

F.H.

Unidad 2

Arquitectura del ordenador

Unidad central de proceso



Unión Europea

Fondo Social Europeo

El FSE invierte en tu futuro



- Índice
 - Historia de la computación
 - Unidad central de proceso
 - Memoria
 - Unidad de entrada y salida. Buses



- Índice

- Historia de la computación
- Unidad central de proceso
- Memoria
- Unidad de entrada y salida. Buses



- **Visión jerárquica de un ordenador**
 - Para comprender el funcionamiento de un ordenador, es necesario tener una visión global de su **estructura jerárquica**
 - Es importante darse cuenta de que el sistema completo está formado a su vez por **subsistemas relacionados entre sí** de forma escalonada



- **Visión jerárquica de un ordenador**

Aplicación
Lenguajes de alto nivel
Sistema operativo
Lenguaje máquina
Microprogramación
Lógica digital

- **Nivel 5 – Aplicación**

- Es el nivel más alejado de la realidad física en el cual el **usuario** no tiene en cuenta los niveles inferiores



- **Visión jerárquica de un ordenador**

Aplicación
Lenguajes de alto nivel
Sistema operativo
Lenguaje máquina
Microprogramación
Lógica digital

- **Nivel 4 – Lenguajes de alto nivel**

- Lenguajes de programación con **alta abstracción** respecto al hardware
- El programador encuentra un **entorno amigable** para la codificación de algoritmos



- **Visión jerárquica de un ordenador**

Aplicación
Lenguajes de alto nivel
Sistema operativo
Lenguaje máquina
Microprogramación
Lógica digital

- **Nivel 3 – Sistema operativo**

- El **intermediario** SW-HW
- Conjunto de programas que proporcionan facilidades a los niveles superiores en la **gestión de los recursos** del sistema
- Se crea un **entorno favorable** para que el usuario interactúe con los niveles inferiores de la máquina



- **Visión jerárquica de un ordenador**

Aplicación
Lenguajes de alto nivel
Sistema operativo
Lenguaje máquina
Microprogramación
Lógica digital

- **Nivel 2 – Lenguaje máquina**

- El nivel mínimo **accesible por el usuario**
- Instrucciones que forman el lenguaje directamente interpretable por el hardware
- Se debe **conocer la arquitectura** de la máquina en cuestión:
 - *Tipos de datos*
 - *Organización del subsistema de memoria*
 - *Modos de direccionamiento*
 - ...



- **Visión jerárquica de un ordenador**

Aplicación
Lenguajes de alto nivel
Sistema operativo
Lenguaje máquina
Microprogramación
Lógica digital

- **Nivel 1 – Microprogramación**

- En este nivel se encuentran los **microprogramas**
- Su tarea consiste en interpretar las instrucciones del nivel superior y convertirlas a **binario**



- **Visión jerárquica de un ordenador**

Aplicación
Lenguajes de alto nivel
Sistema operativo
Lenguaje máquina
Microprogramación
Lógica digital

- **Nivel 0 – Lógica digital**

- Corresponde con el **hardware** real de la máquina
- *Dispositivos y circuitos electrónicos*



- **Visión jerárquica de un ordenador**
 - *Ejemplo: Estamos escribiendo una carta con el procesador de texto y le damos al icono de guardar*

Aplicación
Lenguajes de alto nivel
Sistema operativo
Lenguaje máquina
Microprogramación
Lógica digital

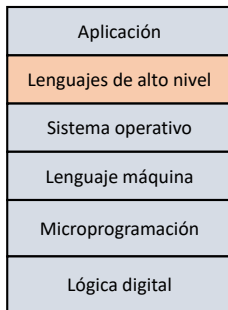
- **Nivel 5 – Aplicación**
 - Cuando le damos a guardar (en Microsoft Word, por ejemplo) no pensamos en todo lo que ocurre por debajo...





