ESTRUCTURA Y TECNOLOGIA DE COMPUTADORES

INGENIERÍA EN INFORMATICA DE GESTION

UCM

ÍNDICE

CARACTERÍSTICAS DE LA ASIGNATURA

- **Área : Arquitectura y Tecnología de Computadores**
- **Titulación : Ingeniero en Informatica de Gestión**
- **Troncalidad : Estructura y Tecnología de C0omputadores**
- **Descriptores**
  - **Lenguaje máquina y ensamblador**
  - **Unidades funcionales**
  - **Procesador (instrucciones, formatos,secuenciamiento,modos de direccionamiento)**
  - **Memoria (jerarquía, organización, dispositivos y gestión)**
  - **Periferia (entrada/salida)**
  - **Perifericos: controladores y procesadores especializados**
  - **Control, microprogramación**
  - **Esquemas de funcionamiento**
- **Nº de créditos: 15**
  - **(9 Troncales +6 ampliados)**
  - **A estructura le corresponden 7,5**

OBJETIVOS DOCENTES

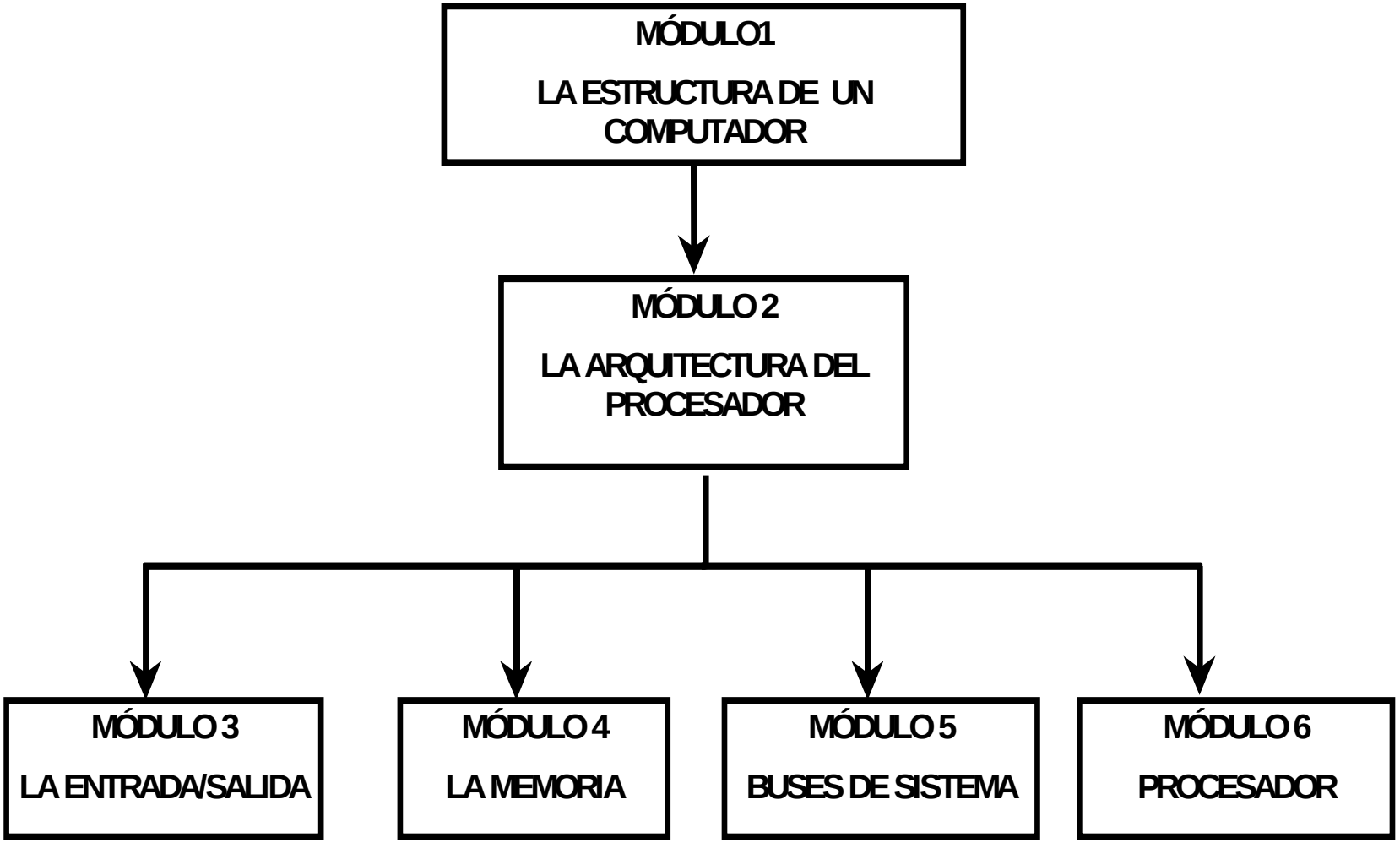
- **Objetivo general:**
  - comprensión de la organización, estructura, diseño y funcionamiento de un computador digital convencional y su interacción con el exterior.

- **Objetivos concretos**:
  - Fijar los conocimientos adquiridos con anterioridad
  - **Arquitectura** como punto de arranque para implementar la estructura
  - **Estructura** la suma de subsistemas mas interconexiones
    - » Estructuras internas de los subsistemas
    - » Formas de conexión de los subsistemas
  - Capacidad de diseñar sistemas completos y con cierto grado de complejidad

CRITERIOS DE ELABORACIÓN DEL PROGRAMA

- DIRECTRICES DEL CONSEJO DE UNIVERSIDADES
- OBJETIVOS DOCENTES FIJADOS
- EXPERIENCIAS DEL DEPARTAMENTO
- CONTENIDO DE LA BIBLIOGRAFÍA
- CONTEXTO DE LA ASIGNATURA EN EL PLAN DE DE ESTUDIOS DE LA INGENIERÍA EN INFORMATICA DE GESTIÓN

ORGANIZACIÓN DEL PROGRAMA



PROGRAMA

MODULO 1. LA ESTRUCTURA DE UN COMPUTADOR

Tema 1.1 Introducción a los computadores
Tema 1.2 Estructura básica de un computador

MODULO2. LA ARQUITECTURA DEL PROCESADOR

Tema 2.1 El repertorio de instrucciones
Tema 2.2 Parámetros de diseño del repertorio de instrucciones
Tema 2.3Influencia de los parámetros en el rendimiento /coste

MODULO 3 SUBSISTEMA DE ENTRADA/SALIDA

Tema 3.1 Técnicas de entrada/salida
Tema 3.2 Interfaces de entrada/salida programables
Tema 3.3 Dispositivos periféricos

MODULO 4 SUBSISTEMA DE MEMORIA

Tema 4.1 La jerarquía de memoria
Tema 4.2 La memoria cache
Tema 4.3 La memoria virtual

MODULO 5. BUSES DE SISTEMA

Tema 5.1 características y protocolos
Tema 5.2 jerarquía de buses y buses estandar

MODULO 6 EL PROCESADOR

Tema 6.1 El camino de datos
Tema 6.2 la ual
Tema 6.3 La unidad de control
Tema 6.4 la segmentación

BIBLIOGRAFÍA

- **COMPUTER ORGANIZATION AND ARCITECTURE. DESIGN FOR PERFORMANCE**
  - W.STALLING
  - PRENTICE HALL. 1996
  - 4º EDICIÓN.
- **COMPUTER ORGANIZATION & DESIGN:THE HARDWARE/SOFTWARE INTERFACE**
  - PATTERSON, HENNESSY
  - MORGAN KAUFMANN
  - 1994
- **COMPUTER ORGANIZATION**
  - V.C.HAMACHER, Z.G.VRANESIC, S.G. ZAKY
  - MC GRAW-HILL . 1996
  - 4º EDICIÓN

ESTRUCTURA DE COMPUTADORES

Modulo 1

FAMILIA DE COMPUTADORES

- Una misma arquitectura tiene diferentes estructuras
- El concepto surge en los 60 con la familia IBM 360
- Cada estructura tiene una relación rendimiento/coste
- Las características de la familia son:
  - Repertorio de I's similar o idéntico
  - Velocidad en incremento
  - Nº de puertos I/O en incremento
  - Tamaño de la memoria creciente
  - Coste creciente
- Compatibilidad
  - Un programa se puede ejecutar en varios modelos
  - La única diferencia es el tiempo de ejecución
  - Compatibilidad ascendente

ARQUITECTURA DE UN COMPUTADOR

- Atributos del computador que puede ver el programador de lenguaje máquina
- La interface entre el software de bajo nivel y el Hardware
- Describe el comportamiento funcional visto por el programador LM
- ¿QUE?
- Objetivo
  - Encontrar un lenguaje máquina que haga fácil
    - » La construcción del hw
    - » Del compilador
  - Maximizar el rendimiento
  - Minimiza el coste
- Atributos de arquitectura son:
  - Repertorio de instrucciones
  - Formato de las instrucciones
  - Códigos de operación
  - Modos de direccionamiento
  - Registros y posiciones de memoria que se pueden manipular directamente
  - Número de bits utilizados para representar diferentes tipos de datos
  - Mecanismos de entrada/salida

ESTRUCTURA DE COMPUTADORES

- **Implementan las especificaciones dadas en la arquitectura**
- **Estudia la organización interna**
- **Se compone de:**
  - **Unidades operacionales**
  - **Las redes de conexión**
- **Atributos de la estructura son transparentes al programador**
  - **Las señales de control**
  - **Interfaces entre el computador y los periféricos**
  - **La tecnología de memoria utilizada**
  - **El tipo de operadores aritméticos seleccionado**
- **¿COMO?**

ARQUITECTURA Y ESTRUCTURA

- **Decisión a arquitectura**
  - **Determinar si se va a disponer de una operación aritmética,**
- **Una decisión de estructura**
  - **Estudiar cómo se va a implementar dicha operación**
    - » **Si mediante un sistema secuencial o combinacional,**
    - » **Mediante una unidad especial, o en la UAL**
- **La decisión de diseño de la estructura se fundamenta en:**
  - **La velocidad de ejecución**
  - **En el tamaño hardware**
  - **Consumo de potencia**
- **La diferencia entre arquitectura y estructura aparece de manera clara en las familias de computadores,**
  - **Comparten la misma arquitectura pero tienen diferentes estructuras**
    - » **Los diferentes modelos de la familia**
      - ◆ **Tienen diferentes precios**
      - ◆ **Diferentes características de rendimiento**

NIVELES DE ESTUDIO DE UN COMPUTADOR

- **Estructural Bell y Newell**
  - **Los clasifica en función del elemento de construccion de cada nivel**
  - **Las primitivas de un nivel son los sistemas construidos en el nivel inferior**
  - **Nivel de Componentes**
    - » **Primitiva: Semconductores n y p**
    - » **Electronica física**
    - » **Bloque: Transistores resistencias**
  - **Nivel Electrónico**
    - » **Primitiva: Transistores n y p**
    - » **Corriente, tension, flujo**
    - » **Bloques: Puertas logicas biestables**
  - **Nivel Digital**
    - » **Primitiva: Puertas logicas, biestables**
    - » **Algebra de boole**
    - » **Bloques: SS.Combinacionales y secuenciales**
  - **Nivel RTL**
    - » **Primitivas:SS.Combinacionales y secuenciales**
    - » **Flujo de información**
    - » **Bloques: Memorias procesadores**
  - **Nivel PMS (Processor Memory Switch)**
    - » **Primitivas:Memorias, procesadores, Sistemas**
    - » **Permite describir con detalle cualquier sistema**
    - » **Bloques: sistemas computadores**

NIVELES DE ESTUDIO DE UN COMPUTADOR

- **Nivel de interpretación**
  - **Funcional**
  - **Propuesto por Levy**
  - **Cada nivel es un intérprete que recibe instrucciones de un tipo y actua de acuerdo a ellas:**
    - » **LAN**
    - » **Sistema operativo**
    - » **Lenguaje maquina**
    - » **Micro instrucciones**

- **Nivel conceptual de Blaauw**
  - **Arquitectura**
  - **Configuración**
  - **Realización**
  - **Arquitectura:**
    - » **Describe el comportamiento funcional visto por el programador LM**
    - » **¿QUE?**
  - **Configuración:**
    - » **Estudia la organización interna**
    - » **RTL y flujo de información**
    - » **¿COMO?**
  - **Realización:**
    - » **Circuitos lógicos**
    - » **Integración**
    - » **Empaquetamiento**
    - » **Conexión de módulos**
    - » **¿QUIEN?**

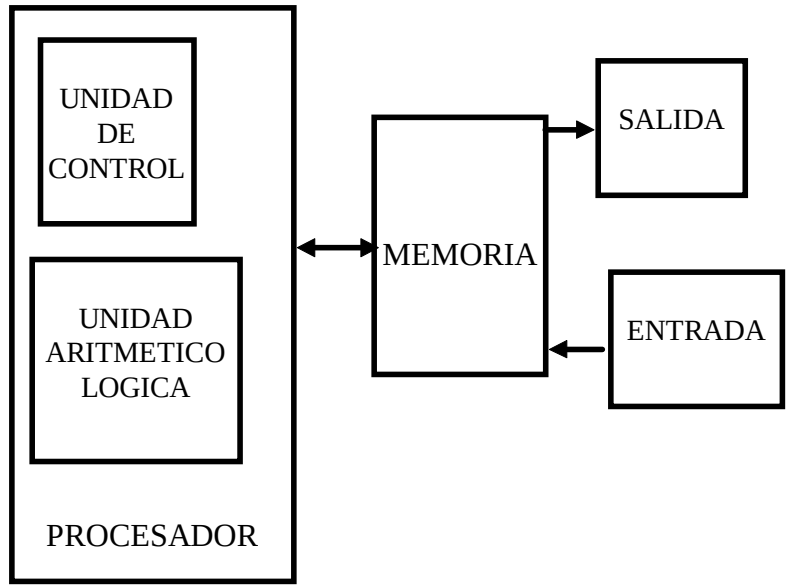
ESTRUCTURA BÁSICA DE UN COMPUTADOR

- Arquitectura de Von Neuman (1945)
- Ejecuta instrucciones almacenadas en la memoria principal
- Es una máquina secuencial
- Función:
  - Procesamiento de datos
  - Almacenamiento de datos
  - Transferencias de datos
  - Control
- Se descompone en los siguiente módulos:
  - Memoria principal
  - Unidad aritméticológica
  - Unidad de control
  - Unidad de entrada/salida
- Es importante recordar
  - Un modulo no tiene porque entrar en un solo chip
  - El retardo entrechips es muy importante
  - Cuanto más módulos incluidos en un chip mayor rendimiento

ESTRUCTURA BÁSICA DE UN COMPUTADOR

- **Memoria principal (MP)**
  - **Su Función es almacenar**
    - » **Instrucciones**
    - » **Datos**
  - **Se divide en palabras de memoria**
  - **Cada palabra en un número de bits**
  - **Una palabra se identifica con una dirección**
  - **DRAM**

- **Unidad Aritmético Lógica (ALU)**
  - **Realiza operaciones elementales:**
    - » **Suma Resta Or And**
  - **Los datos**
    - » **Vienen de la memoria principal**
    - » **Pueden almacenarsen temporalmente en Registros**



ESTRUCTURA BÁSICA DE UN COMPUTADOR

- **Unidad de control**
  - Lee secuencialmente las instrucciones de la MP
  - Genera las señales que controlan el computador
  - Contador de programa es un puntero a la siguiente instrucción que ayuda a la ejecución secuencial

- **Unidad central de proceso (CPU)**
  - Unidad control + registros + UAL
  - también llamada procesador
  - el bloque que ejecuta las instrucciones
  - Para formar el computador hay que añadir el resto de los módulos

- **Unidad de entrada/salida**
  - Transferencias de información entre
    - » El computador
    - » Los sistemas periféricos
      - impresoras
      - diskettes
      - terminales

- **Buses**
  - caminos de conexión entre todos los elementos
  - diversos niveles de buses

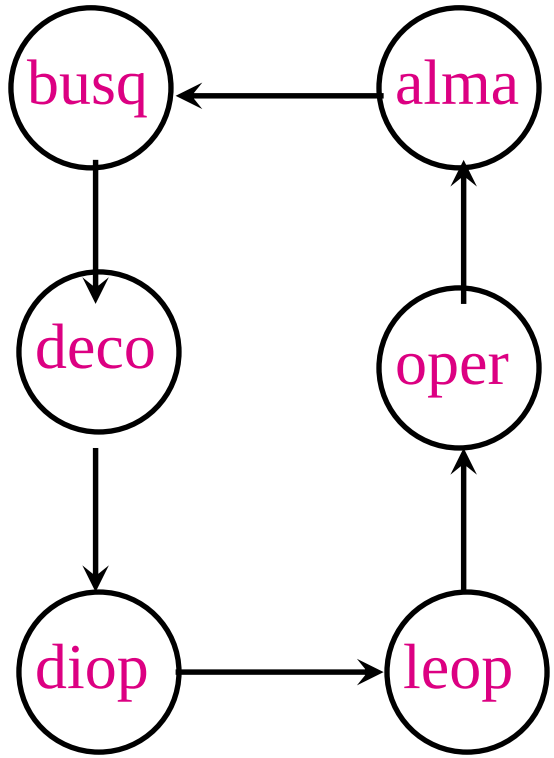
- **Camino de datos**
  - UAL
  - Los registros generales
  - Registros particulares
  - Buses

FASES DE EJECUCIÓN DE UNA INSTRUCCIÓN MÁQUINA

- La función de un computador es la ejecución de programas
- El programa esta compuesto por
  - Instrucciones (I's)
  - Datos almacenados en memoria
- La CPU ejecuta las I's del programa
- Una instrucción se descompone en fases
- Fase de búsqueda
  - Fase común a todas las I's
  - La lectura de la I de una posición de MP
  - Se accede a la posición RI:= M[Pc]
- Fase de ejecución
  - Se descompone en varios pasos que dependen de la I
  - La CPU debe
    - » Interpretar la I
    - » Llevar a cabo la la acción requerida
  - Esta acción puede ser de cuatro tipos diferentes:
    - » Transferencia de datos CPU - MP
    - » Transferencia de datos CPU E/S
    - » Procesamiento de datos
    - » Control.- Alteración de las secuencias de ejecución

FASES DE EJECUCIÓN DE UNA INSTRUCCIÓN MÁQUINA

- **El ciclo de instrucción básico**
  - **búsqueda de la instrucción (busq)**
  - **Descodificación de la instrucción (deco)**
  - **Calculo de las direcciones de los operandos(diop)**
  - **Lectura de los operandos leop)**
  - **Operación con los datos (oper)**
  - **Almacenamiento de operando (alma)**



CLASIFICACIÓN DE LOS COMPUTADORES

- **Hennessy y Patterson basan la clasificación en el precio**

- **Microcomputadores**
  - Pequeños computadores personales
  - Estaciones de trabajo
  - Unos miles de dólares

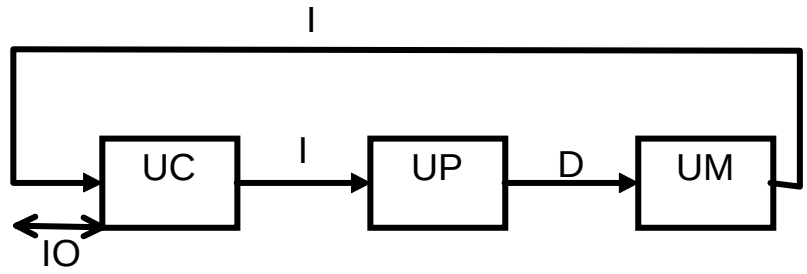
- **Minicomputadores**
  - Tamaño medio
  - Mas de 50.000 dolares

- **Mainframe:**
  - Mas de medio millón de dólares
  - Propósito general de altas prestaciones
  - Se utiliza para tareas de gestión
  - Comercial
  - Soporte para grandes bases de datos
  - Tratamiento de transacciones
  - Aritmética decimal
  - Soporta más terminales y discos que el minicomputador

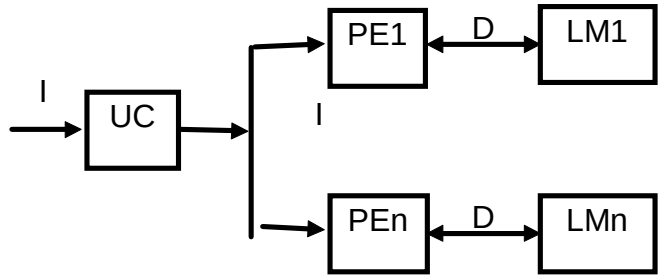
- **Supercomputador:**
  - Mas de un millón de dólares
  - Aritmética de punto flotante
  - Mas caros
  - Aplicaciones científicas
  - Mas alto rendimiento

CLASIFICACIÓN COMPUTADORES
CLASIFICACIÓN FLYN

- **SISD**
  - **Solo una instrucción solo un dato**
  - **Es típico el Von Neumann**
  - **La CPU que**
    - » **Ejecuta una instrucción cada vez**
    - » **Busca o almacena un dato cada vez**
  - **Un procesador**
  - **Solo una memoria**



- **SIMD**
  - **Solo una instrucción múltiples datos**
  - **Una unidad de control**
  - **Varias unidades de proceso trabajando con una misma instrucción sobre datos diferentes**
  - **Memorias locales**

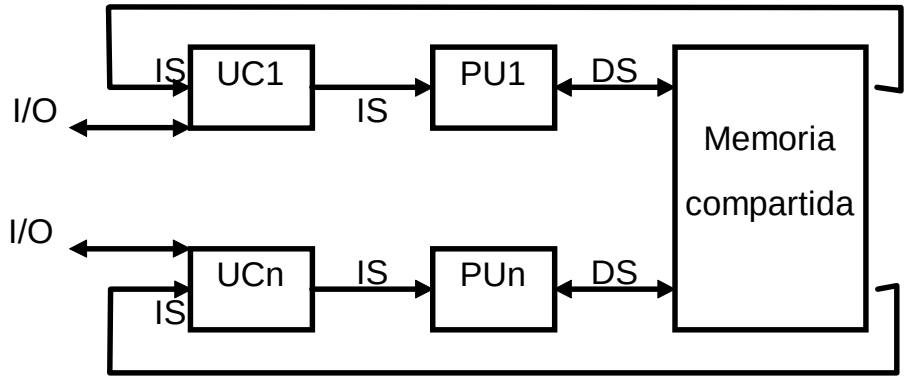


Arquitectura SIMD con memoria distribuida

CLASIFICACIÓN COMPUTADORES
CLASIFICACIÓN DE FLYN

- **MISD**
  - **Múltiples instrucciones y solo un dato**
  - **Diferentes programas sobre el mismo datos**
  - **varios procesadores con una memoria compartida**
  - **Conocidos como arrays sistolicos**

- **MIMD**
  - **Múltiples instrucciones múltiples datos**
  - **Distribuyen el procesamiento entre un nº de procesadores independientes**
  - **Los procesador opera en paralelo**
  - **Cada procesador ejecuta su propio programa**
  - **si la memoria compartida multiprocesador**
  - **si la memoria local multicomputador**



Arquitectura MIMD con memoria compartida

HISTORIA DE LOS COMPUTADORES

- Generaciones basadas en la tecnología
- Cada nueva generación se caracteriza por
  - Una mayor velocidad
  - Mayor densidad de memoria
  - Menor tamaño
- Efecto
  - Modificaciones en Arquitecturas y Estructuras
  - La mayor densidad permite la utilización de estructuras y módulos funcionales que antes, ya fuera por su tamaño, velocidad o complejidad de fabricación, se desechaban
- 5 generaciones:
  - Válvulas de vacio
  - Transistor
  - Circuitos integrado de baja densidad
  - Circuitos VLSI
  - Microprocesadores

HISTORIA DE LOS COMPUTADORES
Primera generación (1943-1954)

- Válvulas de vacío
- ABC (1942)
  - Atanasoff-Berry Computer
  - 1º computador electrónico de propósito específico
- ENIAC (Electronic Numerical Integrator And Computer
  - 1943-1946-1955
  - 1º computador electrónico de propósito general
  - Mauchly y Eckert Univ. Pennsylvania
  - 30 toneladas
  - tamaño :2 ordenes de magnitud mayor que las actuales
  - rapidez: 4 ordenes de magnitud mas lenta
  - 140Kwatios
  - Programación hw
- IAS (institute for advanced Studies de princeton)
  - 1ª arquitectura Von Neumman
  - 1946-1952
- Primeros computadores comerciales
  - Leo (Lyon electric office)
    - » 1951 por Wilkes
  - UNIVAC I y II
    - » Mauchly Eckert
  - IBM 701
    - » 1953
    - » 709 19 unidades
- Caracteristicas generales:
  - Memorias de ferrita
  - Cintas magneticas
  - Gran disipación de calor
  - Enorme refrigeración
  - Gran tamaño
  - Escasa capacidad
  - Dificil mantenimiento
  - Monoprogramación
  - Sin sistema operativo
  - Perifericos controlados por el procesador

HISTORIA DE LOS COMPUTADORES
Segunda Generación (1954-1963)

- Transistores discretos
  - Laboratorios Bell (1947)
  - Mas pequeño
  - Mas barato
  - menos disipación
  - TRADIC
    - » 1º computador digital con transistoresde la casa Bell
- Mayor complejidad de los módulos
- Aparece la empresa DEC
  - Digital Equipment Corporation
  - PDP-1
    - » 1957
    - » Comienza el desarrollo de minicomputadores
    - » se vendieron 50 unidades
- Wilkes propone la microprogramación (1951)
- ATLAS
  - 1962
  - Memoria virtual
  - Universidad de Manchester
  - Disco magnetico como memoria secundario
- IBM7094
  - 1952 serie 7000
  - Canal de datos
    - » Modulo independiente de IO
    - » con procesador propio
    - » repertorio de instrucciones
  - sistema de I/O multiplexado
- Lenguajes de Alto Nivel (Fortran)
- Compiladores
- Software de sistema

HISTORIA DE LOS COMPUTADORES
Tercera Generación(1964-1971)

- **Circuitos integrados (SSI/MSI)**
  - Módulos más próximos
  - Computadores más pequeños
  - Menor consumo
  - Conexiones más fiables
  - Disminuyo el precio

- **Familia IBM System/360**
  - 1964
  - Concepto de arquitectura y estructura
  - Multiprogramación
  - Prioridades de interrupción
  - DMA
  - Cache
  - Microprogramación

- **1964 Control Data 6600 diseñado por Cray**
  - Segmentación
  - Rendimeinto un orden del magnitud superior al IBM7094
  - 1º supercomputador

- **DEC PDP8**
  - Primer minicomputador
  - Reducido tamaño
  - Reducido precio
  - Utiliza un bus en lugar de la estructura centralizada de IBM
  - Gran flexibilidad

- **Surgen sistemas operativos potentes**
- **Gran desarrollo de lenguajes de alto nivel**

HISTORIA DE LOS COMPUTADORES
Cuarta Generación (1972-1987)

- VLSI
- microprocesadores
  - INTEL 4004
    - » 1971
    - » Propósito específico
  - INTEL 8080
    - » 1974
    - » Primero de prósito general
- se maduran los conceptos surgidos en la generación anterior
  - segmentación
  - paralelismo
  - cache
- computador personal
  - apple ii
  - basado en el microprocesador 6502
- estación de trabajo
- arquitectura RISC
- computadores vectoriales tecnología ECL
- memorias semiconductoras
  - Fairchild
    - » 1970
    - » 256 bits
    - » tamaño de un corazon de ferrita
    - » lectura no destructiva
    - » rápida
    - » elevado coste por bit
  - 1974 precio inferior al de la ferrita

HISTORIA DE LOS COMPUTADORES
Quinta Generación (1988-)

- Microrocesador como elemento de diseño
  - Desde portátiles a supercomputadores
  - Segmentación y paralelización en el microprocesador
- Busqueda del paralelismo a toda costa
  - MIMD:
    - » Multiprocesadores
    - » Multicomputadores
  - Antes con microprocesadores específicos hoy con microprocesadores estándares
    - » mas sencillo y rápido
    - » cray3x, ibmsp2
- Vectoriales en tecnología mos
- Problemas actuales
- Predicciones de Moore
  - Nueva generación cada 3 años
    - » Con memorias 4 veces más densas
    - » Procesadores 4 veces más rápidos
  - Cuellos de botella Mp-CPU
  - Manejo de entrada/salida

BASES DEL DESARROLLO DE LOS COMPUTADORES

- **El avance tecnológico**
  - **Aumento de la densidad**
    - » **Disminución de los tiempos de ciclo**
    - » **Disminución de los consumos de potencia**
    - » **Posibilita el estudio de arquitecturas que se debieron abandonar por necesitar mayor número de transistores de los que se podían integrar en un chip**
    - » **disminución de costes**

- **El avance arquitectónico**
  - **La necesidad de mayor número de transistores en un ci impulsa la investigación tecnológica**

- **El avance en las herramientas de diseño automático**
  - **No sería posible el diseño de circuitos con varios millones de transistores si no fuera por la ocultación de los detalles de bajo nivel**

EL RENDIMIENTO

TEMA 2

INTRODUCCIÓN

- El rendimiento del hw es clave para la efectividad de un sistema completo hw/sw
- Determinar el rendimiento de un sistema es difícil
- Definición de rendimiento dependerá del sistema y de la aplicación
- Importante en la elección del sistema
  - Gato por liebre
- Importante en el diseño
  - Comprender como la elección de una estructura afecta al rendimiento es vital para la toma de decisiones
- Ejemplos:
  - Rendimiento = tiempo de respuesta del computador
    - » Dos estaciones de trabajo independientes
    - » Manejadas por un único usuario
  - Rendimiento = número de tareas acabadas en un día (productividad)
    - » Dos grandes máquinas multiusuarios de tiempo compartido
    - » Esto va en detrimento del tiempo de respuesta

MEDIDAS DEL RENDIMIENTO

- ¿Computador más rápido?
  - Mismo trabajo en menos tiempo
- Tiempo de respuesta
  - Tde reloj , T transcurrido
  - T total para completar una tarea
  - Incluye:
    - » Accesos al disco
    - » Accesos a memoria
    - » Actividades de entrada /salida
    - » Gastos de sistema operativo
  - Inconveniente
    - » Computadores son de T compartido
    - » La CPU trabaja en varios programas
  - Solución
    - » Distinguir entre
      - Tiempo de respuesta
      - Tiempo de ejecución de la CPU

- Tiempo de ejecución de la CPU
  - llamado tiempo de CPU
  - T que la CPU emplea para realizar una tarea
  - No incluye
    - » Tiempos de E/S
    - » Tiempo de ejecución de otros programas
  - Este tiempo se descompone en:
    - » Tiempo CPU del usuario
    - » Tiempo CPU del sistema
      - no se suele considerar por la complejidad de las medidas

OTRAS MÉTRICAS UTILIZADAS HABITUALMENTE

- **Tiempo de Ciclo (Tciclo)**
  - **Tciclo = la duración del ciclo**
  - **Frecuencia (F=1/Tc)**
  - **Expresión que las relaciona con las anteriores son:**
    - **Tcpu= Nºciclos ·Tciclo**
    - **Tcpu =Nºciclos /F**
      - **nºciclos = Nº de ciclos de la CPU en ejecutar un programa**
  - **Mejorar el rendimiento:**
    - **Reduciendo la Tciclo**
    - **Reduciendo el Nºciclos**

- **Ciclos de reloj por instrucción (CPI)**
  - **Nº <u>medio</u> de ciclos que una I necesita para ejecutarse**
  - **Sirve para comparar dos implementaciones de una misma arquitectura**
    - **El recuento de Is es el mismo.**
  - **Tcpu=NºIs·CPI·Tciclo**
  - **Se obtiene por simulación detallada de una implementación**

METRICAS DEL RENDIMIENTO
MIPS (Millones de Instrucciones por Segundo)

- $n^o I/(n^o ciclos \cdot Tciclo)10^6=(F/CPI)\cdot 10^6$
- Mide la frecuencia de ejecución de I's
- $Tcpu= n^o I/MIPS \cdot 10^6$
- Ventajas
  - Fácil de comprender
  - Máquinas mas rápidas mayores MIPS, esto coincide con la intuición
- Problemas
  - Depende de la máquina
    - » no se pueden comparar computadores con diferentes RI ya que difieren los recuentos de I's para el mismo programa
  - Depende del compilador
    - » se modifica el número de I en que se convierte cada instruccion LAN
  - Depende del programa
- Con la aparicion de RISC pierde utilidad-->RISC siempre más I que los CISC
- MIPS relativos por comparación con una máquina concreta
  - VAX 11/780
- MIPS de pico o MIPS máximo
  - gato por liebre

METRICAS DEL RENDIMIENTO

MFLOPS (Millones de Operaciones en Punto flotante por segundo)

- Siendo OP el nº de operaciones en punto flotante de un programa
  - » **(OP / Tcpu)· 10^6**
- Se basa en las OP y no en las I's
- Util para comparar diferentes maquinas
  - **El mismo programa corriendo en diferentes máquinas puede tener un nº diferente de Is pero siempre tendrá el mismo OP**
- Depende mucho del programa
  - **Diferentes programas tienen diferente OP**
- Problemas
  - **OP no es consistente entre máquinas**
    - » **Aunque el OP supuestro sea el OP real difiere**
      - **Una multiplicación se implementa mediante sumas**
  - **No todas las OP se implentan igual**
    - » **Diferente rapidez (secuencial o Combinacional)**
- Mflops normalizados- tiene en cuenta la complejidad de la OP
  - **P.E. una division equivale a cuatro sumas**

PROGRAMAS PARA EVALUAR EL RENDIMIENTO

- **El usuario que corre los programas en un computador, los ejecutara en el nuevo computador para evaluar su rendimiento**

- **Benchmark**
  - **Programas escogidos para medir el rendimiento de una maquina.**

- **SPEC**
  - **Compañía formada por IBM, DEC, INTEL hewlwtt etc**
  - **Objetivo:**
    - » **Desarrollar pruebas normalizadas para evaluar la potencia de los computadores**

- **SPECint**
  - **Cálculos con enteros**
  - **Minimizar funiones lógicas**
  - **Traducir ecuaciones booleanas a tablas de verdad**
  - **Cálculo de valores en una hoja de cálculo**

- **SPECfp**
  - **Cálculo en coma fltante**
  - **Simulacioes de circuitos análógicos**
  - **Cálculo de integrales derivativas**
  - **Resolucion de redes neuronales**

Como máquina de referencia la VAX11/780

PROGRAMAS PARA EVALUAR EL RENDIMIENTO

- **Hoy en día se sabe que los mejores benchmark son aplicaciones reales, que pueden ser**
  - **Aplicaciones que el usuario emplea regularmente**
  - **Aplicaciones típicas**

- **La utilización de programas reales como banco de prueba hace difícil que el diseñador pueda utilizar trucos para acelerar el rendimiento.**
  - **para programas de prueba específicos, se diseñan compiladores para acelerar los segmentos de código en que el esfuerzo Computacional era mayor de este modo se obtienen unos resultados mejores de los reales.**

RESUMEN DE MEDIDAS

- Un cuadro de rendimientos da mayor información
- El usuario suele preferir una sola cifra que le ayude a comparar dos máquinas
- Sean dos máquinas A y B y dos programas el programa 1 y el 2
- Rendimiento relativo :TtotalA/TtotalB
  - **B=1001/110=9,1 veces más rápido que B**
- Media aritmética
  - **Suma de todos los tiempos partido por el número total de procesos**
  - **A menor media aritmética mejor rendimiento**
  - **Supone que que todos los trabajos se ejecutan el mismo número de veces**
  - **Si se desea tener en cuenta que algunos programas se usan más que otros se debe realizar una media aritmética ponderada**
    - **P.E. si el programa 1 supusiera el 20% de la carga de trabajo y el 2 el 80% los factores de peso sería 0,2 y 0,8**

	computador A	Computador B
programa 1	1sg	10sg
programa 2	1000sg	100sg
tiempo total	1001sg	110sg

LEY DE AMDAHL

- **El posible aumento de rendimiento de una máquina para una mejora determinada esta limitada por el uso que se de a la característica mejorada**
  - **Por ejemplo de nada sirve mejorar los multiplicadores en coma flotante si luego solo lo utiliza el 0,001% de las instrucciones**

- **Es mejor introducir pequeñas mejoras en elementos que se usan mucho que introducir grandes mejoras en elementos que se usan poco**

- **HACER EL CASO COMUN RÁPIDO**

- **Además conviene recordar que generalmente el caso común suele ser más simple y por lo tanto es más fácil de mejorar**

EL REPERTORIO DE INSTRUCCIONES

TEMA 1

INTRODUCCIÓN

- frontera en la que el diseñador de computadoras y el programador de computadora pueden ver la misma máquina
- Da las especificaciones funcionales de la CPU
- El que puede interpretar y ejecutar directamente el computador.
- Se compone de un conjunto de instrucciones máquina, cada una de las cuales realiza una acción específica y sencilla.
- A este conjunto se le llama juego o repertorio de instrucciones
- El lenguaje debe cumplir los siguientes objetivos:
  - Hw sencillo
  - Compilador sencillo
  - Maximizar el rendimiento
  - Se minimiza el coste

PROPIEDADES DE LAS INSTRUCCIONES MÁQUINA

- **Cada instrucción realizan una función única y sencilla:**
  - **Esto hace la descodificación sencilla**
- **Tiene un número fijo de operandos, con una representación determinada**
  - **Con esto se consigue el hw sea más sencillo**
- **Codificación sistemática**
  - **Para facilitar la descodificación**
- **Autocontenidas e independientes:**
  - **Contienen toda la información para ejecutarse**
  - **No necesitan información de otras instrucciones**
  - **Su interpretación no depende de la posición que ocupan en la memoria.**
- **Información que deben contener**
  - **Operación**
  - **Operandos**
  - **Destino resultado**
  - **Ubicación de la siguiente instrucción**

DEFINICIÓN DEL REPERTORIO DE INSTRUCCIONES

- **Para definir un repertorio Hay que detallar**
  - **Operaciones a realizar**
  - **Representación de los datos**
  - **Modos de direccionamiento**
  - **Formatos de las Instrucciones**
- **Modo de direccionamiento:**
  - **Mecanismos por los que se puede especificar un operando o la ubicación de una operación**
- **Formato**
  - **Indica como se codifica y distribuye la información en la instrucción**
- **La selección de un conjunto de instrucciones demanda un equilibrio entre**
  - **La complejidad del repertorio**
  - **El número de ciclos necesarios para una instrucción**
  - **La velocidad del reloj**

DEFINICIÓN DEL REPERTORIO DE INSTRUCCIONES

- **Existen cuatro principios que deben guiar al diseñador de repertorio de instrucciones**
  - **Cuanto más pequeño sea el HW más rápido será**
  - **La simplicidad de las instrucciones favorece la regularidad del hardware**
  - **Hacer rápido el caso común**
  - **Los diseños siempre demandan compromisos entre diversos factores**

- **El repertorio de instrucciones deber ser completo y eficaz.**
  - **Completo:**
    - » **Se pueda calcular en tiempo finito cualquier tarea computable**
  - **Eficaz:**
    - » **Velocidad de cálculo sin exigir a cambio una complicación excesiva de la unidad de control y de la unidad aritmética**

- **La selección del repertorio de instrucciones de un computador es uno de los puntos críticos de diseño.**

CISC

- **Objetivo**
  - simplificar el diseño de los compiladores
  - reducir el nº de operaciones ejecutadas por un programa.
- **Implementan directamente mediante IM Is del LAN**
- **Estos repertorios son muy grandes**
- **Enorme diversidad de**
  - Modos de direccionamiento
  - Tipos numéricos
  - Tipos de operando etc
- **Desventajas**
  - HW complejo lo que puede hacer su ejecución más lenta
  - Esta lentitud puede ser consecuencia
    - » Ciclos de reloj mas largos
    - » Necesidad de utilizar más ciclos de reloj

RISC

- **De estudios se ha llegado a las siguientes conclusiones**
  - **Un conjunto de I sencillas se ejecutan un número elevado de veces**
  - **Las I complejas prácticamente no se ejecutan**
  - **80 % de las referencias a datos escalares locales**
- **Consecuencia**
  - **Se complica el HW para I que prácticamente no se ejecutan**
  - **esta complejidad afecta a las I que más se ejecutan.**
- **El objetivo es reducir:**
  - **El número de formatos**
  - **El número de instrucciones**
  - **Modos de direccionamiento.**
- **Efecto**
  - **Se simplifica el Hardware y Se pueden aplicar técnicas como la segmentación**

RISC

- **Repertorio de instrucciones sencillo**
  - **Formatos sencillos**
  - **Longitudes de instrucción fija (una palabra)**
  - **Modos de direccionamiento sencillos(generalmente a registros)**
- **Grandes bancos de registros**
  - **Favorecen las operaciones registro a registro**
  - **Que mejoran los tiempos de acceso**
- **Compiladores que optimizan el uso de los registros**
- **Segmentación**
- **Unidad de control cableada**
  - **una instrucción por ciclo. Las señales se mueven por el cauce**

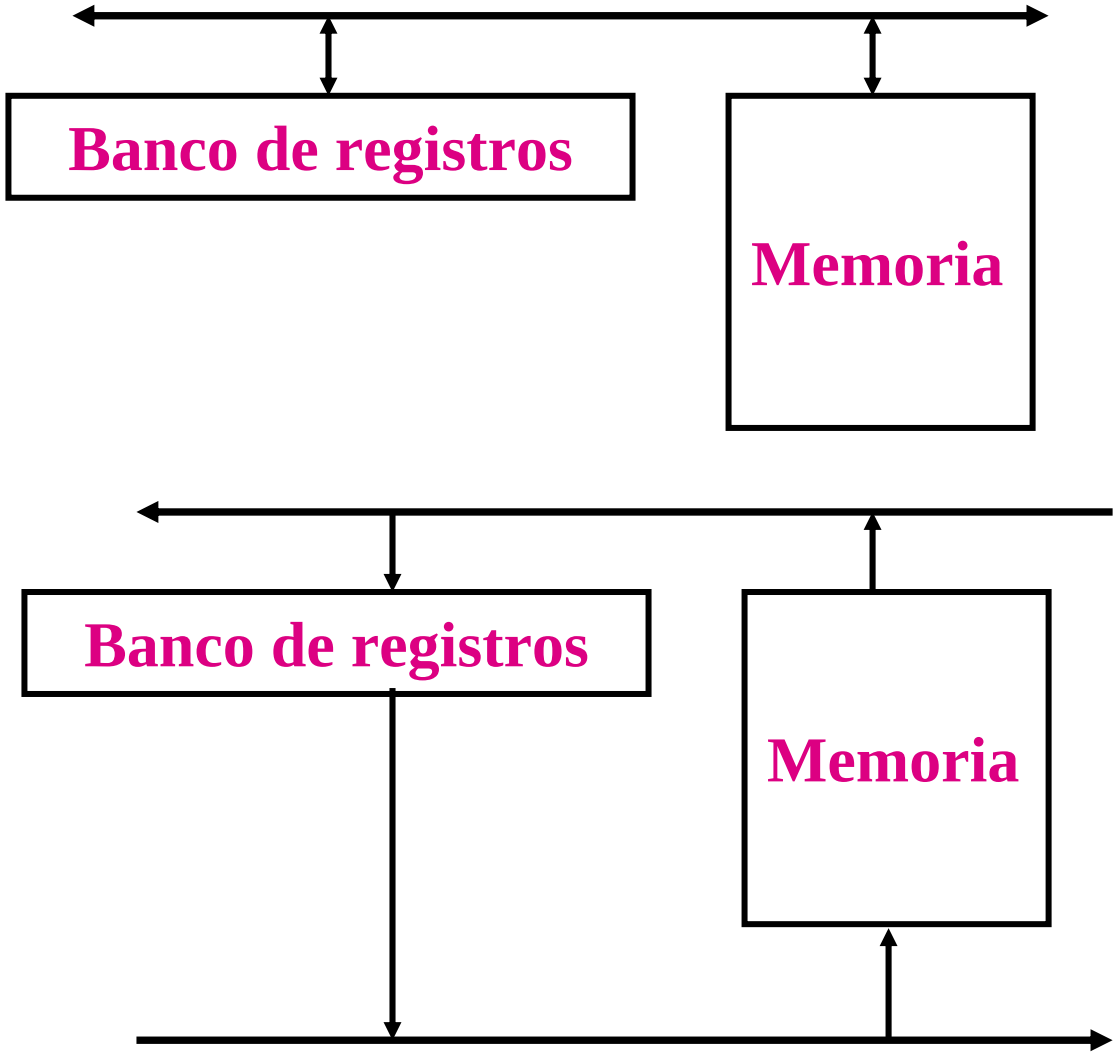
TIPOS DE INSTRUCCIONES

- **Instrucciones más frecuentes:**
  - **Movimiento de datos**
  - **Modificación de secuencia**
  - **Aritméticas**
  - **Comparación**
  - **Lógicas**
  - **Desplazamiento**
  - **De entrada/salida**

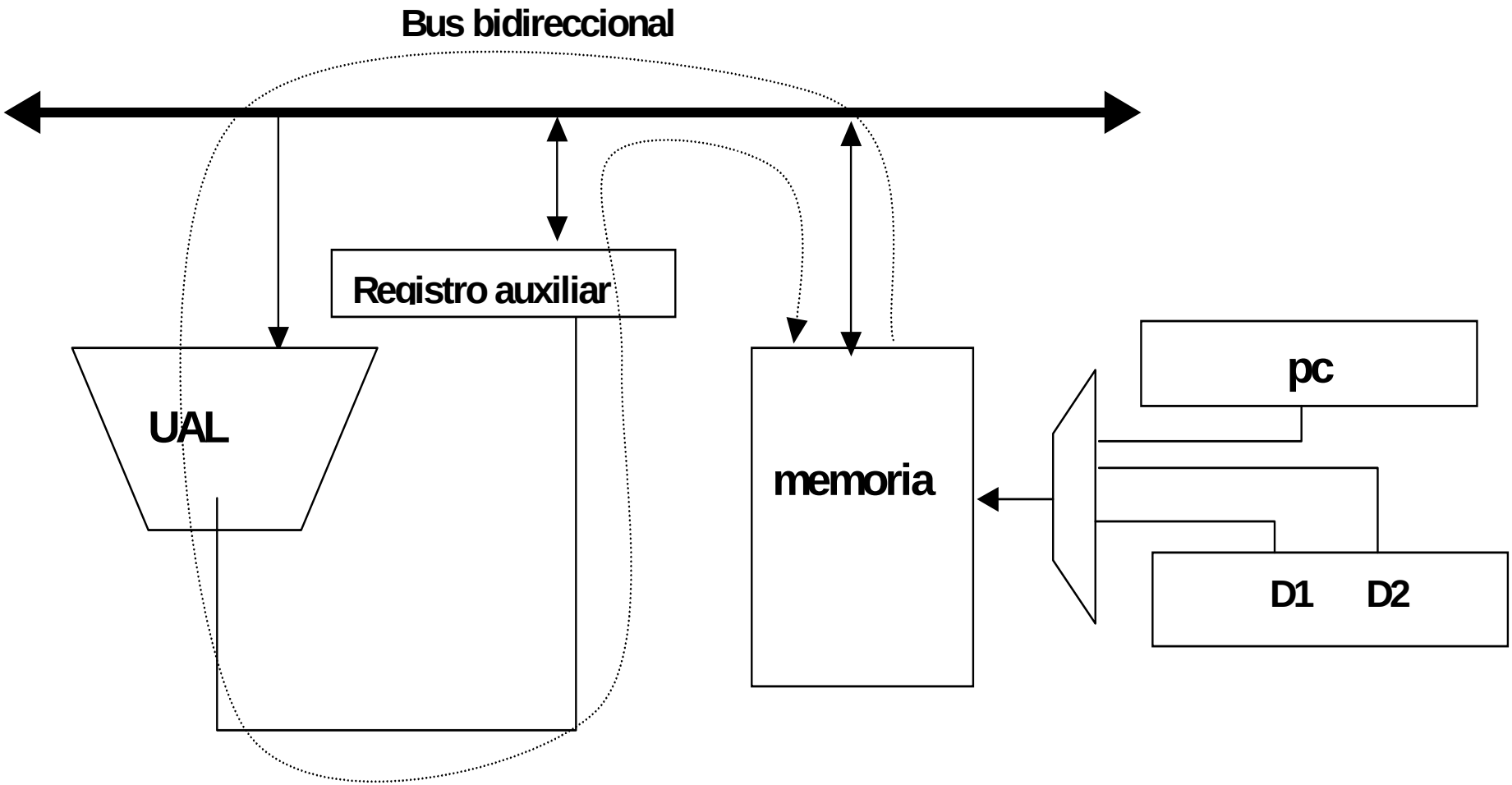
TIPOS DE INSTRUCCIONES
Movimiento de Datos

- La mas importante del repertorio
- Las mas simples
- Cargan en el operando destino la información contenida en el operando origen, quedando éste último sin modificar
- Debe especificar Destino y origen :
  - Pueden ser registros o posiciones de la memoria
  - Pueden ser de 8, 16, 32 …
  - Esta información puede ir en:
    - » opcode (ibm s/370) o en
    - » operando (VAX)
- No modifican los biestables de estado
- move fuente, destino

TIPOS DE INSTRUCCIONES
Movimiento de Datos

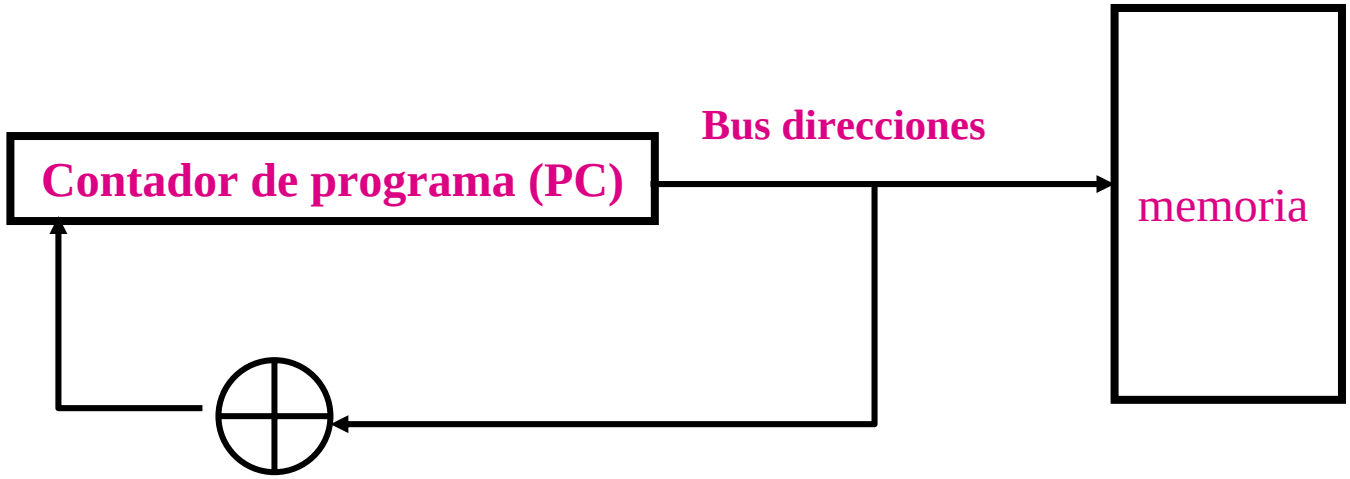


TIPOS DE INSTRUCCIONES
Movimiento de Datos



INSTRUCCIONES DE BIFURCACIÓN

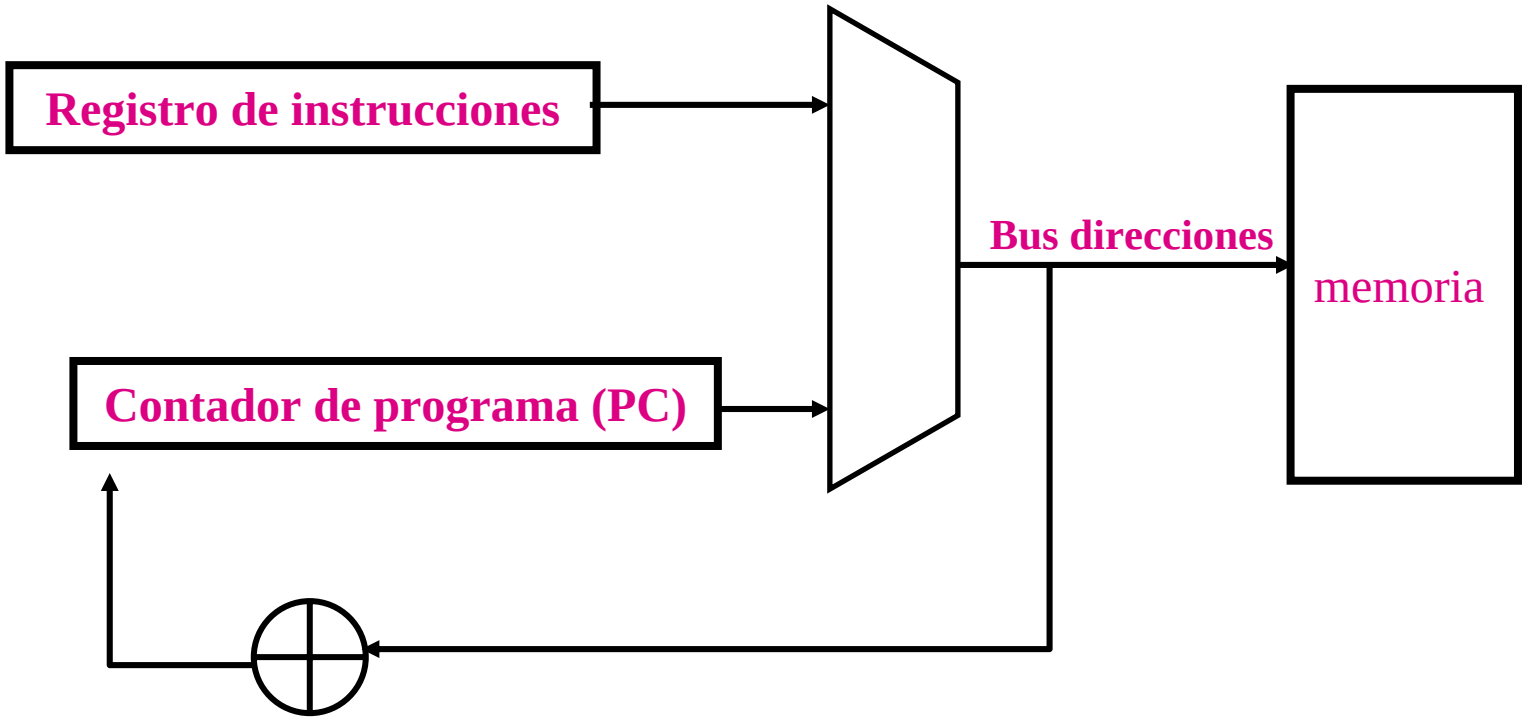
- Toma de decisiones
- Alteran la secuencia normal de ejecución del programa
- El control de la secuencialidad lo lleva el PC
  - PC := PC+1 Si la Instrucción ocupa una palabra
  - PC PC+ Z Si la Instrucción tienen un tamaño de Z palabras



INSTRUCCIONES DE BIFURCACIÓN

- **La bifurcación**
  - **Cargar el PC con la dirección X a la que se desea saltar**
  - **PC :=X**
- **Las Bifurcaciones pueden ser:**
  - **Condicionales**
  - **Incondicionales**
  - **Con retorno**
- **Bifurcaciones incondicionales**
  - **Siempre saltan**
  - **Son las equivalentes al goto de los LAN**

INSTRUCCIONES DE BIFURCACIÓN

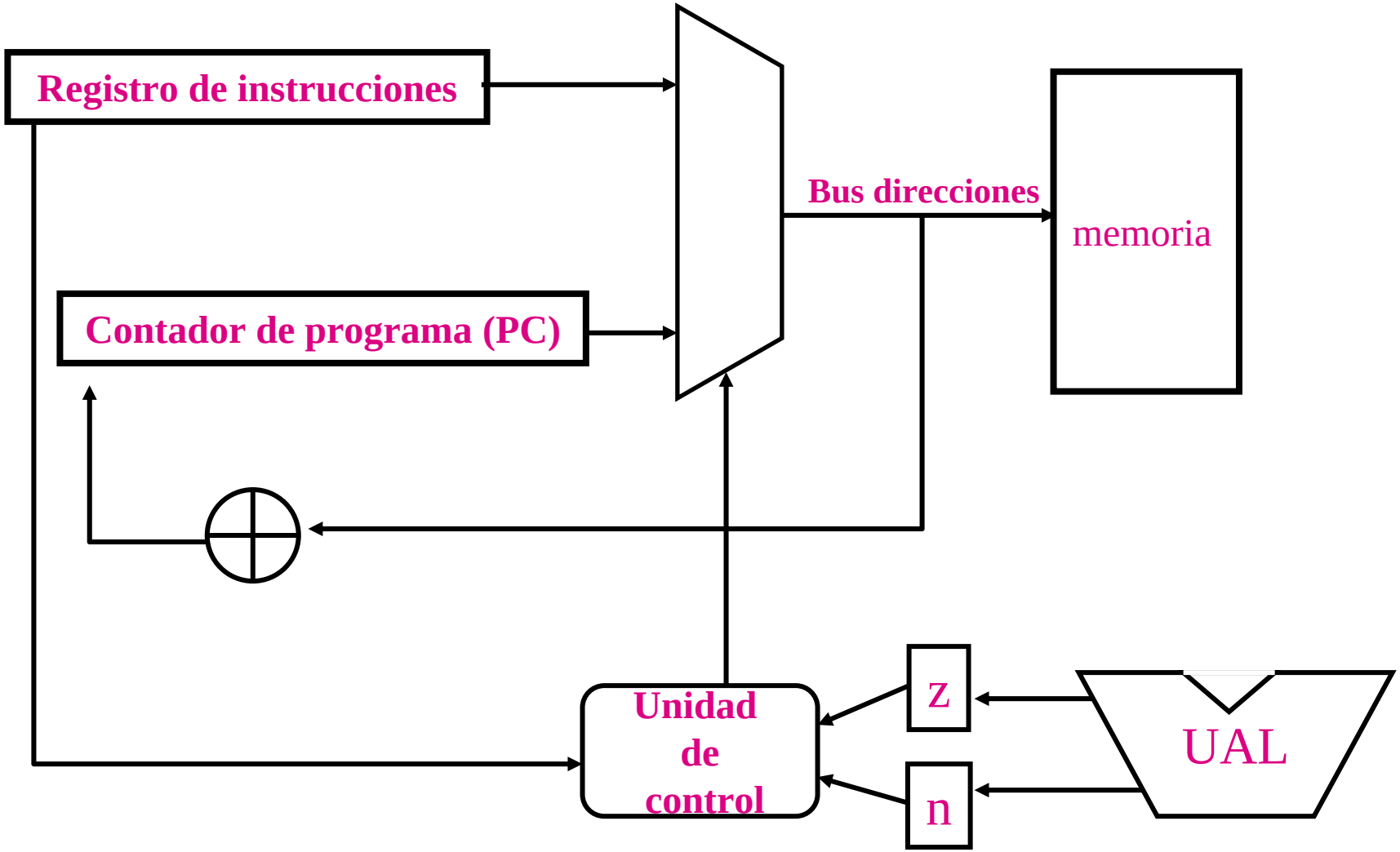


INSTRUCCIONES DE BIFURCACIÓN CONDICIONALES

- **Son las que tienen dos secuencias distintas:**
  - **Si se cumple la condición**
    - » **Salta**
    - » **PC := X**
  - **Si no se cumple la condición**
    - » **Incremento del contador**
    - » **PC := PC + Z**
- **Las condiciones se comprueban en los biestables de estado que almacenan información sobre operaciones realizadas con anterioridad**
- **Condiciones típicas:**

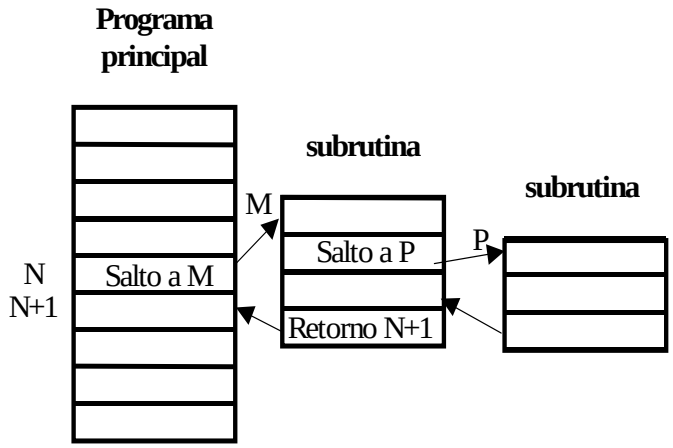
- **Less than (LT)**
- **No Equal (NE)**
- **Greater than (GT)**
- **Zero (Z)**
- **Not Zero (NZ)**
- **Greater than or Equal (GE)**
- **Equal (E)**

INSTRUCCIONES DE BIFURCACIÓN CONDICIONALES

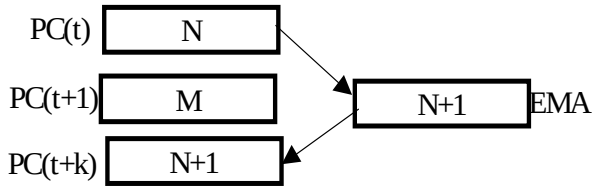


INSTRUCCIONES DE BIFURCACIÓN CON RETORNO (SALTOS A SUBRUTINA)

- **Una subrutina permite la reutilización del código**
- **Característica**
  - **Cuando acaba de ejecutarse vuelve al punto del programa desde el que se la llamó**
- **Implementación**
  - **Guardar (PC+Z) para retornar al punto donde bifurca**
- **Problema**
  - **Dónde guardar la dirección de retorno**
- **Para que el mecanismo sea eficaz deber permitir:**
  - **Llamadas anidada:**
    - » **Una subrutina llama a otra subrutina**
  - **Recursivas**
    - » **Que una subrutina se llame a sí misma**



Evolución de los programas



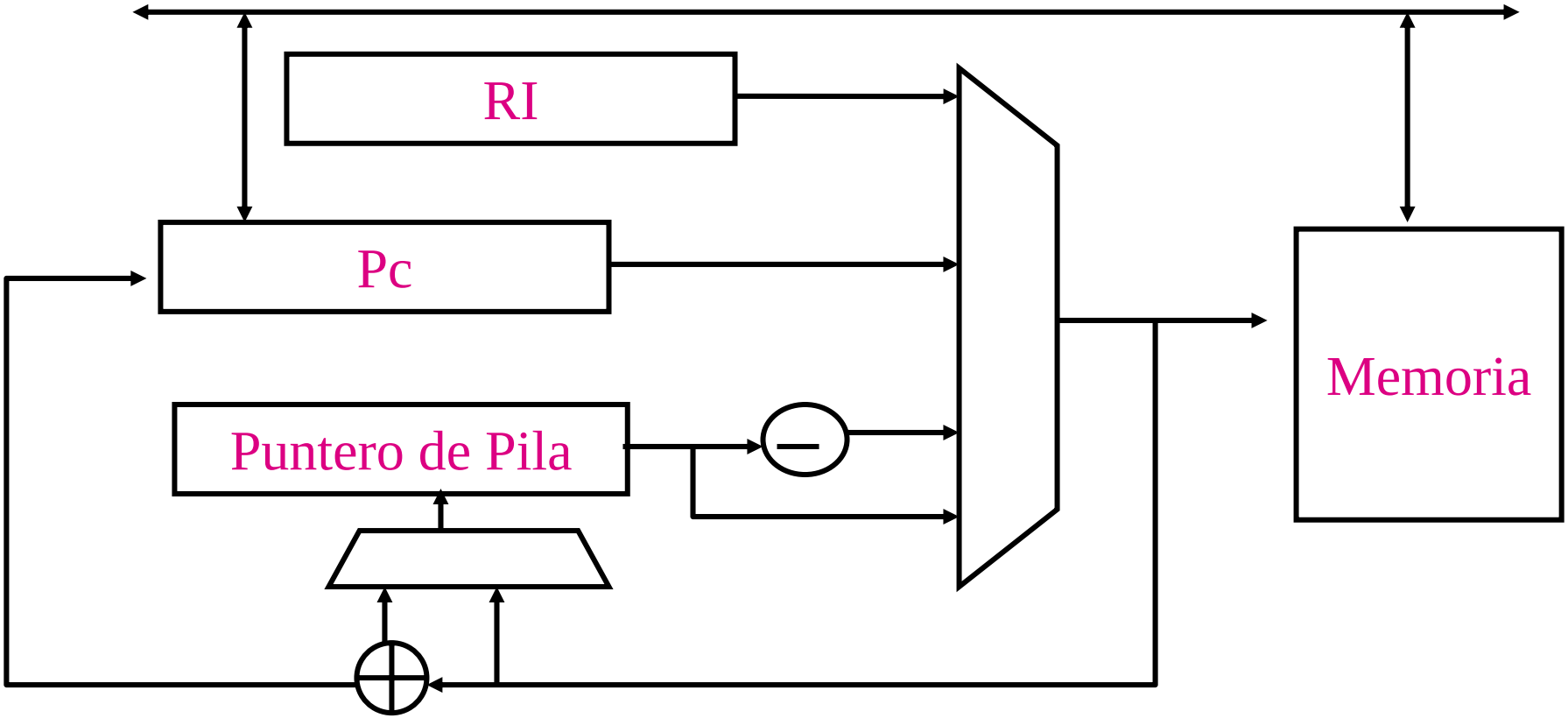
Evolución del pC y del Elemento de Memoria Auxiliar (EMA). Según se implemente el EMA se podrán hacer saltos a subrutina anidados y recursivos

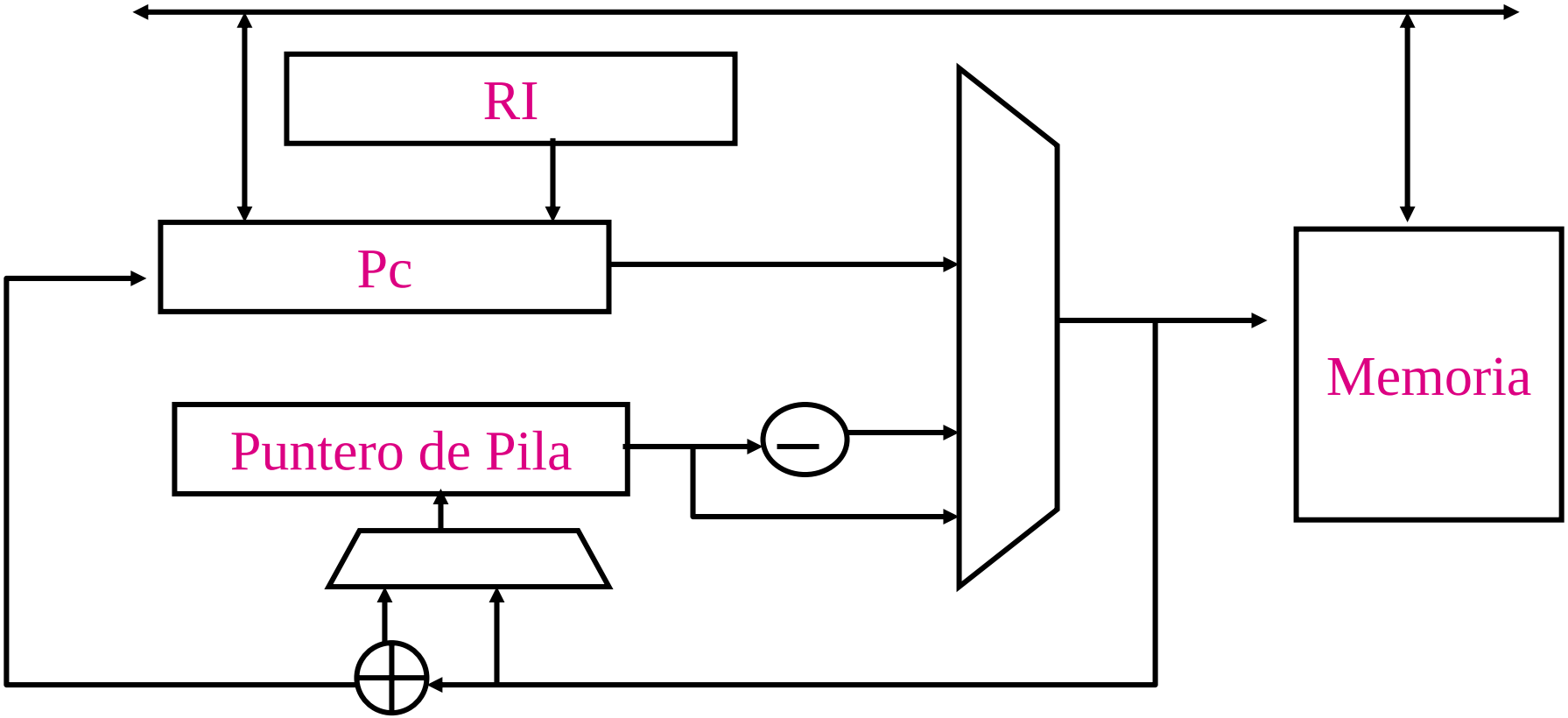
INSTRUCCIONES DE BIFURCACIÓN CON RETORNO (SALTOS A SUBRUTINA)

- **Soluciones:**
  - **Registro especial:**
    - » No permite anidamiento
    - » No permite recursividad
  - **Registro general:**
    - » No permite anidamiento
    - » No permite recursividad
    - » Inconveniente
      - ◆ Utiliza uno de los registros del sistema
  - **Dentro de la subrutina:**
    - » Permite llamadas anidadas
    - » Pero no recusivas
      - ◆ Machaca la información de la 1ª llamada
- **La mejor solución**
  - **Almacenamiento en una pila:**
    - » Permite llamadas anidadas
    - » Recusivas
- **Anidamiento y recursividad**
  - **Problema**
    - » Subrutina modifica los registros utilizados por la rutina actual
  - **Solución**
    - » Convenio para guardar y restaurar los registros
    - » Guardar el invocador
    - » Guarda el invocado

INSTRUCCIONES DE BIFURCACIÓN CON RETORNO (SALTOS A SUBRUTINA)

Suponiendo que la pila apunta a la posición vacía memoria




RI
Pc
Puntero de Pila
Memoria

INSTRUCCIONES ARITMÉTICAS

- Tipos de operandos habituales
  - Enteros con signo
  - Punto flotante
  - Packed decimal
- Operaciones aritméticas básicas para enteros con signo
  - Sumar
  - Restar
  - Multiplicar
  - Dividir
- Algunas otras operaciones típicas:
  - Absoluto
  - Negar
  - Incrementar
  - Decrementar
- La ejecución de las operaciones aritméticas suele incluir operaciones de transferencia de datos
- Todas las operaciones aritméticas afectan a los bits de estado

OTRAS INSTRUCCIONES

- **Instrucciones lógicas**
  - Se realizan sobre cada Bit de forma independiente
  - Modifican los Bit de estado
  - Los más típicos son:
    - » AND
    - » OR
    - » NOT
    - » XOR

- **Instrucciones de entrada/salida**
  - Operaciones de transferencia
  - destino y origen es el registro de un periférico
  - Pueden no existir si los periféricos están mapeados en memoria
  - movimiento al registro del interfaz

- **Instrucciones de desplazamiento**
  - Desplazamientos lógicos
    - » no conservan el signo
  - Desplazamientos aritméticos
    - » conservan el signo
    - » equivalentes a multiplicar o dividir por dos
  - Desplazamientos circulares
  - Todos modifican los biestables de estado

TIPOS DE OPERANDO

- **Tipos de datos más importantes:**
  - **Direcciones**
  - **Números**
  - **Caracteres**
  - **Lógicos**

OPERANDOS NUMÉRICOS

- Todos los lenguajes máquina incluyen datos numéricos
  - Incluidos los procesadores de datos no numéricos
- Características :
  - Limitados en la magnitud representable
  - Limitados en la precisión (como flotante)
- El programador debe conocer las consecuencias del redondeo
  - Overflow
  - Underflow
- Los tipos numéricos más comunes son:
  - Enteros (punto fijo)
  - Coma flotante
  - Decimal
    - » Cuándo se utiliza la codificación decimal?
      - ◆ Aplicaciones en las que hay una gran cantidad de operaciones de I/O y una computación comparativamente pequeña y simple

