Tema 4. Introducción a la Arquitectura de Computadores

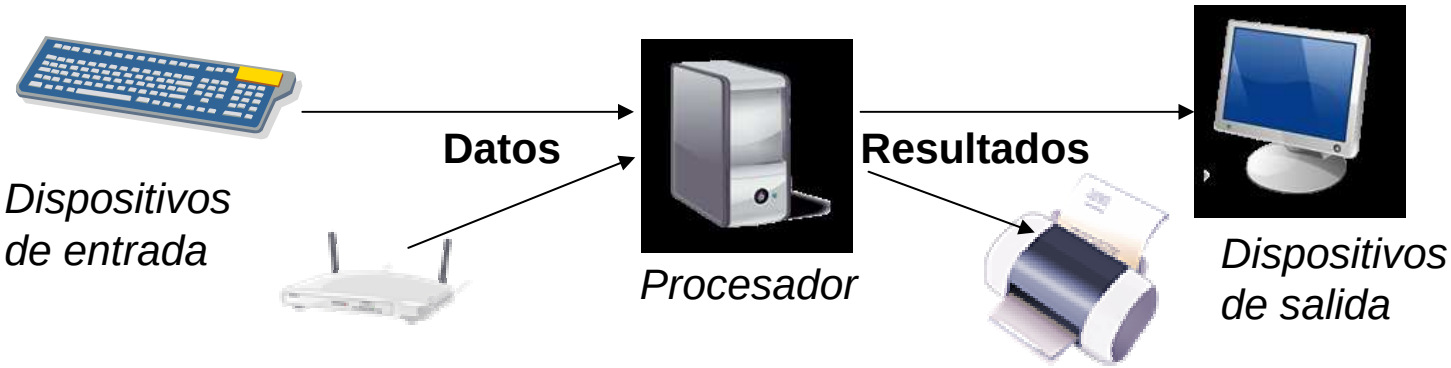
Departamento de Ingeniería y Tecnología de Computadores

Índice

Índice

Estructura funcional de un ordenador

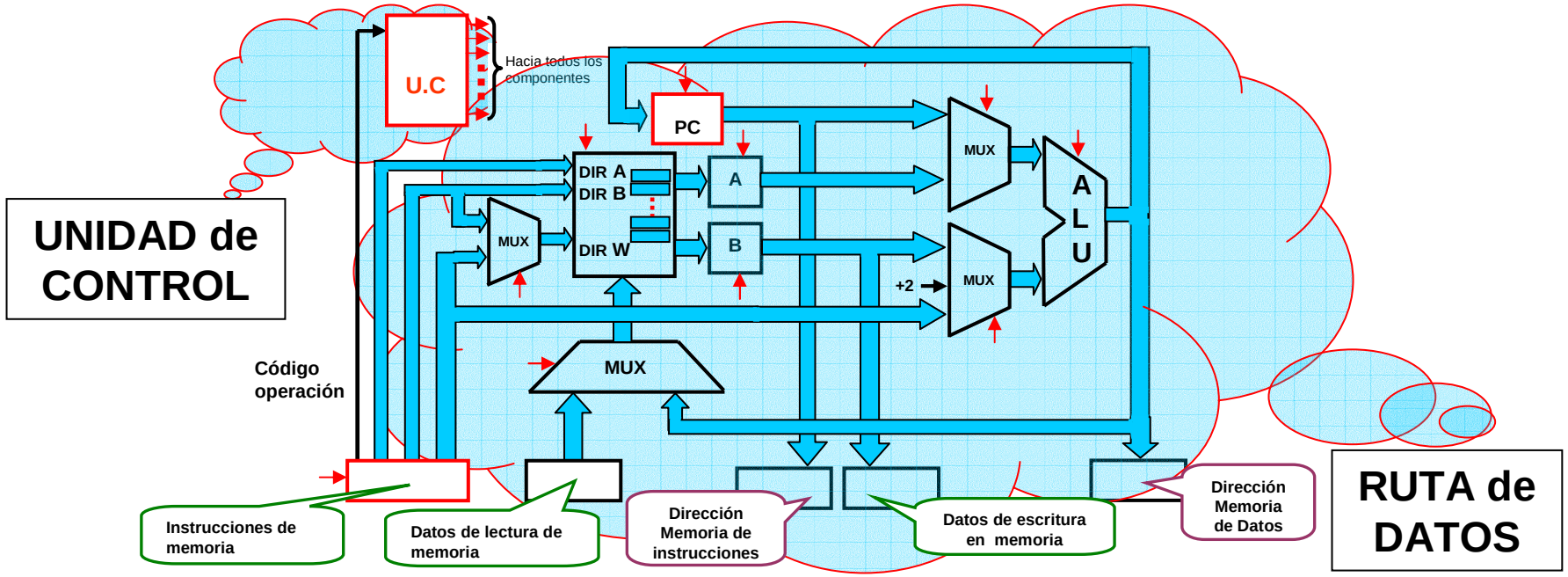
- Ordenador ≡ Máquina que procesa información y produce unos resultados.
  - La información a procesar puede:
    - Estar almacenada previamente en el computador.
    - Ser introducida desde el exterior.
  - Los resultados producidos:
    - Se almacenan en el propio computador.
    - Se saca al exterior.



- Programa ≡ Conjunto de instrucciones que debe ejecutar el computador sobre los datos para procesarlos y obtener un resultado.

Estructura funcional de un ordenador

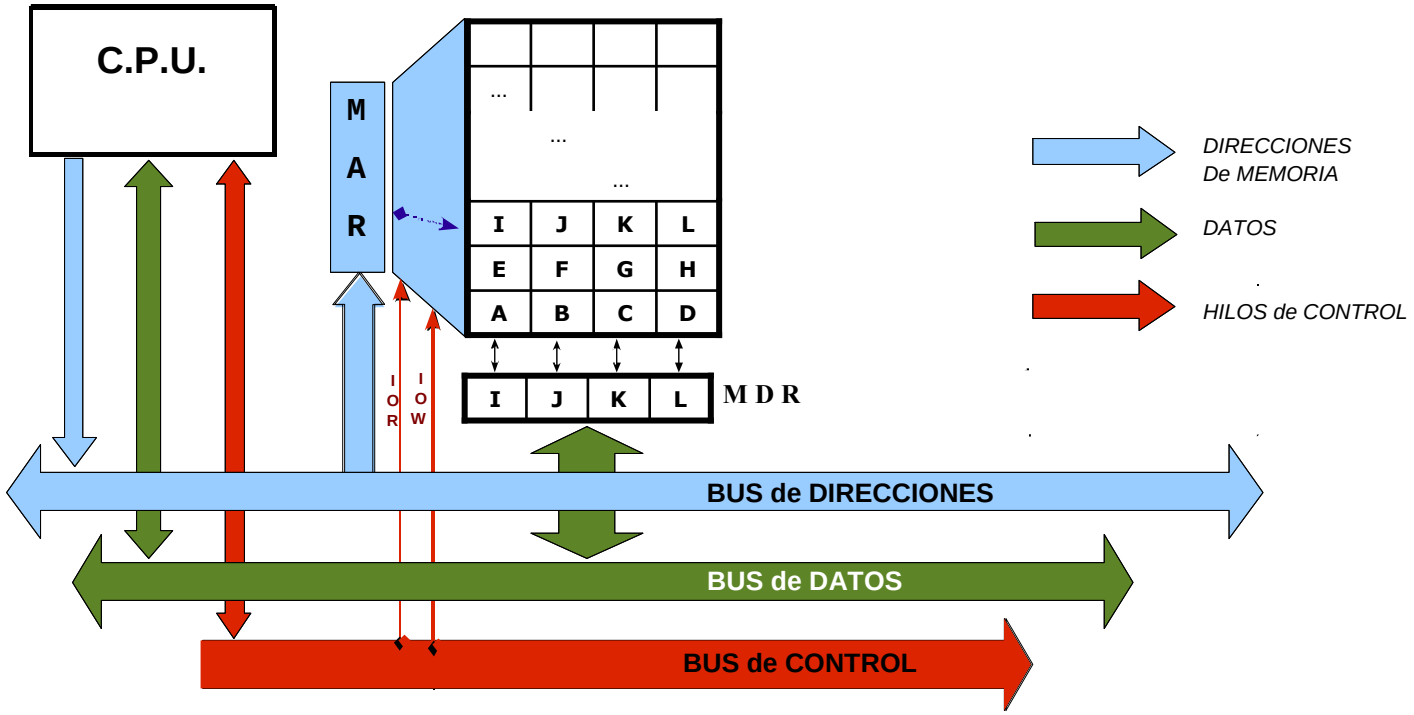
- Tradicionalmente, los computadores se dividen en 3 bloques:
  1. Procesador o CPU (*Central Processing Unit*): Encargado de la ejecución de las instrucciones. Se divide a su vez en:
     - Unidad de control:
       - Busca las instrucciones de la memoria.
       - Decodifica las instrucciones que se van a ejecutar.
       - Genera los valores de las señales (señales de control) que dicen lo que hay que hacer para la ejecución de las instrucciones.
     - Camino de datos: unidades funcionales que realizan las operaciones de las instrucciones.



4.1 Estructura funcional de un ordenador

2.- Memoria:

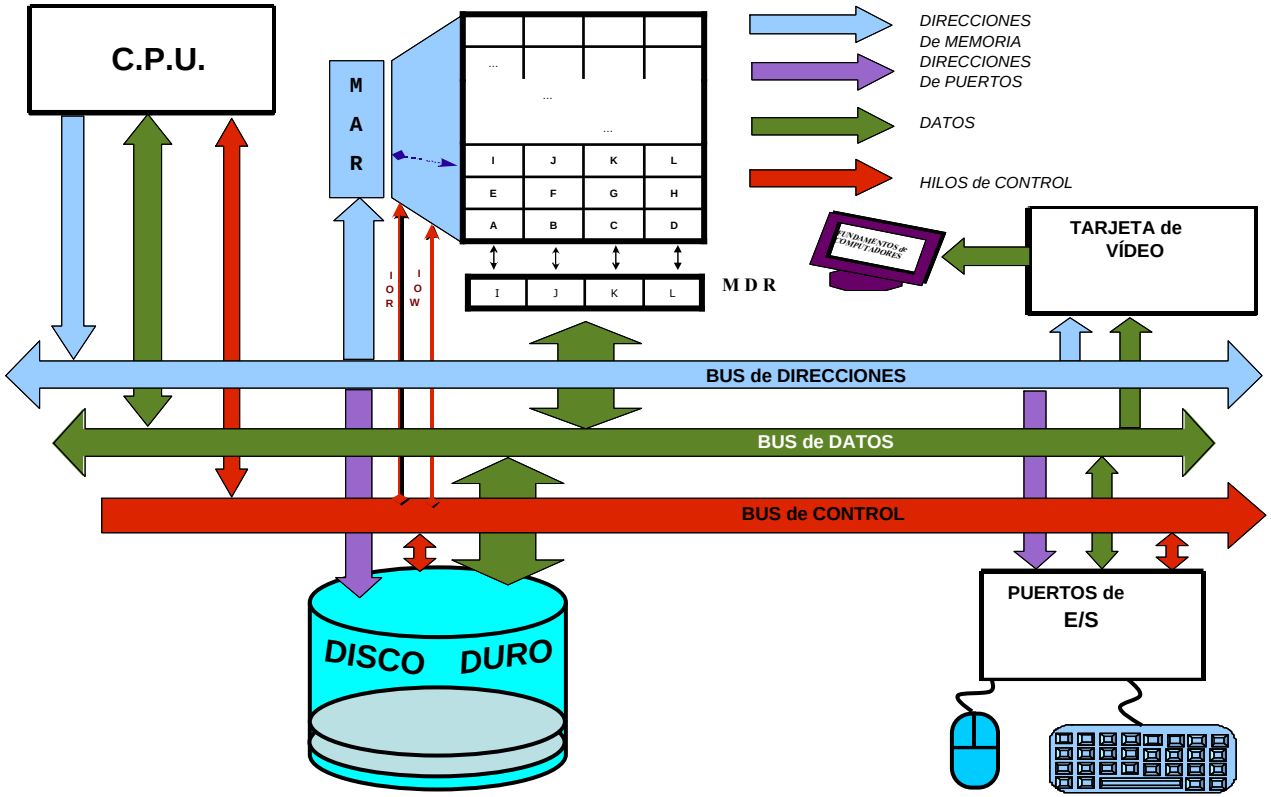
- Almacena los datos e instrucciones de los programas activos.
- Conceptualmente, gran estructura dividida en posiciones, cada una con una dirección única.
- En realidad, la memoria se organiza como una jerarquía con varios niveles, cada uno con características diferenciadas (ver más adelante).
- Para leer hay que indicar la dirección de memoria (MAR, *Memory Address Register*) y activar IOR., para escribir, además de la dirección, hay que proporcionar el dato (MDR, *Memory Data Register*) y activar IOW.



Estructura funcional de un ordenador

3.- Interconexión y dispositivos de Entrada/Salida:

- Dispositivos usados para interaccionar con el usuario del computador (teclado, monitor, ratón, impresora, tarjeta de red, …). Se gobiernan por direcciones de puerto ≠ Direcciones de memoria.
- Interconexiones entre los distintos componentes del ordenador (bus PCI, bus de memoria, …).



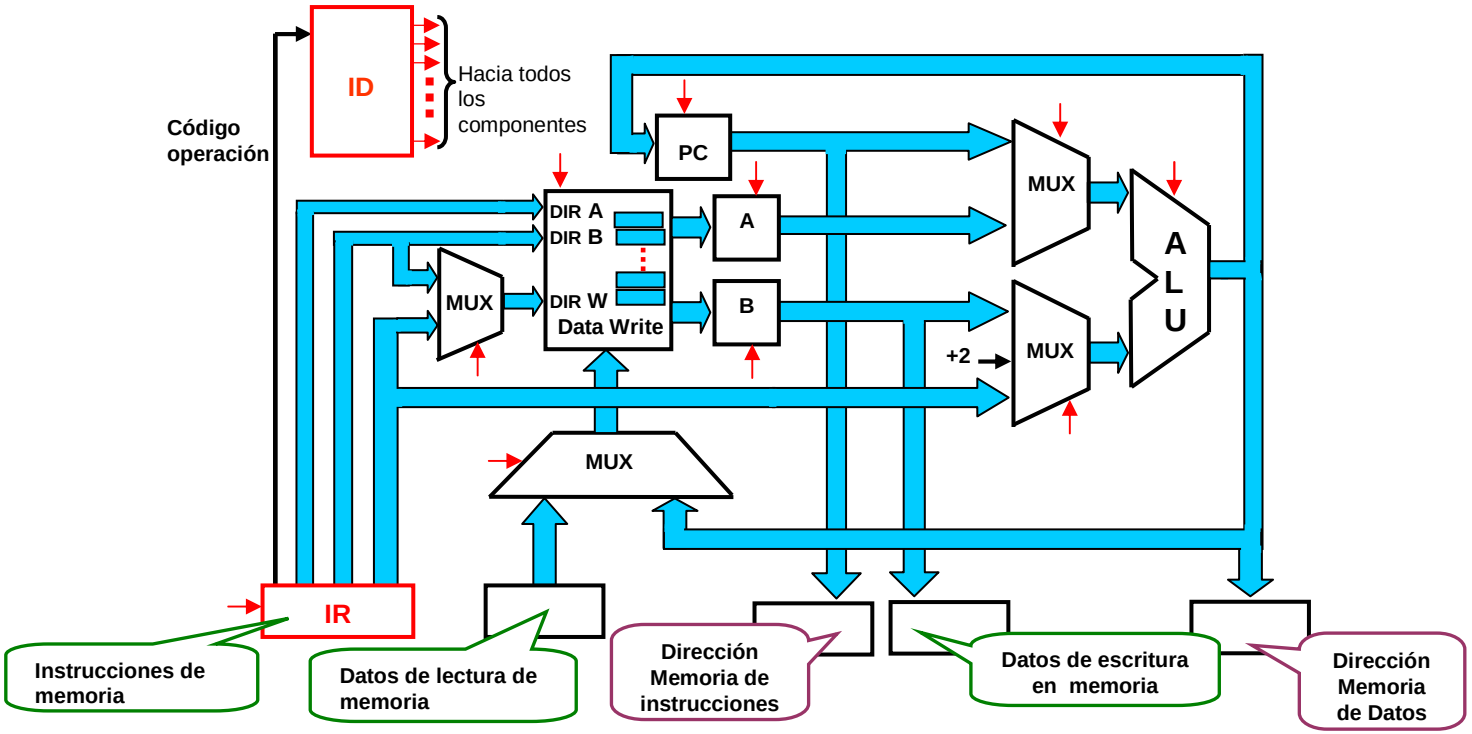
Índice

El procesador

- Dispositivo digital encargado de llevar a cabo las operaciones indicadas por los programas software
  - Instrucción ≡ Conjunto de símbolos que representa una orden de operación o tratamiento para el computador.
  - Programa ≡ Conjunto ordenado de instrucciones que indican al computador una tarea completa.
- Puesto que todos sus componentes pueden ser incluidos en un circuito integrado (*microchip*) se habla normalmente de microprocesador.
- Existen multitud de ejemplos, siendo los productos de AMD e Intel los más empleados en el mundo de los PCs.

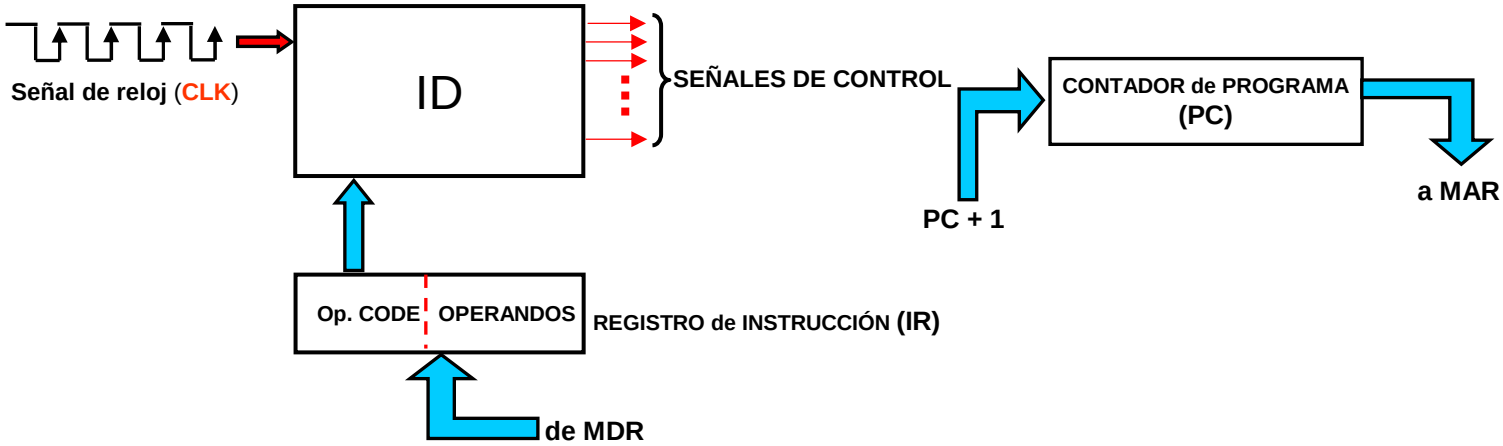
Organización básica del procesador

- Dividido en unidad de control y camino de datos.
- La unidad de control es la parte "activa" del procesador, puesto que es la encargada de buscar las instrucciones de la memoria y ordenar su ejecución al camino de datos.
- La unidad de control se comunica con el camino de datos a través de las *señales de control.*



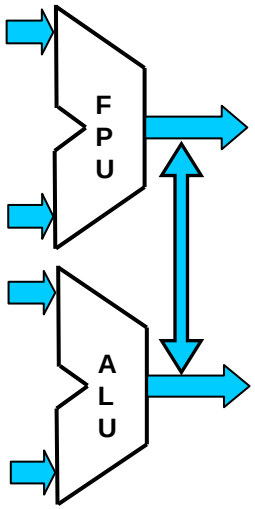
Organización básica del procesador

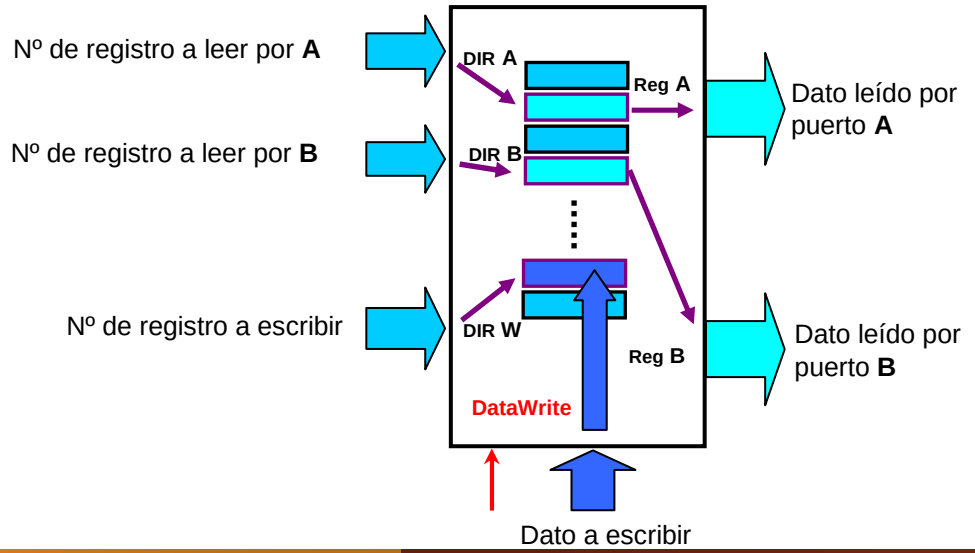
- De forma simplificada, la **unidad de control** dispone de los siguientes elementos:
  - Registro **Contador de Programa** o **PC** (*Program Counter*): almacena la dirección de memoria de la siguiente instrucción a ejecutar.
  - **Registro de Instrucción** o **IR** (*Instruction Register*): almacena la instrucción a ejecutar. El código de operación (Op. CODE) indica la instrucción a ejecutar.
  - **Decodificador de instrucciones** o **ID** (*Instruction Decoder*): genera los valores de las señales de control para la ejecución de cada instrucción
  - **Reloj** o generador de pulsos: marca el ritmo al cual se llevan a cabo las operaciones dentro del procesador.
    - Tiempo de ciclo: periodo de esta señal.
    - Frecuencia de reloj (en GHz o miles de millones de ciclos por segundo) = inversa del tiempo de ciclo.