- **Visión jerárquica de un ordenador**

- *Ejemplo: Estamos escribiendo una carta con el procesador de texto y le damos al icono de guardar*



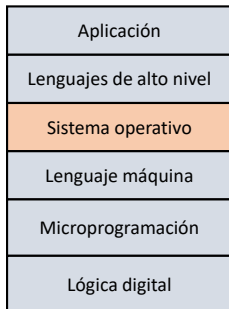
- **Nivel 4 – Lenguajes de alto nivel**

- Las aplicaciones están programadas usando lenguajes de alto nivel (Java, C#, C++,...)

```
int writeFile(FILE* f, const char* filename) {  
    if(f != NULL && filename[0] != '\0') {  
        ...  
    }  
    else {  
        return 1;  
    }  
}
```



- **Visión jerárquica de un ordenador**
 - *Ejemplo: Estamos escribiendo una carta con el procesador de texto y le damos al icono de guardar*

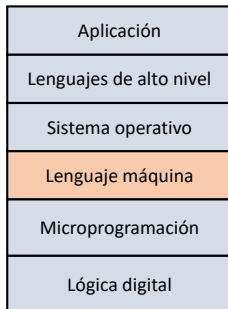


- **Nivel 3 – Sistema operativo**
 - Al sistema operativo (Windows en este caso) le llega la orden de guardar el fichero desde el lenguaje de programación
 - Es él quien sabe cómo acceder al disco duro





- **Visión jerárquica de un ordenador**
 - *Ejemplo: Estamos escribiendo una carta con el procesador de texto y le damos al icono de guardar*



- **Nivel 2 – Lenguaje máquina**
 - Las órdenes se convierten a un lenguaje de bajo nivel (*ensamblador*) adaptado a la arquitectura de la máquina (x86, ARM, SPARC,...)

```
MOV ECX, [ESP+44]  
MOV EAX, 0FFFFFF0H  
SHR EAX, CL  
...
```



- **Visión jerárquica de un ordenador**
 - *Ejemplo: Estamos escribiendo una carta con el procesador de texto y le damos al icono de guardar*

Aplicación
Lenguajes de alto nivel
Sistema operativo
Lenguaje máquina
Microprogramación
Lógica digital

- **Nivel 1 – Microprogramación**
 - En este nivel se traducen las instrucciones anteriores a código binario

0111011100011010101010111011100011011010111



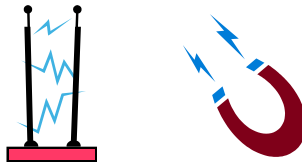
- **Visión jerárquica de un ordenador**

- *Ejemplo: Estamos escribiendo una carta con el procesador de texto y le damos al icono de guardar*

Aplicación
Lenguajes de alto nivel
Sistema operativo
Lenguaje máquina
Microprogramación
Lógica digital

- **Nivel 0 – Lógica digital**

- Se escriben físicamente los datos en el disco duro mediante electricidad, magnetismo, etc.



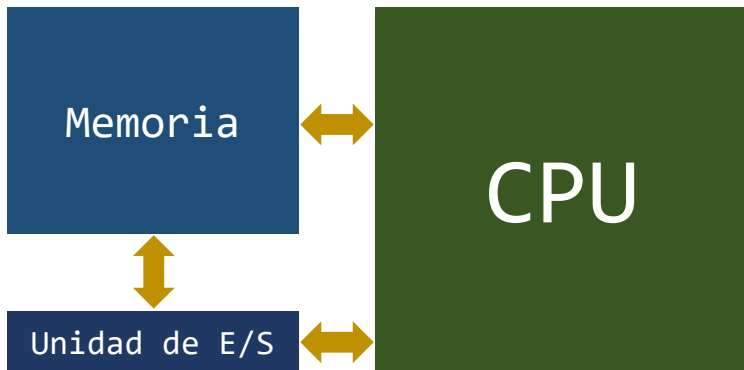


- Máquina de Von Neumann
 - Como ya se estudió en bloque anterior, Von Neumann describió el **fundamento teórico** de construcción de un ordenador
 - La idea era **conectar permanentemente** las unidades del ordenador, de manera que su funcionamiento estuviera **coordinado bajo un control central**
 - La arquitectura es todavía, aunque con algunos cambios, la que emplean la mayoría de los fabricantes de ordenadores