CARACTERES

- Los sistemas computadores no pueden trabajar con caracteres luego tiene que codificarlos.
- Codigo ASCII (american Standar Code for Information Interchange)
- Un carácter mediante 7 bits
- Como se pueden representar 128 caracteres que son más de los caracteres imprimibles, algunos de los patrones se utilizan para representar caracteres de control
- En ocasiones se codifican con 8 bit, que se puede fijar a cero o utilizarlo como bit de error
- Codificación sistematica:
  - 011Xxxx números decimales
  - Desde 100 mayusculas
  - Desde 110 minusculas
  - Desde 0000000 hasta 0011111 de control

MODOS DE DIRECCIONAMIENTO

* Inmediato
* Directo
* Relativo o con desplazamiento
* Indirecto

MODOS DE DIRECCIONAMIENTO

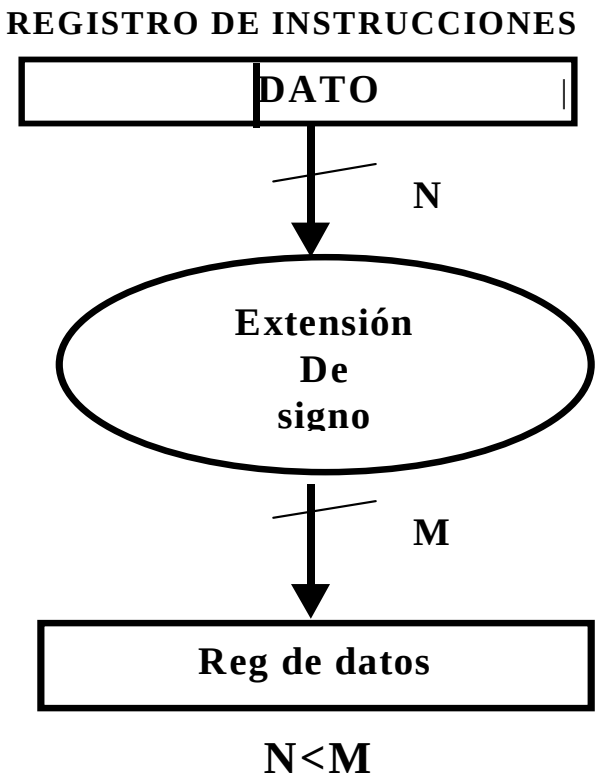
- **Procedimiento que permite determinar**
  - **Un objeto**
  - **La dirección de un objeto**
- **Objeto:**
  - **Operando**
  - **Instrucción**
  - **Resultado**
- **Generalmente lo que se especifica es la dirección**
- **Dirección efectiva**
  - **La dirección en la que se encuentra el objeto**
- **Un objeto puede residir**
  - **En la propia instrucción**
  - **En un registro**
  - **En la memoria principal**

- **Objetivo**
  - **Ahorro de espacio:**
    - » **Cuánto más pequeñas las I, menos MP**
  - **Código reubicable y reentrante:**
    - » **Direccionamientos relativos.**
  - **Estructuras de datos:**
    - » **Se simplifica Su manejo**

DIRECCIONAMIENTO INMEDIATO

- El objeto, que es un operando, se incluye dentro de la propia instrucción.
- Se suele utilizar para:
  - Definir y usar cte.
  - Inicializar variables
- Representación típica en c'2.
- Cuando el dato se carga en el registro de datos se realiza una operación de extensión de signo
- Ventaja:
  - No hace referencias a memoria.
  - Son operaciones mucho más rápidas
- Desventaja:
  - El tamaño del número está restringido al tamaño del campo dirección

DIRECCIONAMIENTO INMEDIATO

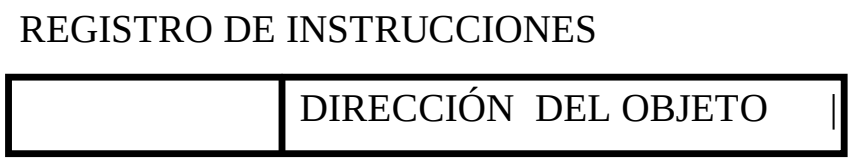


DIRECCIONAMIENTO DIRECTO

- La instrucción contiene la dirección real del objeto
- Se pueden clasificar en
  - Directode registro
  - directo de memoria

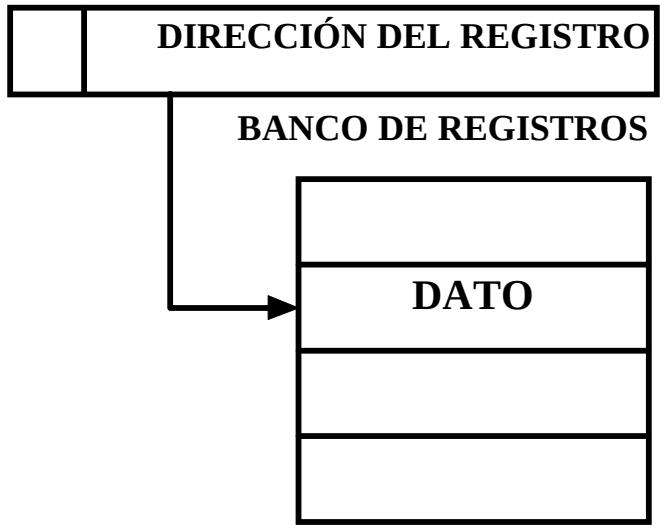
Un único acceso

ocupan mucha memoria



DIRECCIONAMIENTO DIRECTO DE REGISTRO

- El dato contenido en el registro direccionado en la instrucción
- Ventajas:
  - Campo de dirección pequeño
  - No se necesitan referencias a memoria
    - » rapidez
- Desventaja
  - Espacio de direcciones muy limitado
  - Uso ineficiente del banco de registros
    - » Cuando
      - mal un uso intensivo
    - » Causa
      - Su pequeño número
    - » Efecto
      - Añade un paso intermedio a la ejecución al tener que traer el dato de MP al registro
- Uso óptimo
  - Cuando el mismo dato se utiliza en múltiples operaciones si se optimiza el tiempo



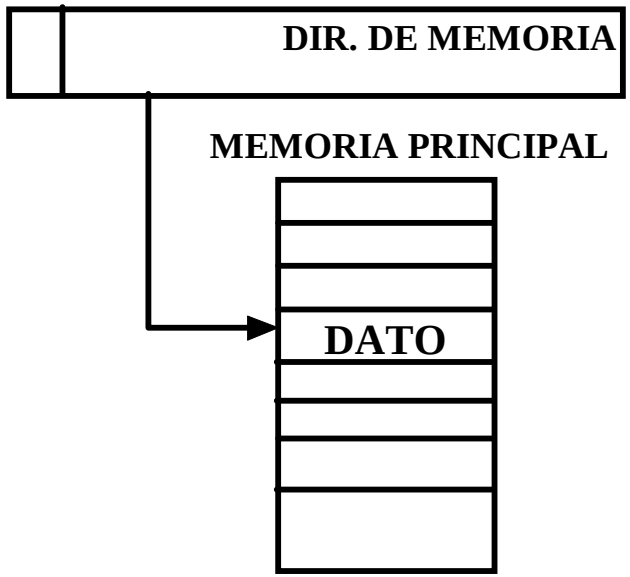
DIRECCIONAMIENTO DIRECTO
DE MEMORIA

- **El dato se encuentra en la dirección de memoria contenida en la instrucción**

- **Característica:**
  - **La lectura del dato siempre supone un acceso a memoria**
  - **Efecto**
    - » **más lento que el absoluto de Registro**

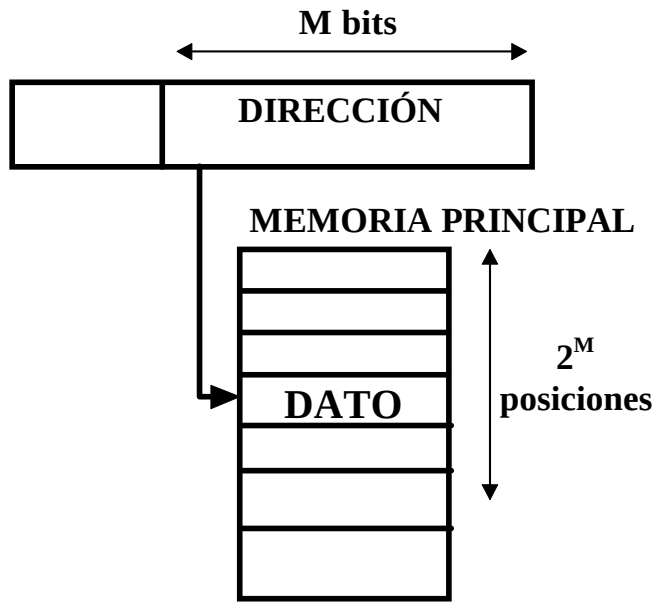
- **Inconveniente:**
  - **Según se implemente**
    - » **Espacio de direcciones limitado**

- **Ventaja:**
  - **Facilidad de implementación**



DIRECCIONAMIENTO DIRECTO
DE PAGINA BASE

- El dato se encuentra en la dirección de memoria contenida en la instrucción
- La información de la dirección está limitada
- Objetivo
  - Que permite referirse solamente a una parte del mapa de memoria
- Página
  - La zona de memoria a la que se permite el acceso
- Ventaja:
  - Se reduce el tamaño de la Instrucción
- Inconveniente
  - No se puede acceder a todas las posiciones de memoria



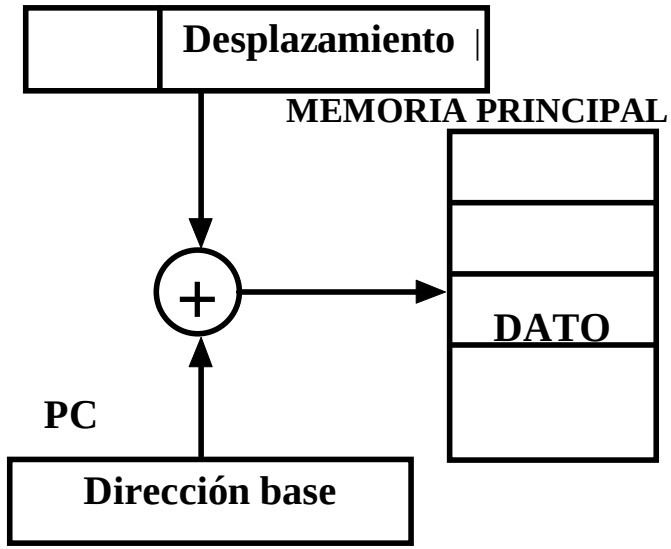
DIRECCIONAMIENTO RELATIVO

- Para encontrar la dirección efectiva hay que sumar un desplazamiento a la dirección
- Menos bits que el directo absoluto
- Los desplazamientos pueden ser :
  - Positivos
  - Negativos
- La suma no produce casi retardo en el cálculo de la dirección efectiva

- Uso:
  - Códigos reubicables
  - Recorrer de forma eficaz estructuras de datos
- Tipos
  - relativo a PC
  - relativo a regbase
  - relativo a índice
  - relativo a pila

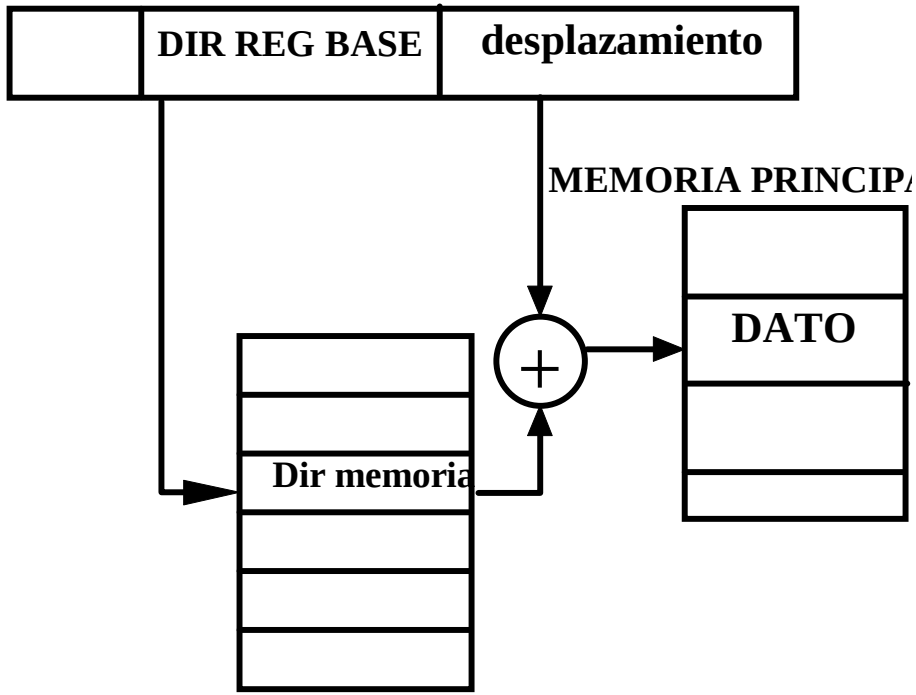
DIRECCIONAMIENTO RELATIVO A PC

- LLamado direccionamiento relativo
- El Registro base es el PC
- Explota el concepto de localidad:
  - Indicado para alcanzar I's próximas a la que se está ejecutando:
    - » ej Bifurcaciones de Bucles
- No se necesita referenciar al PC en la I



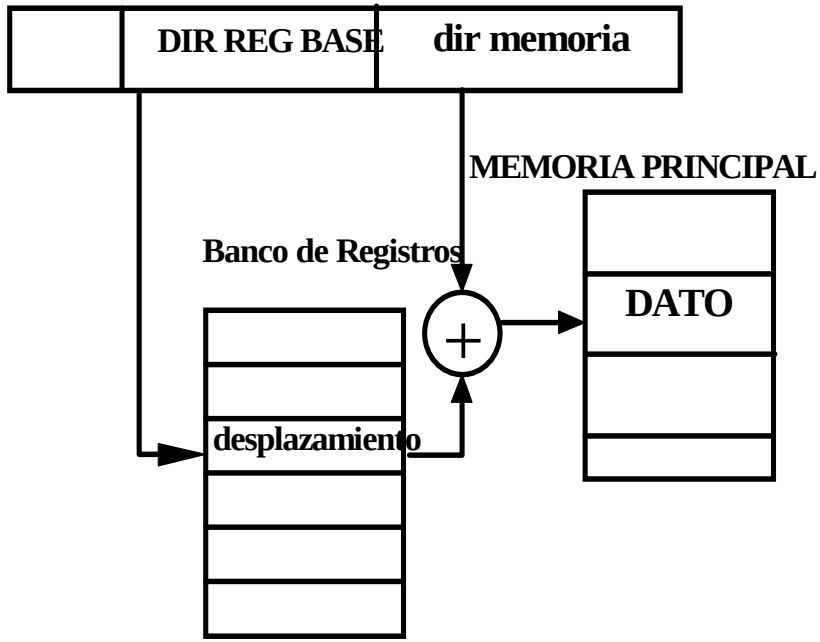
DIRECCIONAMIENTO RELATIVO
A REGISTRO BASE

- **El registro Base**
  - Pertenece al banco de registros del computador
  - La i debe identificar este registro
  - el reg contiene una dir memoria
- **Uso**
  - Estructuras de datos
  - Se carga en el Registro Base la primera posición
  - Se modifica por programa el desplazamiento para recorrer la estructura



DIRECCIONAMIENTO RELATIVO
A REGISTRO INDICE

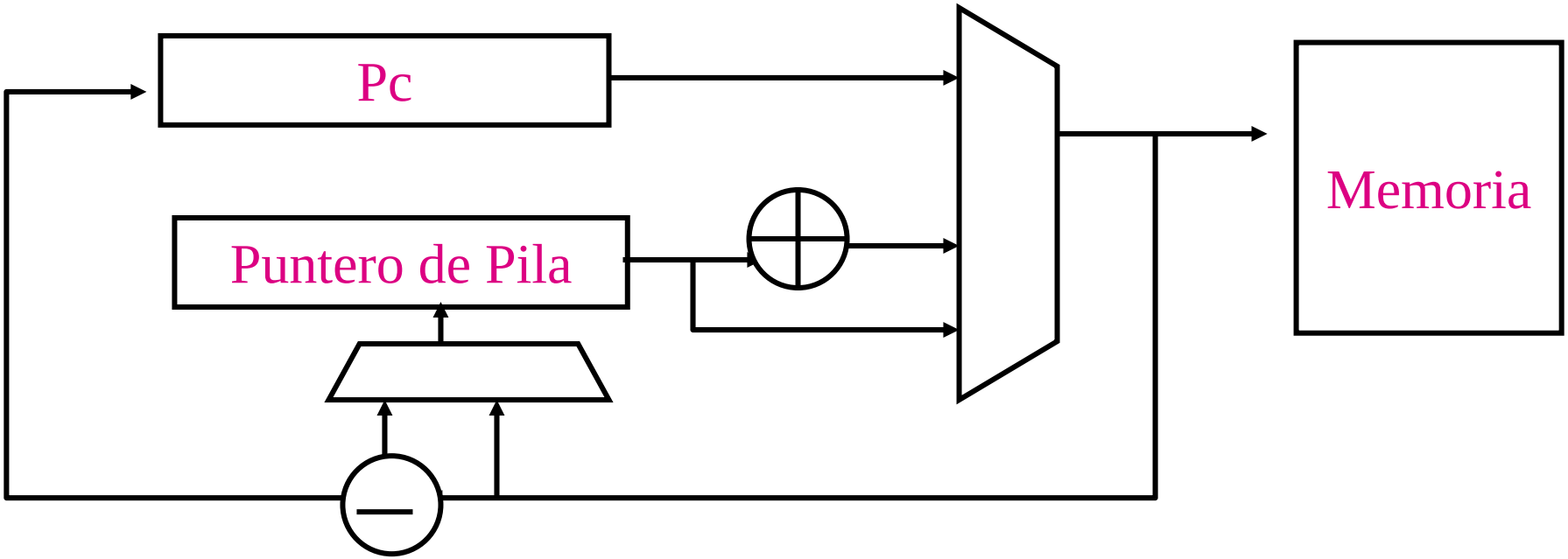
- También llamado indexado
- El registro contiene el desplazamiento
- El campo de dirección la dir de memoria
- El Registro Indice se automodifica
  - Preautoincremento
  - Preautodecremento
  - Postautoincremento.
  - Postautodecremento
- uso en operaciones iterativas
  - recorre los elementos de una tabla o vector
- Si solo hay un reg su dirección implícita en el código



DIRECCIONAMIENTO RELATIVO
A PILA

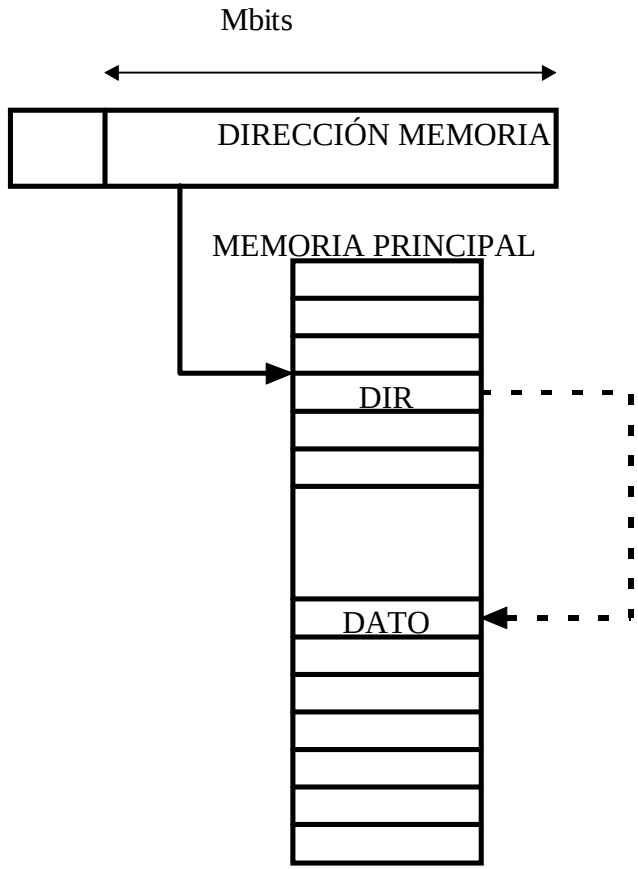
- Llamado también direccionamiento a Pila
- Sin desplazamientos
- La máquina debe disponer de un registro SP (puntero a pila)
- modo de operación
  - Para insertar un elemento a Pila:
    - » direccionamiento relativo a SP con preautoincremento.
  - Para sacar un elemento de la pila:
    - » postautodecremento con SP
- Permite instrucciones muy compactas:
- si sólo hay un SP la I no necesita información de dirección

DIRECCIONAMIENTO RELATIVO
A PILA



DIRECCIONAMIENTO INDIRECTO

- La dirección que contiene la I es la de la Dirección del Objeto
- Acceso adicional a memoria
- problema del direccionamiento Directo
  - dirección de la I usualmente es más pequeña que la longitud de dirección
- No se suele utilizar la indirección multinivel
- uso
  - acceso a diversas informaciones mediante una tabla de punteros



FORMATO DE LAS INSTRUCCIONES

- Especifica el significado de cada BIT de la instrucción
- Característica más importante es la Longitud:
  - Número de Bits que la componen
- INFORMACIÓN que contiene:
  - Operación
  - Dirección de los operandos
  - Dirección resultado
  - Dirección siguiente instrucción
  - Tipos de representación de operandos
- La instrucción se divide en campos.
  - Campo
    - » Cadenas de bits continuos.
  - Cada campo proporciona información específica

FORMATO DE LAS INSTRUCCIONES
LONGITUD

- **Decisión crítica de diseño**
- **Esta decisión afecta y se ve afectada por**
  - Tamaño de la memoria
  - Organización de la memoria
  - Estructura del BUS
  - Complejidad de CPU
  - Velocidad de CPU
- **Determina la riqueza y flexibilidad de la máquina**
- **La longitud debe ser**
  - Igual a la de la palabara de memoria
  - Ser múltiplo de ella
    - » Este caso produce cuellos de botella:
  - PE: dos accesos a MP para ejecutar una I
- **Se busca un equilibrio entre**
  - Potencia de repertorio de Is
    - » La potencia se consigue con longitudes mayores
  - Necesidad de salvar espacio

FORMATO DE LAS INSTRUCCIONES
CAMPOS DE LA INSTRUCIÓN

- **Cadenas de bits contínuos**
- **Cada campo proporciona información específica**
- **Tipos de campos de la instrucción**
  - **Código de operación.**
    - » **Código de operación indica la operación a realizar**
  - **Campo de dirección**
    - » **especifica la dirección de un dato, resultado o instrucción a la que se bifurca**
  - **Codigo de modo**
  - **Extensión de campo**

El Código de operación

- **indica la operación a realizar por la instrucción.**
- **Es un campo con tamaño fijo.**
- **Los modos de direccionamiento pueden ir incluidos en este código, o tener un campo independiente.**
- Como no todas las I's se utilizan con la misma frecuencia, En ocasiones se tienen códigos con menos bits para las Instrucciones más utilizadas, de esta manera se optimiza espacio.

- **Campo de dirección**
  - **Dirección especifica la dirección de un dato, resultado o instrucción a la que se bifurca**
  - **Puede incluir implícitamente los modos de direccionamiento utilizados**
- **Código de modo**
  - **Se puede especificar alguna característica de la operación**
  - **El tamaño de los operandos**
  - **El modo de direccionamiento**

Extensiones de código

- **Se utiliza un nº fijo pequeño de bits para casi todas las operaciones y para algunas de ellas se utiliza una extensión**
- **El código corto se suele utilizar en las instrucciones que más se utilizan**
- La extensión se suele utilizar en instrucciones del tipo bifurcación condicional y Desplazamiento

CARACTERÍSTICAS DEL FORMATO

- **Un computador puede tener varios formatos**
  - **El código es el que diferencia entre formatos**
- **Cada Instrucción encaja en un formato**
- **Cuánto menos formatos, más sencillo el computador**
  - **Los formatos múltiples complican en HW**
- **Formas de reducir la complejidad**
  - **los campos del mismo tipo de diferentes formatos**
    - » **Deben tener misma longitud**
    - » **Deben ocupar la misma posición**
    - » **Esto simplifica**
      - **La codificación de las instrucciones**
      - **Los caminos internos necesarios para mover la información**
  - **Se debe intentar que I próximas utilicen codificaciones similares**
  - **El tamaño debe encajar con facilidad en la palabra máquina.**
- **Para ahorrar el tamaño**
  - **Direccionamiento implicito--> PC**

MODELO DE EJECUCIÓN

- **Especifica el dispositivo en que están almacenados los operandos para realizar las operaciones**
- **El compilador es el encargado de asociar variables de los programas a los registros y las estructuras de datos a las memoria.**

- **Pila**
  - **Los datos en la cabecera de la pila**
  - **El resultado se deja en la pila.**
  - **Las Instrucciones no necesitan direcciones**
- **Registro-Memoria**
  - **1º Operando en Registro**
  - **2º Operando en Memoria**
  - **El resultado se carga en la posición del 2º operando**
  - **Sòlo hace falta un acceso a memoria**
- **Memoria-Memoria**
  - **3 Accesos a memoria**

MODELOS DE EJECUCIÓN

- **REGISTRO - REGISTRO**
  - Operandos en registros
  - Las Instrucciones llevan las direcciones de los Registros
  - Como el banco de registros no es muy grande
    - » Causa
      - ◆ Un gran número de registros incrementaría la duración del ciclo de reloj
        - – La señales tardan más en viajar más lejos
    - » Efecto
      - ◆ Deben existir operaciones de movimiento de datos de la memoria a los registros y viceversa
      - ◆ Incremento del tiempo necesario para traer operandos
        - – Si los datos no están en los registros hace falta dos accesos a MP para cargarlos
  - Es util para progranmas en los que se accede muchas veces a la misma variable
  - Es típico de los RISC

REPERTORIO DE INSTRUCCIONES MC68000
características generales

- **Registros de datos**
  - **D0-d7**
  - **32 bits**
  - **Al usarse como operando solo se modifica la parte menos significativa**
    - » **Tamaño correspondiente al tipo de operando**
- **Registros de direcciones**
  - **A0 A6**
  - **De 32 bits**
- **Pila**
  - **Un registro puntero de pila de usuario a7**
  - **Almacenar el estado del computador**
- **Registro de estados**
  - **SR**
  - **16 Bits**
  - **Registro de códigos de condición**
    - » **El byte menos significativo del SR**
    - » **Conjunto flags relativos al estado interno del Computador**

REPERTORIO DE INSTRUCCIONES DEL MC68000
CARACTERISTICAS GENERALES

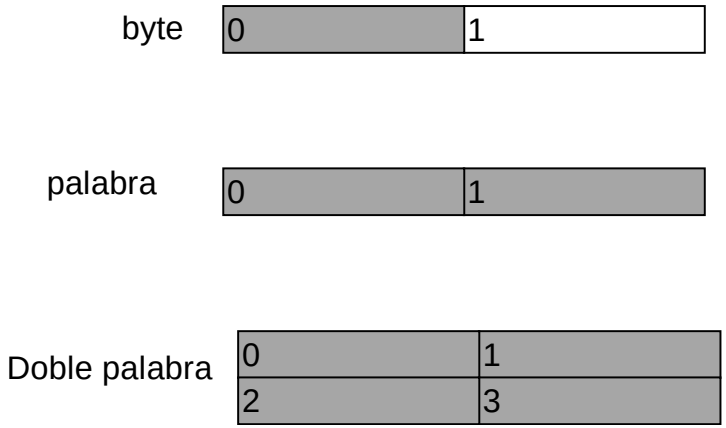
- 24 bits de código de dirección
  - Espacio de direccionamiento de 16 mbytes
- 56 tipos de instrucciones
- Operaciones sobre 5 tipos de datos
- Entrada salida mapeada en MP
- 14 modos de direccionamiento

- Tipos de datos
  - Bit (1)
  - Byte (8) .b
  - Palabra (16) .w
  - Doble palabra (32) .l
  - BCD (dígitos de 4 bits)
- Tamaño de las direcciones:
  - Palabra
  - Doble palabra
- Dir impares acceden sobre un byte

REFERENCIA DE INSTRUCCIONES DEL MC68000

COLOCACIÓN DE DATOS EN MEMORIA

2	3
4	5
6	7
8	9
10	11
12	13
14	15



- La parte más significativa de un dato se guarda en la dirección más baja de memoria
- En los datos de tamaño doble palabra (.l) los incrementos de dirección se hacen de cuatro en cuatro
- En los datos de tamaño palabra la dirección se incrementa de 2 en 2
- En los datos de tamaño byte (.b) los datos se incrementan de uno en uno
- Esta información es importante para recorrer tablas de datos

ENSAMBLADOR DEL 68000

- **Programa fuente**
  - Secuencia de sentencias
  - Tipos de sentencias:
    - » Comentarios
    - » Instrucciones
    - » Directivas de ensamblador

- **Comentario**
  - Se puede colocar al principio de la línea
    - » Comenzando en la columna 1
    - » Siendo el primer carácter *
  - Al final de la instrucción
    - » Con una separación de al menos un carácter en blanco

- **Instrucciones ejecutables**
  - Se componen de los campos
    - » Etiqueta
    - » Código
    - » Operandos
    - » Comentario
  - Separados por un espacio en blanco

- **Etiqueta**
  - Empieza en la 1ª columna de la línea
  - Con símbolo:
    - » Empieza en la primera columna
    - » Acabar con carácter en blanco
  - Sin símbolo
    - » 1ª Columna un espacio en blanco

ENSAMBLADOR 68000

- **Codigo de operación**
  - especifica la instrucción
  - contiene una extensión de tamaño de los datos
  - Por defecto la extensión es .w
- **las extensiones para los datos son:**
  - B byte 8
  - W palabra 16
  - L doble palabra
- **No todos los códigos llevan extensión de tamaño**
- **Campo operandos**
  - Si hay más de 1 operando, debe ir separado por comas
  - Sin espacio en blanco ni antes ni después de la coma
- **Operación fuente,destino**

INSTRUCCIONES 68000

- **Movimiento de datos**
  - Move.Tamaño fuente,destino
  - lea operando,registro
    - » calcula la dir efectiva de un operando y la carga en un registro de direcciones
  - LEA (A3),A4
- **ARITMETICA ENTERA**
  - complemento a dos
  - las operaciones sobre dir tienen más restricciones
  - operacion.tamaño fuente,destino
  - uno de los operandos debe ser un registro
- **suma y resta**
  - add operando1,operando2,
  - sub operando1,operando2
    - » La suma y la resta pueden tener cualquier tamaño
- **multiplicación y division**
  - mul.w operando1,operando2
  - div.w operando1,operando2
    - » destino/fuentedestino
    - » cociente [15:0]
    - » resto[31:16]
  - deben tener tamaño .W
  - pueden ser con signo o sin signo

INSTRUCCIONES 68000

- **Comparación**
  - **CMP.Tamaño operando1,operando2**
  - **Modifican del registro rs el flag Z**

- **Lógicas**
  - **AND**
  - **OR**
  - **EOR**
  - **NOT**

- **Desplazamiento**
  - **Lsl**
    - » **Logico a la izquierda**
  - **Asl**
    - » **Aritmético a la izquierda**
  - **Lsr**
    - » **Logico a la derecha**
  - **Asr**
    - » **Aritmético a la derecha**

INSTRUCCIONES DEL 68000

- **De control de flujo**
  - **Hay cuatro tipos diferentes**
    - » **Bifurcaciones condicionales**
    - » **Bifurcaciones incondicionales**
    - » **Salto subrutina**
    - » **Retorno subrutina**

INTRUCCIONES 68000
de control de programa

- **Bifurcaciones condicionlaes**
  - **Bcc ETIQUETA**
  - **donde cc es:**
    - » **GT mayor que**
    - » **EN no igual**
    - » **EQ igual**
    - » **GE mayor o igual**
    - » **lLEmenor o igual**
  - **se colocan después de una I de comparación**
  - **esto no es necesario porque la consulta se realiza sobre bits del registro CCR**

- **Salto incondicional BRA**
  - **Saltos relativos a PC**
  - **por defecto los desplazamientos son de 16 bits**
  - **se pueden seleccionar de 8**
    - » **El salto corto se fuerza con .s**
    - » **Cuidado con la extensión .s**
      - ◆ **el operando no puede referenciar el salto que va a continuación.**
        - – **Ej**

```
        bra.s lab1
lab1    move #1,d0
```

        - – **Esto es incorrecto**
- **Por defecto el máximo salto es 2767**

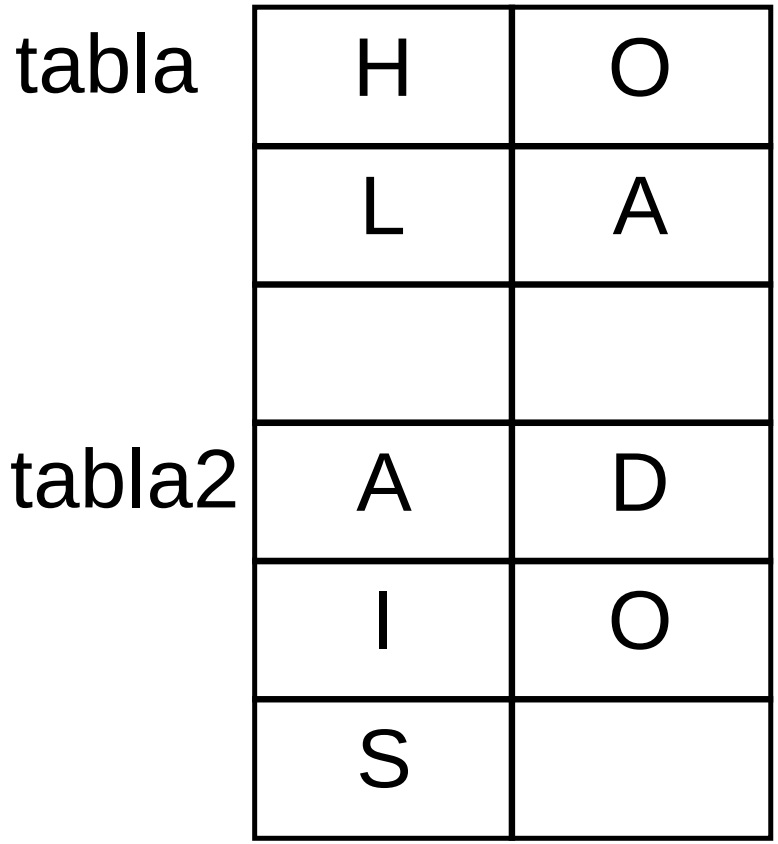
MODIFICACIONES A LAS INSTRUCCIONES

- **A**
  - **registro de direcciones como destino**
  - **movea d3,a5**
- **Q**
  - **quick rápido**
  - **operando inmediato en el rango entre 1 y 8**
  - **addq #10,d0 mal**
  - **addq#1,d0 bien**
- **I**
  - **indica un inmediato sin rango**
  - **movei #10,d0**

- **Formátos numéricos**
  - **% binario**
  - **@ octal**
  - **& decimal**
  - **$ hexadecimal**
  - **Por defecto el formato es decimal**
    - » **movei #16,d0**
    - » **mueve el 16 decimal al registro D0**

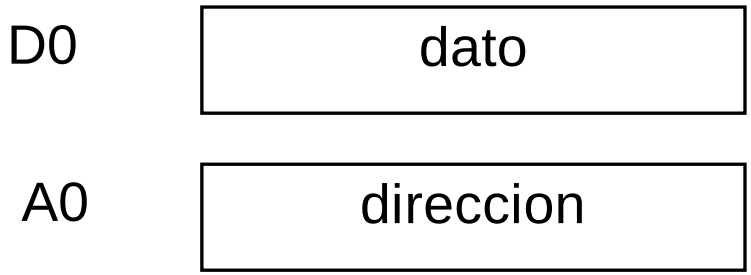
LITERALES ASCII

- **cadena de caracteres ascii**
  - **se especifica metiéndolo entre comillas**
  - **tabla dc.b “Hola”**
  - **tabla2 dc.b “adios”**

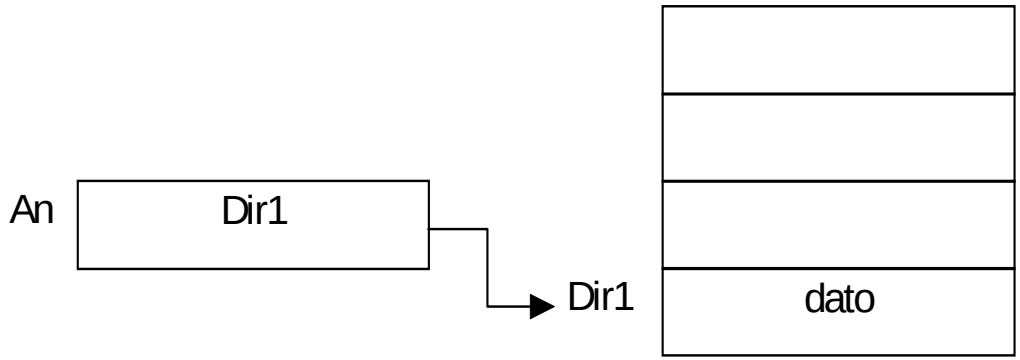


MODOS DE DIRECCIONAMIENTO

- **Directo a registro**
  - **Notación An o Dn**
  - **El registro referenciado contiene un operando**
  - **Movea A1,A2**

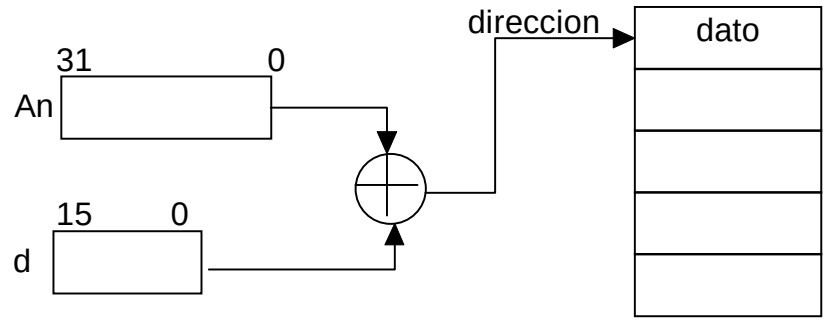


- **Indirecto de registro**
  - **Notación (An)**
  - **Solo para registros de direcciones**
  - **El registro contiene la dir del dato**
  - **Move #5,(a5)**



MODOS DE DIRECCIONAMIENTO

- **indirectro con Postincremento**
  - **direff=<An>**
  - **Notación (an)+**
  - **después de acceder a la dirección esta se incrementa en 1**
  - **El tamaño del incremento depende del tamaño del operando**



- **Indirecto con predecremento**
  - **direffe=<An>-decremento**
  - **-(an)**
  - **Antes del acceso se realiza un decremento de la dirección contenida en an.**
  - **El decremento depende del tamaño del operando**

- **Indirecto de registro con desplazamiento**
  - **notación d(An)**
  - **diref=<An>+<d>**

- **Absoluto**
  - **la dirección se especifica directamente en la instrucción**
  - **JMP $400**
- **Inmediato**
  - **El dato se encuentra en la instrucción**
  - **SUB.l #1,D0**

DEFINICIÓN DE SÍMBOLOS

- **EQU**
  - **Etiqueta equ expresión**
  - **Uno equ 1**
- **SET**
  - **Etiqueta set expresion**
  - **Uno set 1**
  - **Set permite la redefinición a lo largo del programa**
  - **No reserva espacio en memoria**
  - **Definición de datos reserva de memoria**
- **DC**
  - **etiqueta dc.b operando(s)**
  - **dc.w operandos**
  - **dc.c operandos**

Unoa dc.b 1

01	

Unob dc.w 1

00	01

Unoc dc.l 1

00	00
00	01

COMO DEFINIR UNA TABLA

- **Para recorrer las tablas se suele utilizar**
  - **Postincremento (An)+**
  - **Predecremento -(An)**
- **Matrices de datos**
  - **Una dimensión**
  - **Tabla dc.b 1,2,3,4,5**
  - **Tabla es una dirección simbólica a partir de la cual se alinean los datos**
  - **para guardar la dirección de tabla en un registro de direcciones**
  - **move.l #tabla,A1**

Tabla1 dc.b 1,2,3,4,5

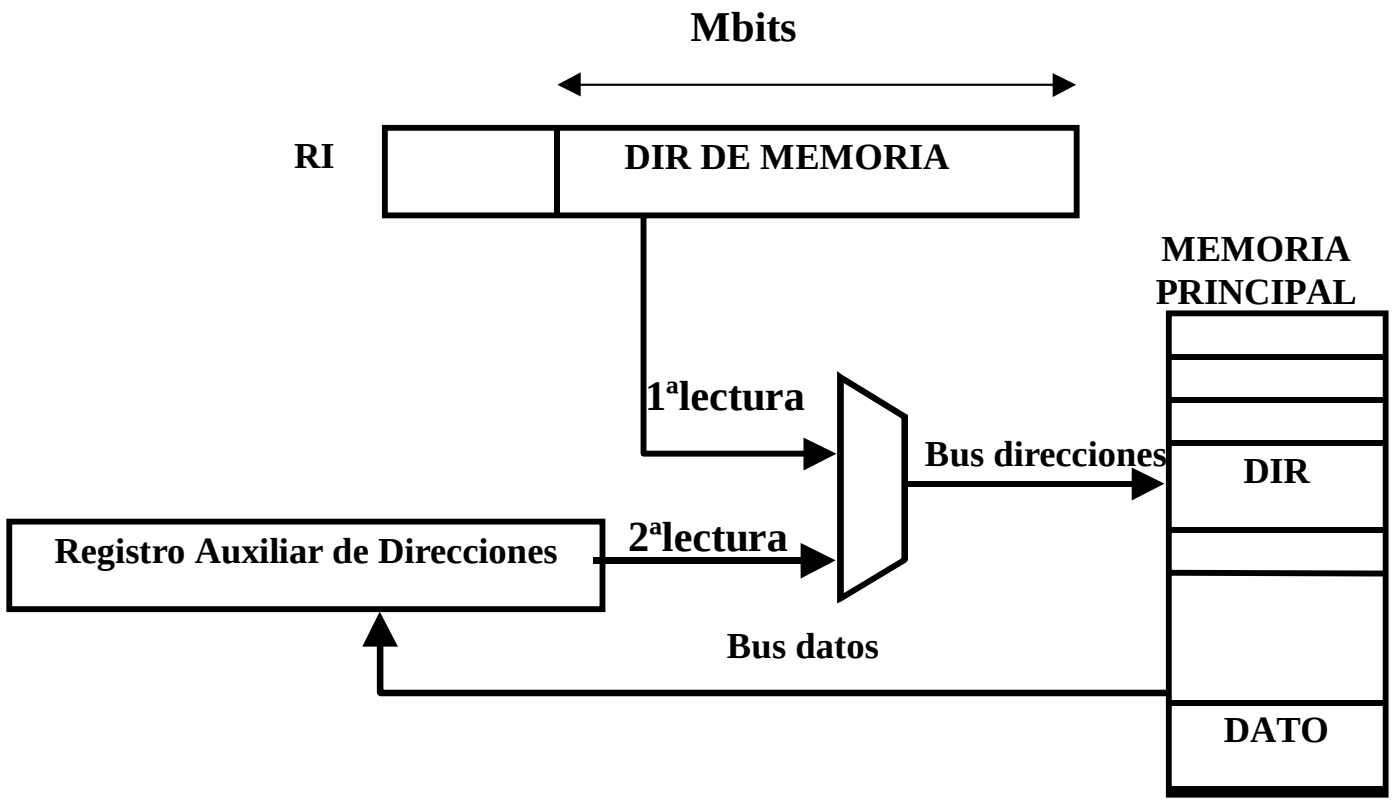
01	02
03	04
05	

Tabla2 dc.w 1,2,3,4,5

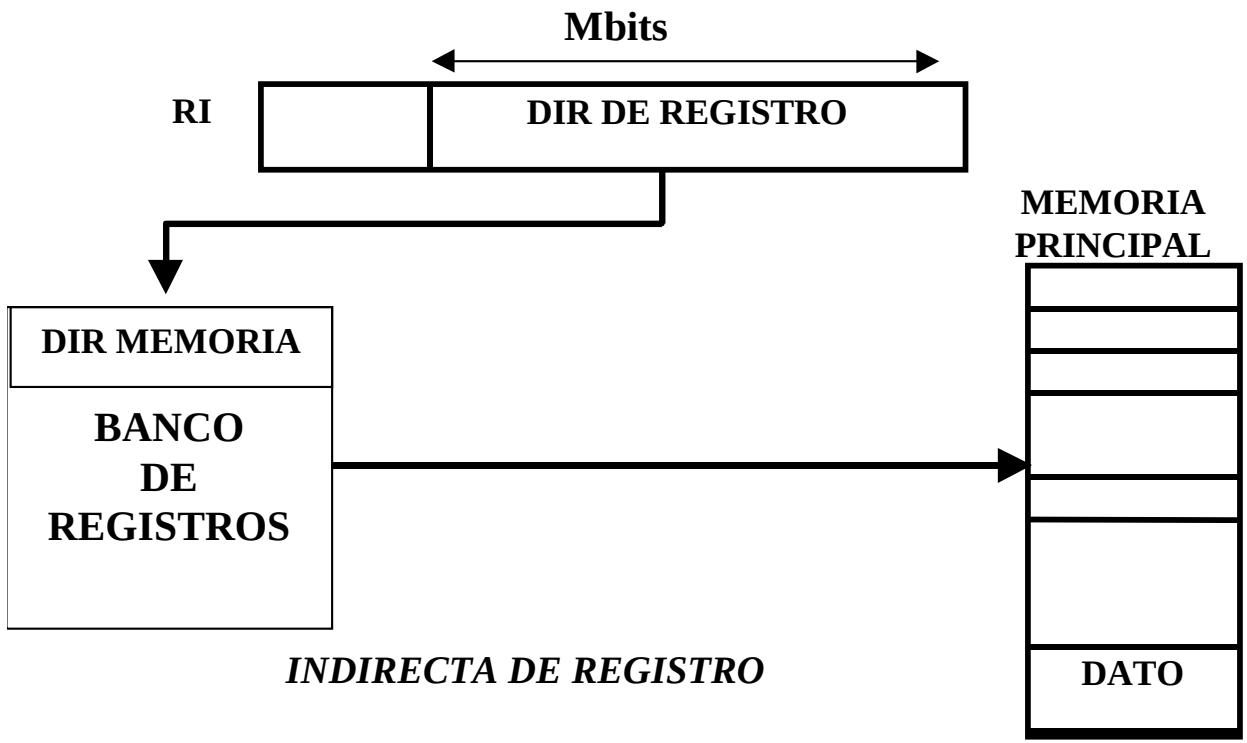
00	01
00	02
00	03
00	04
00	05

Tabla3 dc.l 1,2,3,4,5

00	00
00	01
00	00
00	02
00	00
00	03
00	00
00	04
00	00
00	05



INDIRECTA DE MEMORIA



INDIRECTA DE REGISTRO

BUSES DE SISTEMA

TEMA 4

DEFINICIONES

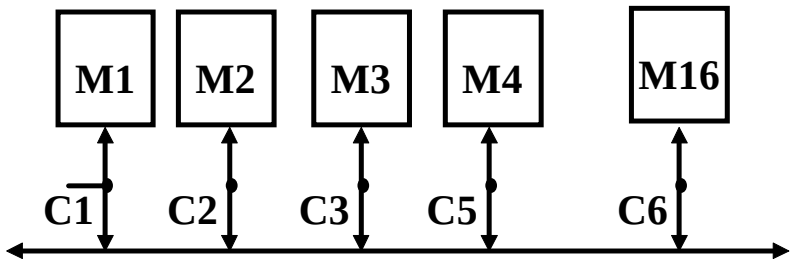
- **Para describir un sistema computador hay que describir:**
  - **El comportamiento de cada uno de los módulos**
    - » **Procesador**
      - ◆ **Camino de datos**
      - ◆ **Unidad de control**
    - » **Memoria**
    - » **Entrada/salida**
  - **La estructura de interconexión y el control necesario para manejarlo**

- **El conocimiento las interconexiones es importante**
  - **Es la que configura la estructura del sistema**
  - **De cara a la evaluación de los rendimientos**
    - » **Ayuda a comprender donde se encuentran los cuellos de botella**

- **En muchas ocasiones la mejora del rendimiento pasa por la mejora del bus en lugar de la mejora de los módulos conectados a él**

DEFINICIONES

- **BUS**
  - Camino de comunicación que conecta dos o más dispositivos
  - Medio de transmisión compartido
  - La señal que se transmite está disponible para todos los dispositivos conectados al bus
- **Sólo un dispositivo puede transmitir cada vez**
  - Puertas triestate
- **Colisión**
  - Si dos dispositivos transmiten simultáneamente
  - Efecto--> Se pierde la información



CARACTERISTICAS DEL BUS

- **Ventajas:**
  - **Flexibilidad**
    - » **Facilidad de ampliación**
  - **Bajo coste**
    - » **Elimina caminos innecesarios**

- **Desventajas:**
  - **Cuello de botella de comunicación**
    - » **Limitando la máxima productividad de la entrada y la salida**
  - **Máxima velocidad del bus limitada por factores físicos:**
    - » **La longitud del bus**
    - » **Número de dispositivos**

ELEMENTOS DE DISEÑO DE UN BUS

- **Elementos de diseño de un bus:**
  - **Capacidad de conexión:**
    - » **nº máximo de módulos que se pueden conectar al bus**
  - **Protocolo de arbitraje:**
    - » **Resolución de conflictos de acceso**
  - **Protocolo de bus.-**
    - » **Sincronización master slave**
  - **Ancho de bus.-**
    - » **Nº total el líneas**
  - **Ancho de banda.**
    - » **Nº de bits que es capaz de enviar por unidad de tiempo**
  - **Ancho de datos.-**
    - » **Nº de líneas de datos**

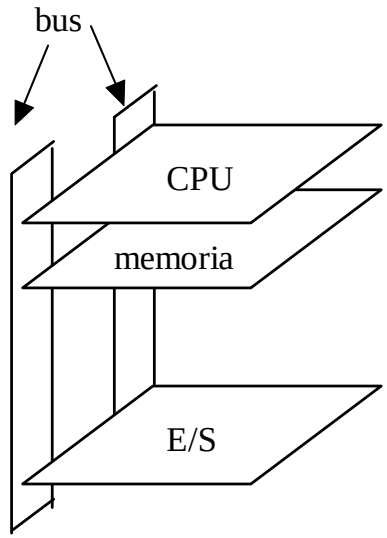
CARACTERISTICAS DEL BUS

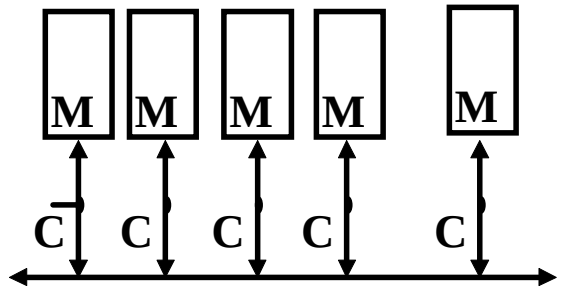
- **Tipos de transferencias:**
  - **Memoria - CPU**
  - **CPU - memoria**
  - **E/S - CPU**
  - **CPU - ES**
  - **Memoria - ES**
  - **ES - memoria. (DMA)**

- **Bus de sistema:**
  - **Conecta los principales componentes del sistema:**
    - » **CPU**
    - » **Memoria**
    - » **E/S**

ESTRUCTURA DE UN BUS

- **50 - 100 líneas separadas**
- **Cada línea tiene una función**
- **Las líneas se pueden clasificar en tres grupos funcionales:**
  - **Datos**
  - **Direcciones**
  - **Control**
- **Función del bus:**
  - **Soportar la información a transmitir**
  - **Garantizar la correcta comunicación entre los elementos que lo comparten**





MODO DE OPERACIÓN

- **Los elementos implicados en una transferencia son:**
  - **MASTER**
    - » **Inicia y dirige la transferencia :**
    - » **Ejemplos**
      - **CPU**
      - **DMA**
  - **SLAVE**
    - » **Obedece y accede a las peticiones del master**
    - » **Ejemplos**
      - **Memoria**
      - **Interfaz de E/S**
- **Tipos de transferencia**
  - **Escritura**
    - » **el Master envía un dato al slave**
  - **Lectura**
    - » **el slave envía un dato al master**
- **Ciclo de bus**
  - **Cualquier transferencia completa**

MODO DE OPERACIÓN

- **Modos de operación del master**
  - **Para enviar información**
    - » **Fase de arbitraje**
      - ◆ **Obtiene el uso del Bus**
    - » **Fase de sincronización**
      - ◆ **Transfiere los datos por el Bus**
  - **Para obtener información :**
    - » **Fase de arbitraje**
      - ◆ **Obtiene el uso del Bus**
    - » **Fase de sincronización**
      - ◆ **Transfiere la petición de información con las líneas de dir y control adecuadas**
      - ◆ **Espera a que le envíen los datos**

PARALELISMO DE BUS

- **Bus paralelo:**
  - El ancho de palabra coincide con el ancho de la información a transmitir
  - También se le llama dedicado
- **Bus multiplexado:**
  - Surge del hecho de tener que ahorrar patillas en los circuitos integrados
  - Transmite tipos de información diferentes, en tiempos diferentes por las mismas líneas
  - Necesita señales adicionales para indicar el tipo de información que se envía
  - Suele ser necesario añadir un descodificador para realizar la demultiplexación temporal
  - Suele ser necesario un multiplexador para realizar la multiplexación temporal
- **Bus serie:**
  - Caso extremo de multiplexación
  - Información se envía bit a bit
  - No existen señal de demultiplexión
  - Registros de desplazamiento

PROTOCOLOS DE ARBITRAJE

- **Objetivo:**
  - **garantizar el acceso al bus sin conflictos cuando existen varios master que pueden solicitarlo**
- **Master**
  - **modulo capaz de solicitar una acceso al bus**
  - **Ejemplos de sistemas con varios master**
    - » **Procesador y controladores de DMA**
    - » **Procesador + procesador de ES+ coprocesador matemático**
    - » **Sistemas multiprocesador**
- **Tipos de protocolo:**
  - **Centralizados**
  - **Distribuidos**

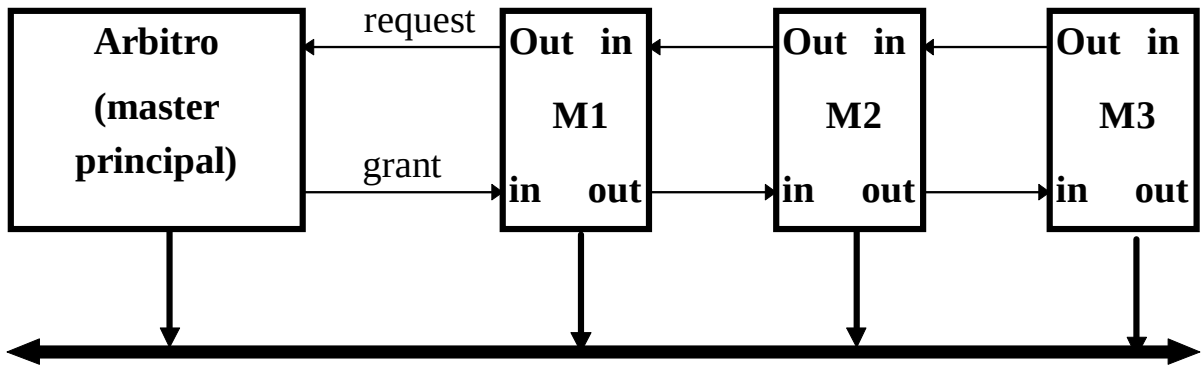
PROTOCOLOS DE ARBITRAJE

- **CENTRALIZADOS**
  - un arbitro de bus o master principal
  - función
    - » controlar el acceso al bus
  - Tipo:
    - » Daisy chain de dos hilos
    - » De tres hilos
    - » De cuatro hilos

- **DISTRIBUIDOS**
  - El control del bus entre todos los masters de una forma cooperante
  - Tipo:
    - » Líneas de identificación
    - » Códigos de identificación

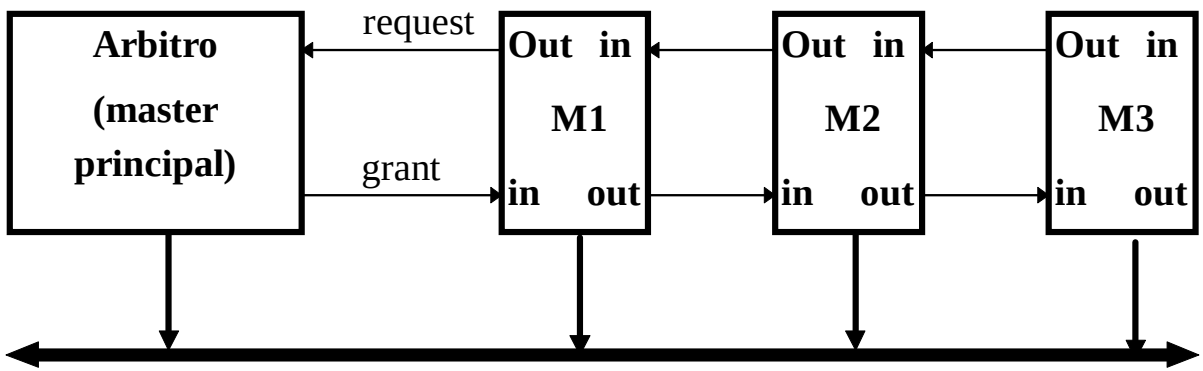
ARBITRAJE CENTRALIZADO
DAISY CHAIN DE DOS HILOS

- Dos líneas de arbitrajes comunes a todos los masters:
  - Bus request.- Petición de bus
  - Bus grant.- Línea de concesión del bus
    - » Se deja pasar hasta el peticionario que la retiene
      - ◆ Request in activada
- La prioridad la determina el orden en que se conectan los master al bus
  - Si M2 y M3 lo solicitan al tiempo el control lo consigue M2
- El control del bus está asignado cuando están activas las dos señales
- Señal de concesión (grant) se desactiva después de desactivar la señal de petición que llega al master principal



ARBITRAJE CENTRALIZADO
DAISY CHAIN DE DOS HILOS

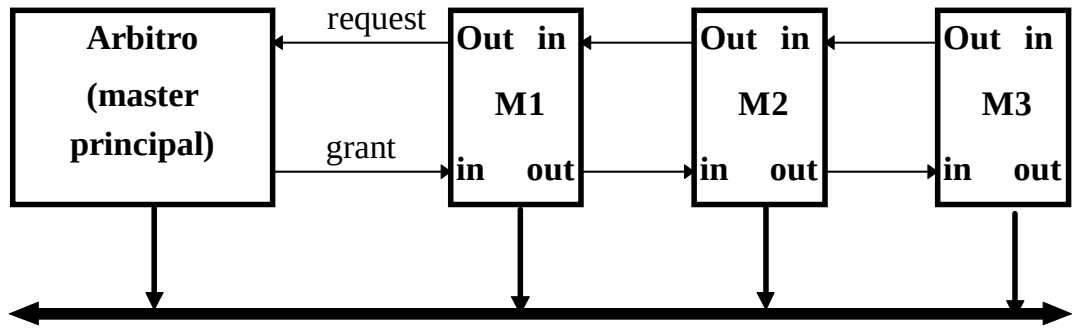
- **Situación conflictiva 1**
  - **M2**
    - » **pide el bus en T1**
    - » **detecta grant in activa y toma el control del bus**
  - **M1**
    - » **pide el bus en T2 con T2>T1**
    - » **detecta grant activa(debido a la concesión del bus M2)**
    - » **también toma el control del bus**
  - **solución:**
    - » **grant funciona por flanco, no por nivel**



ARBITRAJE CENTRALIZADO
DAISY CHAIN DE DOS HILOS

- **Situación conflictiva 2**
  - Cuando M2 deja de controlar el bus
    - » request se desactiva
      - ◆ grant tarda en desactivarse desde el arbitro
- **En este tiempo puede ocurrir**
  - M3 activa request
    - » esta señal llega a M2
    - » como M2 tiene grant todavía activada la deja pasar a M3
    - » M3 toma el control
  - M1 pide el control al arbitro
    - » M1 recibe grant in del arbitro
    - » toma el control

- **Resumen:**
  - Como si existieran dos señales grant independientes:
    - » La que genera el arbitro
    - » La que genera M2
- **Solución**
  - Mi solo propaga grant si request in está activa antes de que Mi desactive su petición local
  - se evita que el arbitro genere una nueva señal grant que cree un conflicto
  - Esto provoca que la concesión del bus sea secuencial. (round robin)

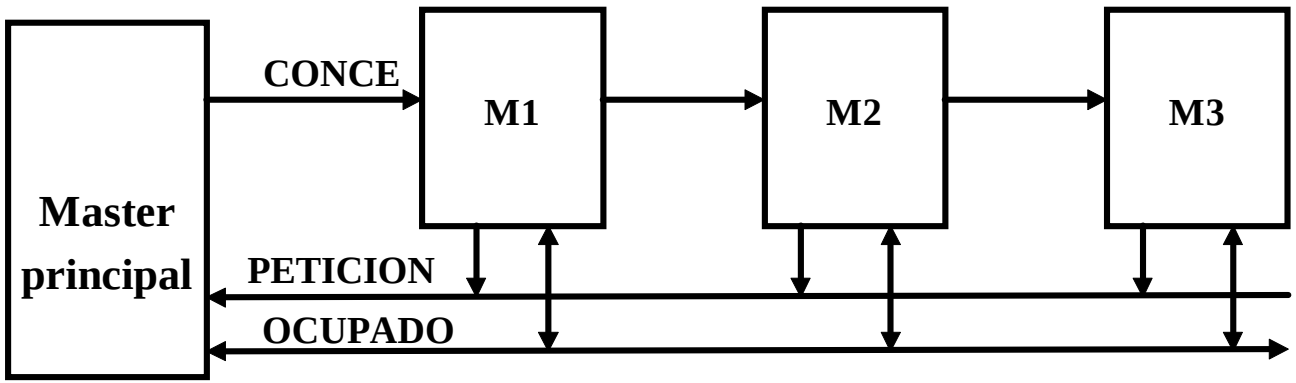


ARBITRAJE CENTRALIZADO
DAISY CHAIN DE DOS HILOS

- **Ventajas**
  - **Muy sencillo de implementar**
  - **Solo necesita dos líneas de arbitraje**
- **Desventajas:**
  - **Retardos en la propagación de las señales de control por tener que atravesar todos los masters**
    - » **Grant**
    - » **Request**
  - **Múltiples situaciones conflictivas**
  - **Prioridades arbitrarias**
- **No se pueden realizar peticiones de bus mientras este está ocupado**
  - **Se sabe que el bus esta ocupado porque tanto peticion como concesión están activas.**
  - **Cuando se deja el bus libre el modulo desactiva petición que provoca la desactivación de conce**
  - **Proceso lento mientras que un bus esta ocupado otros módulos no pueden solicitar el bus.**
- **Ejemplo i8086**

ARBITRAJE CENTRALIZADO
PROTOCOLO DE TRES HILOS

- La señal de petición ya no se transmite a través de los módulos
- Líneas de arbitraje:
  - **Bus petición** línea de petición de bus
  - **Bus conce.**- Línea de concesión de bus
  - **Bus ocupado**.- Línea de bus ocupado
    - » indica que el bus está ocupado
    - » indica que módulo tiene el control
- Utilizar esta nueva línea permite solicitar el bus cuando este todavía está ocupado,
  - aunque la señal de concedido no la genera el arbitro hasta que el bus queda vacío

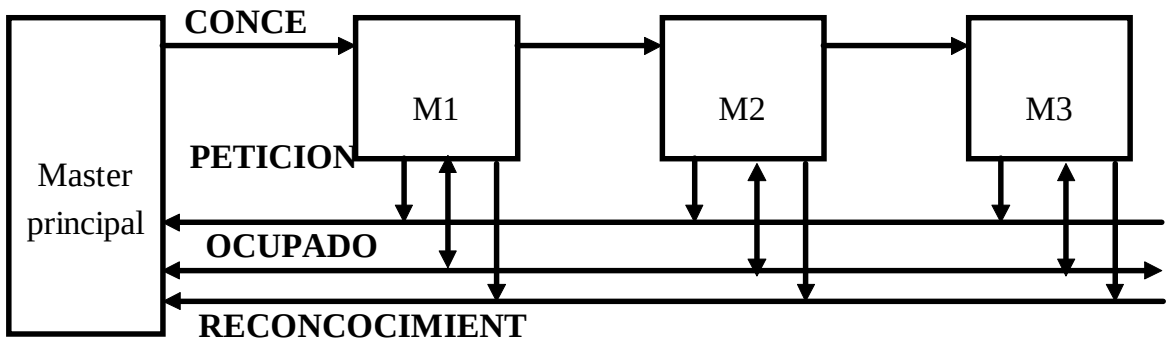


ARBITRAJE CENTRALIZADO
PROTOCOLO DE TRES HILOS

- Un módulo pide el Bus activando PETICION
- El arbitro activa CONCEDIDO cuando detecta
  - PETICION activado
  - OCUPADO desactivado
- Si un módulo recibe CONCEDIDO y no ha realizado la petición: lo transmite al siguiente
- Un módulo toma control cuando se cumplen tres condiciones
  - su PETICION activo
  - OCUPADO inactiva
  - Detecta el flanco de subida de CONCEDIDO
- Cuándo un módulo toma el control
  - » se activa la señal OCUPADO
  - » desactiva PETICION
- El arbitro desactiva CONCEDIDO en cuanto se activa OCUPADO
- Las señales de PETICION queda libre y la puede activar cualquier módulo para ir solicitando el control
- La señal conce esta desactivada y preparada para activarse en cuanto la señal de bus quede libre

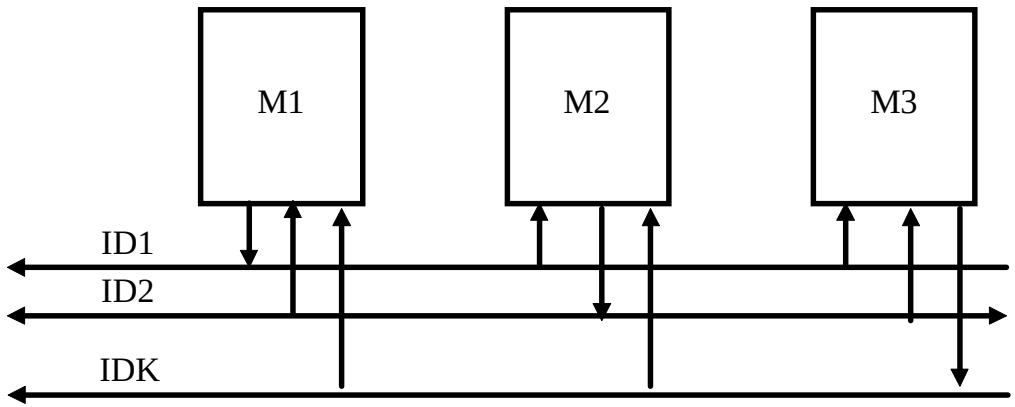
ARBITRAJE CENTRALIZADO
PROTOCOLO DE CUATRO HILOS

- **Permite solapar la transferencia de ciclo actual con el arbitraje del ciclo siguiente**
  - **concede el bus a un master, que solo tiene que esperara a que el bus quede libre para tomar el control**
- **líneas de arbitraje:**
  - **Bus PETICIÓN.- línea de petición de bus**
  - **Bus CONCE.- línea de concesión de bus**
  - **Bus OCUPADO.- línea de bus ocupado**
  - **Bus RECONOCIMIENTO .- línea indica que modulo tiene el control**
    - » **Esta línea la activa el master que solicito el bus en respuesta bus grant, cuando el bus está ocupado**
    - » **Cuando está activada el arbitro queda inhibido**



ARBITRAJE DISTRIBUIDO
PROTOCOLO DE LINEAS DE IDENTIFICACIÓN

- Cuando un master quiere tomar el control del bus, activa su línea de identificación
- Cada línea de identificación tiene asignada una prioridad:
  - **Prioridad (ID1) <Prioridad (ID2)<…<Prioridad(Idk)**
- Si varios masters activan simultáneamente sus líneas de identificación, gana el de mayor prioridad
  - **Los módulos leen el resto de las líneas y comprueban si su petición tiene mayor prioridad que el resto de las peticiones**
- Funcionamiento alternativo: las prioridades pueden ser variables

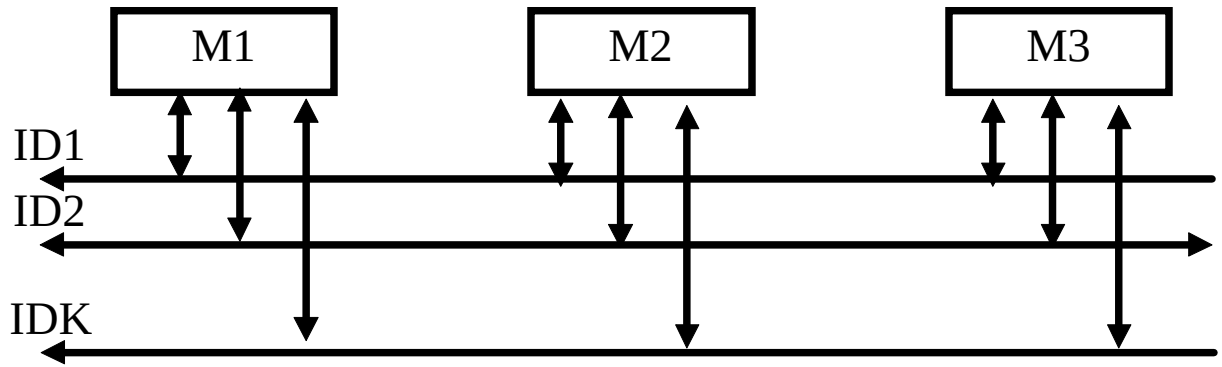


ARBITRAJE DISTRIBUIDO
PROTOCOLO DE LINEAS DE IDENTIFICACIÓN

- **Desventajas:**
  - **Numero de dispositivos limitado por el número de líneas de arbitraje**
- **Ejemplos:**
  - **Prioridad fija**
    - » **vax sbi**
    - » **scsi**
  - **Prioridad variable:**
    - » **dec70000/10000 axp,**
    - » **alpha server 8000**

ARBITRAJE DISTRIBUIDO
CÓDIGOS DE IDENTIFICACIÓN

- Cada master tiene un código de identificación de n bits (máximo de 2^n masters)
- Existen n líneas de arbitraje
- Cuando un master quiere tomar el control del bus pone sus código en las n líneas de arbitraje
- Si varios masters compiten por el bus gana el de mayor código
- Ejemplos:
  - **multibus II**
  - **Future bus+**