Organización básica del procesador

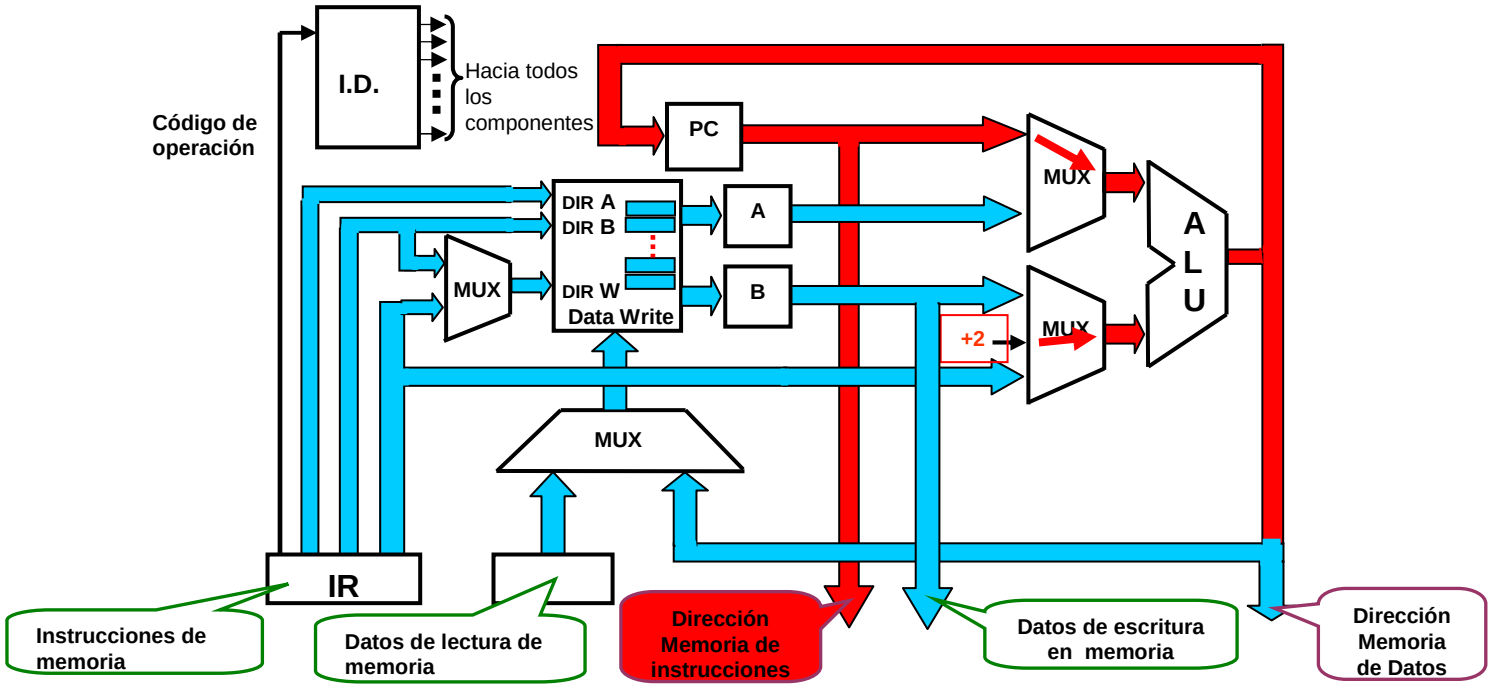
- Por su parte, dentro del **camino de datos** estaría:
  - Unidad aritmético-lógica o **ALU** (*Arithmetic-Logic Unit*): encargada de la realización de operaciones aritméticas sobre números enteros y las operaciones lógicas.
  - **Unidad de coma flotante** o FPU (*Floating-Point Unit*): realiza las operaciones aritméticas con operando de punto flotante.
  - Banco de registros:
    - Estructura que aglutina un número pequeño de registros .
    - Cada registro contiene un dato que puede ser operado por la ALU o FPU.
    - Para leer un registro hay que indicar el número del registro a leer y se obtiene su contenido (puerto de lectura).
    - Para escribir un registro hay que indicar el número de registro a escribir y el dato (puerto de escritura (**DataWrite**)).
    - 2 Bancos de registros separados para enteros (registros de propósito general) y punto flotante.





Organización básica del procesador

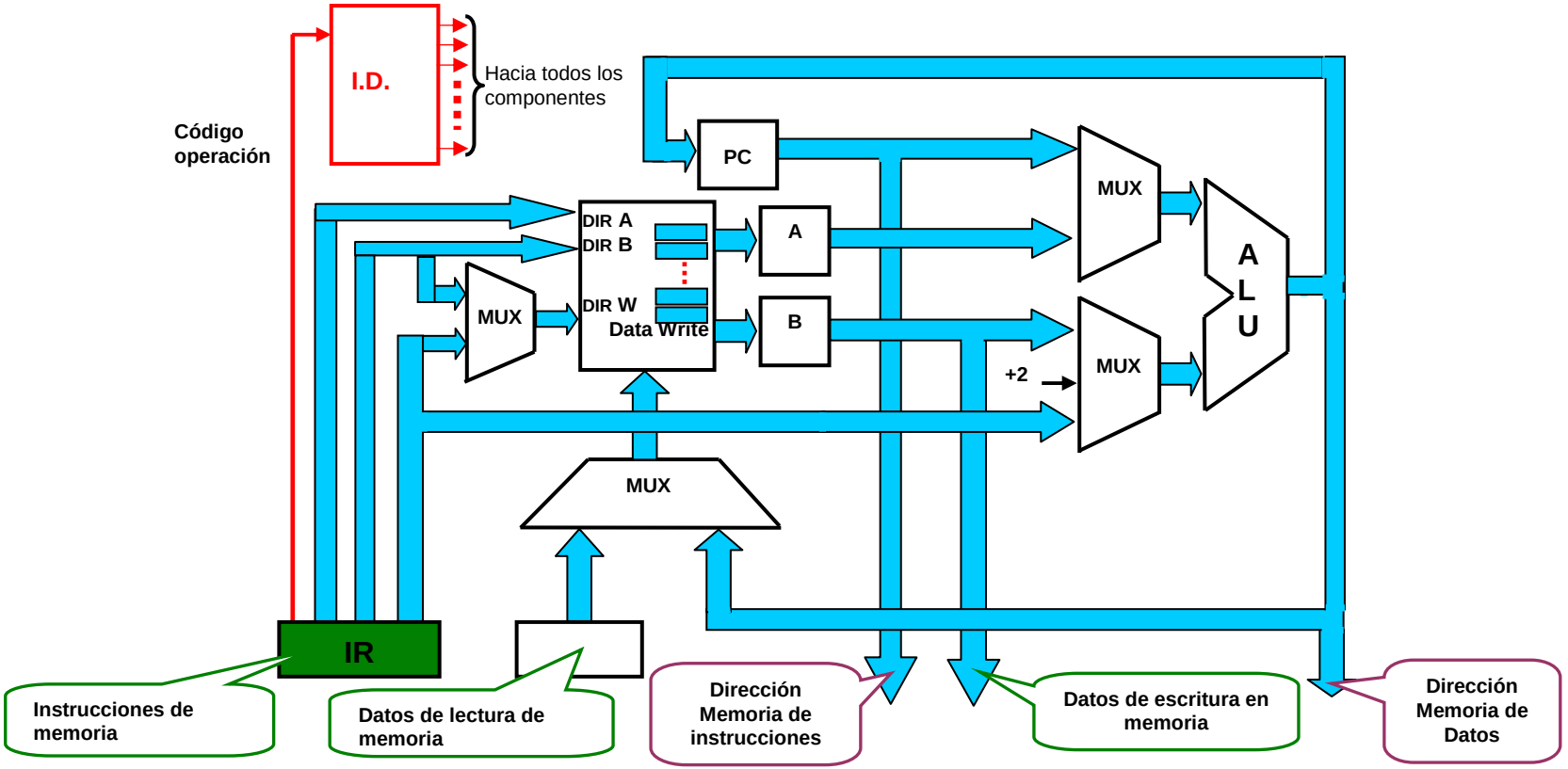
- Pasos para la ejecución de una instrucción:
  1. **Búsqueda de la instrucción e incremento del PC:**
     - Se lee la posición de memoria cuya dirección aparece en el registro contador de programa (PC) y el valor leído se almacena en el registro de instrucción (IR).
     - Se incrementa el PC en 2 (en esta arquitectura de ejemplo, las instrucciones son todas de 2 Bytes) para que contenga la dirección de la siguiente instrucción a ejecutar:



Organización básica del procesador

2. Decodificación de la instrucción:

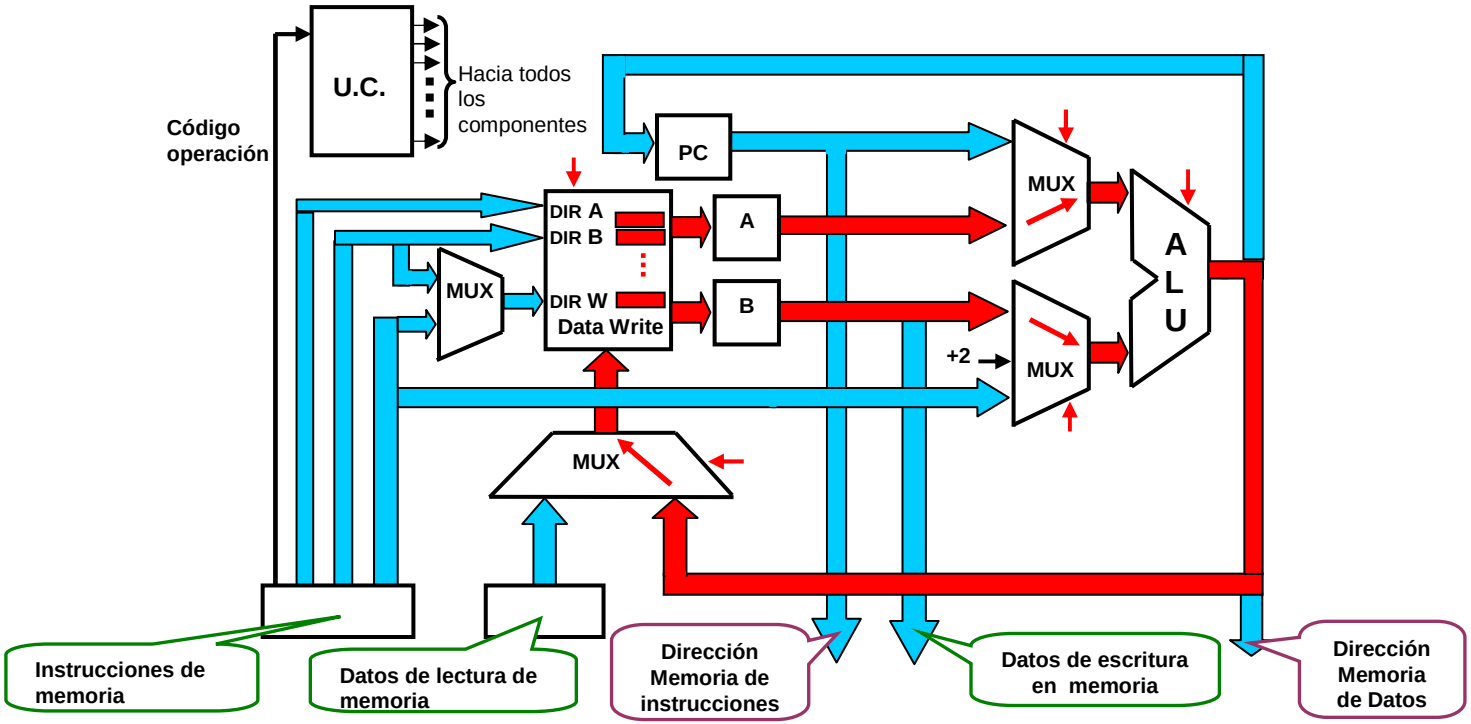
- El decodificador de instrucciones toma los bits del registro IR que identifican a la instrucción (código de operación).
- En función del valor de dichos bits, genera los valores apropiados para las señales de control.



Organización básica del procesador

3. Ejecución de la instrucción:

- Las unidades del camino de datos realizan las operaciones indicadas por la unidad de control mediante las señales de control.
  - Por ejemplo, la ALU podría tener que llevar a cabo una operación de suma, se leerían operandos del banco de registros y escribirían resultados en el mismo …



Organización básica del procesador

- La arquitectura del repertorio de instrucciones (o ISA de *Instruction Set Architecture)* es la interfaz entre la circuitería y el nivel más bajo de programación.
- Entre otros aspectos, determina las instrucciones que el procesador puede ejecutar.
- Podemos agrupar las instrucciones de un ISA en:
  - **Instrucciones de transferencia de datos entre procesador y memoria:**
    - Las instrucciones de carga (*load*) copian el contenido de la posición de memoria especificada por la instrucción en un registro del procesador. Las de almacenamiento (*store*) hacen lo contrario.
  - **Instrucciones aritmético-lógicas:**
    - Instrucciones de suma, resta, and, or, comparación, …
  - **Instrucciones de control:**
    - Instrucciones de salto condicional, incondicional, llamadas a subrutinas, vuelta de subrutinas, …
  - **Instrucciones de punto flotante:**
    - Instrucciones de suma, resta, multiplicación, división,… en punto flotante.
  - **Instrucciones de sistema:**
    - Llamadas al SO, excepciones, interrupciones…

Organización básica del procesador

- Durante los años 70 se popularizaron ISAs con muchas instrucciones (mas de 200 en algunos casos).
  - Complex Instrucion Set Computer (**CISC**).
- Posteriormente se constató que era preferible (para poder obtener implementaciones de alto rendimiento) ISAs con pocas instrucciones (menos de 100), sencillas y que se puedan ejecutar rápidamente.
  - Reduced Instruction Set Computer (**RISC**)
- A día de hoy todos los procesadores se construyen basados en la filosofía RISC.
- Los procesadores para PC de AMD e Intel tanto de 32 bits (IA-32) como de 64 bits (AMD64) son un caso curioso:
  - Al nivel más bajo de programación presentan un ISA CISC.
  - Se implementan como si tuvieran un ISA RISC.
  - Entre los pasos 1 y 2 se añade un nuevo paso en el que las instrucciones CISC son traducidas a instrucciones RISC.

Organización básica del procesador

- Seguiremos, paso a paso, la ejecución de un trozo de programa de 4 instrucciones:
  - 4 Instrucciones, a 3 pasos cada una, 4*3=12 pasos (12 ciclos).
  - Supondremos una memoria de 1024 bytes (10 bits de dirección), registros A, B, C, D, … de 8 bits, e instrucciones de 16 bits (2 bytes).

Dirección (10 bits)	**Instrucción decodificada**	**Instr. Codificada (2 bytes)**
0001001000	load A, @Mem[1001010000] *(Tipo: carga de memoria)*	00000010 01010000
0001001010	load B, 11111101 *(Tipo: carga de valor inmediato)*	00010100 11111101
0001001100	add B, A (A=A+B) *(Tipo: aritmético lógica)*	01000100 00000000
0001001110	store A, @Mem[1001010001] *(Tipo: almacenam. en memoria)*	00100010 01010001
0001010000	…	…

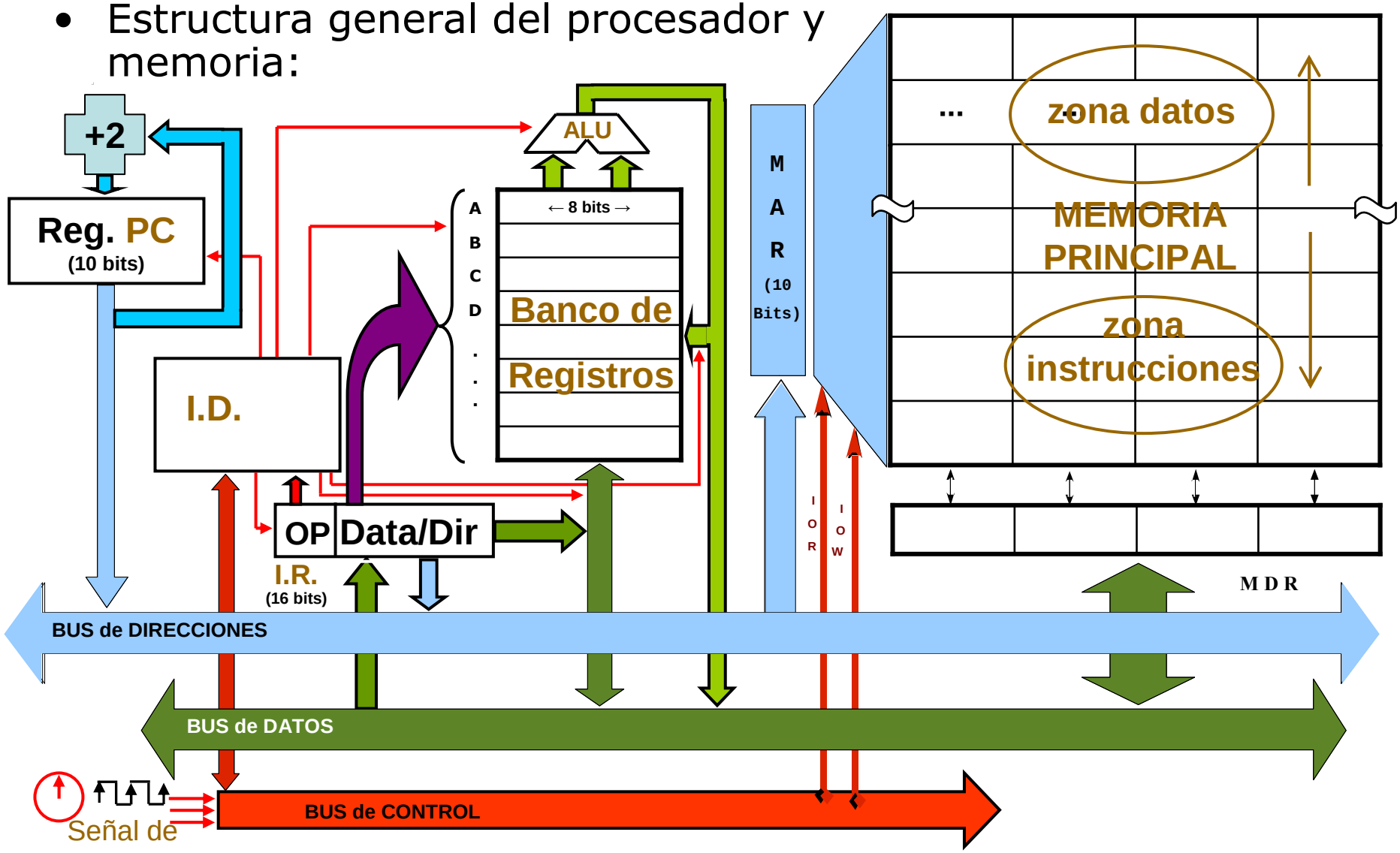
Organización básica del procesador

1. Formato *load de memoria*: 4 bits código operación (0000) + 2 bits registro destino (A=00, B=01, C=10, …) + 10 bits dirección memoria.
2. Formato *load valor inmediato*: 4 bits código operación (0001) + 2 bits registro destino + 2 bits vacíos + 8 bits valor inmediato.
3. Formato *add*: 4 bits código operación (0100) + 2 bits registro fuente + 2 bits registro destino + 8 bits vacíos.
4. Formato *store*: 4 bits código operación (0010) + 2 bits registro fuente + 10 bits dirección memoria.