Unidad central de proceso

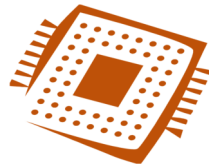


- Máquina de Von Neumann





- **Unidad Central de Proceso (CPU)**
 - También denominada **procesador** o **microprocesador**
 - Encargado de:
 - **Control y ejecución de las operaciones** que se efectúan dentro del ordenador, con el fin de realizar el tratamiento automático de la información



Unidad central de proceso



- El procesador es el cerebro del ordenador:
 - Encargado de **controlar** todas las **tareas y procesos** que se realizan dentro de él:
 - La información que se va a procesar
 - La memoria
 - Los dispositivos periféricos
 - ...



Unidad central de proceso



- Está formado por:
 - La unidad de control (**UC**)
 - La unidad aritmético-lógica (**ALU**)
 - Los **registros** de propósito general
 - Los **buses**

Unidad central de proceso



- Está formado por:
 - La unidad de control (UC)
 - **Interpreta y ejecuta** las instrucciones almacenadas en la memoria principal
 - Genera las **señales de control** necesarias para ejecutarlas

Unidad central de proceso



- Está formado por:
 - La unidad de control (UC)
 - La unidad aritmético-lógica (**ALU**)
 - Recibe los datos sobre los que efectúa **operaciones de cálculo y comparaciones**
 - Toma **decisiones lógicas** y devuelve un resultado
 - Realiza todo esto bajo la supervisión de la **UC**

Unidad central de proceso



- Está formado por:
 - La unidad de control (**UC**)
 - La unidad aritmético-lógica (**ALU**)
 - Los **registros** de propósito general
 - Constituyen el **almacenamiento interno** de la CPU
 - Almacenamiento de **muy alta velocidad**
 - Capacidad muy **pequeña**
 - Se almacena la **información temporal**

Unidad central de proceso



- Está formado por:
 - La unidad de control (UC)
 - La unidad aritmético-lógica (ALU)
 - Los registros de propósito general
 - Los **buses**
 - Caminos por donde circulan instrucciones y datos **entre las distintas unidades** del ordenador





- Registros de propósito general
 - En el interior del procesador existen unas **celdas de memoria** de **alta velocidad** que permiten a la CPU almacenar datos temporalmente mientras se efectúa alguna operación
 - Estos **registros de propósito general** constituyen la **memoria interna del procesador**







- Registros de datos
 - Se usan para **almacenar datos**
 - Cuando un dato es solicitado, si se deja en uno de estos registros, no es necesario acceder a memoria
 - Están formados por un **conjunto de bits** que se manipulan **en bloque**
 - El número de bits varía dependiendo de la CPU, pero siempre son **potencias de 2**
 - 8, 16, 32, 64, 128, 256,...



- Registros de datos
 - Este tamaño de registro indica el número de bits que puede manipular a la vez el procesador
 - Cuanto mayor sea, **más potente** será el procesador, puesto que podrá trabajar con más cantidad de información a la vez

¿Os suenan los microprocesadores de 32 y 64 bits?



- Registros de datos
 - Las primeras CPU eran de **8 bits** (*años 70*)
 - Los primeros PC (*años 80*) tenían CPU de **16 bits**, con lo que solo podían ejecutar software de 16 bits, como MS-DOS o Windows 3.x
 - Con **32 bits** (*Windows 95*) se puede utilizar una **mayor cantidad de memoria**



- Registros de datos
 - Mientras que en la arquitectura de **32 bits**, el máximo de RAM se limita a 4GB (2^{32} bits)...
 - ...con la nueva arquitectura de **64 bits** se puede direccionar una memoria “casi infinita” (2^{64} bits)

16 Exabytes

16 millones de Terabytes



- Registros de estado (flags)
 - Son bits fijados mediante hardware que indican
 - Si una operación devuelve un resultado positivo, negativo o nulo
 - *Calculamos la resta 3-7*
 - *En el registro de estado se indicará que el resultado es negativo*
 - Si hay desbordamiento (overflow)
 - *Calculamos la potencia 10000^{900000}*
 - *En el registro de estado se indicará que ha habido un desbordamiento (el resultado es muy grande y no hay manera de almacenarlo) → Probadlo en la calculadora*
 - Si hay acarreo en una operación
 - *Sumamos en binario 111+111*
 - *En el registro de estado se indicará que hay un acarreo*
 - ...



