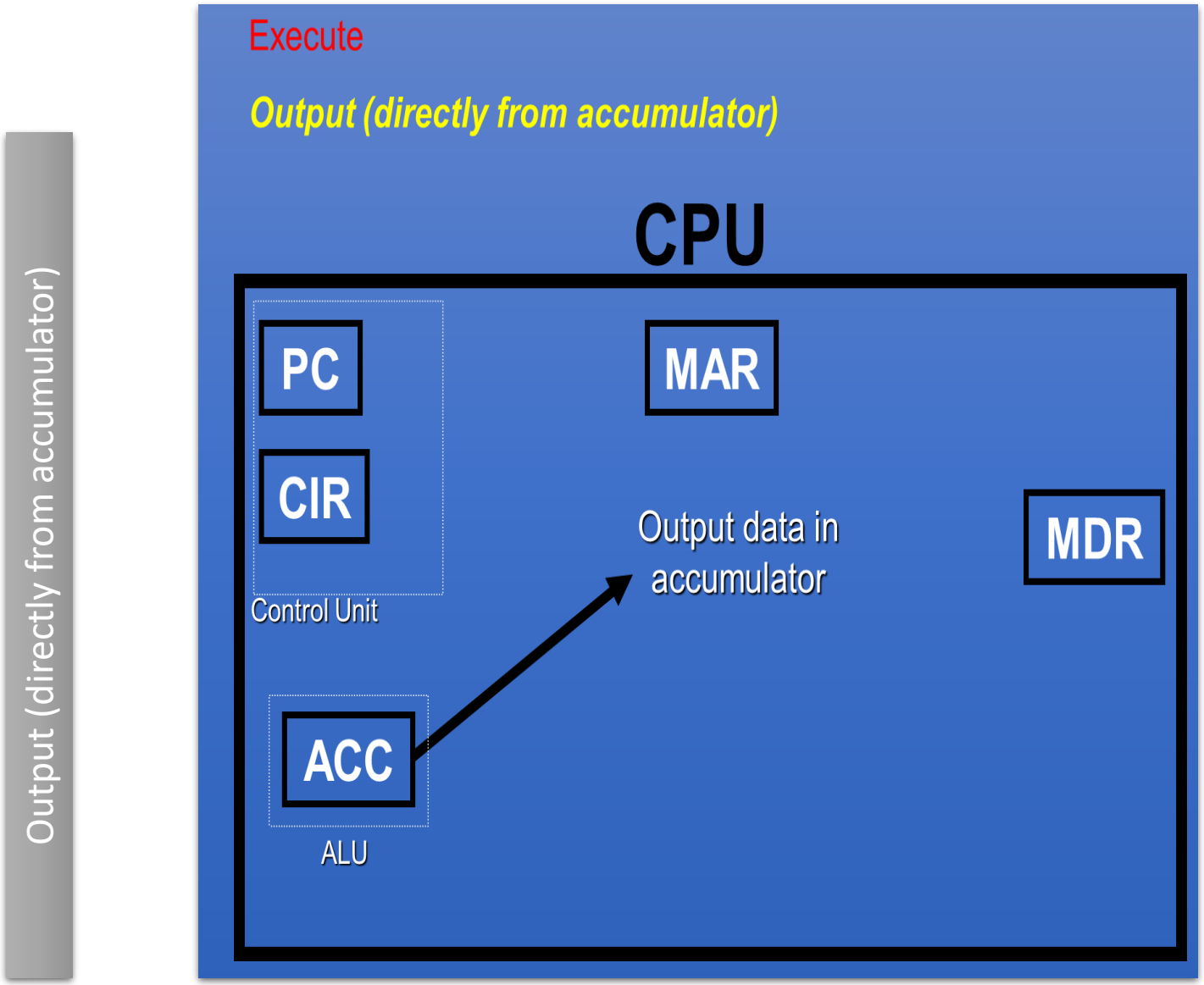
Add (a number directly)



