

# **1. INTRODUCCIÓN**

## **1.1. GESTIÓN DE DATOS EN MISIONES ESPACIALES**

La gestión de datos es un componente crítico de la misión. La mayoría de aspectos de la misión son controlados por ordenadores y software: trayectoria de lanzamiento y adquisición de órbita, control de actitud, control de la plataforma, telecomunicaciones, etc.

El subsistema OBDH se encarga de intercambiar datos entre las unidades funcionales del satélite y el segmento tierra. Además, almacena y distribuye información dentro del satélite. Está muy integrado con el subsistema de TT&C (*Telemetry, tracking and telecommand*). Se basa en uno o más OBC (*On-board Computer*).

## **1.2. COMPUTADORES DE A BORDO**

Requisitos específicos:

- Embebido en el satélite: dispositivos de E/S específicos, potencia, tamaño y peso limitados.
- Operación en tiempo real
- Requisitos de alta fiabilidad: no es posible la reparación en espacio
- Ambiente severo: radiación, temperatura, vibración, aceleración

La tecnología aplicable podría estar limitada, a menudo resultando en el uso de tecnología menos potente.

El diseño del hardware del OBC está regido por los requisitos de fiabilidad en un ambiente severo.

El diseño del software del OBC está regido por los requisitos de alta integridad.

## 2. ESTRUCTURA DE COMPUTADORES

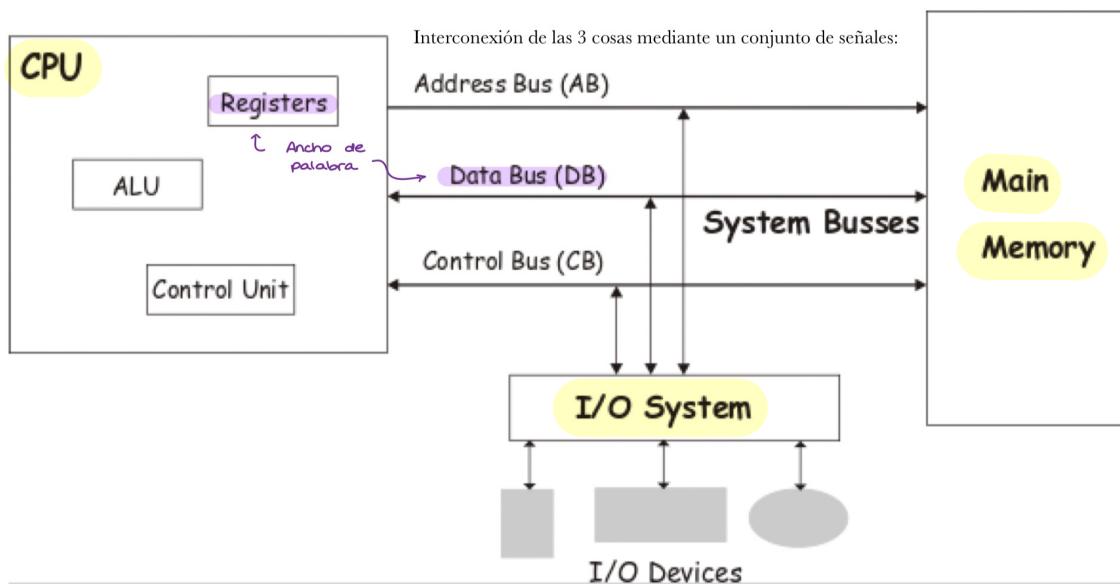
### 2.1. INTRODUCCIÓN

-**Función básica:** Ejecución de instrucciones elementales, en las que están especificados la operación a realizar, los datos o su localización, y la localización del resultado.

-**Arquitectura de Von Neumann:**

- Los datos e instrucciones están almacenados en una memoria única de lectura/escritura.
- El contenido de la memoria es accesible por direcciones.
- La ejecución es implícitamente secuencial -> Registro de contador de programa (PC).

### 2.2. ESQUEMA BÁSICO DEL COMPUTADOR DE VON NEUMANN



#### 2.2.1. Parámetros característicos:

-**Ancho de palabra o palabra:** Número de líneas del bus de datos y número de bits de los registros de la CPU.

-**Frecuencia de reloj:** Frecuencia a la que la CPU realiza las operaciones elementales.

-**Capacidad de cómputo:** Velocidad de ejecución de instrucciones: MIPS (*Millones de Instrucciones Por Segundo*), MFLOPS (Million Floating Point Operations Per Second).

#### 2.2.2. Terminología:

-**Procesador:** CPU

➤ **Microprocesador:** procesador integrado en un único chip.

-**Multiprocesador:** computador con más de un procesador que comparten la memoria principal y ejecutan instrucciones en paralelo.

➤ **Multicore:** multiprocesador integrado en un único chip.

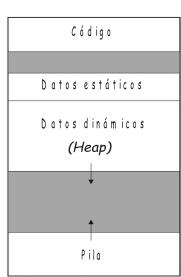
- **Manycore**: multicore con muchos procesadores.

Teniendo en cuenta que la capacidad de integración ha mejorado notablemente, normalmente es conveniente integrar varios procesadores en un único circuito en lugar de utilizar un solo procesador más potente (compensa más utilizar un multiprocesador de 8 CPUs que una CPU 8 veces más potente). Se utiliza en los ordenadores personales.

-**SoC (System on Chip)**: Computador completo (CPU(s), Sistema de entrada/salida e incluso dispositivos periféricos y memorias) en un único chip. Término de uso habitual en hardware que permite construir computadores a medida. La capacidad e integración en este caso se aprovecha integrando el procesador y la memoria en el mismo circuito.

Es lo que se suele implementar en OBC por limitaciones de potencia y tamaño.

### 2.2.3. Organización de la memoria principal:



- Sección de **código o texto**: Contiene las instrucciones del programa cargado en memoria (es decir, el que se está ejecutando).
- Sección de **datos estáticos (data y bss)**: Contiene las variables globales del programa.
- Sección de **pila (stack)**: Contiene las variables locales de los subprogramas.
- Sección de **datos dinámicos (heap)**: se usa para estructuras que se crean y destruyen explícitamente en el programa.