Unidad central de proceso



- Contador de programa (**CP**)
 - Contiene la dirección **de la siguiente instrucción a ejecutar**
 - Su valor es actualizado por la CPU después de capturar una instrucción

Dirección	Contenido
1	Leer dato A
2	Leer dato B
3	Calcular $C=A+B$
4	Guardar C
...	

CP
2

El CP indica que hay que ejecutar la instrucción que está en la dirección 2 (en este caso, "Leer dato B")

Al terminar de ejecutarla, el valor de CP será 3



- Registro de instrucción (RI)
 - Contiene el **código de la instrucción actual** (en binario)
 - Aquí **se analiza** el código de operación
 - *Ejemplo (ficticio):*
 - Código = 10110110 → Operación de sumar dos números
 - Código = 10100101 → Leer un dato desde teclado

RI
10110110



- Registro de dirección de memoria (**RDM**)
 - Contiene la **dirección de una posición de memoria** donde se encuentra o donde va a ser almacenada la **información**
 - Este intercambio se realiza a través del **bus de direcciones**

Dirección	Contenido
...	
120	
121	
122	
...	

RDM
122

El RDM indica que utilizar el contenido de la dirección 122 de memoria (ya sea para leer un dato desde ahí o para escribir un dato ahí)

Unidad central de proceso



- Registro de intercambio de memoria (**RIM**)
 - **Recibe o envía** (según si es operación de lectura o de escritura) **el dato** contenido en la posición apuntada por el RDM
 - El intercambio de datos con la memoria se efectúa a través del **bus de datos**

RDM
122

RIM
77

El RIM tiene almacenado un 77

Si estamos en una operación de lectura de memoria, indicará que se ha leído un 77 en la posición de memoria 122 (indicada por el RDM)

Si estamos en una operación de escritura en memoria, indicará que se va a escribir un 77 en la posición de memoria 122



Ejemplo de funcionamiento

1. El **CP** contiene la dirección de memoria de la siguiente instrucción a ejecutar; para buscarla, el contenido de esa posición se pasa al **RDM**
2. La instrucción apuntada por el **RDM** se carga en el **RIM** y desde aquí se pasa al **RI**
3. El decodificador de instrucciones interpreta el contenido de **RI**, y se generan las órdenes oportunas para su ejecución
4. El **CP** se incrementa en 1, para apuntar a la siguiente instrucción a ejecutar

Unidad central de proceso



- Unidad de control
 - Se encarga de **interpretar y ejecutar las instrucciones** máquina que conforman los programas
 - Genera las **señales de control** para llevarlas a cabo
 - *Es como el director de una orquesta*





- Unidad de control
 - Componentes de la UC:
 - Decodificador de instrucción
 - Reloj
 - Secuenciador
 - Contador de programa
 - Registro de instrucción
- } Ya los hemos estudiado antes



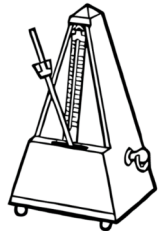
- Unidad de control
 - Componentes de la UC:
 - Decodificador de instrucción
 - **Extrae e interpreta el código de operación** de la instrucción contenida en el registro de instrucción
 - Le indica a la UC:
 - Qué instrucción debe ejecutar
 - Cuántos operandos tiene una operación
 - Dónde están las instrucciones u operandos en memoria
 - Dónde debe guardar los cálculos realizados
 - ...

Unidad central de proceso



- Unidad de control
 - Componentes de la UC:
 - Decodificador de instrucción
 - Reloj
 - Proporciona una **sucesión de impulsos eléctricos** a intervalos constantes
 - **Marca los tiempos de ejecución** de los pasos a realizar para cada instrucción
 - Sincroniza todo el sistema (todo va al compás del reloj)

CPUs de 6,2 GHz
6,2 mil millones de ciclos por segundo
¡6.200.000.000!