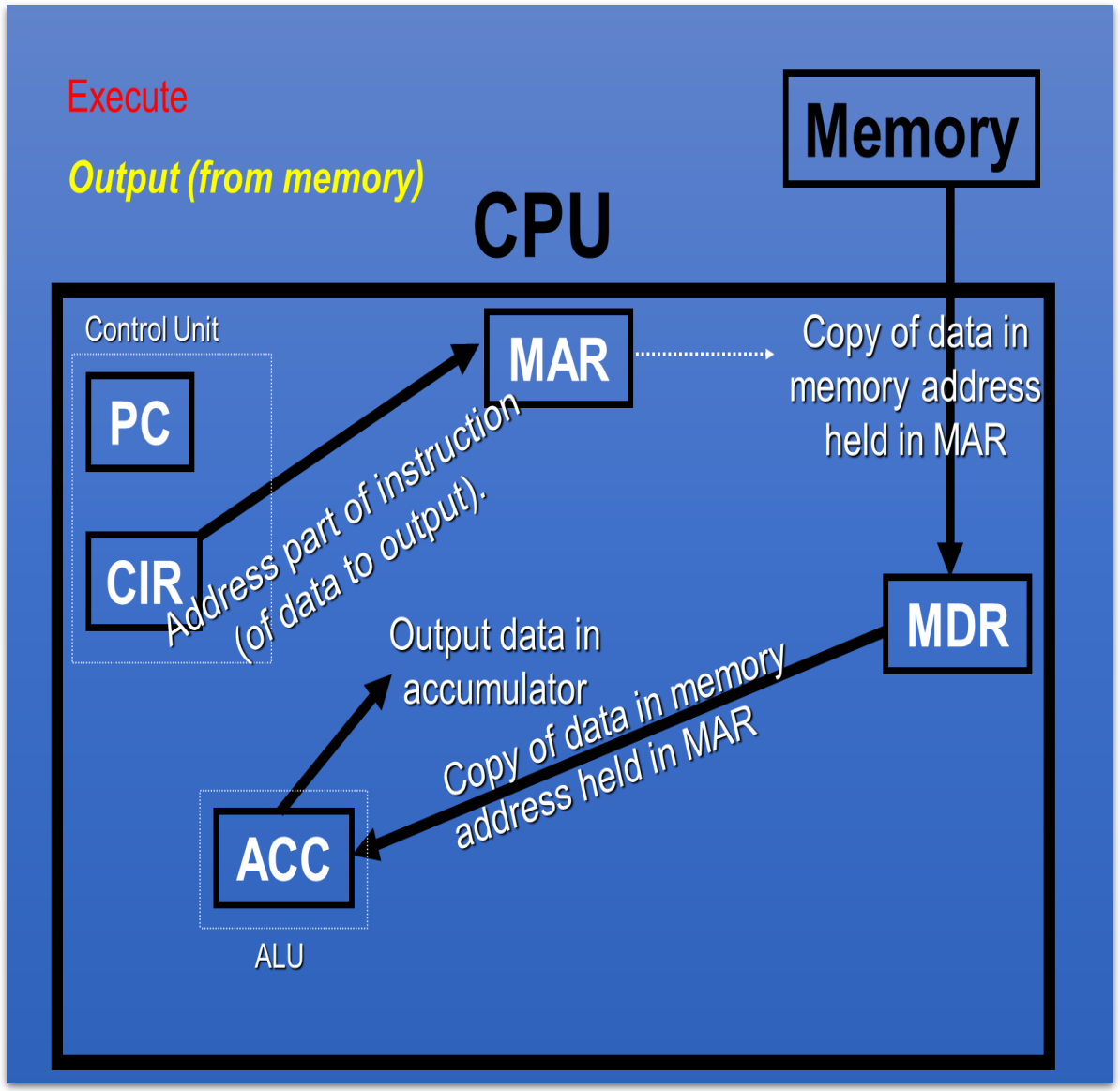
Add (from memory)

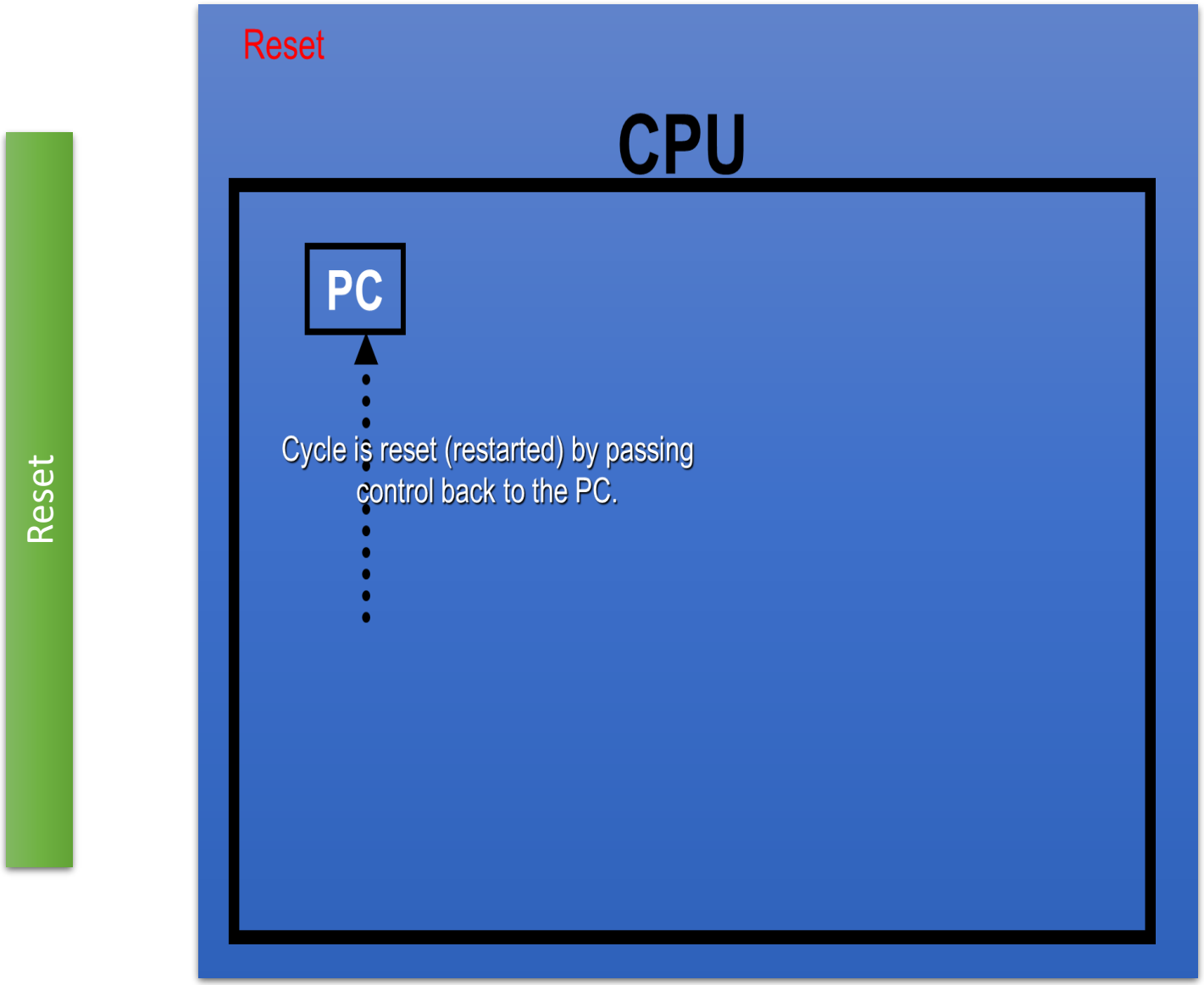



Execute
Store
CPU
Memory
Control Unit
PC
CIR
MAR
Address part of instruction (to store in).
Copy of data in MDR stored in memory address held in MAR
MDR
ACC
Copy of data in accumulator
ALU
Store