SINCRONIZACIÓN
PROTOCOLOS DE BUS

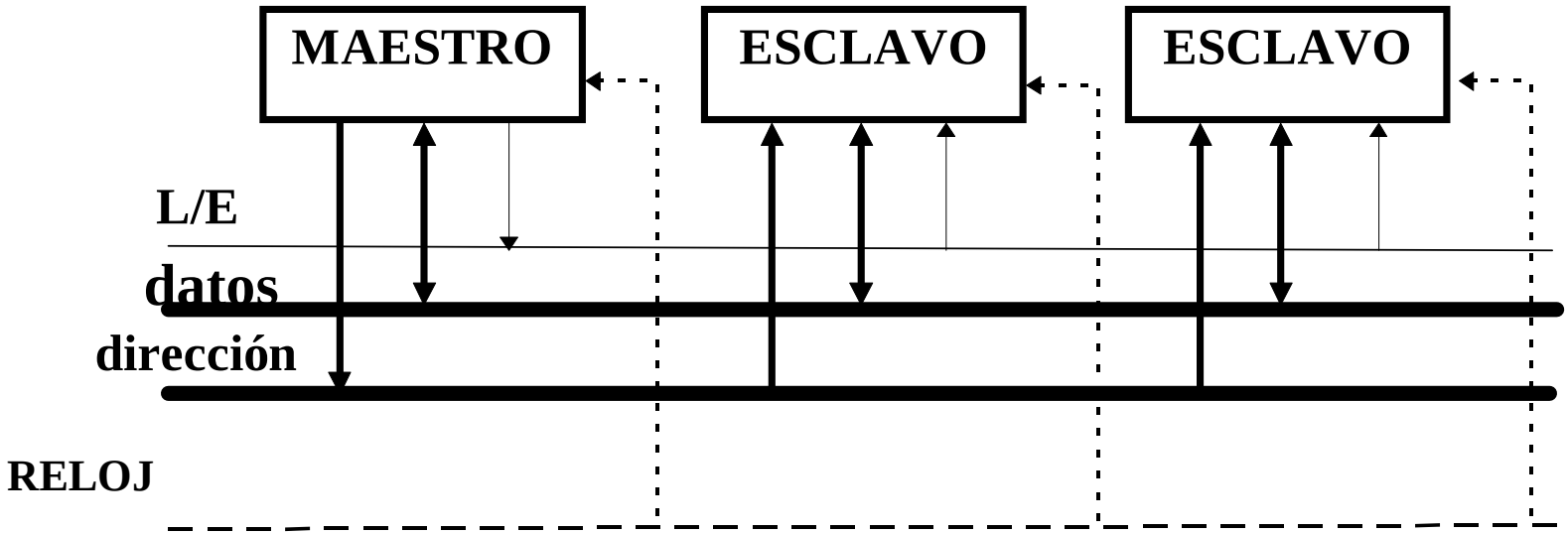
- **Forma de determinar cuando se envía la información entre los elementos implicados en una transición**

- **Hay que indicar el conjunto de líneas que lo implementan**
  - **datos**
  - **dir**
  - **control**

- **Clases:**
  - **Síncrono**
  - **Asíncrono**
  - **Semisíncrono**
  - **Ciclo partido**

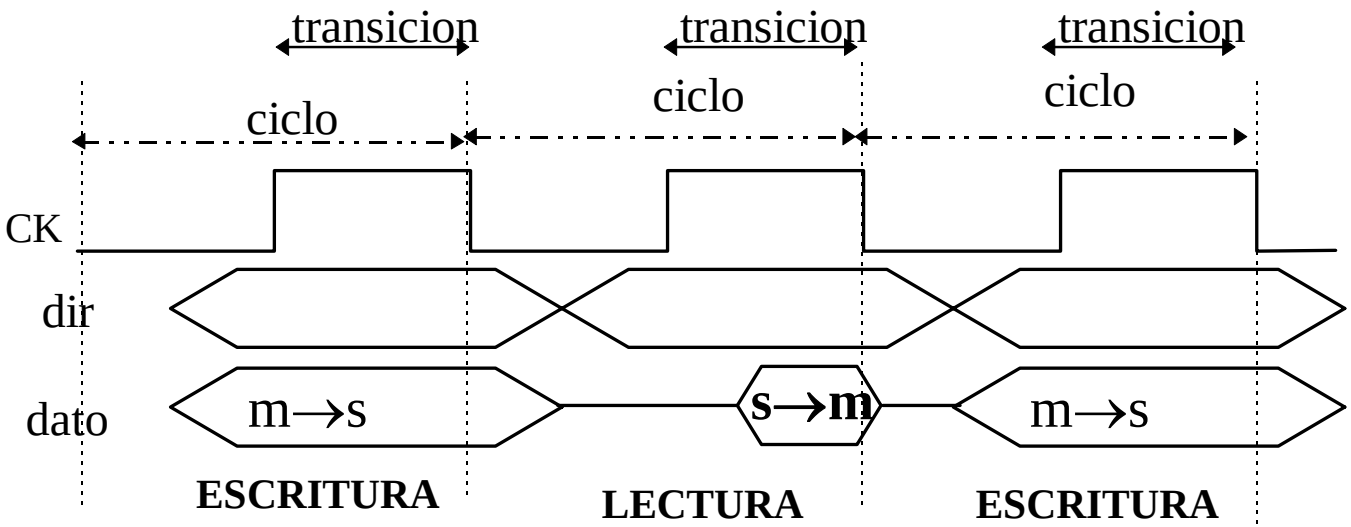
SINCRONIZACIÓN
PROTOCOLOS SINCRONOS

- Transferencia gobernada por una única señal CK
- Esta señal es compartida por todos los elementos
- Los flancos de subida y bajada determinan el comienzo y final de la transición
- Cada transferencia en un ciclo de CK



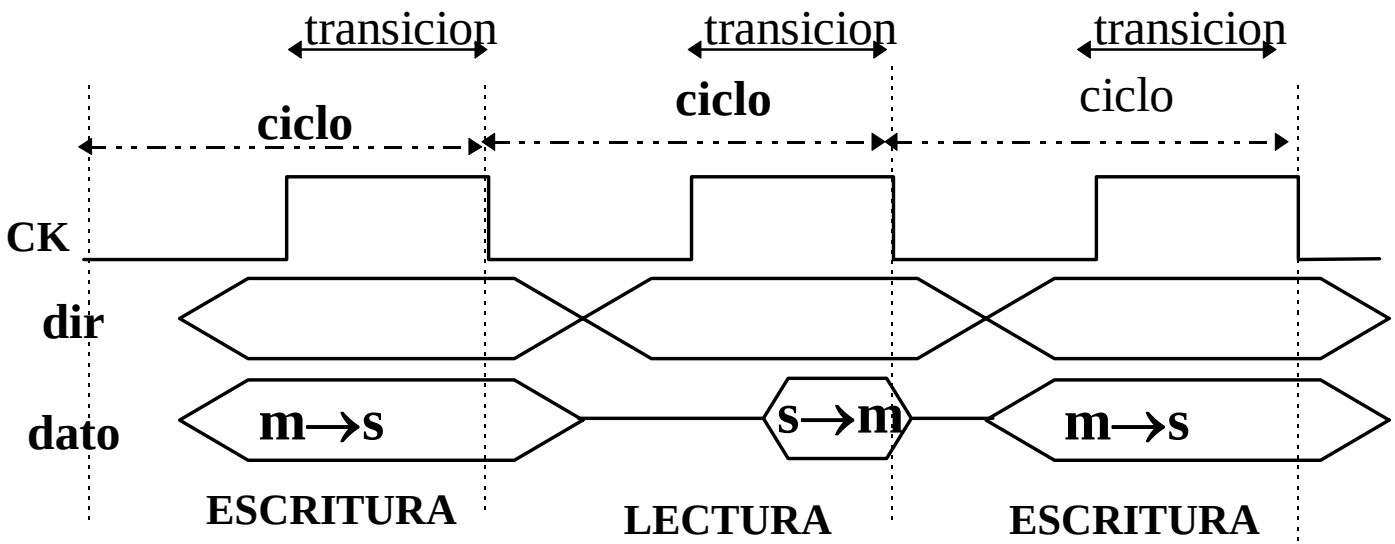
SICRONIZACIÓN
PROTOCOLO SINCRONO

- **Operación de escritura**
  - **El dato lo pone el maestro**
  - **Se debe dar la DIR y DATO un tiempo de setup antes que suba CK**
  - **Debe permanecer DIR y DATO un tiempo hold después que baja CK**



SINCRONIZACIÓN
PROTOCOLO SINCRONO

- **Operación de lectura:**
  - **El dato lo pone el esclavo**
  - **El Maestro pone la dirección**
  - **El DATO aparece en el Bus con retardo ((¿Tiempo de acceso?)).**
  - **La señal de CK debe estar en alta el tiempo necesario para que se produzca la transferencia**



SICRONIZACIÓN
PROTOCOLO SINCRONO

- **Tiempo de descodificación**
  - Tiempo necesario para que el esclavo descodifique la dirección
- **Tiempo de estabilización**
  - Set Up
  - Antes de aplicar el flanco de subida las señales deben estabilizarse y permanecer estables durante un intervalo de tiempo para asegurar su correcto almacenamiento
- **Tiempo de permanencia**
  - Hold
  - Después de aplicar el flanco de bajada las señales deben permanecer estables durante un intervalo de tiempo para asegurar su correcto almacenamiento.
- **Tiempo de desplazamiento relativo de las señales**
  - SKEW
  - Dos señales que parten de un emisor al mismo tiempo, pueden llegar en instantes de tiempo diferentes.
  - Se puede deber a :
    - » Diferentes niveles de puertas
    - » Diferentes tecnologías

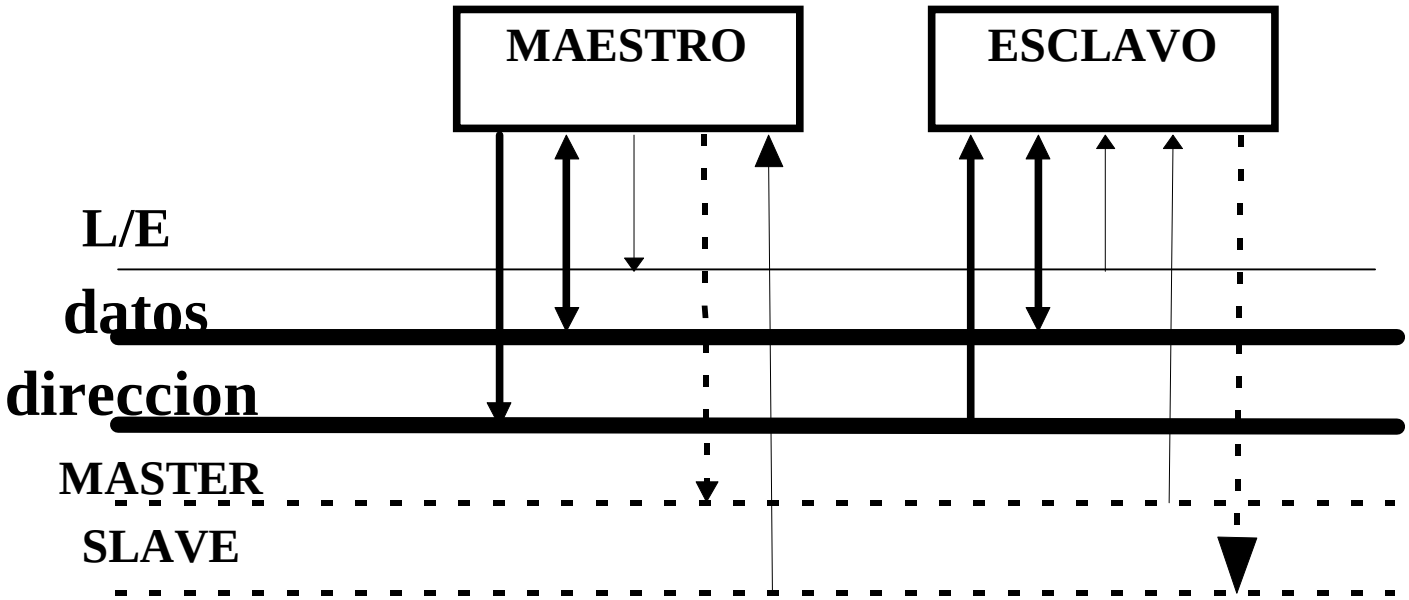
SINCRONIZACIÓN
PROTOCOLO SINCRONO

- **Ventajas:**
  - **Sencillez de diseño**
  - **Solo necesita una señal para realizar la sincronización**
  - **Mayor velocidad que asíncrono en según que casos**
    - » **grandes bloques de información**

- **Inconvenientes:**
  - **Inflexibilidad**
    - » **ciclo de reloj rígido**
    - » **La velocidad la marca el dispositivo más lento**
  - **No se sabe si se ha realizado bien la transferencia**

SINCRONIZACIÓN
PROTOCOLO ASÍNCRONO

- No utilizan CK
- Intercambian señales de control
- Siendo las señales MASTER y SLAVE las señales de sincronización

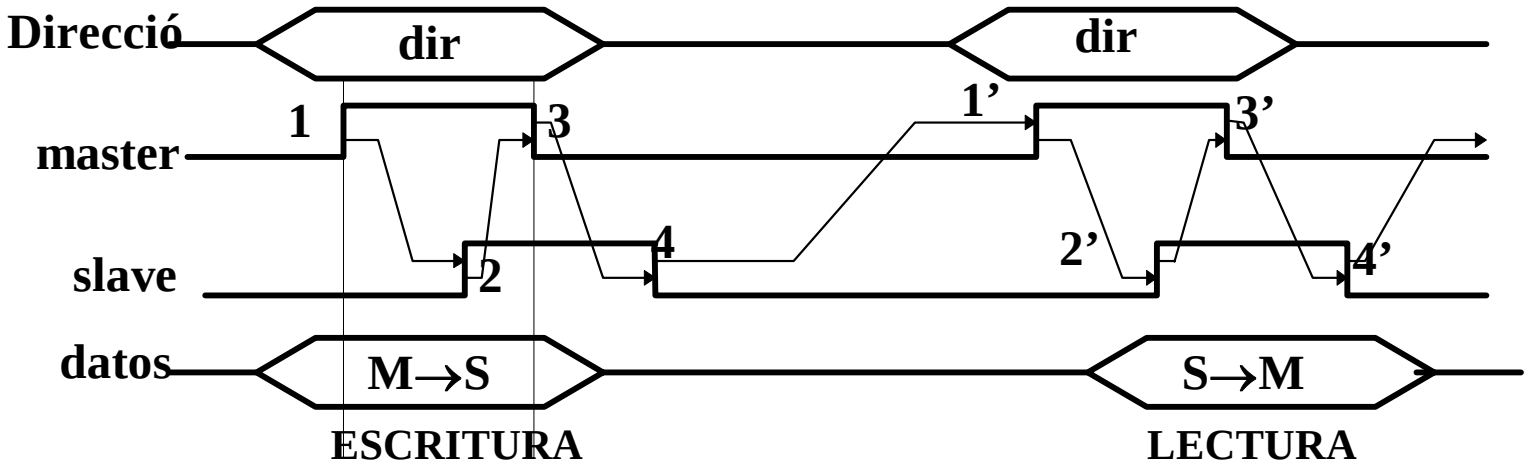


SINCRONIZACIÓN
PROTOCOLO ASINCRONO

- **Escritura:**
  - 1.- M a S: pongo DATO y DIR en el Bus
  - 2.- S a M: Lo he cogido.
  - 3.- M a S: Veo que lo has cogido
  - 4.- S a M: Veo que lo has visto

    queda libre el BUS para otra transmisión

- **Lectura**
  - 1'.- M a S: Quiero el DATO de la DIR
  - 2'.- S a M: Ya he puesto el DATO en el Bus
  - 3'.- M a S: Cojo el DATO
  - 4'.- S a M: Veo que lo has cogido

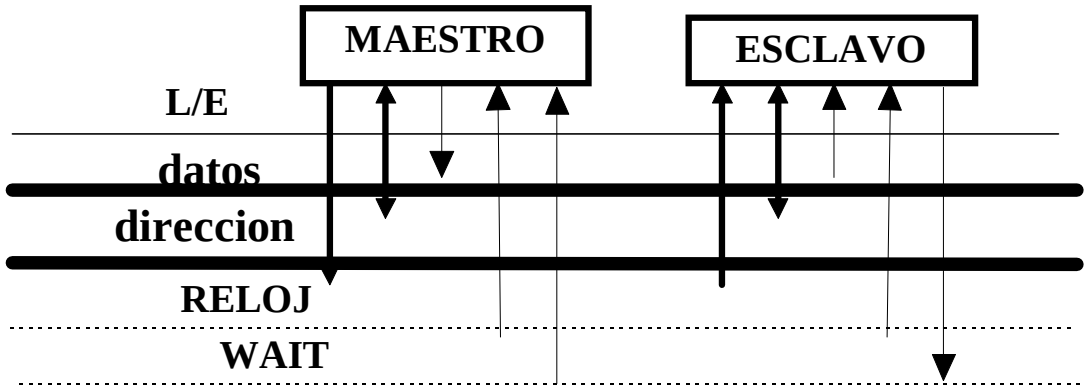


SINCRONIZACIÓN
PROTOCOLO ASINCRONO

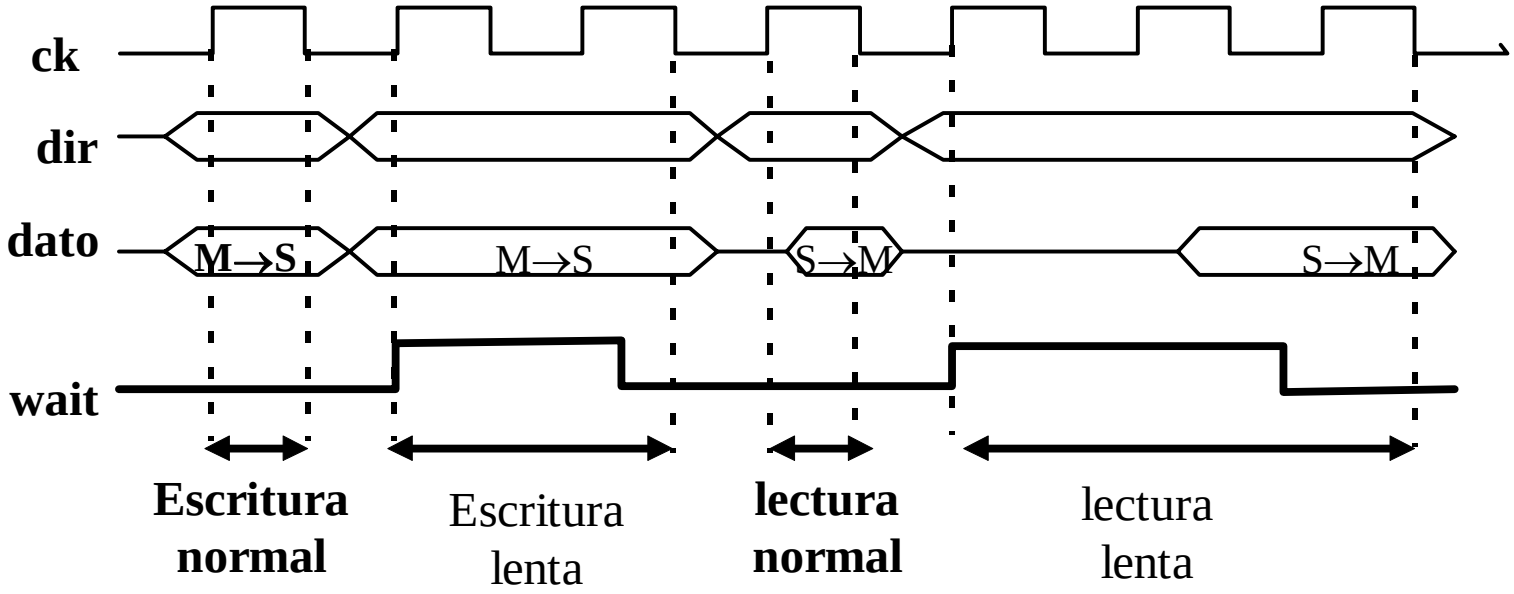
- **Protocolo totalmente interbloqueado:**
  - **A cada flanco del Master le sigue un flanco del Slave**
- **Ventajas:**
  - **Fiabilidad**
    - » **La recepción siempre se confirma**
  - **Flexibilidad**
    - » **Facilidad para conectar elementos de diferentes velocidades**
- **Inconvenientes:**
  - **Más lentos que los Síncronos (a igualdad de dispositivos)**
  - **Retardos en las líneas de gestión del protocolo**
- **Ejemplos:**
  - **Unibus PDP11**
  - **MC68000,10,20,30**
  - **Bus VME**
  - **FutureBus+**

SINCRONIZACIÓN
PROTOCOLO SEMISINCRONO

- Las transferencias pueden ocupar uno o varios ciclos de reloj
- Un bus ciclo con una línea llamada WAIT(que puede activar cualquier esclavo)
- Modo de operación
  - **Dispositivos rápidos operan como en un bus sícrono --> señal de reloj**
- Dispositivos lentos:
  - **Activan la señal wait congelando la actuación del master que no puede realizar ninguna otra operación hasta qu eno se desactiva**
  - **La transferencia ocupa mas de un ciclo**
- Mas flexible y rápido que los sincronos puesto que se adapta a las velocida de s de los dispositivos
- nota wait se debe activar antes que llegue la señal de reloj al master



SINCRONIZACIÓN
PROTOCOLOS SEMISINCRONOS



- **Ejemplos:**
  - **i80x86**
  - **MC68040**
  - **multibus ii**
  - **bus PCI**
  - **dec7000/10000 AXP**

SINCRONIZACIÓN
PROTOCOLO DE CICLO PARTIDO

- Mejora el rendimiento del bus en las operaciones de lectura
- Intenta aunar las ventajas de los síncronos y de los asincronos
- En los protocolos de ciclo completo el Bus queda ocupado en tanto se realiza la operación lectura
- se divide la operación de lectura en dos transacciones :
  - **maestro envía al esclavo la petición de lectura y deja el bus libre**
  - **cuando el esclavo dispone de los datos pasa a actuar como master y los envía al elemento solicitante que actúa como slave**
- Protocolos de ciclo partido:
  - **El tiempo del Bus se divide en una serie de fracciones (ranuras)**
  - **En cada ranura se permite enviar un mensaje**
  - **La duración de la ranura viene fijada por las características de transmisión del Bus,**
    - » **NO por el tiempo de respuesta de los dispositivos**

SINCRONIZACIÓN
PROTOCOLO DE CICLO PARTIDO

- **Inconvenientes:**
  - **Lógica de acceso al bus más compleja**
  - **Necesidad de incluir protocolos de arbitraje**
  - **El tiempo para completar una transferencia crece**
    - » **hay que conseguir dos veces el bus**
  - **Protocolos mas caros y difíciles de implementar**
    - » **debido a la necesidad de seguir la pista a la otra parte de la comunicación**
- **Ventajas:**
  - **al liberar el bus se permite que otro peticionario lo utilice**
  - **Mejora de la anchura de banda efectiva del bus si la memoria es lo suficientemente sofisticada para manejar múltiples transacciones solapadas**

ranura	ranura
M petición de lectura pone la dirección	**Bus libre**

ranura	ranura
Bus libre	**Esclavo solicita la ranura y pone el dato**

CLASIFICACIÓN DE BUSES

- **Clasificación de Petterson-Hennesy :**
  - **Bus procesador-memoria**
  - **Bus de plano posterior**
  - **Bus de entrada/salida**

- **Características del Bus procesador - memoria**
  - **Cortos**
  - **De alta velocidad**
  - **Adaptados al sistema de memoria para maximizar la anchura de banda memoria procesador**
  - **Diseño exclusivo**
  - **Al diseñarse se conocen perfectamente todos los dispositivos que van a conectar**
  - **Suelen estar patentados**
  - **En la actualidad suelen conectarse a los buses**
    - » **De plano posterior**
    - » **De E/S**

CLASIFICACIÓN DE BUSES

- **Bus de plano posterior**
  - **Permiten que coexistan en un bus**
    - » **La memoria**
    - » **El procesador**
    - » **Los dispositivos de E/S**
  - **Necesitan lógica adicional para conectarse**
    - » **Al buses de entrada /salida**
    - » **Dispositivo**
  - **En los Macintosh el bus que conecta el procesador y la memoria es un bus de plano paralelo llamado NUBUS.**
  - **A este bus se puede conectar directamente un bus SCSI de entrada/salida**
  - **Suelen ser estándares**

CLASIFICACIÓN DE BUSES

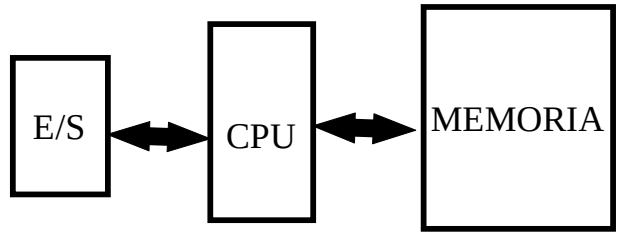
- **Bus de entrada/salida**
  - **Largos**
  - **Muchos tipos de dispositivos**
  - **No tienen una interfaz directa con la MP**
  - **Se conectan a la MP mediante**
    - » **Buses de plano posterior**
    - » **Buses de procesador memoria**
  - **Interfaz sencilla**
  - **Los dispositivos necesitan pocos elementos electrónicos para conectarse a él**
  - **Suelen ser estándares**

CLASIFICACIÓN DE BUSES

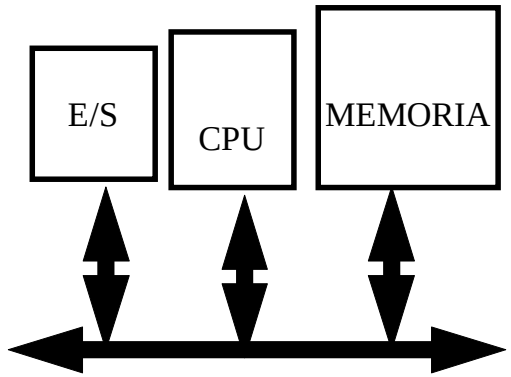
- La existencia de diferentes tipos de buses tiene notable ventajas:
- un bus para cada necesidad
- ejemplo:
  - Bus procesador memoria mucho mas rápidos que uno de plano posterior
  - Sistema de E/S expandible con facilidad
    - » Permite conectar muchos controladores o buses al plano posterior
    - » La expansión no afecta al rendimiento del bus de procesador-memoria

ESTRUCTURAS DE BUS

- **De un bus**
  - **Bajo coste**
  - **Flexibilidad para conectar nuevos módulos**
  - **velocidad de operación baja**
  - **máquinas pequeñas**
  - **Estas estructuras de Bus son demasiado generales y se dan pocas veces**

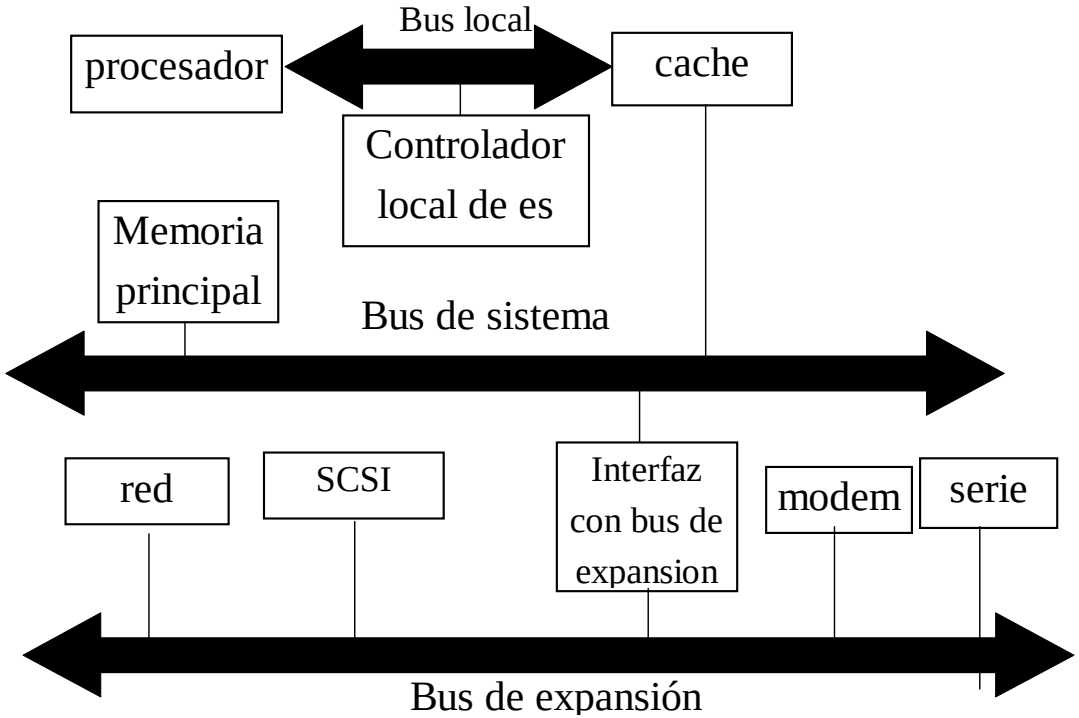


- **problemas:**
  - **Si se conecta un gran nº de dispositivos al bus las prestaciones disminuyen debido al mayor retardo de propagación**
  - **Este retardo determina el tiempo necesario para coordinar el uso del bus**
  - **si el control pasa frecuentemente de un dispositivo a otro, empeoran notablemente las prestaciones**
  - **El bus se convierte en un cuello de botella**



- **Solución**
  - **MULTIBUS cuya objetivo es la Aceleración del sistema**

ESTRUCTURAS DE BUS
MULTIBUS TRADICIONAL



- **Bus local que conecta el procesador a la memoria cache**
- **Bus de sistema conecta procesador y memoria principal a través de la cache**
- **Bus de expansión une los dispositivos de e/s con el bus de sistema a través de la interfaz**
  - **La interfaz del bus de expansión controla las transferencias entre los dispositivos de entrada salida y el bus de sistema**

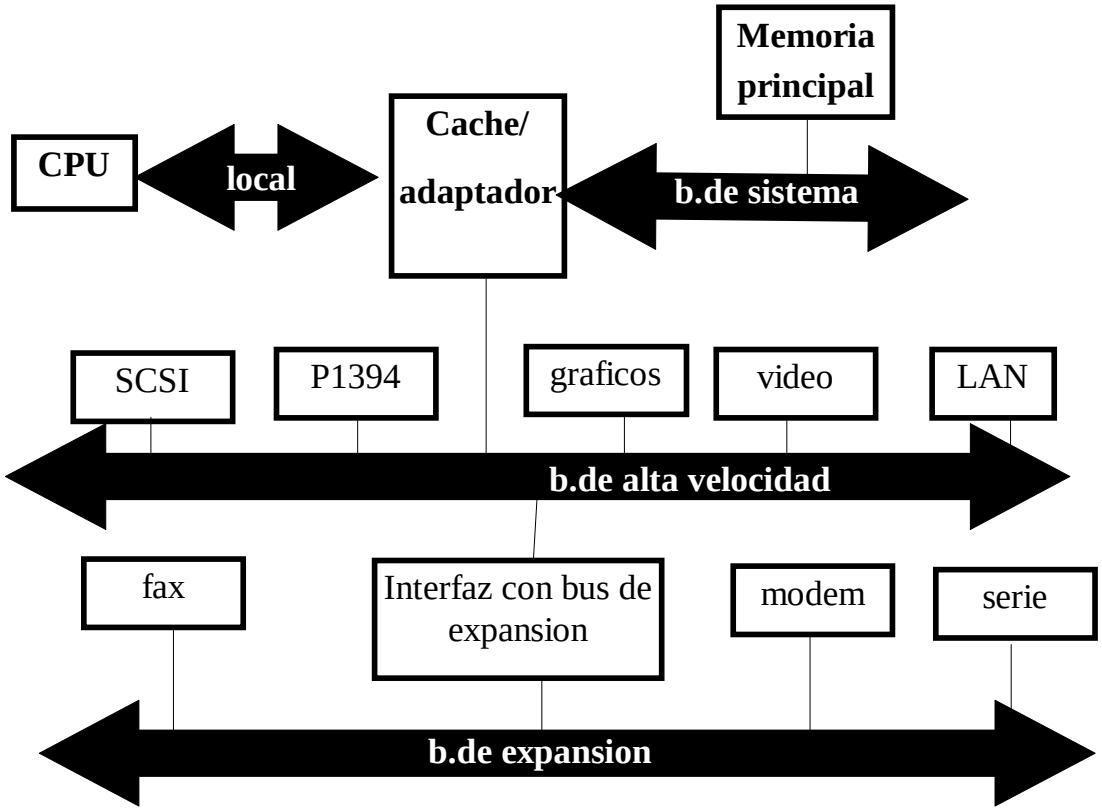
ESTRUCTURAS DE BUS
MULTIBUS TRADICIONAL

- **Ventajas**
  - **Se conecta gran variedad de dispositivos al sistema**
  - **Se aísla el trafico de información entre**
    - » **la MP y el procesador**
      - **El uso de cache alivia los accesos del procesador a la memoria principal**
    - » **la MP y la entrada/ salida**
      - **las E/S no interfieren en la actividad del procesador**
- **Inconvenientes:**
  - **Al mejorar las prestaciones de los dispositivos de E/S se degrada la eficiencia**
- **Solución**
  - **un bus de alta velocidad altamente integrado al sistema**

ESTRUCTURAS DE BUS

MULTIBUS DE ALTAS PRESTACIONES

ARQUITECTURA ENTRESUELO(MEZZANINE)

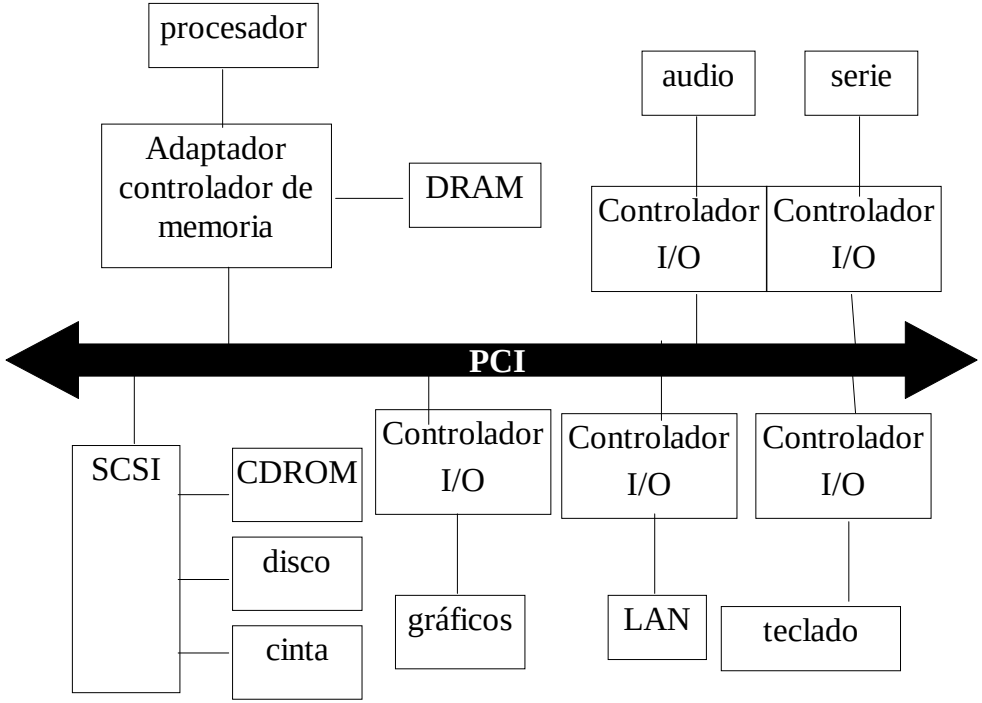


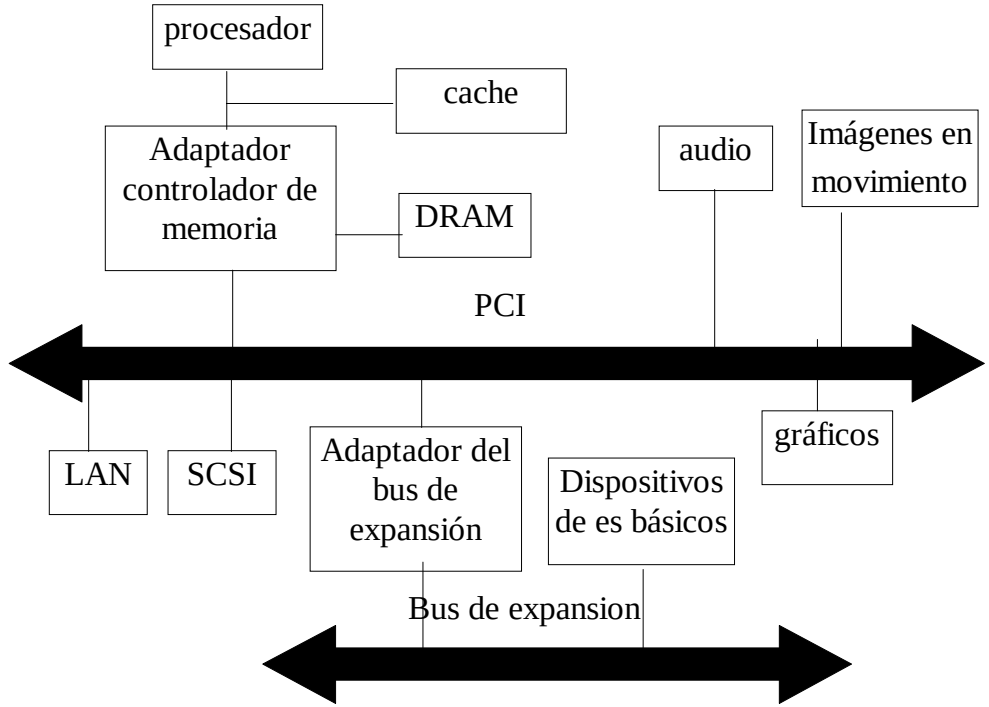
- **bus de alta velocidad entre bus de sistema y el de expansión**
  - **Al bus de alta velocidad se conectan todos los periféricos de altas prestaciones**
  - **bus diseñado para dispositivos de ES de alta velocidad**
- **Los dispositivos de menor velocidad se conectaran el bus de expansión**

PCI
Peripheral component interconnect

- Bus de plano posterior
- Ancho de banda elevado
- Independiente del procesador
- Se puede usar como bus de periférico o bus de alta velocidad
- Prestaciones para sistemas de entrada/salida de alta velocidad
  - Adaptadores de pantalla gráfica, controladores de interfaz de red
  - 32 o 64 líneas de datos
  - Velocidad de transferencia de 264Mbytespso 2.11gbps
- Su principal ventaja:
  - Diseñado par ajustarse económicamente a los requisitos de E/S de los sistemas actuales
  - Se implementa con pocos circuitos integrados y permite la conexión a otros buses.
- Historia
  - Comenzó INTEL en 1990 sistemas basados en pentium.
  - Patentes de dominio público y promueve la sociedad PCI SIGha
  - ampliamente adoptado por:
    - » Computadores personales
    - » Estaciones de trabajo
    - » Servidores

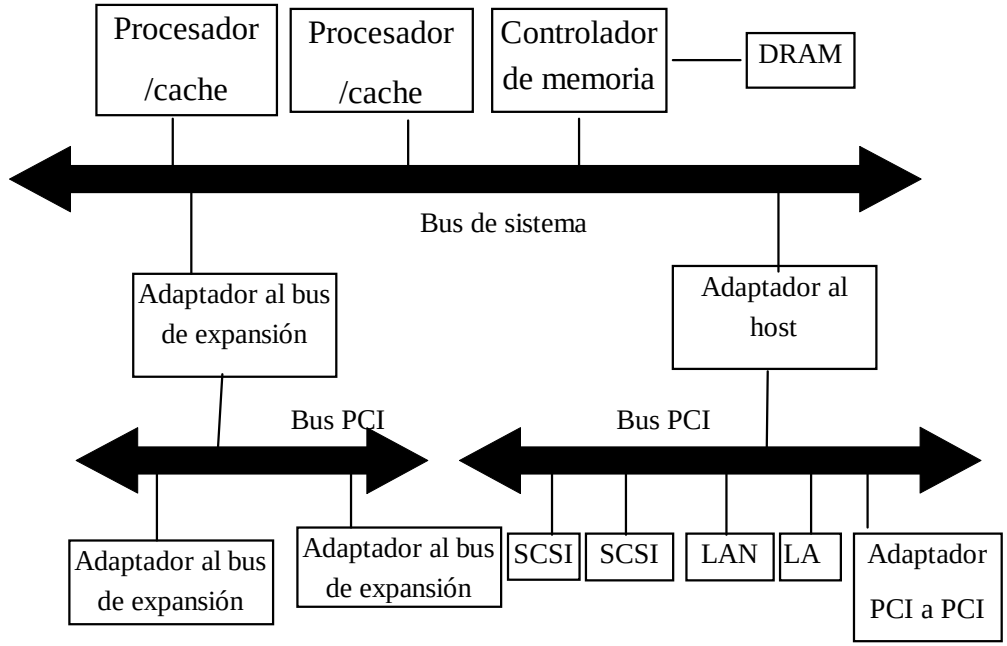
PCI
Sistema de sobremesa uniprocesador





- Se usa com o bus de alta velocidad
- se conecta con la memoria y el procesador a través de un adaptador
  - **Funciona como buffer**

PCI
Sistema multiprocesador



- Cada procesador tiene una cache local
- Una memoria principal común a todos los procesadores
- El bus de sistema conecta todos los procesadores y la memoria entre si
- Los buses pci se conectan al bus de sistema mediante adaptadores:
  - Independencia de la velocidad
- Cada estructura de pci tiene una misión diferente:
  - Una conecta a varios buses de expansión al PCI mediante adaptadores
  - A la otra se conectan todos los periféricos rápidos

LA UNIDAD ARITMETICO LOGICA

TEMA 5

LA UNIDAD ARIMÉTICO LÓGICA

- **Es la unidad encargada de tratar los datos ejecutando la operaciones determinadas por la instrucción en curso**

- **La Unidad de control se encarga de mandarle los datos e indicarle la operación a realizar**

- **Sumador-restador**
  - **La mayoría de los computadores basan su UAL en un sumador-restador**
  - **Implementación de operaciones complejas**
    - **Se descomponen en pasos elementales de sumas y restas que se ejecutan secuencialmente**

REPRESENTACIÓN DE ENTEROS

- **Se estudian cuatro representaciones**
  - **Binario puro**
    - » **Enteros sin signo**
  - **Magnitud y signo**
    - » **Enteros con signo**
  - **Complemento a uno**
    - » **Enteros con signo**
  - **Completo a dos**
    - » **Enteros con signo**

BINARIO PURO

- **Un decimal se descompone:**

  $2^{N-1}b_{n-1}+2^{n-2}b_{n-2}+ ... + 2^2B_2+2^1b_1+2^0b_0$

  Pudiendo valer b=0 , o b=1

- **La representación binaria**

  $B_{n-1}, b_{n-2}, ...B_2 ,b_1 ,b_0$

- **Ejemplo**

  $111_{(2} = 2^2 \cdot 1+ 2^1 \cdot 1+2^0 \cdot 1= 7_{(10}$

- **Rango N bits**
  - **[0, 2^{N-1}]**

- **Resolución**
  - **1 Unidad**

- **Desventajas:**
  - **Desbordamiento**
    - » **El resultado necesita N+1 bits**
    - » **Acarreo en el bit N**
  - **No representa Nº negativos**
    - » **En A-B se debe comprobar que A>B**
    - » **La resta incorrecta genera bits de desbordamiento**
  - **El producto pude necesitar 2N bits**

MAGNITUD Y SIGNO

- Para N bits [N-1,0]
- Bits de magnitud [N-2,0]
- Bit de signo [N-1]
  - Nº positivo. Bit de signo = 0
  - Nº negativo. Bit de signo = 1
- Ejemplo
  - +7 = 0111
  - -7 =1111
- Rango para N bits
  - $[- (2^{N-1} - 1), 2^{N-1} - 1]$
- Resolución
  - 1 unidad
- El cero
  - 0...0
  - 1...0

- Sumas y restas:
  - Analisis de los signos
- Multiplicación y division :
  - Se operan por un lado las magnitudes
  - Por el otro los signos
- Inconvenientes:
  - Probabilidad de desbordamiento
  - La ambigüedad del cero
    - » Complica la detección de negativos
    - » No es suficiente analizar el bit más significativo
  - La operación de suma o resta depende de los operandos

COMPLEMENTO A DOS (C'2)

- **Números positivos y negativos**
- **El bit de signo no es independiente de los de magnitud,**
  - Las operaciones se realizan con los N bits
- **Los números positivos**
  - Bit más significativo = 0
  - Se representan en binario puro
  - Rango es [0, 2^{N-1}-1]
  - Ej +7 =0001
- **Los números negativos**
  - Bit más significativo a uno
  - Se hallan mediante el C'2 del positivo
  - Rango de negativos [-2^{N-1} , 0]
  - Rango total [-2^{N-1}, 2^{N-1}-1]

- **La operación de C'2**
  - para un A representado con N bits
  - restar a 2^N la magnitud del número a representar

    $2^N-A = - A$
  - Ejemplo
    - » para A=2 y N=3
    - » 2^N =1000 y A=010
    - » 1000-010=110
- **Ventajas:**
  - Sólo existe un 0
  - Se puede operar sin tener en cuenta el signo del operando

COMPLEMENTO A DOS

- **¿Para qué sirve la operación de C'2? Cambiar de signo:**
  - **Si es A positivo:**
    - **$-A = 2^N - A$**
    - **A =3 , -3= 1000-011=101**
  - **Si A es negativo:**
    - **$-A= 2^N - A$**
    - **$A = 2^N - (2^N - A) = A$**
    - **-2=110, 2=1000-110=010**
- **A+C con A<0 y C<0:**
  - **(Demuestra que se pueden sumar directamente Nª negativos**
  - **$(2^N - A) + (2^N - C) = 2^N + (2^N - A - C)$**
  - **La expresión buscada es $2^N - A - C$**
  - **La expresiones sólo difieren en el término 2^N**
    - **Es un bit que ocupa la posición N+1**
    - **Aparece como un acarreo C_{N-1}**
    - **Basta con ignorarlo para que el resultado sea correcto**
  - **Desbordamiento.**
    - **Cuando $A + C > 2^{N-1}$**

COMPLEMENTO A DOS

- **A+C con A>0 C<0 :**
  - **si |A| <|C|**
    - **resultado negativo**
    - **A + (2^N - C) = 2^N - (C-A)**
    - **Se obtiene el resultado directamente**
    - **a 2^N se le resta una cantidad positiva--> no hay acarreo**
  - **si |A| > |C|**
    - **Resultado positivo**
    - **A + (2^N - C) =2^N +(A-C)**
    - **a 2^N se le suma una cantidad positiva--> no se puede representar con N bits:**
      - **Acarreo en bit N+1**
      - **Ignorando ese bit se obtiene la representación correcta.**
- **Permite sumar Nº de distinto signo sin problemas**
  - **no se tiene que estudiar la operación a realizar**
  - **siempre suma**

COMPLEMENTO A DOS

- **C'2 rápido**
  - **Complemento binario + 1**
- **Extensión de signo**
  - **Número >0**
    - » **Se añaden Os a la izq.**
  - **Número <0**
    - » **Se añaden 1s a la izq.**

- **Caracteristicas:**
  - **Sumas y restas sin tener en cuenta los signos de los operandos**
  - **No confundir**
    - » **Acarreo**
    - » **Desbordamiento**
  - **Se complica la multiplicación**
  - **El C'2 exige una suma o resta**
  - **Rango asimétrico:**

    $[-2^N-1, 2^{N-1}-1]$
  - **Representación del 0 única.**
  - **Resolución es 1**

COMPLEMENTO A UNO

- Numeros positivos y negativos
- El bit de signo el más significativo
  - 0 Positivo
  - 1 Negativo
- El negativo se representa haciendo 2^N -1 - A
- La negación mediante el complemento lógico
- A+C con A<0 y C<0
  - $(2^N - 1 - A) + (2^N - 1 - C) = 2^N - 1 + [2^N - 1 - (A+C)]$
  - El resultado correcto es $2^N - 1 - (A + C)$
  - El resultado obtenido da
    - Un bit de acarreo superior 2^N
    - Un resultado una unidad inferior a lo deseado (-1)
  - Solución:
    - Reciclar el bit de acarreo, sumándoselo al resultado de la suma

COMPLEMENTO A UNO

- **A+C con A>0 y C<0**
  - **A<C**
    - **A + (2^N -1 - C)**
    - **El resultado debe ser negativo**
    - **Se genera directamente**
    - **R = 2^N- 1 -(C - A)**
  - **A >C**
    - **A + (2^N - 1 - C)**
    - **R = 2^N + (A - C - 1)**
    - **Siendo el resultado correcto A-C**
      - **Existe un acarreo del bit superior**
      - **Al resultado le falta un 1**
    - **Solución:**
      - **Reciclar el acarreo sumándoselo al resultado**

COMPLEMENTO A UNO

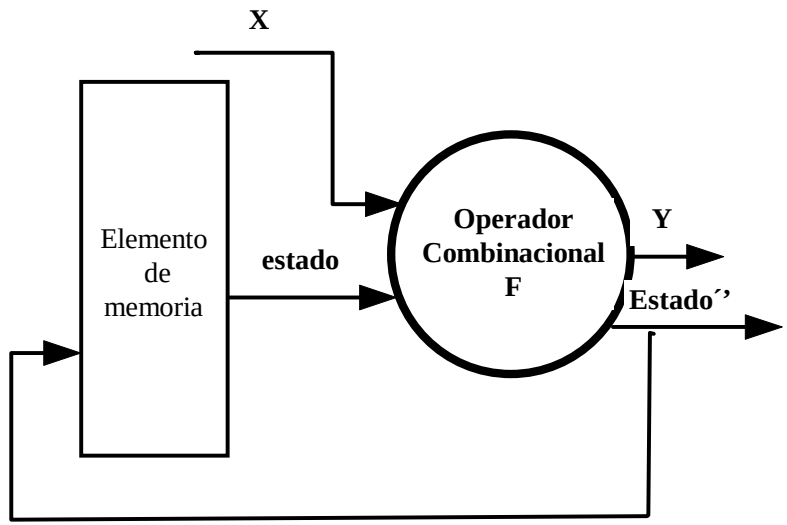
- **Características:**
  - **El C'1= al complemento lógico**
  - **Al sumar cuando aparece un acarreo en el bit de mayor peso se debe incrementar en una unidad el resultado**
  - **Se complica la multiplicación y la división**
  - **Se puede producir desbordamiento**
  - **Rango simétrico:**
  - **$[-(2^{N-1} - 1),\ 2^{N-1} - 1]$**
  - **Dos representaciones del cero**
  - **La Resolución = 1**
  - **Extensión de signo**
    - **Repetir el bit de la izquierda**

OPERADORES COMBINACIONALES

- **COMBINACIONAL:**
  - **No tiene elementos de memoria**
  - **Y=F(x)**
  - **Si se modifica Y, se modifica X**
  - **El tiempo es la suma de los retardos**
  - **Si se quiere guardar el resultado se debe utilizar un elemento de memoria**

OPERADORES SECUENCIALES

- **SECUENCIAL:**
  - **Y=F(X,Estado)**
  - **Varias fases para llegar al resultado final**
  - **Necesita memoria para almacenar la información que se transmite entre fases**
    - » **Un contador de fases**
  - **Se necesita un algoritmo para calcular el resultado**
  - **Debe contener las fases necesarias y la función a realizar en cada una de ellas**

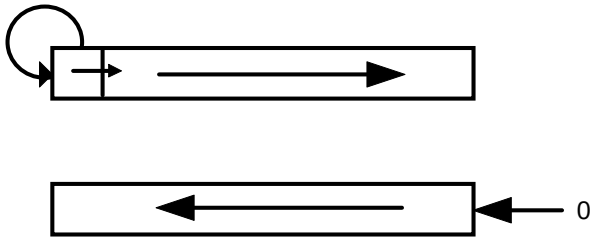


OPERACIONES DE DESPLAZAMIENTO

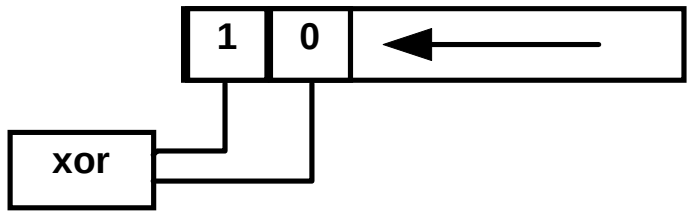
- Correr los bits de una palabra hacia la izquierda o hacia la derecha
- Los computadores sencillos sólo permiten desplazar un posición
- Máquinas complejas, desplazamientos múltiples:
  - Mediante un c.combinacional específico
  - Varios desplazamientos unitarios
- La forma en que se tratan los extremos del desplazamiento permite diferenciar tres tipos de desplazamiento:
  - Lógicos
  - Circulares
  - Aritméticos
- Desplazamiento lógico
  - Los valores extremos se completan con O’s

DESPLAZAMIENTOS

- **Desplazamientos ariméticos:**
  - **A la izquierda Multiplicar por dos**
  - **A la derecha dividir por dos**
  - **Se debe conservar el signo del operando**
- **En complemento a dos**
  - **Izquierda**
    - » **Mete 0's por la derecha**
  - **A la derecha**
    - » **Mete el signo por la izquierda**
  - **El desplazamiento a la izquierda puede producir desbordamiento**
    - » **110 ;100; 000 Desbordamiento**
    - » **001; 010; 100 Desbordamiento**
    - » **Detección de desbordamientos**
      - **Bits [N-1] y [N-2] diferentes**
      - **Un desplazamiento más pone un 0 en [N-1] cambiando el signo del número**

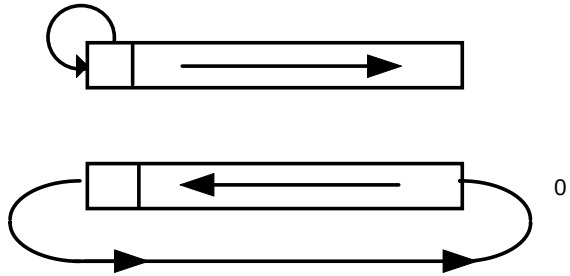


DESPLAZAMIENTOS A LA DERECHA Y A IZQUIERDA EN C2

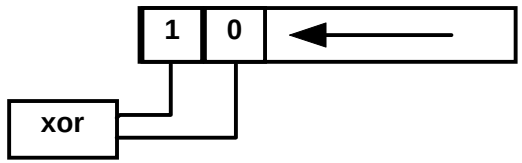


DESPLAZAMIENTOS

- **Desplazamientos en C'1**
  - **a la derecha**
    - » **se mete es el signo**
  - **a la izquierda lo que**
    - » **se mete es el signo**
  - **El desplazamiento a la izquierda puede producir desbordamiento**
    - » **110 ;101; 011 Desbordamiento**
    - » **001; 010; 100 Desbordamiento**
    - » **Detección de desbordamientos**
      - **Bits [N-1] y [N-2] diferentes**
      - **Un desplazamiento más pone un 0 en [N-1] cambiando el signo del número**

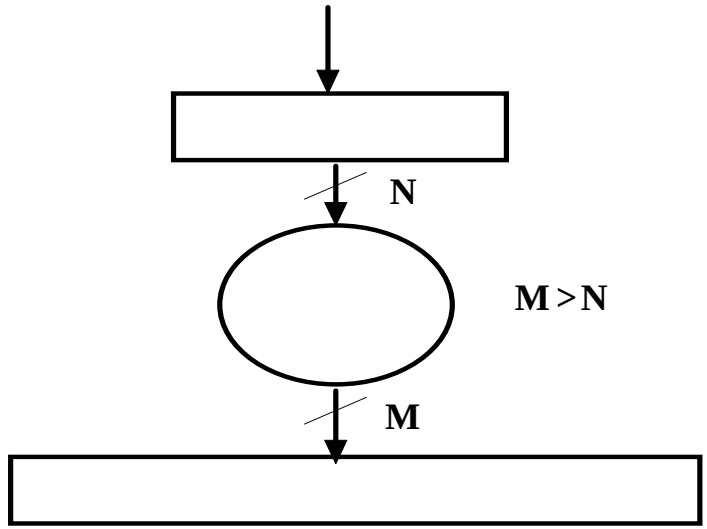


DESPLAZAMIENTOS A LA DERECHA Y A IZQUIERDA EN C1



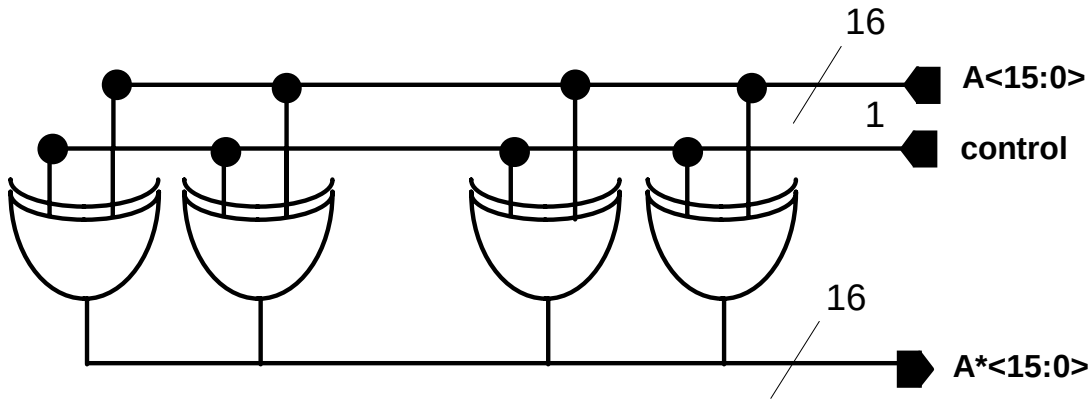
OPERACIONES DE EXTENSIÓN DE SIGNO

- Se transfiere un operando de N bits, a un elemento de M bits
- Si no se rellenan correctamente los bits, se cambia el valor numérico
- Signo y magnitud:
  - Se desplaza el bit N-1 a la posición M-1
  - El resto de los bits se llenan con 0's
- C'1 y C'2:
  - Se repite el bit de signo hasta llenar todos los bits



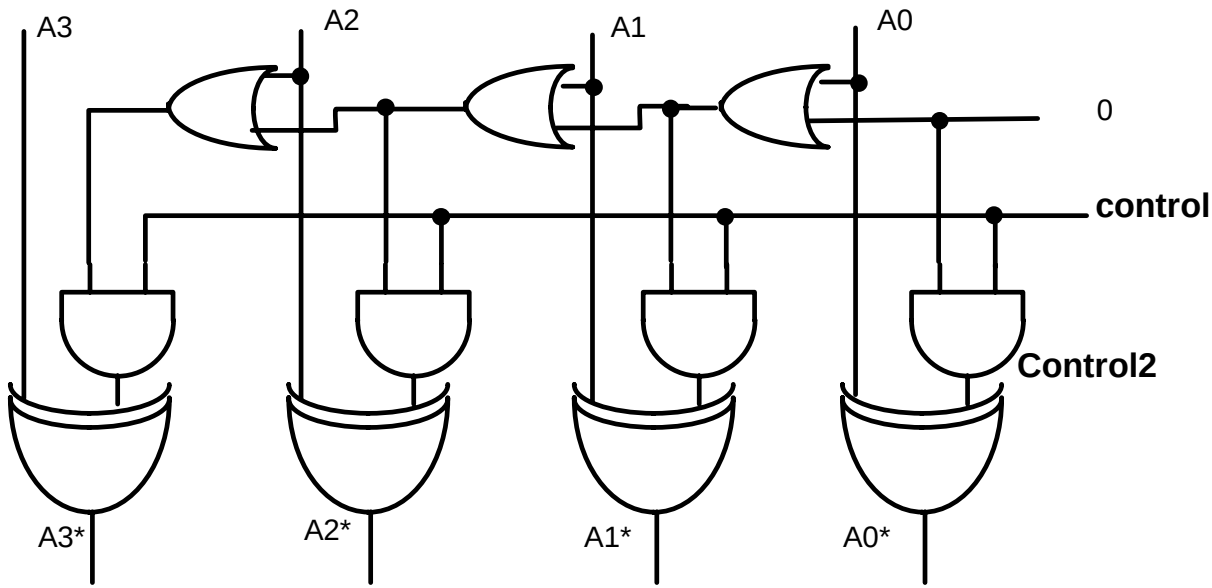
OPERACIONES DE CAMBIO DE SIGNO
DE COMPLEMENTO A UNO

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0



CAMBIO DE SIGNO
COMPLEMENTO A DOS

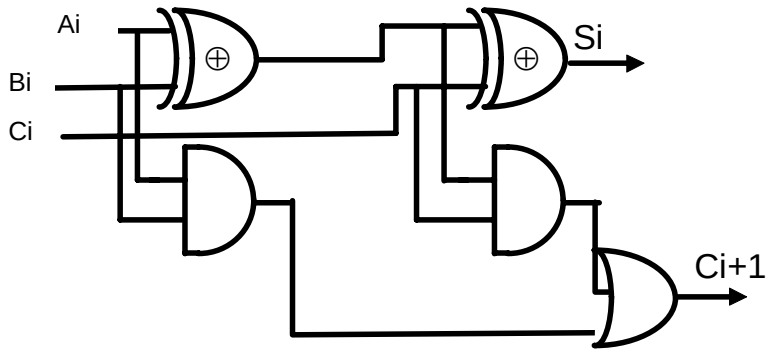
- **Algoritmo**
  - **todos los bits son iguales hasta que se encuentra el primer 1**
  - **a partir de este momento se complementan todos**

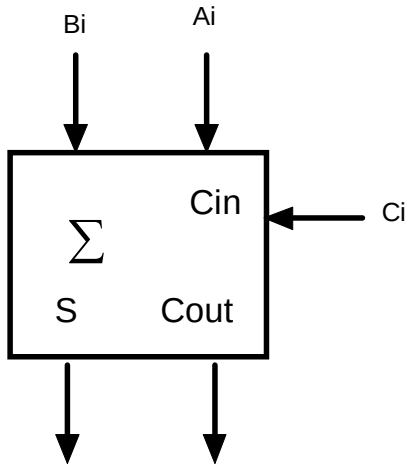


SUMADOR

- $Si=A_i \text{ xor } B_i \text{ xor } C_{in}$
- $C_{out}=A_iB_i+B_iC_{in}+A_iC_{in}$
- **Semisumadores**
- **Implementa:**
  - **Sumador paralelos**
  - **Sumador secuencial**

Ai	Bi	Cin	Si	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

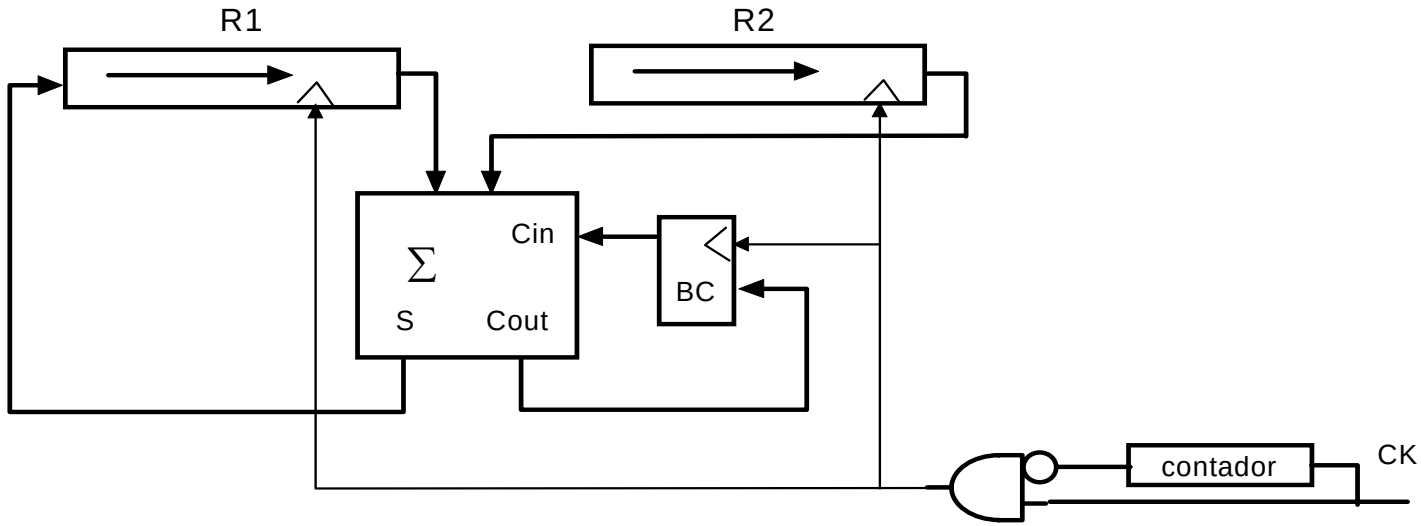




SUMADOR SECUENCIAL

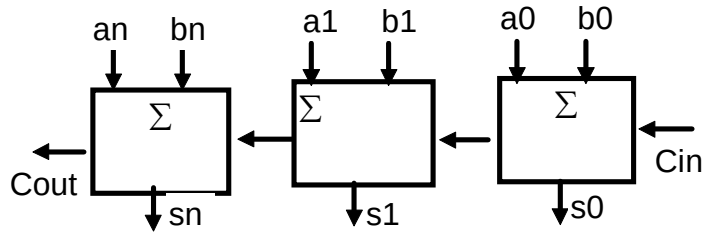
- R1:=R1+R2
- Algoritmo
  - R1:=A;R2:=b;BC:=0
  - R1[3]:=R1[0]+R2[0]
  - BC:=Cout
  - Desp(R1).dch
  - Desp(R2).dch

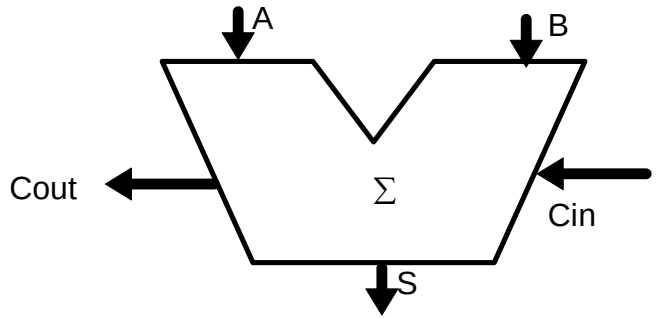
- Los registros permiten desplazamientos a la derecha
- El contador con el número de fases
- Ventaja:
  - Barato en área
- Desventaja:
  - Bajo rendimiento

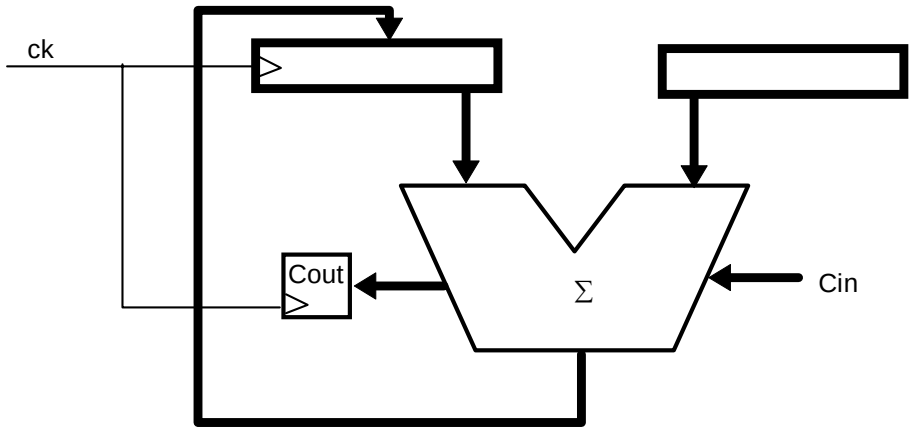


SUMADOR PARALELO

- **Cin =F(representación usada)**
  - **Para binario sin signo Cin = 0**
- **El Cout detecta el desbordamiento**
- **El retardo**
  - **El máximo número de puertas que atraviesan las señales**
  - **El camino más largo es el de los Acarreos**

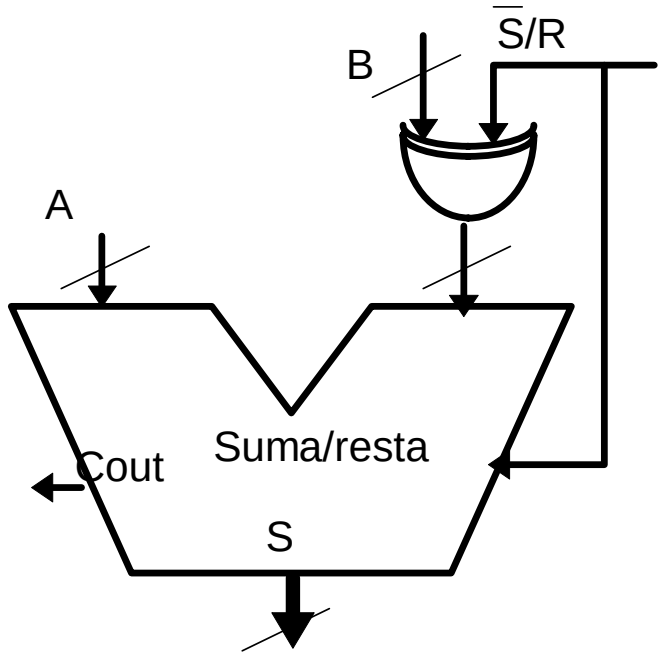






SUMADOR/RESTADOR GENERICO

- **A positivo B positivo**
- **la Resta que se quiere hacer:**
  - **$R=A - B = A - B + 2^n - 2^n = A + (2^n - B) - 2^n$**
  - **donde $2^n - B$ es el C'2 de B**
    - **notB+1**
- **Resta que hace el S/R generico**
  - **$R'=A + (2^n - B)$**
  - **$R=R' - 2^n$ --> restar 1 al bit Sn que no existe**
    - **ignorar el bit de acarreo Cout**
- **Bloque de construccion del resto de S/R**
  - **Hay que particularizar el desbordamiento para la representación que se utiliza**



SUMADOR RESTADOR
BINARIO SIN SIGNO

- **A+B**
  - **Sumador en binario puro**
  - **El desbordamiento de la suma Cout = 1**

- **A-B :**
  - **Restador en C'2**
  - **<u>Para A > B</u>**
    - **Operación correcta en binario sin signo**
    - **En C'2 $A+2^N-B= 2^N+(A-B)$--> Cout =1**
  - **<u>Para A < B</u>**
    - **Desbordamiento --> no existen nº negativos**
    - **en C'2 $A+2^N-B= 2^N-(B-A)$-->Cout=0**
    - **El desbordamiento de la resta Cout=0**

- **Desbordamiento:**
  - **DE=c_{out}xor(control)'**

SUMADOR/RESTADOR
COMPLEMENTO A DOS

- En cantidades de diferente signo NUNCA hay desbordamiento
- Desbordamiento de la suma de cantidades del mismo signo

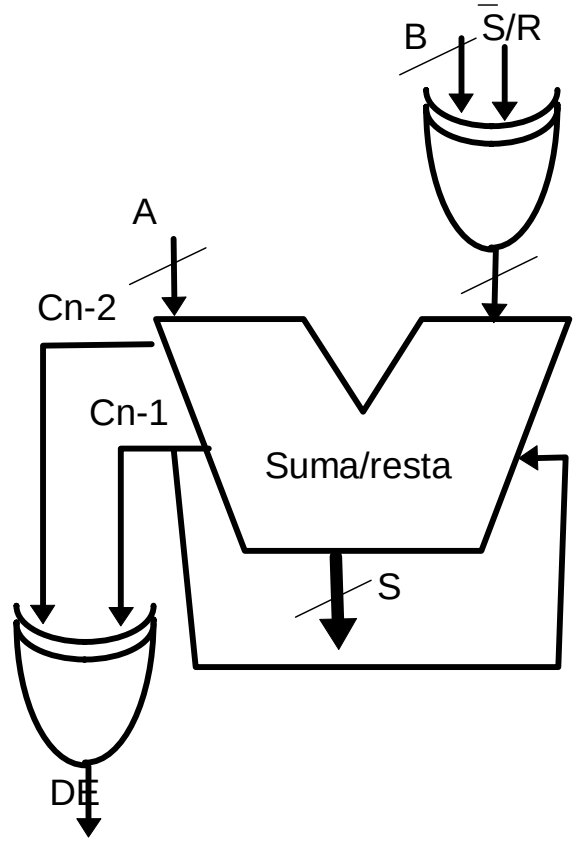
- **Suma de dos positivos**
  - Si $A_{N-1} = 0$ y $B_{N-1} = 0 \rightarrow C_{N-2} = 1$ (que es lo mismo que $S_{N-1} = 1$)
  - La suma de dos números positivos no puede ser un resultado negativo
  - $DE = A'_{N-1} \cdot A'_{N-1} \cdot C_{N-2}$

- **Suma de cantidades negativas:**
  - Si $A_{N-1} = 1$ y $B_{N-1} = 1 \rightarrow C_{N-2} = 0$ (que es lo mismo que $S_{N-1}=0$)
  - La suma de dos cantidades negativas no puede dar una positiva
  - $DE = A_{N-1} \cdot B_{N-1} \cdot C'_{N-2}$

- El desbordamiento total será:
  - $A'_{N-1} \cdot B'_{N-1} \cdot C_{N-2} + A_{N-1} \cdot B_{N-1} \cdot C'_{N-2}$
  - Que se puede simplificar $DE = C_{N-1}$ xor C_{N-2}

SUMADOR-RESTADOR
COMPLEMENTO A UNO

- **No se utiliza el esquema general**
- **Hay que introducir modificaciones:**
  - **La suma de dos números en C'1 se realiza en dos pasos:**
    - **Se suman A+B**
    - **Se suma el carry al resultado inicial**
      - **A+B+Cout**
  - **La negación del C'1 sólo exige la negación lógica**
  - **Las condiciones de desbordamiento se obtienen con un razonamiento idéntico al C'2**



SUMADORES DE ACARREO ANTICPADO

- **En un sumador normal el acarreo debe atravesar N puertas**
  - **Esto provoca importantes retrasos**
- **Aceleración del proceso:**
  - **Función generadora de acarreo G:**
    - » $G_I = A_I \cdot B_I$
    - » **Indica si se genera acarreo en el sumador i**
  - **Función propagadora de acarreo P**
    - » $P_I = A_I + B_I$
    - » **Indica si se propaga el acarreo al siguiente sumador**
  - **Entonces**
    - » $C_i = C_{i-1} \cdot P_i + G_i$
    - » $S_i = P_i + C_{i-1}$
- **Si se desarrolla C_i se ve que**
  - **Aumenta el número de puertas por nivel**
  - **NO aumenta el número de niveles que es siempre DOS**

MULTIPLICADOR DE ENTEROS SIN SIGNO
OPERADOR SECUENCIAL

- **Se necesita :**
  - **Sumador/restador**
  - **Un algoritmo**

- **Las máquinas muy potentes circuito combinacional**

- **A[N-1...0] X B[N-1...0] = C[2N-1...0]**

- **Algoritmos típicos**
  - **Enteros sin signo:**
    - » **Suma y desplazamientos**
    - » **Suma y resta**
  - **Enteros con signo**

MULTIPLICADOR
SUMA Y DESPLAZAMIENTOS (I)

- **Binario sin signo**
- **Reproduce el método manual A x B**
- **Inspección sucesiva de los bits de B**
  - **Si Bi = 1, se suma al resultado parcial A desplazada i-1 posiciones a la izquierda**
  - **Si Bi = 0 No se hace nada**
- **Modificación:**
  - **Se desplaza a la derecha el resultado parcial.**
    - **En lugar de desplazar a la izquierda A**
  - **Causa**
    - **Para evitar circuitería y control**

MULTIPLICADOR
SUMA Y DESPLAZAMIENTOS (II)

- **Algoritmo:**

1.- R3:= 0 y contador := 0

2a.- Si R2[0] = 1 R3 := R3+R1 (CK negativo)

2b.- Si R2[0]= 0 R3 := R3 + 0

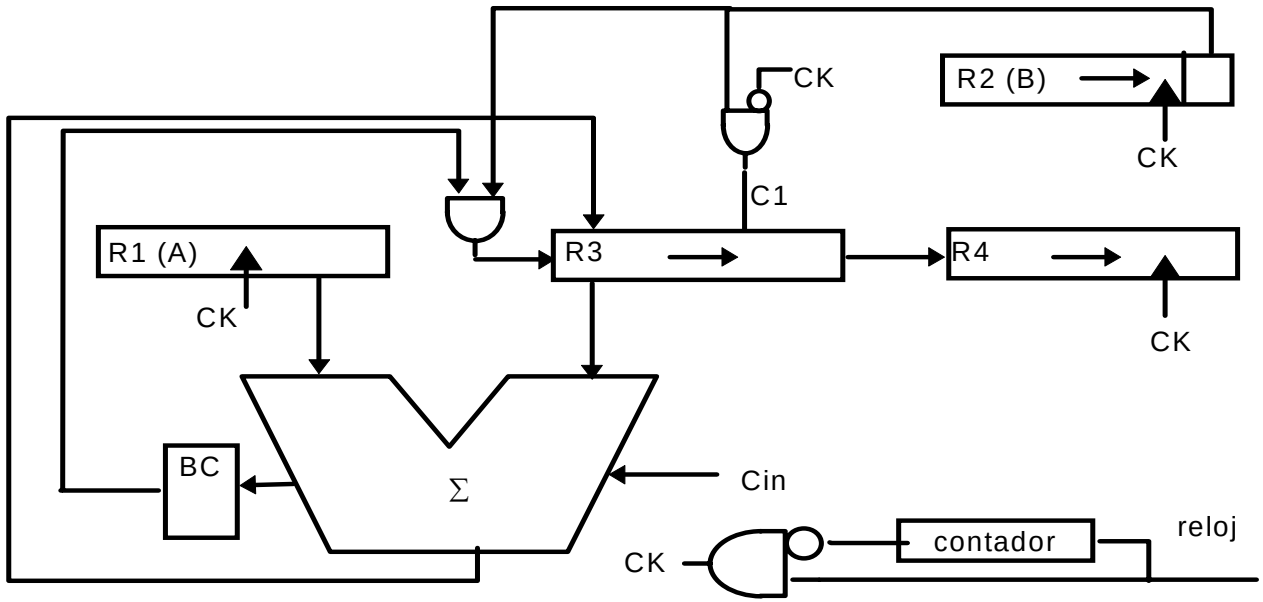
3.- Incrementar el contador (CK positivo)

4.- Desplazar derecha encadenando BC - R3 - R4 (ck positivo)

5.- Rotar a la derecha R2(ck positivo)

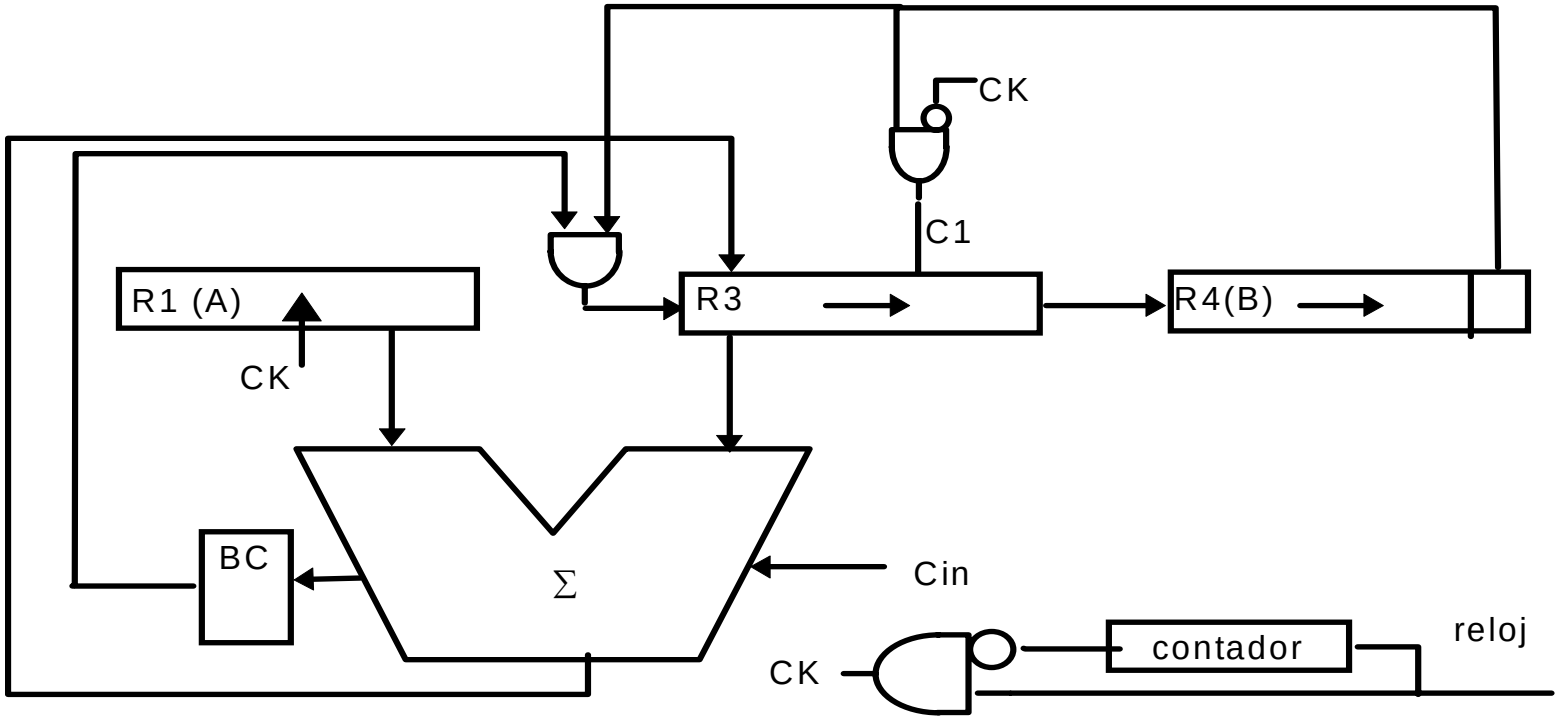
6.- ¿Fin contador ?

