Universidad Nacional de Entre Ríos

Tecnicatura universitaria en desarrollo web

Diseño de CPU

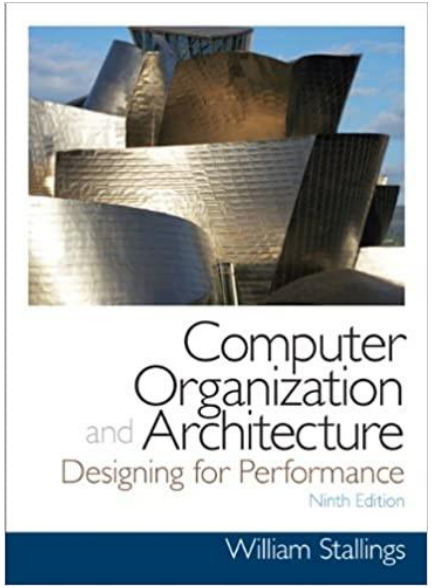
Semana 3 – Arquitectura de computadoras

Docente: José Ruiz {jose.ruiz@uner.edu.ar}

UNER virtual

Esta presentación esta basada en el libro de:

- William Stallings, Computer Organization and Architecture, 9th Edition, 2017



Archivos presentación y ejemplos se alojan en:

https://github.com/ruiz-jose/tudw-arq.git

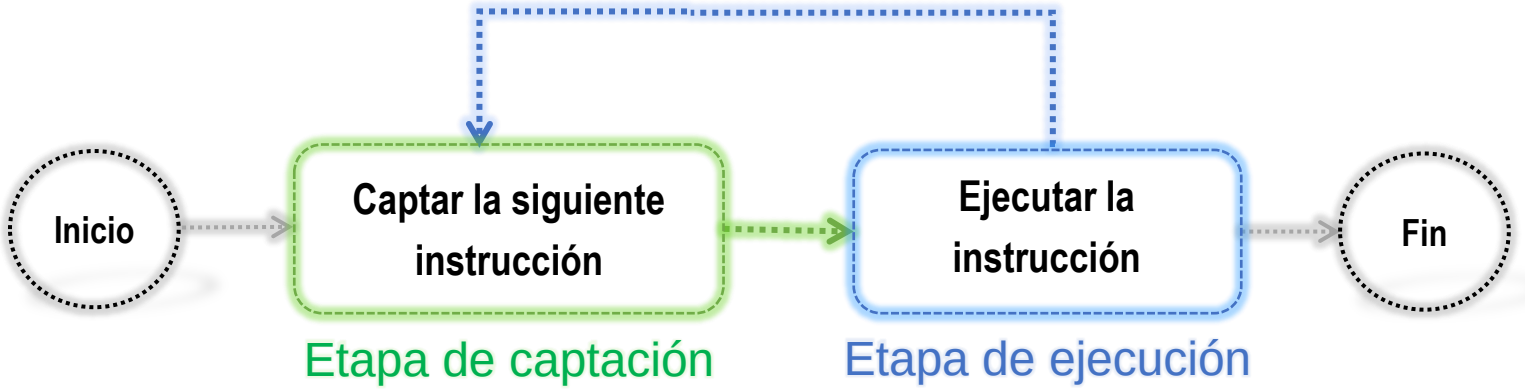
Diseño de CPU

- Ciclo de la instrucción 

  - Etapa de captación y ejecución

- Componentes de la computadoras:
  - ALU y registros
  - Arquitectura acumulador (ACC)

Ciclo de instrucción – Etapa de captación y ejecución



Ciclo de instrucción – Lenguajes de alto y bajo nivel

Lenguaje alto nivel

- **C:**

```
Int z =0, x=3, y=2;
z = x + y;
```

Se traduce

Lenguaje bajo nivel

- **Ensamblador:**

```
load x
add y
store z
```

Mapeo 1 a 1

Lenguaje de maquina

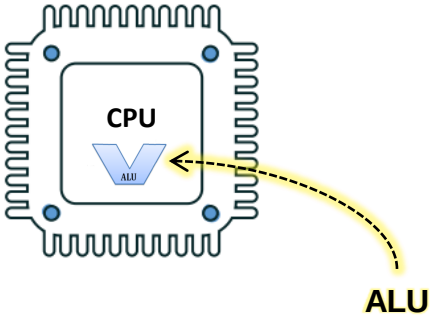
- **Código maquina:**

```
00 110010
10 110011
01 110100
```