Output (directly from accumulator)
Execute
Output (directly from accumulator)
CPU
PC
CIR
Control Unit
MAR
Output data in accumulator
MDR
ACC
ALU

Output (from memory)




Reset
CPU
PC
Cycle is reset (restarted) by passing control back to the PC.
Reset

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**

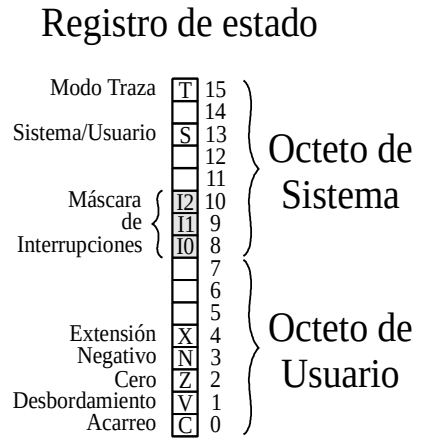
b) **Incremento del PC**

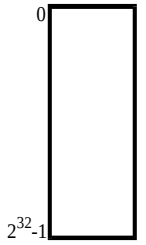
c) **Ejecución de la instrucción**

- **secuencia lineal: ejecuta instrucciones consecutivas**
- **bucle infinito**

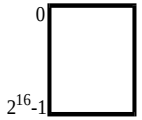
4. ***Niveles de ejecución***: generalmente los SSOO disponen de varios niveles de ejecución:

- 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



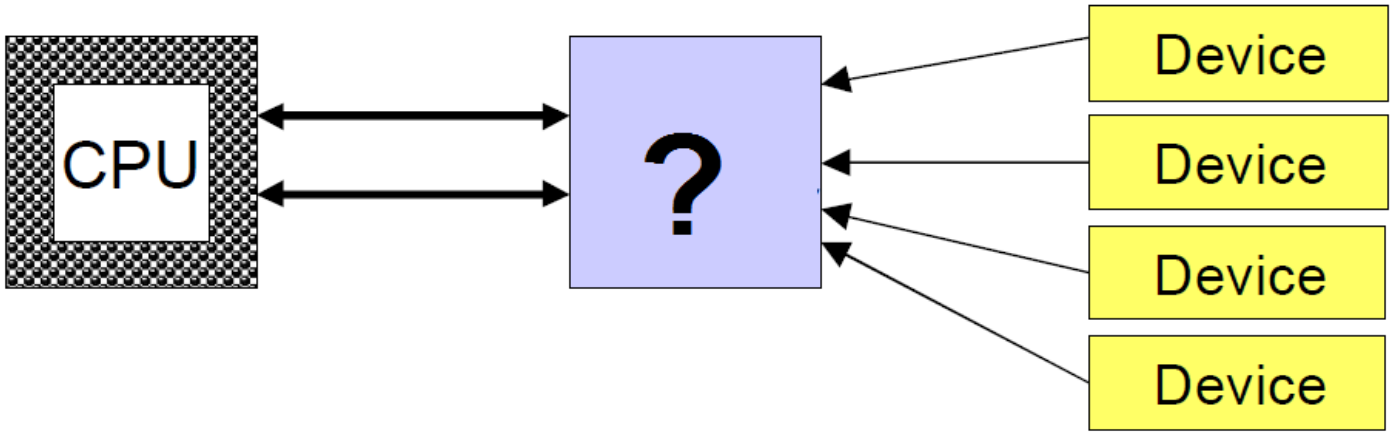
Mapa de E/S

Juego de Instrucciones

Modelo de programación de usuario

Modelo de programación de núcleo

Interrupciones


CPU
?
Device
Device
Device
Device

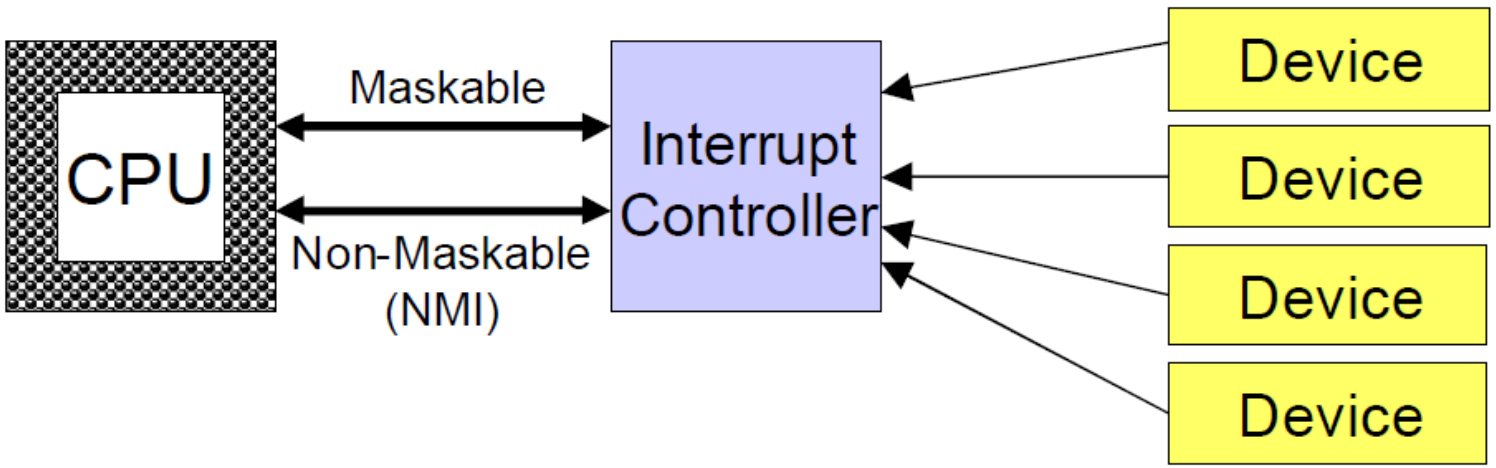
Can multiple devices interrupt processor?