» Si no ha acabado vuelta a 2



MULTIPLICADOR
SUMA Y DESPLAZAMIENTOS (III)

eliminación del registro R2 para simplificar el hw



MULTIPLICADOR
SUMA Y RESTA (I)

- **Nº = 01110 = 10000 - 00010**
- **Ejemplo**
  - **si B = 01110 y quiero hacer A x B**

    A·(01110) = A·(10000 - 00010) = A·10000 - A · 00010
- **Generalizando**
  - **Se modifica el algoritmo para tratar las cadenas de 1's que contenga el multiplicador B**
  - **Avanzando de menos significativo a más significativo:**
    - **Se resta $A \cdot 2^i$ cuando al analizar B<i>se encuentra el 1er bit de una cadena de 1's**
    - **Se suma $A \cdot 2^i$ cuando al analizar B<i> se encuentra el 1er 0 después de una cadena de 1's**
  - **Es importante conservar el signo de las sumas parciales**
  - **Como los resultados parciales pueden ser positivos o negativos, el multiplicando A sólo debe ocupar N-1 bits**
  - **Se suponen los resultados negativos en C'2**

MULTIPLICADOR
SUMA Y RESTA(II)

- **R3 almacena resultados parciales**

1. R3 := 0 y el contador:=0

2. Observar B[0] (contenido en R2)

Si es el primer 1 de una cadena de 1s

- **R3: = R3-R1**

Si es el 1er 0 trás una cadenade 1s

- **R3:=R3+R1**

nada en los demas casos

3.- Incrementar el contador

4.- Desplazar a la derecha

» **BS -->R3--> R4**

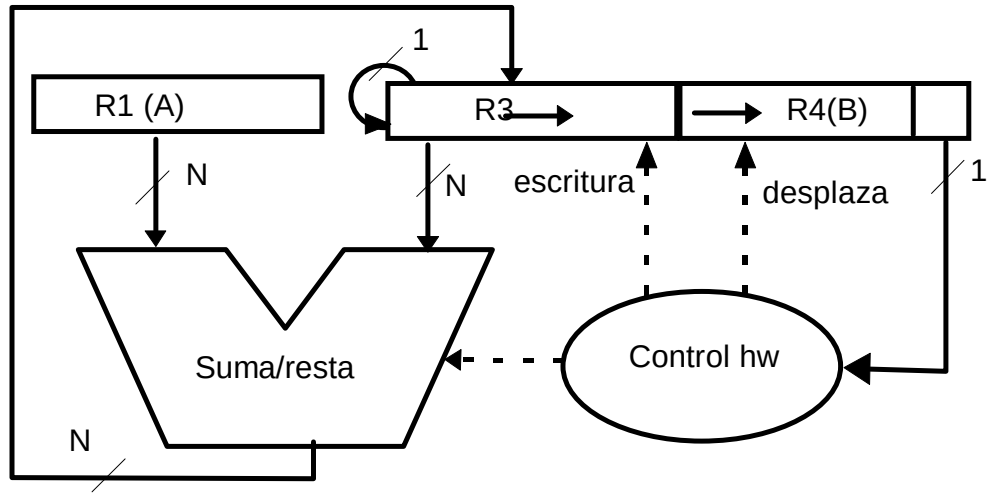
5.- ¿ contador? no ir a 2

6.- Si $B_{N-1} = 1$

se cierra la última cadena de 1's

» **R3 := R3 + R1**

» **equivalente a sumarle $A \cdot 2^N$**



MULTIPLICADOR
SUMA Y RESTA (III)

- **Atención**
  - **La concatenación se hace con el BS**
    - **Para conservar el signo de los productos parciales**
  - **El punto 6 se añade para tratar correctamente el caso que el bit más significativo sea un 1**
    - **En este caso se debe cerrar la cadena**
    - **$A \cdot 1110 = A \cdot (10000 - 0010) = A \cdot 2^4 - A2^1$**
    - **Como 4=N, 2^4 es equivalente a sumarle $A.2^N$**

MULTIPLICACIÓN CON SIGNO
ALGORITMO DE BOOTH (I)

- **Trabaja directamente con los operandos con signo**
- **Representación en C'2**
- **Se utiliza el Algoritmo de sumas y restas**
- **Un A negativo**
  - **No es problema**
  - **Las sumas se hacen directamente obteniéndose el resultado negativo en C'2**
  - **Las extensiones de signo en los productos parciales son importantes**
- **B es negativo**
  - **Problema**
  - **Su representación es 2^N - |B|**
  - **El resultado de aplicar directamente el algoritmo**
    - **A·B --> R* = A (2^N - |B|)**
  - **El resultado correcto sería:**
    - **R = 2^{2N} - A ·|B|**

MULTIPLICACIÓN CON SIGNO
ALGORITMO DE BOOTH (II)

- **para R* sea igual a R:**

$$R^* = A\,(2^N - |B|) = 2^N A - A \cdot |B| + 2^{2N} - 2^{2N}$$
$$= 2^N A - 2^{2N} + 2^{2N} - A \cdot |B| = 2^N A - 2^{2N} + R$$

  - **2^{2N}**
    - **Es el bit de acarreo**
    - **Se puede prescindir**

  - **$2^N \cdot A$**
    - **N desplazamientos**
    - **No realizar el último desplazamiento**
    - **2^N A es el contenido de R1 justo después de N rotaciones a la derecha**
      - **Luego es suficiente con no hacer la operación R3:=R3+R1**

MULTIPLICACIÓN CON SIGNO
ALGORITMO DE BOOTH (III)

- **Algoritmo**

1.- Inicializar: R3 0; contador 0.

2.- Observar B(0)

Si es principio de cadena de 1's R3 := R3 - R1 (equivalente a restar $A \cdot 2^i$)

Si es final de cadenas de 1's R3:= R3 + R1(equivalente a sumar $A \cdot 2^i$)

En caso contrario no se hace nada

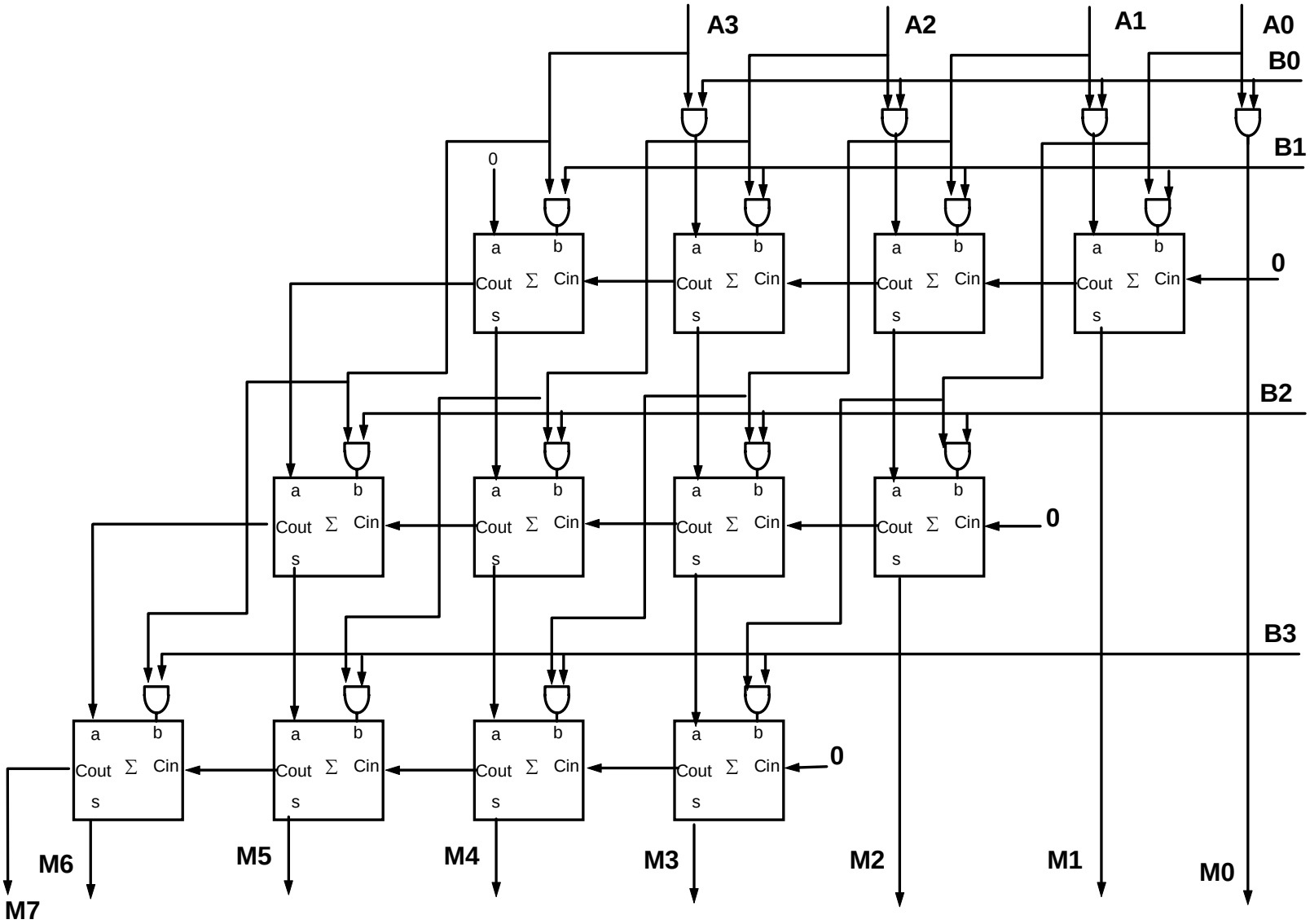
3.- Incrementar contador

4.- Desplazar a la derecha BS - R3 - R4 (equivalente a desplazar A a la izq)

5.- Estudiar el contador para acabar

- **El 6º punto no se añade para no tener en cuenta el desplazamiento 2^N A**
- **(Recuerda es un multiplicador de n bits: como los bits se numeran de 0 - (n-1) el bit n-esimo no pertenece al rango)**

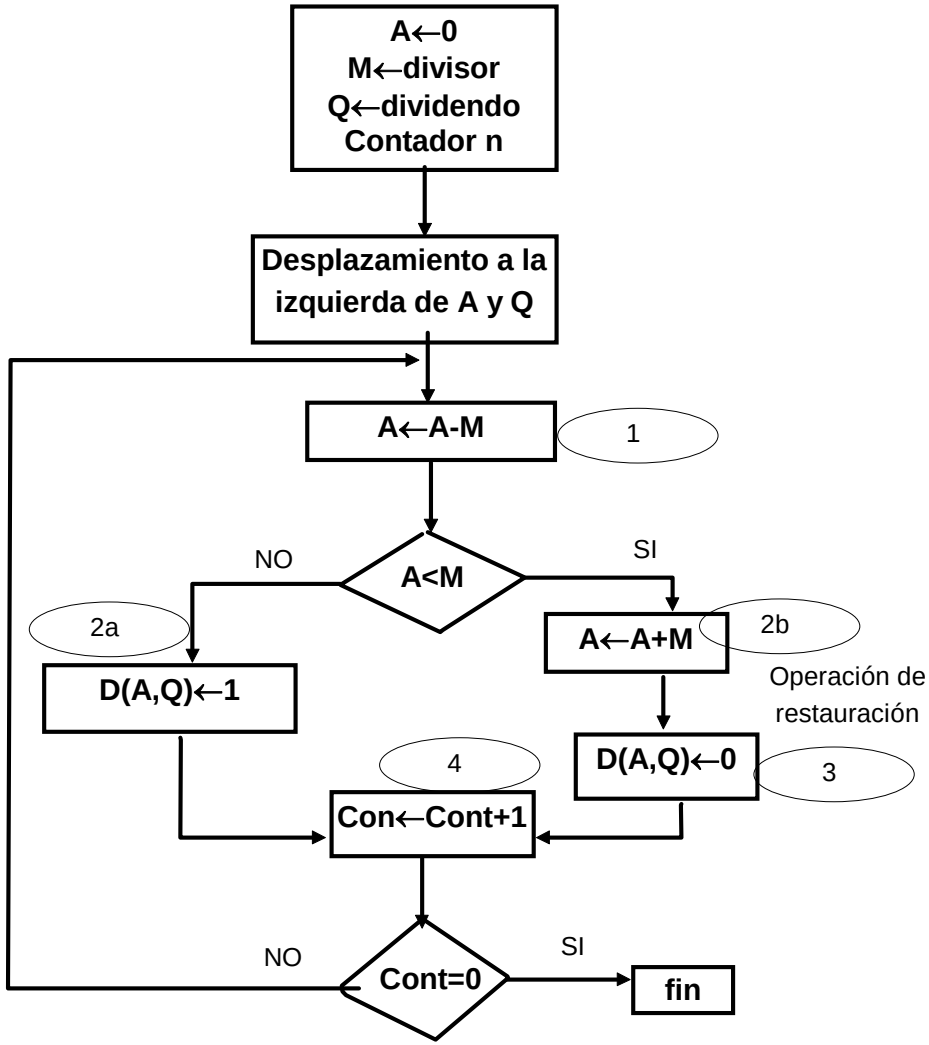
MULTIPLICACIÓN COMBINACIONAL

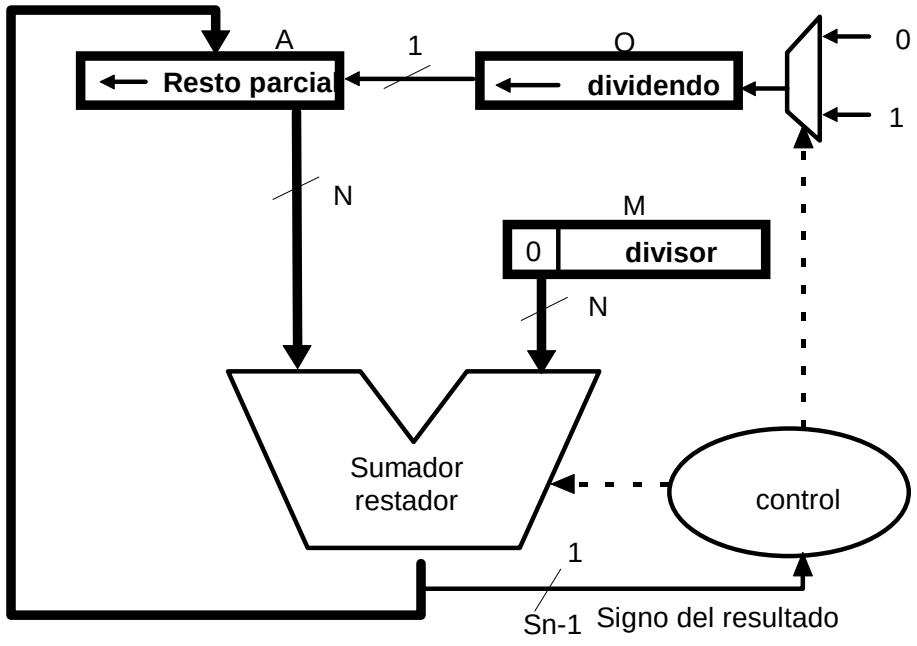


DIVISION
ENTEROS SIN SIGNO

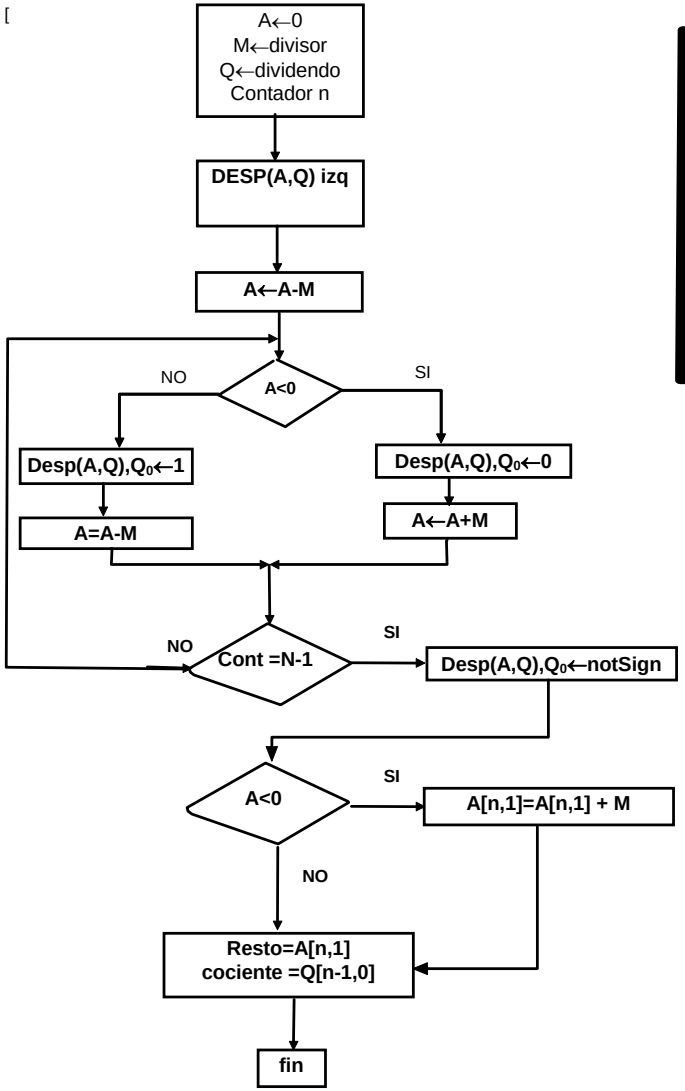
- **Más compleja que la multiplicación**
- **Se suele implementar siempre de manera secuencial es decir un sumador/restador más una algoritmo.**
- **Casi nunca combinacional**
- **Dos algoritmos:**
  - **División con restauración**
  - **División sin restauración**

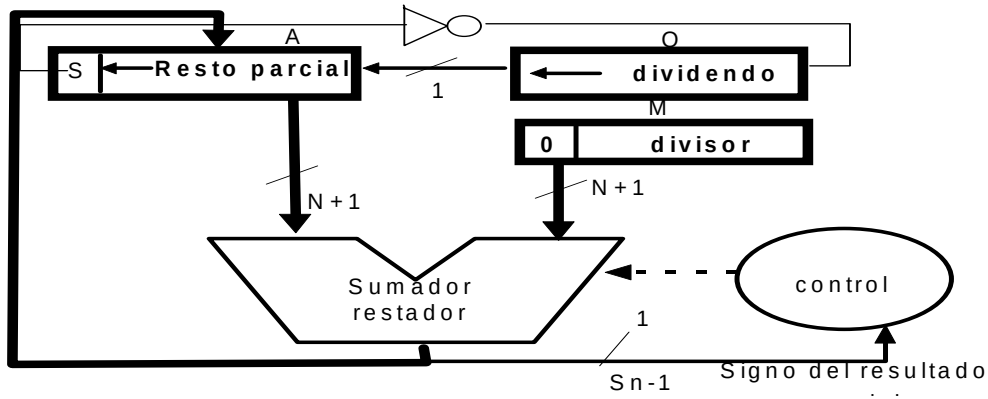
DIVISION
CON RESTAURACIÓN





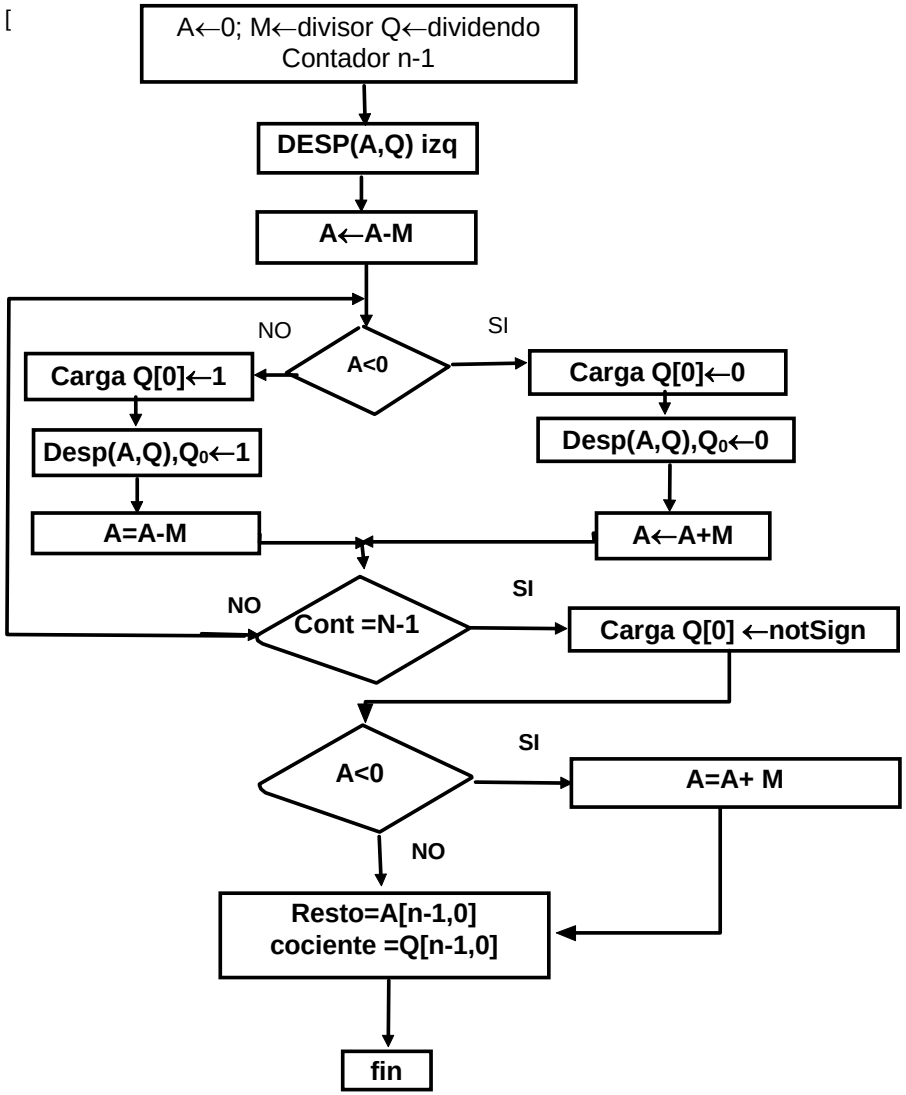
DIVISIÓN
SIN RESTAURACIÓN





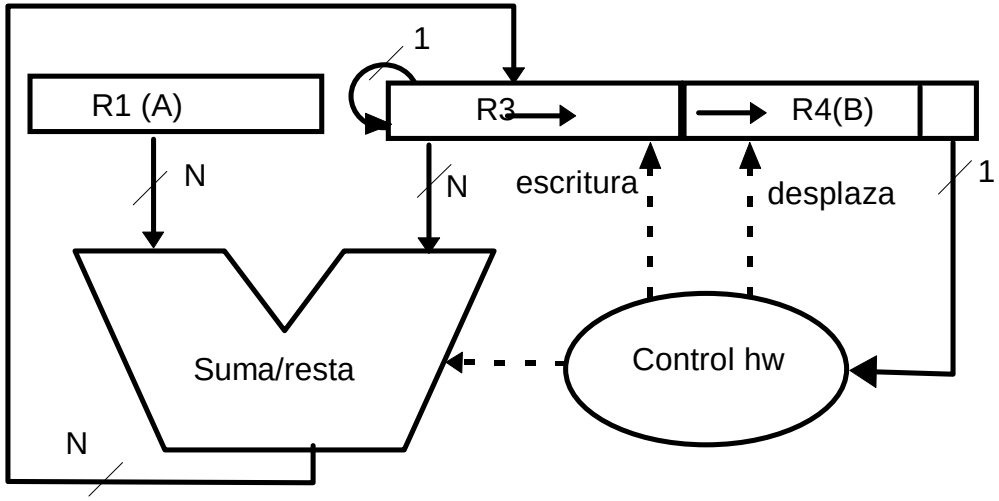
A[n]	A	Q	
	0000	0111	inicio
0	0000	1111	desplaza
-	1101		suma en c'2
1	1101	1111	
1	1011	1110	desplaza
+	0011		suma
1	1110	1110	
1	1101	1100	desplaza
+	0011		suma
0	0000	1100	
0	0001	1001	desplaza
-	1101		suma en C'2
1	1110	1001	
1	**110**1	***0010***	desplaza
0	**011**		
0	***001***		

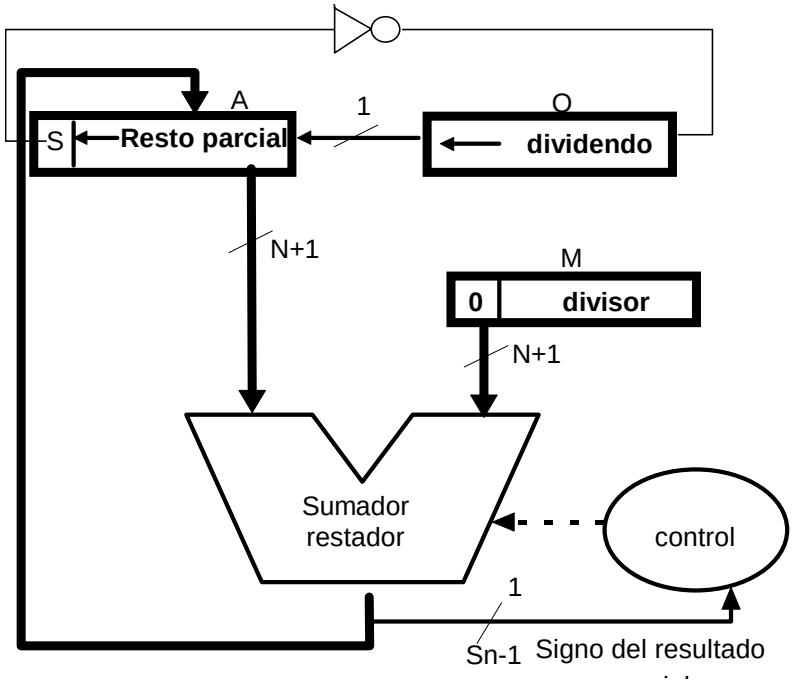
DIVISIÓN
SIN RESTAURACIÓN



A[n]	A	Q	
	0000	0111	inicio
0	0000	111x	desplaza
-	1101		suma en c'2
1	1101	111x	
1	1101	1110	carga q[0]
1	1011	110x	desplaza
+	0011		suma
1	1110	110x	
1	1110	1100	carga q[0]
1	1101	100x	desplaza
+	0011		suma
0	0000	110x	
0	0000	1001	carga Q[0]
0	0001	001x	desplaza
-	1101		suma en C'2
1	1110	001x	
1	1110	***0010***	carga
	0011		suma
0	***0001***		
0			

MULTIPLICACIÓN / DIVISIÓN





REPRESENTACION DE REALES
COMA FIJA

- **Posible representación de reales mediante un punto fijo :**
  - **$b_3 2^3 + b_2 2^2 + b_1 2^1 + b_0 2^0 + b_{-1} 2^{-1} + b_{-2} 2^{-2} + b_{-3} 2^{-3}$,**
  - **Ejemplo**
    - **$111.101 = 2^2 + 2^1 + 2^0 + 2^{-1} + 2^{-3} = 7 + 0.5 + 0.125 = 7.625$**
- **Inconvenientes:**
  - **No se pueden representar numeros grandes**
  - **No se pueden representar fracciones pequeñas**
- **Solución**
  - **Notación científica o coma flotante**

COMA FLOTANTE

- **Notación cientifica**
  - **976.000.000.000.000 = 9.76 10^{14}**
  - **0,000.000.000.000 976 = 9,76 10^{-14}**
- **Ventaja:**
  - **Permite con muy pocas cifras**
    - » **Un rango de números muy grandes**
    - » **Un rango de números muy pequeños**
- **Esta aproximación se puede aplicar a los binarios:**
  - **N^o = M ·B^E**
  - **M Mantisa**
  - **E exponente**
  - **B Base**
- **$001 \cdot 2^3 = 010 \cdot 2^2 = 100 \cdot 2^1 = 1000 \cdot 2^0 = 8$**
- **$0.0110 \cdot 2^6 = 0.1100 \cdot 2^5 = 1.1000 \cdot 2^4 = 11.0000 \cdot 2^3 = 24$**

COMA FLOTANTE
REPRESENTACIÓN TÍPICA (I)

- **sig exponente mantisa**
- **sig es el signo de la mantisa**
- **Base**
  - » **se utiliza implicitamente**
  - » **suele ser 2 o 16**
- **Exponente:**
  - **Tiene su propio signo**
  - **q el nº de bits del exponente**
  - **Se representa en exceso 2^{q-1}**
    - » **Con q= 8 la representación en exceso a 128**
  - **Para hallar el valor verdadero del exponente hay que restarle 128 al valor almacenado en este caso el rango del exponente será -128 a 127**

0 0 0 0	-8
0 0 0 1	-7
0 0 1 0	-6
0 0 1 1	-5
0 1 0 0	-4
0 1 0 1	-3
0 1 1 0	-2
0 1 1 1	-1
1 0 0 0	0
1 0 0 1	1
1 0 1 0	2
1 0 1 1	3
1 1 0 0	4
1 1 0 1	5
1 1 1 0	6
1 1 1 1	7

COMA FLOTANTE
REPRESENTACIÓN TÍPICA(II)

- **Mantiasa**
  - **Se suele representar en modo fraccionario**
    - **$B_{-1}\ 2^{-1}+b_{-2}\ 2^{-2}+b_{-3}\ 2^{-3}$**
  - **Se suele normalizar:**
    - **Permite el intercambio de información con mucha facilidad**
    - **Simplifica el cálculo aritmético**
    - **Aumenta la exactitud al eliminar ceros que no contiene información por bits que si la contiene**
  - **0.1Bbb ... B, donde b puede ser 0 o 1**
  - **Problema:**
    - **Los 2 bits de la izquierda siempre son los mismos**
      - **No aportan información**
      - **Consumen bits**
  - **Solución**
    - **Suponerlos implicitamente incorporarlos a la mantisa dos bits más de precisión**
    - **Explicitos 0.1Bbbbbb**
    - **implicitos bbbbbbbb ejemplo. : M = 0--- 0 --> 0.10 --- 0**

COMA FLOTANTE
REPRESENTACIÓN TÍPICA(III)

- **El rango decimal que puede abarcar la mantisa es el siguiente:**
  - **M = 00----0**
    - **Añadiendo los bits implícitos 0,100----0**
    - **$M_{10} = 1 / 2^1 = 0,5$**
  - **M=11----1**
  - **Añadiendo los bits implícitos 0,11---1**

$$\frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \cdots + \frac{1}{2^{N+1}} = 1 - \frac{1}{2^{N+1}}$$

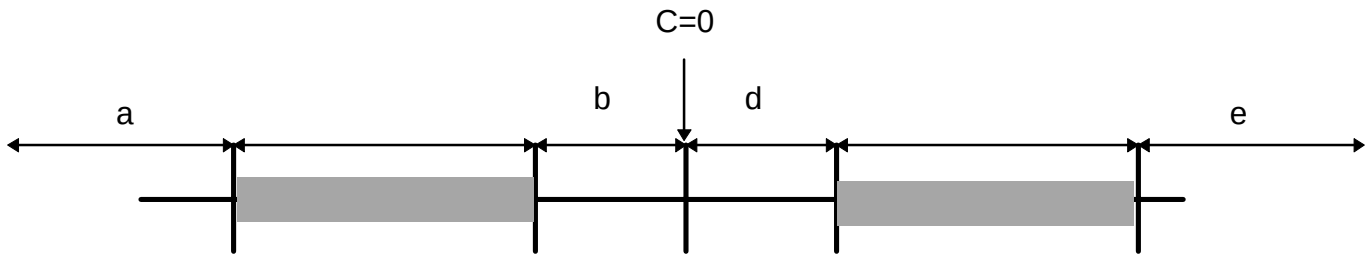
- **El rango de la mantisa, suponiéndola N bits, incluido los implícitos, es:**

$$\frac{1}{2^1} \leq |M| \leq 1 - \frac{1}{2^{N+1}} < 1$$

COMA FLOTANTE
REPRESENTACIÓN TÍPICA (IV)

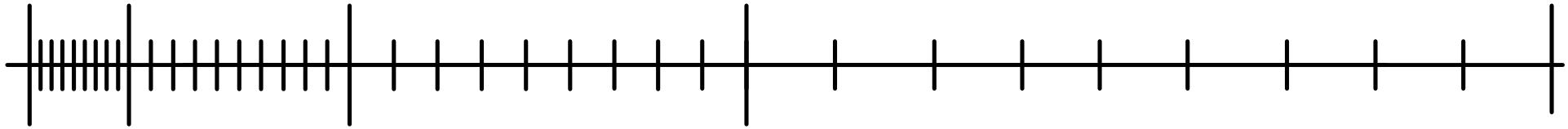
- **n= 32 bits**
  - **1 bit para el signo**
  - **8 bits para exponente en exceso 128**
  - **23 explícitos para la mantisa**
- **El rango de representación es**
  - **Negatvos [-(1- 2^{-24}) . 2^{127}, - 0,5· 2^{-128}]**
  - **Positivos [0,5 . 2^{128} , (1 - 2^{-24}) . 2^{127}]**

- **Overflow negativo:**
  - **nº negativos < - (1 - 2^{-24}) . 2^{127}**
- **underflow negativo**
  - **nº > -0,5 . 2-128**
- **cero**
- **underflow positivo:**
  - **nº positivos < 0,5 . 2^{-128}**
- **overflow positivo:**
  - **nº positivos >(1 - 2^{-24}) . 2^{-128}**



COMA FLOTANTE
REPRESENTACIÓN TÍPICA (V)

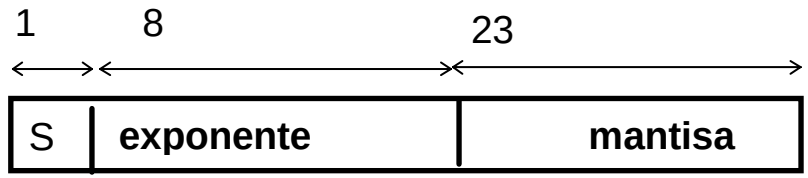
- El nº de valores representados es el mismo en cualquier formato
- Los nº no se distribuyen uniformente en el espacio numérico.
  - Están más próximos entre si en el origen



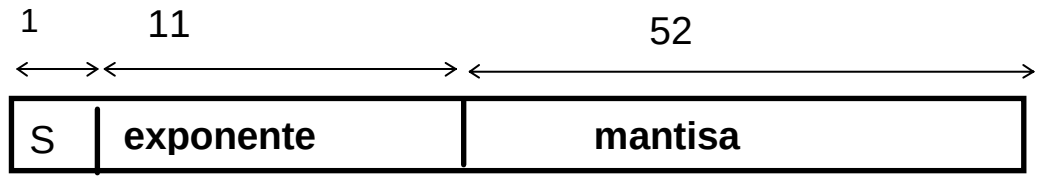
- Debe existir un equilibrio entre rango y precisión
- Si aumentamos el nº de bits del exponente
  - Aumentamos el rango
  - Disminuimos la precisión
  - La única manera de aumentar rango y precisión, es aumentar el nº de bits total de la representación
- Muchos computadores:
  - Precisión simple
  - precisión Doble

COMA FLOTANTE
ESTANDAR DEL IEEE 754(I)

- Facilitar la portabilidad de programas de un procesador a otro
- Desarrollo de programas orientados al cálculo numérico
- Define dos formatos sencillos
  - **De 32 bits**
    - » **1 de signo de mantisa**
    - » **23 de mantisa**
    - » **8 de exponente**
    - » **Base implícita = 2**



  - **De 64 bits**
    - » **1 de signo**
    - » **52 de mantisa**
    - » **11 de exponente**
    - » **Base implícita = 2**



COMA FLOTANTE
ESTANDAR DEL IEEE 754 (II)

- **Define 2 Formatos ampliados**
  - **Su forma exacta depende de la implementación**
  - **Bits adicionalesen el mantisa (mayor precisión) y en el exponente (mayor rango)**
    - **simple ampliado**
      - **palabra 43 bits (o más)**
        - **11 exponente**
        - **31 mantisa**
    - **Doble ampliado**
      - **mas de 79 bits de palabra**
        - **mas de 15 bits de exponente**
        - **mas de 63 de mantisa**
  - **Se usa para cálculos intermedios**
    - **Evitan los redondeos excesivos**
    - **Evita abortar un cálculo intemedio por overflow cuando el resultado podría incluirse en el tamaño final**
    - **El formato simple ampliado se suele usar mas, funcionalidad parecida al formato doble ampliado sin la penalización qu eproduce una excesiva precisión**

COMA FLOTANTE
ESTANDAR DEL IEEE 754 (III)

- **No todos los patrones de bits se interpretan igual**
  - **Los exponentes = [00---0] y [11----1] definen casos especiales**
- **Caso general**
  - **Rango de los exponentes que se encuentran entre [00---01] y [11--10]**
  - **Están representando números normalizados, no nulos en coma flotante**
  - **Características de los números normalizados**
    - **Exponentes:**
      - **Los exponentes se representan en exceso a 127(simple) o 1023(doble)**
      - **Rango de exponentes -126 a 127 (simple)o de -1022 a 1023 (doble)**
    - **Mantisa**
      - **El primer bit de la izquierda del punto decimal es un 1 implícito 1.Bbbbbbb**
      - **Mantisa efectica de 24 o 53 bits**
      - **A los bits codificados se les suele llamar fracción , mantisa =1+fracción**
  - **Nº decimal= $(-1)^S$ (mantisa) 2^{Er}= $(-1)^S$ x(1+fracción) $2^{Ecodificado-desplazamiento}$**
    - **S el bit de signo,**

COMA FLOTANTE
ESTANDAR DEL IEEE 754 (IV)

- **Según que los exponentes y fracciones toman el valor 1 y 0 se dan los siguientes casos particulares**
  - **Exponente = 0 fracción = 0 --> -0 o +0**
  - **Exponente =0 fracción distinta de 0**
    - » **Números desnormalizados**
    - » **El bit de la izquierda del punto binario que antes era 1(implícitamente) ahora es cero**
    - » **El verdadero exponente -126 o -1022 según sea la representación simple o doble**
    - » **El nº es positivo o negativo según el bit de signo**
  - **Exponente =1 fracción =0 mas infinito, menos infinito**
  - **Exponente =1 fracción distinto de 0 numero utilizado para señalar excepciones**

ARITMÉTICA EN COMA FLOTANTE

- **La suma y resta**
  - **Deben asegurar que ambos números tienen el mismo exponente:**
    - **Se pueden necesitar operaciones de desplazamiento del punto flotante**
- **La multiplicación y la división**
  - **Más sencillas**
- **Problemas**
  - **Overflow del exponente:**
    - **Un exponente positivo excede del mx posbile**
    - **A veces se representa como mas, menos infinito**
  - **Underflow del exponente:**
    - **Un exponente negativo excede del mx valor negativo permitido**
    - **Nº demasiado pequeño--> puede aproximarse al 0**
  - **Underflow de la mantisa:**
    - **En el proceso de alinear mantisas los dígitos pueden salirse por la derecha de la mantisa: redondeo.**
  - **Overflow de la mantisa:**
    - **La suma de dos mantisas del mismo signo puede producir un carry out del bit más significativo**

ARITMETICA EN COMA FLOTANTE

SUMA Y RESTA

- **Pasos:**
  1. **Comprobar los ceros**
  2. **Alinear los exponentes**
  3. **Sumar/restar las mantisas**
  4. **Normalizar el resultado**
- **Si el formato incluye un bit implícito debe hacerse explícito para la operación**
- **Los exponentes y mantisas se suelen guardar por separado y sólo se reunen en el resultado final**

ARITMETICA EN COMA FLOTANTE

SUMA Y RESTA

- **Alinear los exponentes**
  - desplazamiento del punto decimal
  - un cambio en el valor del exponente
- **Se puede desplazar**
  - el más pequeño
  - el más grande
- **Para alinear el pequeño**
  - Incrementar su exponente en una unidad
    - » equivale a multiplicarlo
  - Desplazando la mantisa a la derecha
- **Para alinear el número grande**
  - Se resta el exponente en una unidad
  - Desplaza la mantisa a la izquierda

- **Ejemplo $123 \cdot 10^0 + 456 \cdot 10^{-2}$**
  - vamos a alinear el nº más pequeño $45{,}6 \times 10^{-1} = 4{,}56 \times 10^0$
- **problema:**
  - la alineación puede dar lugar a perdidas de dígitos
  - debe ser el nº más pequeño el que se desplace
    - » bits que se pierdan serán los de menor importancia
- **Si la base es 16**
  - los desplazamientos de la mantisa deben ser de 4 bits
- **Si el resultado es 0 en la mantisa, el resultado de la operación es el otro nº**
- **Si dos exponentes difieren mucho**
  - el nº menor se pierde

ARITMETICA EN COMA FLOTANTE
SUMA Y RESTA

- **Sumar/restar las mantisas**
  - **Se suman las mantisas teniendo en cuenta los signos**
  - **Problema 1**
    - **El resultado puede ser 0**
    - **Solución1**
      - **Modificar el exponente para que rerpesente el 0**
  - **Problema 2**
    - **Overflow de la mantisa:**
    - **Solución2:**
      - **La mantisa de desplaza a la derecha para no perder el bit significativo**
      - **El exponente se incrementa**
  - **Problema 3**
    - **underflow de la mantisa**
    - **si existe bit de guarda-->redoneo**
  - **Problema 4**
    - **Overflow , underflow del exponente**
    - **¡¡Alto no tiene solución**

ARITMETICA EN COMA FLOTANTE
SUMA Y RESTA

- **Normalizar el resultado:**
  - **Desplazar los dígitos de la mantisa a la izquierda hasta que el dígito más significativo sea 1**
  - **Para cada desplazamiento de la mantisa decremento en una unidad el exponente**
    - **problema**
      - **Se puede producir un underflow del exponente.**
      - **solución:**
        - **no tiene ¡¡alto!!**
- **La última acción es realizar el redondeo**

ARITMETICA EN COMA FLOTANTE
MULTIPLICACIÓN

- 1.- Comprobar si los operandos son 0 En caso afirmativo el resultado = 0
- 2.- Sumar exponentes:
  - Si están almacenados en exceso, el valor del exceso se debe restar del resultado
  - Problema: overflow o underflow del exponente
    - Solución: no tiene ¡¡alto!!
- 3.- Multiplicar las mantisas teniendo en cuenta los signos:
  - Igual que con los enteros
  - Se trabaja con magnitud y signo
  - Resultado Doble longitud que multiplicando y multiplicador
  - Problema : Overflow de la mantisa:
    - Solución:
      - La mantisa de desplaza a la derecha para no perder el bit significativo
      - El exponente se incrementa
  - Los bits extra de menor peso se pierden en el redondeo
- 4.- Normalización del redondeo:
  - Podrían dar lugar a un underflow del exponente

ARITMETICA EN COMA FLOTANTE
DIVISION

1.- Comprobar los ceros

1A.- Si dividendo es cero el resultado 0

1B.- Si el divisor es cero

¡¡ERROR!!

Resultado infinito

2.- Se resta el exponente del divisor al del dividendo

- **Se elimina el exceso del resultado;--> se debe volver a sumar**
- **Problema**
  - » **Puede dar overflow o underflow**
  - » **Solución no tiene :¡¡¡alto!!!**

3.- División

4.- Normalización

5.- Redondeo

BITS DE GUARDA

- **Cuando se realizan operaciones en coma flotante,**
  - **Los operadores deben traerse a los registros de la UAL**
  - **Los exponentes y las mantisas se ponen en registros separados**
  - **Los registros que contienen las mantisas tiene mayor longitud que la de la mantisa**
  - **Bits de guarda son los bits sobrantes de estos registros**
    - **Se utilizan para no perder precisión durante las operaciones**

BITS DE GUARDA

- **P.E. Suponiendo mantisa de 24 bits y el 1 implicito a la izquierda del punto:**

  X = 1. 000...00 2^1

  Y = 1. 111...11 2^0

- **x-y Sin bit de guarda**

  - **Se desplaza el número menor a la derecha**

    Y' = 0. 111...11 2^1 se pierde un 1 por la derecha

  - **1. 000...00 2^1 - 0. 111...11 2^1= 0. 000...01 2^1**
  - **Al normalizar 1.000..00 2^{-22}**
  - **En el desplazamiento de "y", se ha perdido un bit de importancia**

- **x-y Con bits de guarda:**

  - **X = 1. 00... 000 <u>0000</u> 2^1**
  - **Y = 0. 11... 111 <u>1000</u> 2^1**
  - **Y el resultado de la resta es:**

    0.000...00 <u>1000</u>= 1.000…00 <u>0000</u> $\cdot 2^{-23}$ = 1.000…00$\cdot 2^{-23}$ = 0.100..00$\cdot 2^{-22}$

TRUNCAMIENTO Y REDONDEO

- **Los resultado se guardan en registros de mayor tamaño, que el que los contiene finalmente**
- **Objetivo:**
  - **Conservar en la mayor medida de lo posible la precisión de la operación**
- **Efecto:**
  - **Habrá que eliminar parte de los bits que contienen el resultado obtenido**
- **Trucamiento:**
  - **Eliminar los bits de la derecha que no caben en la representación**
  - **Problema:**
    - **El error inducido en los resultados, es siempre por defecto**
    - **El error acumulado crece rápidamente**
- **Redondeo:**
  - **El estándar IEEE propone 4 aproximaciones:**

    1.- Redondeo al más cercano

    2.- Redondeo a +infinito

    3.- Redondeo a -infinito

    4.- Redondeo a 0

TRUNCAMIENTO Y REDONDEO

- **Redondeo al más cercano:**
  - **Es el utilizado por defecto que da el estandar**
  - **Se representa el valor representable más próximo al valor del resultado obtenido**
  - **Existen tres casos:**
    - **<u>Bits de guarda = 1bbbb</u> siendo como poco uno de los b's=1**
      - **La fracción > 0.5**
      - **Se redondea por exceso sumando 1 al ultimo representable**
    - **<u>Bits de guarda = 0bbbb</u>**
      - **La fracción < 0.5**
      - **Redondeo por defecto --> truncar**
    - **Bits de guarda = 10000**
      - **La fracción = 0.5**
      - **Forzar el nº par en el representable:**
        - **Si el representable acaba en 1 se le redondea por exceso (suma 1)**
        - **Si el representable es 0 se le deja como está**

ORGANIZACIÓN Y DISEÑO DE LA RUTA DE DATOS

TEMA 6

INTRODUCCIÓN

- **La implementación del procesador determina:**
  - **La duración del ciclo de reloj**
  - **El número de ciclos de reloj por instrucción**

- **Principios de diseño**
  - **Hacer rápido el caso común**
  - **La simplicidad favorece la regularidad**

REGISTROS DEL PROCESADOR

- Nivel de la jerarquía de memoria más elevado
- Características son
  - Más pequeños
  - Más rápidos
  - Más caros

- Existen dos tipos de registros en la CPU
  - Visibles por el usuario
    - » Los utiliza el usuario para minimizar los accesos a Mp
  - De estado y de control
    - » U.C. Para controlar la operación de la CPU
    - » S.O. Para controlar la ejecución de programas

REGISTROS VISIBLES POR EL USUARIO

- **Referenciables por el Lenguaje Máquina**
- **Propósito general**
  - **Datos(enteros, coma flotante …)**
  - **Dirección**
- **Códigos de condición**
  - **Se activan tras una operación**
  - **Uso en las bifurcaciones condicionales**
  - **Pueden formar parte del Registro de Estado:**
    - » **Pueden leerse**
    - » **No pueden alterarse directamente**
- **Propósito Específico**
  - **Direccionamiento implícito**
  - **Ventaja ahorro de bits**
  - **Desventaja falta de flexibilidad**
  - **La Tendencia actual es hacia registros específicos**
  - **ejemplos : puntero a pila, puntero a segmento, registro de indices**

REGISTROS VISIBLES POR EL USUARIO
DECISIONES DE DISEÑO

- **Específico o general**
  - **específicos--> direccionamiento implícito**
    - » **ventaja ahorra bits**
    - » **desventaja-- falta de flexibilidad**
- **Número de Registros**
  - **A más registros, más bits de dirección**
  - **Normalmente entre 8 y 32 registros**
    - » **Menos registros más referencias a MPrincipal**
    - » **Más registros**
      - **No reducen las referencias a memoria**
      - **Aumentan los retardos**
- **Longitud de Registro**
  - **Registro de Dirección**
    - » **Como poco la longitud de la dirección**
  - **Registro de Datos**
    - » **Deben ser capaces de almacenar la mayoría de tipos de datos**
  - **Algunas máquinas permiten registros de doble longitud**

REGISTROS DE ESTADO Y CONTROL

- **Controlan las operaciones de la CPU**
- **Pueden ser visibles al ejecutar instrucciones en modo control o S.O.**
- **Dependen de la máquina**
- **Contador de programa**
  - **Contiene la dirección de la siguiente Instrucción**
  - **Lo actualiza la CPU después de cada búsqueda de instrucción**
  - **Siempre existe un puntero a la siguiente instrucción**
  - **Se modifica en los saltos y bifurcaciones**
- **Registro de Instrucciones:**

  Contiene la última instrucción traída de la M.Principal
- **Registro de Direcciones**
  - **MAR**
  - **Registro de Direcciones de Memoria**
- **MBR:**
  - **(Memory Buffer Register)**
  - **Contiene una palabra leída de la memoria o que se va a escribir en la memoria**

REGISTROS DE ESTADO Y CONTROL

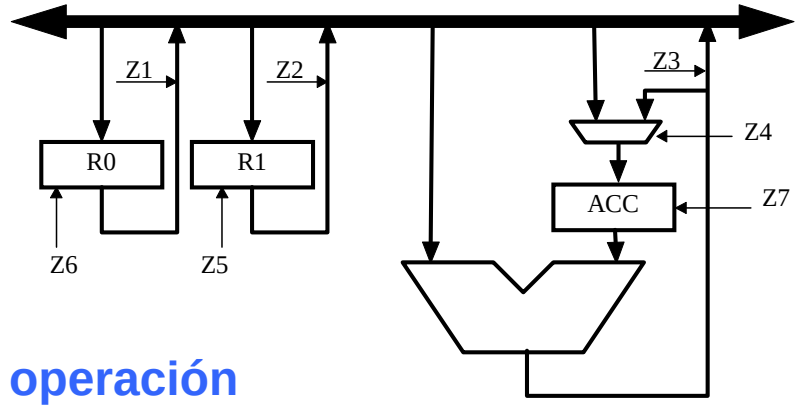
- **Program Status Word (PSW)**
  - **Contiene información de estado**
  - **Sus campos más comunes:**
    - » **Signo : signo de la última operación aritmética**
    - » **Zero**
    - » **Carry**
    - » **Igual**
    - » **Overflow**
    - » **Supervisor**
- **Registro acumulador de la UAL**
  - **Registros frontera entre**
    - » **El camino de datos**
    - » **UAL**
  - **No suelen ser accesibles por el programador**
  - **Destaca el acumulador (ACC)**
  - **Tamaño = longitud de palabra del procesador**
  - **Contiene uno de los operandos o el resultado**

CAMINO DE DATOS SECUENCIAL

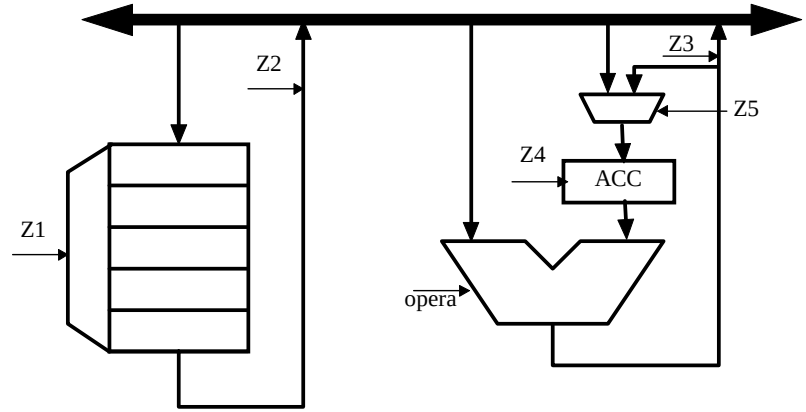
- En el camino de datos secuencial sólo existe una UAL
- Implementación de algoritmos complejo
  - Operaciones sencillas en serie
- El BUS es un Elemento primordial en el camino de datos
- BUS grupo de líneas digitales que soportan un mismo tipo de información del computador que se usan para unir módulos del mismo
  - Direcciones
  - Datos
  - Control
- Bus del microprocesador
  - Nivel más bajo de la jerarquía
  - Especificaciones no son de dominio publico
  - Influyen decisivamente
    - » Coste
    - » Rapidez de sistema
      - ◆ Buses de comunicación principales cuellos de botella

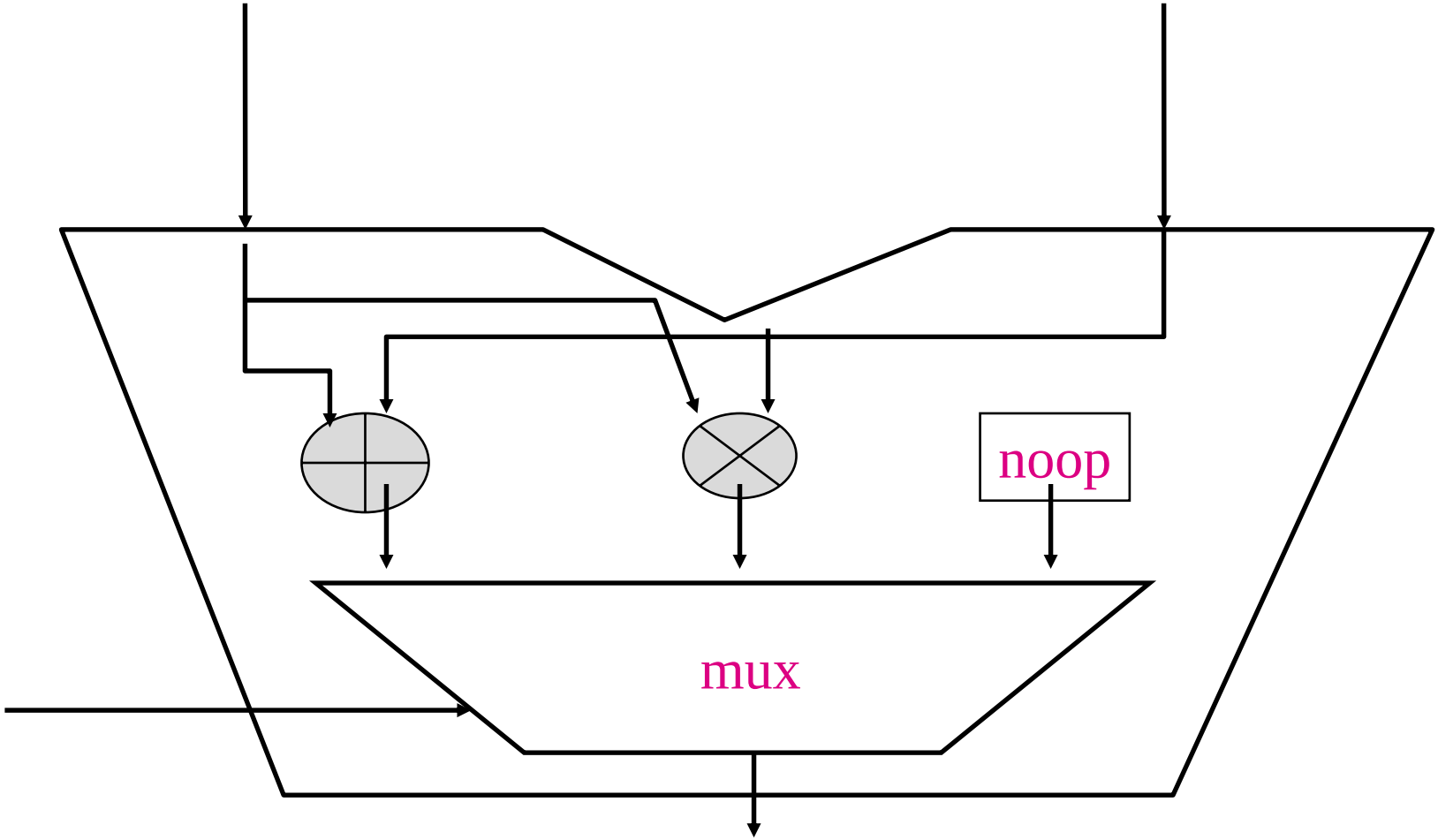
CAMINO DE DATOS SECUENCIAL
DE UN BUS

- **El más usado comercialmente**
  - **Debido a su bajo coste**
- **Bus Bidireccional**
  - **Para depositar datos en los Registros**
  - **Para cogerlos datos de los Registros**



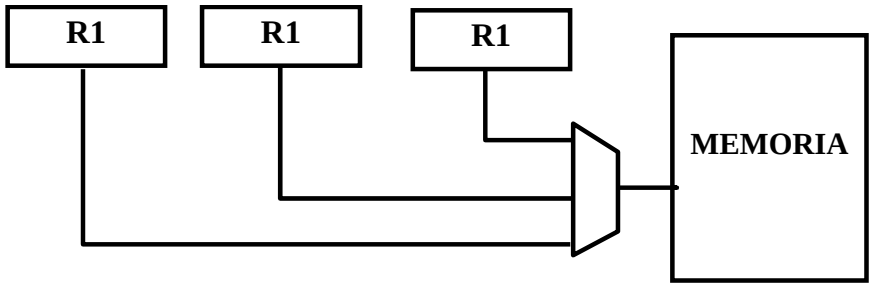
- **Se necesita un Acc que guarde datos de la operación**
- **Inconveniente**
  - **Lentitud:**
    - » **Más ciclos para la misma instrucción**
- **Variante**
  - **Agrupar los registros en un banco de registros**

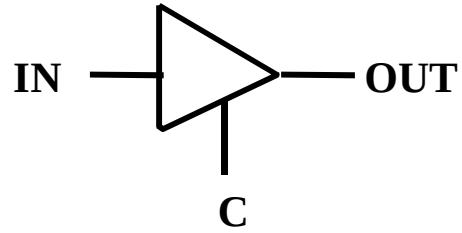


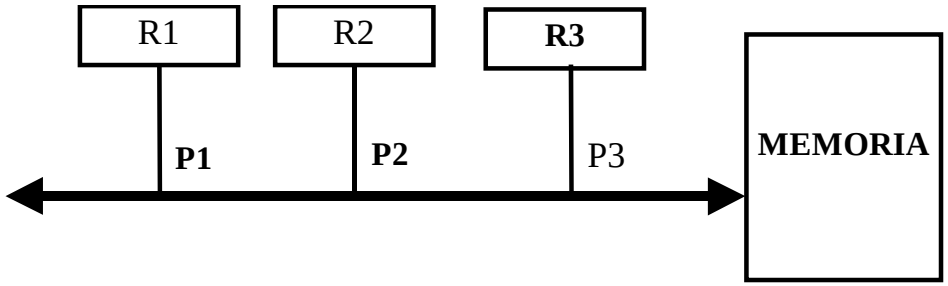

noop
mux

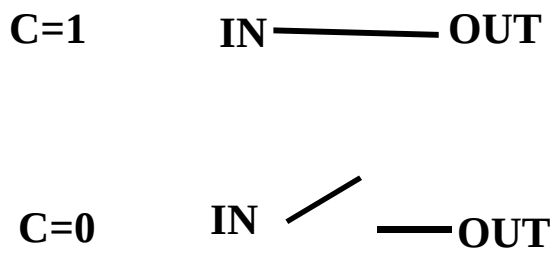
Puertas triestate

Se utilizan para controlar las salidas de diferentes módulos a un bu



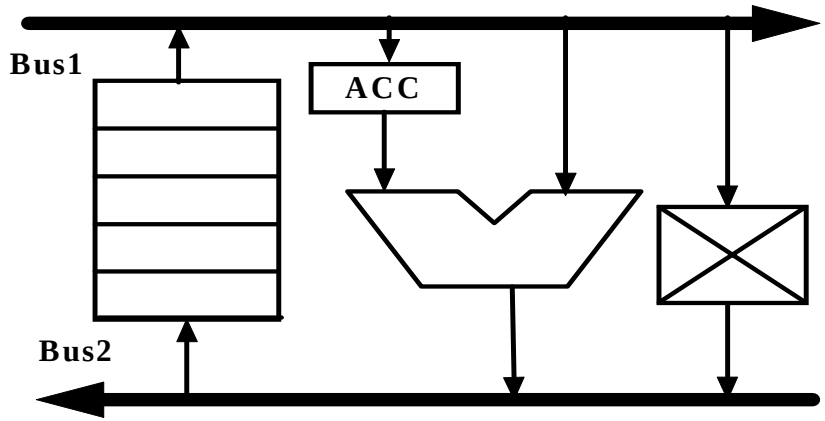






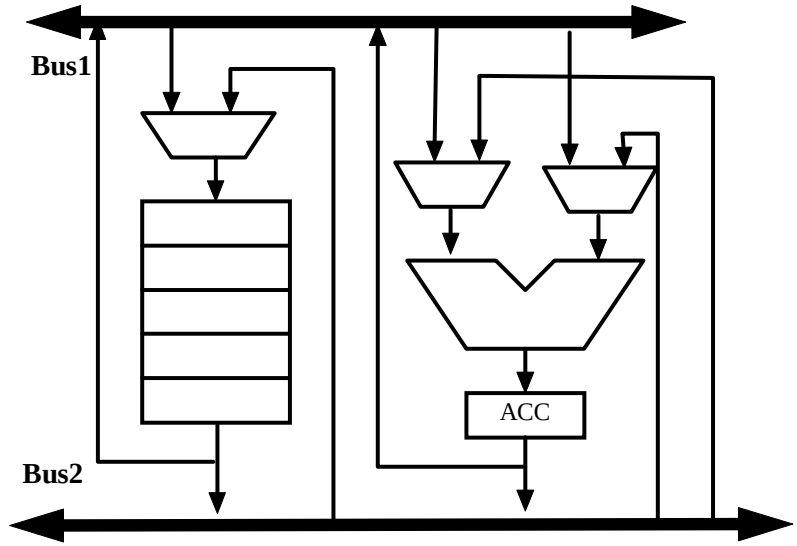
CAMINO DE DATOS SECUENCIAL
DE DOS BUSES

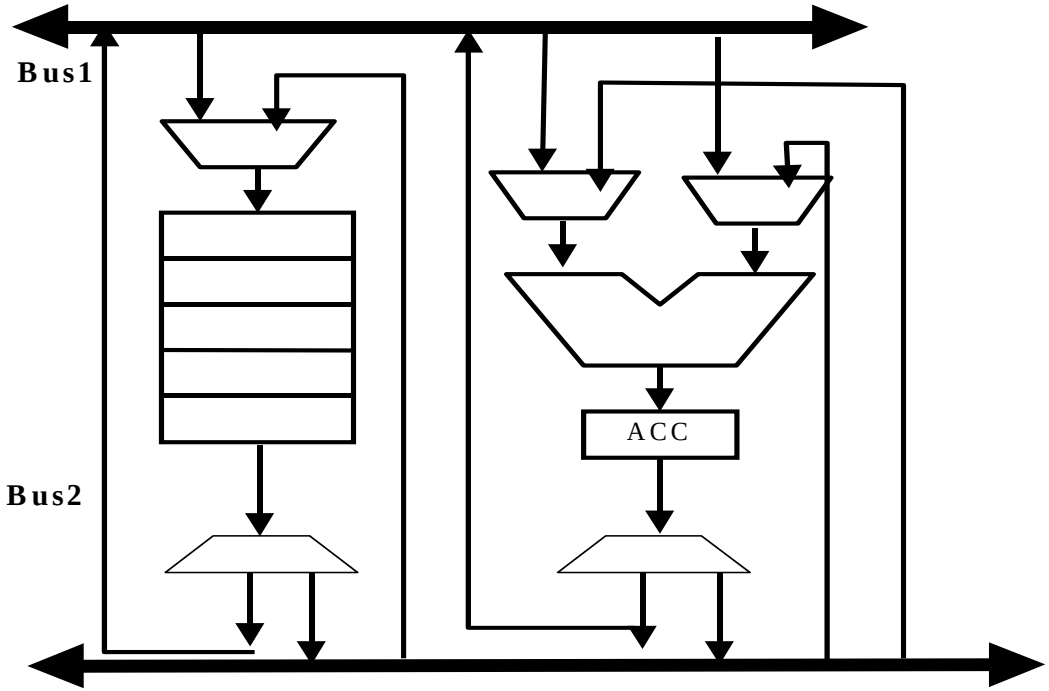
- Incremento de coste
- Reduce los ciclos: mejora la velocidad
- Puede ser de dos tipos :
  - unidireccionales
  - bidireccionales
- Buses unidireccionales:
  - Bus 1
    - » lleva los datos de la MP y del BR a la UAL
  - Bus 2
    - » lleva los resultados de la UAL
      - a la Memoria Principal
      - al Banco de Registros
  - Puede existir un enlace entre el Bus 1 y el Bus 2 (registro auxiliar,puerta triestate, UAL)
  - Para llevar un dato de memoria a registro
    - » a través de la UAL
    - » a través del enlace

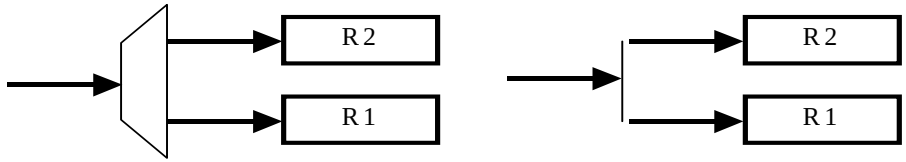


CAMINO DE DATOS SECUENCIAL
DE DOS BUSES

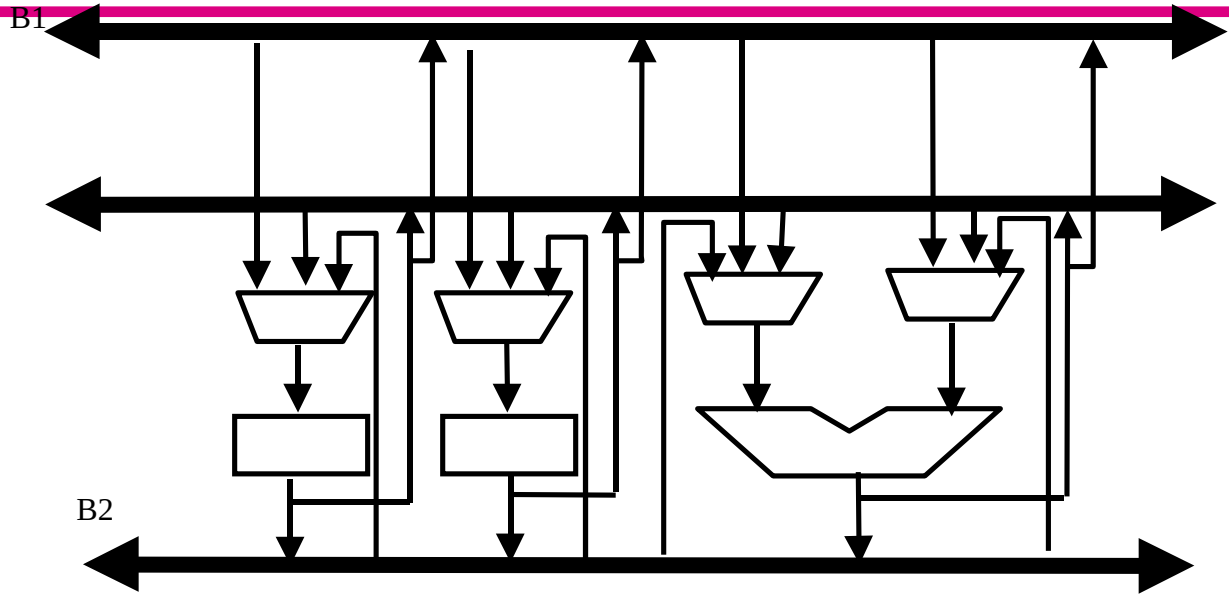
- **Buses bidireccionales:**
  - Dos buses bidireccionales
  - La memoria , la UAL y el banco de registros se pueden cargar o leer por cualquiera de ellos
  - Aumento del rendimiento
  - Mejora de la capacidad de procesamiento
  - Encarece el producto
  - Complejidad de control




Bus1
Bus2
ACC


R2
R1
R2
R1

CAMINO DE DATOS SECUENCIAL
DE TRES BUES



- Los tres buses son bidireccionales
- La menos usada comercialmente,
  - elevado coste
- Incremento del conexionado
- Computadoras especializadas de gran potencia de cómputo.