### 2.2.3. Tipos de memoria

**-Memoria volátil:** Es la memoria cuya información se pierde al desconectar la corriente eléctrica. Esto es, cuando se apaga la alimentación, se borran todos los datos. El tipo más común:

- **RAM (Random Access Memory)**: Memoria que se puede leer y escribir con el mismo tiempo de acceso. Se utiliza como memoria principal para contener instrucciones y datos porque el tiempo de acceso a esta memoria es mucho menor.
- Variantes: SRAM (mientras la memoria tenga alimentación, los datos no se pierden) y DRAM (cada cierto tiempo hay que reescribir la información porque la pierde).

**-Memoria no volátil:** Tipo de memoria que, al apagar la alimentación, mantiene el almacenamiento de datos. Si toda la memoria de un ordenador fuese RAM, no se podría hacer que el ordenador arranque porque el procesador necesita una memoria a la que acceder para buscar instrucciones.

- **ROM (Read Only Memory)**: Memoria que solo se puede leer y se escribe durante su construcción.
- **EEPROM (Electrically Erasable Programmable Read-Only Memory)**. Memoria que se puede leer y escribir pero con distintos tiempos de acceso. No se puede escribir con la CPU, necesita aparatos específicos para el proceso de escritura. Se utiliza como

unidad de almacenamiento en computadores empotrados. Una variante es la memoria Flash, que la puede borrar y reescribir el propio procesador.

#### 2.2.4. Memoria del OBC del UPMSat-2

-Memoria RAM estática (SRAM): Contiene las instrucciones del programa y sus datos.

-Memoria EEPROM. Contiene:

- Las instrucciones y datos inicializados del programa que se copian a RAM durante el inicio.
- Los datos de configuración de los distintos subsistemas software: ADCS, Housekeeping, TM/TC, ...
- La telemetría que se mandará en el próximo periodo de cobertura
- Los telecomandos diferidos que aún no han sido procesados.

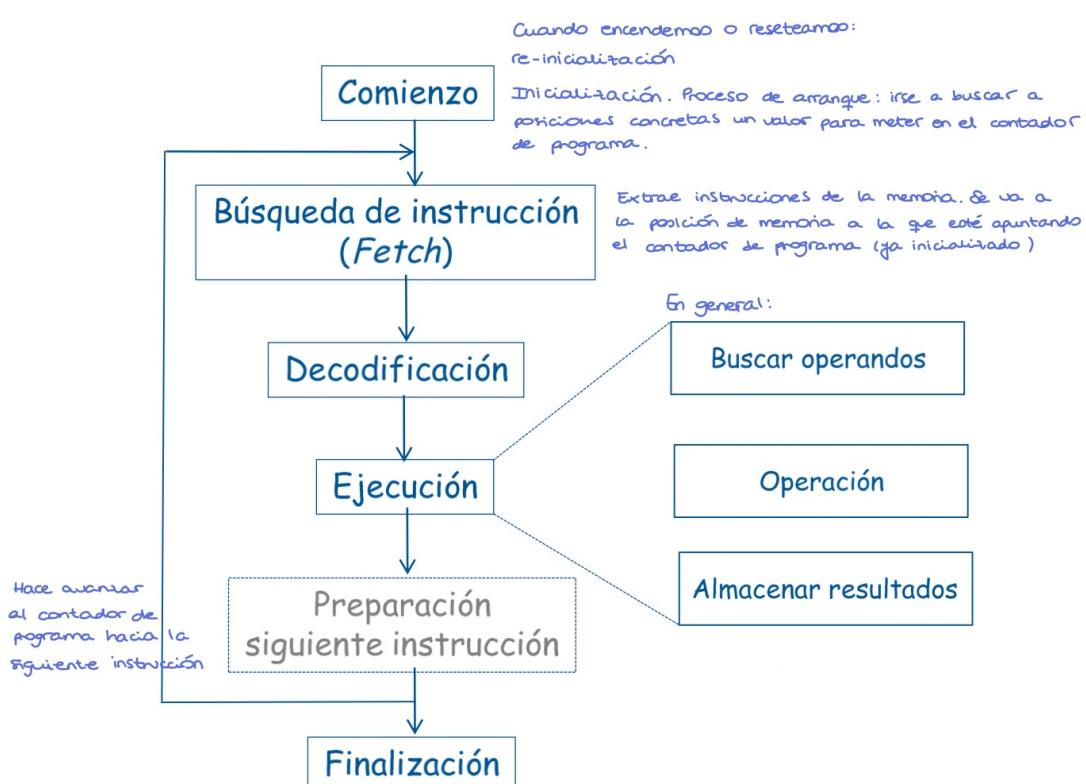
#### 2.2.5. Lenguaje máquina

Cada programa está compuesto por datos e instrucciones almacenados en memoria.

-Instrucción máquina: Función básica elemental que puede ejecutar un computador. Son cadenas de 1 y 0 con un significado particular para cada procesador.

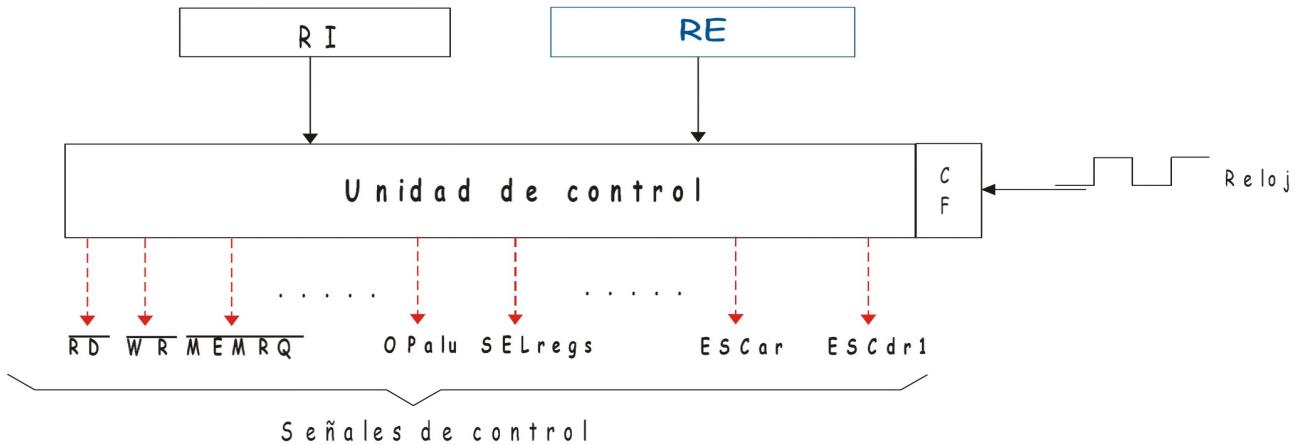
Propiedades:

- Realizan una función única y sencilla
- Tienen un número fijo de operandos
- Autocontenidas: Contienen todo lo necesario para su ejecución (operación, operandos, dirección del resultado y dirección de la siguiente instrucción)

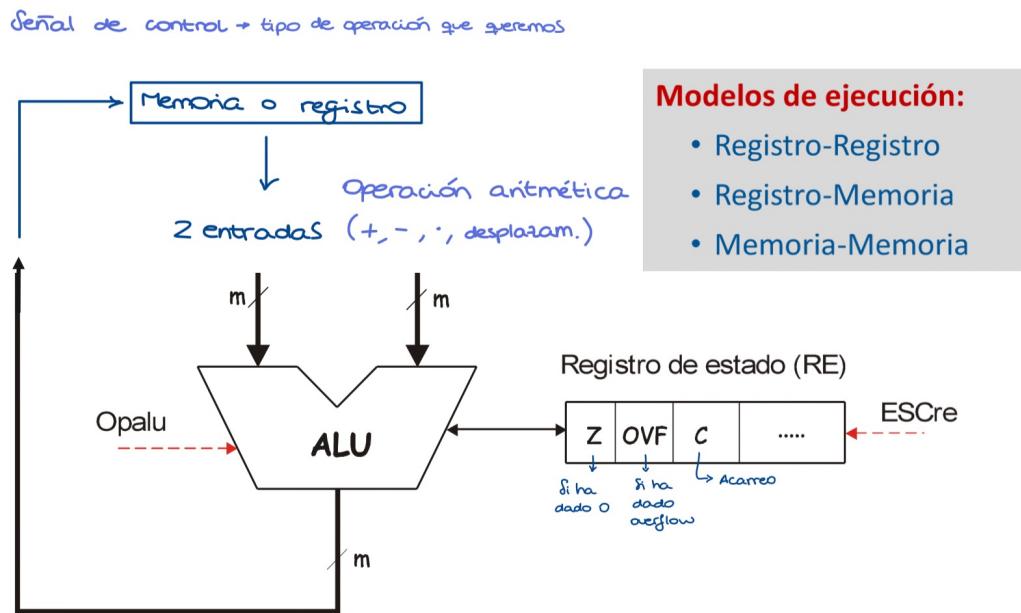


### 2.3. UNIDAD CENTRAL DE PROCESO (CPU)

- **Unidad de Control:** Decide qué hay que activar para acceder a memoria, qué hay que poner en el bus de direcciones, qué hay que poner o recoger en el bus de datos, etc.
  - Extrae de la memoria principal la instrucción a ejecutar (fetch)
  - La analiza (decodifica)
  - Da las órdenes al resto de componentes



- **Unidad Aritmético-Lógica (ALU):** Realiza la operación indicada por la Unidad de Control sobre los datos de entrada.



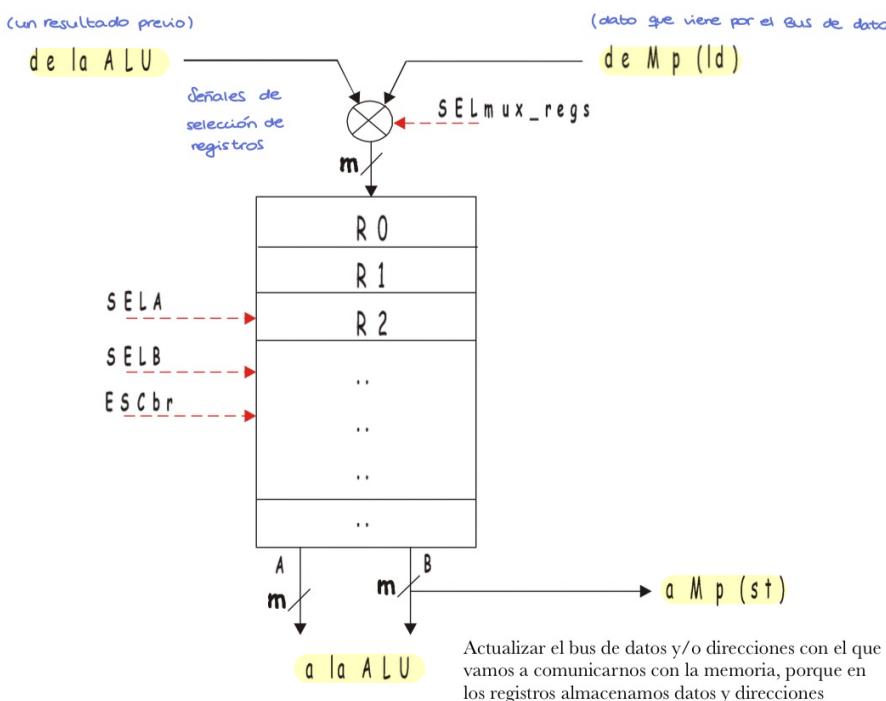
- **Juego de registros:** Espacio de memoria interno del procesador que contiene registros para manejar datos, tratar direcciones, conocer y controlar el estado del programa, etc. Es una memoria a corto plazo. Las instrucciones normalmente solo se guardan en un registro (registro de instrucción). Tipos de registros:
  - De propósito general (BR): Contienen datos y direcciones de memoria.