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


CPU
Maskable
Non-Maskable
(NMI)
Interrupt
Controller
Device
Device
Device
Device

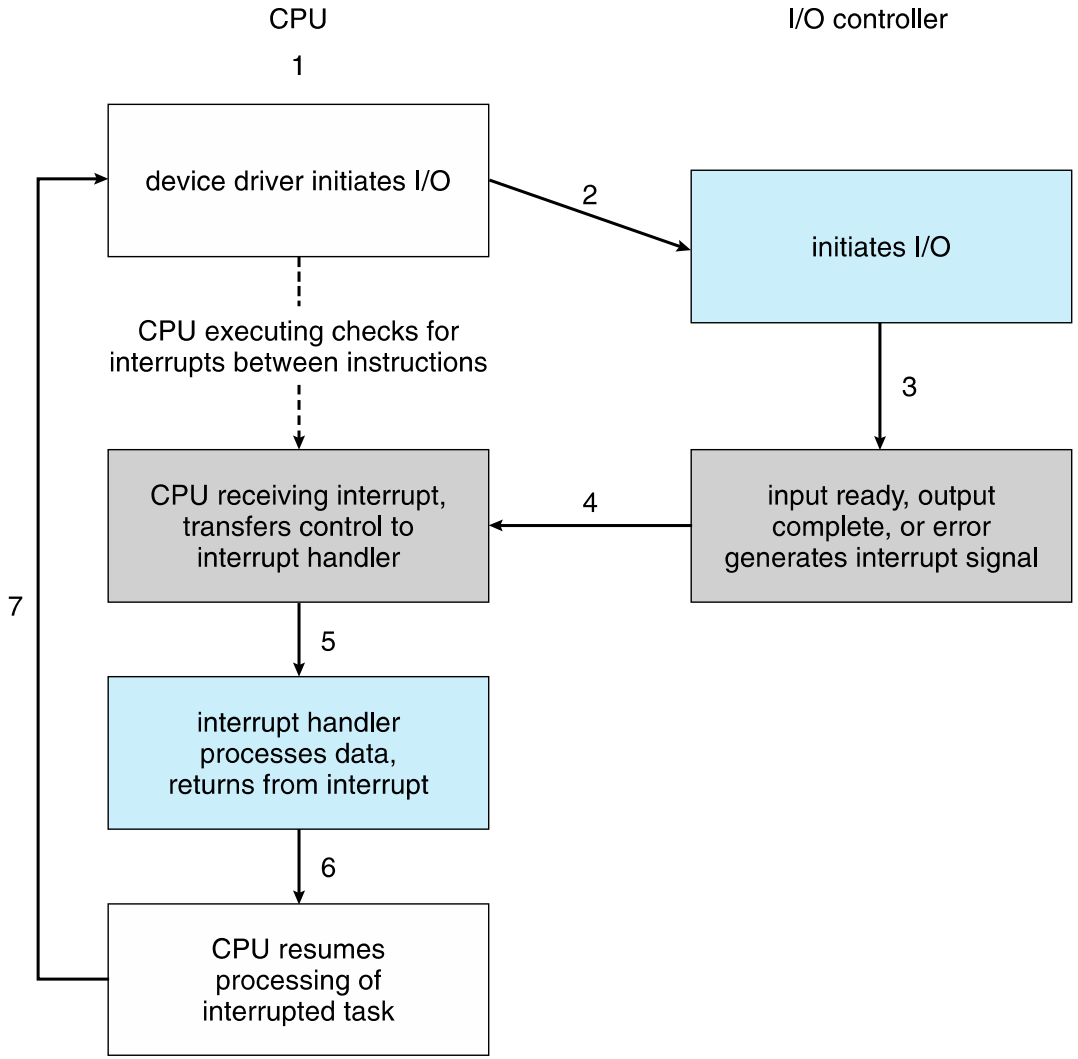
- Una interrupción se solicita activando una señal que llega a la unidad de control. El elemento que necesita ser atendido y genera la señal se denomina **agente generador**.

- 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.