DISEÑO DE LA RUTA DE DATOS
DISPARO POR FLANCO DE RELOJ

- **Al diseñar de una máquina se deberá decidir**
  - **Como será lógica de la máquina**
  - **Como será el ck de la máquina**
- **Los datos se actualizan en los flancos de ck**
- **La entrada a un circuito combinacional**
  - **Viene de un conjunto de elementos de estado**
  - **Se han cargado en los elementos de estado en el ultimo ciclo de ck**
- **La salida de un circuito combinacional**
  - **Se escriben en un elemento de estado**
  - **Son los valores que se cargan en los elementos de estado en el siguiente flanco de ck**

DISEÑO DE LA RUTA DE DATOS
DISPARO POR FLANCO DE RELOJ

- Un elemento puede ser leído y escrito en el mismo ciclo sin producir datos erróneos
- El ciclo debe permitir que las entradas estén estables cuando se llega el flaco
- Los flancos pueden ser positivos o negativos
- Señal de capacitación de escritura de un registro
  - Cuando no se escribe en todos los ciclos
  - Estas señales de capacitación las genera la unidad de control

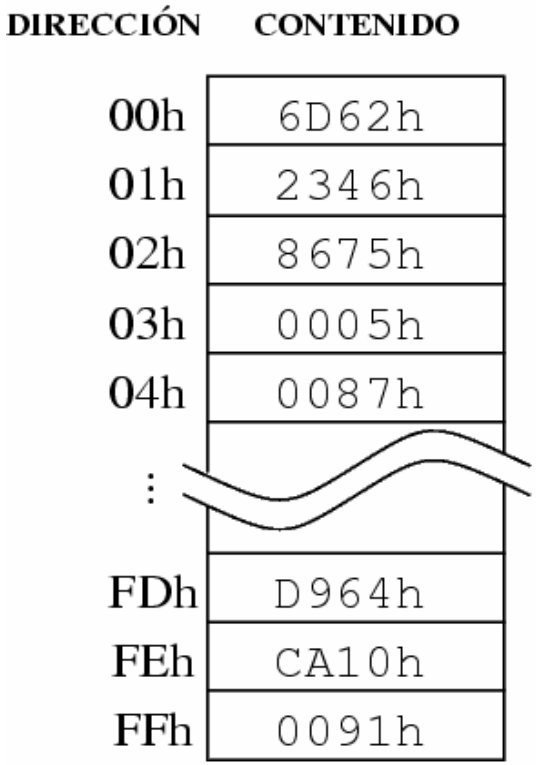
ARQUITECTURA

Francisco Tirado
Roman Hermida

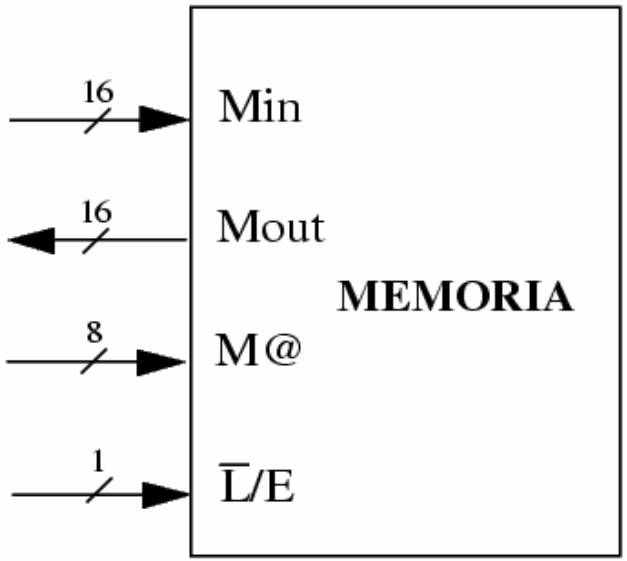
ARQUITECTURA

- **Tipos de datos enteros de 16 bits codificados en C’2**
- **Memoria RAM de 256 palabras de 16 bits**
  - 2 Buses de datos de 16 bits
    - Entrada
    - Salida
  - bus de direcciones de 8 bits
- **Registro de instrucciones de 16 bits**
- **Secuenciamiento implícito**
  - Contador de programa de 8 bits
- **Banco de 8 registros de 6 bits cada uno (R0 -R7)**
  - Un puerto de entrada
  - Un puerto de salida
- **Instrucciones de tres operandos**
- **Modelo de ejecución registro - registro**
- **Dos indicadores de condición: cero y signo**

Memoria: dirección, contenido e interfaz

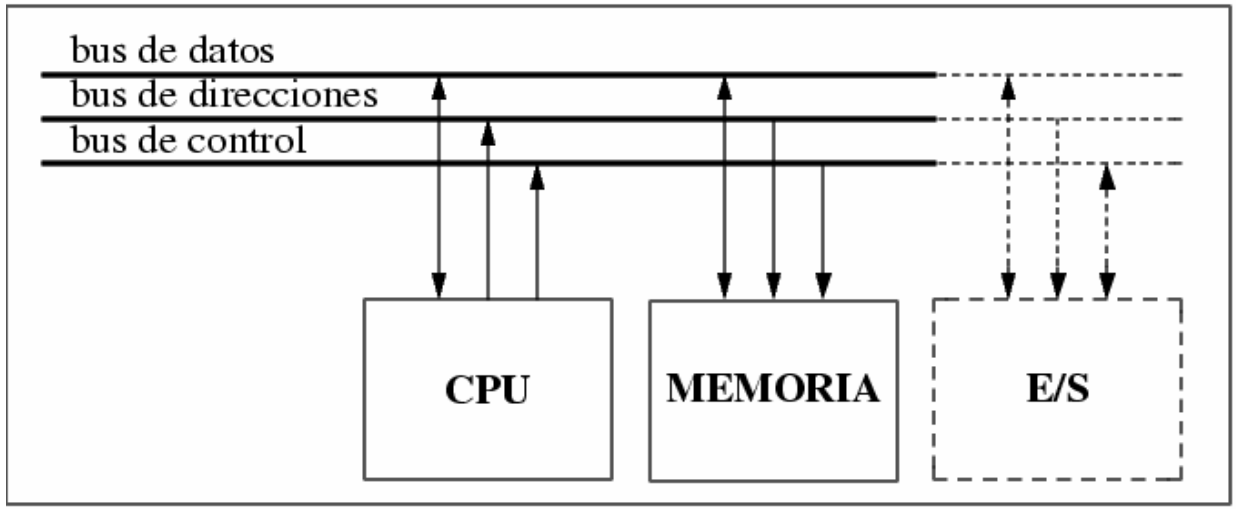


(a)

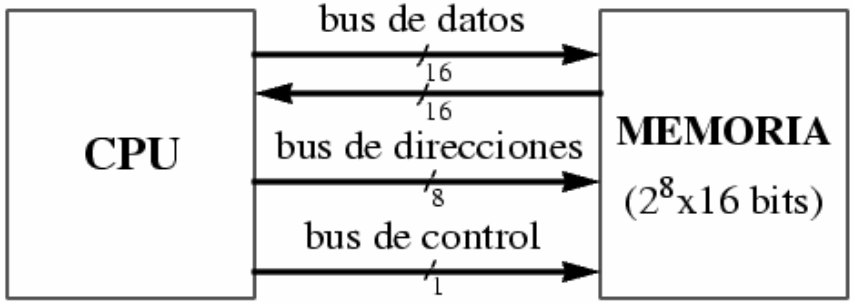


(b)

Comunicaciones entre las unidades de la MR

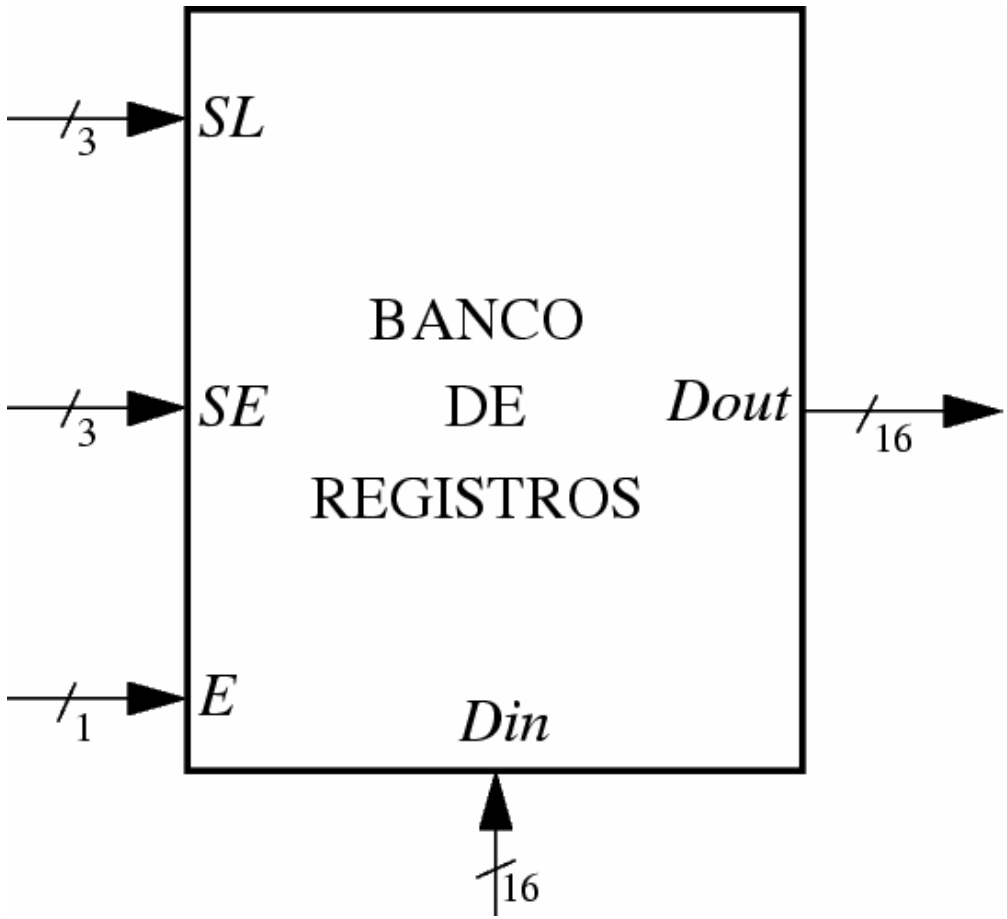


(a)



(b)

Banco de registros de la Máquina Rudimentaria



Din - Dato de Escritura

Dout - Dato de Lectura

E - Permiso de Escritura

SE - Selección de Escritura

SL - Selección de Lectura

Diseño interno del banco de registros

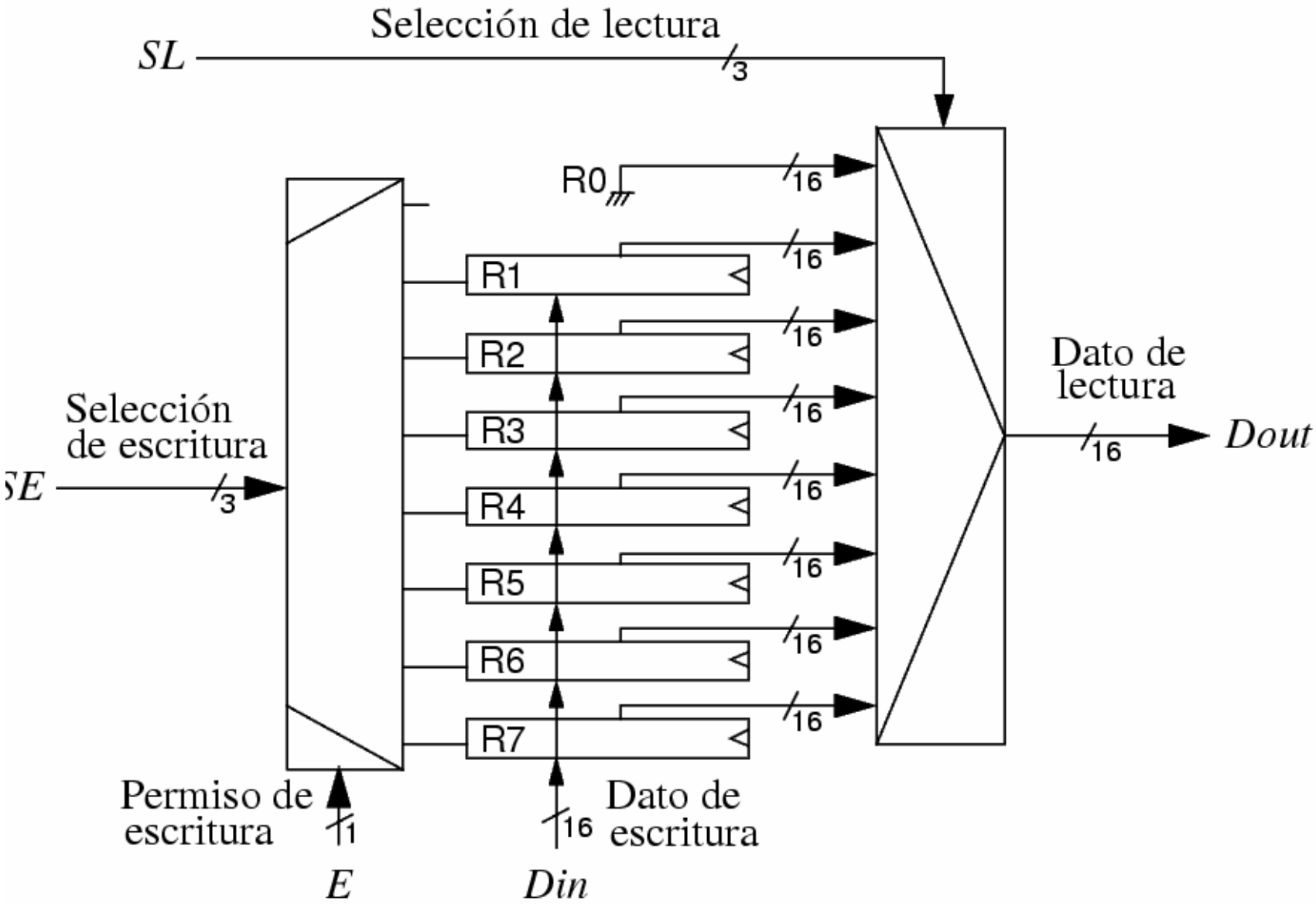
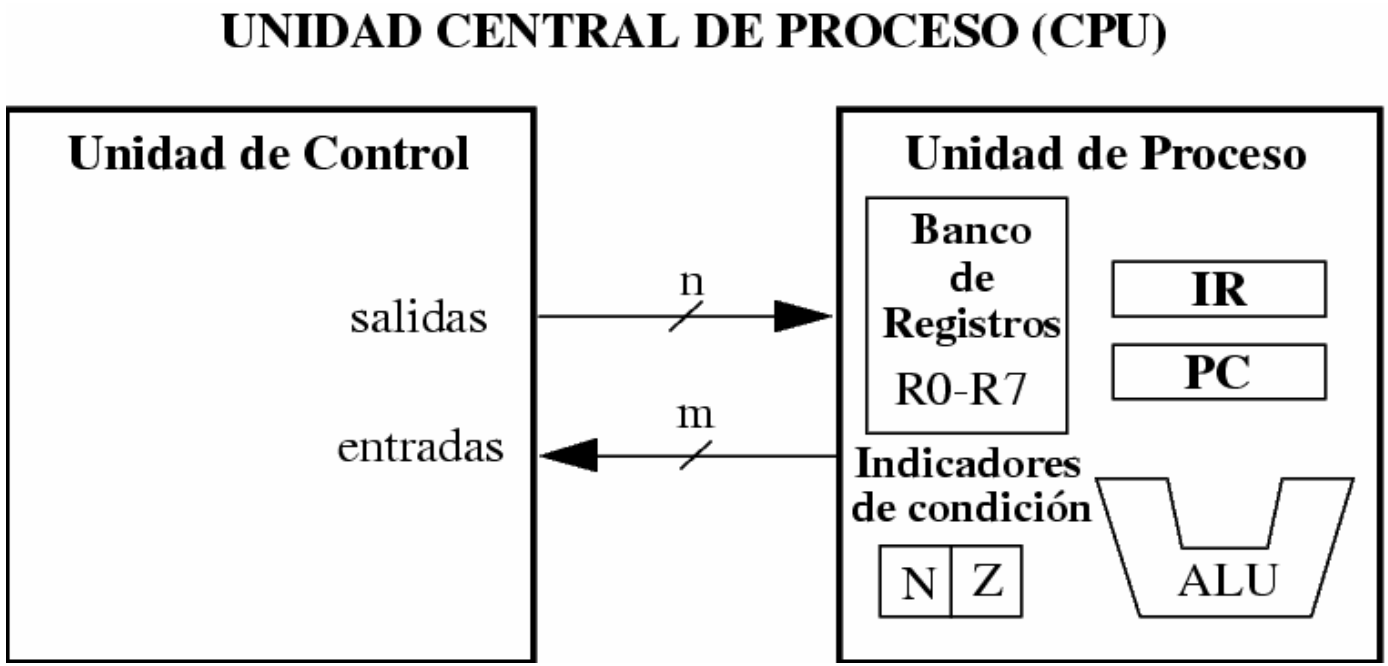


Diagrama de bloques de la CPU



Repertorio de instrucciones

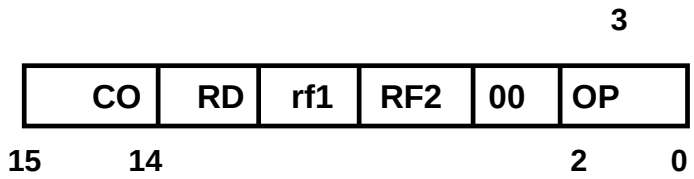
- **Aritmético lógicas de dos operandos en registro**
  - **ADD Rf1, Rf2, Rd**
  - **SUB Rf1, Rf2,Rd**
  - **ASR Rf2,Rd**
  - **AND Rf1, Rf2,Rd**
  - **modo de direccionamiento directo a registro**
- **Aritmético lógicas con inmediato**
  - **ADDI Rf1, #n,Rd**
  - **SUBI Rf1,#n,Rd**
  - **modo de direccionamiento directo primer operando directo a registro**
- **movimiento**
  - **LOAD A(Ri),Rd**
  - **STORE Rf, A(Ri)**
  - **modo de direccionamiento relativo a registro índice**
- **bifurcación condicional**
  - modo de direccionamiento directo a memoria

BL bifurcar si es menor	BG bifurcar si es mayor
BEQ bifurcar si es igual	BNE bifurcar si es distinto
BLE menor o igual	BGE mayor o igual
BR salto incondicional	

formato

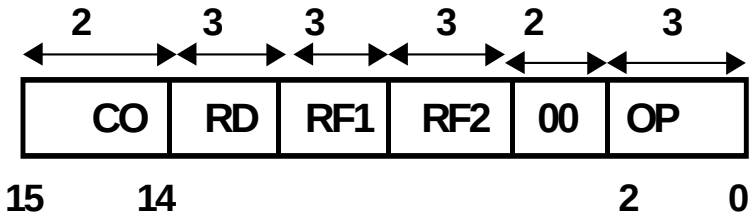
- Formato es la forma en al que se especifica el significado de cada bit de la instrucción
- Información:
  - Operación
  - Dirección de operandos
  - Dirección de resultado
- El formato se divide en campos. Cadena de bits continuos que codifican una determinada información

- CO código de operación de las instrucciones
  - 00 load
  - 01 store
  - 10 salto
  - 11 aritmético - lógicas



Instrucciones aritméticas de dos operandos

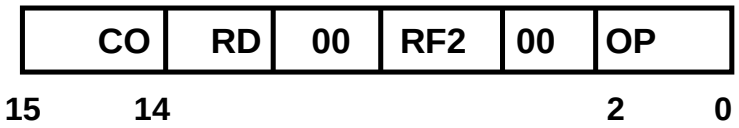
ADD Rf1, Rf2, Rd
SUB Rf1, Rf2,Rd
ASR Rf2,Rd
AND Rf1, Rf2,Rd



CODIFICACIÓN

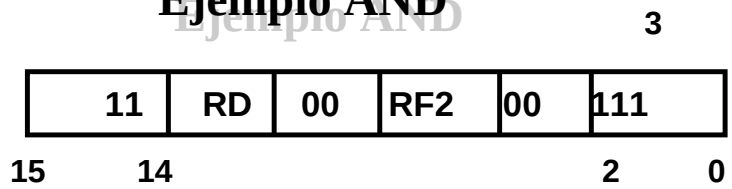
- CO código de operación
  - 11 aritmético - lógicas
- OP (a la UAL)
  - 100 ADD
  - 101 SUB
  - 110 ASR
  - 111 AND

ASR



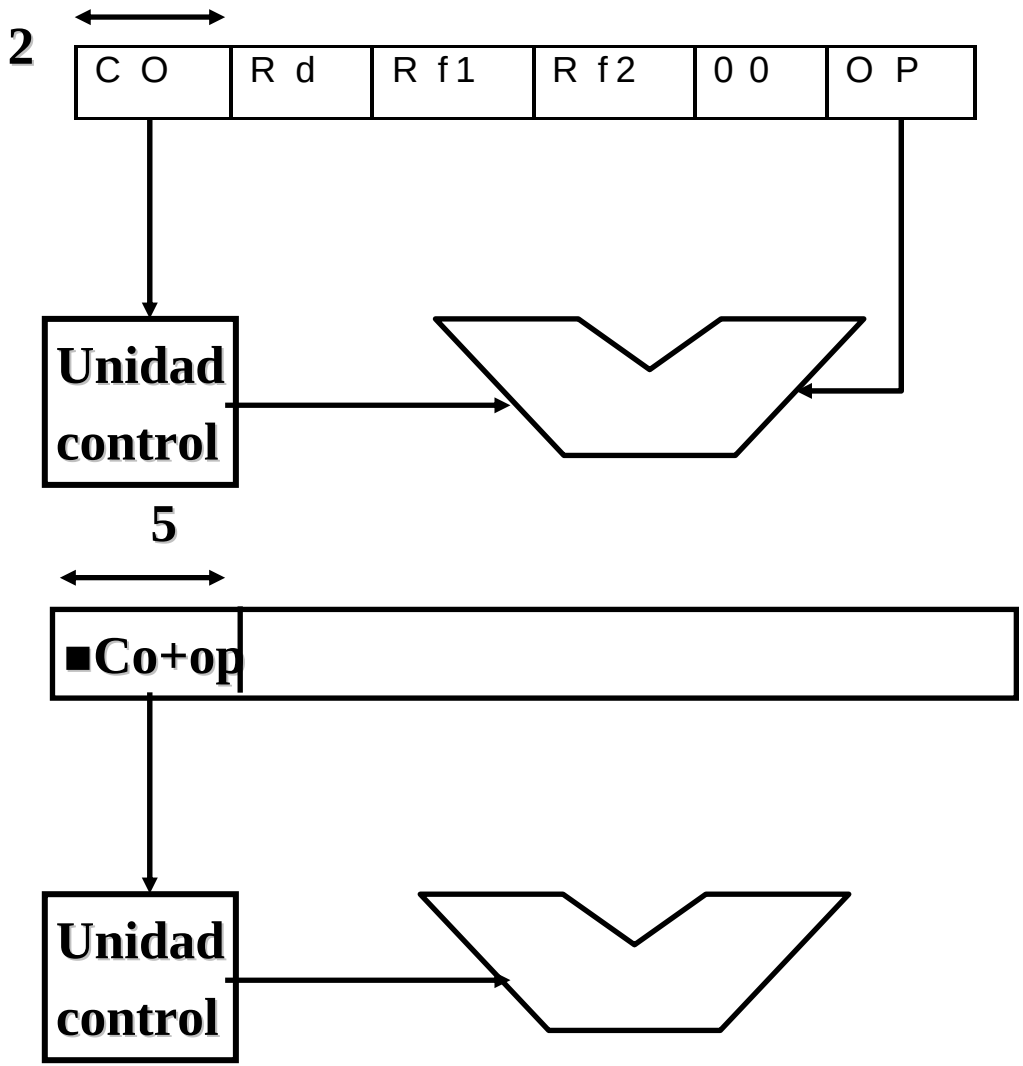
Modo de direccionamiento directo de registro

Ejemplo AND

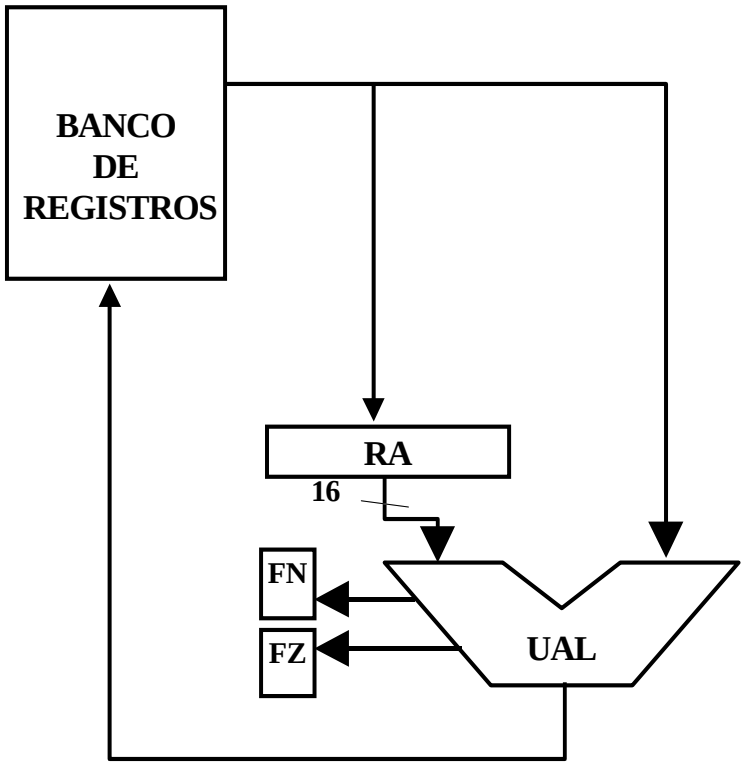


Instrucciones aritméticas de dos operandos

Decisión de diseño que afecta al UC



Instrucciones aritméticas de dos operandos



ADD Rf1, Rf2, Rd

- Ra :=Rf1
- Rd:=Ra+Rf2
- Actualiza N y Z

SUB Rf1, Rf2,Rd

- Ra<--Rf1
- Rd<--Ra-Rf2
- Actualiza N y Z

ASR Rf,Rd

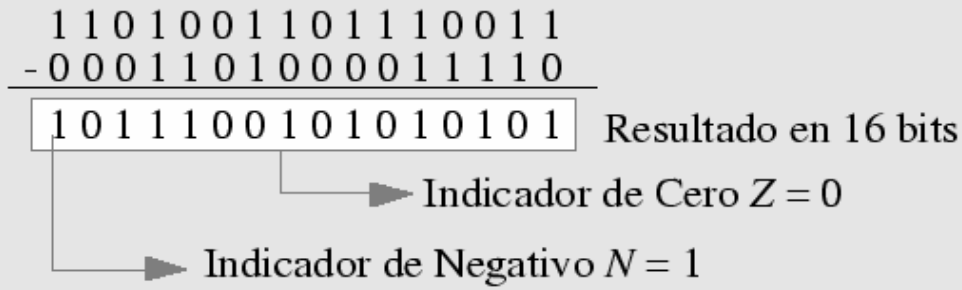
- Ra:=Rf1
- Rd:=deplaza(Rf2)
- Actualiza N y Z

AND Rf1, Rf2,Rd

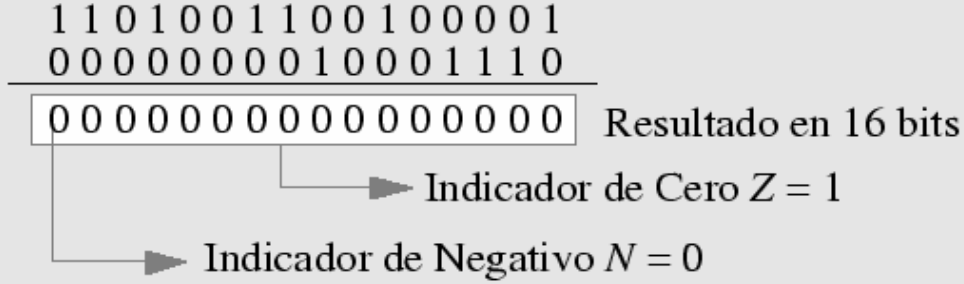
- Ra:=Rf1
- Rd:=Ra and Rf2
- Actualiza N y Z

Instrucciones aritmético-lógicas: ejemplos

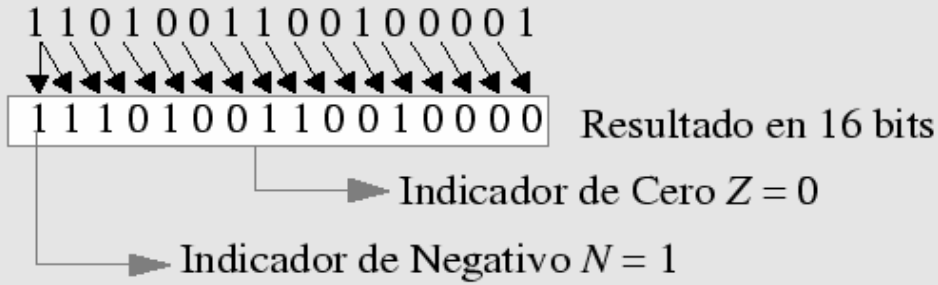
Resta



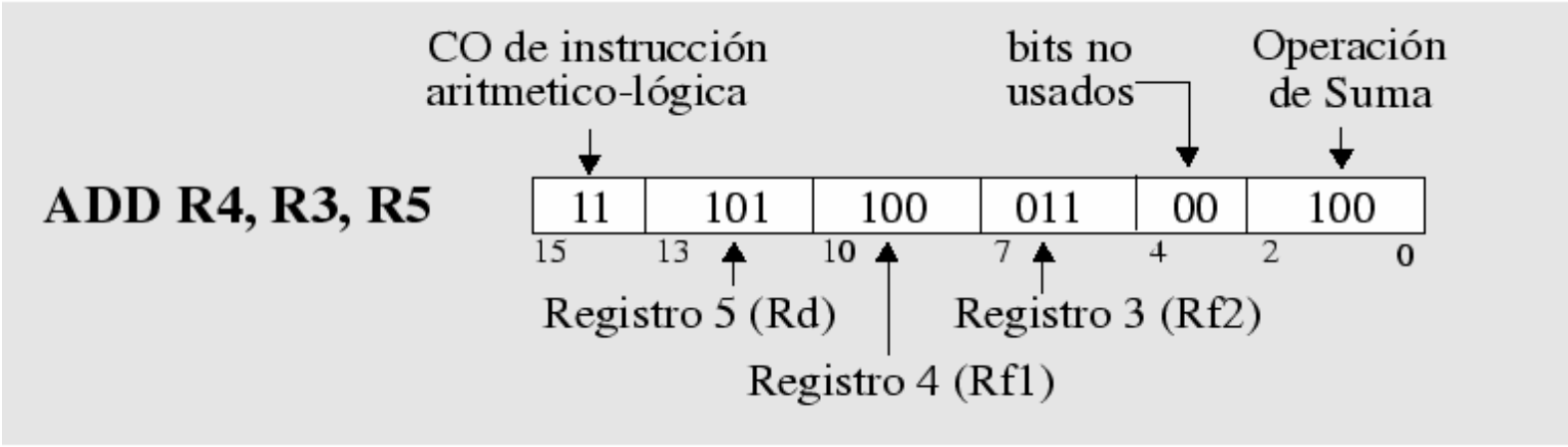
***And* lógica**



Desplazamiento aritmético de un bit a la derecha



Ejecución de una instrucción de suma



Antes de la ejecución:

1000 1110 1111 1011
Registro 3

x
Z

0011 1000 1010 1010
Registro 4

x
N

xxxx xxxx xxxx xxxx
Registro 5

Después de la ejecución:

1000 1110 1111 1011
Registro 3

0
Z

0011 1000 1010 1010
Registro 4

1
N

1100 0111 1010 0101
Registro 5

Aritméticas con Inmediato

FORMATO

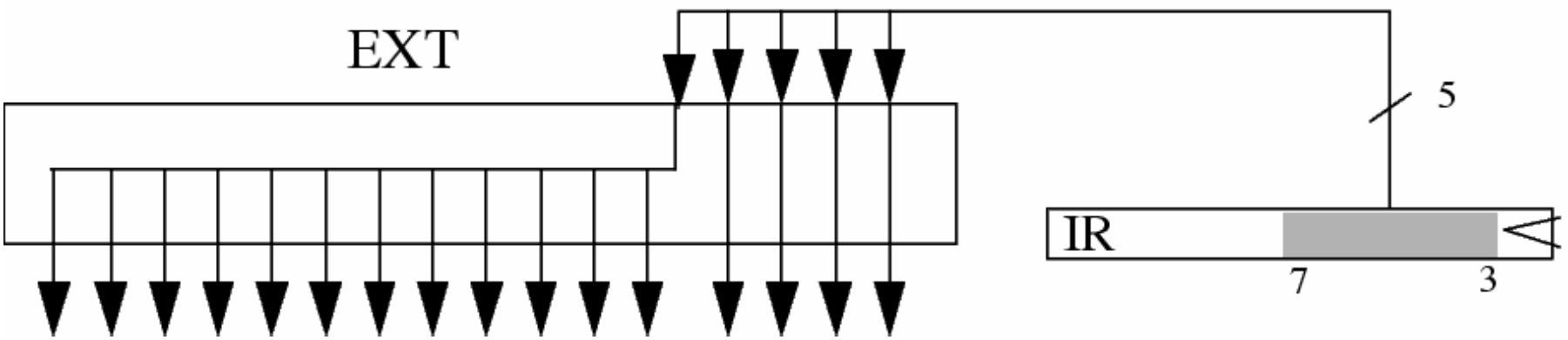
ADDI Rf, #n,Rd
SUBI Rf,#n,Rd

- **CO 11**
- **OP**
  - 000 suma
  - 001 resta

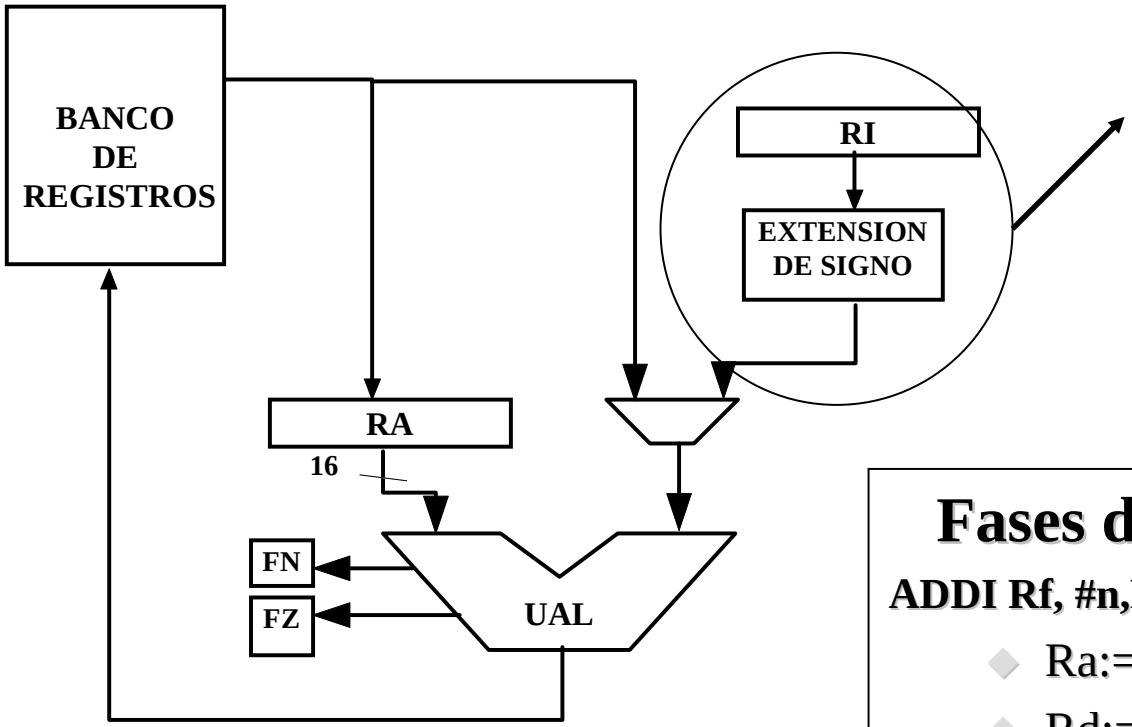
CO	Rd	Rf1	INMEDIATO	OP

Modos de direccionamiento
inmediato (5 bits)
directo a registro

Extensión de signo del operando inmediato



SUBCAMINO ARITMETICAS CON INMEDIATO



Implementa el modo de direccionamiento inmediato

El direccionamiento directo a registro se ve más adelante

Fases de la instrucción

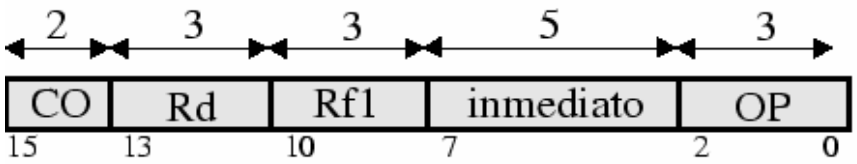
ADDI Rf, #n,Rd

- Ra:=Rf1
- Rd:=Ra + exte(#n)
- Actualiza N y Z

SUBI Rf,#n,Rd

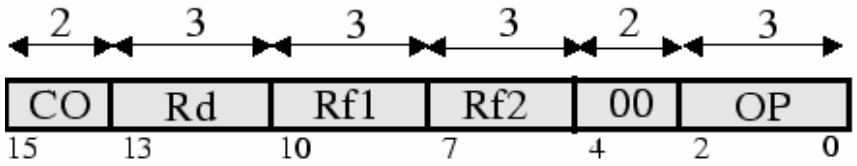
- Ra:=Rf1
- Rd:=Ra - exte(#n)
- Actualiza N y Z

Formato de las instrucciones aritmético-lógicas



Formato de las instrucciones de suma y resta con inmediato.

11	Rd	Rf1	inmediato	000	Suma con inmediato
11	Rd	Rf1	inmediato	001	Resta con inmediato



Formato de las instrucciones de suma, resta, desplazamiento a la derecha y *and* lógica.

11	Rd	Rf1	Rf2	00	100	Suma
11	Rd	Rf1	Rf2	00	101	Resta
11	Rd		Rf2	00	110	Desplazamiento a la derecha
11	Rd	Rf1	Rf2	00	111	*And* lógica

INSTRUCCIONES DE MOVIMIENTO

FORMATO

- **STORE Rf, A(Ri)**

CO	Rd	Ri	dirbase

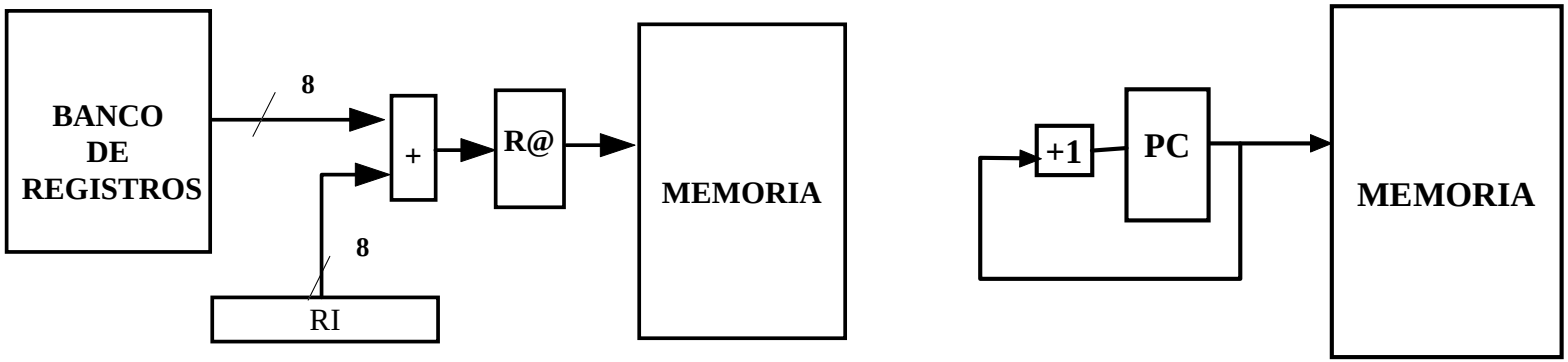
- **LOAD A(Ri),Rd**

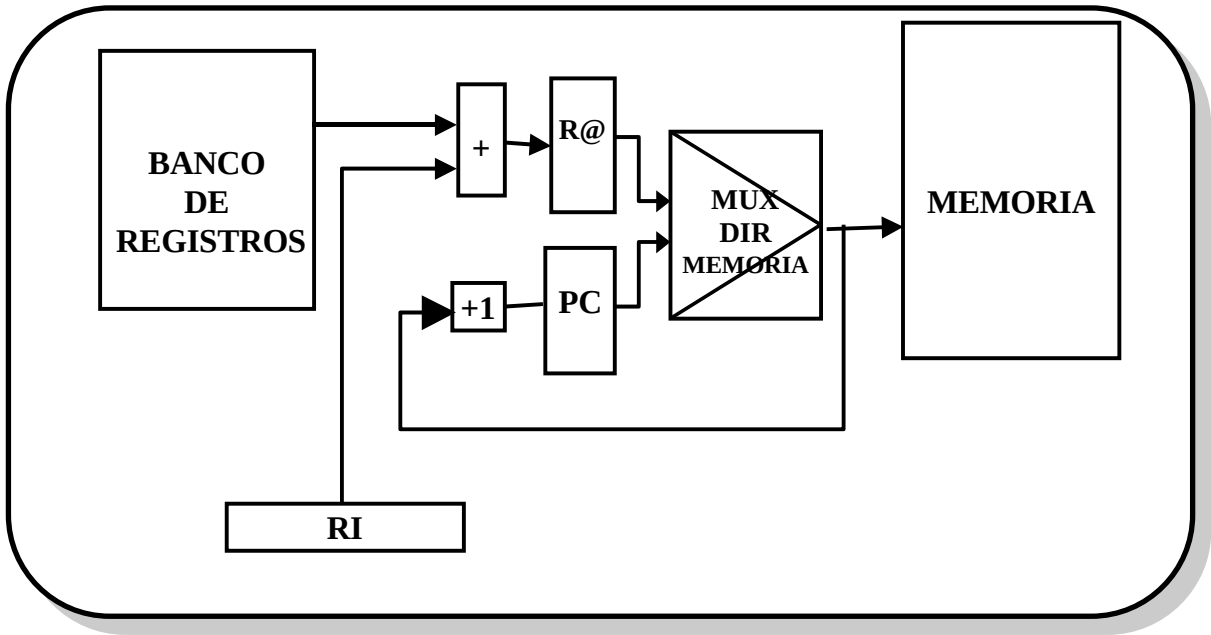
CO	Rf	Ri	dirbase

Modo de direccionamiento relativo a registro índice

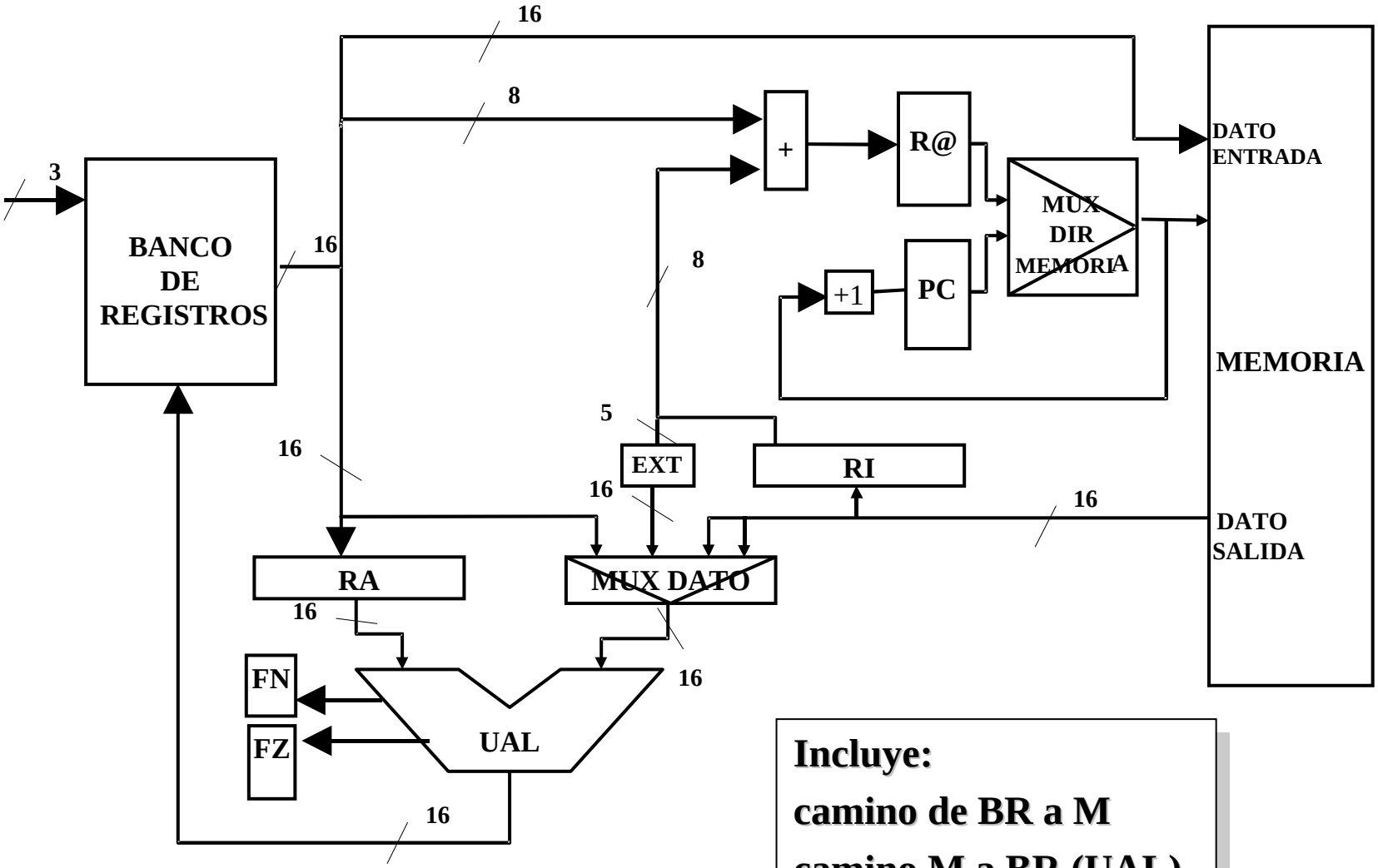
- **Ri**
  - un registro del banco usado para generar la dir
  - Contiene el desplazamiento de 8 bits
- **CO**
  - 00 LOAD
  - 01 STORE

INSTRUCCIONES DE MOVIMIENTO. Unidad Secuenciadora





INSTRUCCIONES DE MOVIMIENTO. Camino de datos



Incluye:

camino de BR a M

camino M a BR (UAL)

U.Secuenciadora

Fases de ejecución

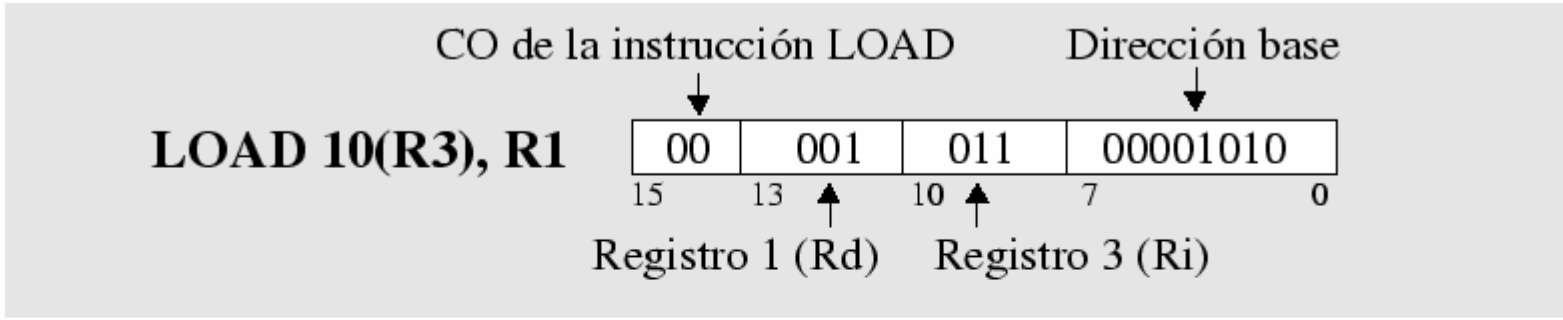
LOAD A(Ri),Rd

- R@:=<A+Ri>
- Rd:=m<R@>
- R@- reg aux de direcciones
- Actualiza N y Z
- este dato llega al banco de registros a través de la UAL

STORE Rf, A(Ri)

- R@ :=<A+Ri>
- m<R@>:=Rf

Ejecución de una instrucción de carga



Antes de la ejecución:

xxxx xxxx xxxx xxxx — Registro 1 — x (Z)

xxxx xxxx 0000 0010 — Registro 3 — x (N)

Dirección	Contenido
: 0Ah	xxxx xxxx xxxx xxxx
: 0Bh	xxxx xxxx xxxx xxxx
: 0Ch	1101 0100 0101 0101

Dirección base + desplazamiento

MEMORIA

Después de la ejecución:

1101 0100 0101 0101 — Registro 1 — 0 (Z)

xxxx xxxx 0000 0010 — Registro 3 — 1 (N)

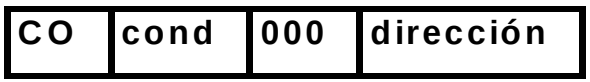
Dirección	Contenido
: 0Ah	xxxx xxxx xxxx xxxx
: 0Bh	xxxx xxxx xxxx xxxx
: 0Ch	1101 0100 0101 0101

MEMORIA

Instrucciones de salto. Formato

BL bifurcar si es menor	BG bifurcar si es mayor
BEQ bifurcar si es igual	BNE bifurcar si es distinto
BLE menor o igual	BGE mayor o igual
BR salto incondicional	

Modo de direccionamiento directo a memoria

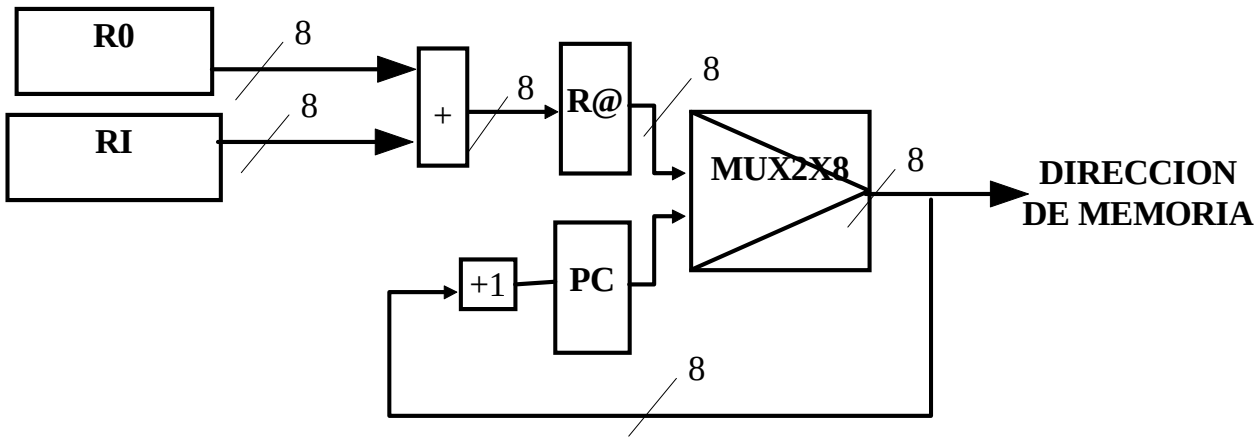


Objetivo
ahorro de HW
simplicidad de UC

- **CO =10**
- **CONDICIÓN=RI<13:11>**
  - **000 incondicional**
  - **001 igual**
  - **010 menor**
  - **011 menor o igual**
  - **101distinto**
  - **110 si mayor o igual**
  - **111 si mayor.**

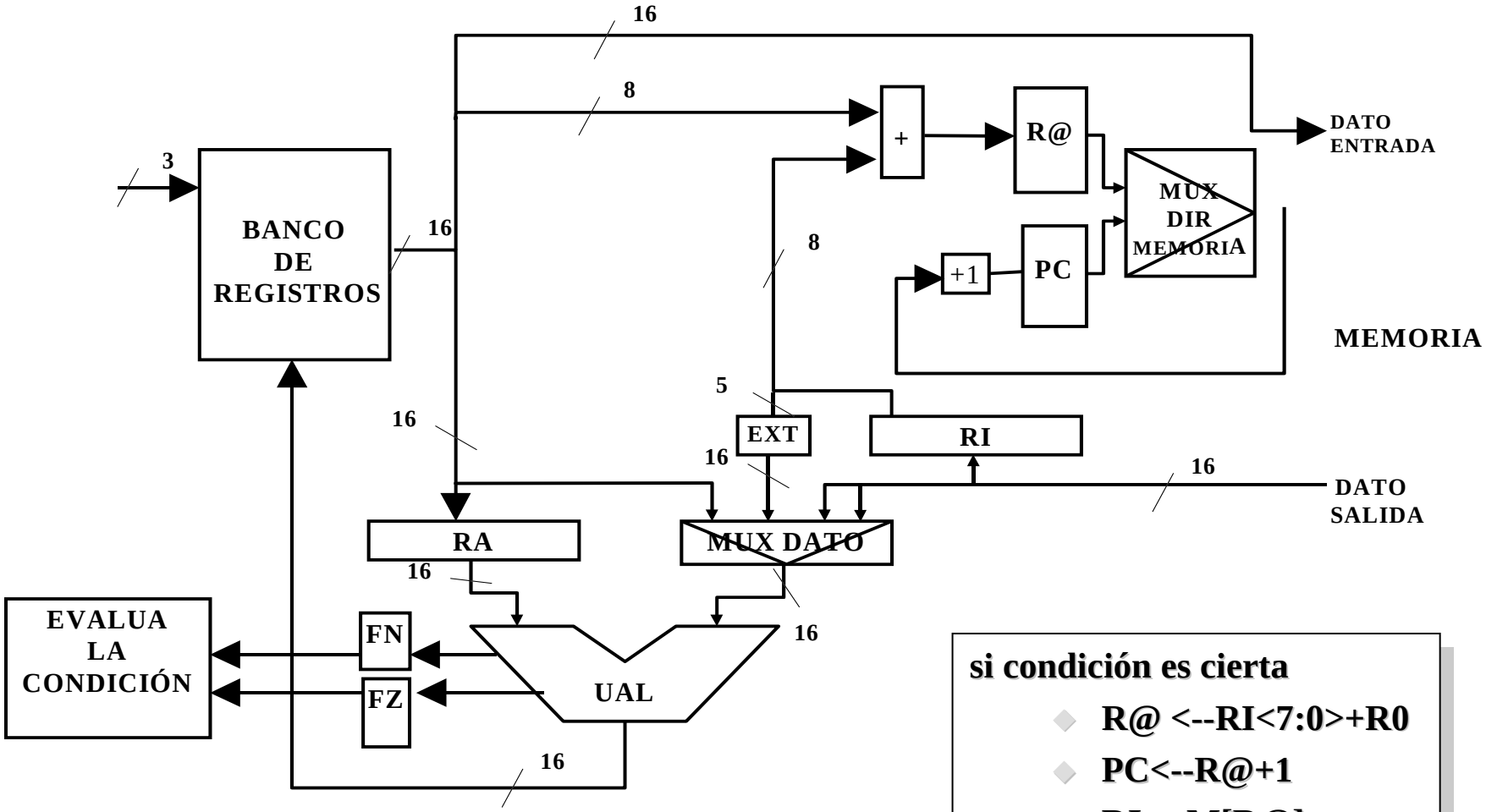
Instrucciones de salto. Unidad Secuenciadora

Implementa el modo de direccionamiento directo a memoria



Luego la unidad secuenciadora no se ve modificada selección del R0 por unidad de control

Instrucciones de salto. Camino de datos



si condición es cierta

- R@ <--RI<7:0>+R0
- PC<--R@+1
- RI<--M[R@]

si no

- PC<--PC+1

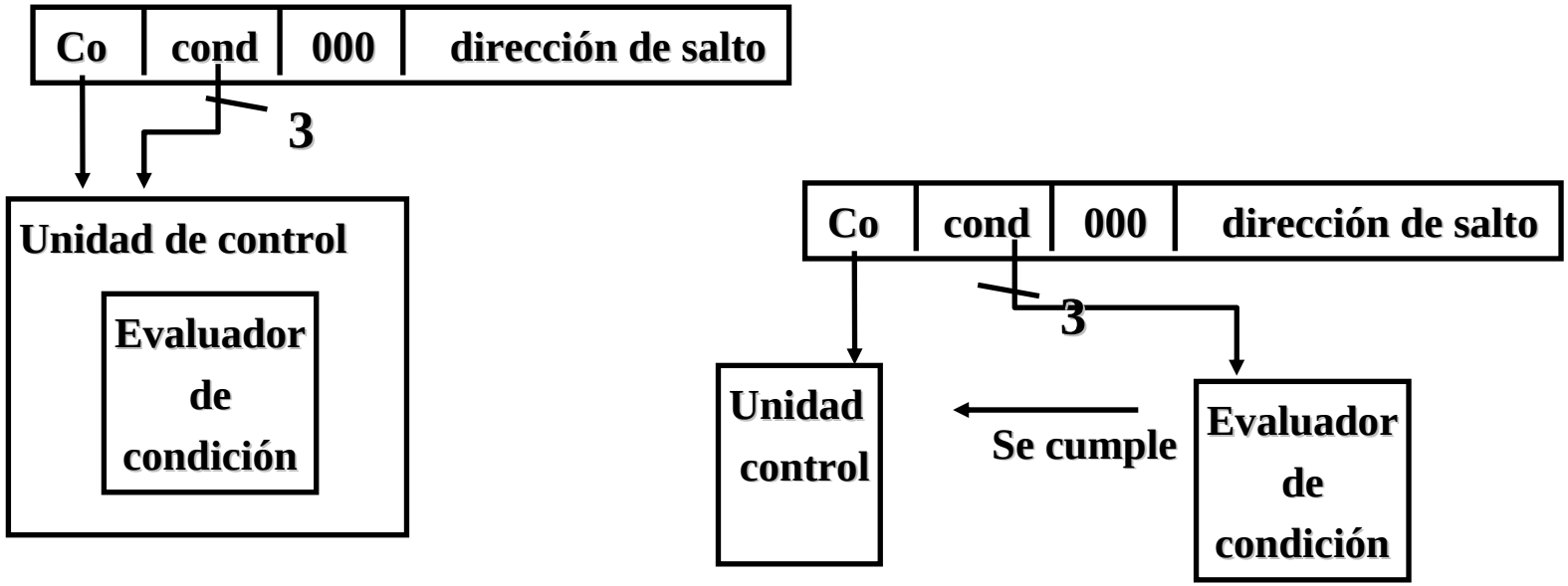
Las instrucciones de bifurcación no modifican la unidad secuenciadora

atención al evaluador

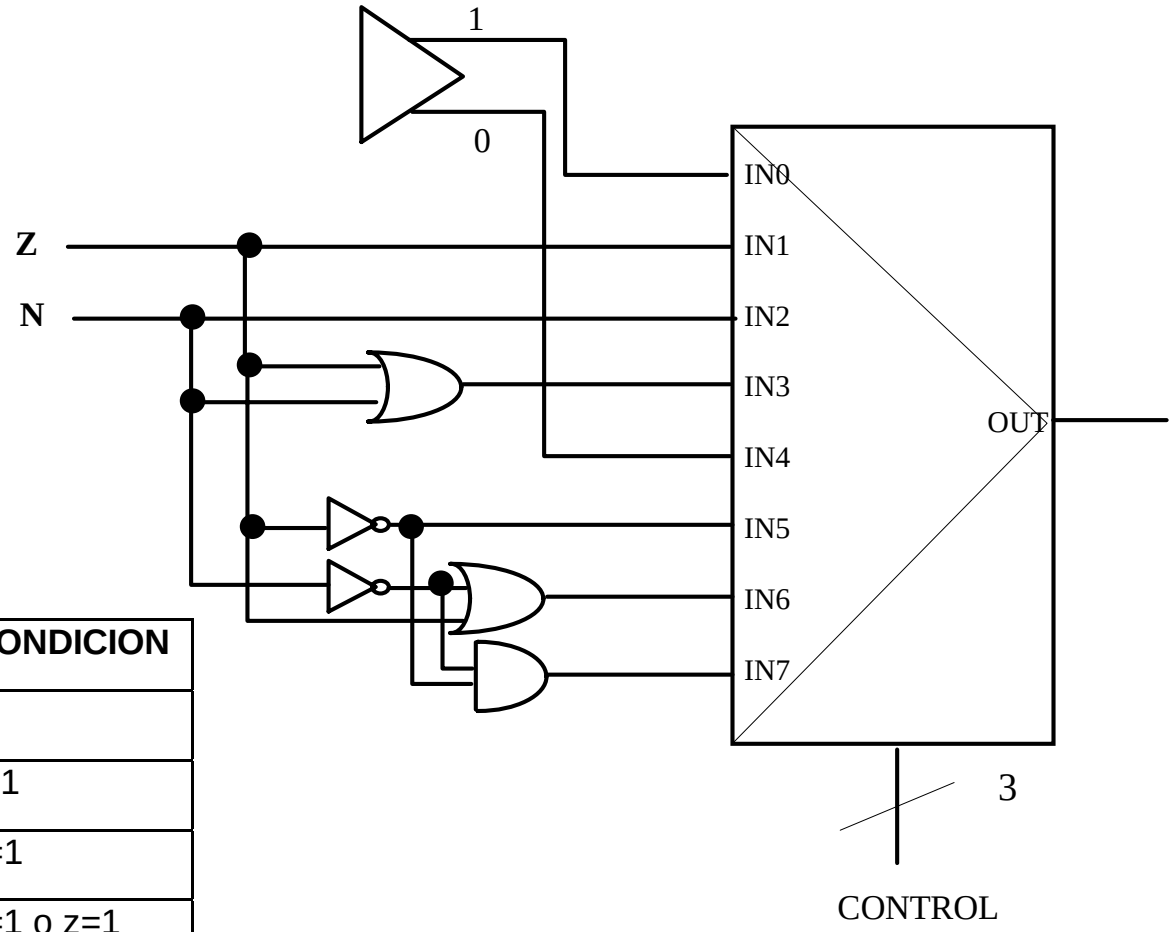
Instrucciones de salto. Decisiones de diseño

Decisión de diseño que simplifican la U.C

Sacar el evaluador de condiciones fuera de la Unidad de Control

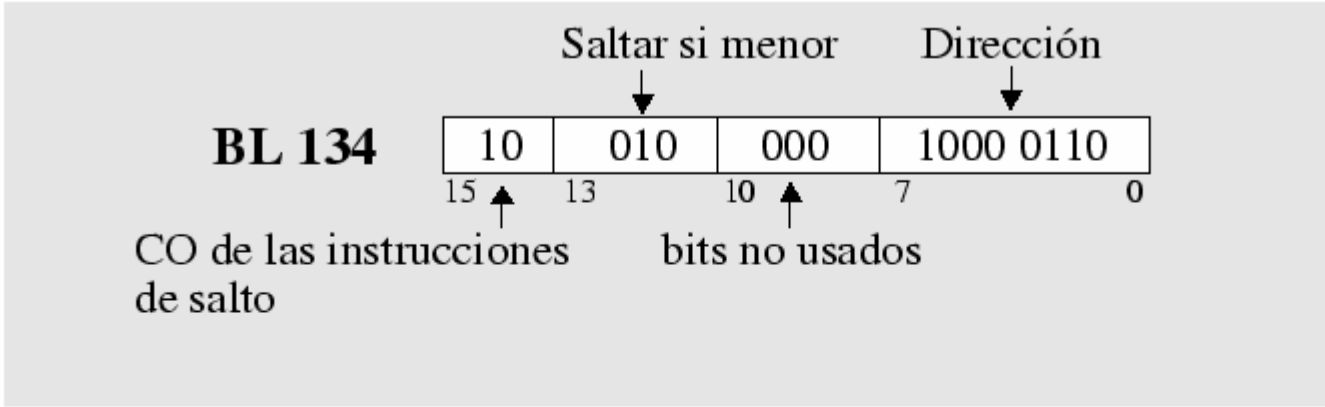


Instrucciones de Salto. Módulo evaluador

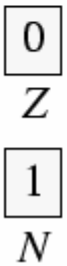


TIPO DE SALTO	CODIGO	CONDICION
BR salto incondicional	000	1
BEQ si igual	001	Z=1
BL si menor	010	N=1
BLE menor o igual	011	N=1 o z=1
BNE si distinto	101	Z=0
BGE si mayor o igual	110	N=0 o Z=1
BG si mayor	111	N=0 y Z=0

Ejemplo: Salto si menor

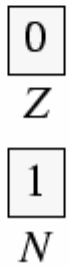


Antes de la ejecución:



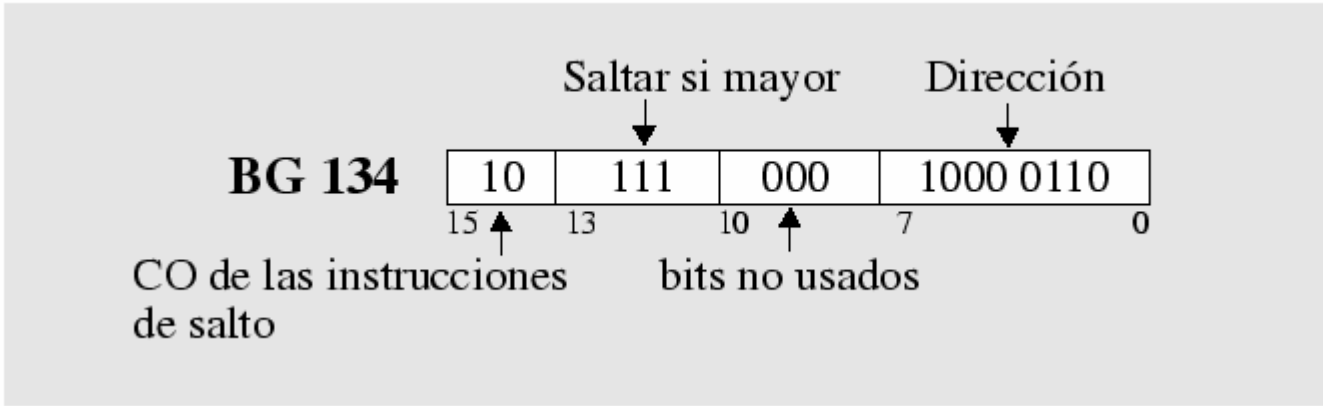
Se ha de realizar el salto a la dirección 134, ya que N=1.