- De propósito específico: Contienen el estado del programa. Son transparentes o no accesibles para las instrucciones, pues sirven de uso interno para las operaciones del procesador. Los utiliza la Unidad de Control para conectar diferentes partes de la CPU.

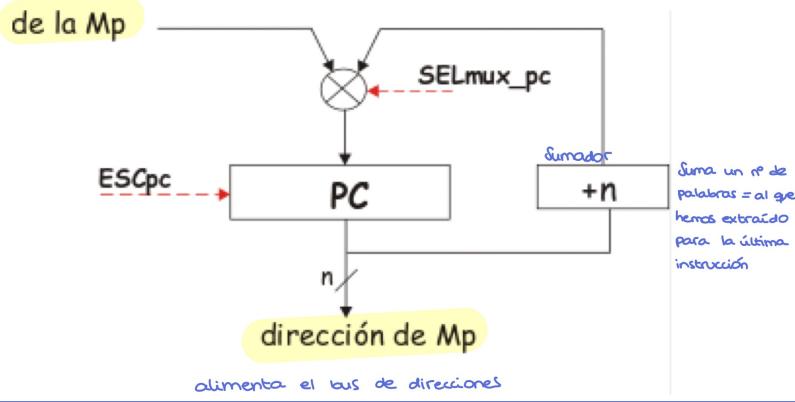
- Contador de programa (PC): Contiene la dirección de memoria donde se encuentra la siguiente instrucción a ejecutar. Básicamente, es un registro de direcciones, pero direcciones que apuntan a instrucciones (cuando hablábamos antes de registro de direcciones, se refería a direcciones de datos)
- Registro de estado (RE): Z, OVFL, C, etc.
- Registro de instrucción (RI): Donde la CPU guarda la instrucción que va a ejecutar.
- Registro de direcciones: Alimenta el bus de direcciones, bien desde la memoria o bien desde el contador de programa
- Registro de datos: Es un registro bidireccional, según si se está realizando lectura o escritura.

TRANSPARENTES

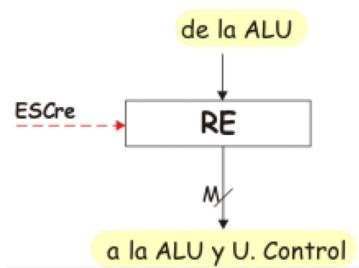
### PROPÓSITO GENERAL



### CONTADOR DE PROGRAMA



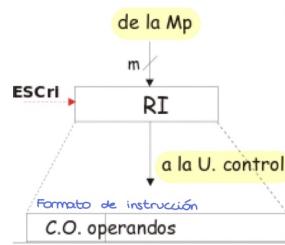
### REGISTRO DE ESTADO



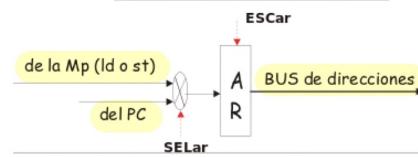
## TRANSPARENTES

### Registro de instrucción

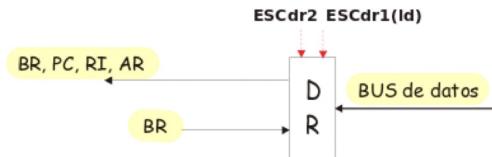
De la memoria principal con el contador de programa direccionalos la siguiente instrucción a ejecutar. La dirección de esa instrucción nos viene por el bus de datos a la CPU, alimenta el registro de instrucción, guardamos la siguiente instrucción a ejecutar y eso es la etapa de Búsqueda de instrucción (*Fetch*)



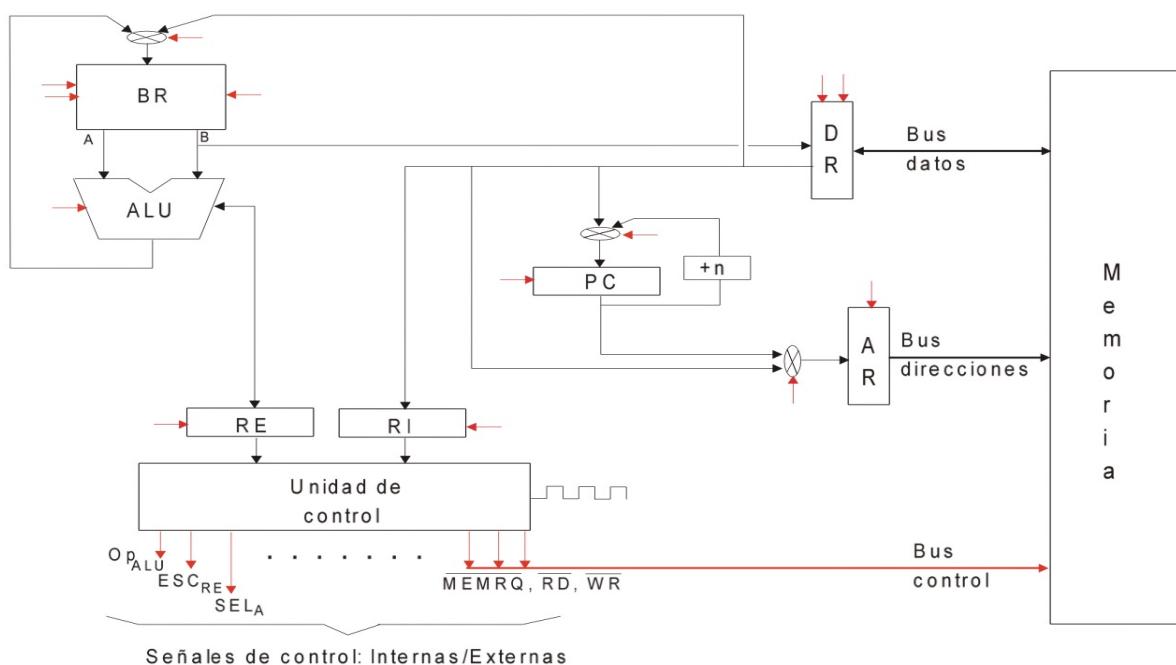
### Registro de direcciones



### Registro de datos



## Diagrama simplificado



## 2.4. COPROCESADOR DE COMA FLOTANTE (FPU)

Una CPU normalmente realiza las operaciones de suma, resta, etc., en complemento a 2. Las operaciones en coma flotante se suelen realizar con otro procesador.