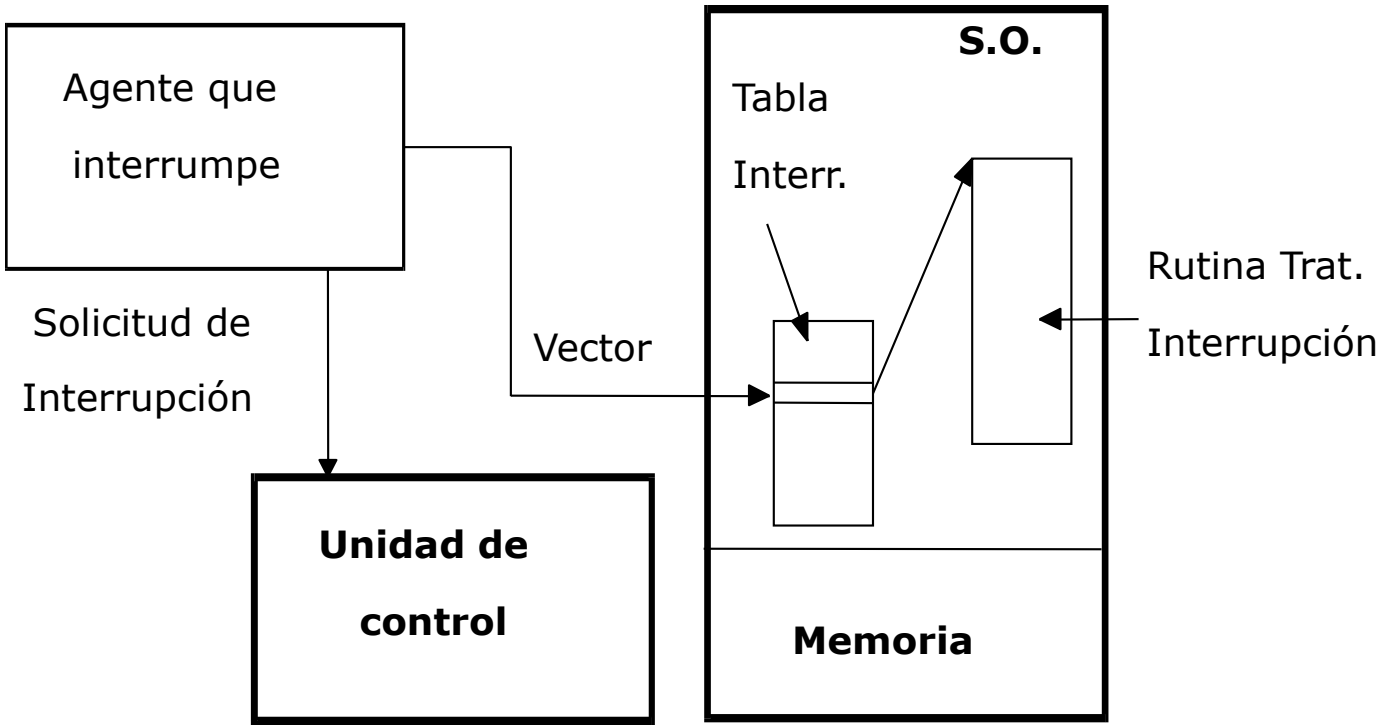
Ciclo de E/S regido por interrupciónes



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.


Agente que interrumpe
Solicitud de Interrupción
Vector
S.O.
Tabla Interr.
Rutina Trat. Interrupción
Unidad de control
Memoria

- 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).

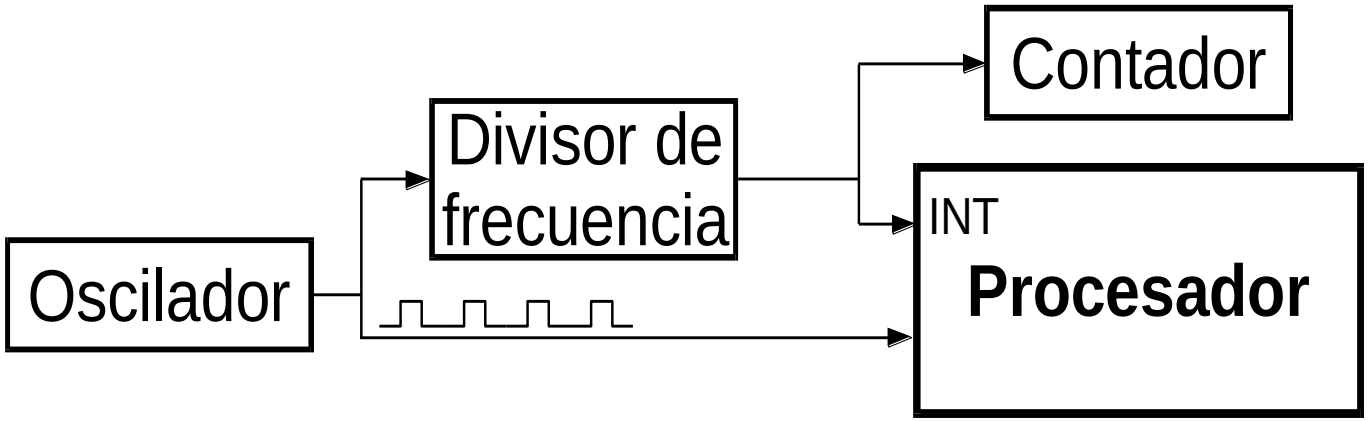
Interrupción periódica. La señal de oscilador se divide para generar interrupciones cada cierto periodo de tiempo. Objetivo: conseguir que el SO pase a ejecutar de forma sistemática cada cierto intervalo de tiempo, evitando así que algún programa acapare los recursos del sistema.