	Instrucción decodificada	**Instr. Codificada**
1	load A, @Mem[1001010000] *(Tipo: carga de memoria)*	0000 00 1001010000
2	load B, 11111101 *(Tipo: carga de valor inmediato)*	0001 01 00 11111101
3	add B, A (A=A+B) *(Tipo: aritmético lógica)*	0100 01 00 00000000
4	store A, @Mem[1001010001] *(Tipo: almacenam. en memoria)*	0010 00 1001010001

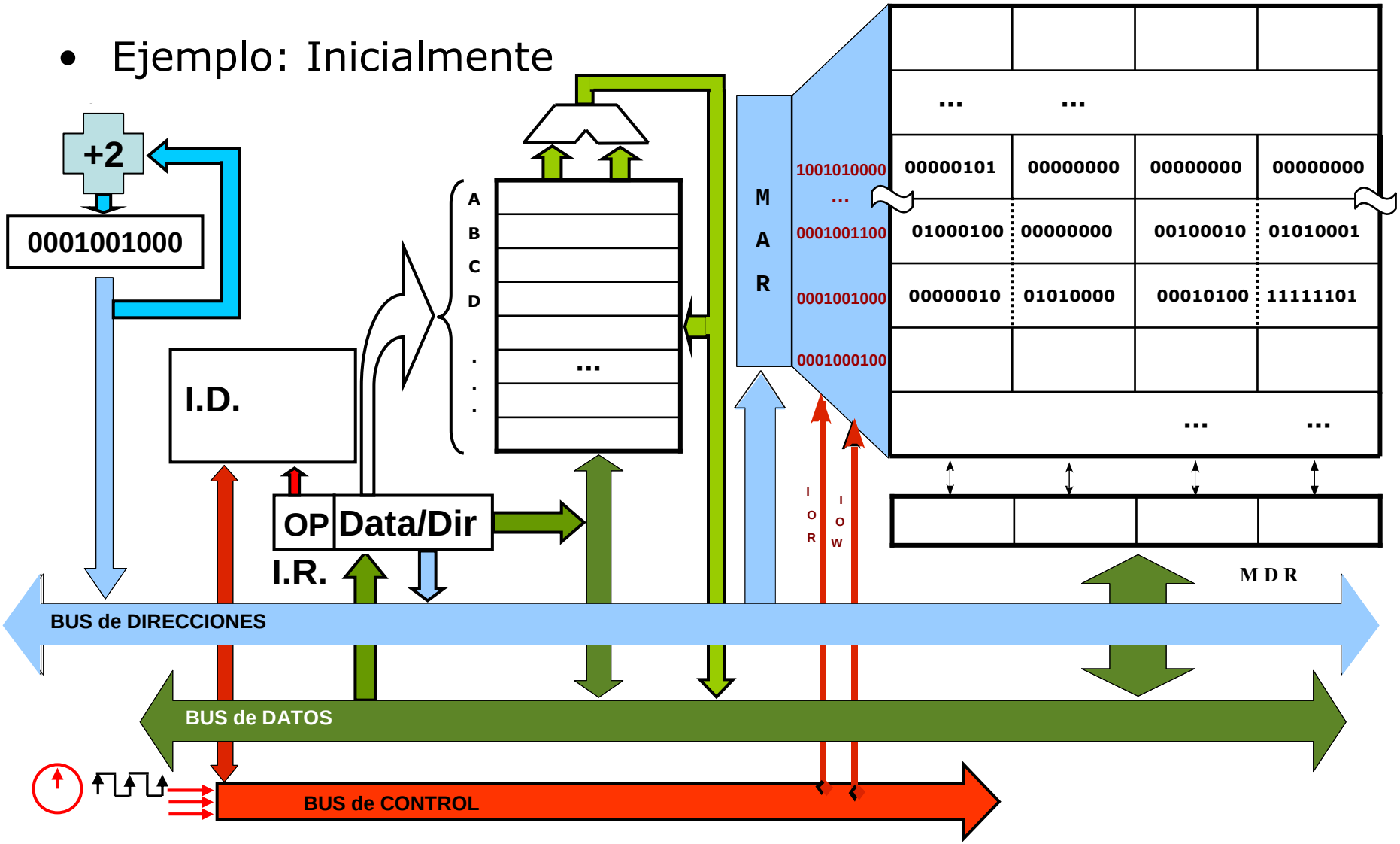
Organización básica del procesador

- Estructura general del procesador y memoria:



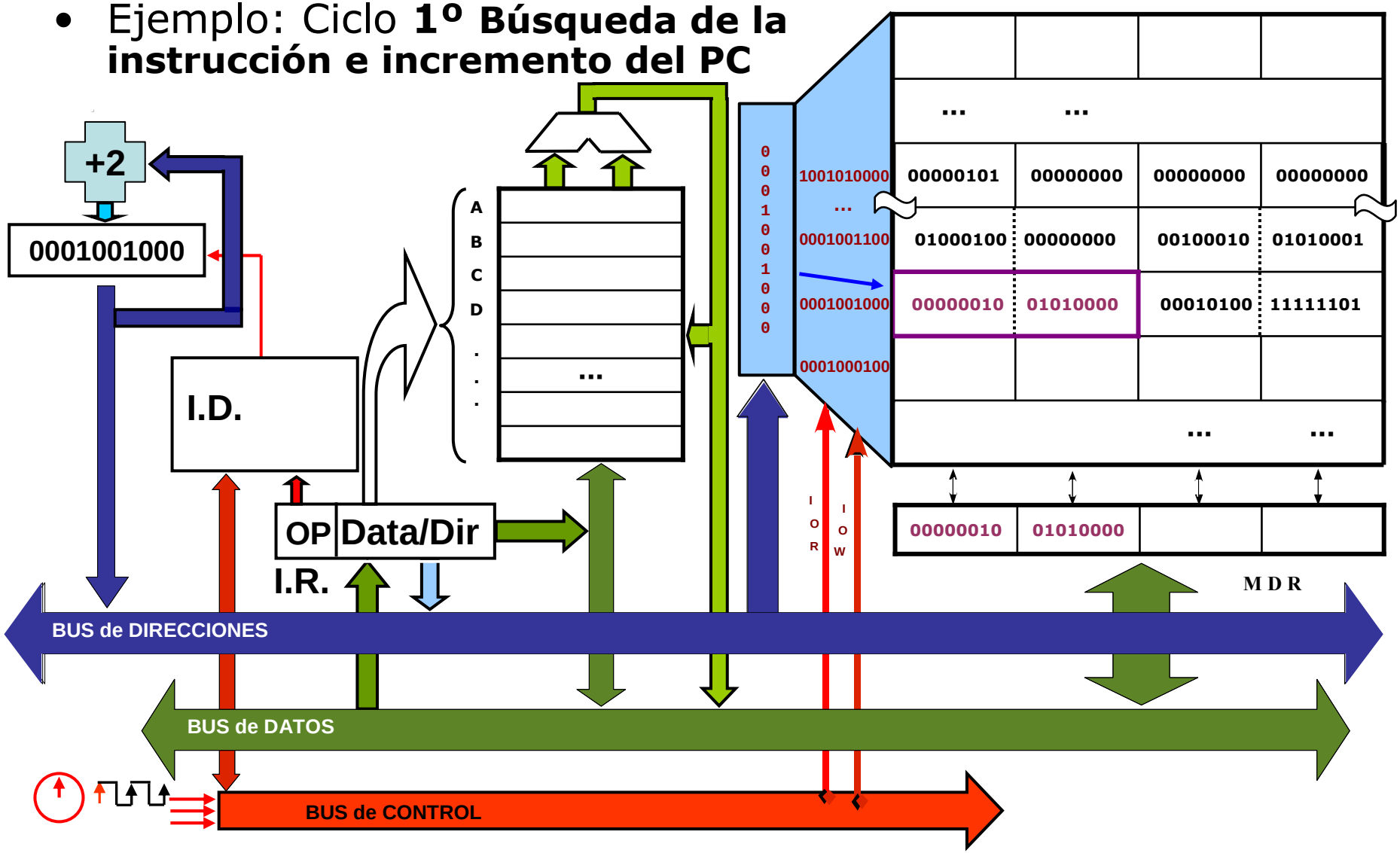
Organización básica del procesador

- Ejemplo: Inicialmente



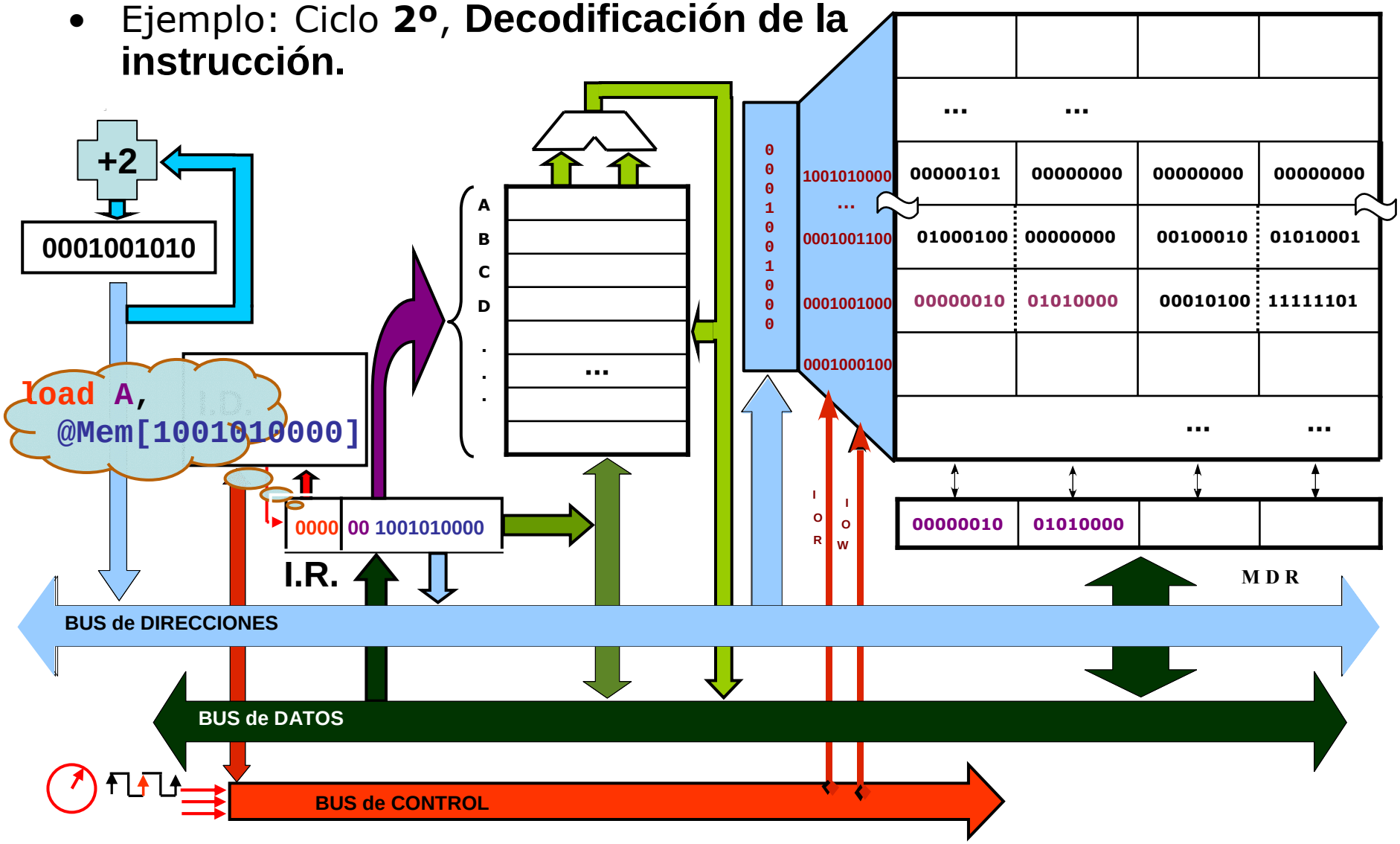
Organización básica del procesador

- Ejemplo: Ciclo **1º Búsqueda de la instrucción e incremento del PC**



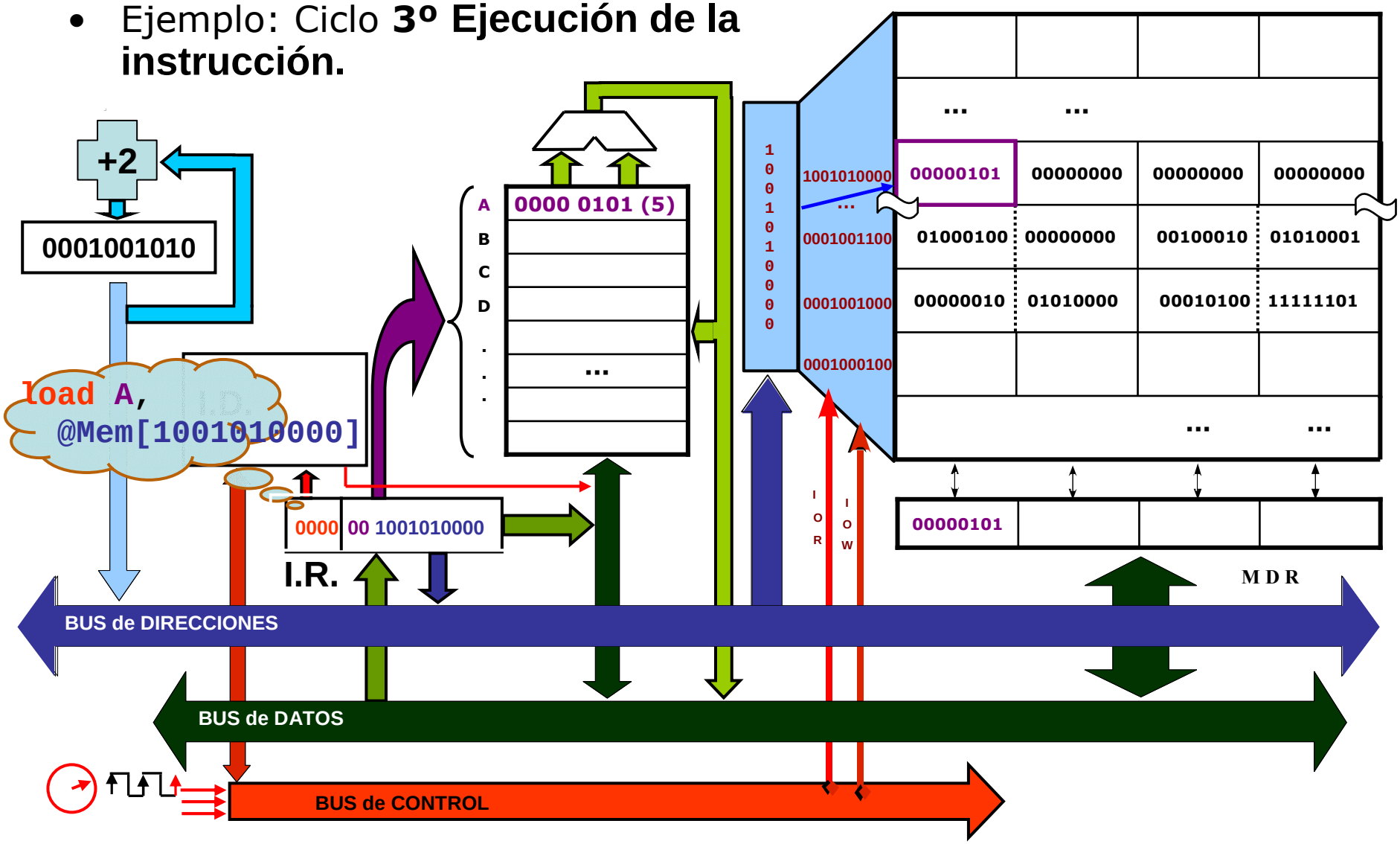
Organización básica del procesador

- Ejemplo: Ciclo **2º**, **Decodificación de la instrucción.**



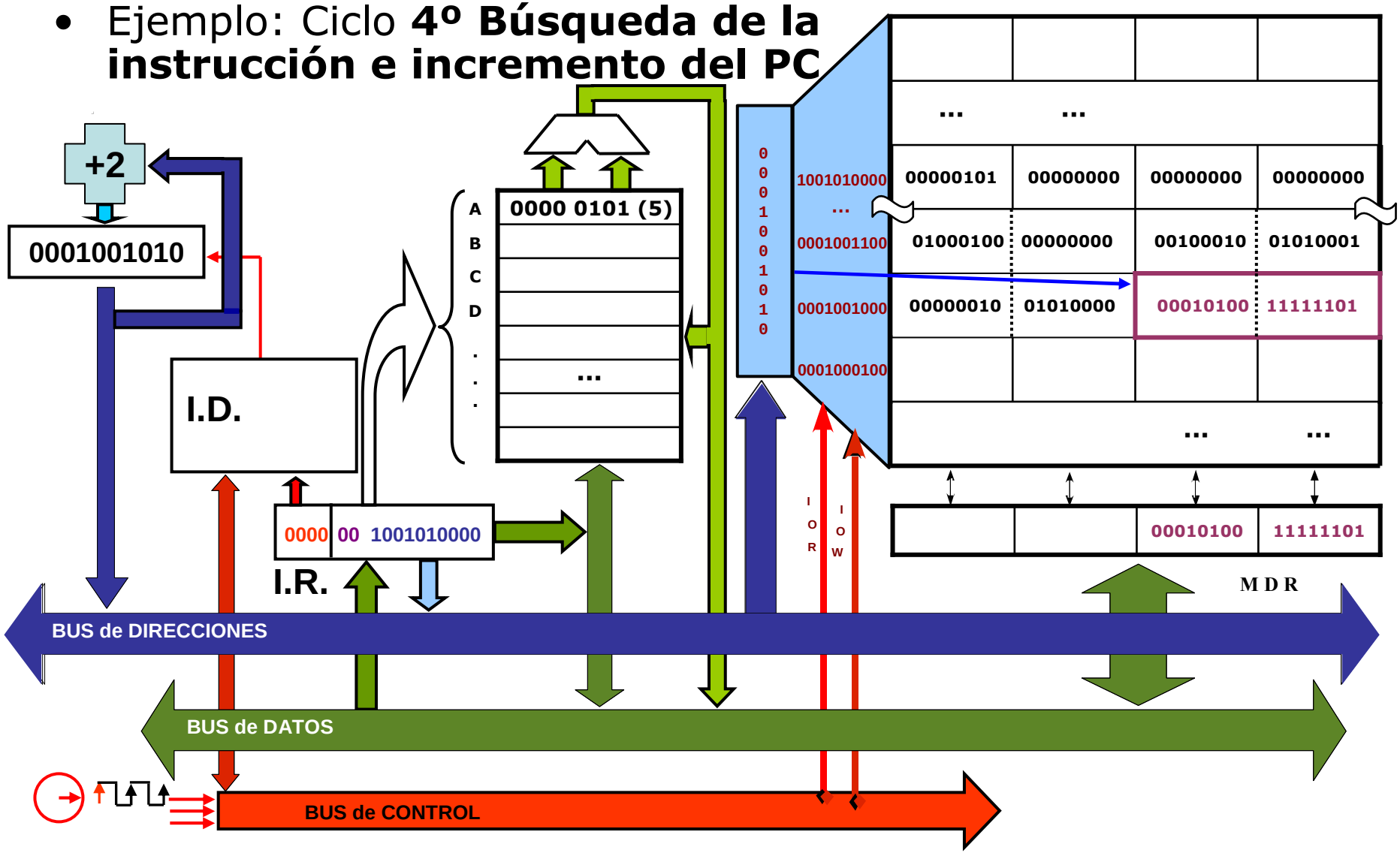
Organización básica del procesador

- Ejemplo: Ciclo **3º Ejecución de la instrucción.**



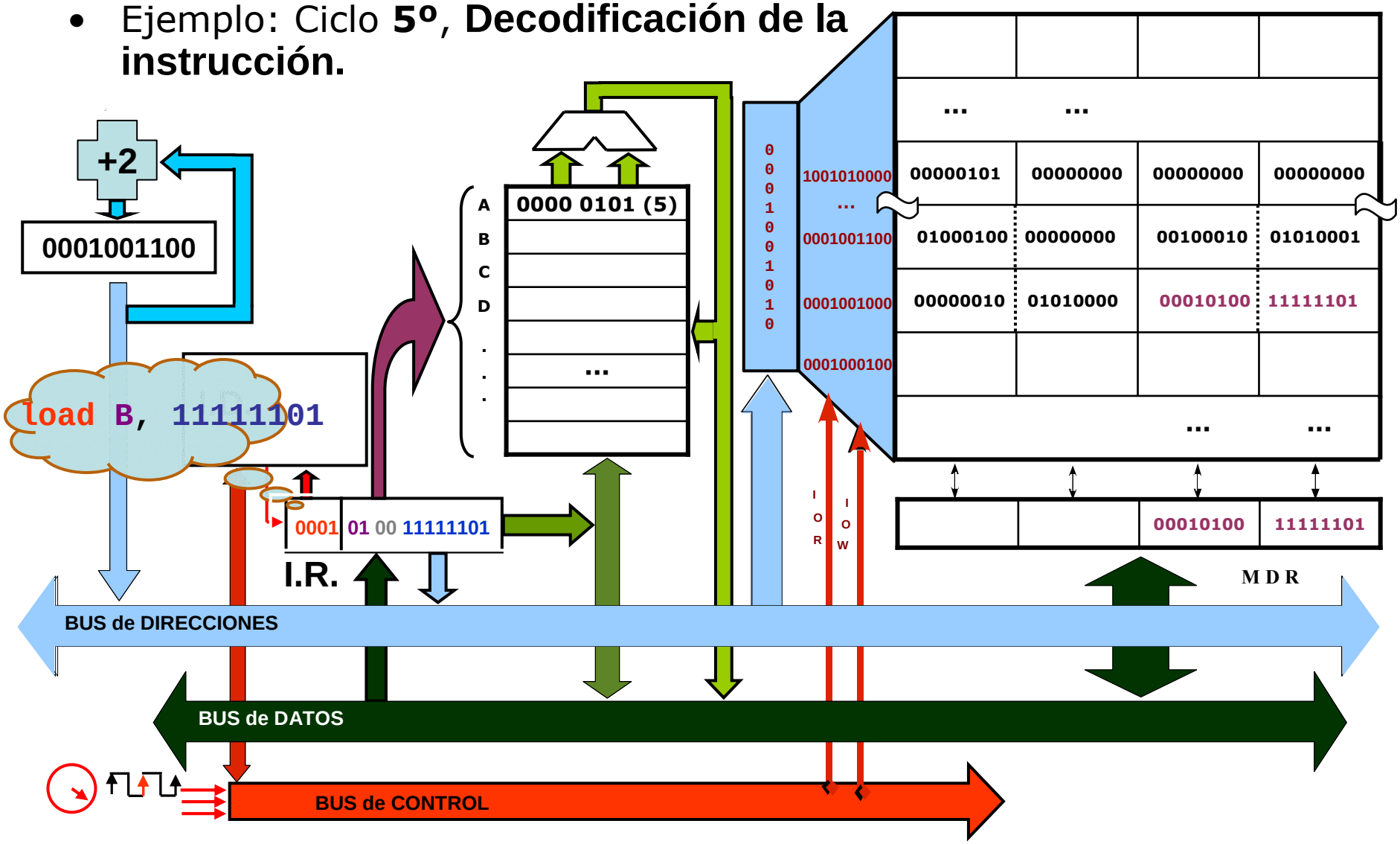
Organización básica del procesador

- Ejemplo: Ciclo **4º Búsqueda de la instrucción e incremento del PC**



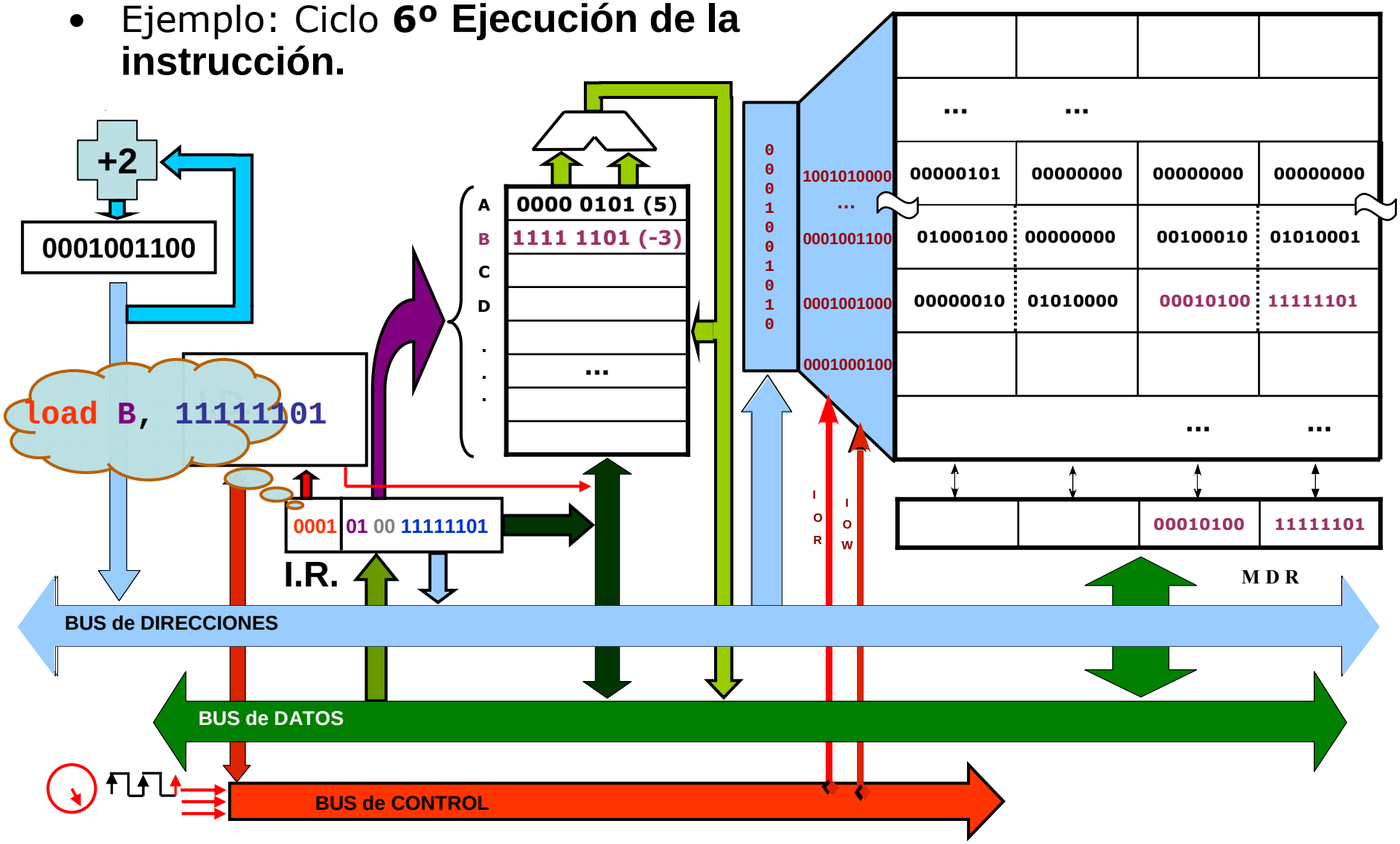
Organización básica del procesador

- Ejemplo: Ciclo **5º**, **Decodificación de la instrucción.**



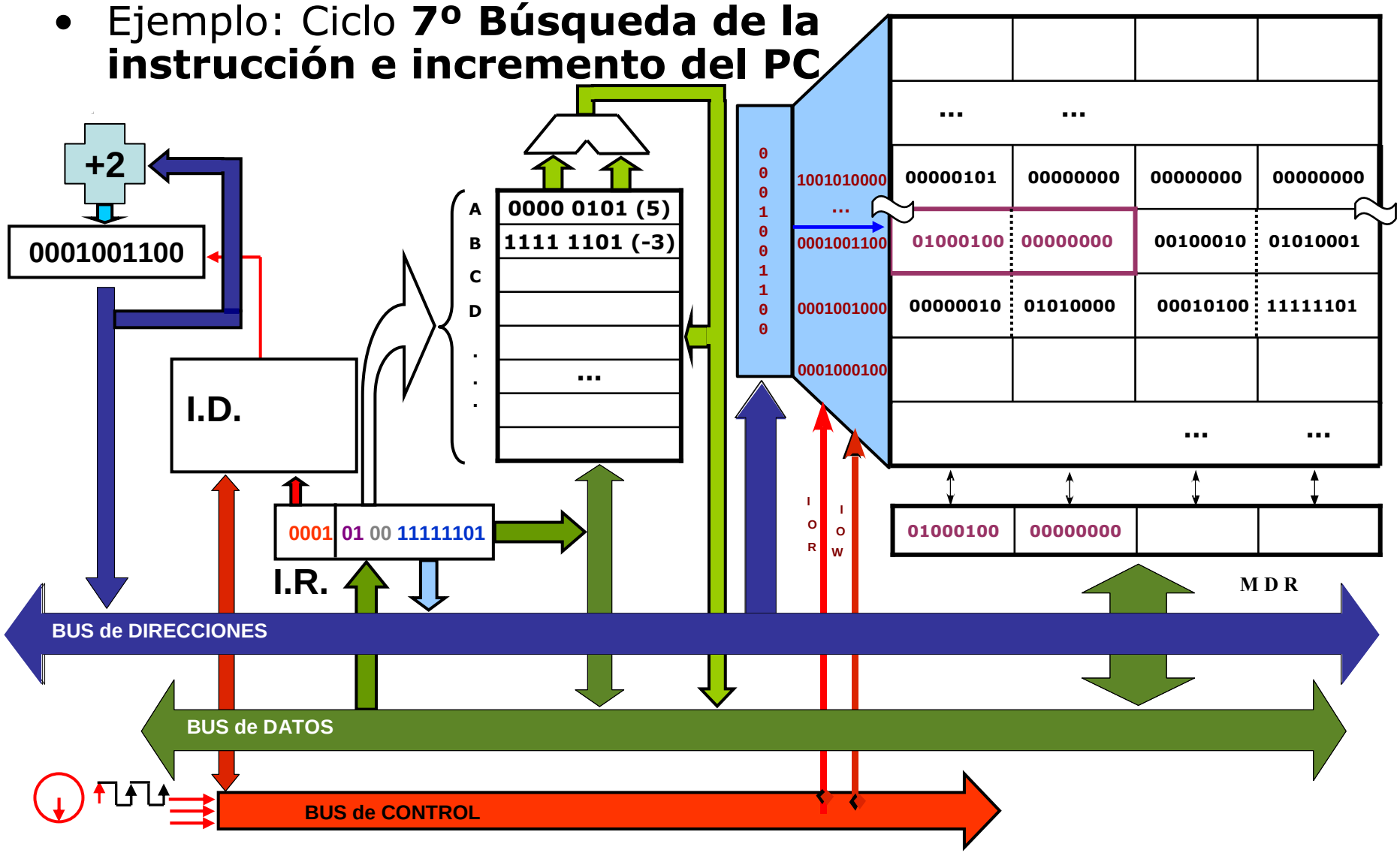
Organización básica del procesador

- Ejemplo: Ciclo **6º Ejecución de la instrucción.**



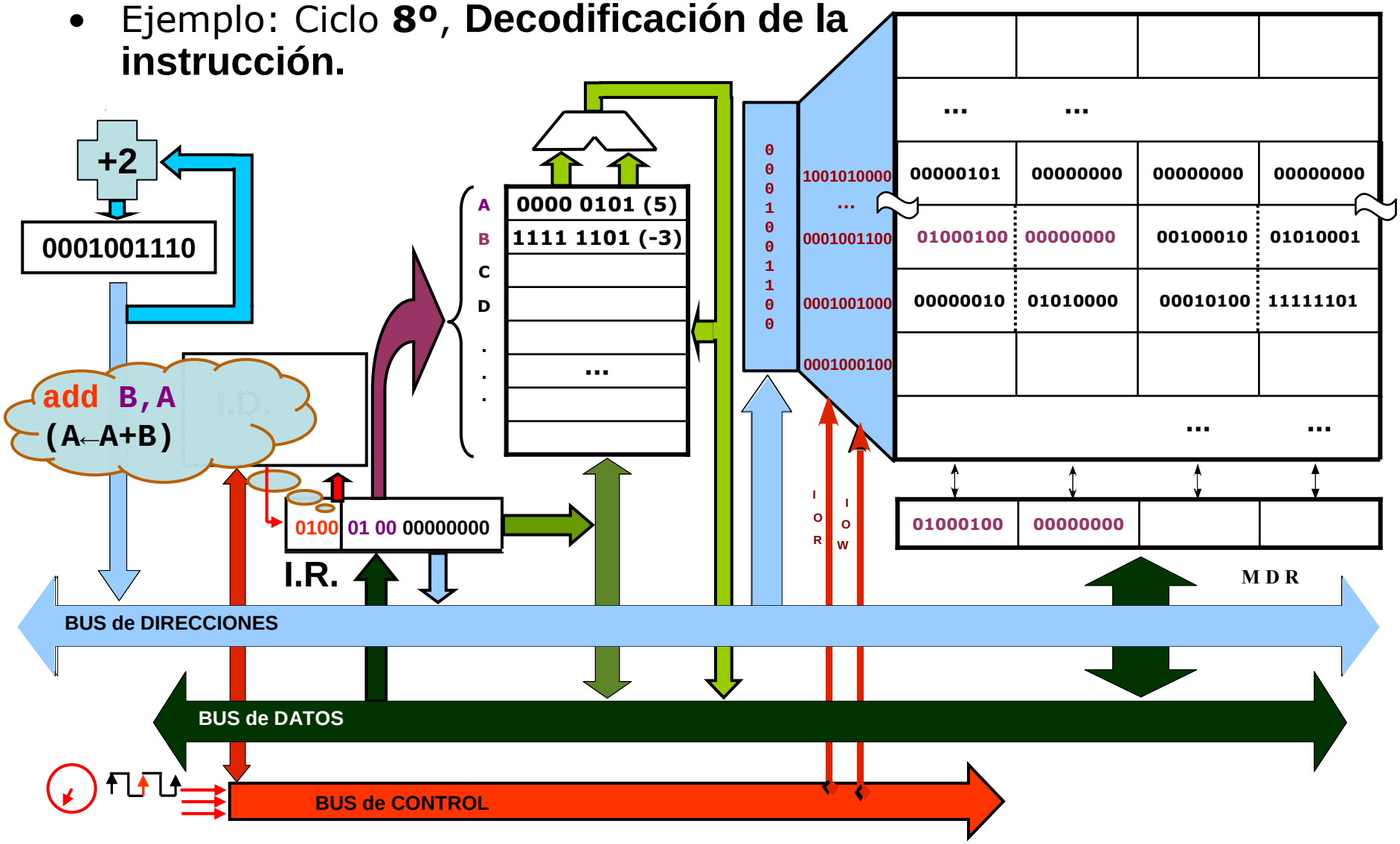
Organización básica del procesador

- Ejemplo: Ciclo **7º Búsqueda de la instrucción e incremento del PC**



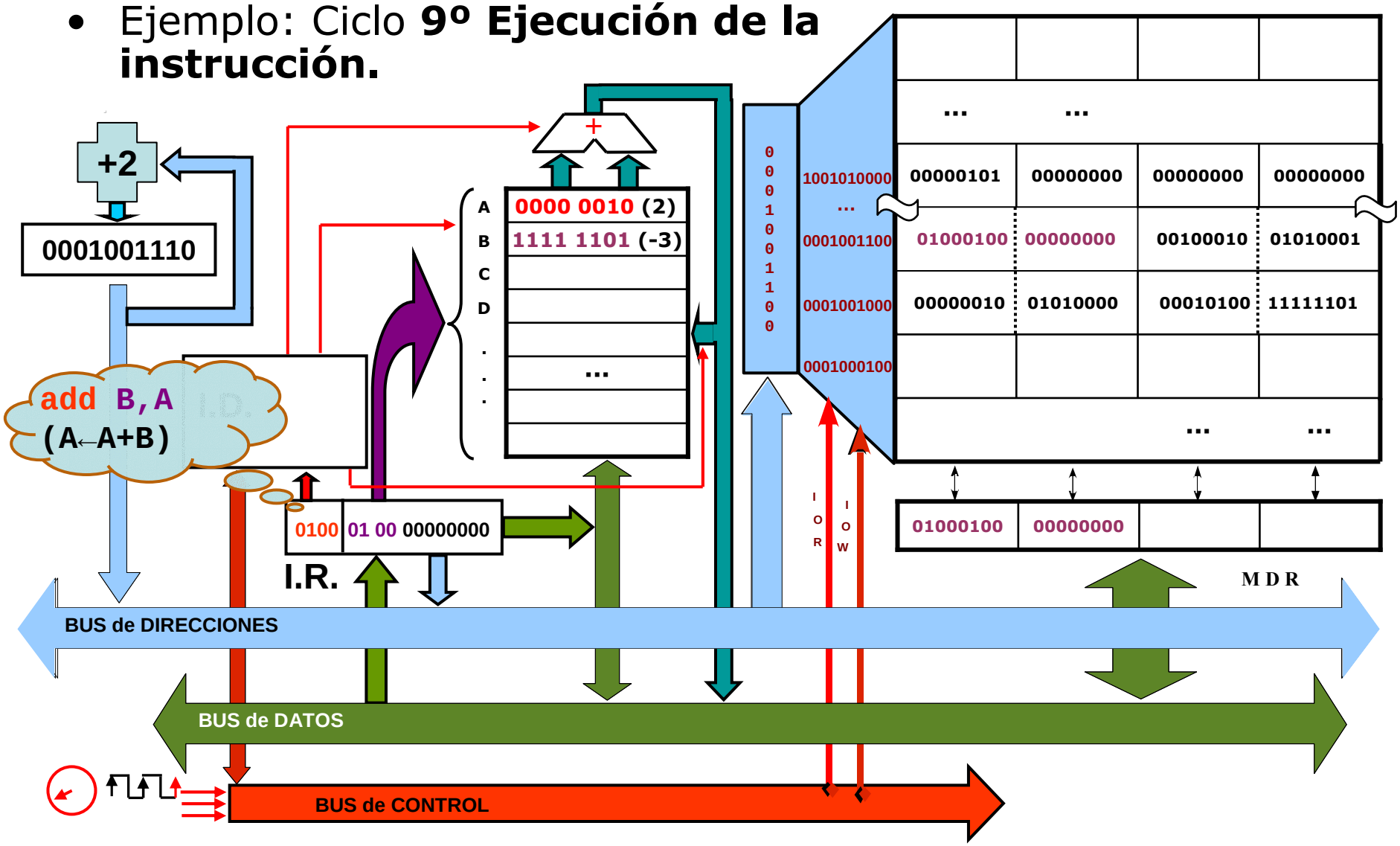
Organización básica del procesador

- Ejemplo: Ciclo **8º**, **Decodificación de la instrucción.**



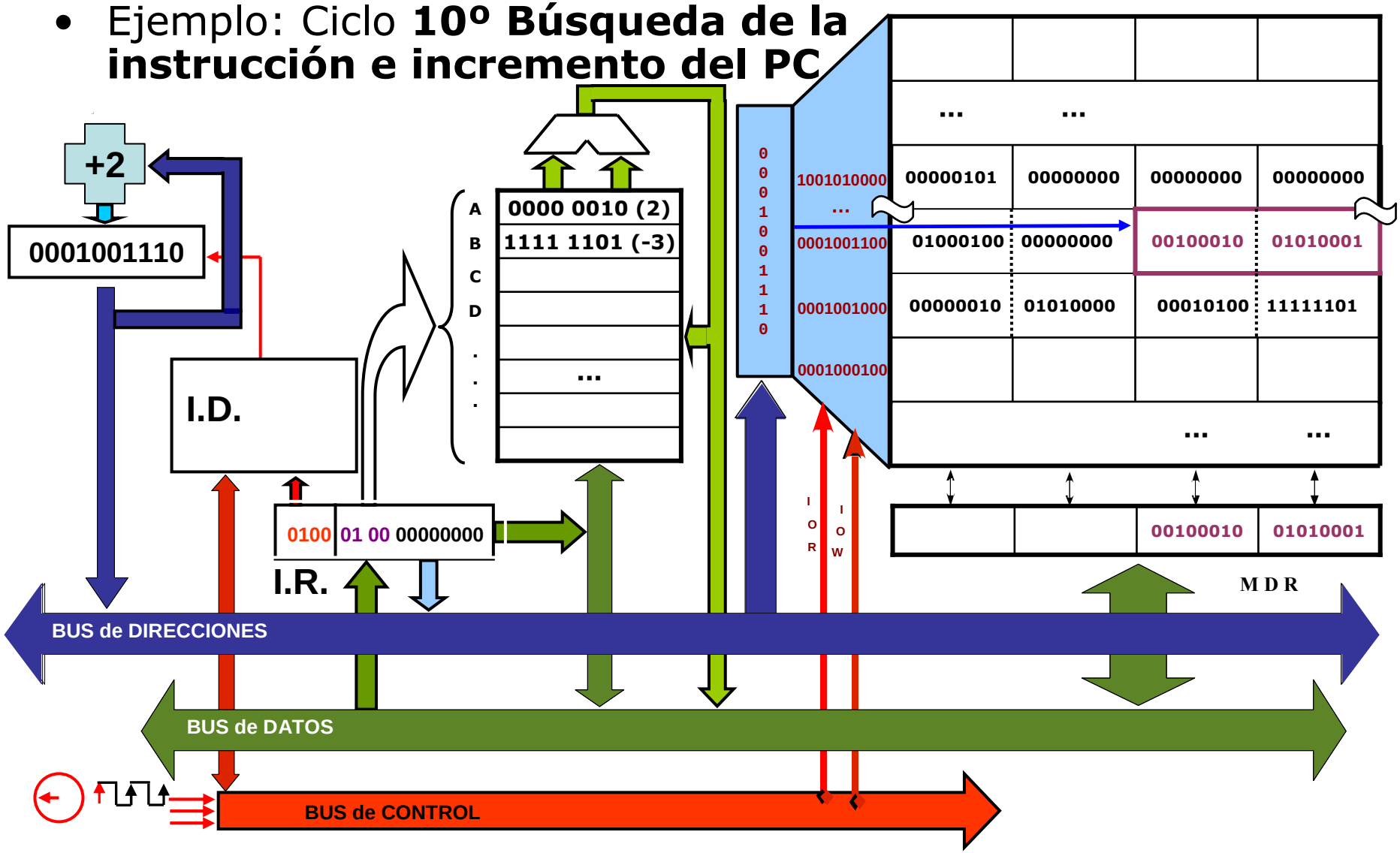
Organización básica del procesador

- Ejemplo: Ciclo **9º Ejecución de la instrucción.**



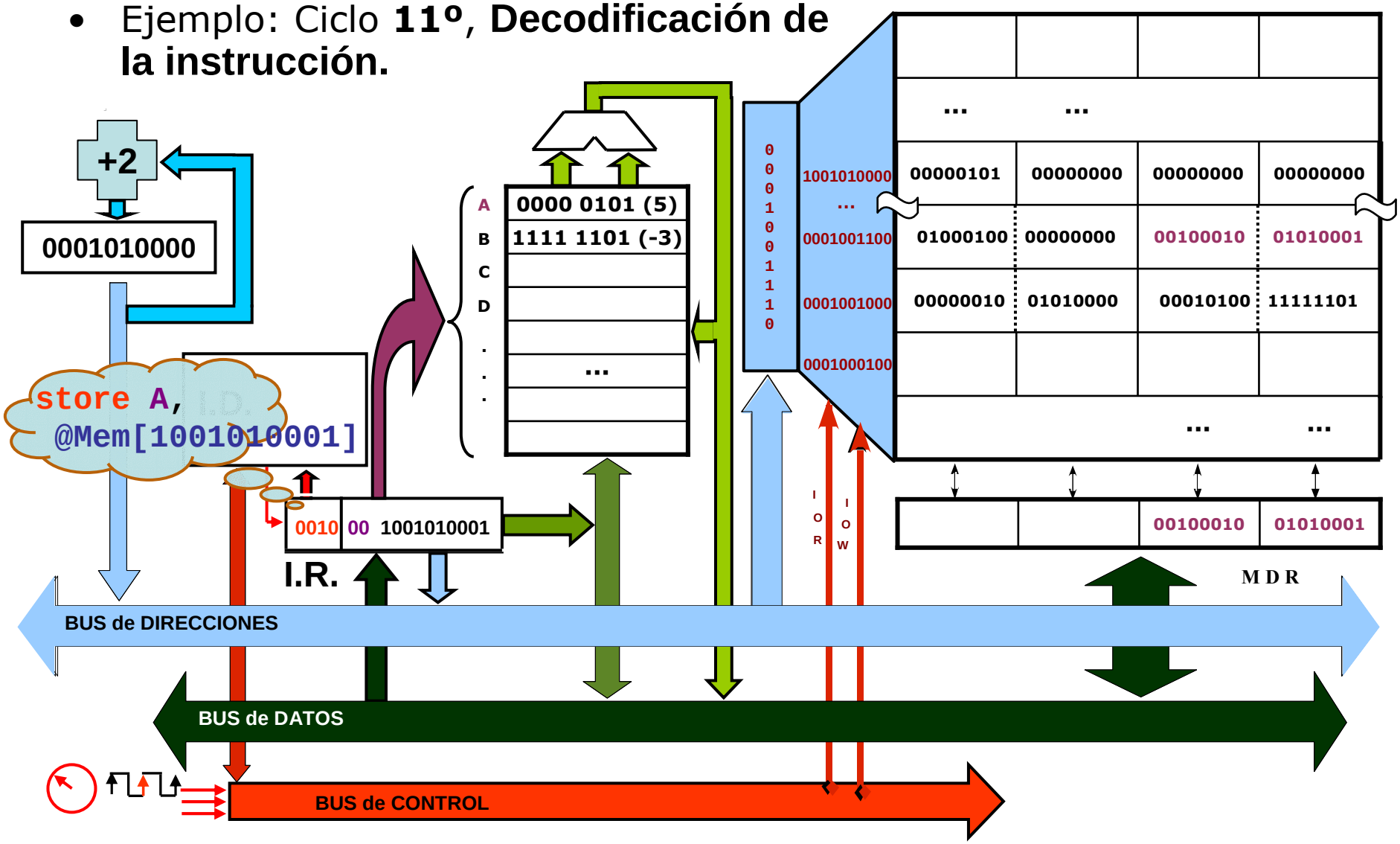
Organización básica del procesador

- Ejemplo: Ciclo **10º Búsqueda de la instrucción e incremento del PC**



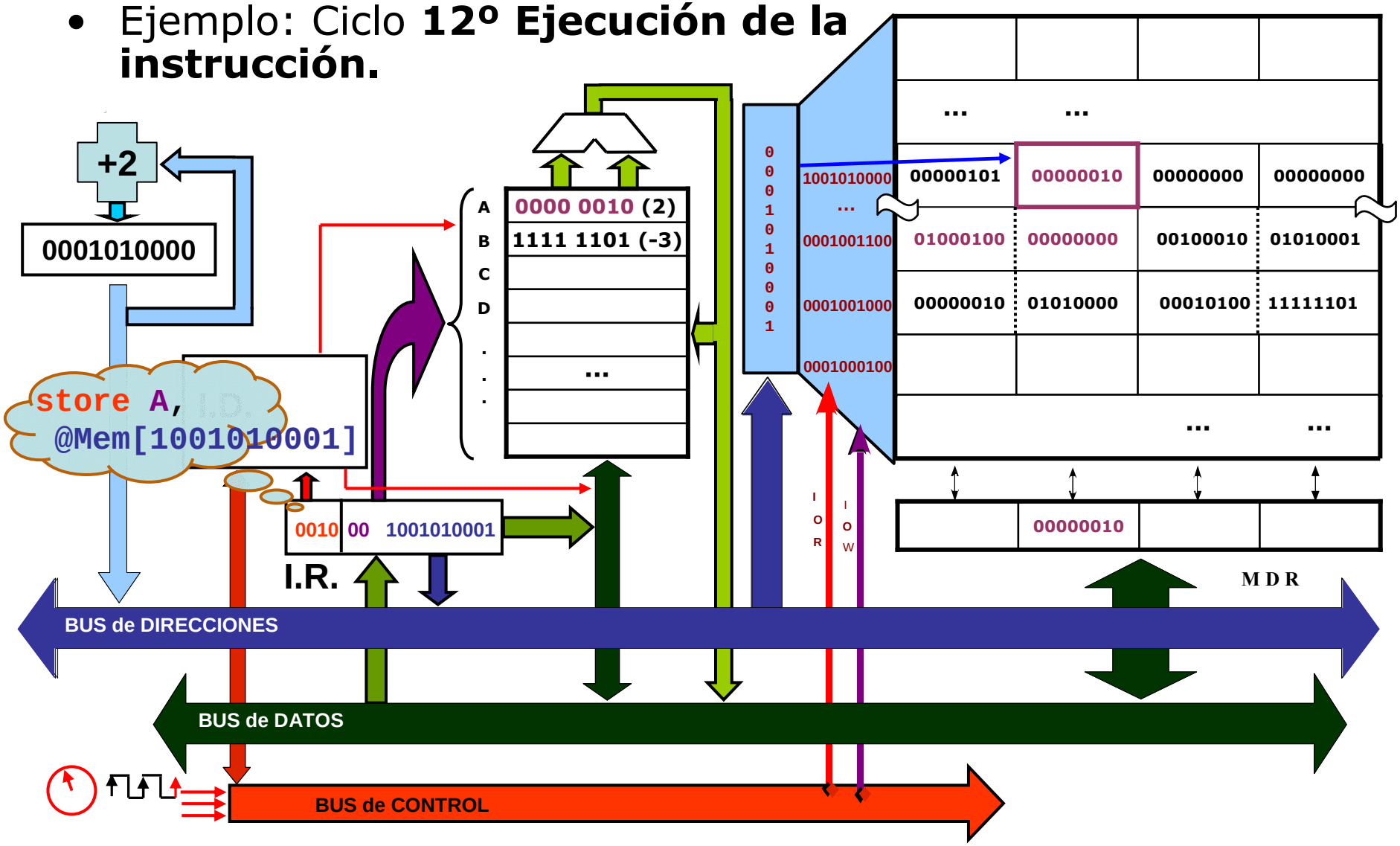
Organización básica del procesador

- Ejemplo: Ciclo **11º**, **Decodificación de la instrucción.**



Organización básica del procesador

- Ejemplo: Ciclo **12º Ejecución de la instrucción.**



Organización básica del procesador

- Recordar:
  - Tanto instrucciones como datos se almacenan como secuencias de 0's y 1's en memoria.
  - Las instrucciones van de memoria a la unidad de control:
    - Las instrucciones se almacenan en memoria en un orden dado por el programa.
    - Las instrucciones se ejecutan en secuencia, sólo rota por las instrucciones de salto.
    - A través del código de operación la unidad de control determina qué hay que hacer.
  - Los datos van de memoria al camino de datos y viceversa.

Parámetros más importantes del procesador

- Velocidad o frecuencia de reloj (GHz)
  - La ejecución de cada instrucción supone la realización de diversos pasos elementales.
    - Por ejemplo: Búsqueda de la instrucción, Decodificación, Ejecución.
  - Cada uno de dichos pasos se ejecuta en uno o varios ciclos de reloj (en nuestro ejemplo, sólo en uno).
  - Ejemplo 1: Para el procesador anterior en el que las instrucciones requieren 3 pasos para su ejecución, ¿cuánto tiempo tardaría en ejecutarse un programa con 10 M instrucciones si el procesador funciona a 2 GHz?

    Sol:

    10×10^6 inst $\Rightarrow$ $3\times10\times10^6 = 30\times10^6$ de ciclos.

    Dado que $T_{ciclo} = 1/(2\times10^9) = 5\times10^{-10}$ seg.

    Tiempo = $30\times10^6 \times 5\times10^{-10} = 15$ mseg.

Parámetros más importantes del procesador

- Velocidad o frecuencia de reloj (GHz)
  - Ejemplo 2: Calcular la frecuencia de reloj de un procesador suponiendo que:
    - El procesador ejecuta 500 MIPS (millón de instrucciones por segundo).
    - Cada instrucción consta de 5 fases.
    - Cada fase se ejecuta en 1 ciclo de reloj.

    Sol:

    $5x10^8$ instr./seg × 5 fases/instr. × 1 ciclo/fase ⇒ 2.5 GHz
  - Ejemplo 3: Dado un procesador con una frecuencia de reloj de 2 GHz, sabiendo que cada instrucción requiere 5 ciclos para su ejecución, ¿cuál es su frecuencia MIPS?