-**FPU**: Unidad subordinada de la Unidad de Control que está especializada para hacer operaciones aritméticas con datos en coma flotante (suma, resta, multiplicación y división; funciones trigonométricas y transcendentales).

- También contiene registros de propósito general para contener datos en coma flotante.
- No realiza fetch de instrucciones, sino que está subordinada a la CPU.
- Normalmente son conformes al IEEE Standard for Floating-Point Arithmetic (IEEE 754).
- Realiza las operaciones en simple (32 bits), doble (64 bits) y precisión extendida (128 bits).
- Tarda mucho más en ejecutar la operación. Una división en coma flotante, que es una operación que requiere significativamente más tiempo que la suma o resta, puede tardar unas 10 veces lo que tarda una división en complemento a 2.
- Se maneja prácticamente igual que la ALU, con la diferencia de que realiza las operaciones en coma flotante.

## 2.5. JUEGO DE INSTRUCCIONES

Conjunto de instrucciones que ejecuta el procesador. Debe ser completo y eficaz. No todas las instrucciones tienen el mismo tamaño (1, 2, 3 palabras), ni todos los códigos de instrucciones tienen mismo número de bits.

-**Formato de instrucción**: Representación de una instrucción y especificación de cada campo:

- Código de operación
- Operandos (direcciones)

CO	Operando1	Operando2
----	-----------	-----------

La codificación de las instrucciones debe encajar en pocos formatos, la tendencia actual es que ocupen una palabra. Tipos de instrucciones:

- Transferencia: Transferencia de datos entre espacios (registro-registro, registro-memoria, memoria-memoria).
- Bifurcaciones: Modifican la secuencia del programa.
  - Incondicionales.
  - Condicionales. Se utilizan para realizar iteraciones hasta que se cumple una condición determinada). *Ej: Si cc = 1 , se ejecuta la bifurcación . Si no, se continúa la secuencia .*
  - Con retornos. Se utilizan para implementar saltos a subrutinas. Una subrutina es un fragmento de código que se utiliza en varias partes de un programa. Una vez realizado el subprograma, se retorna a la instrucción siguiente desde la que se bifurcó. Por tanto, es necesario almacenar la dirección de retorno, que habitualmente se almacena en la pila.
- Aritméticas. Sumas, restas, multiplicaciones, divisiones ...

- Lógicas. Se realizan bit a bit (AND, OR, NOT, ...)
- Desplazamiento y rotación. Realizan desplazamientos o rotaciones de bits a izquierda/derecha.
- De bit: Ejecutan operaciones con un bit: pone a 1 o a 0 un bit.

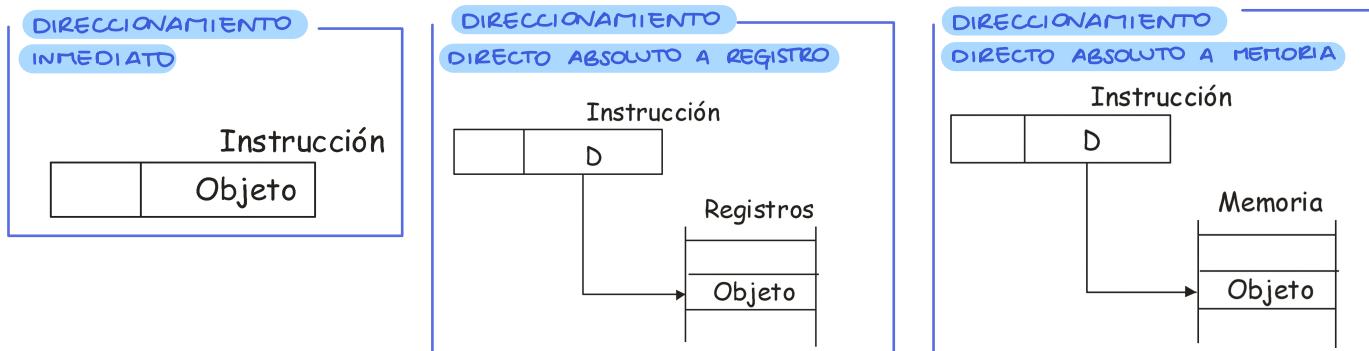
## 2.6. MODOS DE DIRECCIONAMIENTO

Forma en la que se accede a una instrucción o dato.

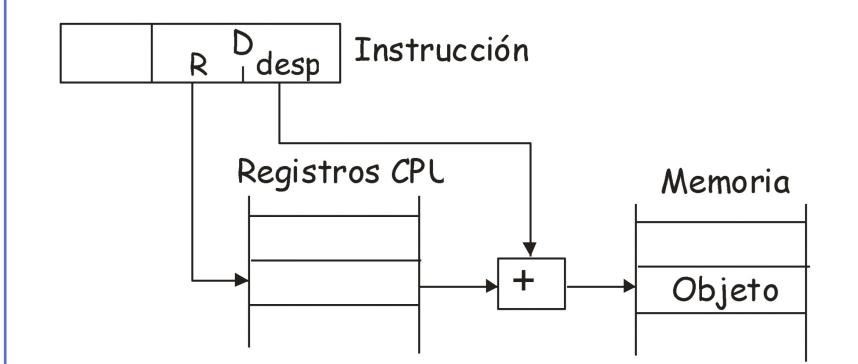
-**Objeto**: Instrucción o dato al que se desea acceder.

-**Dirección**: Lugar en el que reside el objeto. Puede estar almacenado en la instrucción, un registro o la memoria.