Después de la ejecución:



La siguiente instrucción a ejecutar se encuentra en la dirección 134.

Salto si mayor



Antes de la ejecución:

1
Z
0
N

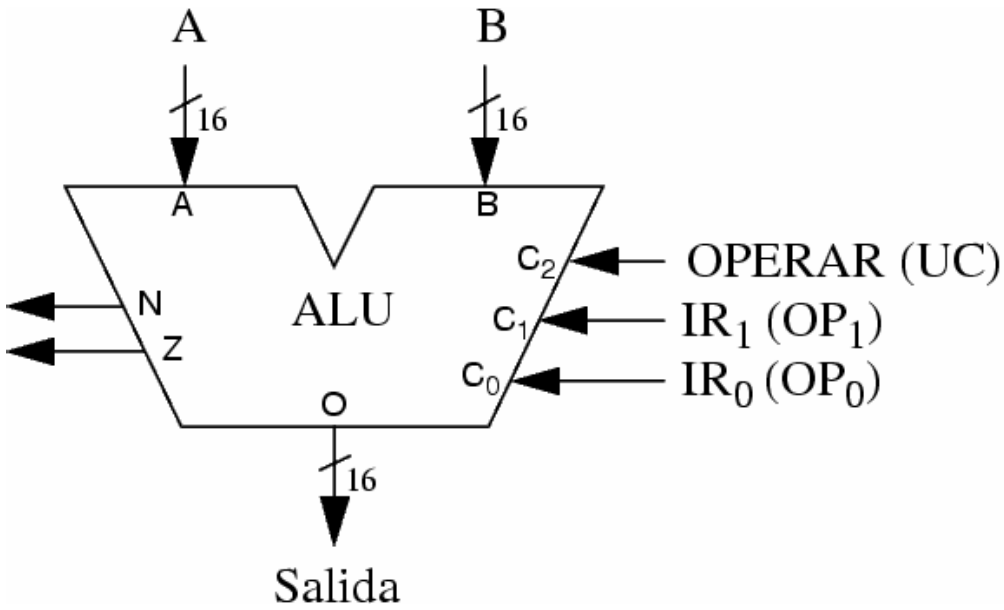
No se ha de realizar el salto a la dirección 134, ya que *Z*=1.

Después de la ejecución:

1
Z
0
N

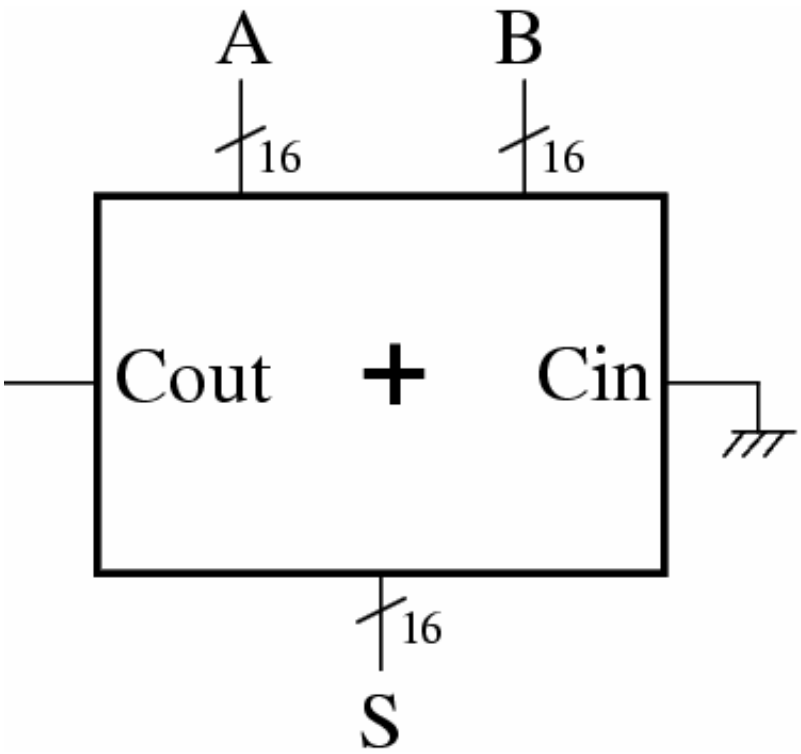
La siguiente instrucción a ejecutar es la inmediatamente posterior a la instrucción de salto.

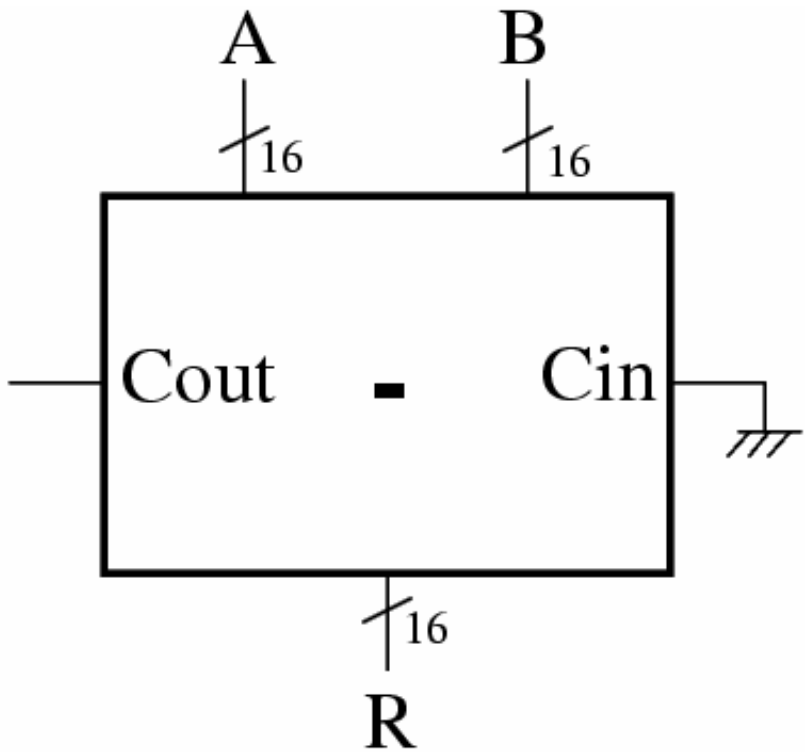
Especificaciones de la ALU



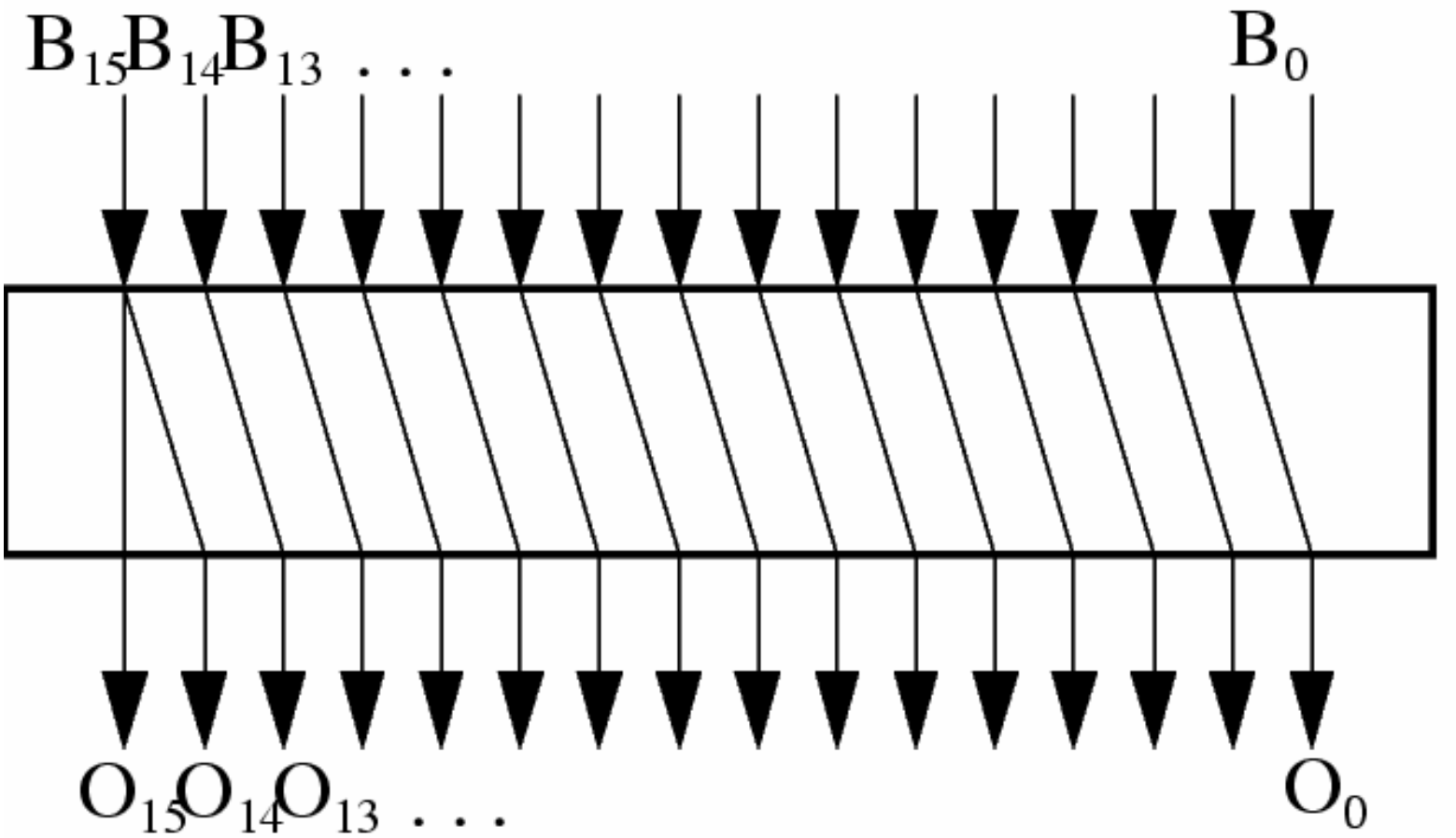
C_2	C_1	C_0	Operación	
0	x	x	B	(dejar pasar B)
1	0	0	A+B	(suma)
1	0	1	A-B	(resta)
1	1	0	B>>1	(desp. derecha)
1	1	1	A and B	(and lógica)

Sumador y restador de 16 bits usados en la ALU

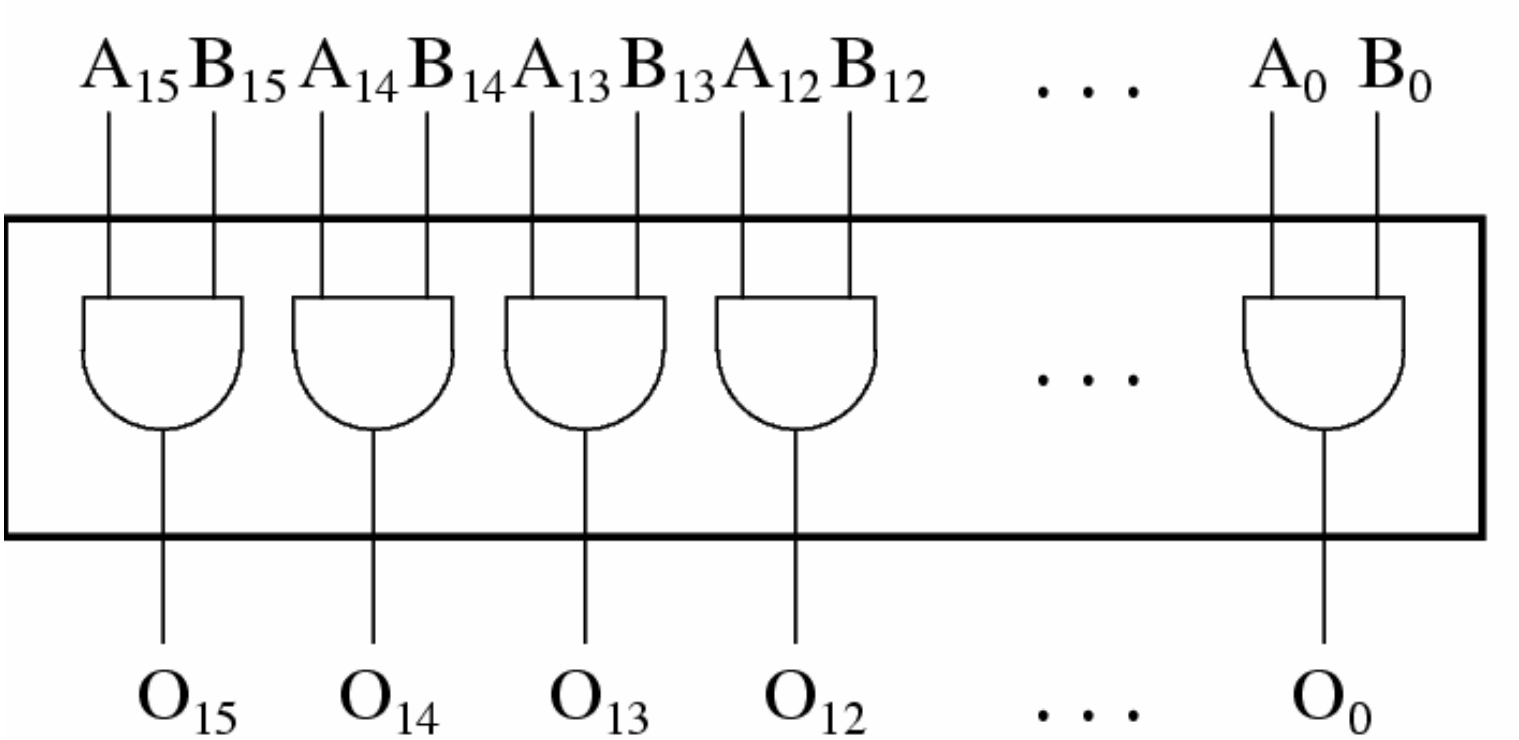




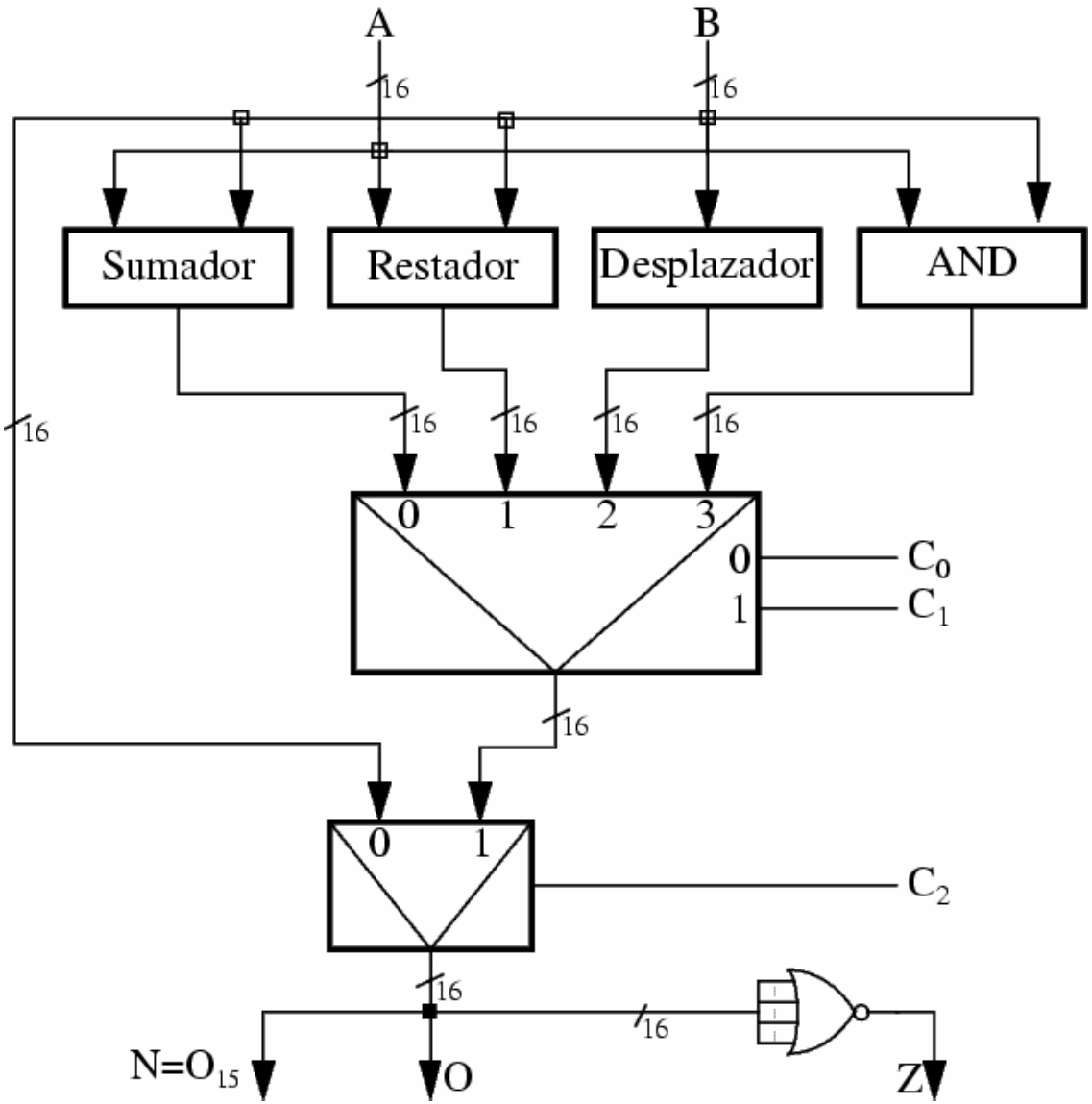
Desplazador aritmético de 1 bit a la derecha usado en la ALU



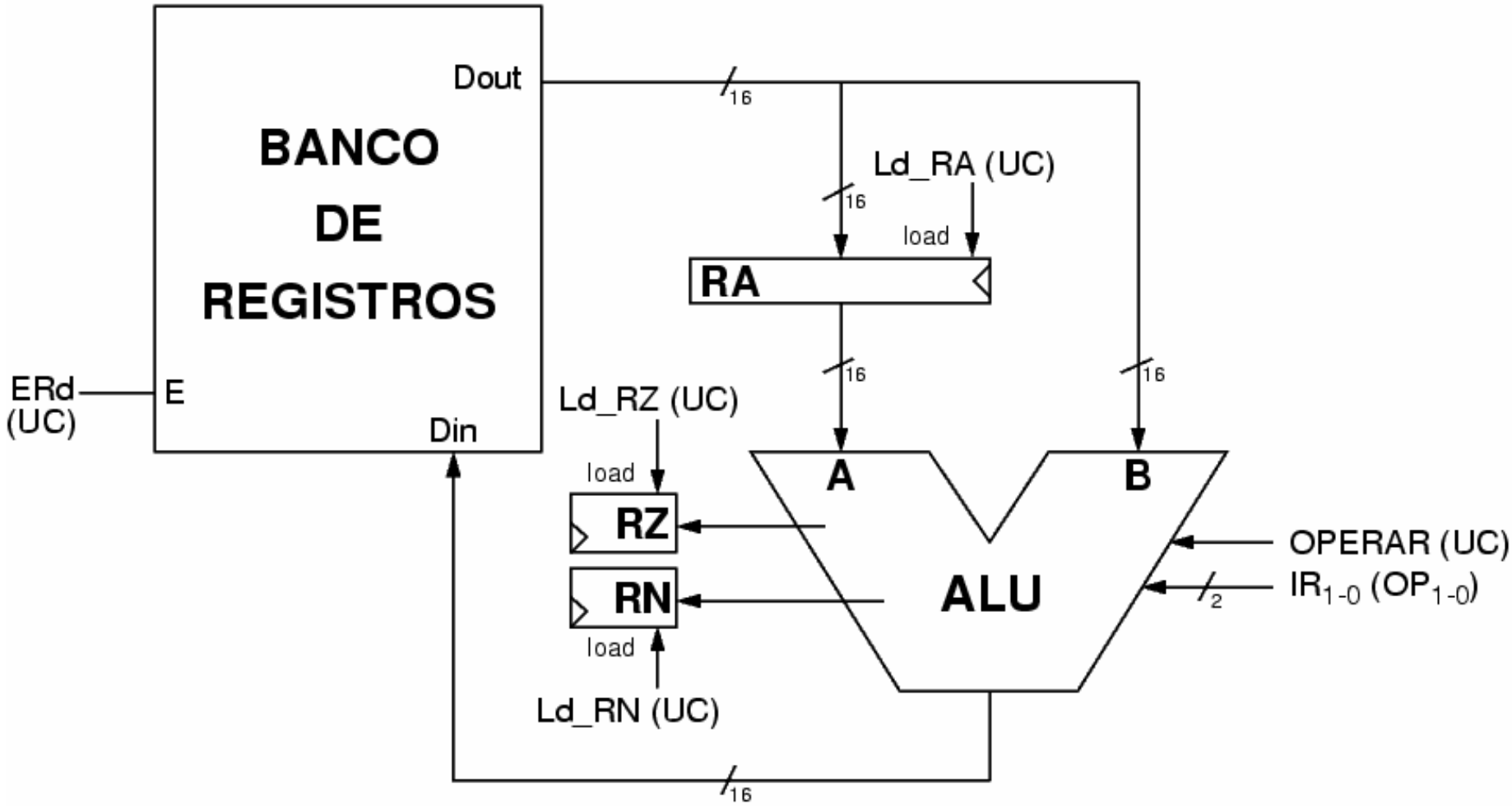
Realización de la and lógica de dos operandos de 16 bits



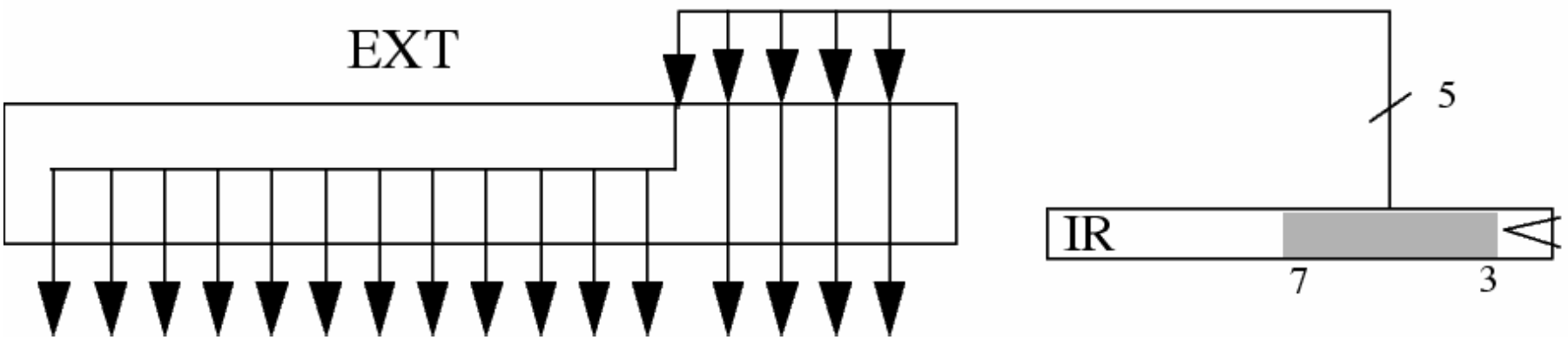
Diseño interno de la ALU



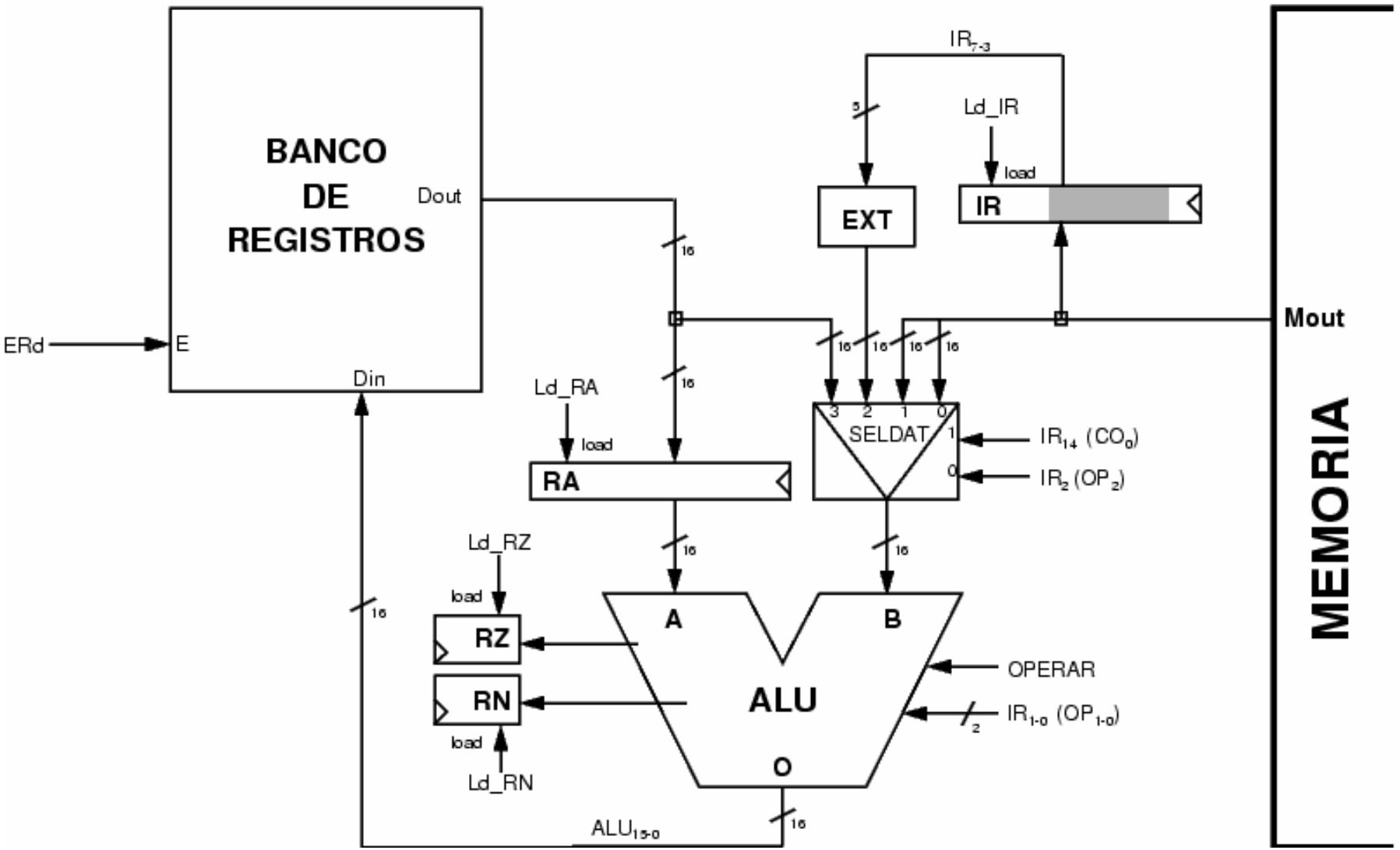
Interconexión de la ALU con el banco de registros, RN y RZ



Extensión de signo del operando inmediato



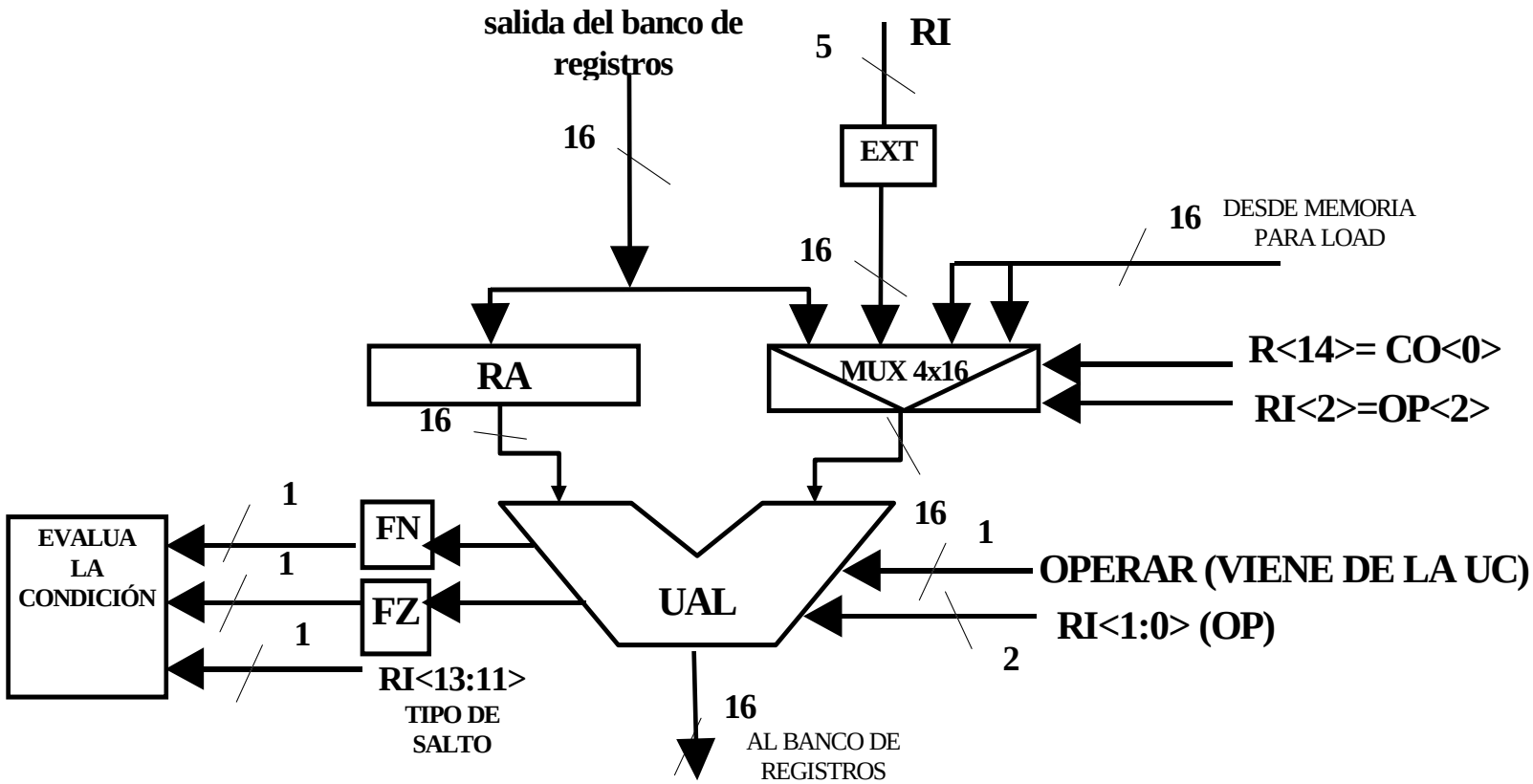
Interconexión de la ALU con el resto de bloques



Decisiones para simplificar la Unidad de Control

Todo el control que se pueda tomar directamente desde el RI se toma

- **Control del multiplexor del segundo operando**
- **Control de la UAL**



Selección Segundo Operando. Multiplexor de entrada a la ALU

- **Controlmux<1,0> =< RI<14>, RI<2> >**

- **Con RI<14> (CO<0>) selecciono: el tipo de operación**
  - 1.- Una operación aritmética
    - Entradas 3 y 2
    - Se decide cual mediante el RI<2>
  - 0.-Una operación de load
    - Entradas 1 y 0 que vienen de la memoria

- **Con RI<2> (OP<2>) selecciono dentro de las aritméticas**
  - 0 entrada 2 (el inmediato del RI)
  - 1 entrada 3 del registro
  - Recordar
    - 000 suma con inmediato
    - 100 suma con dos registros

¿ de donde llegan las entradas al mux?

¿ Esa información es fácilmente extraible del RI?

CO código de operación
- 00 load (R<--M)
- 11 aritmético - lógicas

Control de la UAL

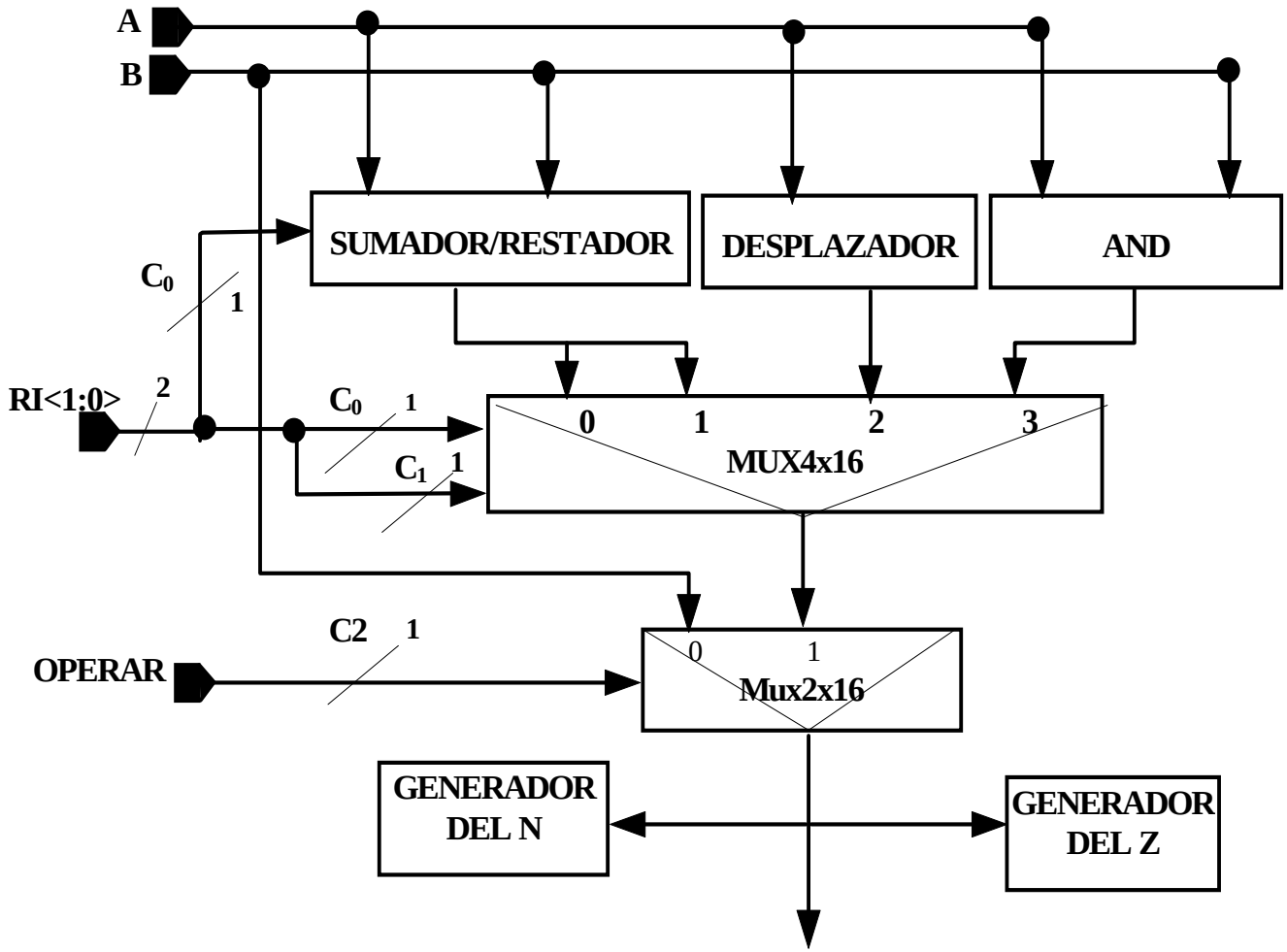
- **Operaciones de la UAL**
  - suma
  - resta
  - desplazamiento
  - and
  - dejar pasar(operaciones de movimiento)

C2	C1	C0	función
0	X	X	PASA B
1	0	0	A+B
1	0	1	A-B
1	1	0	DESP A
1	1	1	AND

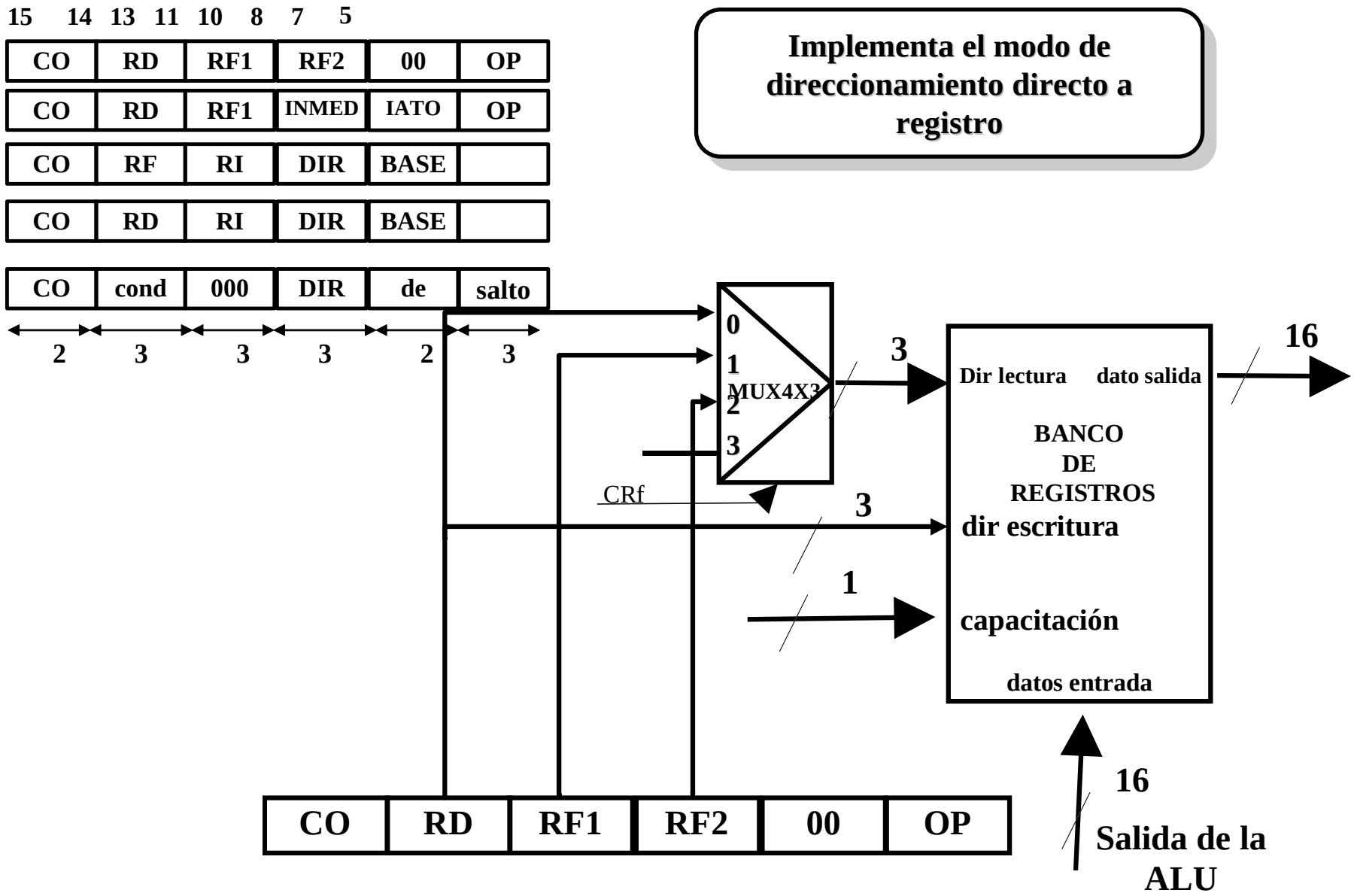
- **C2 es la señal de control operar**
  - esta señal también podría ser la RI<14> pero en este caso no simplifica la UC porque en todo caso la unidad debe saber que tipo de operación se está realizando

- **C1 y C0 vienen directamente del registro de instrucciones y se corresponde con op**

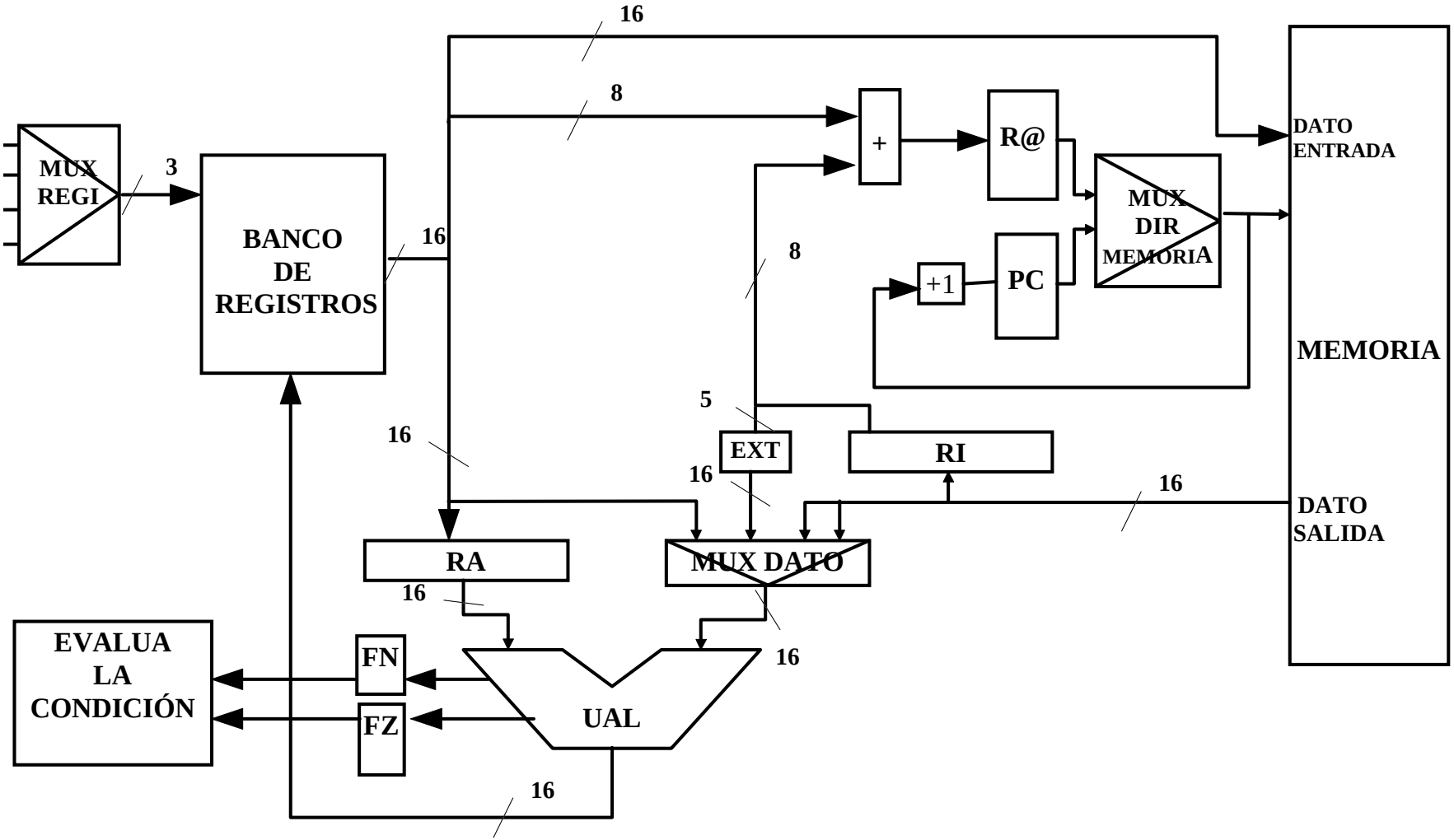
Control de la UAL



Unidad Direccionadora del Banco de Registros

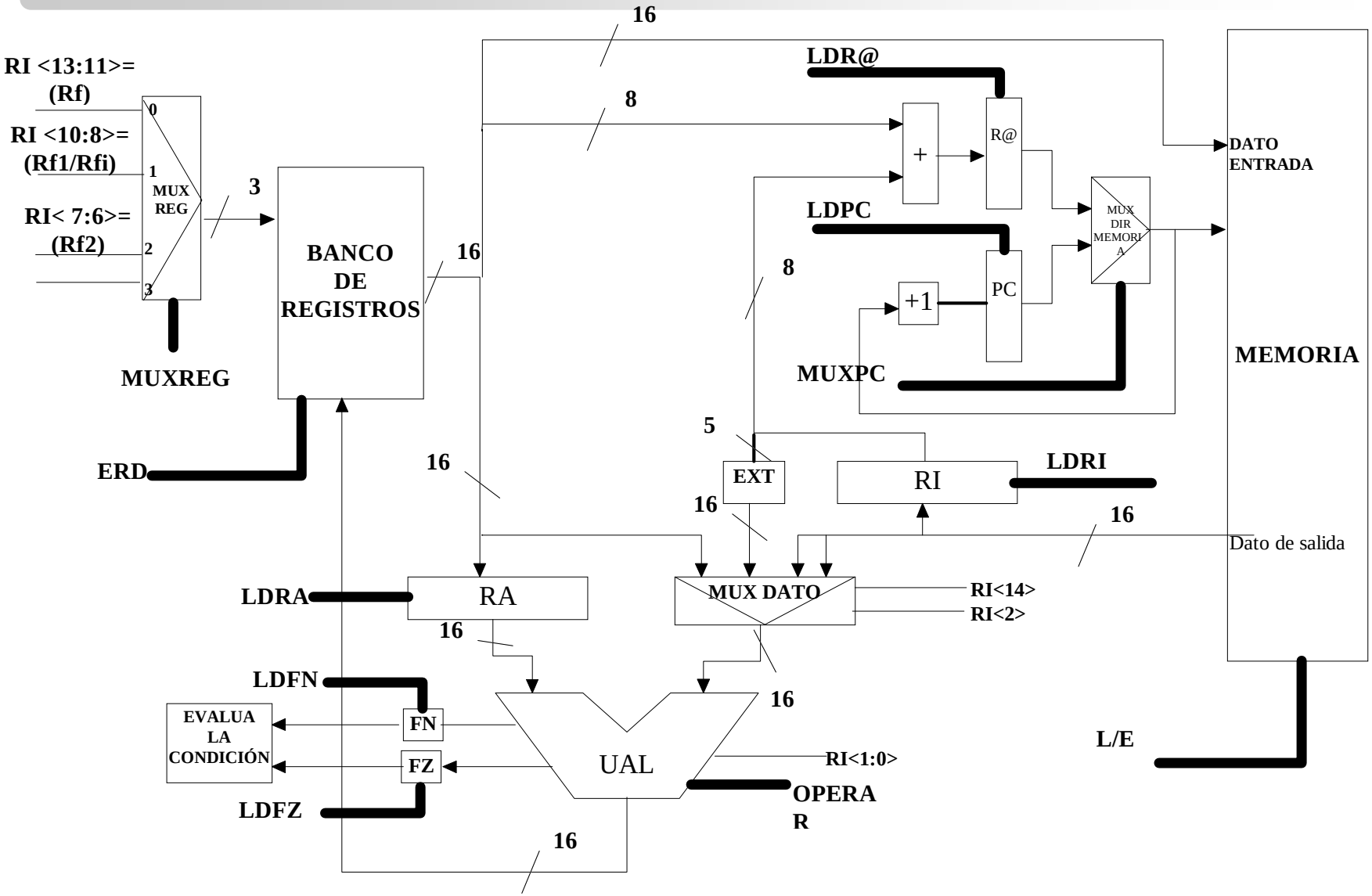


CAMINO DE DATOS TOTAL



SIMPLIFICACION DE LA UC

- **Para simplificar el diseño de la unidad de control se han tomado las siguientes decisiones**
  - Evaluador de condiciones fuera de la uc
  - El formato separa los campos CO y OP
  - El formato separa CO y tipo de bifurcación
  - Todo el control que se pueda tomar directamente desde el RI se toma
  - Regularidad en los formatos


16
LDR@
8
RI <13:11>=
(Rf)
RI <10:8>=
(Rf1/Rfi)
RI< 7:6>=
(Rf2)
0
1
2
3
MUX
REG
3
BANCO
DE
REGISTROS
16
+
R@
DATO
ENTRADA
LDPC
8
MUX
DIR
MEMORIA
+1
PC
MEMORIA
MUXPC
MUXREG
5
ERD
16
EXT
RI
LDRI
16
16
Dato de salida
LDRA
RA
MUX DATO
RI<14>
RI<2>
16
LDFN
16
EVALUA
LA
CONDICIÓN
FN
FZ
UAL
RI<1:0>
L/E
OPERA
R
LDFZ
16

Fases del estudio del Camino de datos

- **Búsqueda de los formatos**
  - Un sistema suele tener más de un formato
  - Búsqueda de instrucciones del mismo tipo
  - Tipos diferentes de direccionamiento por cada tipo de instrucciones

- **Para cada formato estudiar el camino de datos**
  - camino de datos
    - Afecta al movimiento de datos e instrucciones
      - Registros auxiliares
      - Mux
  - Unidad secuenciadora.
    - Afecta a la generación de las direcciones de memoria y del banco de registro.
    - Muy influida por los modos de direccionamiento

- **Estudio de posibles simplificaciones de la Unidad de Control**
  - Señales de control extraibles del propio registro de Instrucciones

ESPECIFICACION
DE LA
UNIDAD DE CONTROL

TEMA 7

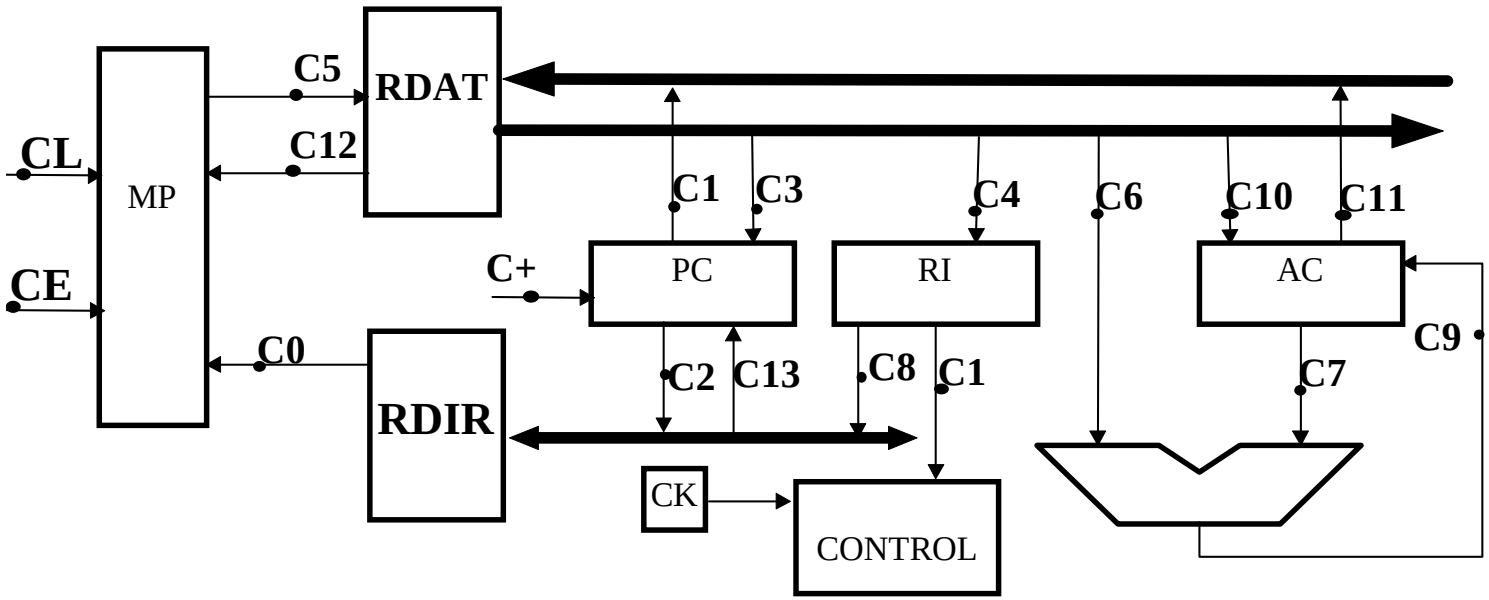
INTRODUCCION

- La parte de la unidad central de proceso, que controla
  - Los movimientos de datos en el interior de la CPU
  - Los movimientos de datos entre la memoria y la CPU
  - El intercambio de datos y de señales de control con los Interfaces
  - las operaciones de la ALU

- En este tema se estudia como se definen las señales de control que en cada instante de tiempo debe generar la UC para que el computador funcione

- Para ello hay que estudiar:
  - El camino de datos de la CPU
  - Las operaciones que debe ejecutar la CPU en cada ciclo
  - Las señales de control necesarias para realizar cada operación

CAMINO DE DATOS

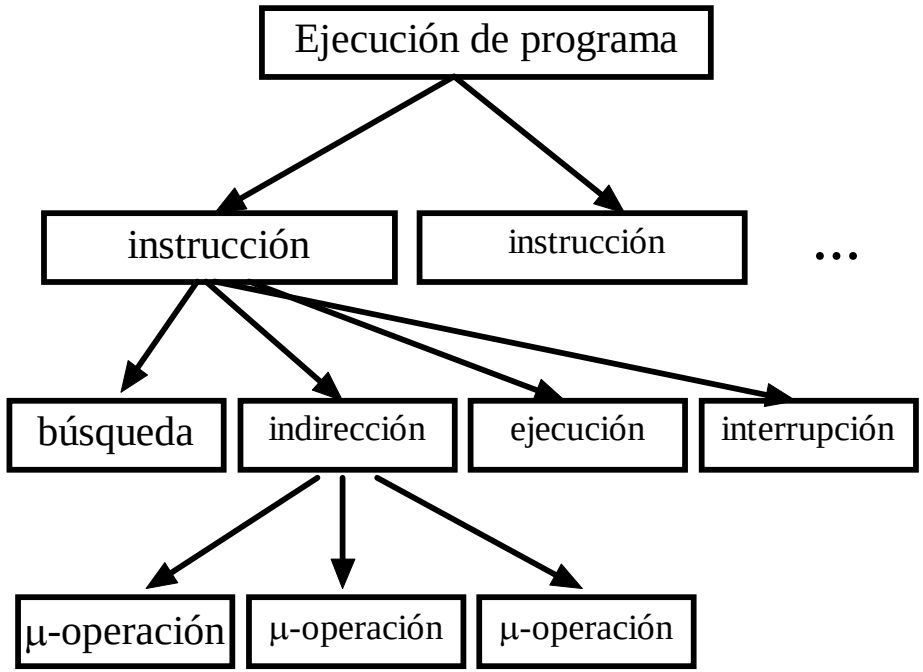


LAS MICROOPERACIONES

- **Un instrucción se divide en fases:**
  - **Búsqueda (Fetch)**
  - **Cálculo de las dir de los operandos**
  - **Ejecución de la operación**
  - **Cálculo de la dir de los resultados**
  - **Interrupción**

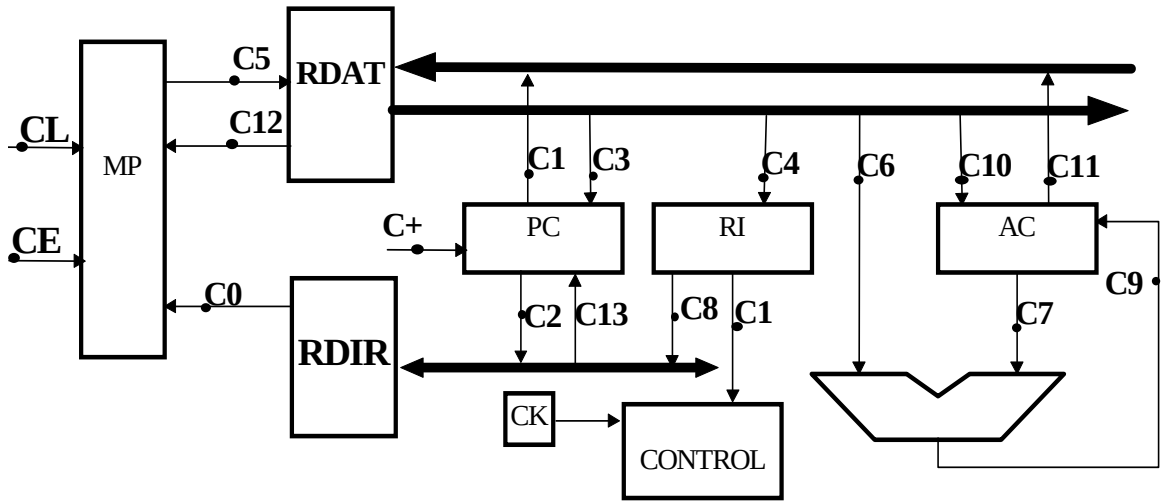
- **Una fase se compone de microoperaciones**
  - **Son las operaciones atómicas de la UCP**
  - **movimientos elementales de información**

- **Estudiaremos las microoperaciones de las fases de**
  - **Búsqueda**
  - **Ejecución**
  - **Indirección**
  - **Interrupción**



FASE DE BÚSQUEDA

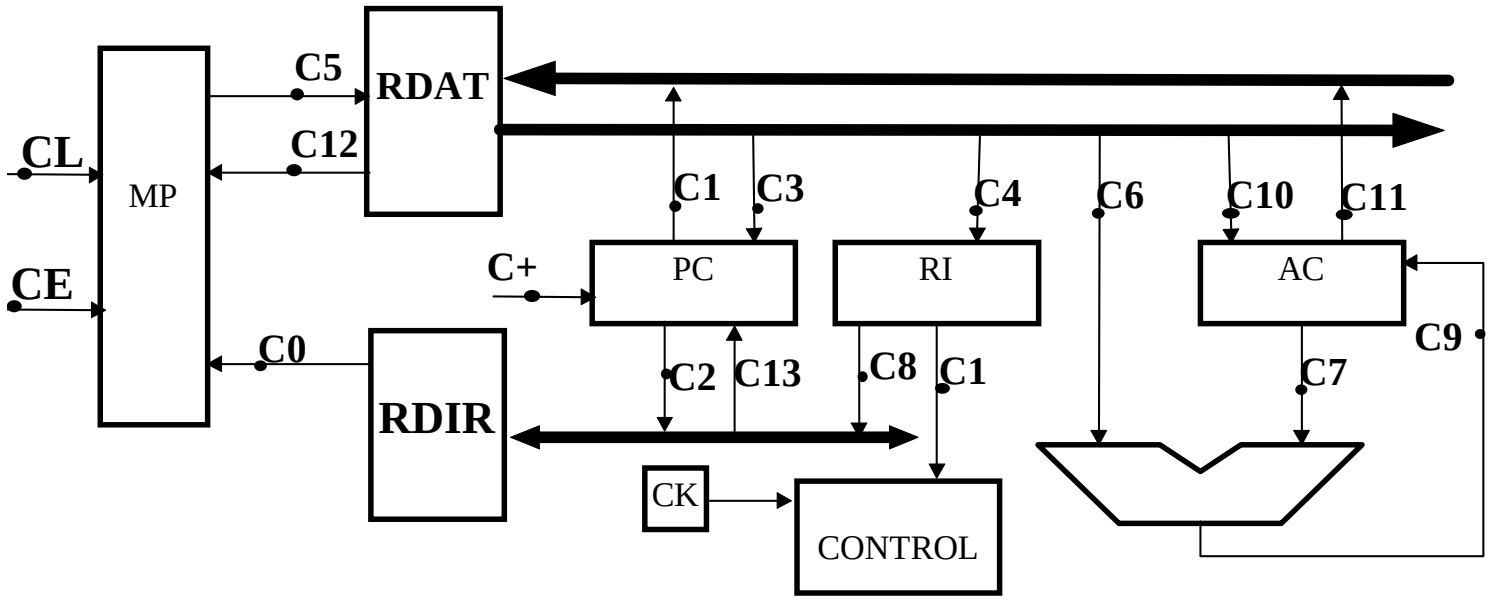
- **Operaciones:**
  - Cargar el RI con una I leída de la MP
  - Actualizar el PC
- **movimientos elementales:**
  - T1: RDIR := <Pc>
  - T2:
    - » RDAT := <MEMORIA[RDIR]>
    - » PC :=<PC> + 1
  - T3: RI:=<RDAT>
- **La fase de búsqueda consta de**
  - 3 ciclos de reloj
  - 4 microoperaciones



- **Cada microoperación produce movimiento de datos a, o, desde registros**
- **¿Cuando agrupar microoperaciones en un mismo ciclo?**
  - Cuando un movimiento no interfiera con otro
    - » Cuando no comparten el mismo camino
    - » Cuando no comparten la misma unidad funcional

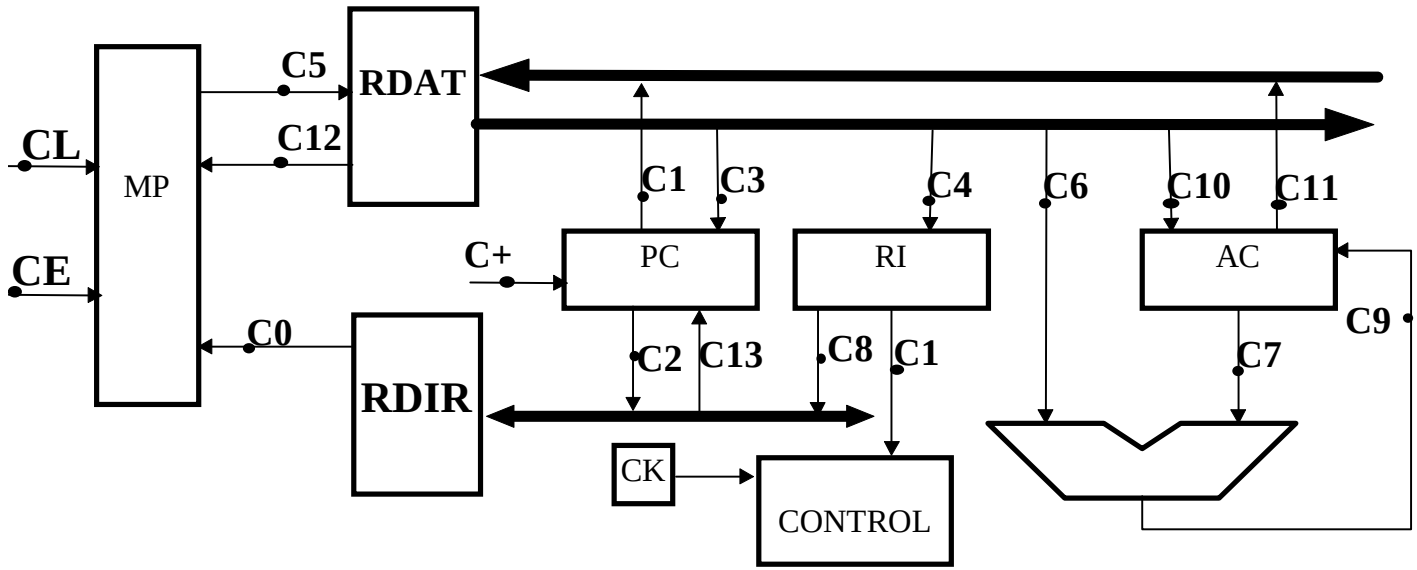
FASE DE INDIRECCIÓN

- Consiste en calcular la dirección de un operando
- Pasos:
  - T1: RDIR := <RI. DIRECCION>
  - T2: RDAT:= <MEMORIA. RDIR>
  - T3: RI. DIRECCION := <RDAT>
- Tiene tres ciclos de reloj y tres microoperaciones



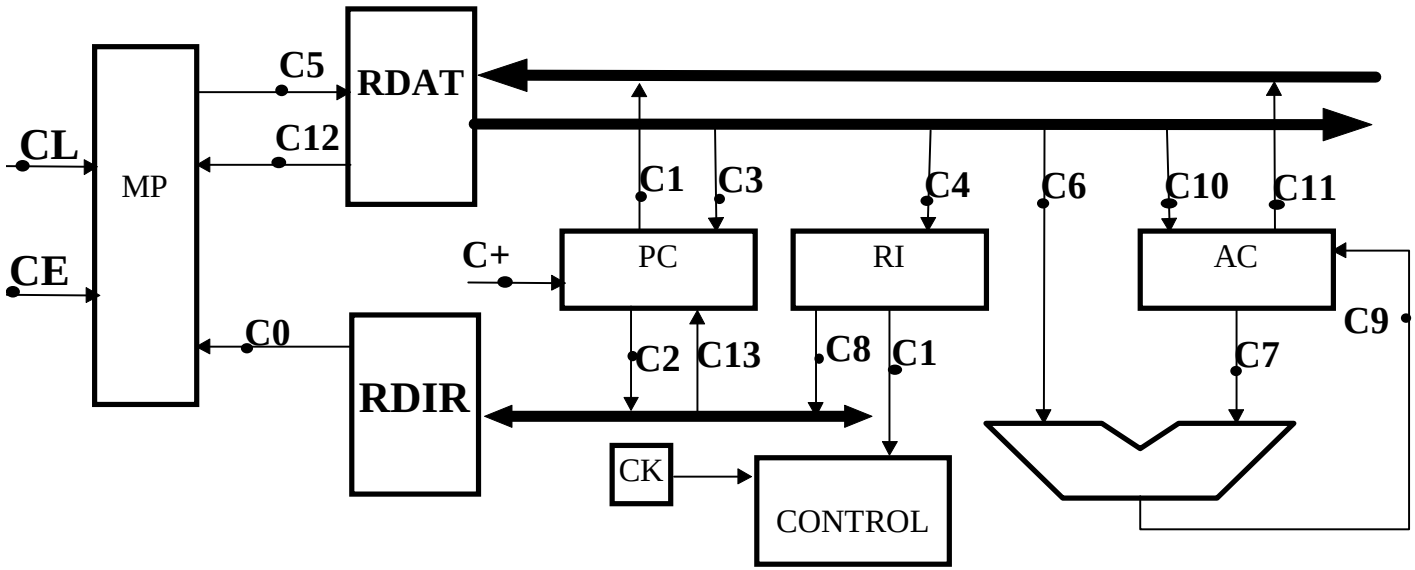
FASE DE INTERRUPCIÓN

- Se lleva a cabo al terminar la fase de ejecución
- Un ejemplo de secuencia de operaciones:
  - T1: RDAT := <Pc>
  - T2:
    - » RDIR:= Dirección de salvaguarda
    - » PC := Dirección de subrutina de tratamiento de interrupción
  - T3: MEMORIA[RDIR]:= <RDAT>
- Tres ciclos de reloj y Cuatro microoperaciones



FASE DE INTERRUPCIÓN(II)

- **T_1:**
  - » **RDAT ← <PC>**
  - » **RDIR ←Dirección de salvaguarda**
- **T_2:**
  - » **PC ← Dirección de la subrutina de interrupción**
  - » **MEMORIA[RDIR] ← <RDAT>**



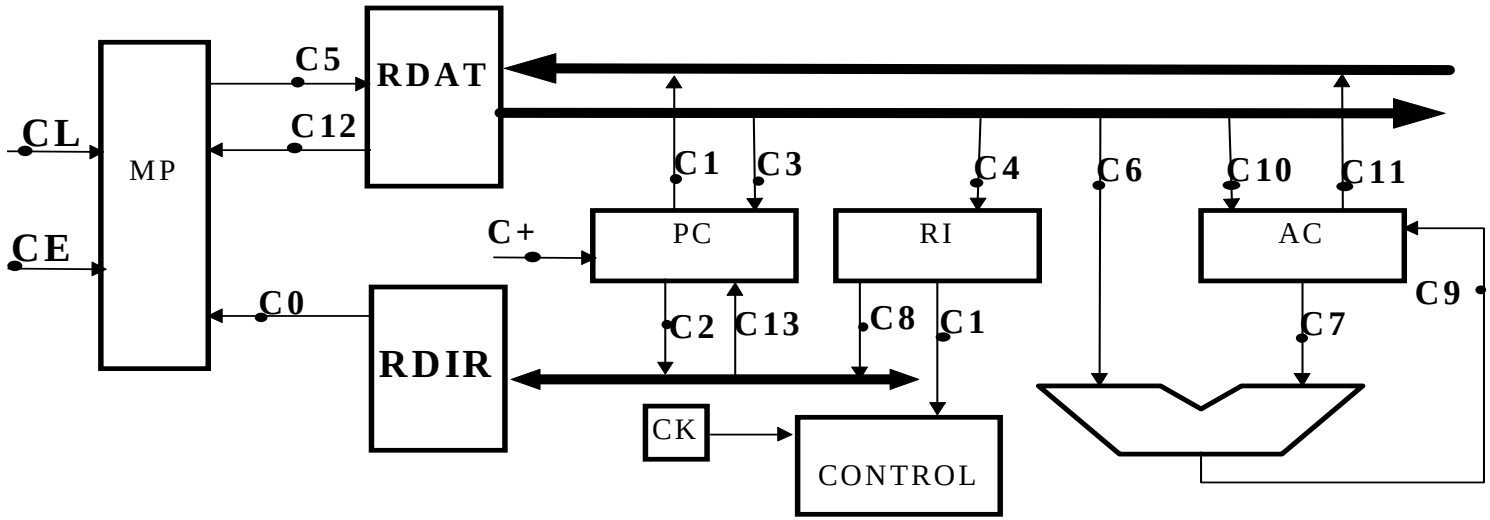
FASE DE EJECUCIÓN

- Las fases de
  - Búsqueda
  - Indirección
  - Interrupción
    - » Son :
      - Simples y predecibles
      - Cada uno implica una pequeña secuencia fija de operaciones

- Esto no es cierto para la fase de ejecución
  - Si un computador tiene N códigos de operación diferentes, existen N secuencias diferentes