Una instrucción en **alto nivel** equivale a varias instrucciones en **bajo nivel**.

```
.data
x db 3
y db 2
z db 0
```

Ciclo de instrucción – Componentes de la computadora



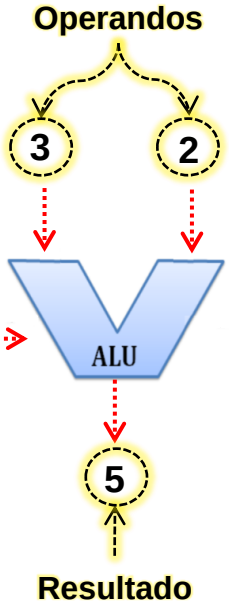
- **C:**

```
Int z =0, x=3, y=2;
z = x + y;
```

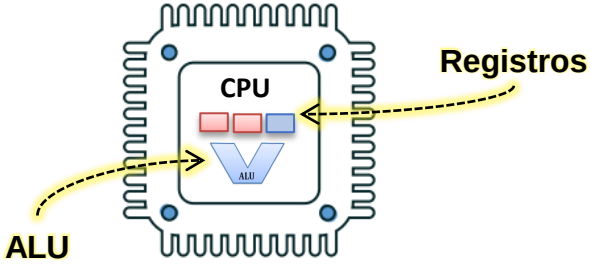
- **Ensamblador:**

```
load x
add y
store z
```

Código operación ----> **add**



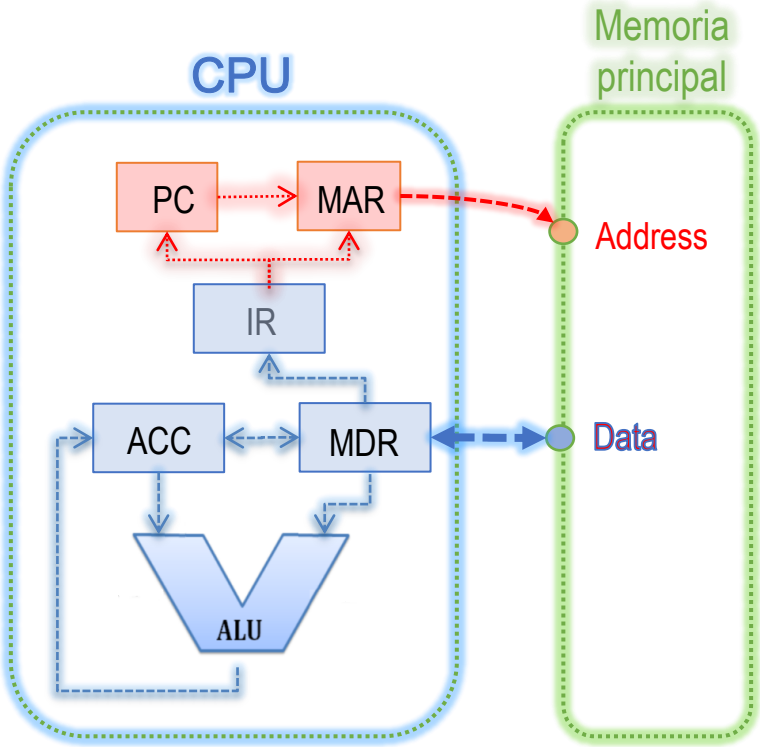
¿**de dónde salen los datos x e y**?, ¿dónde están ese 3, ese 2 y el número que representa la operación de suma? En algún lado tienen que estar almacenados, ¿no?



Registros:

- **Contador de programa de PC**: contiene la dirección de la próxima instrucción que se ejecutará
- **Registro de direcciones de memoria MAR**: contiene la ubicación de la memoria de los datos a los que se debe acceder.
- **Registro de datos de memoria MDR**: contiene datos que se transfieren a/o desde la memoria.
- **Acumulador ACC:** se almacenan resultados aritméticos y lógicos intermedios.
- **Registro de instrucción IR**: contiene la instrucción actual durante el procesamiento.