    Sol:

    $2x10^9$ ciclos/seg.

    Cada instrucción tarda 5 ciclos ⇒ $(2/5)x10^9$ instr./seg. ⇒ 400 MIPS.

Parámetros más importantes del procesador

- Tecnología de fabricación:
  - Lo avanzado de una tecnología de fabricación se indica mediante una medida relacionada tamaño del elemento más pequeño del chip: el transistor.
  - Actualmente se mide en decenas de nanómetros (*nm*, la millonésima parte de un milímetro (10^{-6} *mm*)).
    - El primer microprocesador de la era PC (Intel 8088) se fabricó con tecnología de 3 micras (incluía 29.000 transistores).
    - Los procesadores de Intel y AMD actuales se fabrican con tecnología de 45 nm (o 0,045 micras) e incluyen más de 750 M de transistores.
  - Mejoras en la tecnología de fabricación:
    - Permiten aumentar el número de chips por oblea, y, por tanto, disminuir el coste del microprocesador.
    - Se posibilita alcanzar mayores frecuencias de reloj (GHz).
    - Se puede reducir el voltaje necesario para el funcionamiento y por tanto la cantidad de calor que se genera (aunque es más difícil de eliminar).
    - Se pueden incorporar nuevos elementos (memorias cachè, …).

Índice

Organización del subsistema de memoria

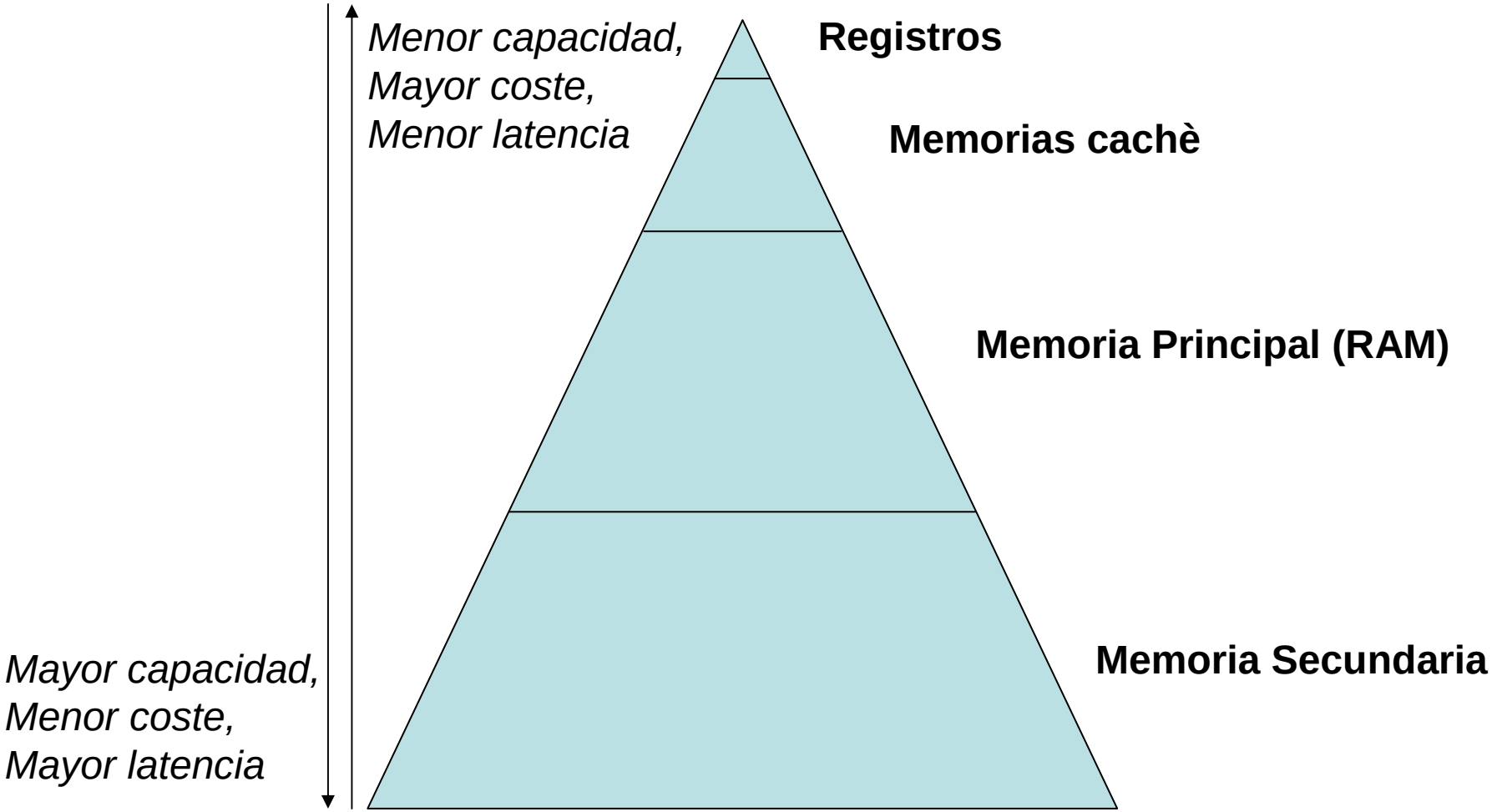
- Hasta ahora hemos considerado la memoria del computador como una gran estructura lógica donde se almacenan tanto instrucciones como datos.
- Sin embargo, la memoria de un ordenador moderno está conformada por varios tipos de estructuras de memoria, cada uno con características diferentes en cuanto a costo, tiempo de acceso y tamaño.
- El objetivo final es ofrecer al procesador una memoria lógica con un tiempo de acceso muy bajo y un tamaño muy grande, todo dentro de un presupuesto razonable.
- Analogía: Mesa en la biblioteca.
  - Una buena selección de libros en la mesa (memoria pequeña y rápida).
  - Gran probabilidad de encontrar lo buscado sin ir a la estantería.
  - Visión de memoria grande (biblioteca) a la que se accede rápido (tiempo de coger un libro de la mesa).

Concepto de jerarquía de memoria

- Objetivo: Llevar zonas del programa y datos que tendrán más probabilidad de ser accedidas a una memoria más rápida.
- Dos propiedades de los programas hacen viable la organización de la memoria del ordenador en una jerarquía de varios niveles:
  - Localidad Temporal: cuando se consulta un dato, seguramente será consultado poco después.
    - Un libro de la mesa será consultado varias veces.
    - Ejemplo en programas: bucles (datos e instrucciones).
  - Localidad Espacial: cuando se consulta un dato, seguramente otros cercanos serán consultados poco después.
    - Traemos un libro del estante → los libros que están próximos versarán sobre el mismo tema.
    - Ejemplo en programas: instrucciones (salvo por los saltos, se ejecutan secuencialmente tal como están en memoria) y estructuras de datos como las tablas.

Concepto de jerarquía de memoria

- Jerarquía de memoria:



Concepto de jerarquía de memoria

- El objetivo de organizar la memoria de un ordenador como una jerarquía de memorias es aprovechar la localidad temporal y espacial:
  - Para aprovechar la Localidad Temporal de un programa:
    - Mantener los datos accedidos más recientemente cerca del procesador.
  - Para aprovechar la Localidad Espacial de un programa:
    - Mover bloques de varios datos contiguos a los niveles próximos al procesador.
- Consideración: Las memorias más rápidas son las más caras por bit y por tanto suelen ser de menor capacidad.

¿Qué es una memoria cachè?