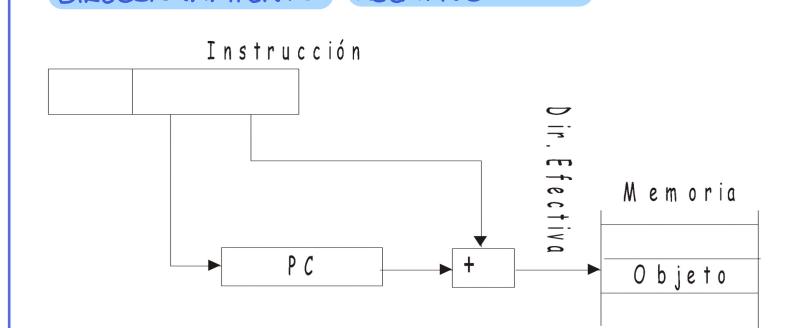
- **DIRECCIONAMIENTO INMEDIATO**: El objeto está contenido en la propia instrucción.
- **DIRECCIONAMIENTO DIRECTO**: El objeto no está contenido en la propia instrucción. La instrucción contiene la dirección donde está almacenado el objeto.
  - **ABSOLUTO**: Si la instrucción contiene la dirección completa del objeto.
    - Direccionamiento directo absoluto a registro: El objeto del direccionamiento está contenido en un registro. La instrucción contiene el registro que contiene el objeto del direccionamiento.
    - Direccionamiento directo absoluto a memoria: El objeto del direccionamiento está contenido en una dirección de memoria. La instrucción contiene la dirección completa de memoria que contiene el objeto del direccionamiento.
  - **RELATIVO**: El objeto del direccionamiento está contenido en una dirección de memoria, y la instrucción contiene la dirección del objeto de forma parcial. La instrucción contiene la dirección especificada en "partes". Todos los direccionamientos relativos lo son a memoria. Dependiendo de cómo se especifique la dirección:
    - Direccionamiento relativo a registro base: La dirección de memoria viene especificada en dos partes: registro base (registro de propósito específico o general que contiene una dirección a memoria) y desplazamiento (valor entero con signo). La dirección efectiva se calcula como Registro base + Desplazamiento.
    - Direccionamiento relativo a PC: Es un direccionamiento relativo a registro base en el que el registro base es el PC. El objeto de este direccionamiento suelen ser instrucciones. Permite alcanzar instrucciones "cercañas" a la que se está ejecutando.
    - Direccionamiento indirecto a registro: La instrucción contiene la especificación del registro que contiene la dirección de memoria donde está almacenado el objeto.
    - Direccionamiento indirecto a memoria: La instrucción contiene una dirección de memoria donde está contenida la dirección donde se almacena el objeto.



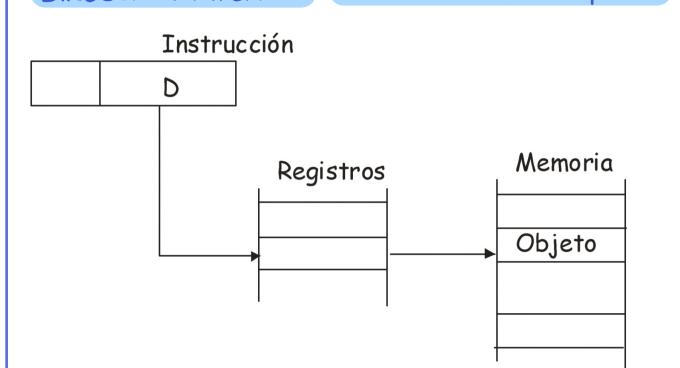
DIRECCIONAMIENTO RELATIVO A REGISTRO BASE