- Unidad de control
 - Componentes de la UC:
 - Decodificador de instrucción
 - Reloj
 - Secuenciador
 - **Genera las órdenes elementales** que, sincronizadas con los impulsos de reloj, hacen que se ejecute paso a paso y de manera ordenada la instrucción cargada en él
 - Contador de programa
 - Registro de instrucción



- Unidad aritmético-lógica (ALU)
 - Es la parte de la CPU que se encarga de realizar las operaciones de tipo **aritmético** (suma, multiplicación,...), así como las de tipo **lógico** (comparación, negación,...)
 - Para llevar a cabo una operación aritmética, necesita el **código de operación** que indique la operación a efectuar
 - Esto lo sabe la UC después de haber decodificado la instrucción
- ✍ Por ejemplo: si queremos hacer una suma, hay que indicar:*
- *La operación es una suma*
 - *Las direcciones de las celdas de memoria donde están los operandos*
 - *La celda donde se almacenará el resultado*



- Unidad aritmético-lógica (ALU)
 - Componentes de la ALU:
 - Circuito combinacional
 - Realiza las **operaciones** con los datos de los registros de entrada
 - Registro acumulador
 - Almacena los **resultados** de las operaciones
 - La ALU es la que utiliza el registro de estado
 - Registra las **condiciones** de la operación anterior



- Unidad aritmético-lógica (ALU)
 - Una parte importante de la ALU es la **unidad de coma flotante (FPU)**, también conocido por *coprocesador matemático*
 - Se encarga de manejar las operaciones en coma flotante
 - Números fraccionarios
 - Matemática trigonométrica
 - Matemática logarítmica
 - Antes de su aparición, la ALU efectuaba las operaciones en coma flotante de manera muy lenta
🌀 Lo que la FPU hace en 1 ciclo de reloj, la ALU lo hacía en 100



¿Sabías que...?

- Una primera medida de la velocidad de un procesador lo da su **frecuencia** de reloj (Hz)
- Pero para medir su rendimiento, se tienen en cuenta (también) otros factores:
 - Cantidad de **instrucciones** ejecutadas por segundo (**MIPS**, *millones de instrucciones por segundo*)
 - Cantidad de **operaciones en coma flotante** por segundo (**MFLOPS**, *millones de operaciones en coma flotante por segundo*). Expresa la potencia de cálculo científico de un ordenador
 - **Instrucciones por ciclo** de reloj (**IPC**, *instrucciones por ciclo*)....



- Ejecución de una instrucción
 - La CPU ejecuta los programas que se encuentran **cargados** en la **memoria principal**
 - Estos programas están formados por un conjunto de instrucciones de las cuales, a la hora de ejecutarse, podemos distinguir **dos fases**:
 - Fase de **búsqueda**
 - Fase de **ejecución**



- Ejecución de una instrucción
 - Fase de búsqueda
 - **Localiza la instrucción** a ejecutar dentro de la memoria principal y **la lleva a la UC** para procesarla
 - Fase de ejecución
 - **Se realizan las acciones** que llevan asociadas las instrucciones
 - Por ejemplo, una suma, una resta, leer desde teclado un número,...



Caso práctico

- Vamos a ver los pasos que hay que seguir para ejecutar un programa que:
 - Lee dos números que introduce el usuario por teclado
 - Los suma (almacenando el resultado en memoria)
 - Visualiza el resultado por pantalla
- El mapa de memoria dado sería el siguiente:

Unidad central de proceso



Caso práctico

Memoria			
Programa		Datos	
1	9	17	25
2	10	18	26
3	11	19	27 Dato A
4 Leer A	12	20	28 Dato B
5 Leer B	13	21	29 Dato C
6 Calcular C=A+B	14	22	30
7 Visualizar C	15	23	31
8	16	24	32