- Definición de cachè: pequeña memoria ultrarápida que se coloca entre la memoria principal (RAM) y el procesador con el objetivo de acelerar los accesos a datos e instrucciones.
- En la actualidad se intercalan varias memorias cachè (niveles de cachè) entre el procesador y la memoria principal, cada nivel tiene un tiempo de acceso y tamaño distinto.
  - En general, los niveles más cercanos al procesador son los más rápidos y pequeños.
  - Las memorias cachè se suelen denominar por el nivel en el que se encuentran, siendo L1 el más cercano al procesador.
- Ejemplo: los procesadores actuales incluyen hasta 3 niveles de cachè dentro del chip:

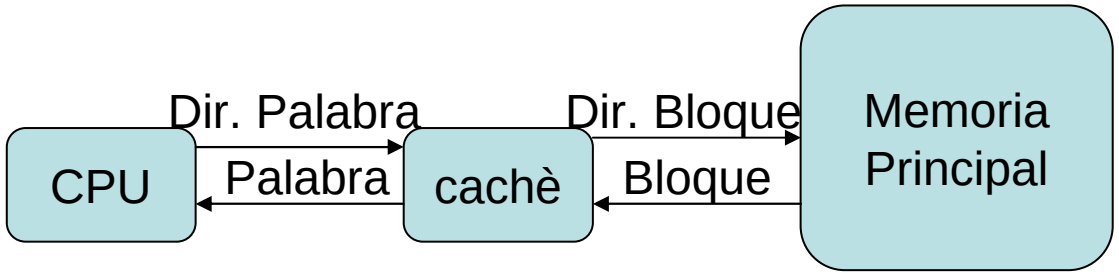
  cachè L1: 32KB-64KB (suele haber una para datos y otra para instrucciones), 2-4 ciclos de procesador de latencia.

  cachè L2: 256KB-512KB, 10-15 ciclos de reloj de latencia.

  cachè L3: 6MB-8MB, 40-50 ciclos de reloj de latencia.

¿Qué es una memoria cachè?

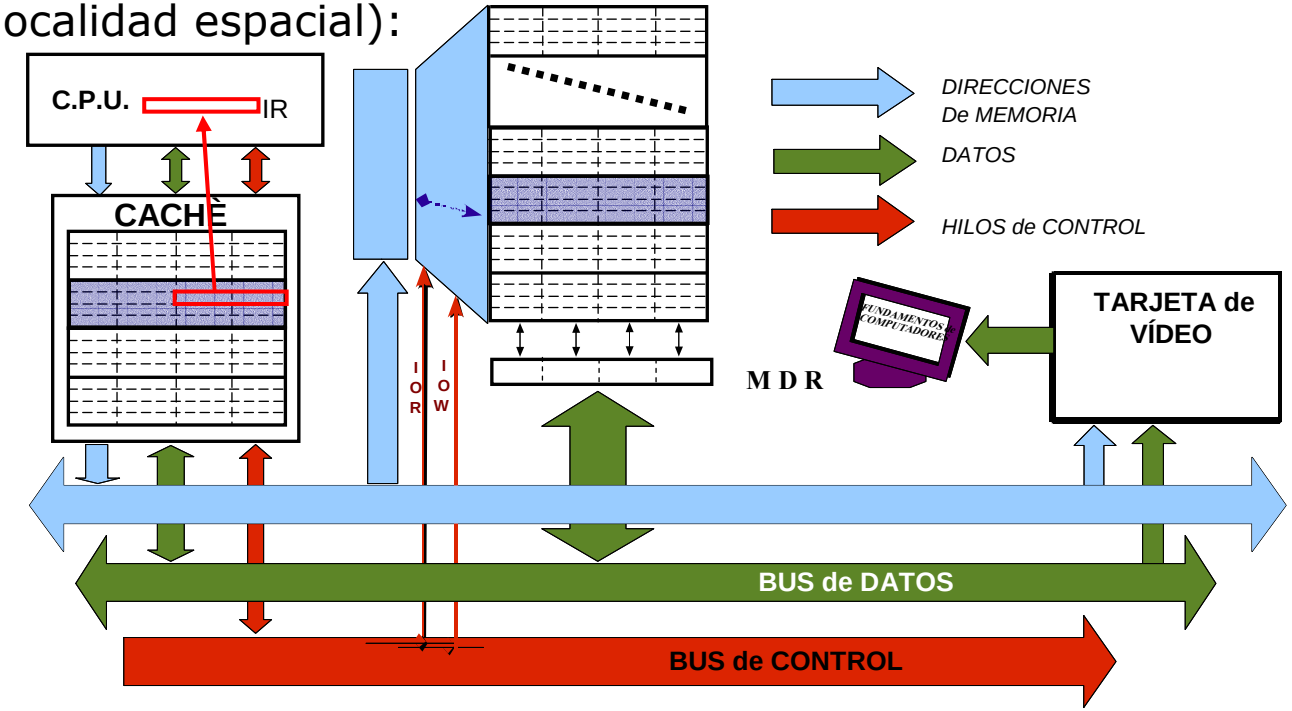
- Funcionamiento Básico:



  - El procesador manda al controlador de cachè la dirección de la palabra a buscar (por ejemplo, para un load).
  - Caso de estar, el controlador de cachè suministra al procesador la palabra solicitada.

¿Qué es una memoria cachè?

- Funcionamiento Básico:
  - Caso de no estar, el controlador de cachè pide a memoria un bloque de varias palabras que entre otras incluye la solicitada por el procesador (localidad espacial):



  - El controlador de cachè almacena el bloque y suministra al procesador la palabra solicitada.

¿Qué es una memoria cachè?

- Puesto que una memoria cachè puede contener un subconjunto muy reducido de los datos e instrucciones almacenados en memoria principal
  - ¿Cómo se sabe si un dato está o no en cachè?
  - Y si lo está, ¿cómo se localiza?

- Ejemplo 1: Dados los siguientes parámetros:
  - Tamaño de las direcciones de memoria: **10 bits**.
  - **Cada dirección** se refiere a **un byte**.

  ¿Cuál es el tamaño máximo que podría tener la memoria principal en Bytes?

Sol:

Puesto que se dispone de 10 bits para codificar las direcciones, el número máximo de direcciones (lo que limita el máximo de posiciones direccionables) es $2^{10} = 1024 = 1K$.

¿Qué es una memoria cachè?

- Ejemplo 2: Supongamos un ordenador con la memoria principal del ejemplo anterior al que añadimos un nivel de cachè para datos e instrucciones de **64Bytes.** Supongamos también que los **bloques** que se transfieren entre memoria cachè y memoria principal son **de 4 palabras de 4 bytes (16 bytes).** ¿Cuántos bloques caben en la cachè? ¿Cuántos bloques habrá en memoria principal?

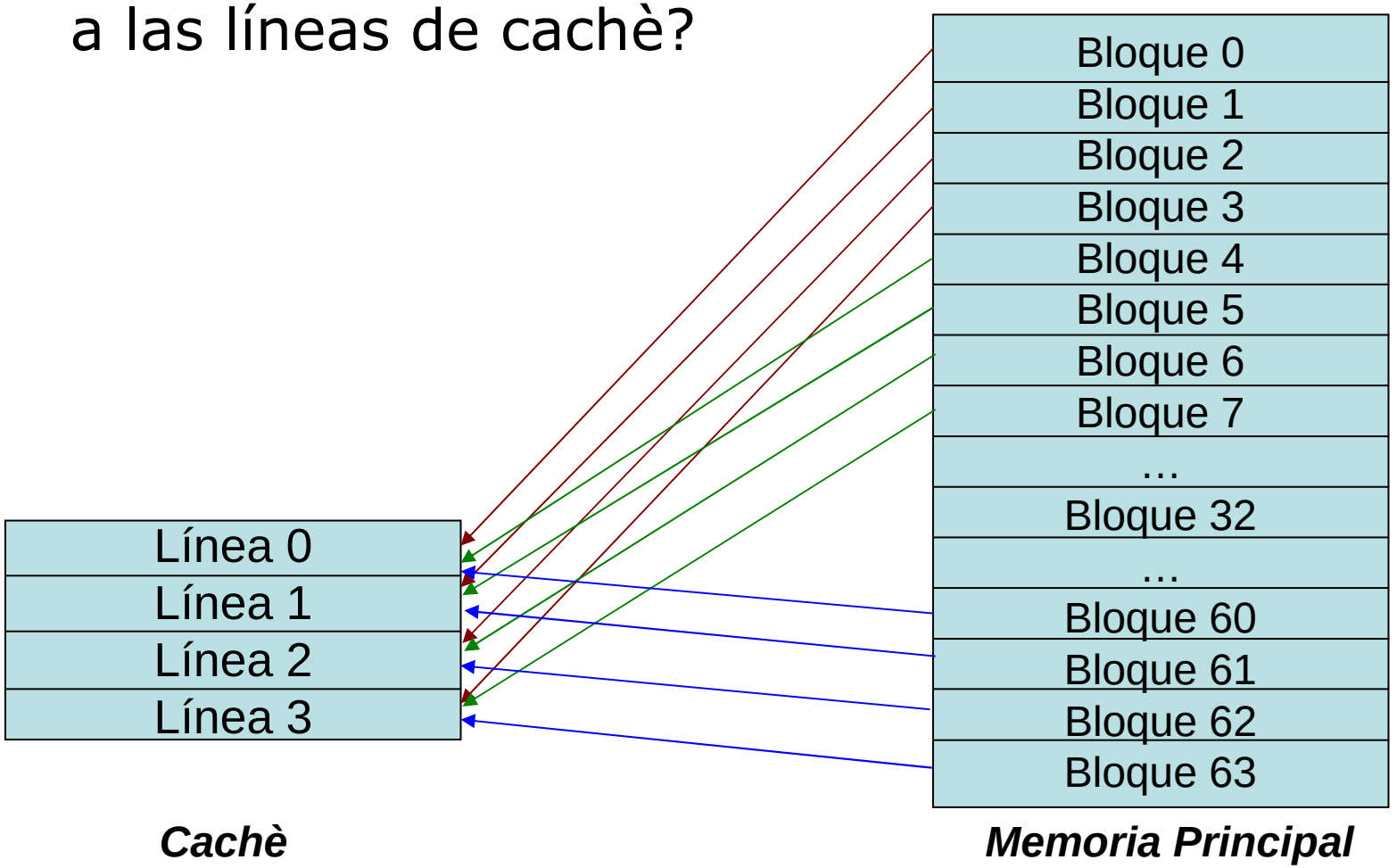
  Sol:

  Puesto que en memoria principal hemos visto que había 1k bytes, y cada bloque está formado por 4 palabras de 32bits, el número de bloques que hay en ***memoria principal*** es 1024/(4 * 4) = ***64 bloques***.

  Por su parte, puesto que la ***cachè*** tiene un tamaño de 64B, el número máximo de palabras de 32bits que podrá almacenar es de 64/4 = 16. Y el número máximo de bloques es 64/16 = 4. A cada uno de estos **4** huecos en los que se va a poder almacenar un bloque en cachè lo vamos a llamar ***línea de cachè***.

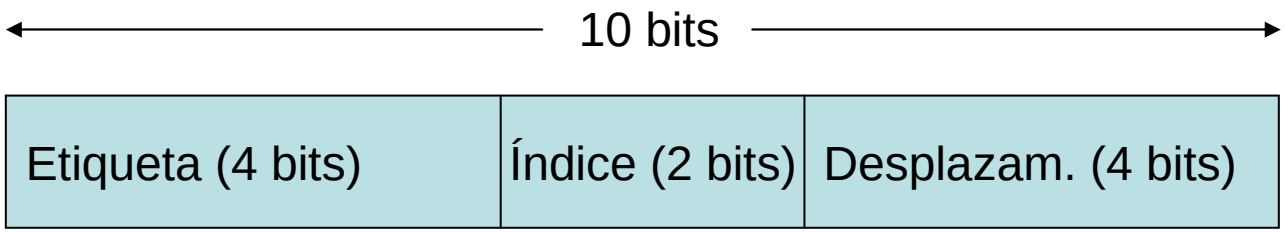
¿Qué es una memoria cachè?

- Para la configuración de los ejercicios anteriores, ¿cómo podríamos asignar los bloques de memoria a las líneas de cachè?



¿Qué es una memoria cachè?

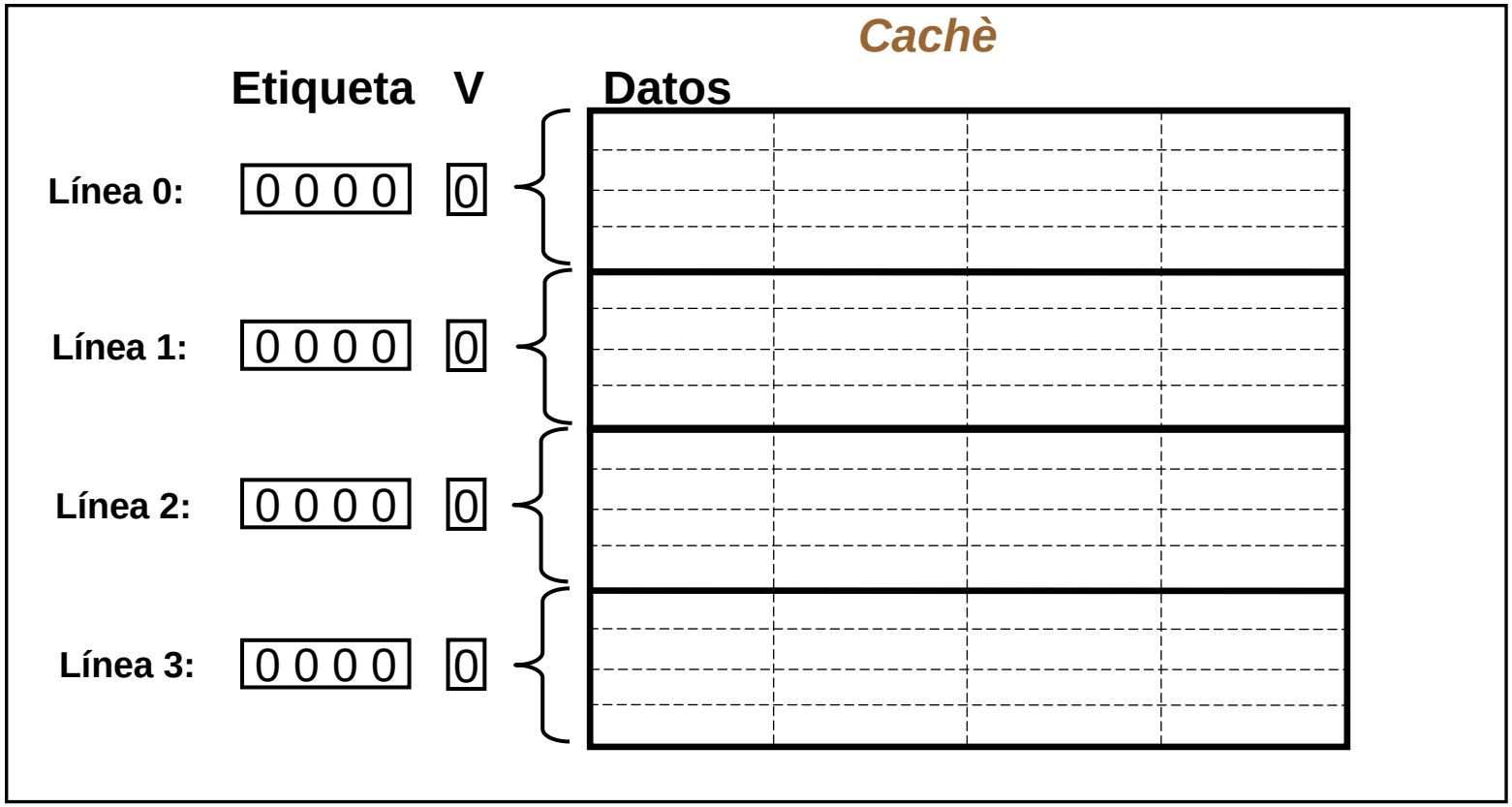
- O lo que es lo mismo, los bits menos significativos de la *dirección de bloque* nos indican la línea de cachè donde debe ir cada bloque.
- Puesto que varios bloques pueden ir a la misma línea de cachè, ¿cómo sabemos si lo que hay en cachè es el bloque que buscamos?
  - Además del bloque de datos, cada línea de cachè almacena una etiqueta (los bits más significativos de la dirección del bloque) y un bit de validez (indica si hay datos válidos en la línea).
- Ejemplo 3: Para la configuración del sistema de memoria del ejercicio anterior, ¿cómo se distribuyen los 10 bits de la dirección de memoria desde el punto de vista de la cachè?



¿Qué es una memoria cachè?

- Para el ejemplo del apartado 4.2.1, ¿cuáles de los accesos a memoria encontrarían la información en cachè?

Inicialmente:

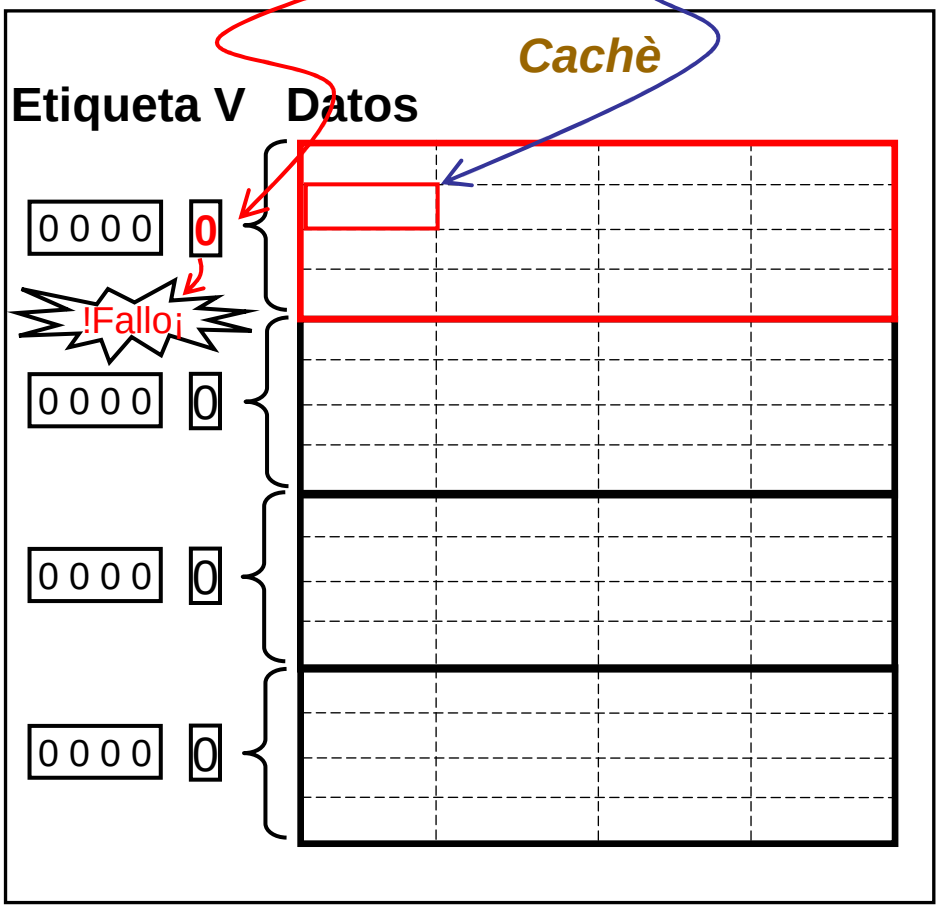


¿Qué es una memoria cachè?

Ciclo 1: búsqueda de instr. en

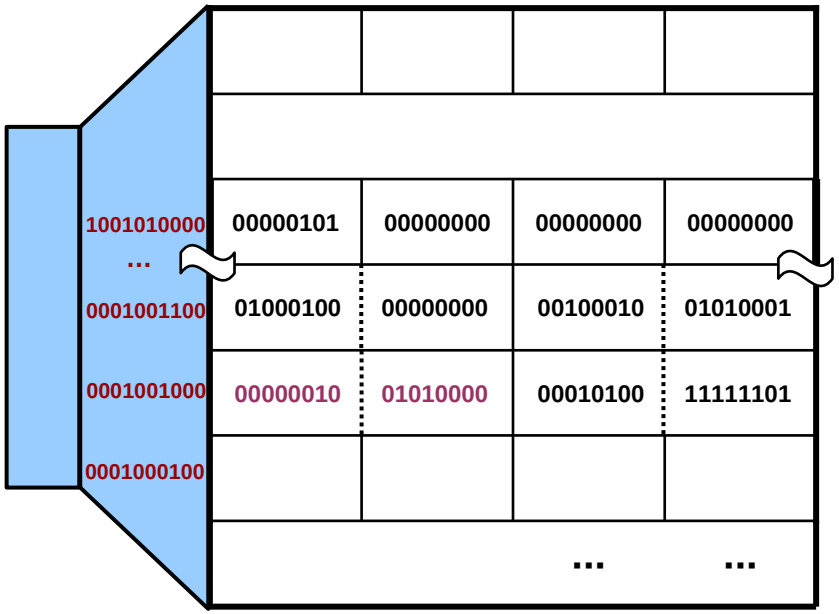
0001001000 → 0001 00 1000

Etiqueta Índice Despl.