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



SEGMENTACIÓN: ANTICIPACIÓN y PARALELISMO

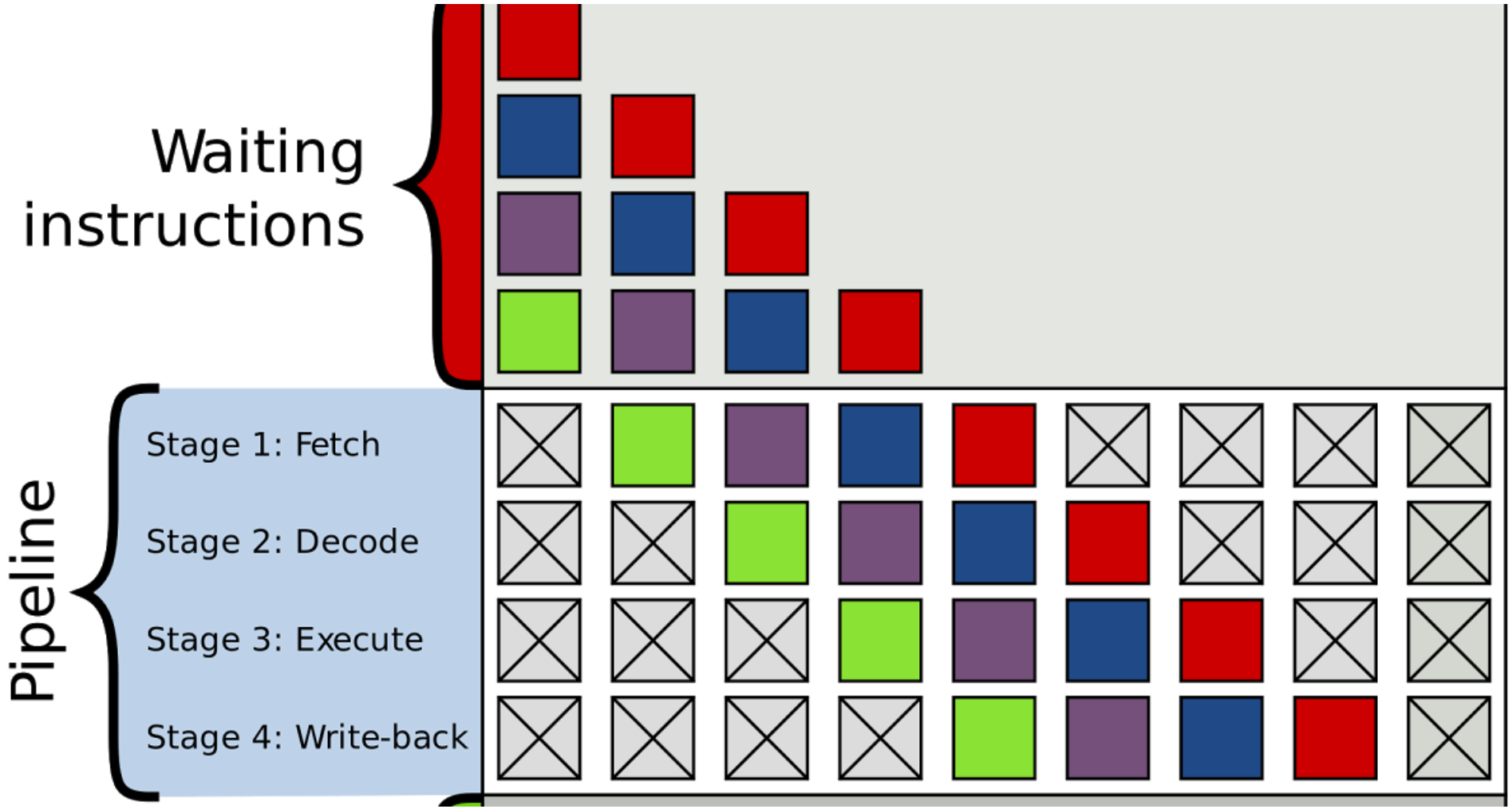
SEGMENTACIÓN
Descomponer la ejecución de cada instrucción en varias etapas para poder empezar a procesar una instrucción diferente en cada una de ellas y trabajar con varias a la vez.

ANTICIPACIÓN
Búsqueda “anticipada” de instrucciones para superponer las fases de búsqueda, descodificación y ejecución.

PARALELISMO FUNCIONAL: tareas que se pueden ejecutar al mismo tiempo.
- Espacial: replicación de hardware.
- Temporal: solapamiento de operaciones funcionales en el tiempo, segmentación.

PARALELISMO DE DATOS
- Se ejecuta una instrucción sobre un conjunto de datos.

