La Máquina de Von Neumann

Preparado por :

Pedro Rodríguez Moreno (QEPD)
Manuel Crisosto Muñoz
Christian Vidal Castro
Departamento de Sistemas de Información
Facultad de Ciencias Empresariales

Lecturas Recomendadas para este capítulo

- John Hayes. “Computer Architecture and Organization”. 2ª. Edición 1998. McGraw Hill.
- Andrew Tanenbaum. “Organización de Computadores”. 4ª Edición, Prentice-Hall, 1999.
- Pedro de Miguel Anasagasti. “Fundamentos de los Computadores”. 1ª Edición, Paraninfo, 1992.

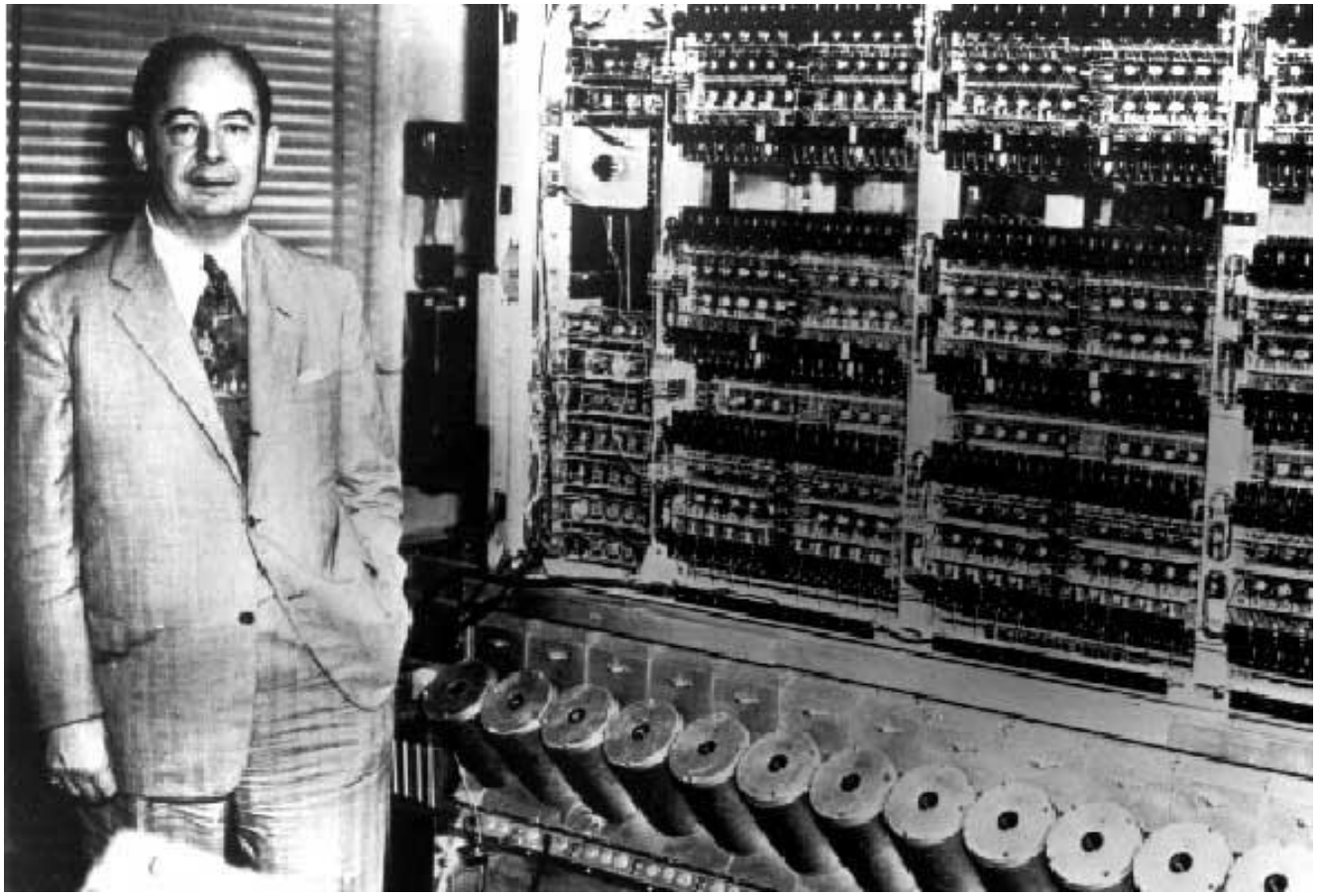
- “John von Neumann (su nombre en húngaro es Margittai Neumann János Lajos) es un matemático húngaro considerado por muchos como la mente más genial del siglo XX, comparable solo a la de Albert Einstein”. Visitar:

  http://www.eumed.net/cursecon/economistas/neumann.htm

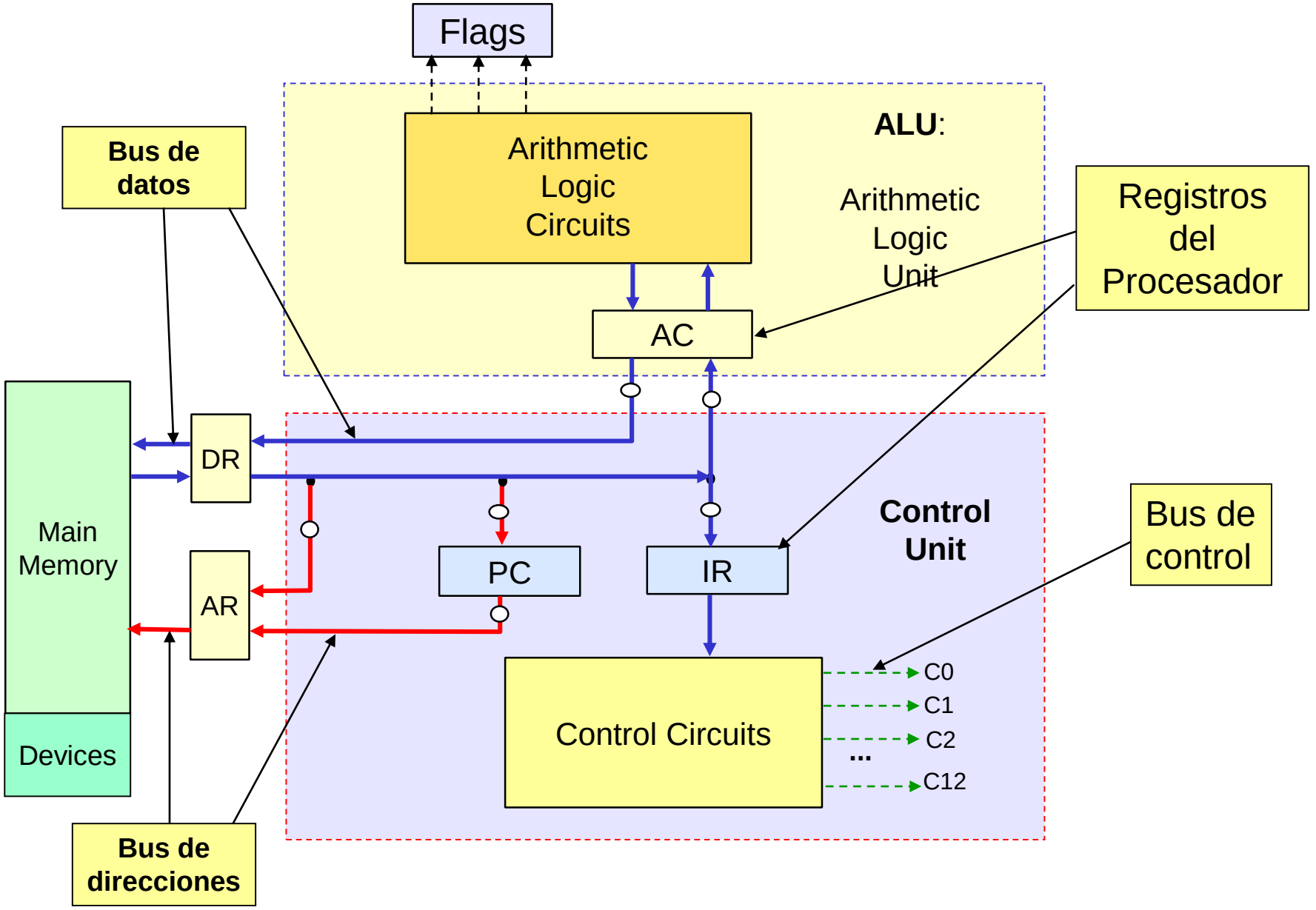
  http://www.zyvex.com/nanotech/vonNeumann.html

- Este modelo sirvió de base para el diseño de los procesadores modernos.
- Es un procesador muy sencillo de programar.
- Es una **máquina de una dirección**, lo que significa que las instrucciones de máquina pueden tener como máximo un operando. Por ejemplo: **add a**

John Von Neumann (1903-1957)



La CPU de von Neumann



Descripción de los Registros

- **AC:** Acumulador, almacena operandos y resultados.
- **DR:** Registro de Datos (Data Register),Almacena datos de entrada/salida, instrucciones.
- **AR:** Registro de Direcciones (Address Register), Almacena direcciones de memoria de datos e instrucciones.
- **IR:** Registro de Instrucciones (Instruction Register), Almacena el código de operación de la instrucción.
- **PC:** Contador de Programa (Program Counter), contiene la dirección de memoria de la siguiente instrucción a ejecutar.
- **Flags:** Almacena el estado del procesador.