Caso práctico

● Instrucción: Leer A

1. En el **CP** se almacena la dirección de memoria de comienzo del programa, la 4

$$CP = Dir\ 4$$

2. La **UC** envía la orden para que el contenido del **CP**, que es la dirección de la instrucción que estamos analizando, se transfiera al **RDM**

$$RDM = 4$$



Caso práctico

● Instrucción: Leer A

3. El selector de memoria localiza la posición 4 y transfiere su contenido al **RIM**

RIM = "Leer A"

4. La **UC** da la orden de transferir el contenido del RIM al registro de instrucción (**RI**), donde se deposita el código de la instrucción a ejecutar

RI = "Leer A"



Caso práctico

● Instrucción: Leer A

5. El decodificador de instrucción analiza el código contenido en el **RI** y genera las señales de control para ejecutar correctamente la instrucción
6. El **CP** se incrementa en 1 y apuntará a la instrucción siguiente

CP = Dir 5



Caso práctico

● Instrucción: Leer A

7. Una vez conocido el código de operación, la **UC** establece las conexiones con el dispositivo de entrada para recibir el dato A

Periférico → CPU



Caso práctico

● Instrucción: Leer A

8. La **UC** da la orden de que el dato leído se cargue en el **RIM** y que en el **RDM** se cargue la dirección de memoria donde se va a almacenar el dato leído (la 27)

$$RIM = \text{Dato A}$$

$$RDM = 27$$

9. El selector de memoria se encarga de guardar en la dirección indicada por el **RDM** (la 27) el contenido del **RIM**

$$Mem(27) = \text{Dato A}$$



Caso práctico

● Instrucción: Leer B

1. En el **CP** se almacena la dirección de memoria de la instrucción a ejecutar, la 5

$$CP = Dir\ 5$$

2. La **UC** envía la orden para que el contenido del **CP**, que es la dirección de la instrucción que estamos analizando, se transfiera al **RDM**

$$RDM = 5$$



Caso práctico

● Instrucción: Leer B

3. El selector de memoria localiza la posición 5 y transfiere su contenido al **RIM**

RIM = "Leer B"

4. La **UC** da la orden de transferir el contenido del **RIM** al **RI**, donde se deposita el código de la instrucción a ejecutar

RI = "Leer B"



Caso práctico

● Instrucción: Leer B

5. El decodificador de instrucción analiza el código contenido en el **RI** y genera las señales de control para ejecutar correctamente la instrucción
6. El **CP** se incrementa en 1 y apuntará a la instrucción siguiente

$$CP = Dir + 1$$



Caso práctico

● Instrucción: Leer B

7. Una vez conocido el código de operación, la **UC** establece las conexiones con el dispositivo de entrada para recibir el dato B

Periférico → CPU

Fase de ejecución



Caso práctico

● Instrucción: Leer B

8. La **UC** da la orden de que el dato leído se cargue en el **RIM** y que en el **RDM** se cargue la dirección de memoria donde se va a almacenar el dato leído (la 28)

$$RIM = \text{Dato } B$$

$$RDM = 28$$

9. El selector de memoria se encarga de guardar en la dirección indicada por el **RDM** (la 28) el contenido del **RIM**

$$Mem(28) = \text{Dato } B$$



Caso práctico

● Instrucción: Calcular $C=A+B$

1. En el **CP** se almacena la dirección de memoria de la instrucción a ejecutar, la 6

$$CP = Dir\ 6$$

2. La **UC** envía la orden para que el contenido del **CP**, que es la dirección de la instrucción que estamos analizando, se transfiera al **RDM**

$$RDM = 6$$



Caso práctico

● Instrucción: Calcular $C=A+B$

3. El selector de memoria localiza la posición 6 y transfiere su contenido al **RIM**

RIM = "Calcular $C=A+B$ "

4. La **UC** da la orden de transferir el contenido del **RIM** al **RI**, donde se deposita el código de la instrucción a ejecutar

RI = "Calcular $C=A+B$ "