Ciclo 1: Tras el fallo de cachè, la cachè deberá traer 16 Bytes en bloque, en 4 accesos consecutivos a memoria principal, para rellenar de datos toda la línea de cachè 0 (suponemos bus de datos de 32 bits = 4 bytes por acceso)

MEMORIA Principal

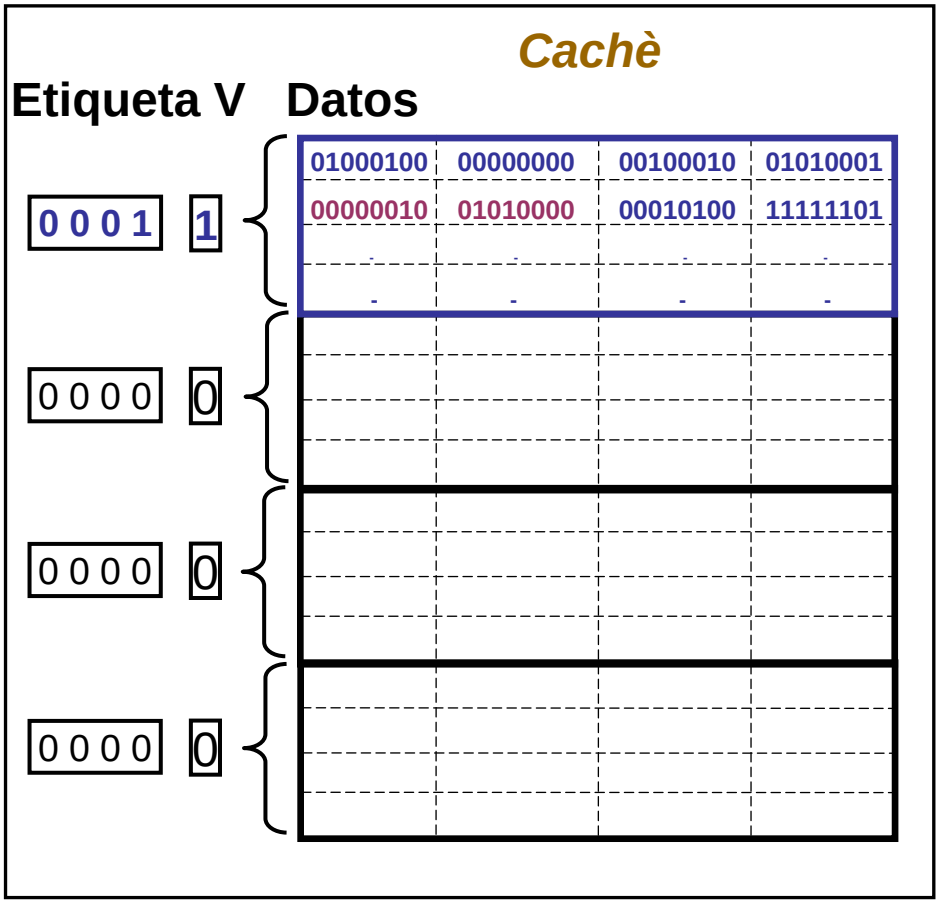


¿Qué es una memoria cachè?

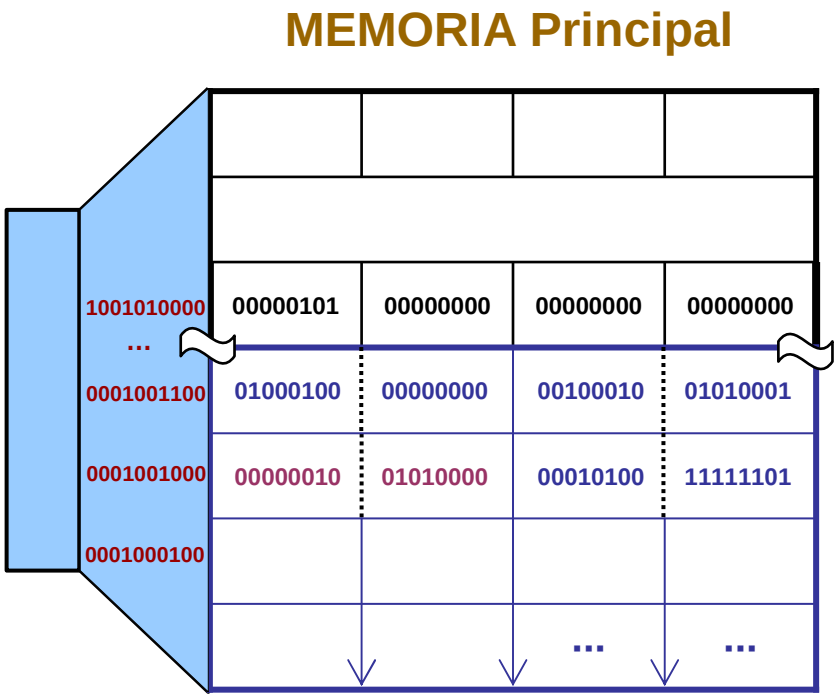
Ciclo 1: búsqueda de instr. en

0001001000 → 0001 00 1000

Etiqueta Índice Despl.



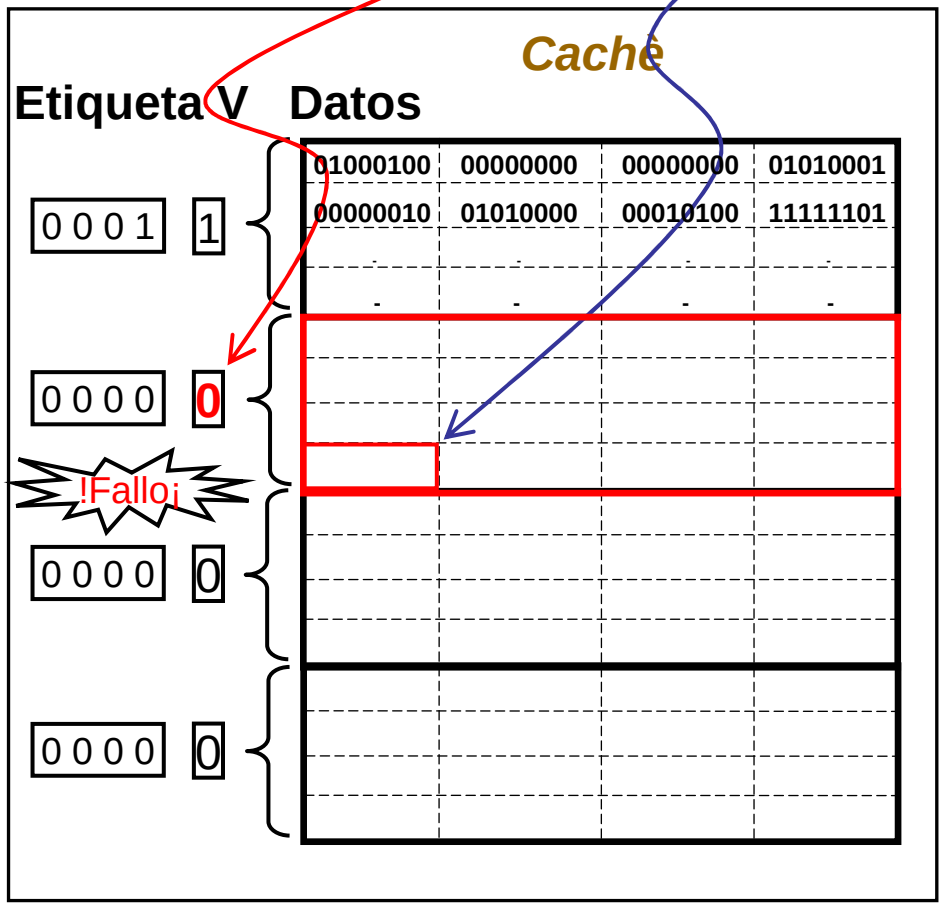
Ciclo 1: Se activa el bit de validez y se colocan los 4 bits más significativos como etiqueta.
La CPU toma los 16 bits de la instrucción **load A, @Mem[...]** (dirs. **0001001000 y 0001001001**).



¿Qué es una memoria cachè?

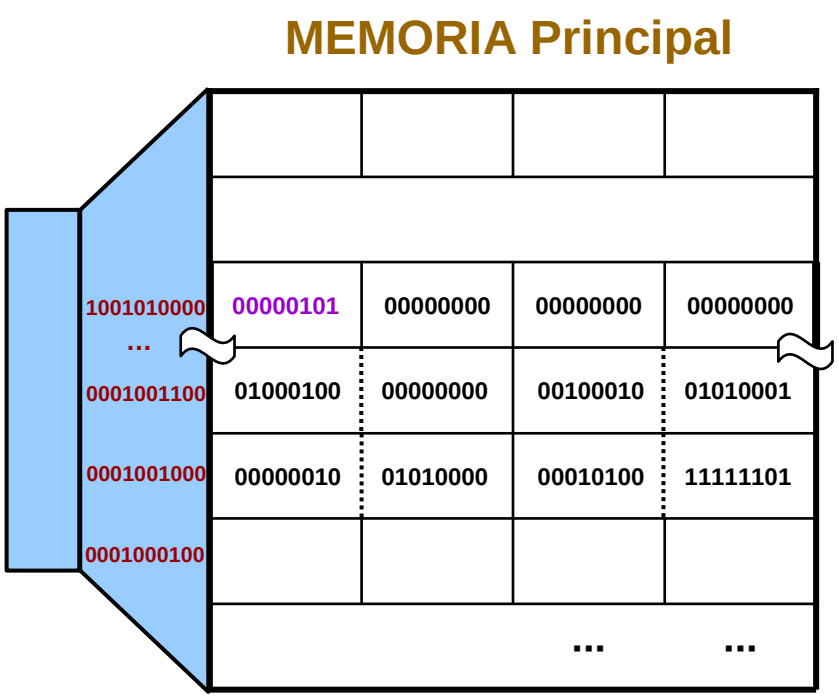
Ciclo 3: Ejecución de instr.

1001010000 → 1001 (Etiqueta) 01 (Índice) 0000 (Despl.)



Ciclo 3: Instrucción

load A, @Mem[1001010000]. El bit de validez provoca un fallo de cachè. La cachè deberá traer el correspondiente bloque para rellenar toda la línea de cachè 1.

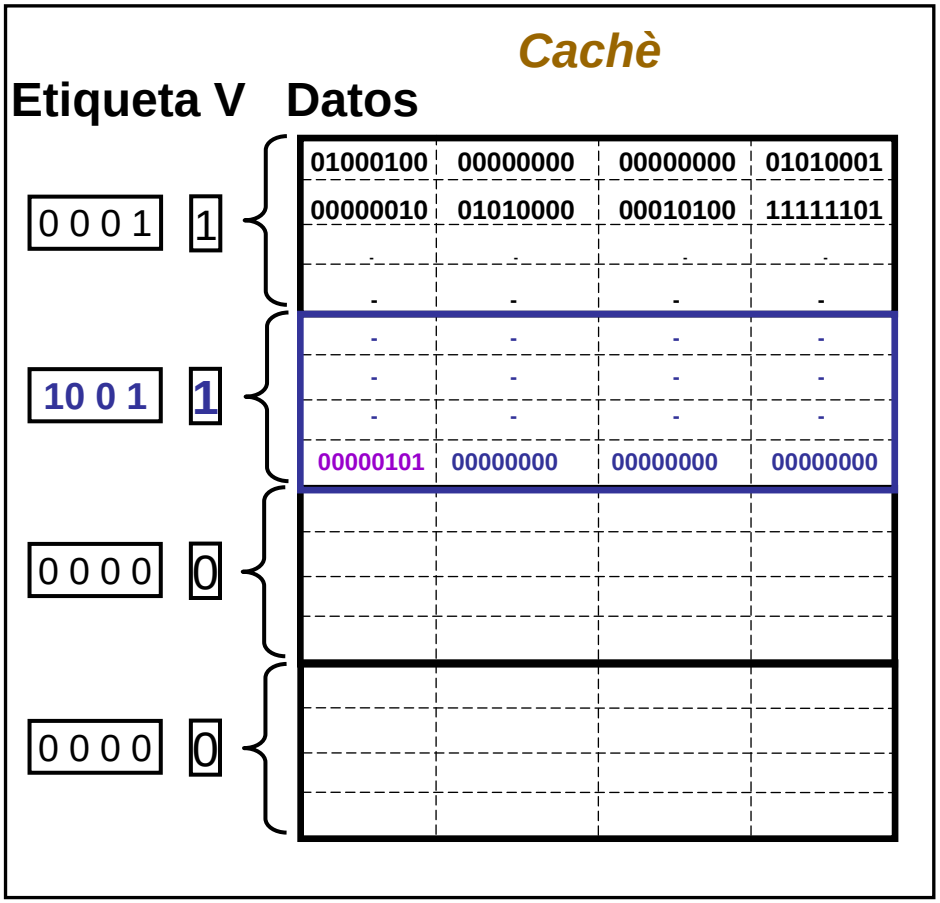


¿Qué es una memoria cachè?

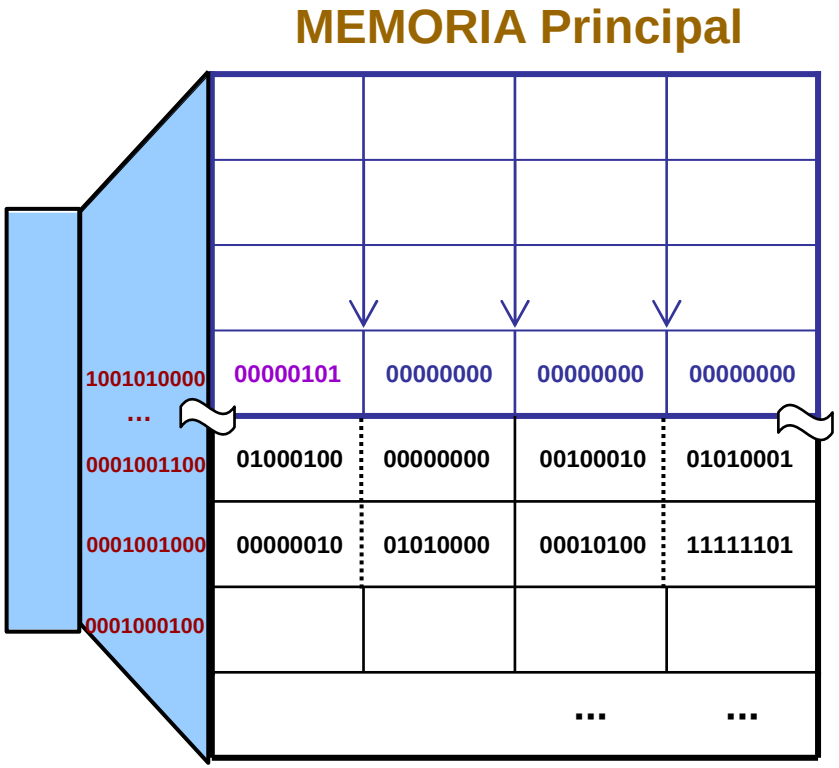
Ciclo 3: búsqueda de datos en

1001010000 → 1001 01 0000

Etiqueta Índice Despl.



Ciclo 3: Se activa el bit de validez y se colocan los 4 bits más significativos como etiqueta.
La CPU toma los 8 bits del dato en **1001010000** (instrucción **load A, @mem[1001010000]**).

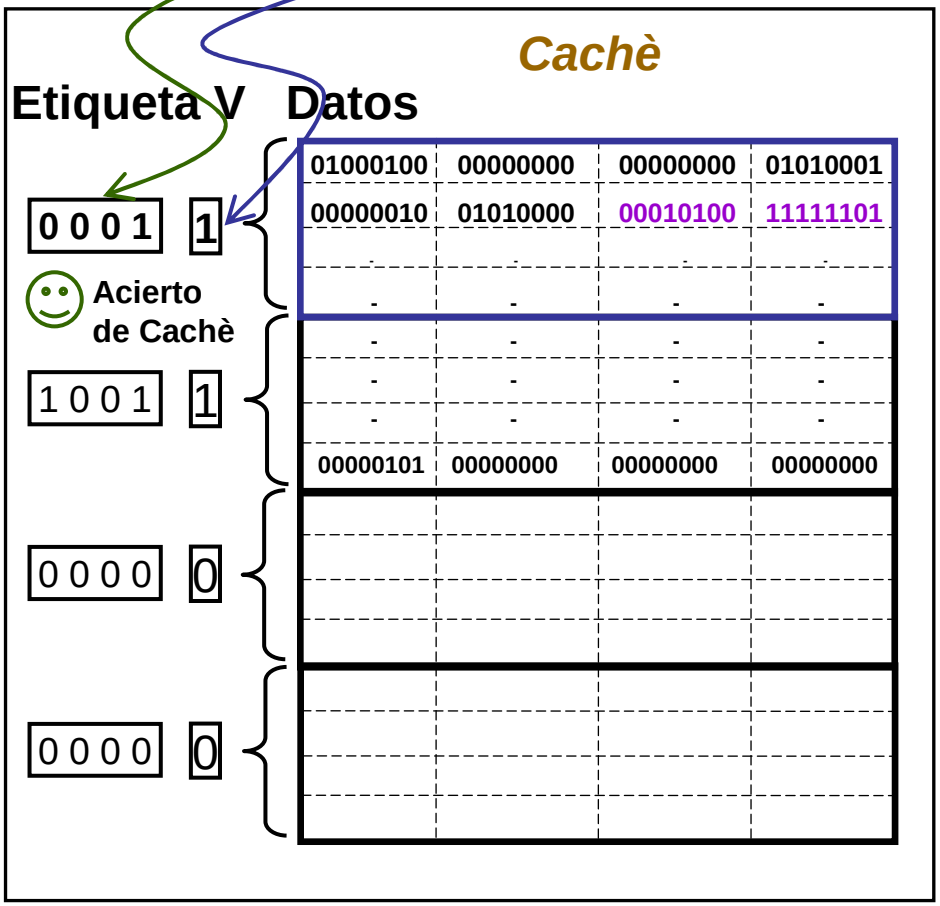


¿Qué es una memoria cachè?

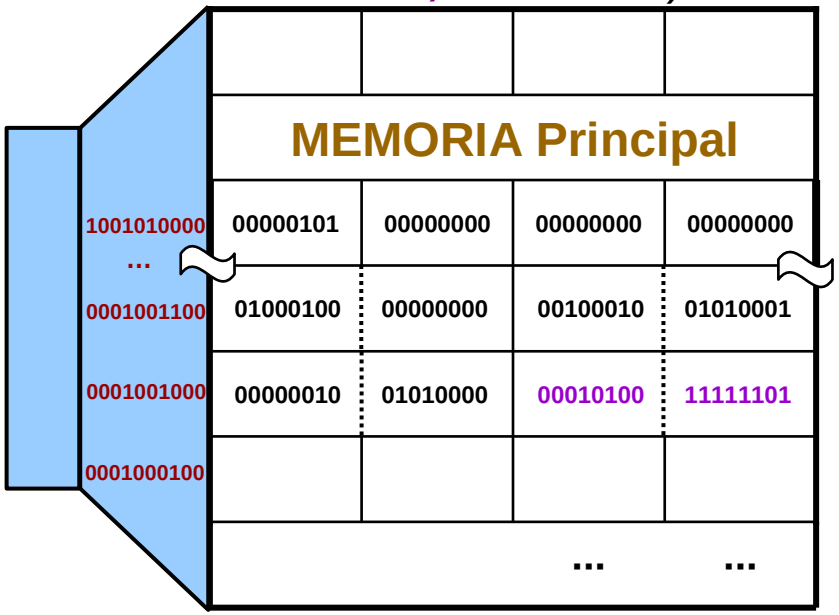
Ciclo 4: búsqueda de instr. en

0001001010 → 0001 00 1010

Etiqueta Índice Despl.



Ciclo 4: El campo índice busca en la línea de cachè 0 si el bit de validez = 1; Como es así compara los valores del campo etiqueta con los suyos propios, y como es así, los datos en la línea de cachè son los que se corresponden. (Dirs **0001001010** y **0001001011** instrucción **load B, 11111101**).