- Son canales de comunicación y de transmisión de información que permiten conectar a dos elementos del procesador. La información que pueden transmitir son: datos, direcciones de memoria, señales de control e instrucciones.
- Existen tres tipos de buses:
  - **Bus de datos**: es bidireccional (transmite información an ambos sentidos), permite la transmisión de operandos e instrucciones.
  - **Bus de direcciones**: es unidireccional y transmite direcciones de memoria.
  - **Bus de control**: es bidireccional y transmite señales de control.
- Físicamente, los buses son cables por donde transitan señales eléctricas.

Componentes de la CPU

- La **CPU** está constituida de 6 registros básicos.
- La **ALU** está conformada por los circuitos aritméticos y lógicos, y el registro **AC**.
- La **Unidad de Control** la constituye los circuitos de control, que se encarga de recibir y emitir señales de control, y de los registros **IR** y **PC**.
- Los registros **DR** y **AR** actúan como buffers (almacenamiento temporal) de datos, instrucciones y direcciones de memoria.
- La **Memoria Principal** (Main Memory) se comunica con la **CPU** a través de los registros **DR** y **AR**.

- Son del tipo CISC (**C**omplex **I**nstruction **S**et of **C**omputer) porque tienen la capacidad de extraer datos desde la memoria y además realizar la operación.

Instrucciones aritméticas	Código de Operación (Lenguaje assembly)
Suma	add
Resta	sub
Multiplicación	mul
División	div

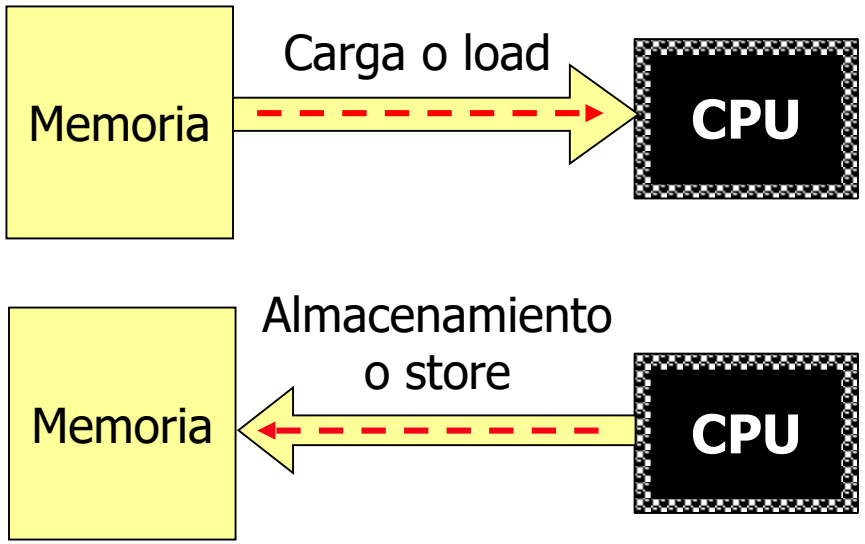
Instrucciones de Transferencia de Datos

- Permiten la transferencia de datos entre memoria y CPU.

Instrucción de Transferencia de Datos	Código de Operación (Lenguaje assembly)
Carga (load)	lda
Almacenamiento o escritura (store)	str

Descripción de las instrucciones de carga y almacenamiento

- La instrucción de carga (**lda**) extrae un operando desde memoria y lo coloca finalmente en el registro acumulador (**AC**). Ejemplo: **lda x** (x es la dirección de memoria).
- La instrucción de almacenamiento (**str**) realiza la operación inversa, toma un dato que se encuentre en el acumulador (**AC**) y lo escribe en memoria, en la dirección especificada por la misma instrucción. Ejemplo: **str x** (x es la dirección de memoria).

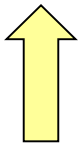


- Supongamos que tenemos la siguiente suma:

```
c = a + b;      # a, b y c representan localizaciones o direcciones de
                # memoria
```

- Suponiendo que los operandos **a** y **b** ya se encuentran en la memoria, esta instrucción de alto nivel (lenguaje C), equivale en lenguaje de máquina (assembly) al siguiente conjunto de instrucciones:

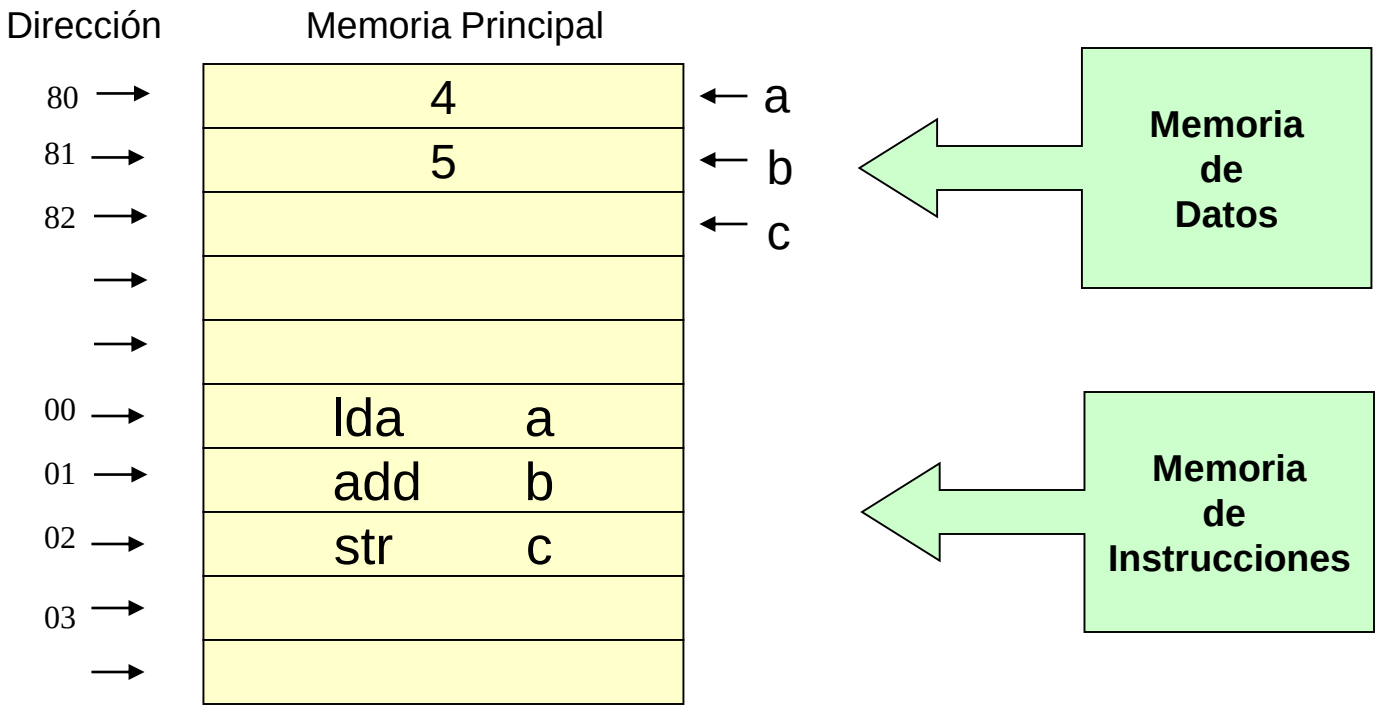
```
lda   80    # extrae o lee el contenido de 80 desde memoria y lo deja
            # en AC
add   81    # extrae o lee el contenido de 81 desde memoria y lo suma
            # con el contenido del AC. El resultado queda en AC.
str   82    # el resultado de la suma se escribe en la memoria en la
            # localización 82.
```



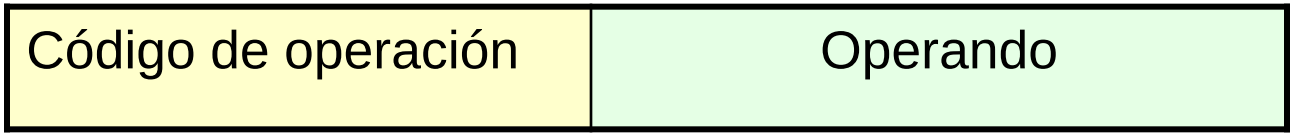
¡Nuestro primer programa!

El programa almacenado en la memoria

- Cada instrucción ocuparía una palabra de memoria



- Las instrucciones de máquina poseen un formato que especifica su estructura.
- En este procesador es posible manejar un solo operando por instrucción. Los procesadores modernos permiten dos o más operandos.
- El formato de instrucción para la máquina de Von Neumann es el siguiente:



Modos de direccionamiento básicos

- La Máquina de Von Neumann soporta tres modos de direccionamiento. Con ellos una instrucción de máquina le indica al procesador donde se encuentra el dato (en un registro o en memoria), y cuántos accesos a memoria realizar. Los tres modos de direccionamiento básicos son los siguientes.
  - **Inmediato**. La instrucción de máquina contiene el operando. Ejemplo:

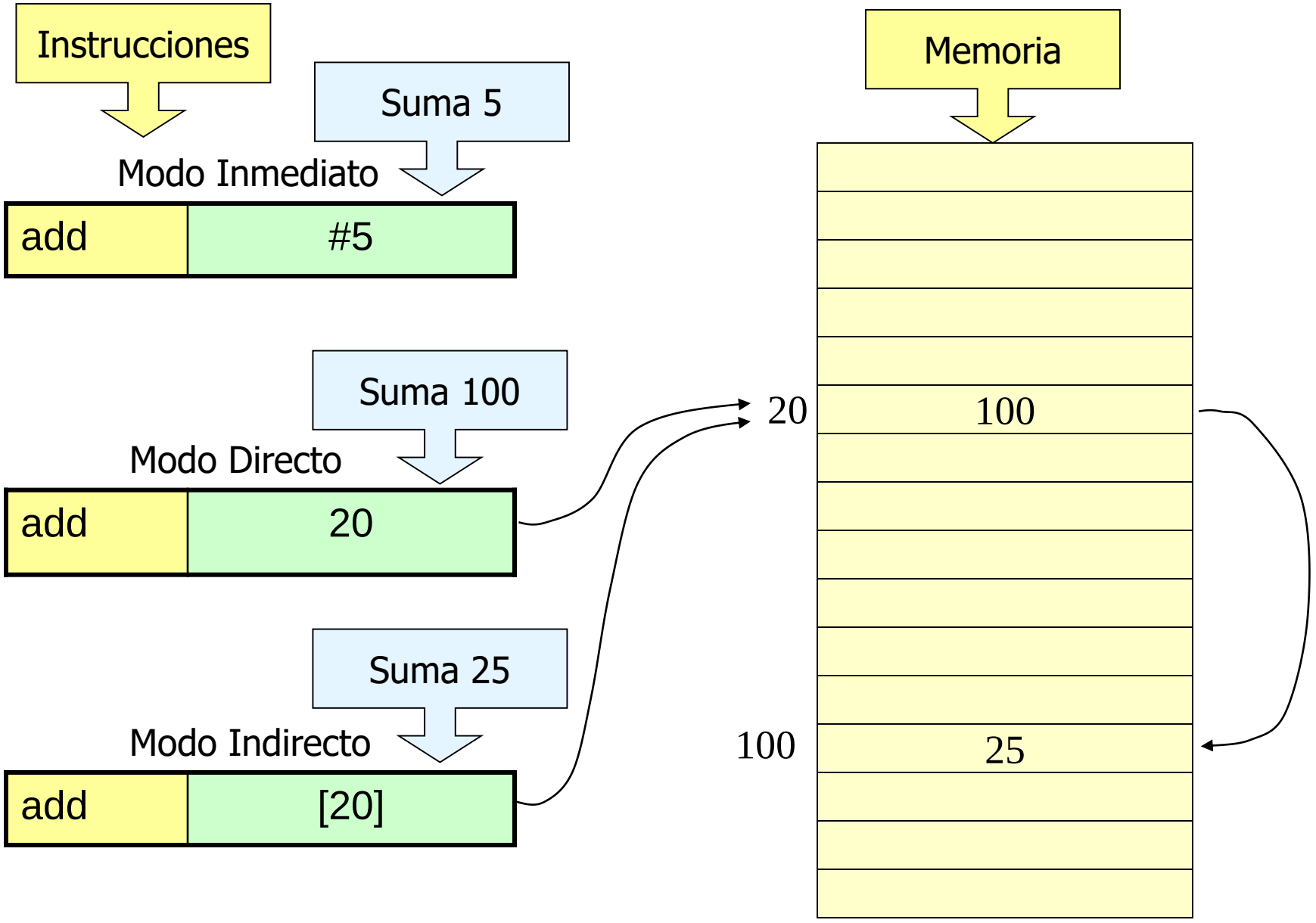
    add #5 # 5 es el operando
  - **Directo**. La instrucción contiene la dirección de memoria del dato, por tanto la CPU debe realizar un acceso a memoria para extraerlo. Ejemplo:

    add dir # donde x es una dirección de memoria cualquiera.