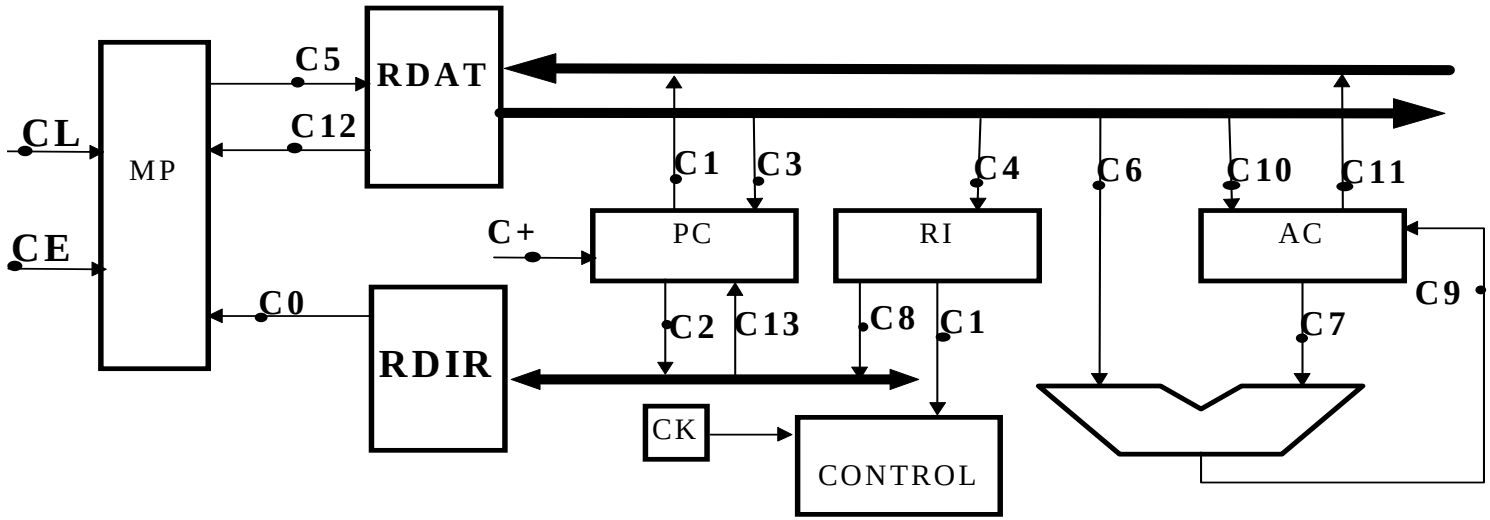
FASE DE EJECUCIÓN

- **ADD R1, X**
  - **T1: RDIR:= <RI .Dirección>**
  - **T2: RDAT:= <MEMORIA[RDIR]>**
  - **T3: R1:= <R1> + <RDAT>**



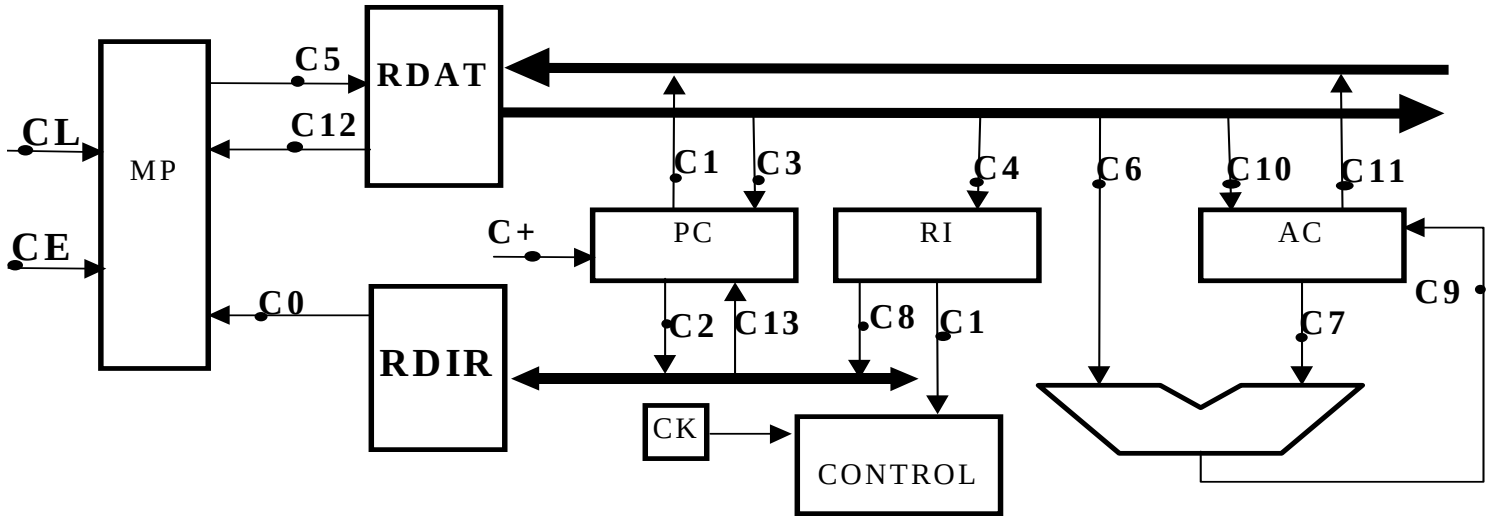
FASE DE EJECUCIÓN

- ISZ X Increment and Skip if Zero
- x:=m<X> +1 y si x=0 pc:=<PC>+1
  - T1: RDIR:= <RI.Dirección>
  - T2: RDAT := <MEMORIA[RDIR]>
  - T3: AC := <RDAT> + 1
  - T4: RDAT:=<AC>
  - T5: MEMORIA:= <RDAT>
  - If <RDAT > = 0 then Pc :=<Pc>+1



FASE DE EJECUCIÓN

- **BSA X Branch and Save address instruction**
  - **T1:**
    - » **RDIR := <RI. Dirección>.**
    - » **RDAT := <Pc>**
  - **T2:**
    - » **PC:= <RI. Dirección>**
    - » **MEMORIA := <RDAT>**
  - **T3: PC <PC> + 1**

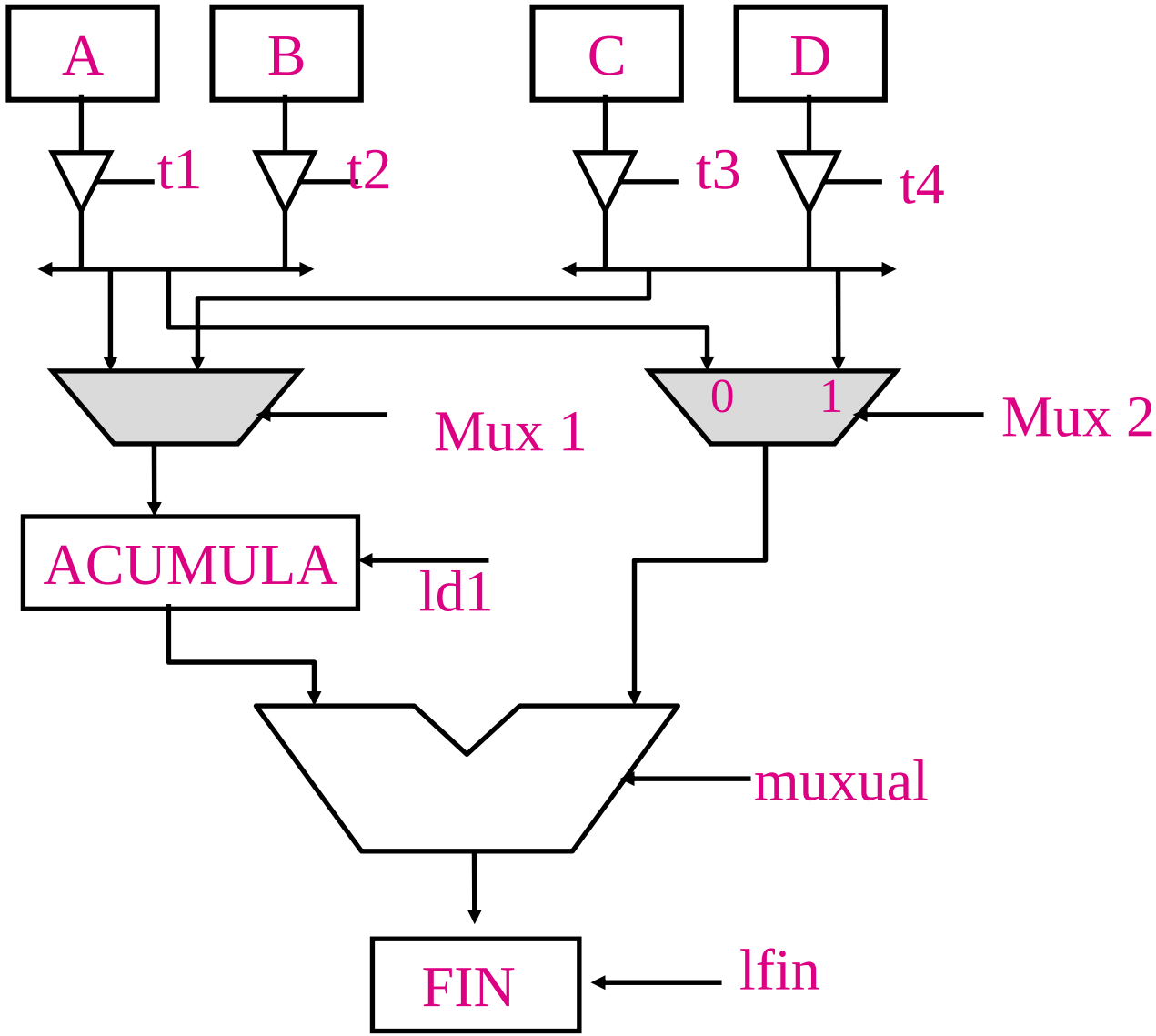


ESPECIFICACIÓN DE LAS SEÑALES DE CONTROL

- **Ejemplos de señales de control:**
  - **Lectura/escritura de memoria**
  - **Carga de registros**
  - **Control de puertas triestate**
  - **Operaciones de la ual**
  - **Selección de entradas en mux**

- **Como encontrar la señal de control estudiando las microoperaciones**
  - **Estudiar los caminos de comunicación entre los módulos**
    - » **Control de puertas triestate**
    - » **Control de multiplexores**
  - **Estudiar los registros que se deben cargar**
    - » **Capacitación de la carga de registros**
  - **Estudiar las operaciones Aritmético -Logicas a realizar**

ESPECIFICACIÓN DE LAS SEÑALES DE CONTROL



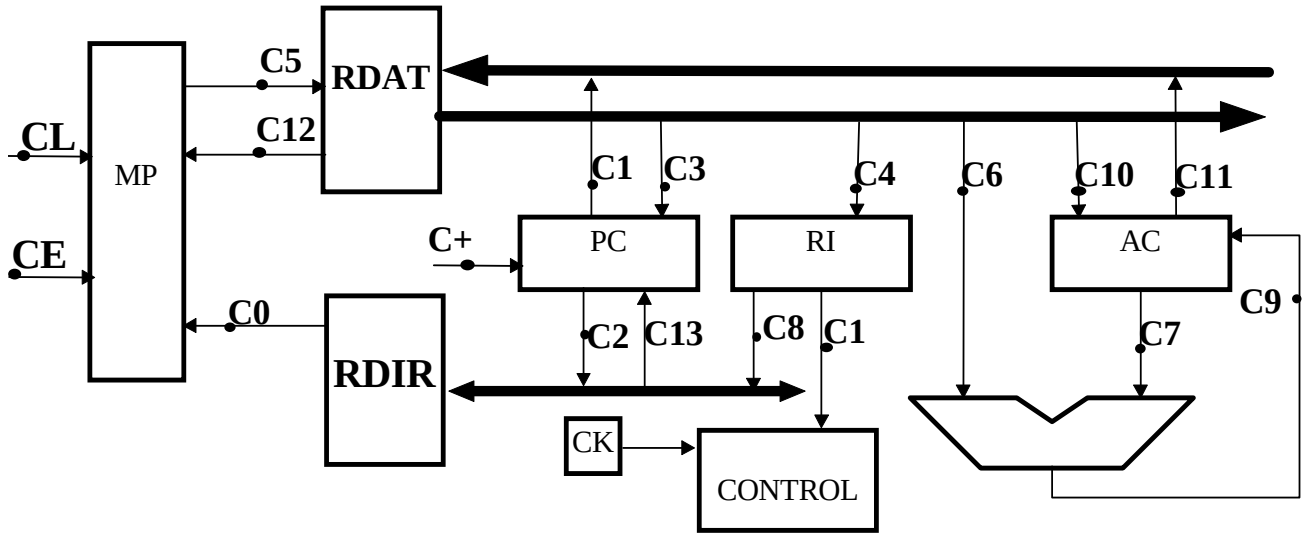
Fin<--<D>
t4 activa
mux2=1
muxual=nop
lfin activa

ESPECIFICACIÓN DE LAS SEÑALES DE CONTROL

- **Búsqueda:**
  - T1: RDIR := <PC> C2
  - T2:
    - » RDAT :=<MEMORIA[RDIR]> C5, CL,C0
    - » PC := PC + 1 C+
  - T3: RI := <RDAT> C4
- **Indirección:**
  - T1: RDIR:= <RI. Dir> C8
  - T2: RDAT := <MEMORIA[RDIR]> C5, CL,C0
  - T3:RI.DIRECCIÓN:=<RDAT> C4

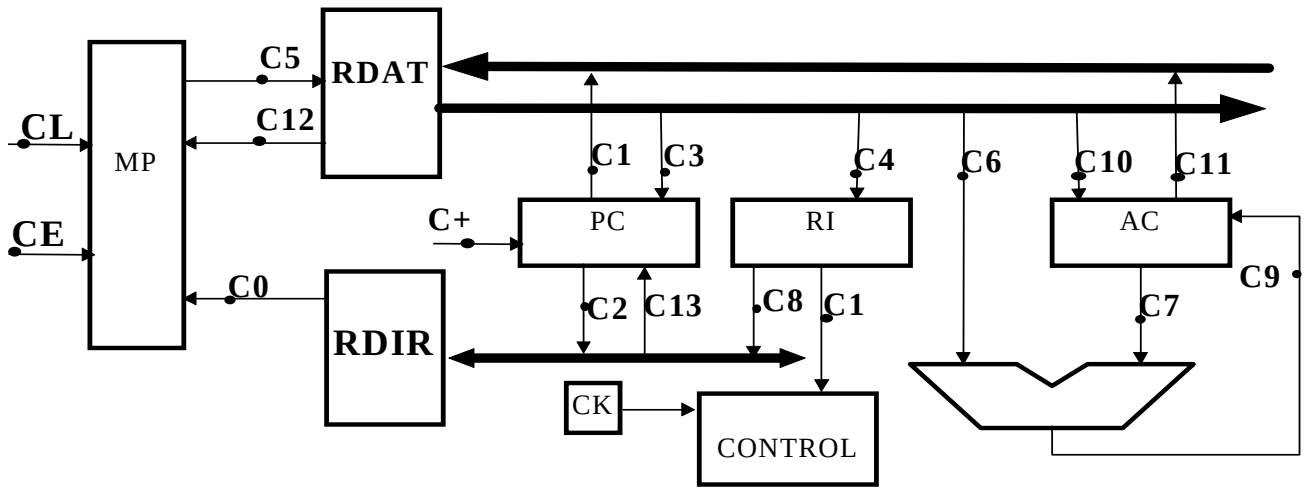
Interrupción:

- T1: RDAT := <PC>
  C1
- T2:
  - »RDIR:=dir salva
  - »PC dir interr
- T3: MEMORIA[RDIR]:= <RDAT> C12, CE,C0

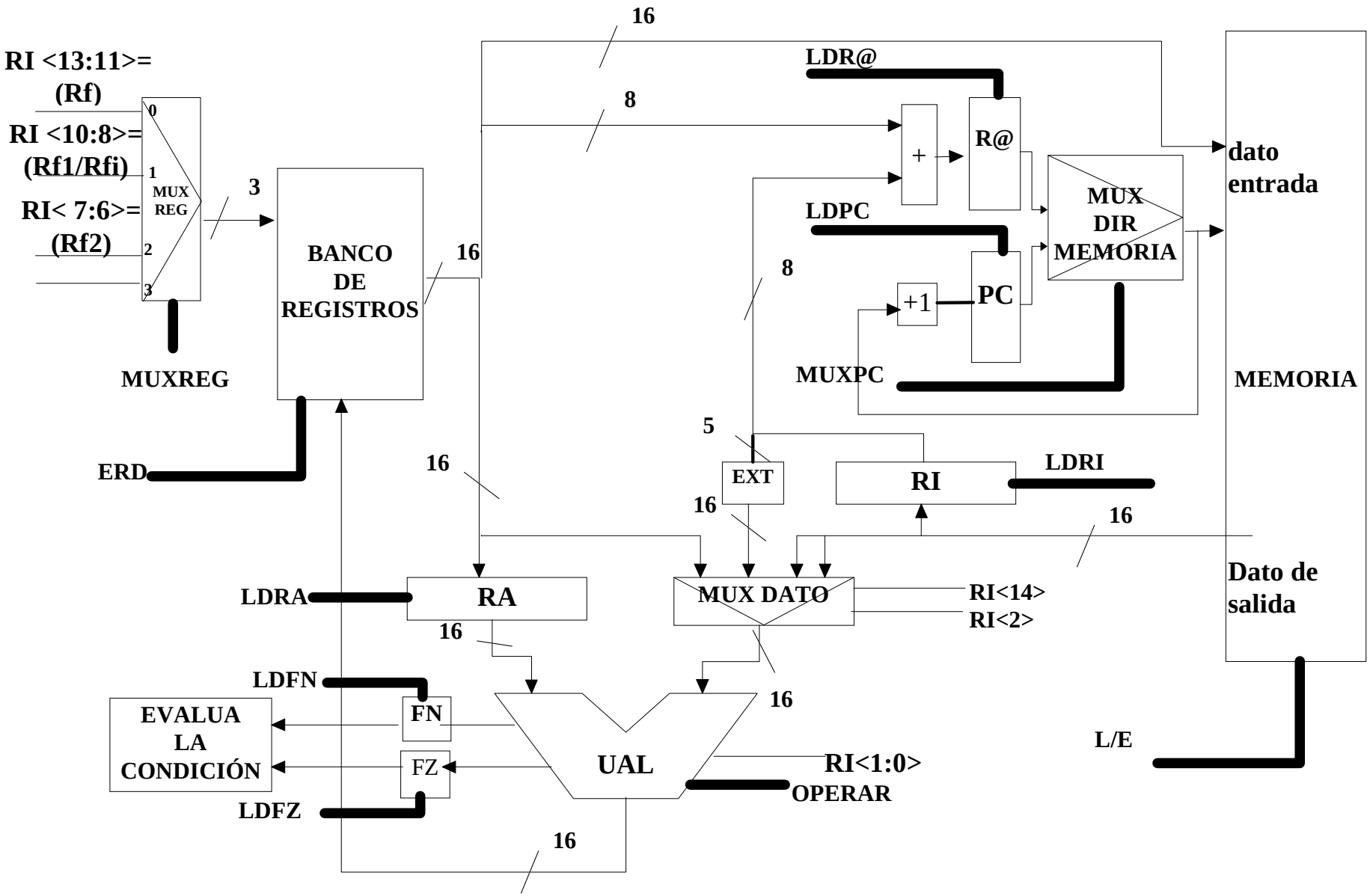


- **ADD AC,X**
  - T1: RDIR:= <RI .dirección> C8
  - T2: RDAT:= <MEMORIA[RDIR]> CL, C5
  - T3: AC :=<AC> + <RDAT> C9,C6,C7
- **ISZ X**
  - T1: RDIR := <RI.dirección) C8
  - T2: RDAT:= <MEMORIA[RDIR]> CL, C5,C0
  - T3: AC:= <RDAT> + 1 C6,C9
  - T4: RDAT:= <AC > C11
  - T5: MEMORIA[RDIR]:= <RDAT> CE ,C0,C12
  - if <RDAT = 0> then PC <PC>+1 C+

- **BSA X**
  - T1:
    - »RDIR := <RI. Dirección> C8
    - »RDAT := <PC> C1
  - T2:
    - »PC :=<RI. Dirección> C13
    - »MEMORIA[RDIR] <RDAT> C0, C12, CE
  - T3: PC:= <PC> + 1 C+



UNIDAD DE CONTROL



SEÑALES DE CONTROL

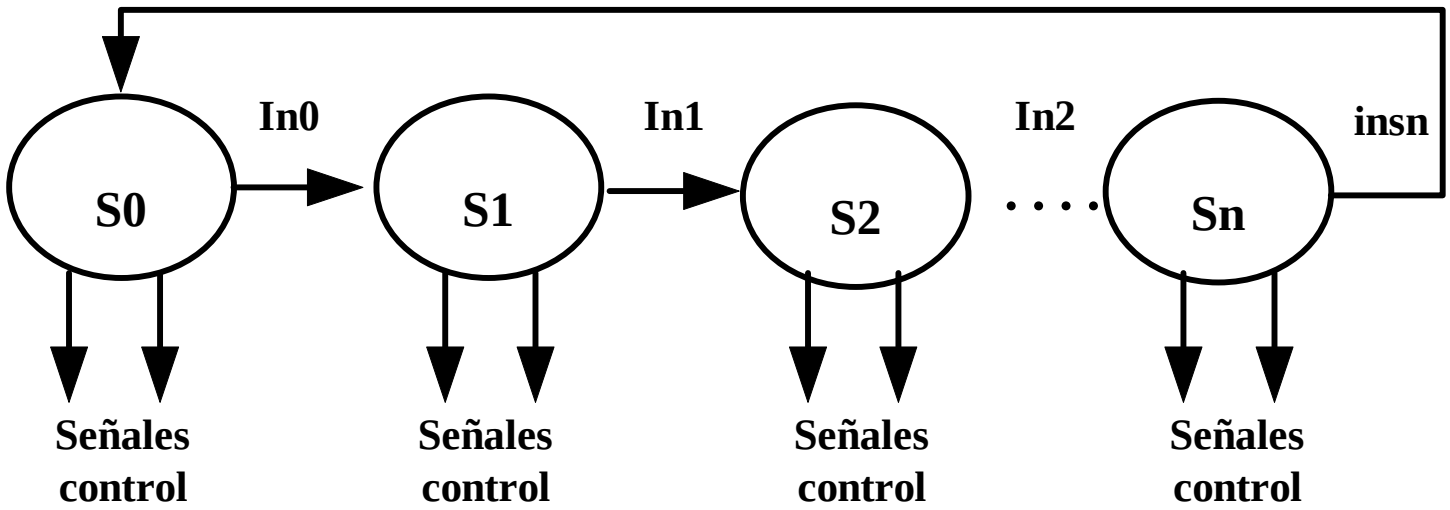
- **LDRA capacitación del registro acumulador**
- **LDRI capacitación del Registro de Instrucciones**
- **LDPC capacitación del contador de programa**
- **LDR@ capacitación del registro auxiliar de direcciones**
- **LDFZ capacitación del flag cero**
- **LDFN capacitación del flag de signo**
- **ERD capacitación de escritura en el banco de registros**
- **L/E señal de escritura lectura de la memoria**
- **MUXPC selecciona la dirección de memoria**
- **MUXRG<1:0> selecciona la dirección del banco de registros**
- **OPERAR selecciona el tipo de operación de la ALU**

Unidad de Control como S.Secuencial

- **Unidad de control es un sistema secuencial**
  - **Qué señales se activan**
  - **Cuando se activan**
- **Para especificar un sistema secuencial se necesita definir:**
  - **Conjunto de señales de entrada al sistema**
    - **Reloj**
    - **RI**
    - **Flags**
    - **Interrupciones**
  - **Conjunto de señales de salida**
  - **Conjunto de estados del sistema**
  - **Indicar como se relacionan estos tres conjuntos**
- **Esto último mediante un diagrama de estados**
  - **Maquina Mealy**
  - **Máquina de Moore**

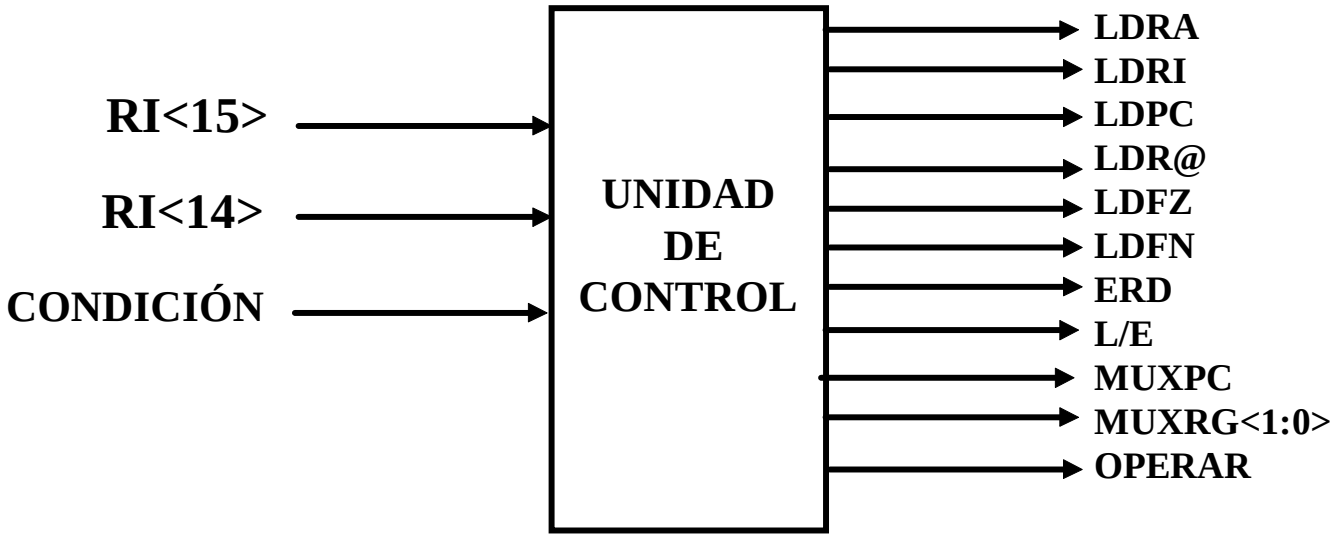
Unidad de Control como S.Secuencial

- **Para interpretar los grafos de estado debemos saber que:**
  - **Los círculos representan los estados**
  - **Las flechas entre circulos indican las relaciones entre estados**
  - **Los códigos sobre las flechas indican la información de entrada a la unidad de control**



La unidad de control de la máquina rudimetaria

- De nuestra unidad de control conocemos
  - las señales de entrada
  - las señales de salida
- Falta por definir:
  - Los estados
  - Como se relacionan
  - Que señales se activan en cada estado



Fases

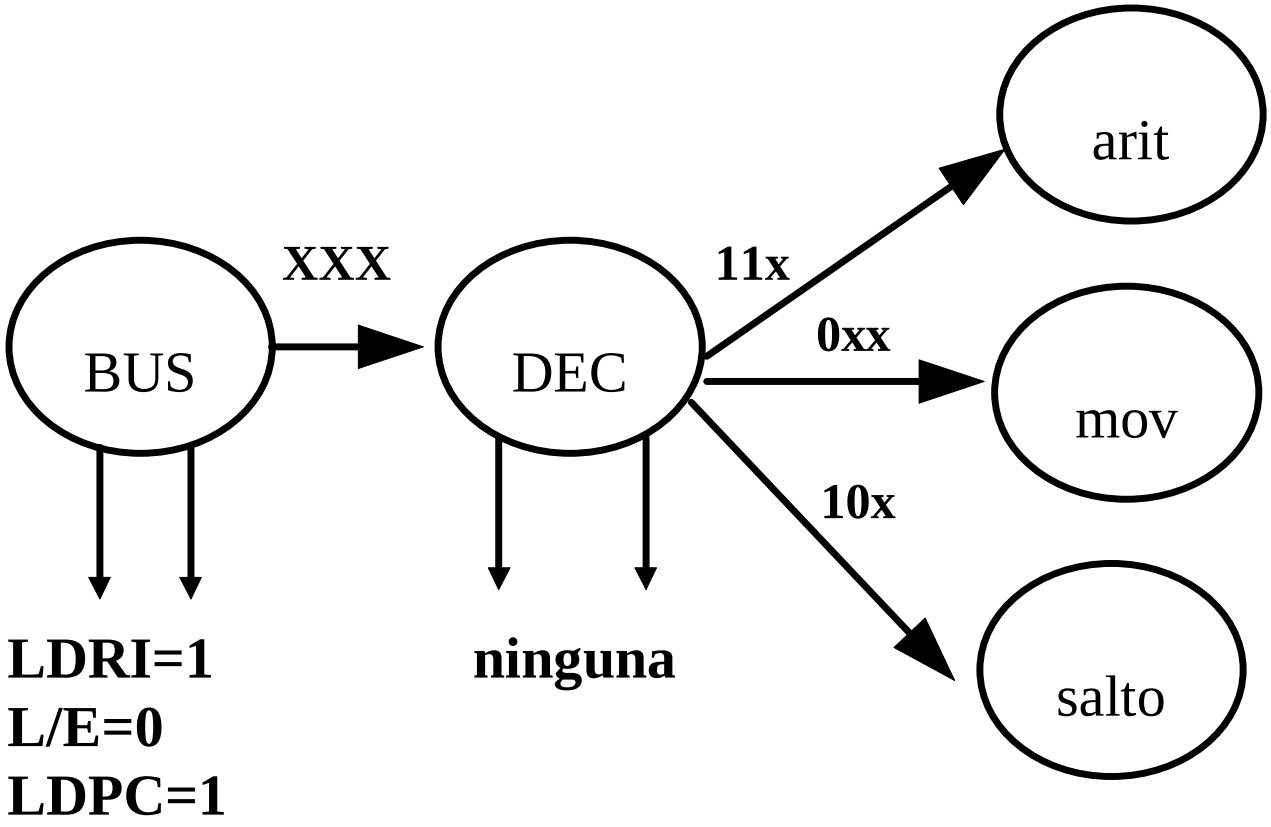
- **Fase de búsqueda de la instrucción**
- **Fase de decodificación**
- **Fases de busqueda del priemr operando o evaluación de condiciones**
- **Fase de ejecución**

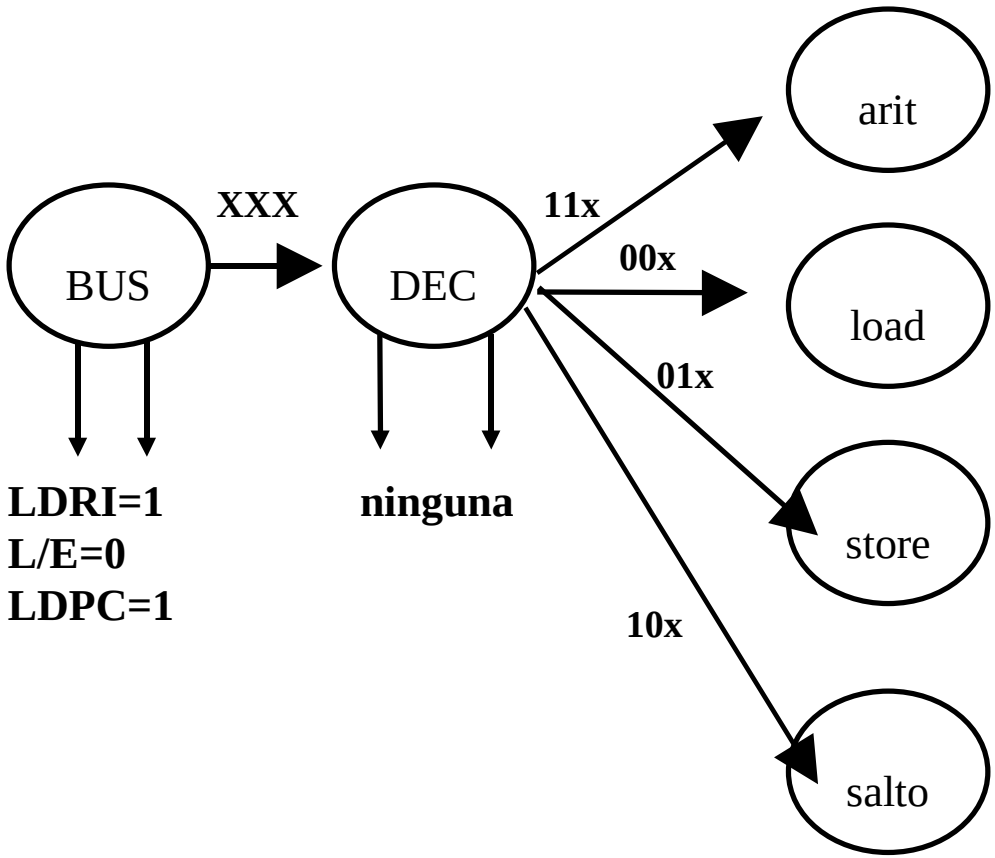
FASES COMUNES

- **Fase de búsqueda**
  - **RI<--m[PC] LDRI=1, L/E=0**
  - **PC<--<PC>+1 LDPC=1**
  - **<u>ESTADO = BUS</u>**

- **Fase de decodificacion**
  - **00 instrucciones LOAD**
  - **01 instrucciones STORE**
  - **10 instrucciones de SALTO**
  - **11 instrucciones aritmético - lógicas**
  - **<u>ESTADO DEC</u>**

BUSQUEDA Y DECODIFICACIÓN




BUS
XXX
DEC
11x
00x
01x
10x
arit
load
store
salto
LDRI=1
L/E=0
LDPC=1
ninguna

instrucciones aritmético-lógicas

- **ADD Rf1, Rf2, Rd**
  - Ra <--Rf1
  - Rd<--Ra+Rf2
- **SUB Rf1, Rf2,Rd**
  - Ra<--Rf1
  - Rd<--Ra-Rf2
- **ASR Rf,Rd**
  - Ra<--Rf1
  - Rd<--deplaza(Rf2)
- **AND Rf1, Rf2,Rd**
  - Ra<--Rf1
  - Rd<--Ra and Rf2
- **ADDI Rf, #n,Rd**
  - Ra<--Rf1
  - Rd<--Ra + exte(#n)
- **SUBI Rf,#n,Rd**
  - Ra<--Rf1
  - Rd<--Ra - exte(#

FASES DE LAS ARITMETICO LOGICAS

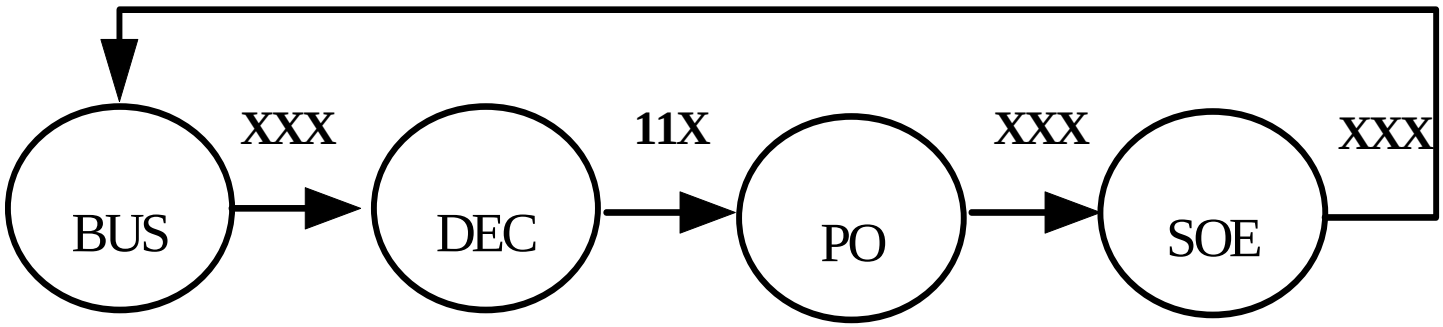
- **Fase de busqueda del primer operando**
  - **RA<-- RF1** ***LDRA=1 MUXREG=1***
  - **ESTADO PO**

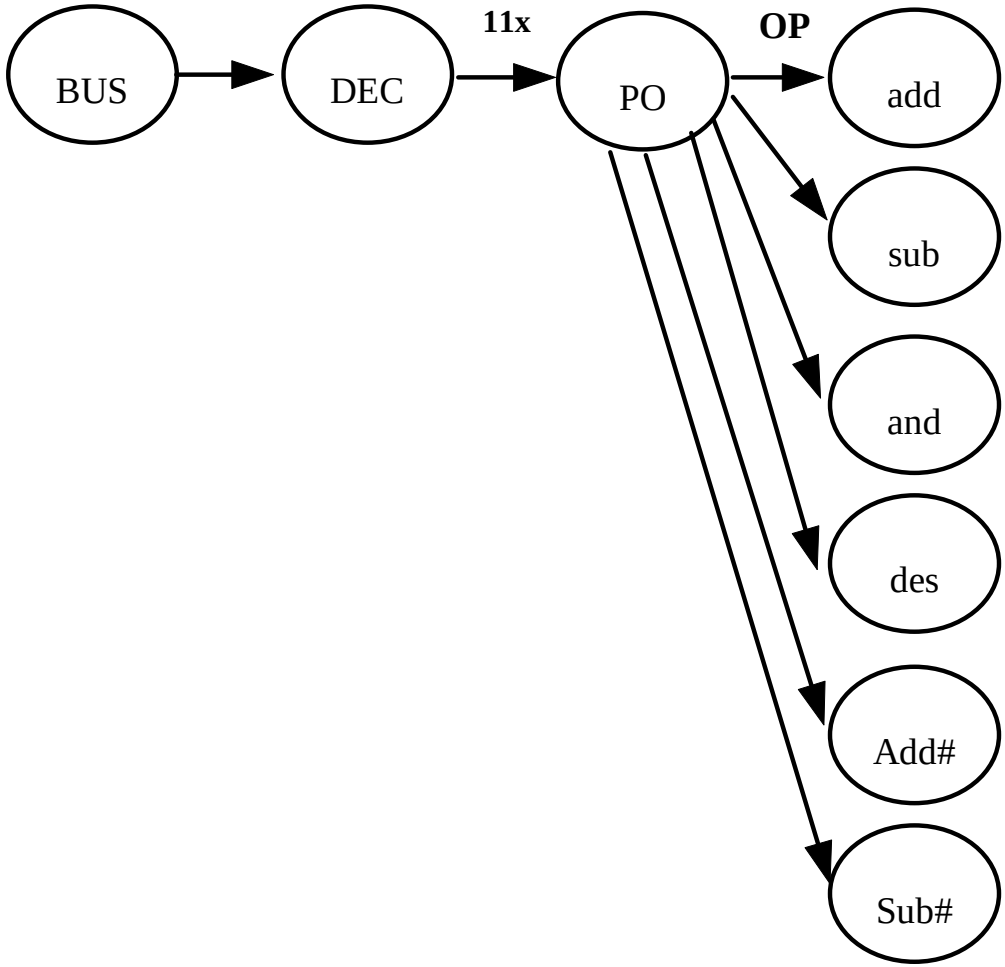
- **Fase de busqueda segundo operando y ejecucion**
  - **dos operandos en el banco**
    - **RD <-- RA OP RF2** ***ERD=1 OPERAR=1, MUXREG=2***
    - **FZ, FN** ***LDFN=1, LDFZ=1***
    - **4 estados posibles , debido a decisiones de diseño Estado SOE1**
  - **ESTADO SOE**

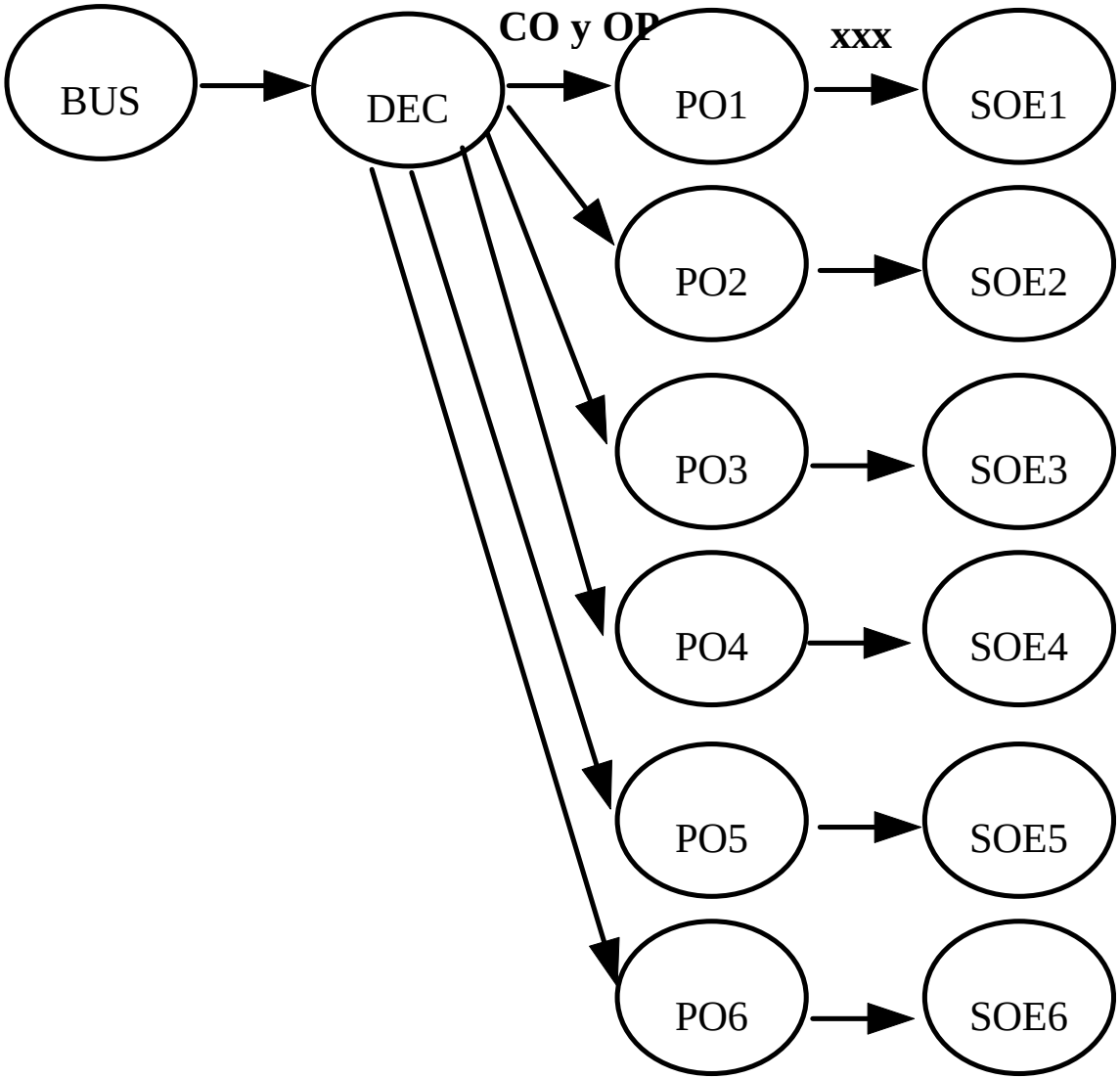
DIAGRAMA DE ESTADOS DE LA ARIMETICO-LOGICAS

- **Los cuatro estados en los que se divide la ejecución de una instrucción aritmético lógica son:**
  - **Búsqueda de la instrucción (BUS)**
  - **Descodificación (DEC)**
  - **Búsqueda del primero operando (PO)**
  - **Búsqueda del segundo operando y ejecución (SOE)**

- **Diagrama de estados:**




BUS
DEC
11x
PO
OP
add
sub
and
des
Add#
Sub#


BUS
DEC
CO y OP
PO1
xxx
SOE1
PO2
SOE2
PO3
SOE3
PO4
SOE4
PO5
SOE5
PO6
SOE6

SEÑALES DE CONTROL DE LAS ARITMETICO LOGICAS

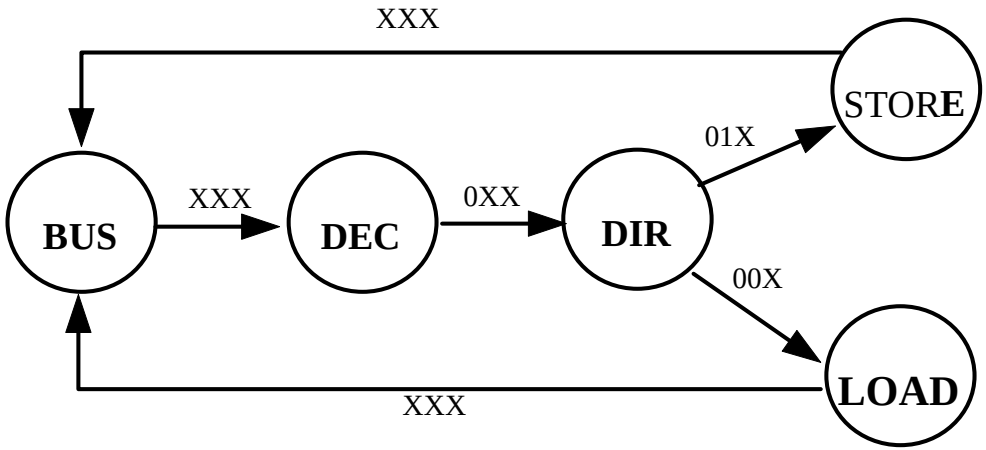
señales de control	BUS	DEC	PO	SOE
LDRA	0	0	**1**	0
LDRI	**1**	0	0	0
LDPC	**1**	0	0	0
LDR@	0	0	0	0
LDFZ	0	0	0	**1**
LDFN	0	0	0	**1**
ERD	0	0	0	**1**
L/E	0	0	0	0
MUXPC	0	X	X	X
MUXREG<1:0>	X	X	**1**	**2**
OPERAR	X	X	X	**1**

FASES DE LAS INSTRUCCIONES DE MOVIMIENTO

- **Fase de Cálculo de la dirección de memoria:**
  - **R@ <-- Ri<7:0>+RI<7:0>** ***LDR@ =1,MUXREG=1***
  - **ESTADO=DIR**

- **Fases de acceso a memoria**
  - **STORE (fase de Escritura en memoria)**
    - **M[R@]<--RF** ***L/E=1, MUXREG=0, MUXPC=1***
    - **ESTADO =STORE**
  - **LOAD (Fase de Leer la memoria y cargar el contenido en el banco de registros)**
    - **RD<-- M[R@]** ***L/E=0, ERD=1,MUXPC=1,OPERAR=0***
    - **FNFZ** ***LDFN=1,LDFZ=1***
    - **ESTADO= LOAD**

DIAGRAMA Y SEÑALES DE CONTROL



señales de control	dir	store	load
LDRA	0	0	0
LDRI	0	0	0
LDPC	0	0	0
LDR@	**1**	0	0
LDFZ	0	0	**1**
LDFN	0	0	**1**
ERD	0	0	**1**
L/E	0	1	**0**
MUXPC	X	**1**	**1**
MUXRG<1:0>	**1**	**0**	X
OPERAR	X	X	**0**

FASES DE LAS INSTRUCCIONES DE SALTO

- **Fase de evaluación de la condición de salto**

  Como en todas las fases de decisión no se genera ninguna señal de control

  - **ESTADO = EVAL**

- **Fase calculo dir salto (dir2)**
  - **R@ <-- RI<7:0> +R_0 LDRA=1, MUXREG=1**
  - **ESTADO = DIR2**

- **Fase de salto (SAL)**
  - **PC<--R@+1 LDPC=1, MUXPC=1**
  - **RI<--M[R@] LDRI=1, L/E=0,**
  - **ESTADO = SAL**

Unidad Direccionadora del Banco de Registros

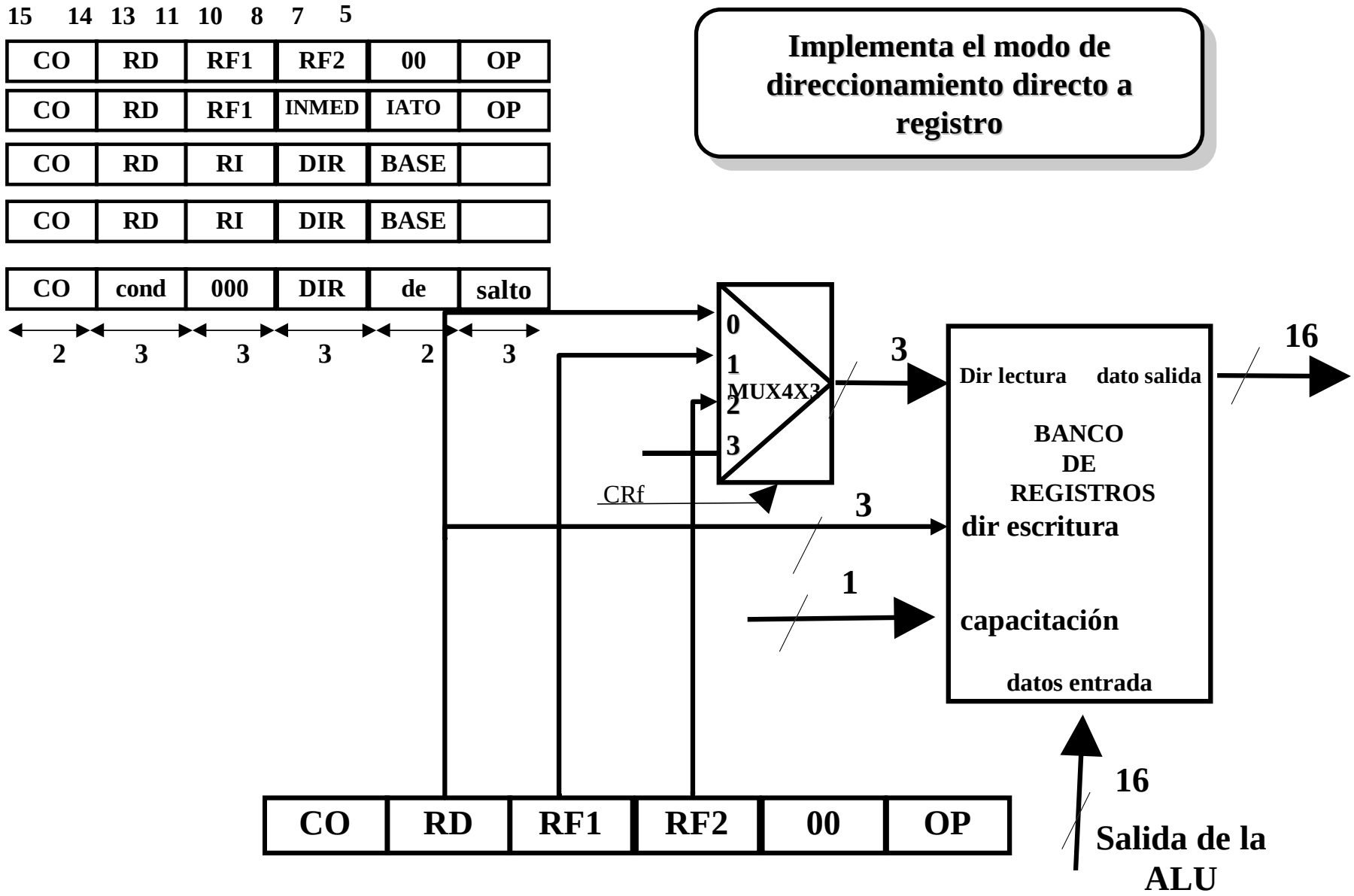
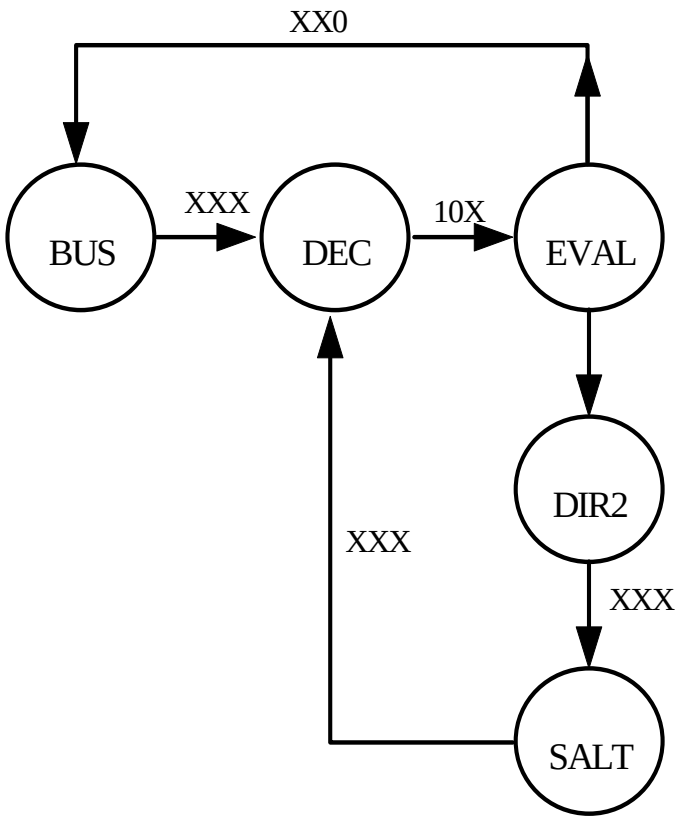
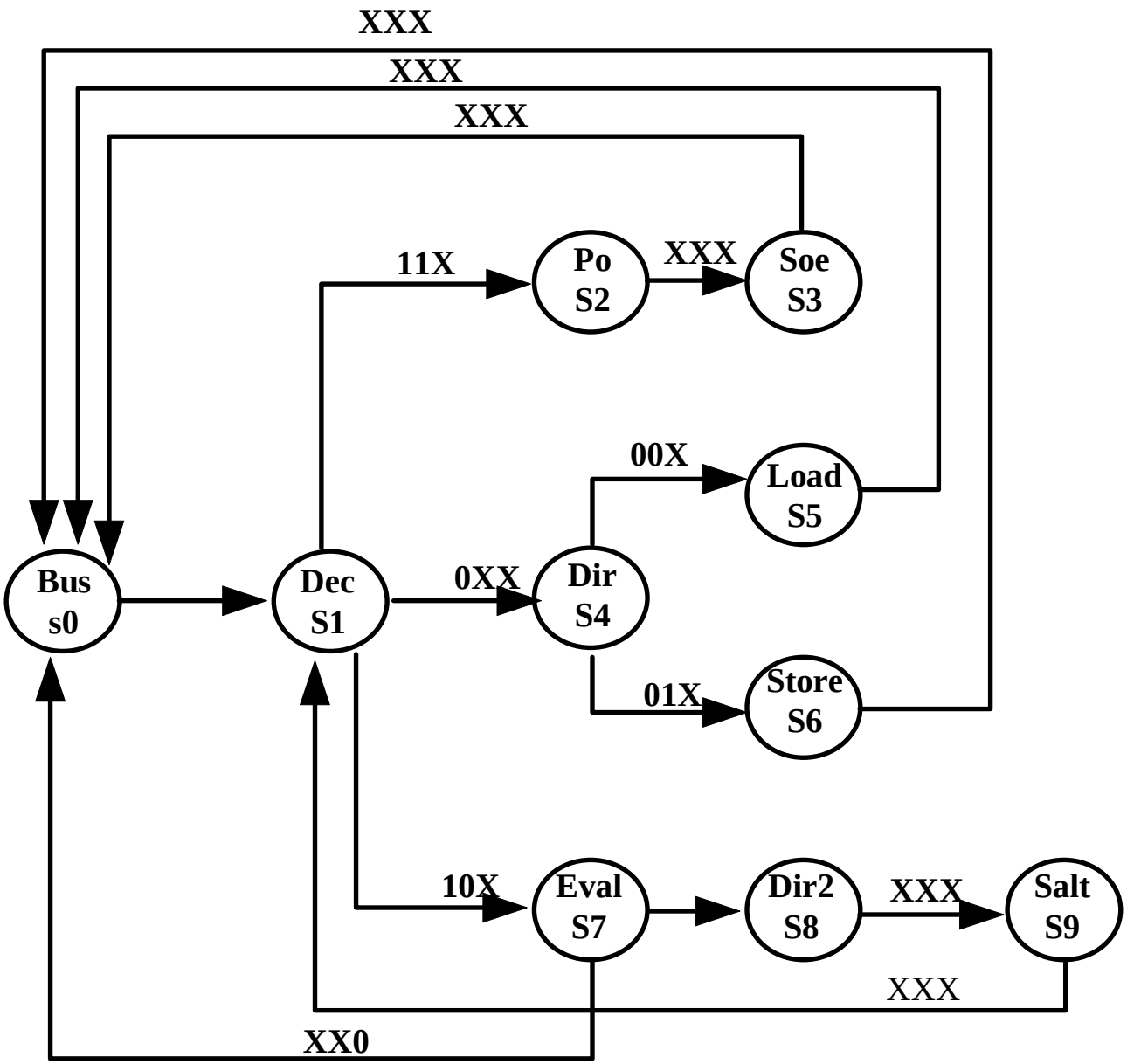


DIAGRAMA Y SEÑALES DE CONTROL



señales de control	eval	dir2	salt
LDRA	0	0	0
LDRI	0	0	**1**
LDPC	0	0	**1**
LDR@	0	**1**	0
LDFZ	0	0	0
LDFN	0	0	0
ERD	0	0	0
L/E	0	0	**0**
MUXPC	X	X	**1**
MUXRG<1:0>	0	**1**	X
OPERAR	X	X	x

DIAGRAMA DE ESTADOS LA UNIDAD DE CONTROL



SEÑALES QUE GENERA LA UNIDAD DE CONTROL

salidas	bus	dec	dir	load	store	po	soe	eval	dir2	salt
LDRA	0	0	0	0	0	1	0	0	0	0
LDRI	1	0	0	0	0	0	0	0	0	1
LDPC	1	0	0	0	0	0	0	0	0	1
LDR@	0	0	1	0	0	0	0	0	1	0
LDFZ	0	0	0	1	0	0	1	0	0	0
LDFN	0	0	0	1	0	0	1	0	0	0
ERD	0	0	0	1	0	0	1	0	0	0
L/E	0	0	0	0	1	0	0	0	0	0
MUXPC	0	X	X	1	1	X	X	X	x	1
MUXRG<1:0>	X	X	1	X	0	1	2	X	1	X
OPERAR	X	X	X	0	X	X	1	X	x	X

IMPLEMENTACIÓN DE LA UNIDAD DE CONTROL

TEMA 8

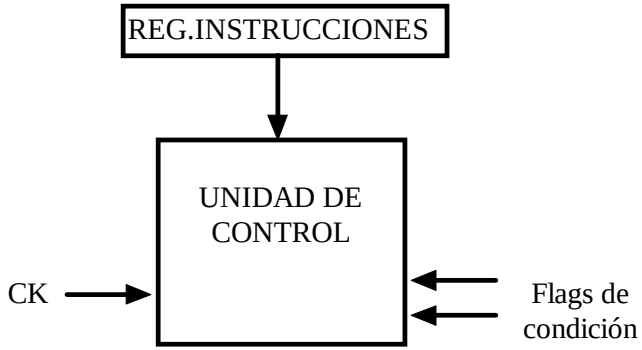
IMPLEMENTACIÓN DE LA UNIDAD DE CONTROL

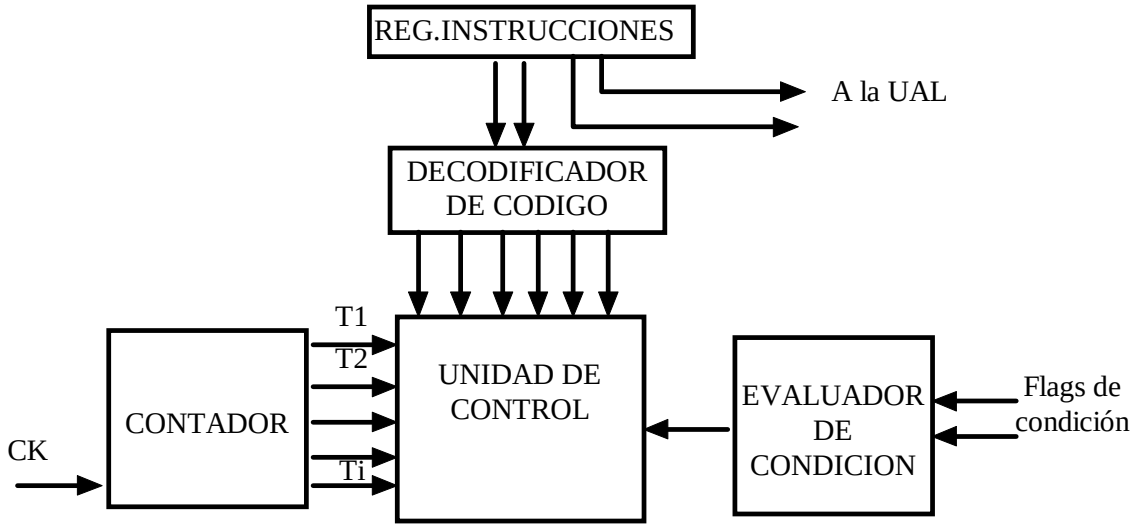
- **Existen dos formas de implementar una unidad de control**
  - **Cableada**
  - **Microprogramada**
- **En la cableada se**
  - **Utilizan circuitos combinacionales para generar las señales de control**
  - **Biestables para almacenar los estados**
- **En la microprogramada**
  - **Se utiliza una memoria almacenar los microprogramas que implementan las instrucciones**

UNIDAD DE CONTROL CABLEADA

- Es un circuito secuencial clásico
- Las señales de entrada se transforman en un conjunto de señales de salida que son las señales de control
- Las entradas de la Unidad de Control:
  - Registro de instrucciones
  - Flags de condición
  - Ck
- Estas entradas se pueden procesar antes de su entrada a la Unidad de control de manera que simplifiquen su diseño
  - Se utiliza un decodificador entre el RI y la Unidad de control
  - Se introduce un contador entre la señal de reloj y la Unidad de control
  - Se introduce un evaluador de condiciones entre los flags de condición y la Unidad de control
- El objetivo de estas modificaciones es sacar complejidad fuera de la unidad de control

UNIDAD DE CONTROL CABLEADA
ENTRADAS





simplificación de la Unidad de Control

- **Decodificación del registro de instrucciones**
  - **Entran a la unidad de control tantas señales como códigos de operación existen**
  - **Todas descapacitadas menos la que se está ejecutando en la actualidad**
  - **La unidad de control sabe perfectamente cual es la instrucción que se esta ejecutando**

SIMPLIFICACIÓN DE LA UNIDAD DE CONTROL

- Para una fase de la instrucción la unidad de control genera diferentes señales en diferentes instantes de tiempo Ti.
- pe: para la fase de búsqueda
  - T1: RDIR := <Pc>
  - T2:
    - » RDAT := <MEMORIA[RDIR]>
    - » PC :=<PC> + 1
  - T3: RI:=<RDAT>
- Si el CK ataca directamente a la U.C., ésta debe incluir un S.S. que indique exactamente en que instante de tiempo nos encontramos para cada fase.
- Se puede utilizar un contador exterior a la U.C. que indique el instante Ti en el que se encuentra cada instrucción.
- Hay una señal de control diferente para cada Ti
- Al finalizar el ciclo de instrucción se debe inicializar el contador

ESPECIFICACIÓN DE LAS SEÑALES DE CONTROL

- **Búsqueda:**
  - T1: RDIR := <PC> C2
  - T2:
    - » RDAT :=<MEMORIA[RDIR]> C5, CL,C0
    - » PC := PC + 1 C+
  - T3: RI := <RDAT> C4
- **Indirección:**
  - T1: RDIR:= <RI. Dir> C8
  - T2: RDAT := <MEMORIA[RDIR]> C5, CL,C0
  - T3:RI.DIRECCIÓN:=<RDAT> C4
- **ADD AC,X**
  - T1: RDIR <RI .dirección> C8
  - T2: RDAT <MEMORIA[RDIR]> CL, C5
  - T3: R1 <R1> + <RDAT> C9,C6,C7
- **ISZ X**
  - T1: RDIR <RI.dirección) C8
  - T2: RDAT <MEMORIA[RDIR]> CL, C5,C0
  - T3: AC <RDAT> + 1 C6,C9
  - T4: RDAT <AC > C11
  - T5: MEMORIA[RDIR] <RDAT> CE ,C0,C12
  - if <RDAT = 0> then PC <PC>+1 C+

Interrupción:

- T1: RDAT := <PC>
  C1
- T2:
  - »RDIR:=dir salva
  - »PC dir interr
- T3: MEMORIA[RDIR]:= <RDAT> C12, CE,C0

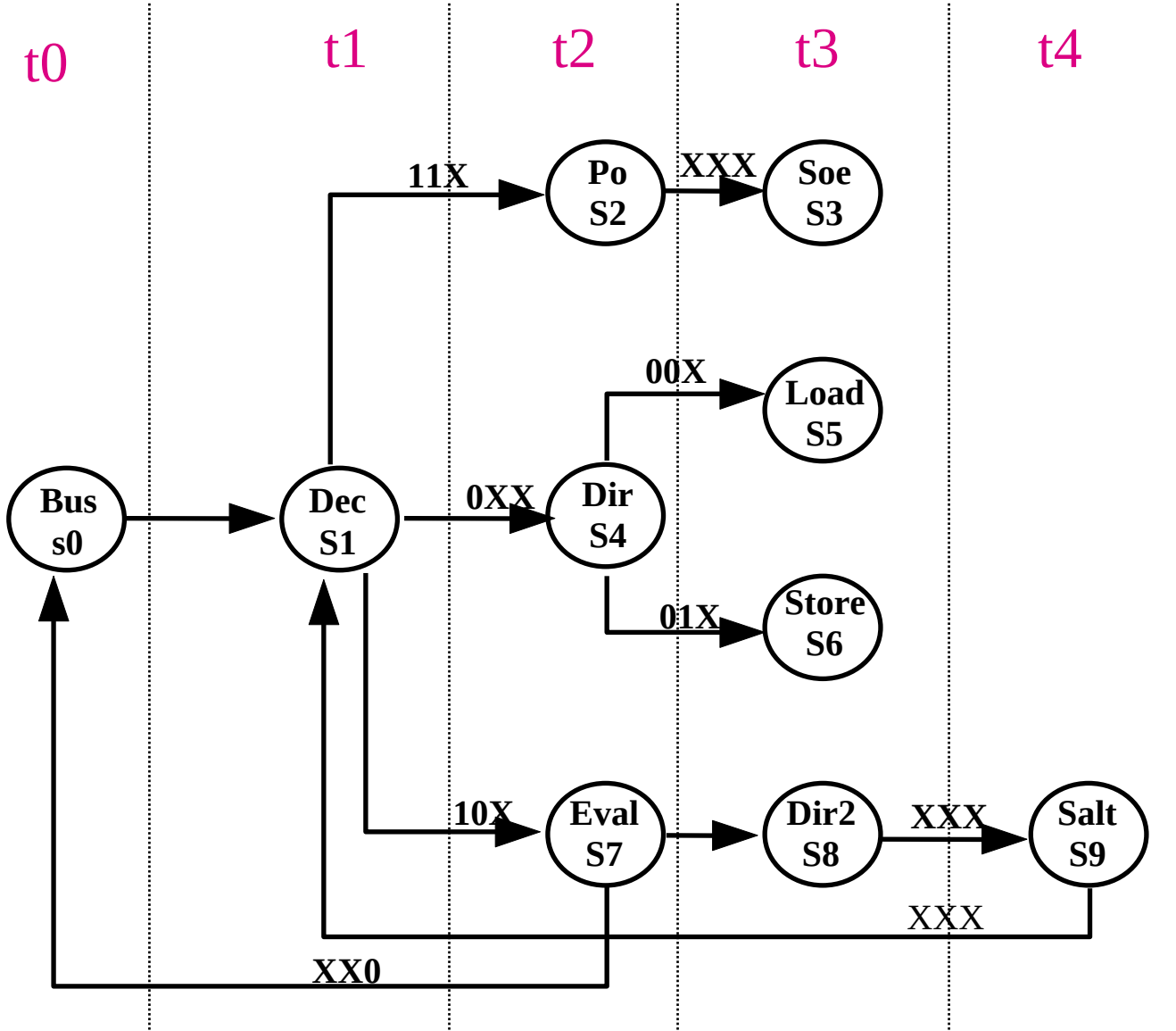
- **BSA X**
  - T1:
    - »RDIR <RI. Dirección> C8
    - »RDAT <PC> C1
  - T2:
    - »PC <RI. Dirección> C13
    - »MEMORIA[RDIR] <RDAT> C0, C12, CE
  - T3: PC <PC> + 1 C+

LÓGICA DE LA UNIDAD DE CONTROL CABLEADA

- **Suponernos un registro PQ**
  - **transparente al usuario**
  - **indica para cada T en que fase se encuentra la ejecución de la instrucción:**
    - » **00 Búsqueda**
    - » **01 Indirección**
    - » **10 Ejecución**
    - » **11 Interrupción**
- **Un contador indica el estado dentro de cada ciclo**

- **Ejemplo :**
  - $C_5 = P'\cdot Q'\cdot T_2 + P'\cdot Q\cdot T_2 + PQ'(ADD\cdot T_2 + ISZ\cdot T_2)$
  - $C_2 = P'Q'\cdot T_1$
  - $C_0 = P'Q'\cdot T_2 + P'Q\cdot T_2 + PQ'(ISZ\cdot T_2 + BSA\cdot T_2) + PQ\cdot T_3$
  - $C_4 = P'Q'\cdot T_3 + P'QT_3$
  - $C_8 = P'Q\cdot T_1 + PQ'\cdot(ADD\cdot T_1 + ISZ\cdot T_1 + BSA\cdot T_1)$

DIAGRAMA DE ESTADOS LA UNIDAD DE CONTROL



SEÑALES QUE GENERA LA UNIDAD DE CONTROL

	T0	T1	t2	t3	t3	t2	t3	t2	t3	t4
salidas	bus	dec	dir	load	store	po	soe	eval	dir2	salt
LDRA	0	0	0	0	0	1	0	0	0	0
LDRI	1	0	0	0	0	0	0	0	0	1
LDPC	1	0	0	0	0	0	0	0	0	1
LDR@	0	0	1	0	0	0	0	0	1	0
LDFZ	0	0	0	1	0	0	1	0	0	0
LDFN	0	0	0	1	0	0	1	0	0	0
ERD	0	0	0	1	0	0	1	0	0	0
L/E	0	0	0	0	1	0	0	0	0	0
MUXPC	0	X	X	1	1	X	X	X	x	1
MUXRG<1:0>	X	X	1	X	0	1	2	X	1	X
OPERAR	X	X	X	0	X	X	1	X	X	X
RESETCK	0	0	0	1	1	0	1	???	0	1

Entradas del sistema:

- **T0...T4**
- **AL**
- **ST**
- **LD**
- **BIF**
- **EVALUA**

HABRA QUE CREAR UN NUEVO ESTADO PARA RESETEAR EL CK EN ESTE CASO

$LDRA=T_2 \cdot AL$

$LDRI=T_0AL+T_0ST+T_0LD+T_0BIF+T_4BIF$

$LPC= T_0AL+T_0ST+T_0LD+T_0BIF+T_4BIF$

$LDR@=T_2ST+T_2LOAD+T_3BIF \cdot EVALUA$

$LZ=T_3LOAD+T_3AL$

$LN=T_3LOAD+T_3AL$

$ERD=T_3LD+T_3AL$

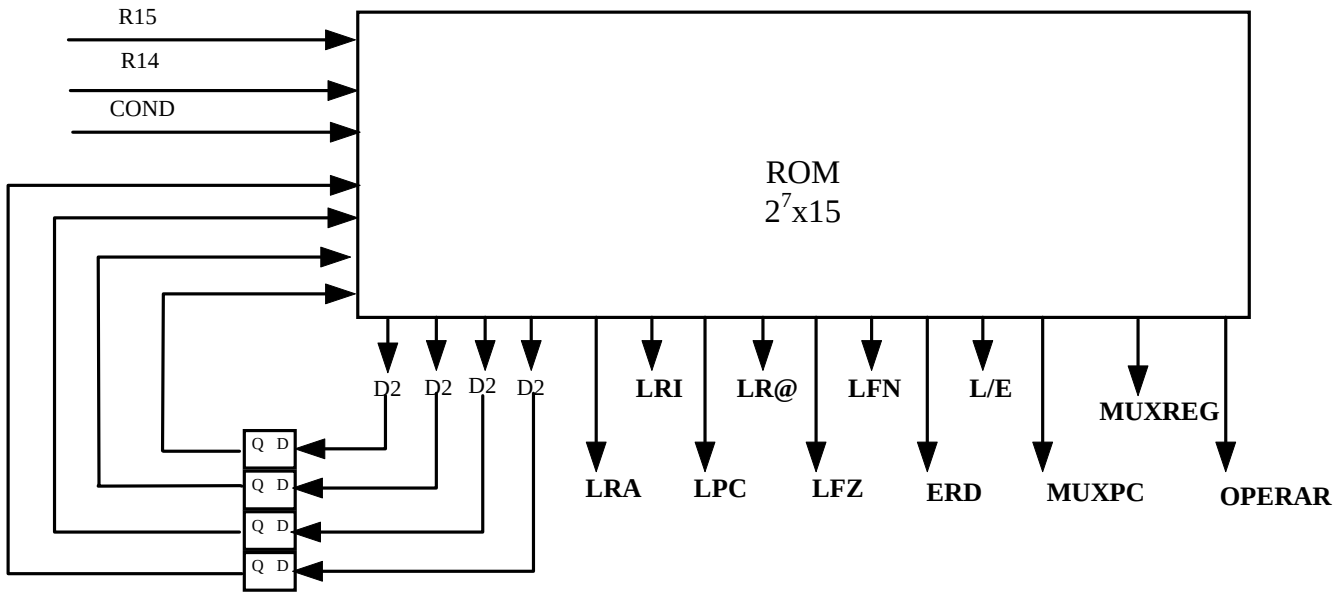
$L/E=T_3ST$

$MUXPC=T_2LD+T_2ST+T_4BIF$

$MUXREG<0>=T_2LD+T_2ST+T_2AL+T_3BIF \cdot EVALUA$

$MUXREG<1>T_3AL$

con 10 estados necesito cuatro bietsables para guardar la historia del sistema
el circuito combinacional se implementa mediante una ROM
no existe circuitería exterior