Caso práctico

- **Instrucción: Calcular $C=A+B$**

5. El decodificador de instrucción analiza el código contenido en el **RI** y genera las señales de control para ejecutar correctamente la instrucción
6. El **CP** se incrementa en 1 y apuntará a la instrucción siguiente

$$CP = Dir + 1$$



Caso práctico

● Instrucción: Calcular $C=A+B$

7. Se transfiere la dirección del primer operando A (27) desde el **RI** hasta el **RDM**
 - *Recuerda que en el RI está la instrucción del cálculo de la suma: el código de operación y la dirección de los operandos*

$$RDM = 27$$

8. El selector de memoria extrae el contenido de la posición 27 y lo deposita en el **RIM**

$$RIM = \text{Dato A}$$



Caso práctico

● Instrucción: Calcular $C=A+B$

9. El contenido del **RIM** se carga en el registro acumulador de la **ALU**

Acumulador = Dato A

10. Se transfiere la dirección del segundo operando B (28) desde el **RI** hasta el **RDM**

RDM = 28

11. El selector de memoria extrae el contenido de la posición 28 y lo deposita en el **RIM**

RIM = Dato B



Caso práctico

● Instrucción: Calcular $C=A+B$

12. Se envía la orden para que el contenido del **RIM** se sume al contenido del registro acumulador de la **ALU** y el resultado se almacene en el acumulador

$$\text{Acumulador} = \text{Dato A} + \text{Dato B}$$

13. Este resultado es enviado desde el registro acumulador al **RIM** para almacenarlo en memoria, pero antes hay que localizar en qué dirección de memoria se va a almacenar

$$RIM = A+B$$



Caso práctico

● Instrucción: Calcular $C=A+B$

14. Se transfiere desde el **RI** al **RDM** la dirección donde se va a almacenar el resultado; es decir, la dirección de C (29)

$$RDM = 29$$

15. Finalmente, se transfiere el resultado desde el **RIM** a la dirección indicada por el **RDM**, quedando el resultado C almacenado en la posición 29

$$M(29) = A+B$$



Caso práctico

● Instrucción: Visualizar C

1. En el **CP** se almacena la dirección de memoria de la instrucción a ejecutar, la 7

$$CP = Dir\ 7$$

2. La **UC** envía la orden para que el contenido del **CP**, que es la dirección de la instrucción que estamos analizando, se transfiera al **RDM**

$$RDM = 7$$



Caso práctico

● Instrucción: Visualizar C

3. El selector de memoria localiza la posición 7 y transfiere su contenido al **RIM**

RIM = "Visualizar C"

4. La **UC** da la orden de transferir el contenido del **RIM** al **RI**, donde se deposita el código de la instrucción a ejecutar

RI = "Visualizar C"



Caso práctico

● Instrucción: Visualizar C

5. El decodificador de instrucción analiza el código contenido en el **RI** y genera las señales de control para ejecutar correctamente la instrucción
6. Como el programa ha terminado, el **CP** no se incrementará



Caso práctico

● Instrucción: Visualizar C

7. Una vez conocido el código de operación, la **UC** establece las conexiones con el dispositivo de salida para visualizar el dato C

CPU → Periférico

Fase de ejecución



Caso práctico

● Instrucción: Visualizar C

8. La dirección del dato a visualizar se carga en el **RDM** (29)

$$RDM = 29$$

9. El selector de memoria selecciona esa posición y guarda en el **RIM** el contenido del dato C

$$RIM = \text{Dato } C$$

10. Se generan las ordenes para que el contenido del **RIM** salga por la unidad de salida



Ejercicio práctico

- Vamos a ver los pasos que hay que seguir para ejecutar un programa que:
 - Lee un número que introduce el usuario
 - Lo multiplica a uno que había en memoria
 - Le suma un número almacenado en un registro de datos (R3)
 - Visualiza el resultado
- El mapa de memoria dado sería el siguiente:

Unidad central de proceso



Ejercicio práctico

Memoria			
Programa		Datos	
1	9	17	25 Dato B
2 Leer A	10	18	26
3 Calcular $D=A \times B + R1$	11	19 Dato A	27
4 Visualizar D	12	20	28
5	13	21	29
6	14	22	30 Dato D
7	15	23	31
8	16	24	32