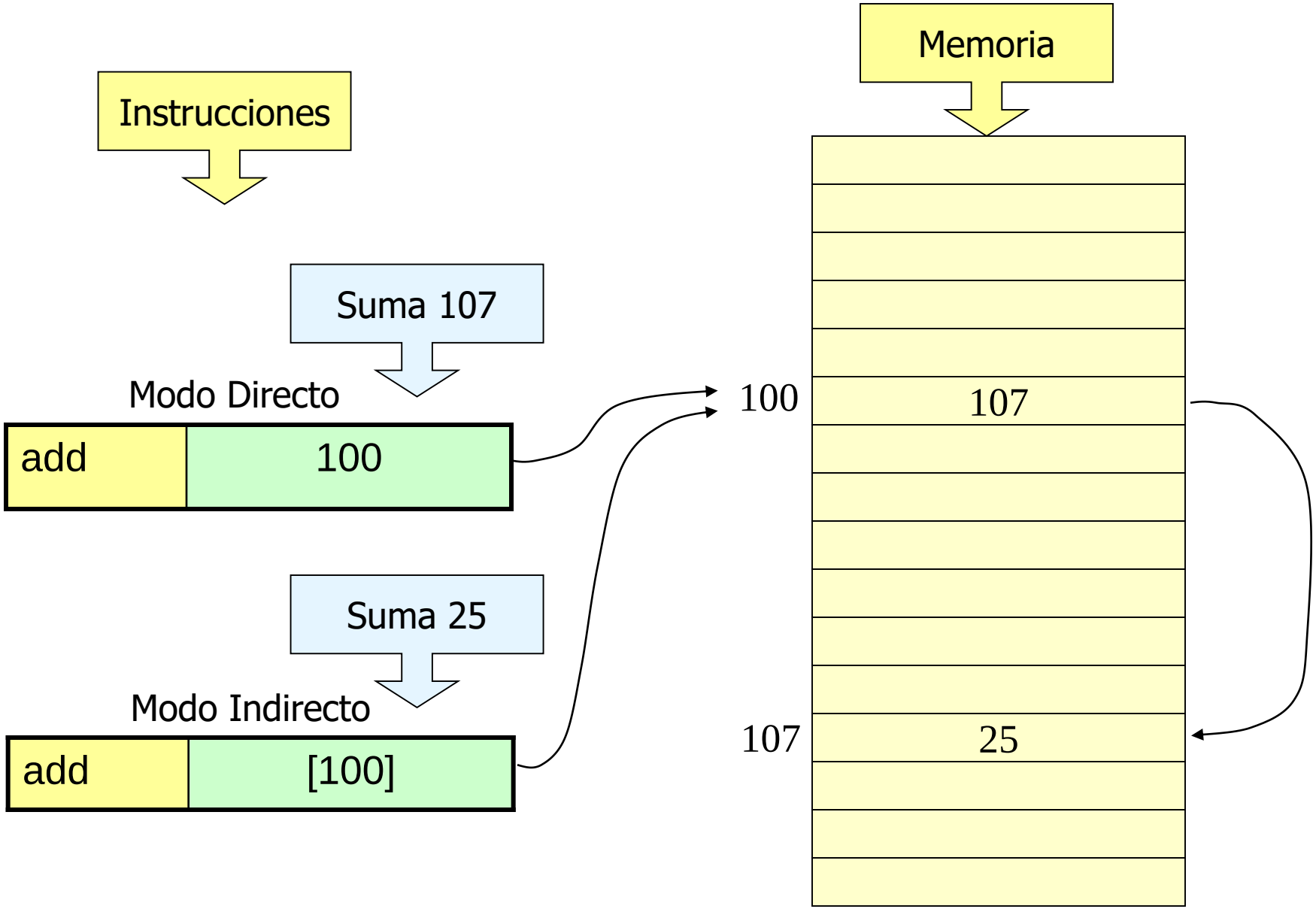
    add 100 # 100 es la dirección absoluta.
  - **Indirecto**. La instrucción contiene la dirección de la dirección del dato. En este caso la CPU debe realizar dos acceso a memoria para extraerlo. Ejemplo:

    add [100] # 100 es una dirección de memoria que contiene otra dirección

Aplicación de los modos de direccionamiento



Modos de direccionamiento con direcciones absolutas



El ciclo de instrucción

- Es un conjunto de etapas que utiliza el procesador para ejecutar una instrucción de máquina. Cada instrucción se ejecuta siguiendo este mismo ciclo repetitivo. Las etapas que conforman el ciclo de instrucción, son las siguientes:
  - **Fetch de Instrucción**. Realiza la búsqueda de la instrucción en memoria. El procesador extrae la dirección desde el contador de programa, **PC**, para conocer la ubicación de esa instrucción. Luego se realiza la lectura de la instrucción, quedando ésta en el registro de datos. El contador de programa es incrementado en 1, para contener la dirección de la siguiente instrucción.
  - **Decodificación**. El código de operación de la instrucción que se encuentra en el **DR**, se deposita en el registro de instrucciones **IR**. Luego, los circuitos de control interpretan ese código de operación para determinar qué operación se va a ejecutar.
  - **Fetch de Operando**. Realiza la búsqueda de un operando en memoria, si es necesario. Generalmente es la misma instrucción quien proporciona la dirección del operando al procesador, según el modo de direccionamiento.
  - **Ejecución de la Operación**. Una vez conocida la naturaleza de la instrucción, y el operando ha sido extraído desde memoria, el procesador ejecuta la operación indicada por el código de operación contenido en la instrucción.

Codificación de las instrucciones de máquina

- Otro aspecto importante, es que el procesador sólo entiende información de tipo binaria, lo que significa que tanto el código de operación como el operando, vienen especificados en forma binaria dentro de la instrucción.
- Para ello, cada operación debe poseer un código binario. Por tanto, y como ejemplo, vamos a definir una tabla con los nombres de las instrucciones y su correspondiente código:

Instrucción de máquina	Código Binario
add	011
sub	101
mul	001
div	000
lda	111
str	110
jmp	101
end	100