Arquitectura acumulador (ACC)



Ciclo de instrucción – Componentes de la computadora

En la lenguaje ensamblador de la **arquitectura acumulador** se utilizan los nemónico **lda** (load acc) y **sta** (store acc) para indicar transferencia desde o a memoria en vez de **load** y **store**, entonces se utiliza:

load x → lda x
store z → sta z

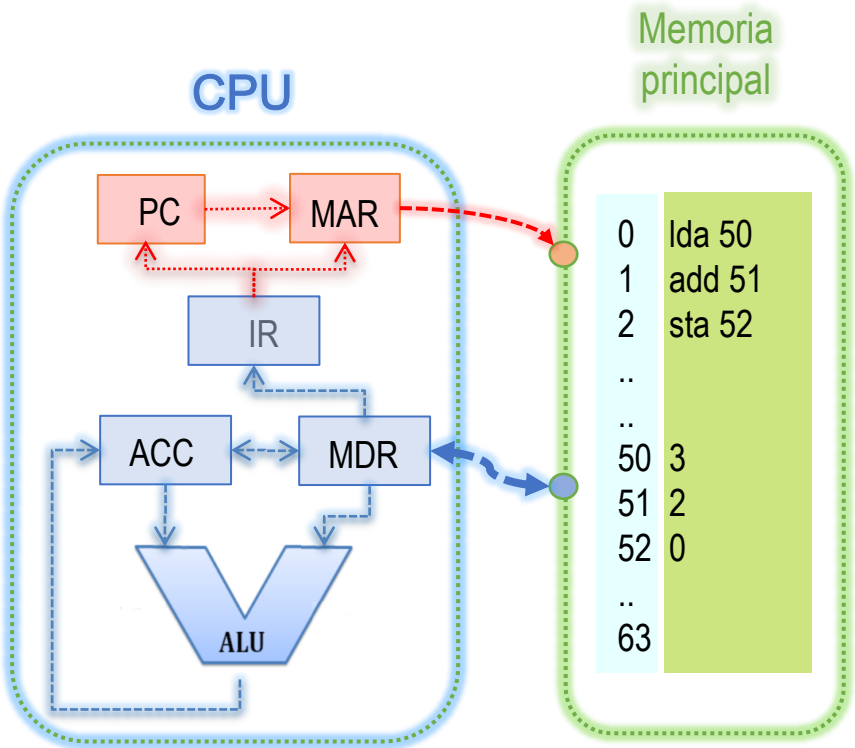
- **C:**

```
Int z =0, x=3, y=2;
z = x + y;
```

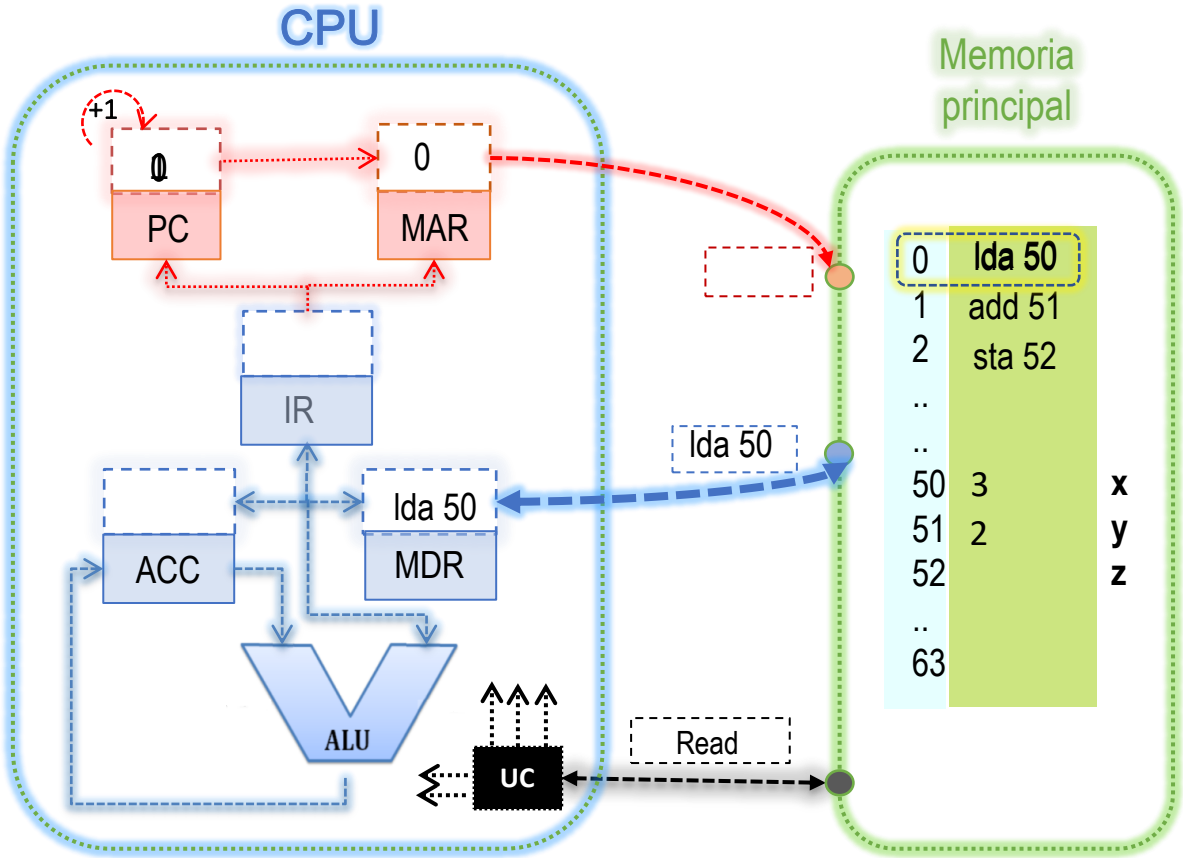
- **Ensamblador:**

```
lda x
add y
sta z
```

```
.data
x db 3
y db 2
z db 0
```