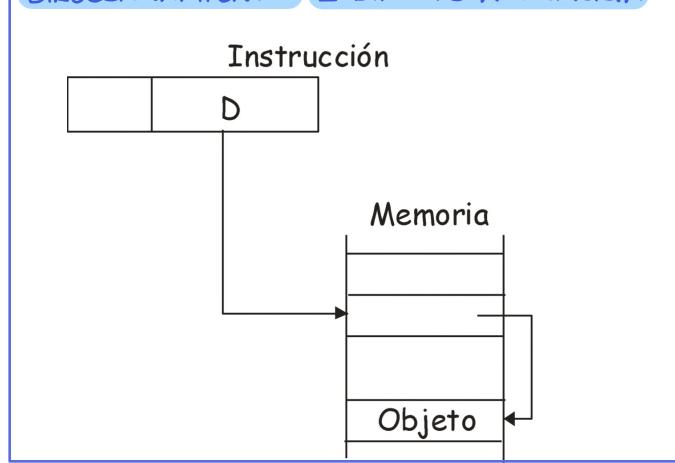
DIRECCIONAMIENTO RELATIVO A PC



DIRECCIONAMIENTO INDIRECTO A REGISTRO



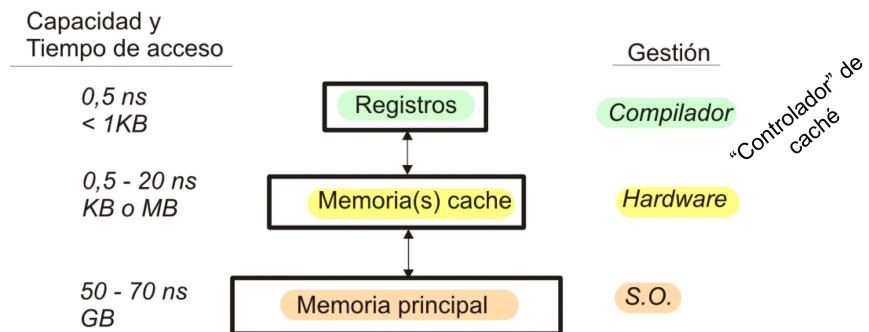
DIRECCIONAMIENTO INDIRECTO A MEMORIA



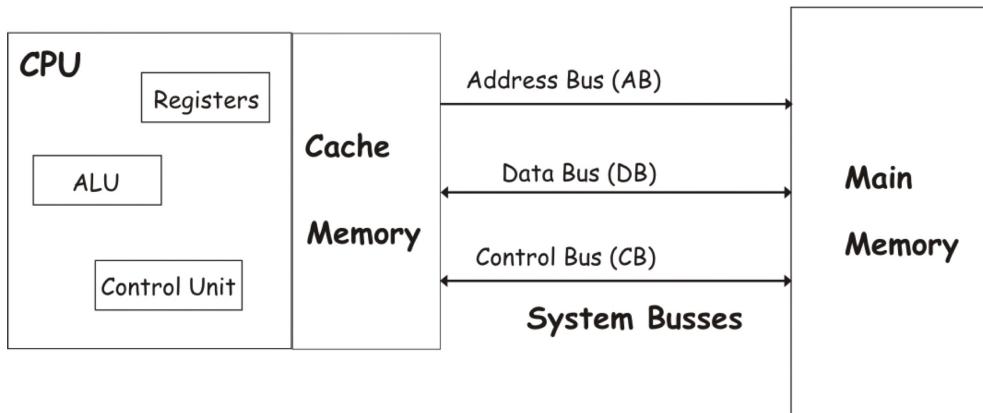
## 2.7. MEMORIA CACHÉ

Los procesadores han ido incrementando su velocidad mucho más que las memorias. Por tanto, las ejecuciones están restringidas por los tiempos de acceso de las memorias y no por los procesadores. Este problema se intenta resolver por la memoria caché. Es un tipo de memoria intermedia usada por la CPU, cuyo objetivo es mejorar su rendimiento. Se sitúa entre la CPU y la memoria principal.

Su uso reduce el tiempo de acceso a la memoria. Es más rápida y con un tamaño mucho más reducido que el de la memoria RAM, y tiene una velocidad similar o ligeramente inferior a la CPU con mayor capacidad que ésta.



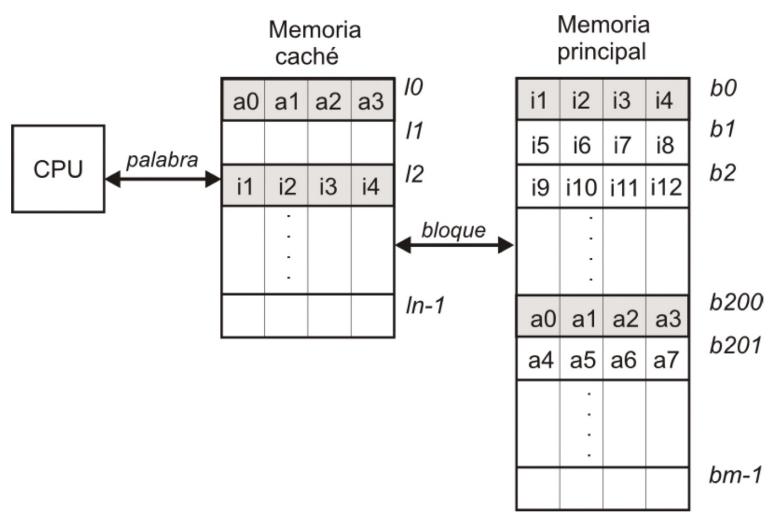
### ESQUEMA BÁSICO CON MEMORIA CACHÉ



### 2.7.1. FUNCIONAMIENTO

- La CPU accede a una posición de memoria (*fetch* o *ld/st*)
- Se comprueba si la información está guardada en memoria caché.
  - Si está, se sirve el acceso desde memoria cache (4-10 veces más rápido).
  - Si no está, se accede a la memoria principal, se transfiere la información desde memoria principal a memoria cache. La memoria cache transfiere la información a la CPU.

Cuando se pasa información a la memoria cache, se copia un conjunto de direcciones consecutivas: bloques de caché (unas decenas de bytes), de forma que, aunque el procesador le esté pidiendo sólo una, almacena las posibles siguientes, cercanas a la posición que le han pedido. Esto se hace porque los programas tienden a hacer referencias a datos e instrucciones “próximos” a los recientemente utilizados.



## 2.8. SISTEMA DE ENTRADA/SALIDA

Hay una gran diversidad de dispositivos periféricos con características muy diferentes:

- Modos de funcionamiento
- Formato y tamaño de los datos
- Velocidad de transferencia
- Tiempo de acceso

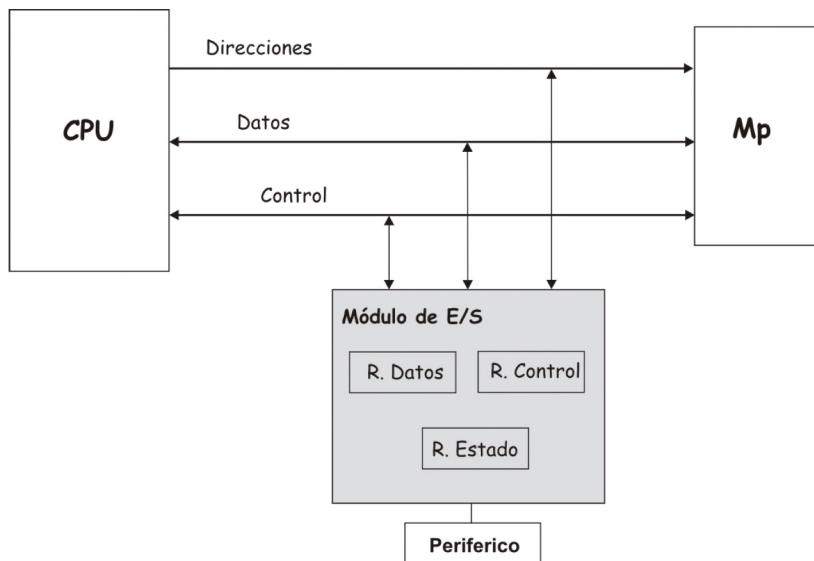
Por lo tanto, es necesario unificar la visión hardware de los periféricos.

### 2.8.1. MÓDULOS DE E/S

Ocultan las particularidades de cada periférico. La CPU dialoga con todos estos módulos con los mismos mecanismos.