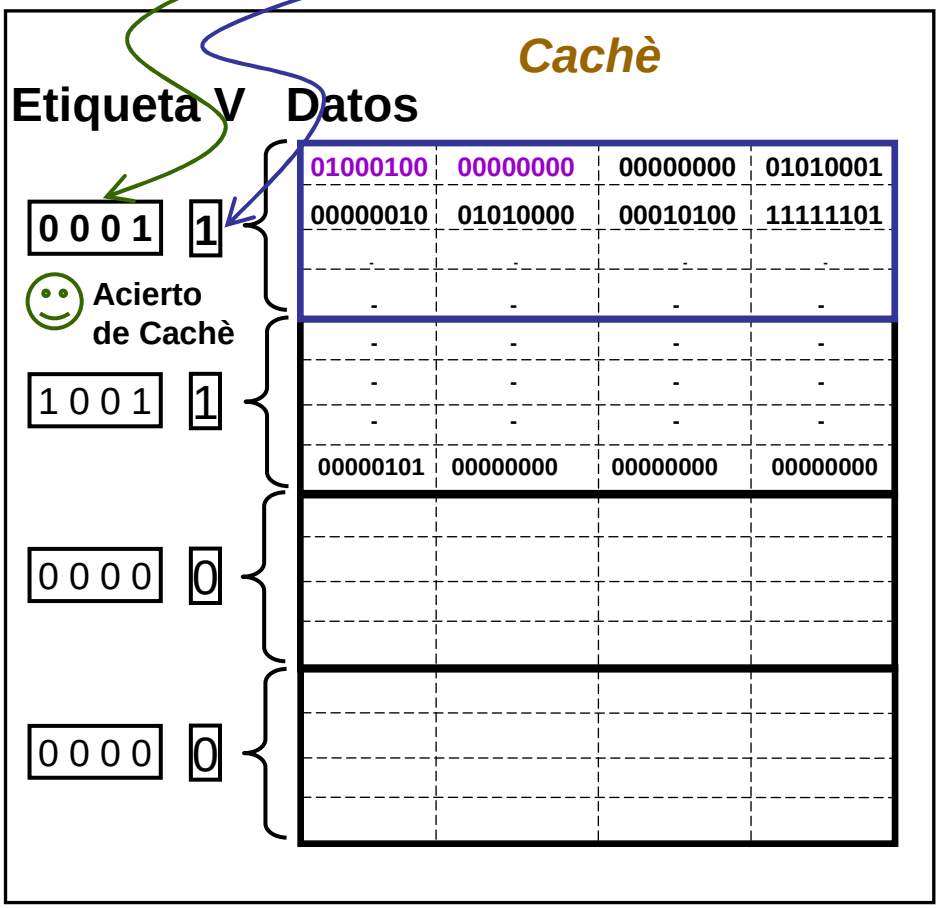


¿Qué es una memoria cachè?

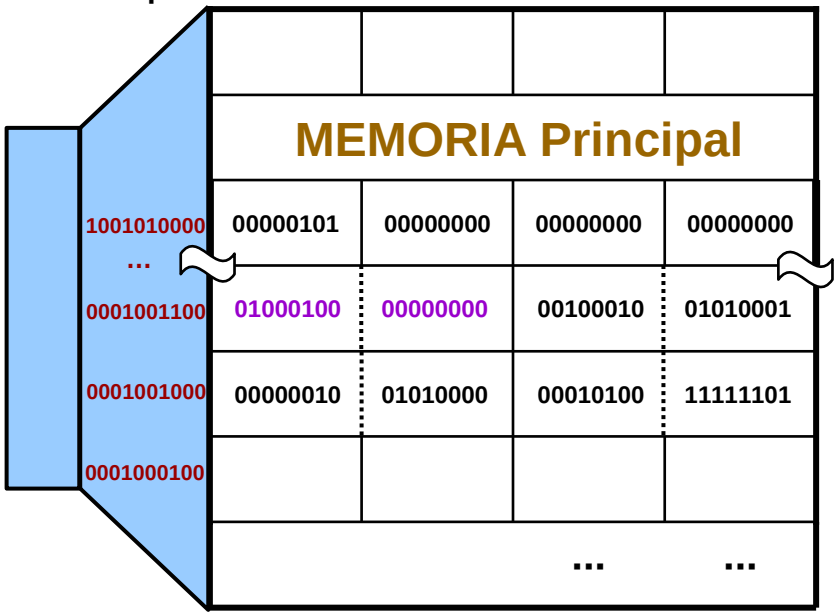
Ciclo 7: búsqueda de instr. en

0001001100 → 0001 00 1100

Etiqueta Índice Despl.



Ciclo 7: (Instrucción **add B,A**). Bit de validez = 1 en la línea de cachè 0, el campo etiqueta coincide con los cuatro bits más significativos de la dirección → los datos en la línea 0 de cachè son válidos. Si el campo etiqueta no coincidiera, habría que hacer un reemplazo de la línea de cachè 0.

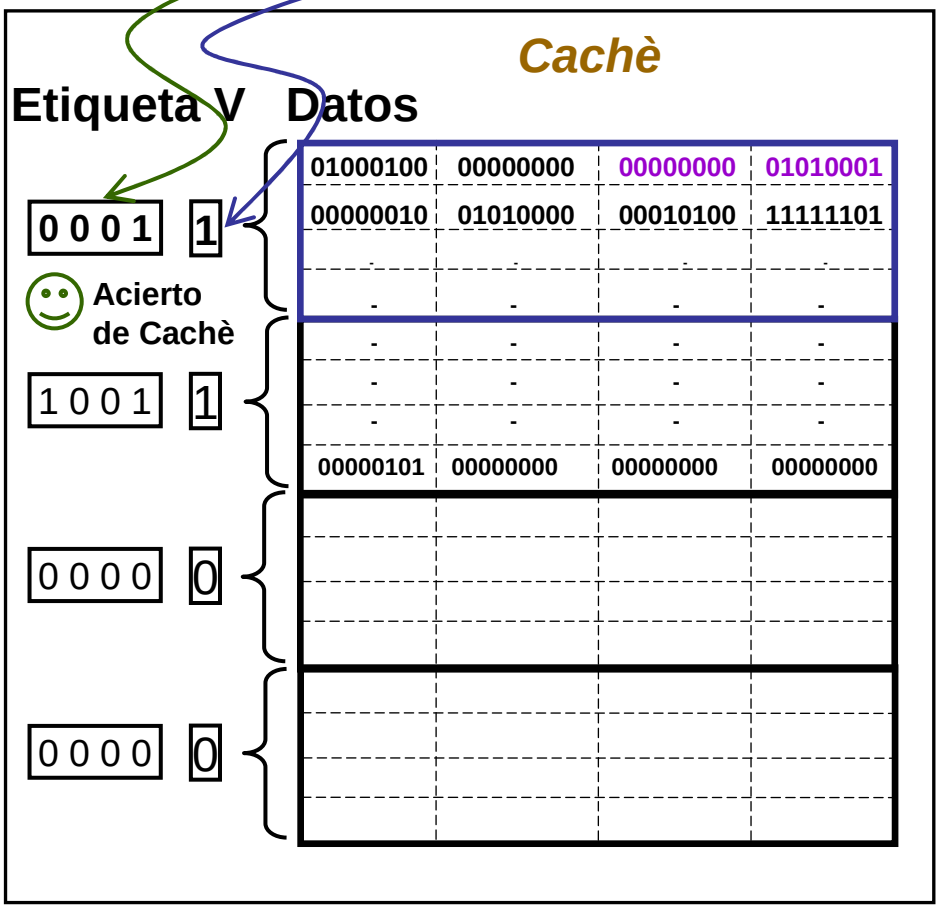


¿Qué es una memoria cachè?

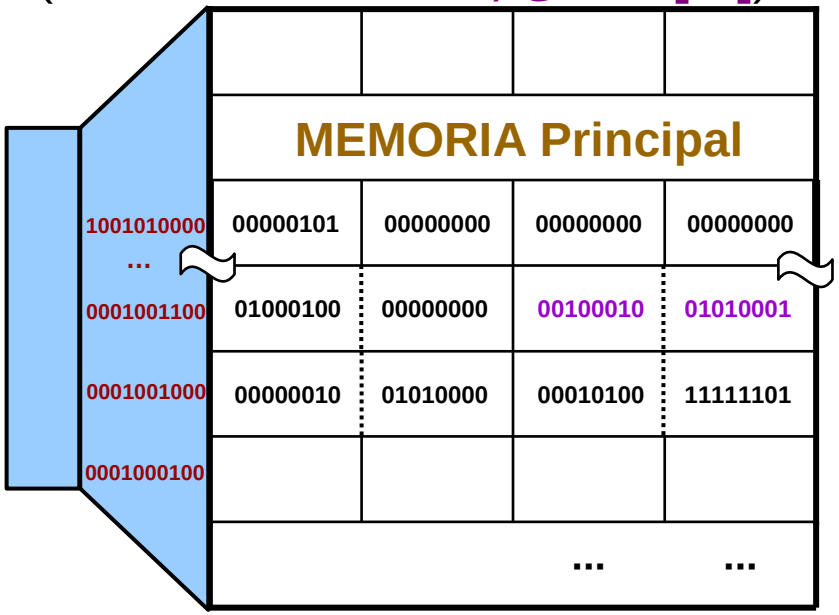
Ciclo 10: búsqueda de instr. en

0001001110 → 0001 00 1110

Etiqueta Índice Despl.



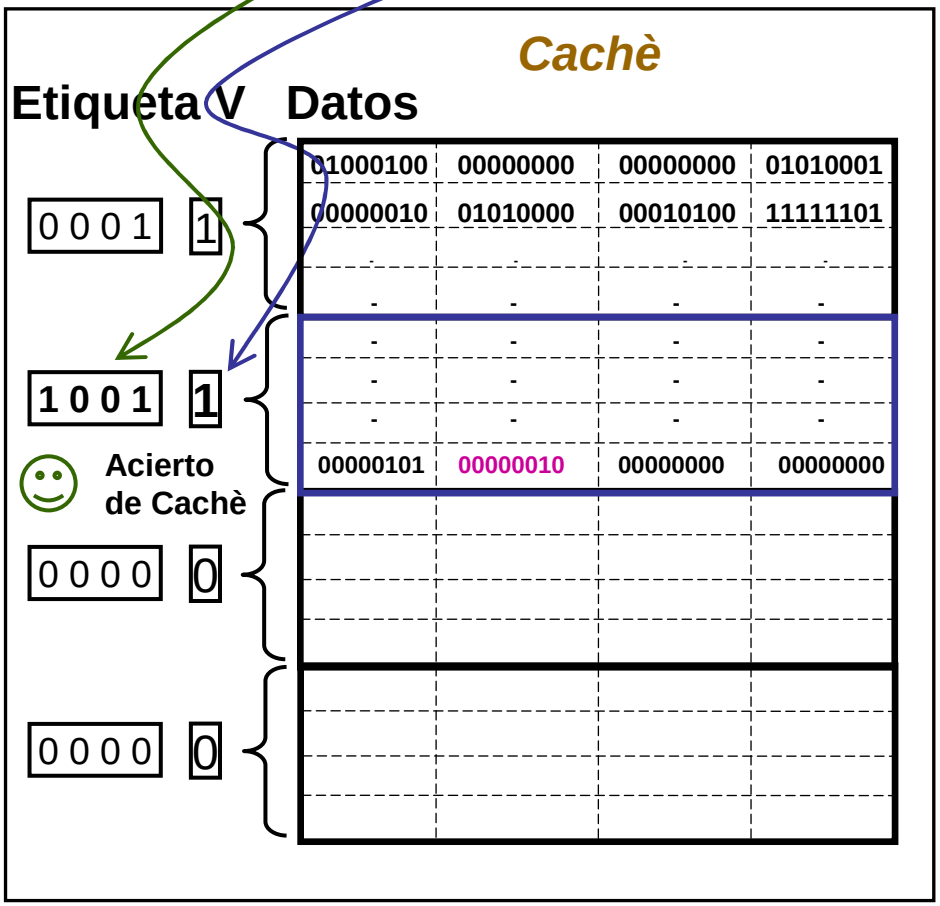
Ciclo 10: Bit de validez = 1 en la línea de cachè 0, el campo etiqueta coincide con los cuatro bits más significativos de la dirección de memoria → los datos en la línea de cachè 0 son válidos. La CPU lee el byte con dirs. = **0001001110** y **0001001111** de la cachè (instrucción **store AL, @Mem[…]**).



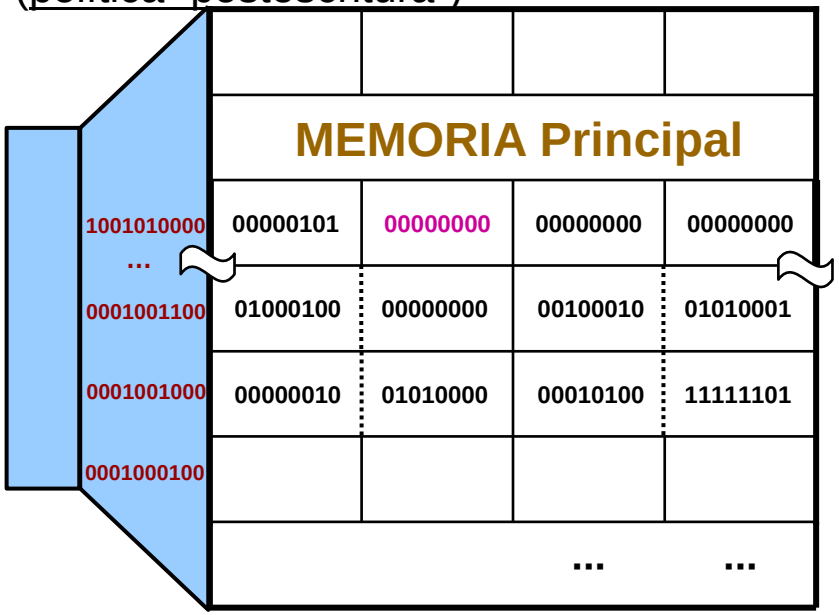
¿Qué es una memoria cachè?

Ciclo 12: ejecución de instr.

1001010001 → 1001 (Etiqueta) 01 (Índice) 0001 (Despl.)



Ciclo 12: Instrucción **store AL, @Mem[1001010001]** (escribir en dirección **1001010001**). La CPU escribe el byte **0000000010** en cachè. Se marcaría con un bit de "modificado" la línea de cachè 1 para que cuando sea reemplazada no se "machaque" con la nueva entrada sino que previamente sea reescrito el bloque en memoria principal (política "postescritura")



La memoria principal y sus parámetros fundamentales

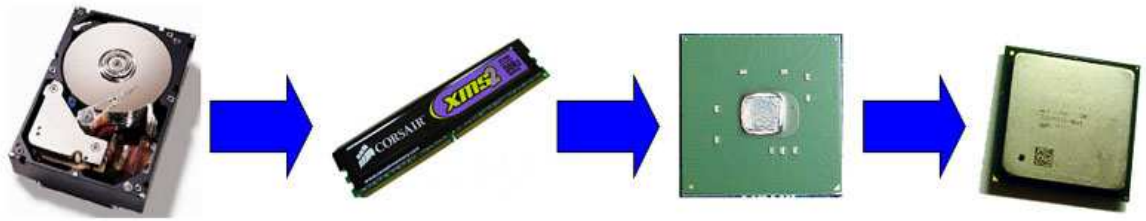
- Bus de memoria
  - Conecta el controlador de memoria (antes fuera de la CPU, ahora ya habitualmente dentro) y los chips de memoria.
  - Se divide en:
    - Bus de direcciones: parte encargada de enviar las direcciones de memoria (en algunos casos también las direcciones de puertos). Dependiendo de su ancho (en bits) se podrá direccionar una determinada cantidad de memoria.
    - Bus de datos: parte encargada de transmitir los datos. Cuanto más ancho sea, más datos pueden enviarse en cada ciclo de reloj.
    - Bus de control: Hilos donde fluye la información de control para gobernar la lectura/escritura en memoria (IOW, IOR…), señales de reloj (Clk), señales de sincronización, alimentación (Vcc), tierra (ground)…

La memoria principal y sus parámetros fundamentales

- Ancho de banda (MB/s)
  - Máxima cantidad de memoria que teóricamente podría obtenerse por segundo (se mide en MB/s).
  - P.e., la memoria DDR-200 (Double Data Rate) opera con un bus físico a 100 MHz (frecuencia base), pero tiene doble aprovechamiento de cada ciclo de reloj (de ahí el apellido "200"). Puesto que, además, su ancho del bus de datos es de 64 bits (8 bytes), es capaz de transmitir con un ancho de banda máximo de $100*10^6*2*8 = 1600$ MB/s. Es por esto que a dichos módulos DDR-200 también se les conoce como PC-1600.
  - Hay también módulos DDR2, que trabajan al cuádruple de la frecuencia base, e incluso DDR3, que trabajan a ocho veces la frecuencia base. Así, algunos otros ejemplos de módulos de memoria serían:
  - DDR (doble de frecuencia base):
    - P.e., con 150 MHz de frec. base -> DDR-300 -> PC-2400 = 2400 MB/s
  - DDR2 (cuádruple de frecuencia base):
    - P.e., con 200 MHz de frec. base -> DDR2-800 -> PC2-6400 = 6400 MB/s
  - DDR3 (ocho veces la frecuencia base):
    - P.e., con 200 MHz de frec. base -> DDR3-1600 -> PC3-12800 = 12800 MB/s
  - Más ejemplos en `http://es.wikipedia.org/wiki/Double_Data_Rate`

Memoria secundaria

- Tanto memorias cachè como memoria principal son estructuras de memoria volátiles → sus contenidos se pierden al apagar el ordenador.
- Programas y datos deberían ser guardados en memorias no volátiles para volver a usarlos con posteridad.
  - Las memorias secundarias (discos duros, DVDs, CDs, …) mantienen la información aún cuando el ordenador está apagado.
- Cuando ejecutamos un programa, este suele estar almacenado en el disco duro del ordenador y primero se carga en memoria RAM. De ahí, las instrucciones se llevan a la cachè conforme el procesador las va solicitando.



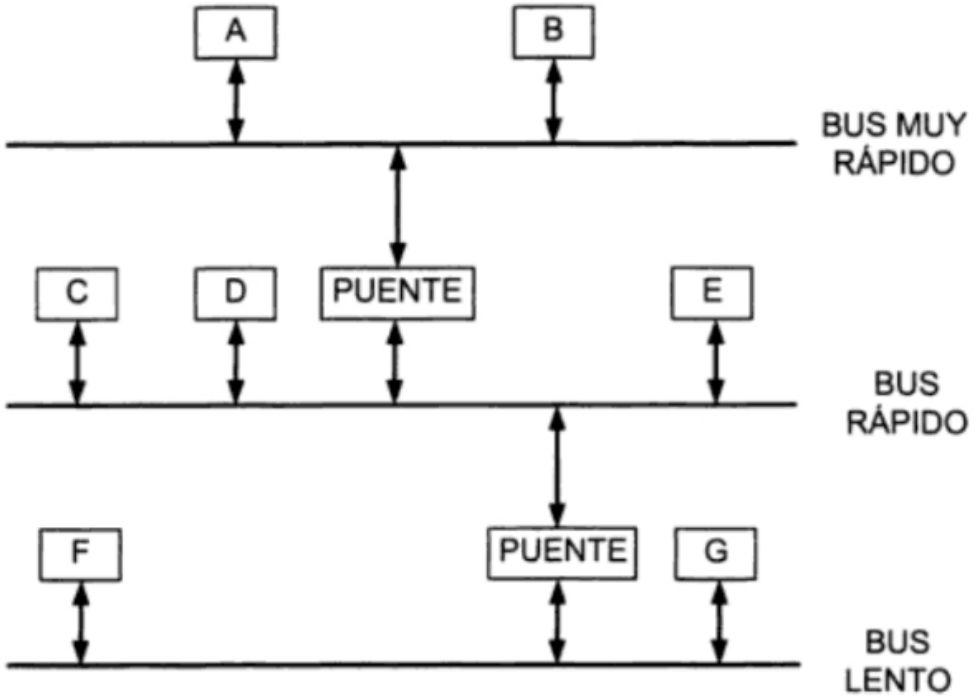
Índice

Jerarquía de buses

- Concepto de bus:
  - Canal de comunicación compartido por varios dispositivos.
  - De forma simplificada podría verse como un conjunto de líneas a las que se conectan los dispositivos y que permiten que la información escrita por un dispositivo pueda ser leída por el resto.
  - En un instante determinado únicamente un dispositivo podría poner información sobre el bus. Además, de todos los dispositivos conectados hay uno que actúa como maestro de bus (tiene el control del bus y decide quién puede poner datos sobre el mismo).
- La conexión de todos los componentes de un computador a través de un único bus plantea varios problemas:
  - Cuanto mayor es el número de dispositivos conectados, peor es el compartimiento temporal de las señales que viajan por el bus, lo que disminuye las prestaciones del mismo.
  - El bus se convierte en un cuello de botella ya que todas las transferencias de información pasan a través de él.
  - El bus debe soportar elementos de velocidades muy dispares, lo que implica un diseño de bus poco óptimo.

Jerarquía de buses

- Para resolver los problemas anteriores los computadores modernos implementan una jerarquía de buses
  - El bus de mayor velocidad es al que estaría conectado el procesador y el bus inferior conectaría dispositivos de entrada/salida lentos
  - Los distintos buses se interconectan por medio de puentes *(bridges)*
- Ejemplo:



- En la práctica, la jerarquía de buses forma parte de una jerarquía de interconexión más general dentro del computador.