Ciclo de instrucción – Etapa de captación



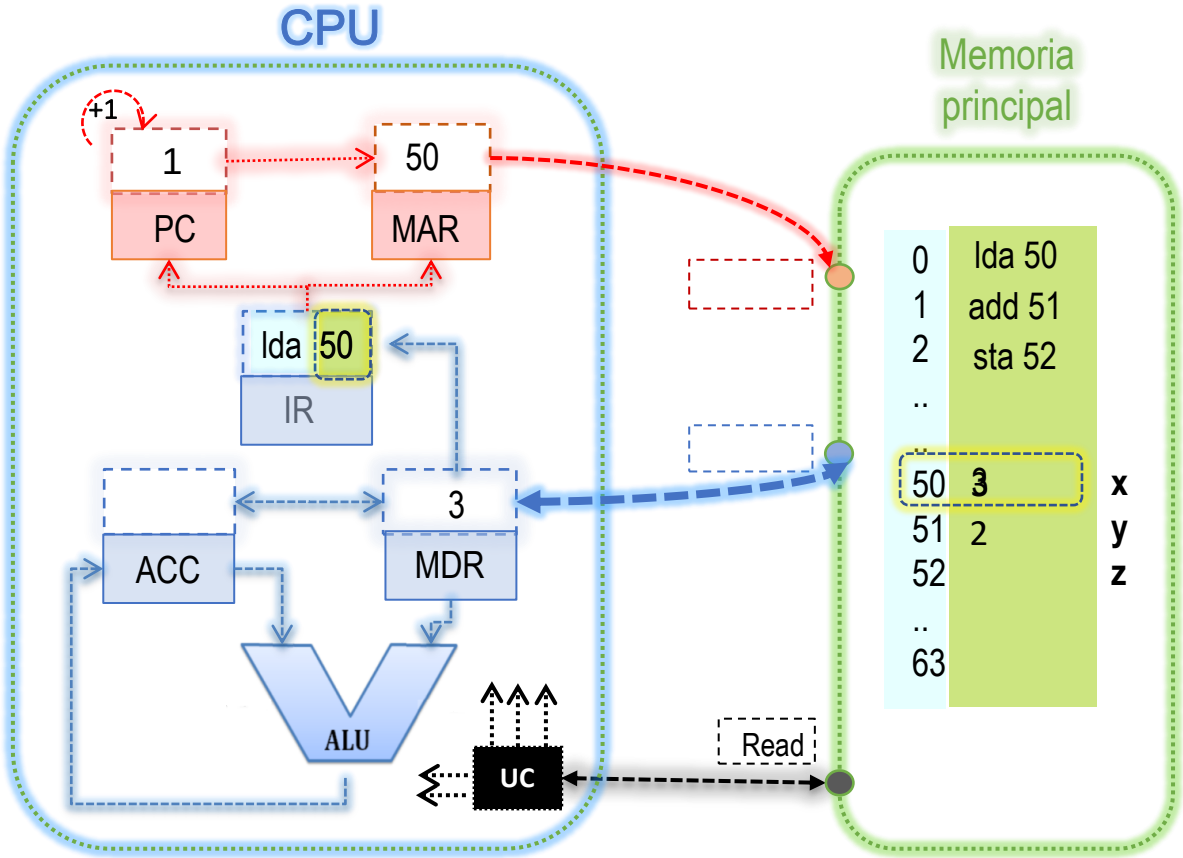
Etapa de captación

La secuencia de pasos de la **unidad de control (UC)** para captar una instrucción.