Tienen dos interfaces: uno para dialogar con la CPU (estandarizado) y otro con el periférico (específico de cada periférico). Sus funciones son:

- Sincronización: control y temporización de la comunicación
- Comunicación con la CPU.
- Comunicación con el periférico.
- *Buffering*, por la diferencia de velocidades y tipos de datos.
- Control del periférico.



### 2.8.2. TÉCNICAS DE ENTRADA/SALIDA

Los periféricos son más lentos que la CPU y memoria. E/S y periférico son independientes y trabajan en paralelo.

Técnicas de E/S:

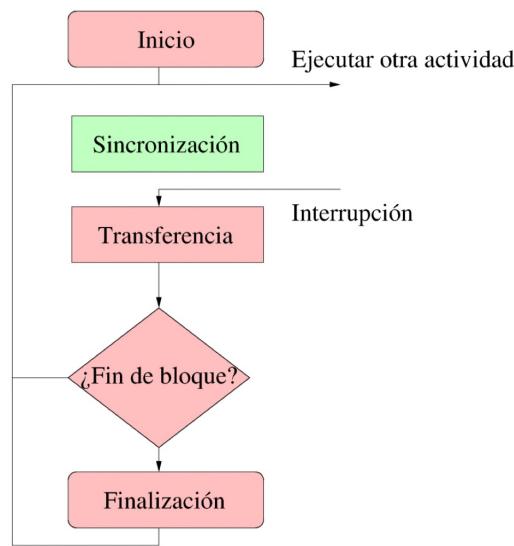
- **E/S programada:** Todas las fases las realiza la CPU mediante la ejecución de un programa. La sincronización se realiza ejecutando instrucciones en un bucle de espera. Iterativamente se lee el registro de estado del módulo E/S y se comprueba la condición de nuevo dato (*polling*). La

transferencia a/desde memoria se realiza ejecutando instrucciones in/out sobre el registro de datos e instrucciones ld/st sobre la dirección correspondiente de memoria.

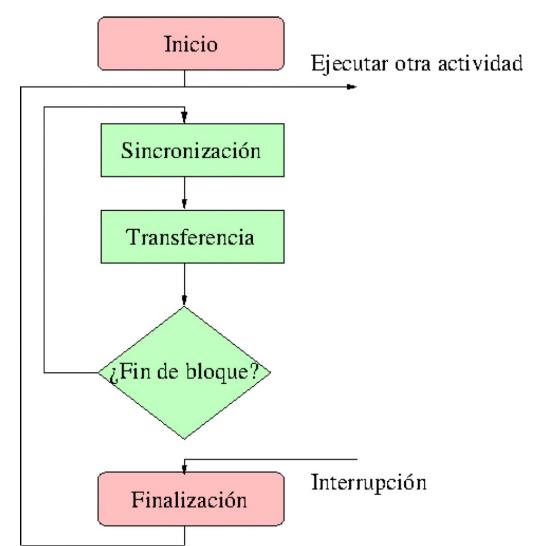
En la E/S programada, las operaciones se realizan por un programa que nunca abandona su flujo de ejecución. Los periféricos son “lentos” y se utiliza mucho tiempo de CPU en la sincronización (ej.: el tiempo medio de acceso de un disco duro es del orden de ms. Una CPU con un reloj de GHz podría ejecutar millones de instrucciones durante ese tiempo de acceso). Se podrían usar esas instrucciones para realizar otras actividades en un sistema concurrente. Esta técnica es válida para pequeños sistemas dedicados.

- **E/S por interrupciones.** La CPU no se encarga de la sincronización. El módulo avisa a la CPU cuando está listo para una nueva transferencia. Se ahorra mucho tiempo de CPU que se usa para ejecutar otras actividades.
- **E/S por DMA.** La CPU se encarga de iniciar la operación. El módulo de entrada/salida se encarga por hardware de la sincronización y transferencia y avisa cuando ha terminado mediante una interrupción. La CPU finaliza la operación. Hay una única interrupción por operación: se ahorra mucho tiempo de CPU con dispositivos de bloque.

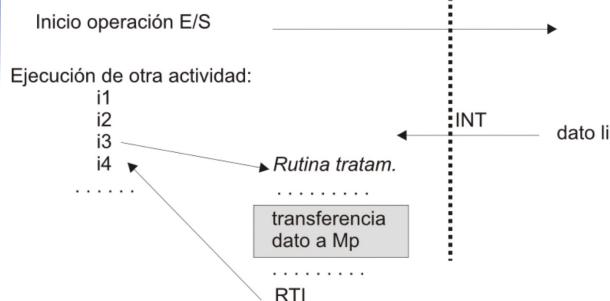
#### E/S MEDIANTE INTERRUPCIONES



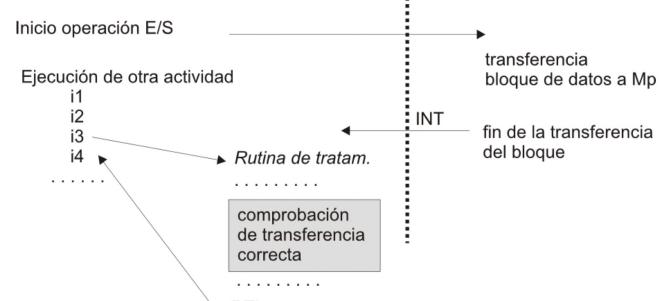
#### E/S MEDIANTE DMA



CPU



CPU



## **2.9. DISEÑO E IMPLEMENTACIÓN**

### **2.9.1. CODISEÑO SOFTWARE/HARDWARE**

El desarrollo “clásico” de sistemas empotrados consiste en desarrollar el software para un computador previamente seleccionado o diseñado. Actualmente, las plataformas de hardware configurable (ASIC, FPGA, ...) pueden integrar un gran número de componentes: procesadores, memorias, periféricos, componentes hardware a medida y componentes software escritos en un lenguaje de definición de hardware. De este modo, el software y el hardware no se diseñan por separado sino conjuntamente.