Segmentación

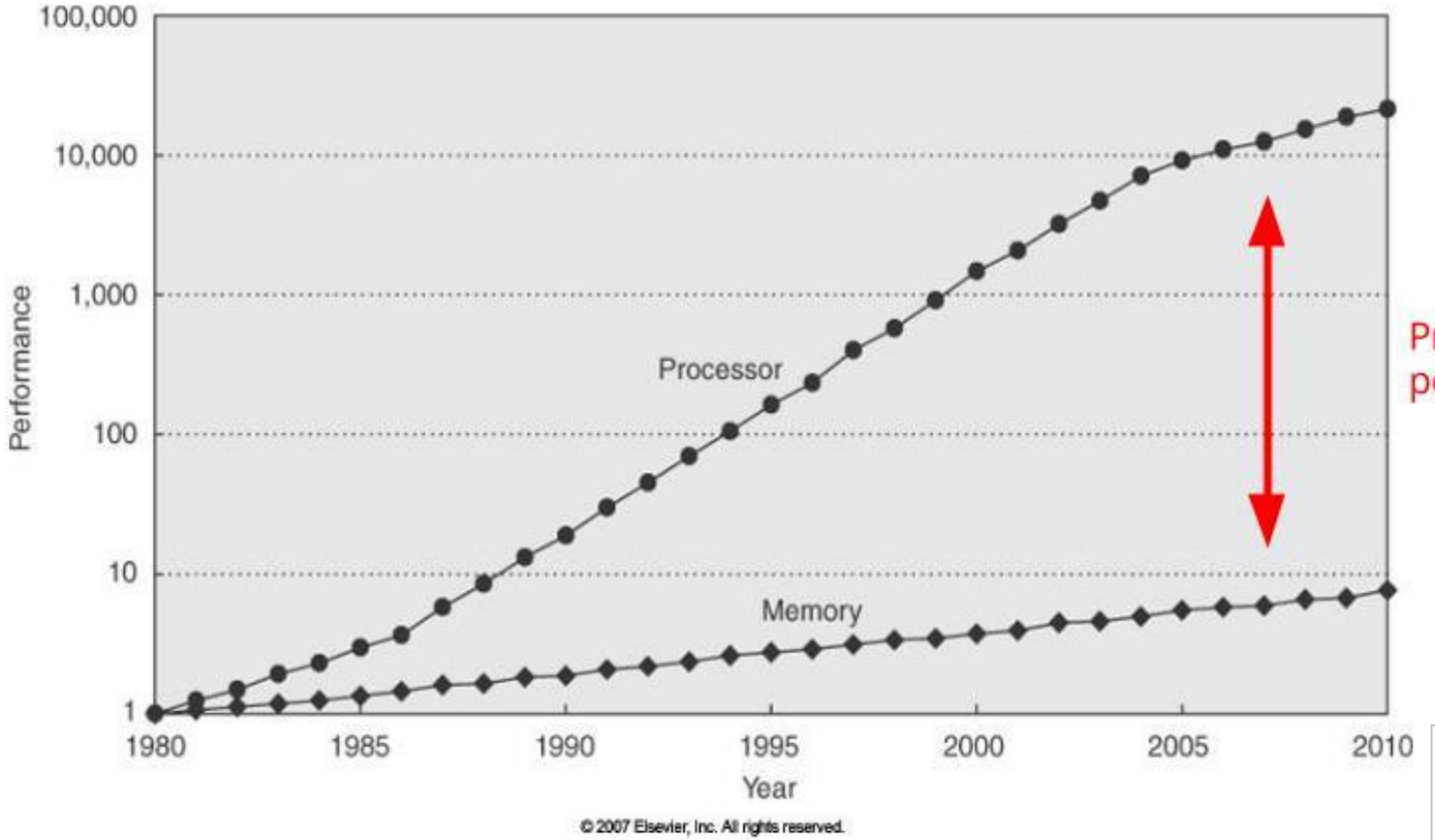


ARQUITECTURAS RISC

- Número limitado de instrucciones simples.
- Instrucciones de longitud fija (32 bit) + codificación con campos fijos.
- Operaciones Registro a registro.
  - Arquitectura Load/Store.
- Alto número de registros de propósito general (32).
- Opcionalmente.
  - Arquitectura de 64bits.
  - 3 direccionamientos: registro, inmediato, desplazamiento.
  - Un único modo de direccionamiento para los load/store: base + desplazamiento.
  - Instrucciones aritméticas tipo reg-reg de 3-direcciones.

ARQUITECTURAS CISC

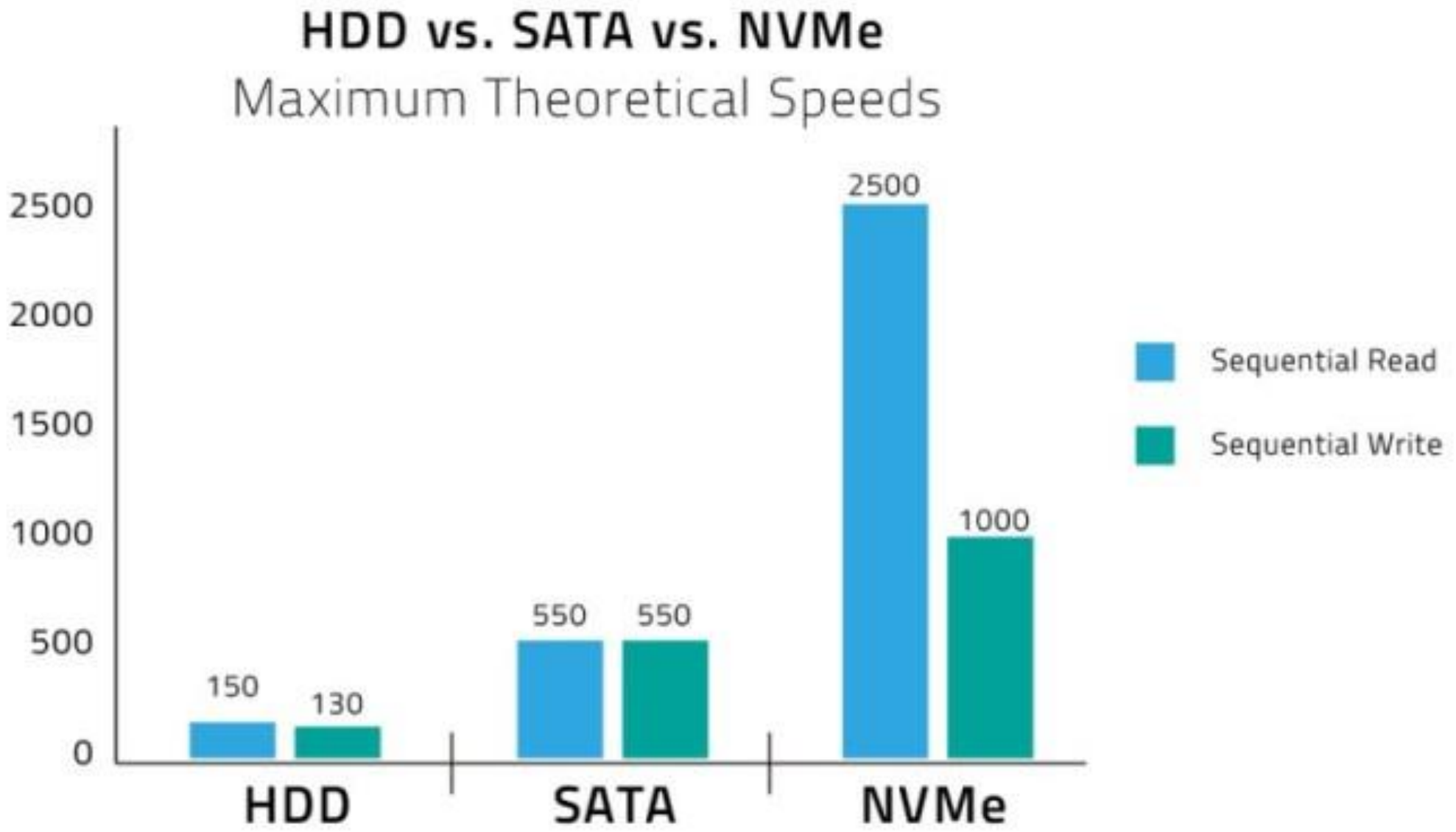
- Muchas instrucciones complejas.
- Instrucciones de longitud variable.
- Operaciones tipo Memoria a registro.
- Pocos registros “generales/implícitos” (8).
- Opcionalmente.
  - Arquitectura 32/64 bit.
  - 2-3 direccionamientos.
  - Varios modos de direccionamiento en las instrucciones load/store:

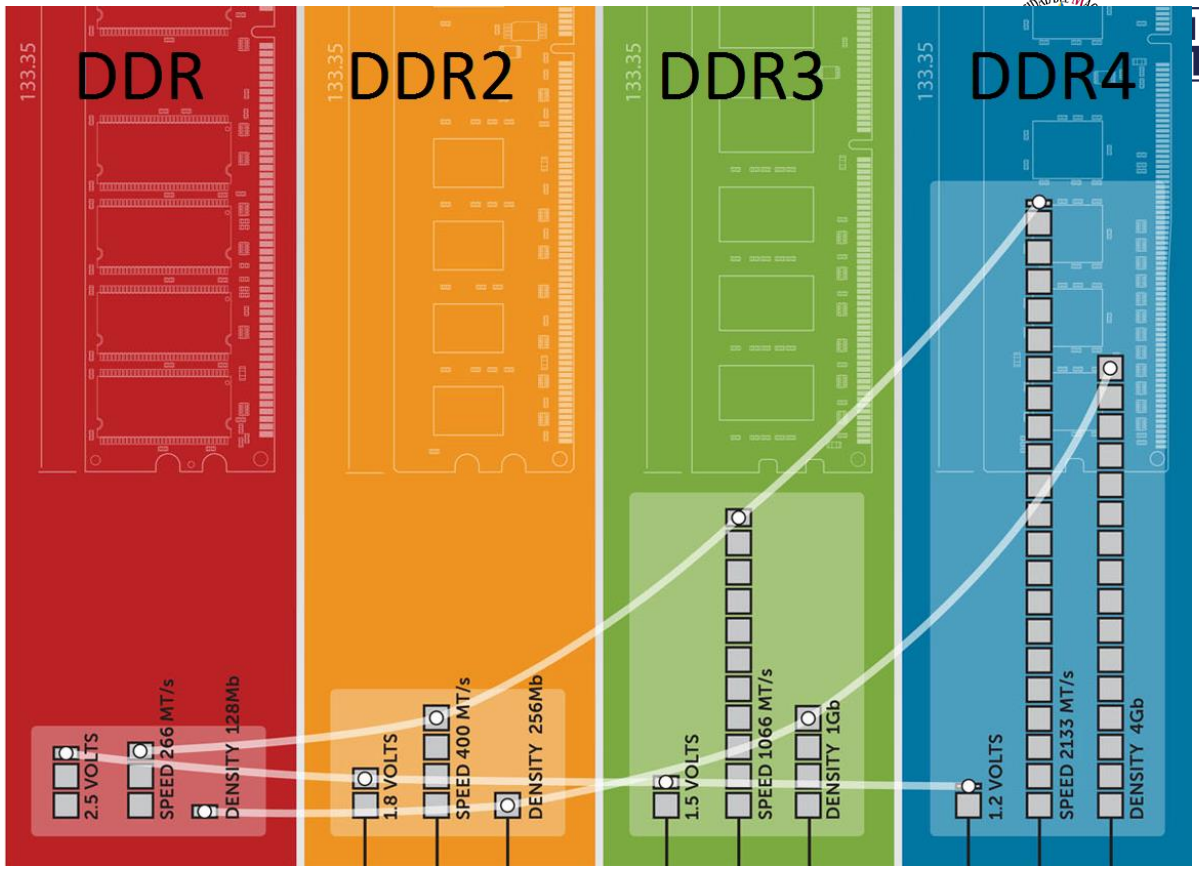


Processor vs. Memory performance gap

Memory speed lags behind CPU speed

execute typical instruction	1/1,000,000,000 sec = 1 nanosec
fetch from L1 cache memory	0.5 nanosec
branch misprediction	5 nanosec
fetch from L2 cache memory	7 nanosec
Mutex lock/unlock	25 nanosec
fetch from main memory	100 nanosec
send 2K bytes over 1Gbps network	20,000 nanosec
read 1MB sequentially from memory	250,000 nanosec
fetch from new disk location (seek)	8,000,000 nanosec
read 1MB sequentially from disk	20,000,000 nanosec


HDD vs. SATA vs. NVMe
Maximum Theoretical Speeds
2500
2000
1500
1000
500
0
150
130
550
550
2500
1000
HDD
SATA
NVMe
Sequential Read
Sequential Write



DDR SDRAM Standard	Internal rate (MHz)	Bus clock (MHz)	Prefetch	Data rate (MT/s)	Transfer rate (GB/s)	Voltage (V)
SDRAM	100-166	100-166	1n	100-166	0.8-1.3	3.3
DDR	133-200	133-200	2n	266-400	2.1-3.2	2.5/2.6
DDR2	133-200	266-400	4n	533-800	4.2-6.4	1.8
DDR3	133-200	533-800	8n	1066-1600	8.5-14.9	1.35/1.5
DDR4	133-200	1066-1600	8n	2133-3200	17-21.3	1.2

Historical Cost of Computer Memory and Storage