1: MAR ← PC

2: MDR ← read(Memoria[MAR]) ;
PC ← PC + 1

3: IR ← MDR



Etapa de ejecución

La secuencia de pasos de la **unidad de control (UC)** para ejecutar una instrucción.

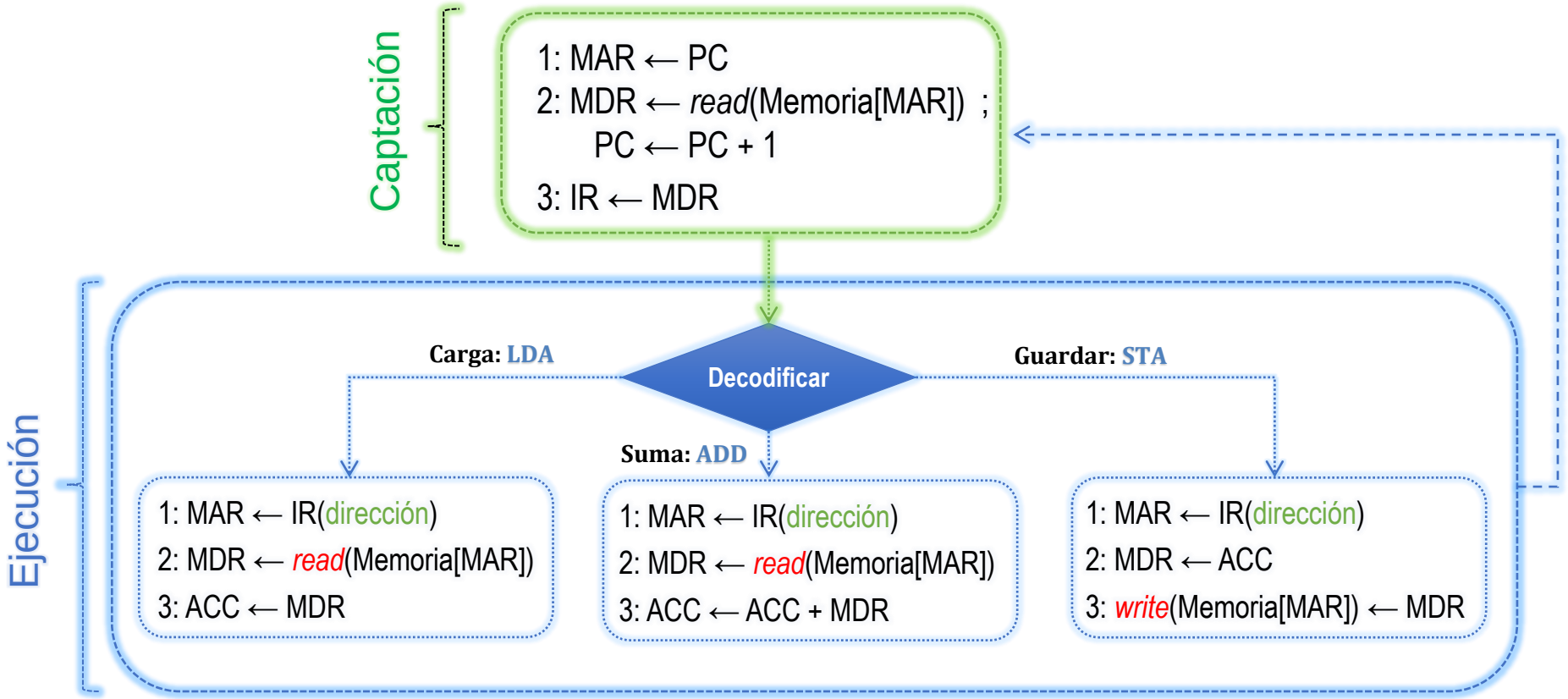
Cada tipo de instrucción tiene sus **propios pasos de ejecución.**

1: MAR ← IR(dirección)

2: MDR ← read(Memoria[MAR])

3: ACC ← MDR

Ciclo de instrucción – Etapa de captación y ejecución



Preguntas?