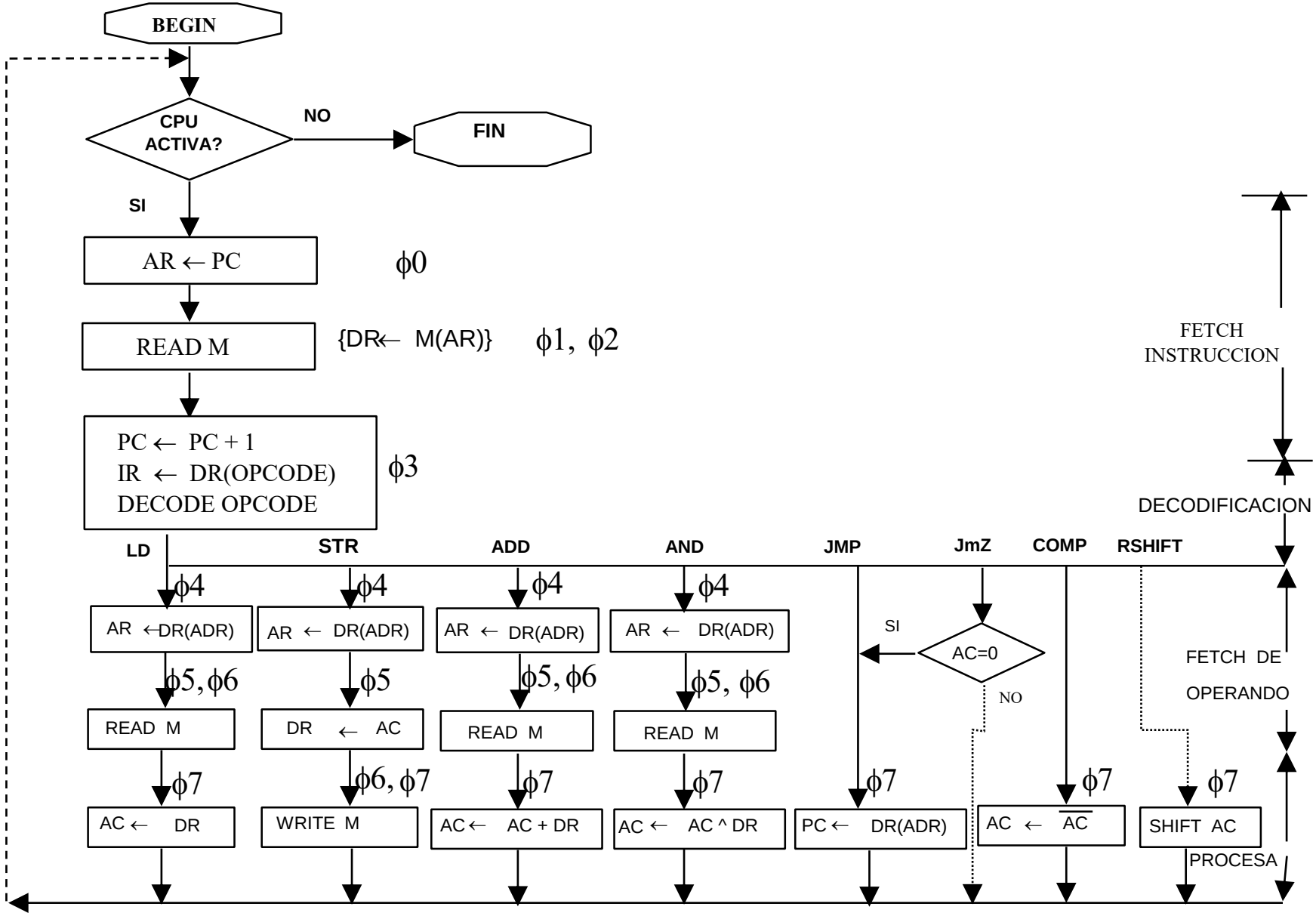
Set completo de instrucciones

Instrucciones de Transferencia de Datos	**lda #Num**	AC ← Num
	lda Dir	AC ← MD[Dir]
	lda [Dir]	AC ← MD[[Dir]]
	str Dir	MD[Dir] ← AC
	rda Dir	MD[Dir] ← Teclado
	wrt	Monitor ← AC

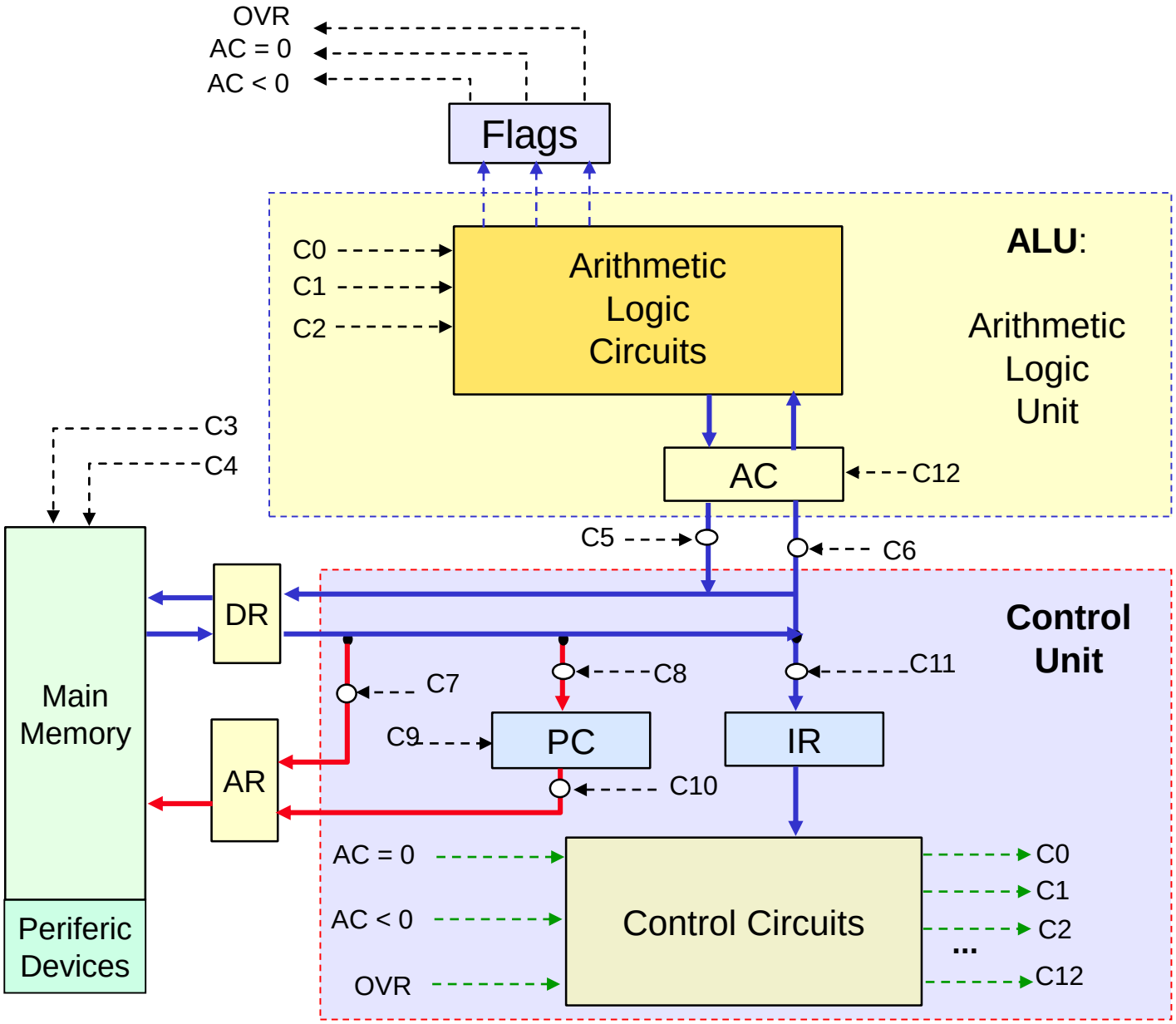
Instrucciones Aritméticas y Lógicas Todas las operaciones asumen que el primer operando está almacenado en el Acumulador	**add #Num**	AC ← AC + Num
	add Dir	AC ← AC + MD[Dir]
	add [Dir]	AC ← AC + MD[[Dir]]
	sub #Num	AC ← AC - Num
	sub Dir	AC ← AC - MD[Dir]
	sub [Dir]	AC ← AC - MD[[Dir]]
	mul #Num	AC ← AC * Num
	mul Dir	AC ← AC * MD[Dir]
	mul [Dir]	AC ← AC * MD[[Dir]]
	div #Num	AC ← AC / Num
	div Dir	AC ← AC / MD[Dir]
	div [Dir]	AC ← AC / MD[[Dir]]
	sqr	AC ← raíz(AC)
	and Dir	AC ← AC and Dir
	or Dir	AC ← AC or Dir
	not	AC ← negación AC

Instrucciones de transferencia de control	**jmp Dir_p**	**PC ← Dir_p**
	jmz Dir_p	**Si (AC = 0) entonces PC ← Dir_p**
	jml Dir_p	**Si (AC < 0) entonces PC ← Dir_p**

Ciclo de Instrucción de la CPU básica de von Neumann



La CPU de von Neumann: las señales de control



Imprimir los números pares

```
01h  lda #0
02h  wrt
03h  add #2
04h  jmp 02h
```

Evaluar la función:

$$f(x) \begin{cases} (3*x + 5) / x & \text{si } x > 0 \\ 5*x^2 & \text{si } x <= 0 \end{cases}$$

```
00h  rda 80
01h  lda 80
02h  jmz 08
03h  jml 08
04h  mul #3
05h  add #5
06h  div 80
07h  jmp 0Ah
08h  mul #5
09h  mul 80
0Ah  wrt
```

1. Sumar dos números ingresados por teclado.

2. Calcular el perímetro de un círculo a partir de del ingreso del radio

3. Calcular el valor de un tercer ángulo ingresando por teclado el valor de los dos restantes.

4. Determinar el mayor de dos números positivos ingresados por teclado