UNIDAD DE CONTROL MICROPROGRAMADA

- Una memoria de control almacena las señales de control de cada periodo de tiempo
  - La señal no se genera mediante un circuito combinacional
- A cada instrucción máquina le corresponde un microprograma
- Un microprograma se compone de microinstrucciones
- Microinstrucción:
  - Una palabra de micromemoria que define las señales de control en un periodo de la Instrucción
  - Secuencia de ceros y unos
    - » Señales activas = 1
    - » Señales inactivas = 0
- Para ejecutar un microprograma se van leyendo cada una de las instrucciones y se envían las señales leídas al computador como señales de control
  - Implica un mecanismo de secuenciamiento
- La cadencia de lectura es la del reloj básico del computador
- Firmware Conjunto de programas de una máquina

ESTRUCTURA DE UNA UC MICROPROGRAMADA

- **Breve historia**
  - La propone 1956 Wilkes
  - Se utiliza por primera vez comercialmente en la familia de computadores IBM 360
  - Inicialmente se rechazó debido a la inexistencia de Memorias rápidas y baratas

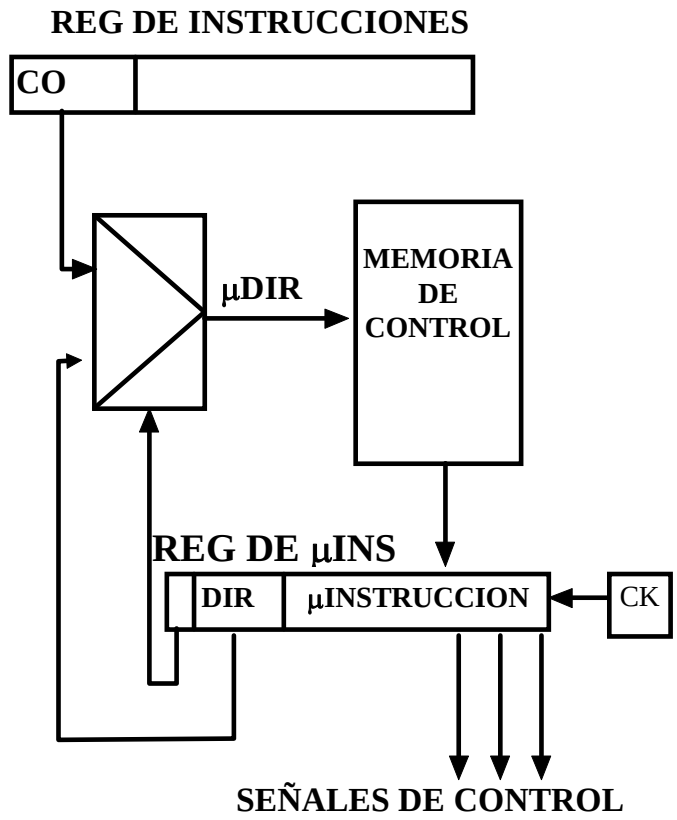
- **La U.C. microprogramada debe disponer de los siguientes mecanismos:**
  - Memoria de control con capacidad para almacenar todos los microprogramas
    - » micromemoria
  - Mecanismo que convierta el código de operación en la dirección de la micromemoria donde se almacena el programa
  - Mecanismo de secuenciamiento y bifurcación a nuevo programa
- **secuenciamiento:**
  - explícito
  - implícito
- **Elementos a tener en cuenta al estudiar el secuenciamiento:**
  - Tamaño de la memoria de control
  - Tiempo de generación de la siguiente instrucción

DIRE	LRA	LRI	LPC	R	LFZ	LFN	ERD	L/E	MUXPC	MUXRG	OPERA	CONTROL DE LA UC
BUS	0	1	1	0	0	0	0	0	0	X	X	
DEC	0	0	0	0	0	0	0	0	X	X	X	DIR A MICROPC
AL	1	0	0	0	0	0	0	0	X	1	X	
	0	0	0	0	1	1	1	0	X	2	1	SALTO ABUS
LOAD	0	0	0	1	0	0	0	0	X	1	X	
	0	0	0	0	1	1	1	0	1	X	0	SALTO A BUS
STOR	0	0	0	1	0	0	0	0	X	1	x	
	0	0	0	0	0	0	0	1	1	0	x	SALTO A BUS
BIF	0	0	0	0	0	0	0	0	X	X	X	
	0	0	0	1	0	0	0	0	X	1	X	
	0	1	1	0	0	0	0	0	1	X	X	SALTO A BUS

Dir es la dirección que se saca del código de operaciones y se carga al microPC

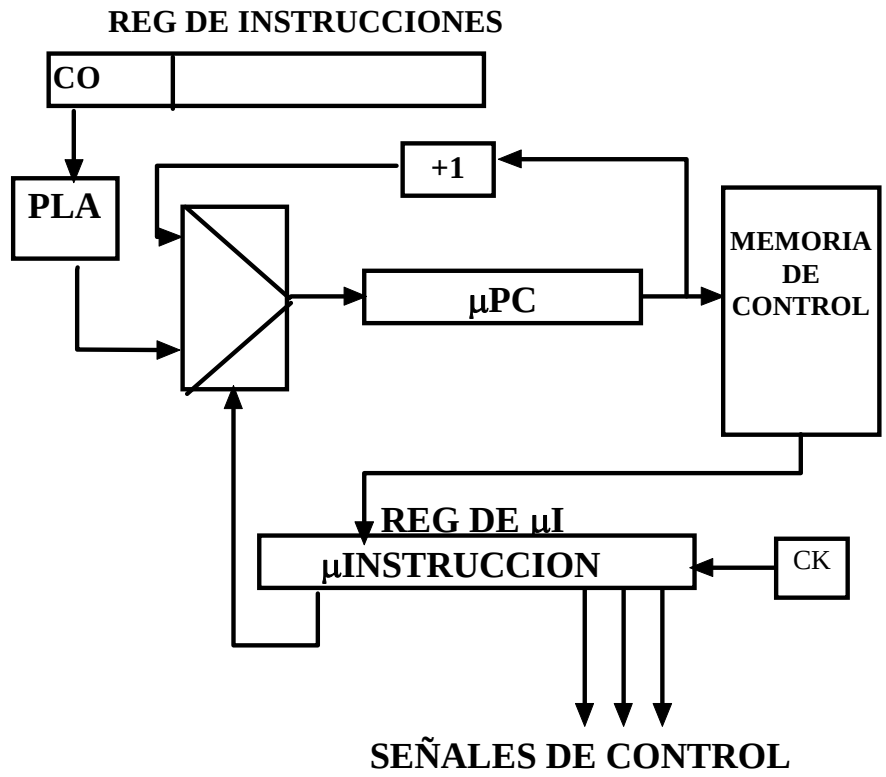
SECUENCIAMIENTO EXPLICITO

- Lo propuso Wilkes
- Incluye en la instrucción la dirección de la instrucción siguiente
- La instrucciones de un programa pueden estar desordenadas
- Posición de l microprograma
  - Se guarda la 1ª instrucción en la posición que determine el código de operación de la Instrucción
- Mecanismo secuenciamiento
  - No hace falta mecanismo de secuenciamiento
  - La bifurcación a un nuevo programa se consigue con una señal de control que toma como dirección el nuevo código de operación.
- Inconveniente:
  - Enorme gasto de memoria de control



SECUENCIAMIENTO IMPLÍCITO

- Microprograma secuencial
- La instrucción no contienen la siguiente dir
- Ahorro de memoria
- Posición del microprograma
  - Etapa traductora entre el código de operación y el Mux de direcciones.
  - Suele ser una PLA o una ROM. Y
- Secuenciamiento
  - Microcontador de programa
  - Un incrementador
- Falta el tema de bifurcación



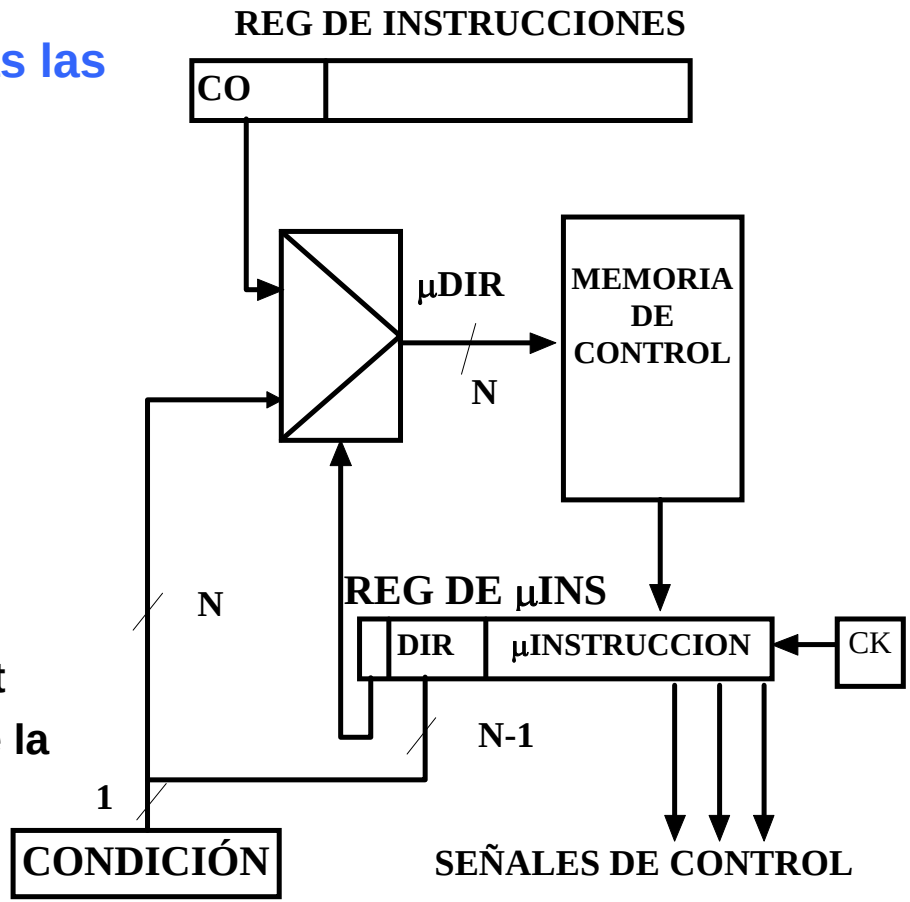
BIFURACACIONES

- **Mecanismo que permita la bifurcación a una rama o a otra en función del valor de un bit de condición**
- **Causa**
  - **Las Instrucciones tienen con frecuencia partes comunes entre sí**
    - » **Llamadas a subrutinas-->ahorro de memoria**
  - **Las secuencias de instrucciones tienden a ser muy cortas**
    - » **Cada 3 o 4 microinstrucciones una ruptura de secuencia**
  - **Capacidad de bucles**
- **Es importante que las técnicas de bifurcación estén optimadas en el tiempo**

BIFURACACIÓN

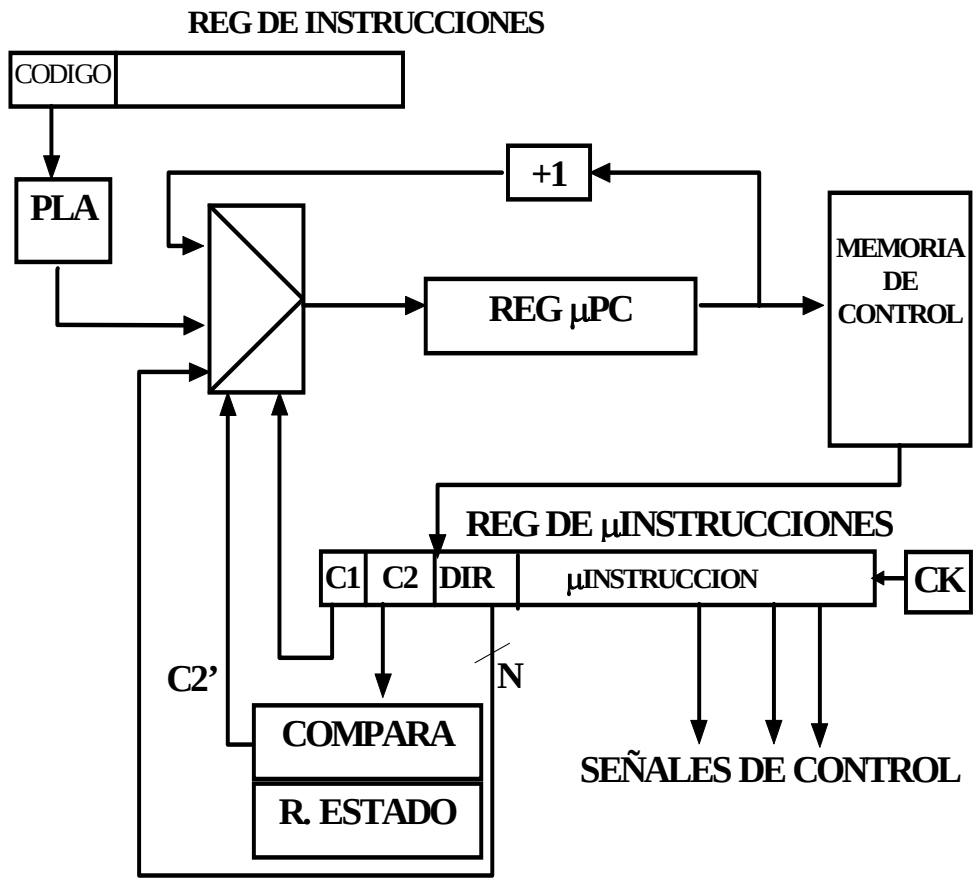
SECUENCIAMIENTO EXPLÍCITO

- solución sabiendo que son muy pocas las microinstrucciones que se activan simultáneamente
- La I contiene dos direcciones
  - La de secuencia normal
  - La de bifurcación
- Es costoso en memoria
- Solución:
  - Las direcciones difieran solo en un bit
  - El bit toma el valor del comparador de la bifurcación



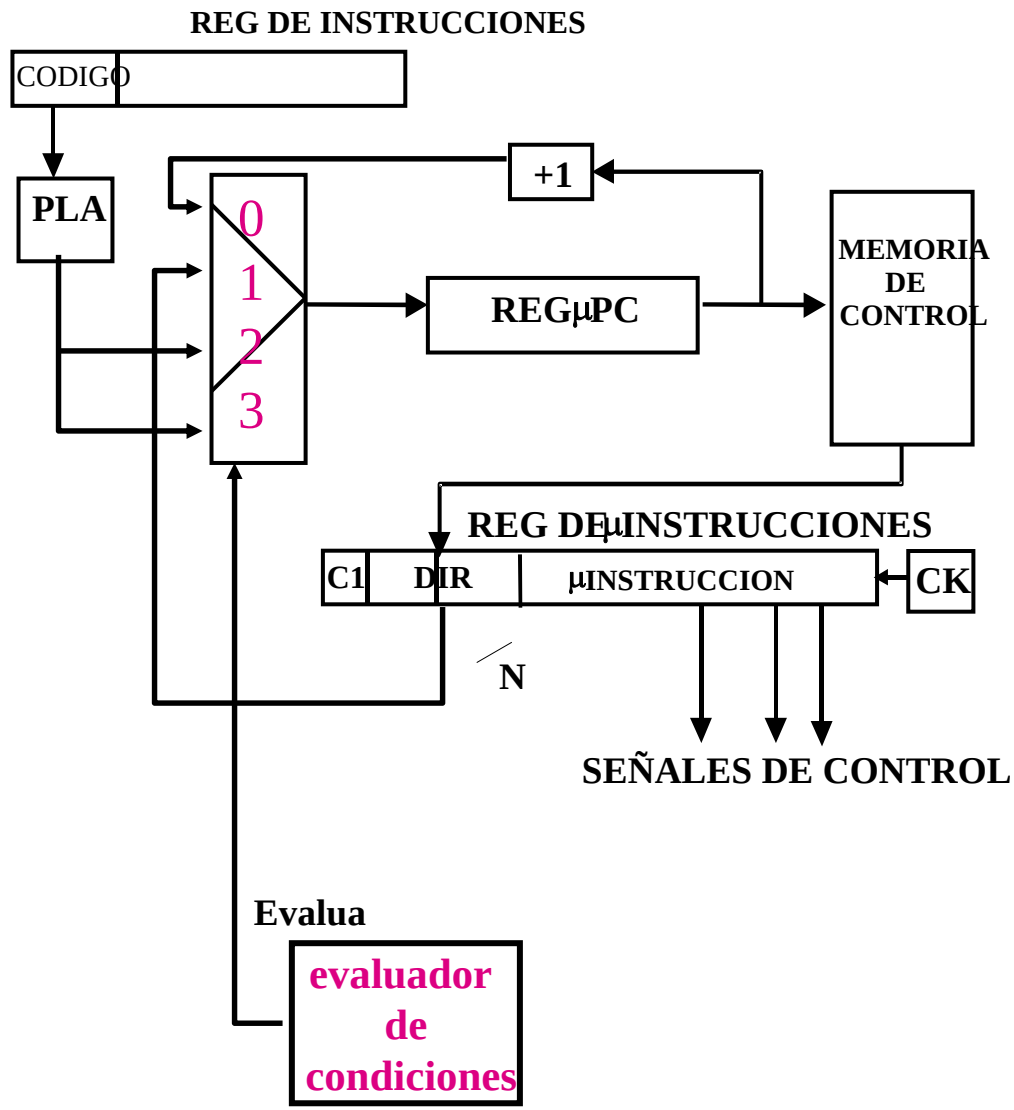
BIFURCACIONES
SECUENCIAMIENTO IMPLICITO

- C1 selecciona la dirección del RI o la dirección interna a la Unidad de control
- C2' selecciona entre dos direcciones internas de la UC
  - la secuencia normal (microPc)
  - la de salto (DIR)
- C2' se genera con información que viene del registro de estados del computador y del campo C2 del microinstruccion
- Problema
- suponiendo Dir=8 y el nº de micro instrucciones 14 esto indica que el registro de tiene 24bits
  - 33% solo se usa en las instrucciones de salto



Para la UC del camino de datos eliminamos el c2 y el comparador, porque esta información la metimos en el evaluador

Ev C1
0 0 pc
0 1 dir
1 0 pla
1 1 pla



FORMATO Y CODIFICACIÓN

- En su forma más sencilla una microinstruccion tiene
  - 1 Bit por cada señal de control
  - Una duración de un periodo
  - Solución sencilla
    - » poco empleada por ser muy cara--> mucho gasto de memoria
- generalmente una microins activa solo una señal de control en cada ciclo --> muchos 0's
- Solución:
  - Codificación de las instrucciones para que tengan una longitud más reducida
    - » modificación de los formato
- Inconveniente
  - En ocasiones más de un periodo para reducir la memoria
- Formato horizontal:
  - se usan instrucciones no codificadas
  - Mayor tamaño y paralelismo
- Programación vertical:
  - Se usan instrucciones codificadas
  - No permite el paralelismo,
  - exige descodificación (más lento),
  - menor memoria

DIRE	LRA	LRI	LPC	R	LFZ	LFN	ERD	L/E	MUXPC	MUXRG	OPERA	CONTROL DE LA UC
BUS	0	1	1	0	0	0	0	0	0	X	X	
DEC	0	0	0	0	0	0	0	0	X	X	X	DIR A MICROPC
AL	1	0	0	0	0	0	0	0	X	1	X	
	0	0	0	0	1	1	1	0	X	2	1	SALTO ABUS
LOAD	0	0	0	1	0	0	0	0	X	1	X	
	0	0	0	0	1	1	1	0	1	X	0	SALTO A BUS
STOR	0	0	0	1	0	0	0	0	X	1	x	
	0	0	0	0	0	0	0	1	1	0	x	SALTO A BUS
BIF	0	0	0	0	0	0	0	0	X	X	X	
	0	0	0	1	0	0	0	0	X	1	X	
	0	1	1	0	0	0	0	0	1	X	X	SALTO A BUS

DIRE	LRA	LI-PC	R	LFZ-N ERD	L/E	MUXPC	MUXRG	OPERA	CONTROL DE LA UC
BUS	0	1	0	0	0	0	X	X	
DEC	0	0	0	0	0	X	X	X	DIR A MICROPC
AL	1	0	0	0	0	X	1	X	
	0	0	0	1	0	X	2	1	SALTO ABUS
LOAD	0	0	1	0	0	X	1	X	
	0	0	0	1	0	1	X	0	SALTO A BUS
STOR	0	0	1	0	0	X	1	x	
	0	0	0	0	1	1	0	x	SALTO A BUS
BIF	0	0	0	0	0	X	X	X	
	0	0	1	0	0	X	1	X	
	0	1	0	0	0	1	X	X	SALTO A BUS

FORMATO

- **Especifica**
  - **el número de bits que tiene la microinstrucción**
  - **el significado de cada uno de ellos**
- **Las señales de control que sirven para accionar un mismo órgano, se agrupan formando campos**
  - **Por ejemplo:**
    - » **Señales triestado de acceso a bus**
    - » **Señales de la UAL**
    - » **Señales de banco de registro**
    - » **Señales de memoria**

reduccion del tamaño de microinstruccion

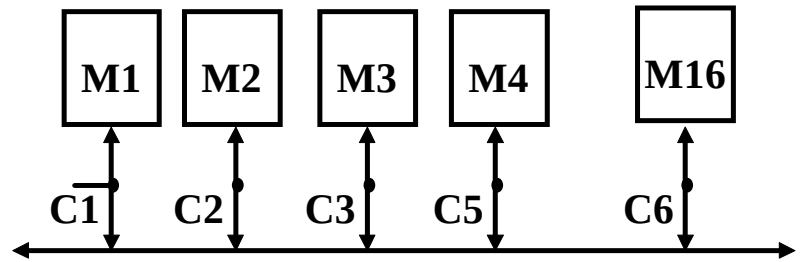
- señales de control diferentes que se activan y desactivan en los mismos ciclos se pueden juntar en una sola.

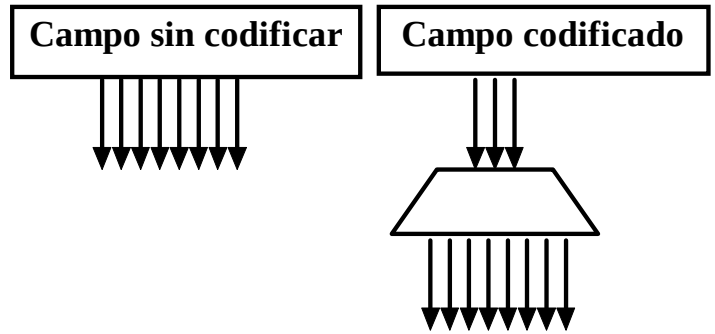
DIRE	LRA	LRI	LPC	R	LFZ	LFN	ERD	L/E	MUXPC	MUXRG	OPERA	CONTROL DE LA UC
BUS	0	1	1	0	0	0	0	0	0	X	X	
DEC	0	0	0	0	0	0	0	0	X	X	X	DIR A MICROPC
AL	1	0	0	0	0	0	0	0	X	1	X	
	0	0	0	0	1	1	1	0	X	2	1	SALTO ABUS
LOAD	0	0	0	1	0	0	0	0	X	1	X	
	0	0	0	0	1	1	1	0	1	X	0	SALTO A BUS
STOR	0	0	0	1	0	0	0	0	X	1	x	
	0	0	0	0	0	0	0	1	1	0	x	SALTO A BUS
BIF	0	0	0	0	0	0	0	0	X	X	X	
	0	0	0	1	0	0	0	0	X	1	X	
	0	1	1	0	0	0	0	0	1	X	X	SALTO A BUS

DIRE	LRA	LI-PC	R	LFZ-N ERD	L/E	MUXPC	MUXRG	OPERA	CONTROL DE LA UC
BUS	0	1	0	0	0	0	X	X	
DEC	0	0	0	0	0	X	X	X	DIR A MICROPC
AL	1	0	0	0	0	X	1	X	
	0	0	0	1	0	X	2	1	SALTO ABUS
LOAD	0	0	1	0	0	X	1	X	
	0	0	0	1	0	1	X	0	SALTO A BUS
STOR	0	0	1	0	0	X	1	x	
	0	0	0	0	1	1	0	x	SALTO A BUS
BIF	0	0	0	0	0	X	X	X	
	0	0	1	0	0	X	1	X	
	0	1	0	0	0	1	X	X	SALTO A BUS

REDUCCION DEL TAMAÑO DE MICROI
CODIFICACIÓN DE CAMPOS

- Suponer un bus al que acceden 16 módulos diferentes
  - 15 señalaes de control
  - desventaja:
    - » solo una señal activa en cada ciclos
    - » 14 desaprovechados en cada ciclo:
- solución codificacion
  - solo se necesitan 4 bits
- Inconveniente:
  - Se necesita una etapa decodificadora
    - » Aumento del coste
    - » Aumento del retardo





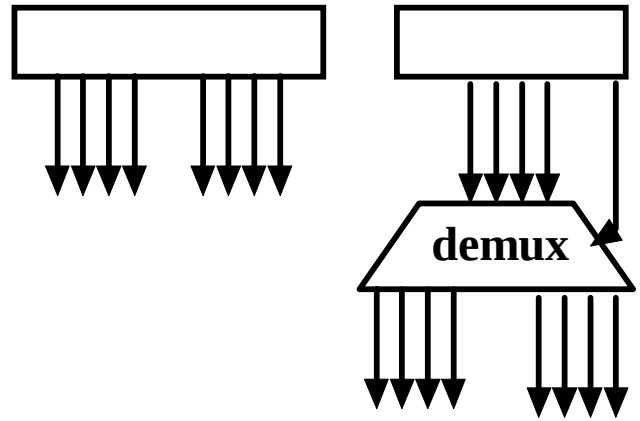
- **En nustro ejemplo se podrían codificar las señales LDR@, LRI y LRA mediante un campo de dos bits:**

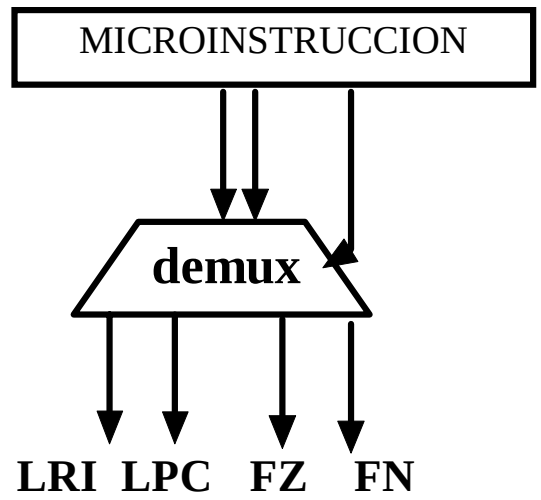
DIRE	LRA	LRI	LPC	R	LFZ	LFN	ERD	L/E	MUXPC	MUXRG	OPERA	CONTROL DE LA UC
BUS	0	1	1	0	0	0	0	0	0	X	X	
DEC	0	0	0	0	0	0	0	0	X	X	X	DIR A MICROPC
AL	1	0	0	0	0	0	0	0	X	1	X	
	0	0	0	0	1	1	1	0	X	2	1	SALTO ABUS
LOAD	0	0	0	1	0	0	0	0	X	1	X	
	0	0	0	0	1	1	1	0	1	X	0	SALTO A BUS
STOR	0	0	0	1	0	0	0	0	X	1	x	
	0	0	0	0	0	0	0	1	1	0	x	SALTO A BUS
BIF	0	0	0	0	0	0	0	0	X	X	X	
	0	0	0	1	0	0	0	0	X	1	X	
	0	1	1	0	0	0	0	0	1	X	X	SALTO A BUS

C1	c0	LDRA	LDRI	LDR@
0	0	0	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

REDUCCION DEL TAMAÑO DE MICROI
SOLAPAMIENTO DE CAMPOS

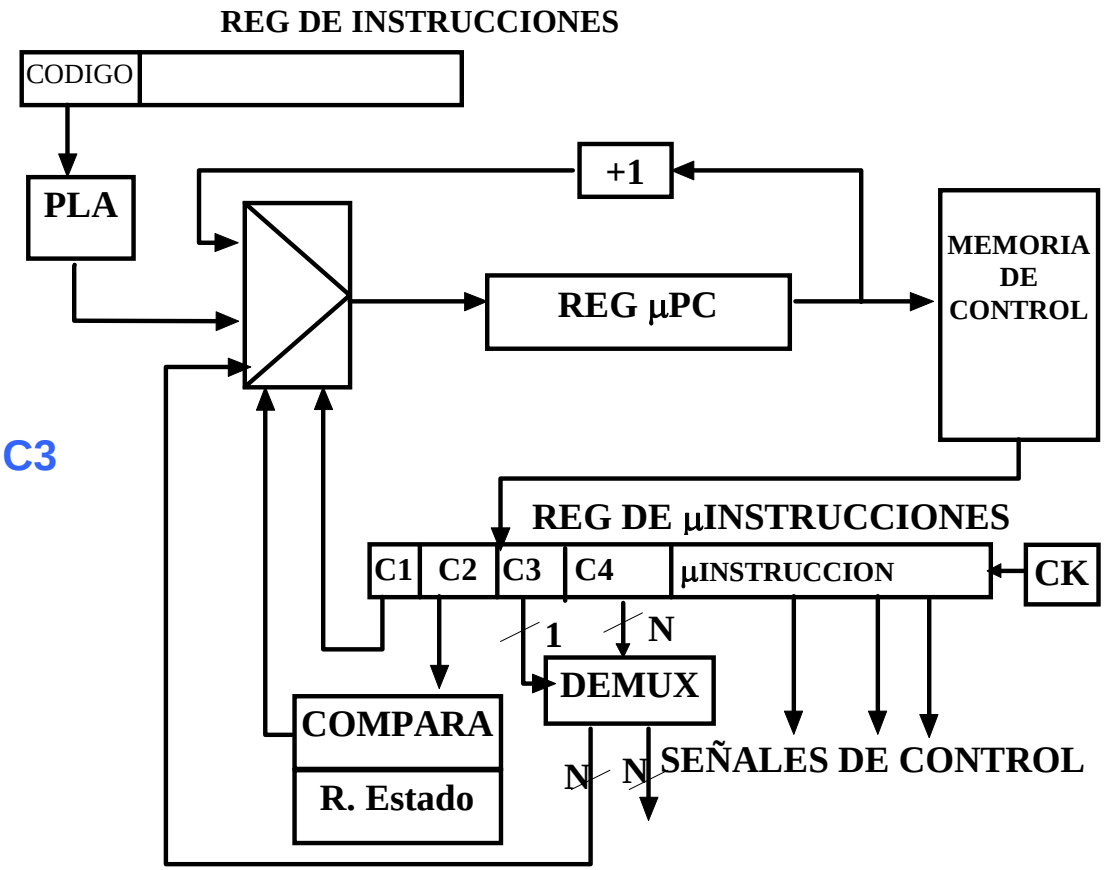
- **Solapamiento de campos:**
  - **Existen con frecuencia señales excluyentes**
    - » **No se pueden activar simultáneamente.**
  - **Permite emplear un único grupo de bits para dos campos distintos y se emplea un señal auxiliar para descodificar**
- **Inconveniente:**
  - **Se necesita una etapa decodificadora**
    - » **Aumento del coste**
    - » **Aumento del retardo.**





BIFURCACIONES
SECUENCIAMIENTO IMPLICITO

- Instrucción deberá contener la dir
- Se usan campos solapados
- C4 puede ser
  - la dirección de una I
  - otra operación
- Se demultiplexan mediante la señal C3
- C2
  - contiene la condición
  - sirve para seleccionar el salto



MICROPROGRAMACIÓN

VENTAJAS Y DESVENTAJAS

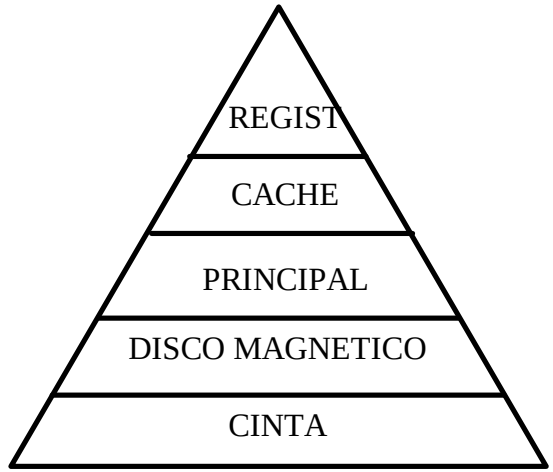
- **Desventaja:**
  - **Demasiado compleja para máquinas sencillas**
  - **Demasiado lenta para máquinas complejas**
- **Ventaja:**
  - **Su simplicidad conceptual.**
  - **Información almacenada en memoria ROM y RAM**
  - **Correcciones más sencillas**
  - **Instrucciones complejas con muchos periodos de duración: M. Control grande**
  - **Algunas funciones del S.O. Se pueden integrar a este nivel**
  - **Un mismo computador tiene varios juegos de instrucción**
  - **Emulación de otros computadores.**
  - **Rutinas de diagnóstico muy completas**
- **Aplicación de la microprogramación**
  - **Computadores de tamaño medio**
  - **En los pequeños la estructura es demasiado compleja y no resulta económica**
  - **En los grandes demasiado lento**

LA JERARQUÍA DE MEMORIA

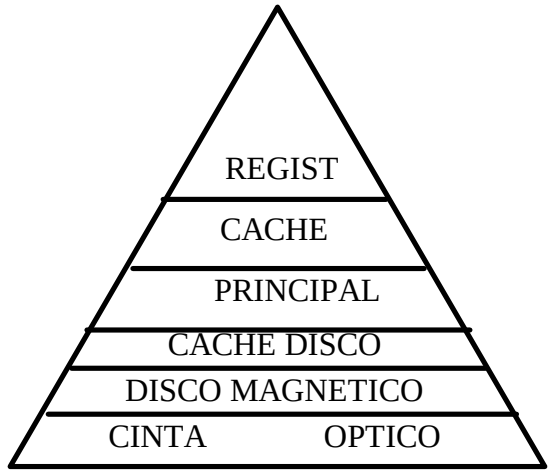
TEMA9

INTRODUCCIÓN

- **La elección de la memoria depende de**
  - **Cantidad**
  - **Rapidez**
  - **Precio**
- **Cantidad**
  - **A más memoria mayor necesidad de los programas**
- **Rapidez**
  - **Objetivo es que la memoria y la CPU trabajen a velocidades parecidas para eliminar tiempos de espera**
- **Hay que buscar un equilibrio entre las tres ligaduras:**
  - **rapidas: caras y pequeñas**
  - **grandes baratas y lentas**
- **La solución**
  - **La jerarquía de memoria:**
    - » **Diferentes tipos de memoria organizados y relacionados de manera que optiman todas las ligaduras**



tradicional



actual

INTRODUCCIÓN

- **Al descender en la jerarquía:**
  - » **Decrementa el coste/bit**
  - » **Incrementa la capacidad**
  - » **Incrementa los tiempos de acceso**
  - » **Decrementa la frecuencia de accesos**

- **La clave de la jerarquía es el Decremento de las frecuencias de acceso a los niveles inferiores**

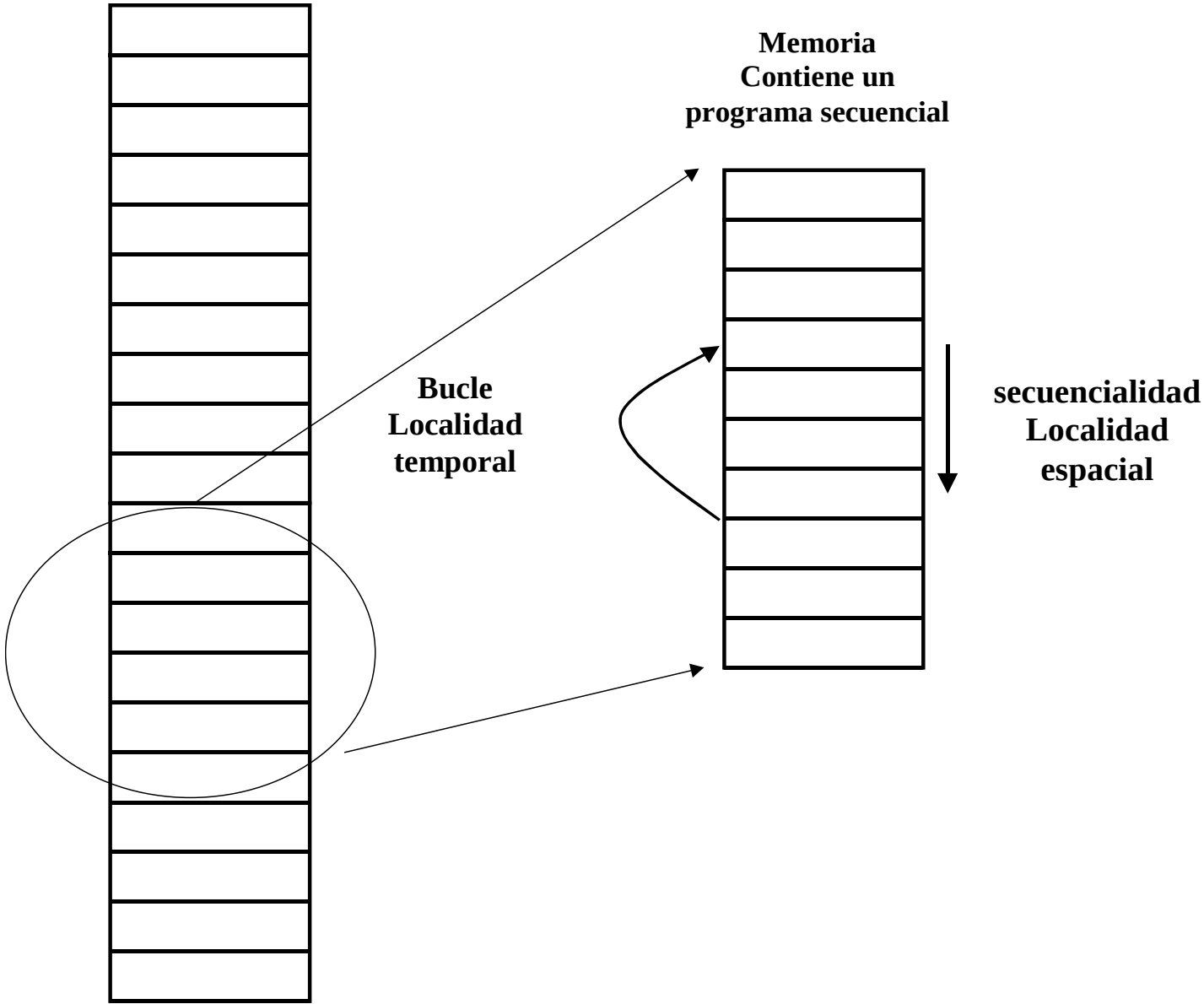
- **La base teórica que lo apuntala es el principio de localidad de referencia**

TECNOLOGÍA	TIEMPOS DE ACCESO	DOLARES POR MBYTE(1993)
SRAM	8-35NS	100-400
DRAM	90-120NS	25-50
MAGNETICOS	10.000000-20.000.000	1-2

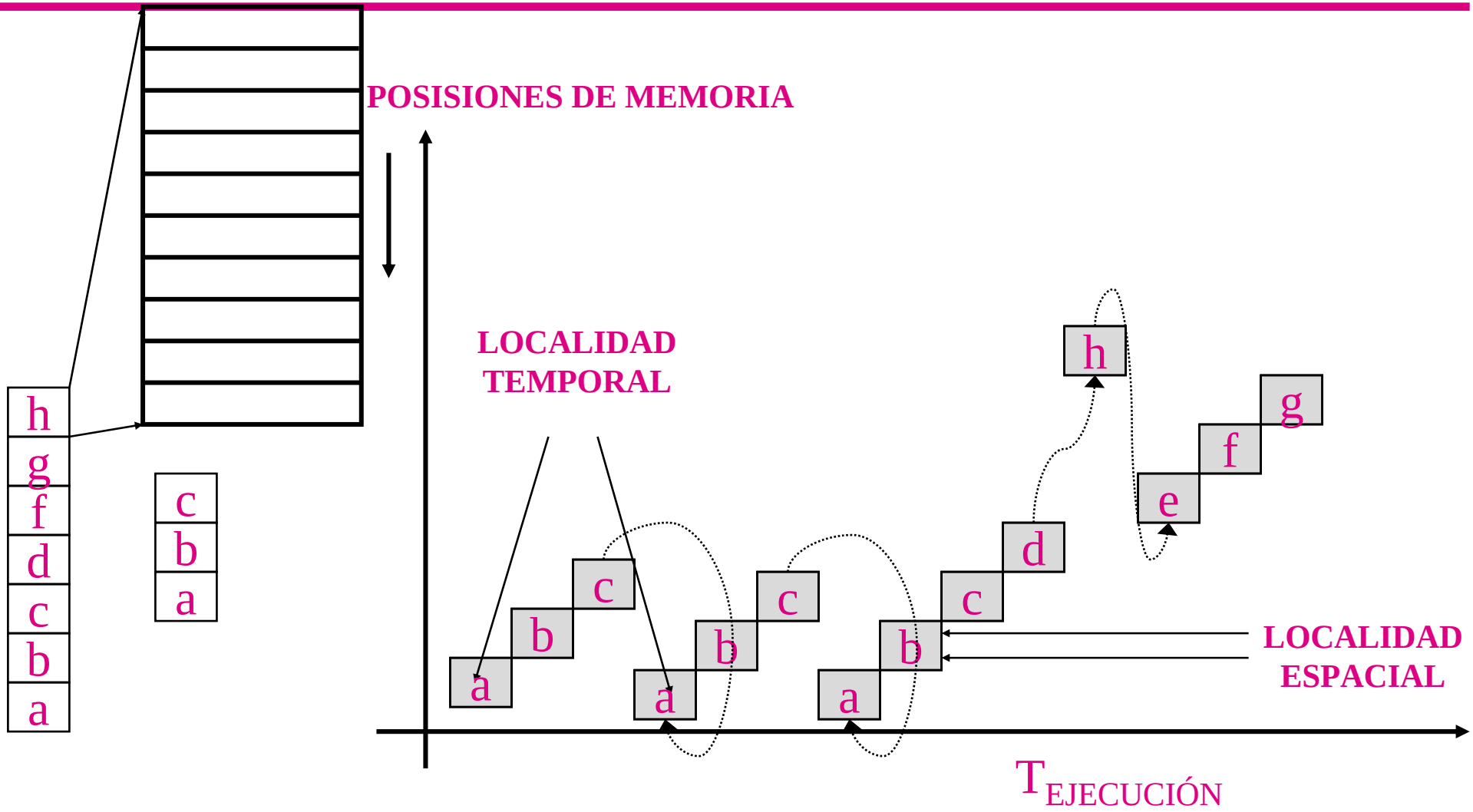
PRINCIPIO DE LOCALIDAD

- **Dado un intervalo de tiempo, un conjunto de instrucciones se referencian repetidamente mientras que el acceso al resto del programa se efectúa con poca frecuencia**
- **Localidad temporal:**
  - **Probabilidad de que una instrucción recién referenciada se vuelva a referenciar en un instante breve de tiempo.**
    - **Bucles**
    - **Subrutinas**
- **Localidad espacial:**
  - **Probabilidad de que la instrucción a referenciar este próxima en el espacio a la recién referenciada**
    - **Estructuras de datos como tablas y arrays**
    - **Programas secuenciales**

PRINCIPIO DE LOCALIDAD

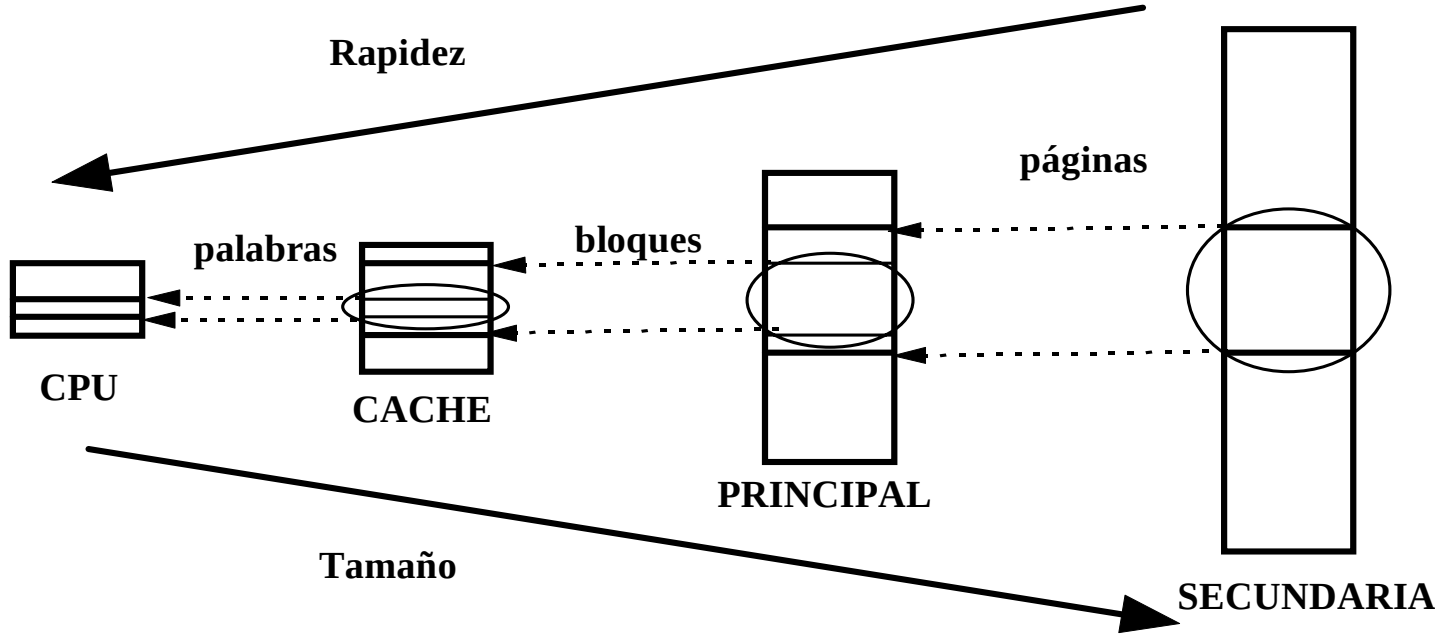


PRINCIPIO DE LOCALIDAD



ORGANIZACIÓN DE LA JERAQUÍA

- Es posible organizar los datos en la jerarquía, de manera que el porcentaje de accesos a cada nivel inferior sea menor al del nivel superior
- Organización de la jerarquía
  - El espacio de direcciones del nivel superior es un subconjunto del espacio del nivel inferior
  - Problema de consistencia el nivel i+1 se modifica con la información del i

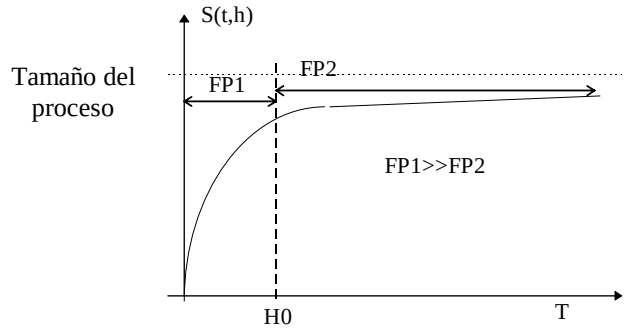


DEFINICIONES

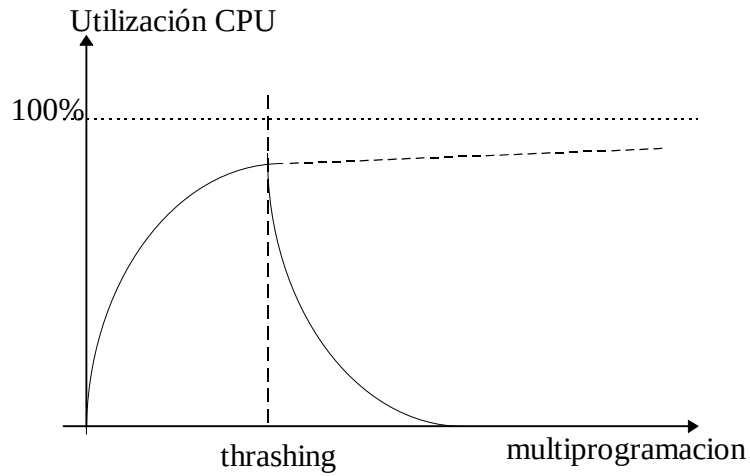
- **Acierto**
  - **El dato requerido por el procesador aparece en el nivel superior de la jerarquía**
- **Fallos**
  - **El dato no se encuentra en el nivel superior**
- **Tasa de aciertos**
  - **% De accesos encontrados en el nivel superior**
- **Tasa de fallos**
  - **% Accesos no encontrados en el nivel superior = (1- tasa de aciertos)**
- **Como el rendimiento es la razón principal de una jerarquía de memoria, la velocidad de aciertos y de fallos es importante.**
- **Tiempo de aciertos**
  - **Tiempo que se tarda en acceder al nivel superior de memoria.**
  - **Incluye el tiempo necesario para comprobar si el acceso es un acierto o un fallo**
- **Penalización de fallos**
  - **Tiempo que se necesita para sustituir un bloque de nivel superior por uno de nivel inferior mas el tiempo para entregar este bloque al procesador**
- **Como el nivel superior es más pequeño y se construye utilizando partes de memoria más rápidas, el tiempo de acceso será mucho menor que el necesario para acceder al siguiente nivel.**

MEMORIA VIRTUAL
FUNDAMENTOS

- **CONJUNTOS DE TRABAJO**
  - **CONJUNTO DE PÁGINAS DEL PROCESO QUE PUEDEN SER REFERENCIADAS EN UN INTERVALO DE TIEMPO**
  - **w(T, H) T INSTANTE DE TIEMPO, H NUMERO DE REFERENCIAS**
- **SI UN PROGRAMA TIENE TODO SU CONJUNTO DE TRABAJO EN MP OCURRIRÁN POCOS FALLOS DE PÁGINA SEGÚN AVANZA LAS COMPUTACIÓN**
- **LOS FALLOS SE INCREMENTAN NOTABLEMENTE SI LOS CONJUNTOS DE TRABAJO NO ESTÁN DISPONIBLES**
- **EL ESFUERZO DE ADMINISTRACIÓN DEBE DIRIGIRSE A QUE TODO EL CONJUNTO DE TRABAJO ESTE EN MEMORIA**
- **SI ESTO NO SE CUMPLE THRASHING**
- **CUANTO MÁS FALLOS DE PÁGINA MÁS SE ALEJA LOS TIEMPOS DE ACCESO DE LOS DE LA MP**
- **IMPORTANCIA EN LA MULTIPROGRAMACIÓN**



h→ nº bloques del conjunto de trabajo
h0 → tamaño de conjunto de trabajo óptimo
FP Nº de fallos de página



TIPOS DE MEMORIAS

- **REGISTROS:**
  - Rápidas
  - Pequeñas
  - Caras.
  - Volátil
  - Semiconductor

- **CACHE:**
  - En muchas ocasiones, tecnología bipolar
  - Estáticas (SRAM)
  - Entre los registros y la m.p.
  - No accesible por el usuario
  - Volátil

- **MEMORIA PRINCIPAL:**
  - También llamada real
  - Semiconductor
  - Dinámica (DRAM)
  - Alta densidad
  - Tecnología CMOS
  - Volátil
  - Gran tamaño, velocidad media

TIPOS DE MEMORIAS

- **LA CACHE DE DISCO:**
  - No esta físicamente separada de la MP
  - Porción de la MP usada como almacenamiento temporal de datos que deben escribirse en el disco
  - Mejoras:
    - » En lugar de muchas transferencias pequeñas, solo una grande
    - » Mejora del rendimiento del disco
    - » Reduce la intervención del procesador en I/O
    - » Algunos datos se pueden referenciar antes de hacer la escritura a disco, lo que evita hacer una nueva lectura del disco

- **MEMORIA SECUNDARIA:**
  - Almacenamiento permanente de datos y programas
  - Dispositivos externos
  - Información en forma de ficheros y registros
  - Discos, cintas y ópticos
  - Disco como extensión de M.P. da lugar a la virtual

MEDIDAS DEL RENDIMIENTO

- **TIEMPO DE ACCESO**
  - **Para memoria aleatorias**
    - » **Idéntico para todas las posiciones**
    - » **El tiempo que tarda en ejecutarse una operación de lectura/escritura,**
      - **Desde el instante en que la dirección está presente en la memoria, hasta el instante en que el dato se ha escrito o leído**
  - **Memorias de acceso secuencial**
    - » **Tiempo que se tarda en situar el mecanisnmo de lectura / escritura en la posición deseada**
    - » **Se suele utilizar el tiempo de acceso promedio**

- **TIEMPO DE CICLO**
  - **Es el tiempo de acceso, más el tiempo adicional que se necesita para realizar el siguiente acceso**
  - **La existencia de este tiempo adicional se debe:**
    - » **Regeneración del dato en las lecturas destructivas**
    - » **La precarga**

MEDIDAS DEL RENDIMIENTO

- **VELOCIDAD DE TRANSFERENCIA**
  - **La velocidad a la que los datos pueden transferirse a, o , desde la memoria**
  - **En memorias aleatorias :**
    - » **$1/T_{CICLO}$**
  - **Para memorias de acceso no aleatorias:**
    - » **$T_N = T_A + N/R$**
    - » **donde:**
      - **T_N promedio en leer o escribir N-bits**
      - **T_A tiempo de acceso promedio**
      - **N número de bits**
      - **R razón de transferencia en bits por segundo.**

LA MEMORIA PRINCIPAL

TEMA 10

TIPOS DE MEMORIA PRINCIPAL

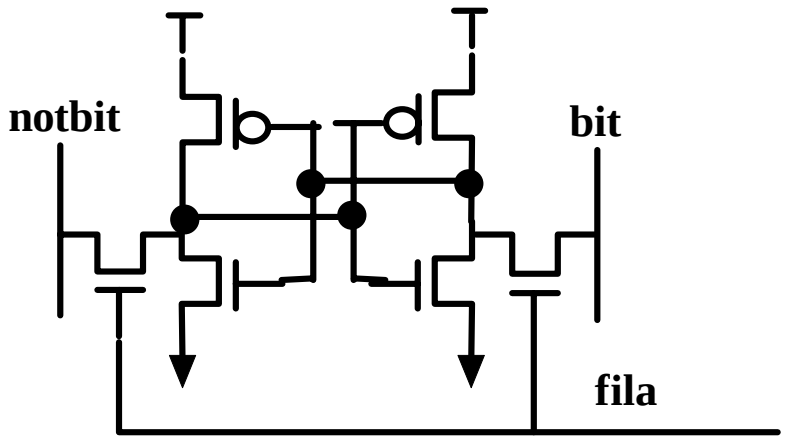
- **RAM**
  - **escritura/lectura**
- **ROM**
  - **solo lectura**

RAM
CARACTERISTICAS

- **Semiconductoras**
  - **CMOS**
- **Acceso aleatorio**
  - **el mismo tiempo de acceso para cualquier posición de la memoria**
- **Operaciones**
  - **Lectura**
  - **Escritura**
- **Característica**
  - **Rápidas**
    - » **entre 8-35 ns y 90 - 120 ns**
- **Información volátil**
  - **Al interrumpir Vdd se pierde**
- **Pueden ser :**
  - **Estáticas (SRAM)**
  - **Dinámicas (DRAM)**

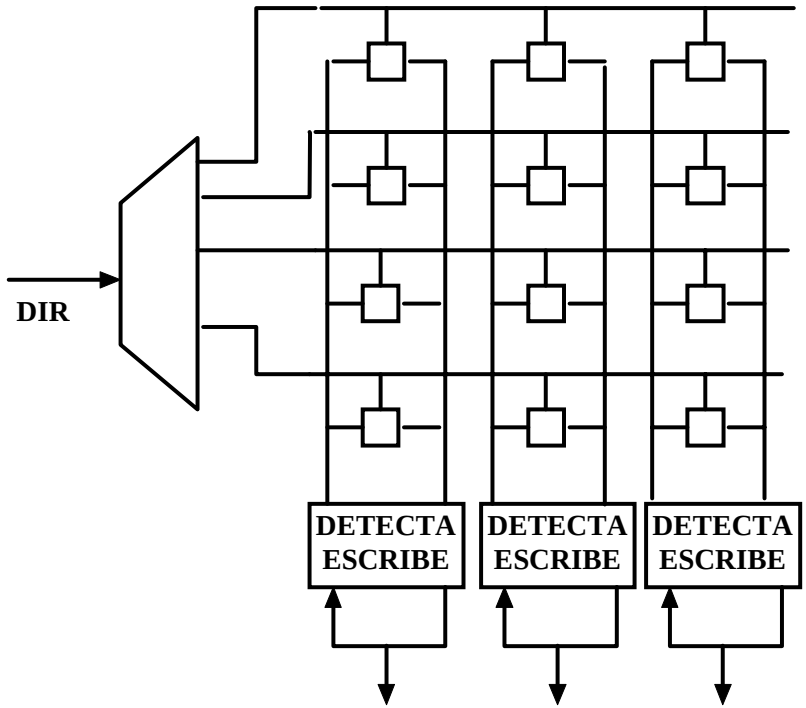
RAM ESTATICAS
(SRAM)

- SRAM
- Tecnología CMOS
- Realimentación: configuraciones flip-flop clásicas
- No necesita refresco
- Celdas grandes (6 transistores)
- Más caras: menor densidad



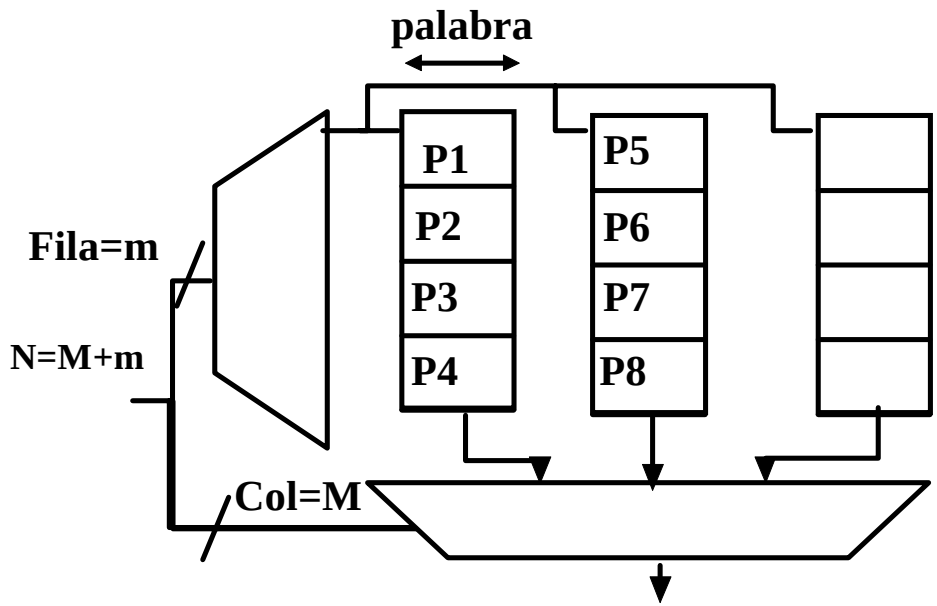
SRAM
ORGANIZACIÓN INTERNA (I)

- **Arrays de celdas básicas.**
  - Una fila: una palabra de memoria
  - Una columna: un bit de la palabra
- **Celdas unidas por 2 líneas llamadas de bit**
  - Conectadas a un circuito detectar/escribir
    - » Circuito de I/O
- **Los circuitos detectar/escribir conectados a líneas bidireccionales**
  - Para ahorrar patillas al Circuito Integrado
- **Esta organización se utiliza pocas veces**
  - Razones:
    - » En memorias grandes (16M)
      - Descodificadores demasiado grandes y complejos
        - Una salida de un decodificador es un mintermino
      - Grandes retardos en las líneas de bits



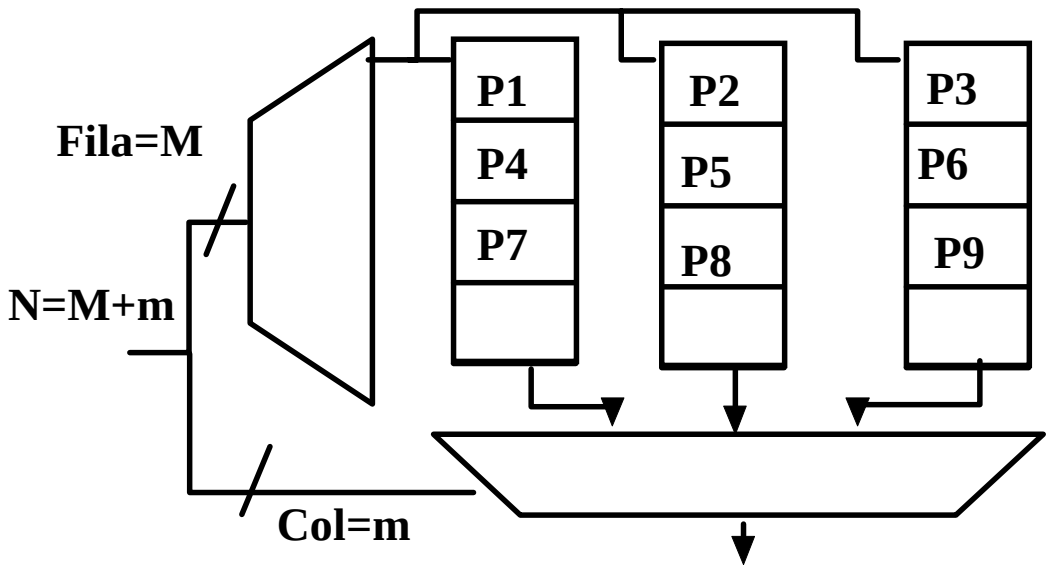
SRAM
ORGANIZACIÓN INTERNA (II)

- **Solución:**
  - Se descompone la memoria en varios bloques
- **La dirección se descompone en**
  - Una dirección de fila
  - Una dirección de columna
- **Ventaja**
  - Se eliminan los descodificadores demasiado grandes
  - Se eliminan las lineas de bit grandes
- **El descodificador de las columnas se implementa como un multiplexor**
- **Existen dos organizaciones diferentes.**
  - Palabras consecutivas se colocan en posiciones consecutivas del mismo bloque de memoria
    - » lo mas significativo selecciona la columna



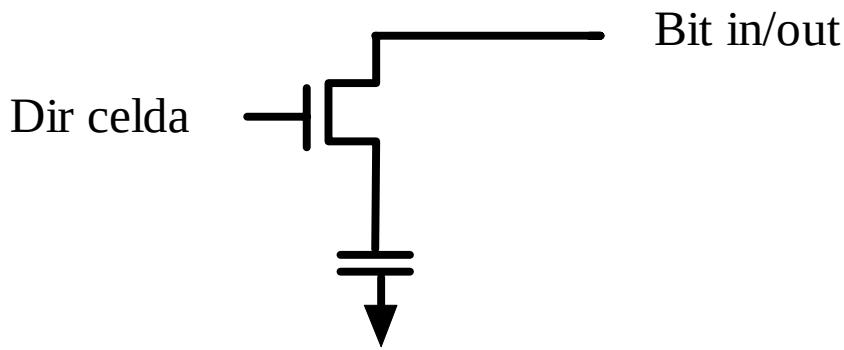
ORGANIZACIÓN INTERNA (III)

- **Palabras consecutivas se colocan en las mismas posiciones de bloques consecutivos**
  - » **Lo mas significativo selecciona la fila**
  - » **Paralelismo de acceso**



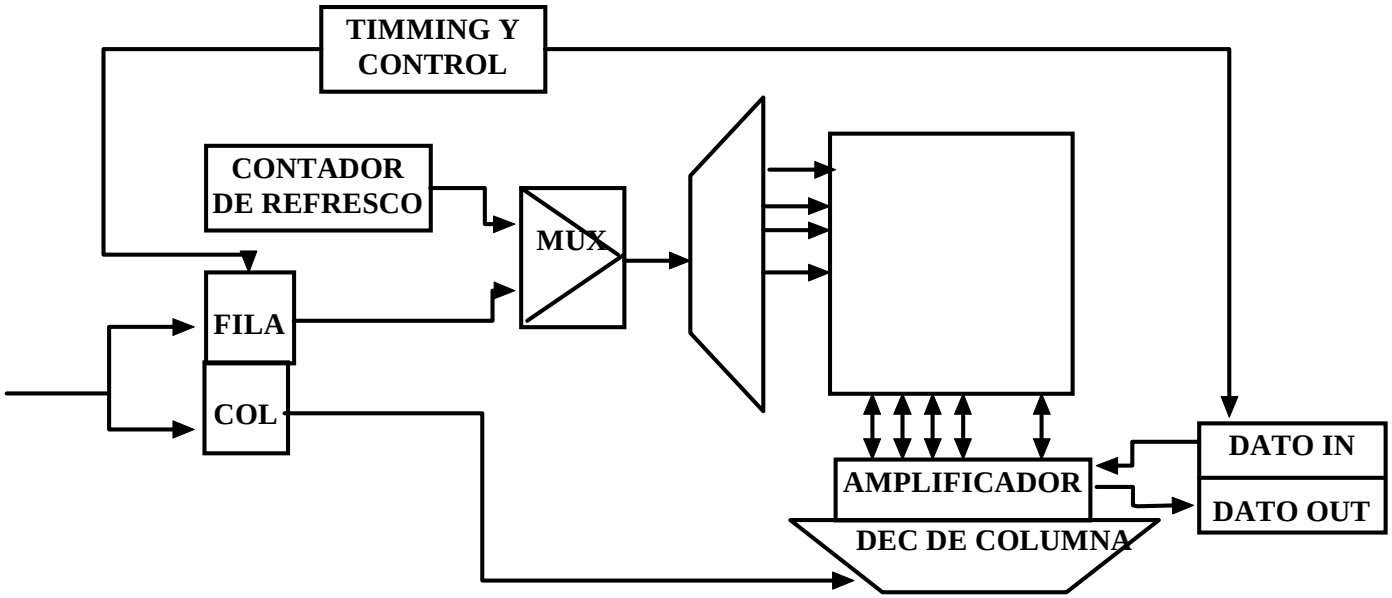
RAM DINÁMICAS

- DRAMS
- El dato se guarda en una capacidad
- Necesidad de refresco
- Celda de memoria simple (1 transistor)
- Alta densidad de celdas
- Precio por bit pequeño
  - Aprovecha mucho el área
- Circuitería extra de refresco
- Circuitería I/O compleja:
  - Relentiza las operaciones de I/O



ORGANIZACIÓN RAM DINÁMICA

- La memoria debe incluir la circuitería de refresco
- No se refresca toda la memoria a la vez, sino fila a fila
- El contador de refresco contiene la dirección que se debe refrescar
- Generalmente la forma de actuar es realizar una escritura de un buffer y posteriormente una lectura



ROM

- Acceso aleatorio
- Tecnologías semiconductoras
  - CMOS
- Sólo lectura: patrón permanente de datos que no se puede cambiar
- No volatil
- Aplicación:
  - microprogramación
  - Librerías de subrutinas frecuentes
  - Programas de sistemas
  - Tablas de funciones
- Ventaja:
  - Programa almacenado en Memoria principal sin realizar carga de un sistema secundario
  - Los datos están cableados en el CI como parte del proceso de fabricación.
- Desventajas:
  - Coste elevado independiente del número de copias de la ROM
  - No hay lugar para el error. Sin un bit falla, falla toda la ROM

ROM

- **Existen diferentes alternativas ROM:**
  - **PROM**
  - **EPROM**
  - **EEPROM**
  - **FLASH MEMORY**

- **PROM:**
  - **ROM programable**
  - **se usan cuando el número de ROM a utilizar es pequeño**
    - » **Abarata el coste**
  - **Información no volátil**
  - **Se escribe eléctricamente**
  - **Sólo se puede escribir una vez**
  - **Se realiza después de la fabricación**
  - **La alternativa ROM se suele dejar para grandes volúmenes de producción**

ROM

- **Aplicaciones de sobre todo lectura**
  - **Operaciones de lectura bastante más frecuentes que las de escritua**
  - **Necesidad de almacenamiento no volatil**
  - **Tipos:**
    - **EPROM**
    - **EEPROM**
    - **FLASH**

ROM

- **EPROM (Erasable Programmable Read Only)**
  - **Se lee y escribe eléctricamente**
  - **Antes de la escritura se borra toda la memoria mediante radiación ultravioleta**
  - **Este proceso se puede realizar repetitivamente**
  - **Dura 20 minutos**
  - **Puede modificarse muchas veces**
  - **Información no volatil**
  - **Más cara que la PROM**

ROM

- **EEPROM (Electically Erasable Programable Read Only Memory)**
  - **Se puede escribir sin tener que borrar previamente el contenido**
  - **Solo se actualizan los bits direccionados**
  - **Escritura más tiempo que lectura**
  - **Información no volátil**
  - **Ventajas:**
    - » **No volatilidad**
    - » **Actualización en el lugar: usando el bus control ordinarios**
  - **Desventajas:**
    - » **Menos densa**
    - » **Más cara que la EPROM**

ROM

- **Flash Memory:**
  - **Entre la EPROM y la EEPROM tanto en coste como en funcionalidad**
  - **Tecnología de borrado eléctrico**
  - **Se puede borrar toda la memoria en unos pocos segundos: más rápido que la EPROM**
  - **Permite borrar bloques de datos**
  - **No permite borrar palabras independientes**
  - **Solo usa un transistor por bit: alta densidad comparada con EEPROM**

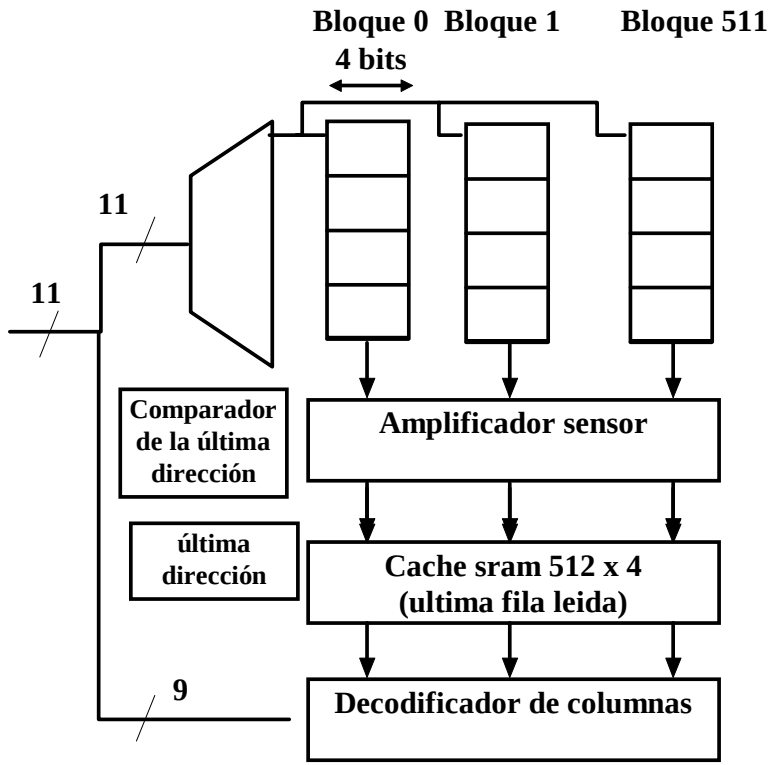
DRAMS AVANZADAS
DRAM MEJORADA (EDRAM)

- **Los chips de DRAM tradicionales tiene sus principales ligaduras**
  - **En su arquitectura interna**
  - **En el interfaz con el bus de memoria**

- **DRAM MEJORADA (EDRAM)**
  - **La más simple de las nuevas arquitecturas**
  - **Desarrollada por Ramtron**
  - **Incluye una pequeña cache (2kb) SRAM en el chip DRAM**
  - **Cada fila almacena varias palabras**
  - **La cache almacena el contenido de la última fila leída**
  - **Ventajas:**
    - » **Si el siguiente acceso se realiza a la misma fila, ésta ya se encuentra en la cache.**
    - » **Las operaciones de refresco se realizan en paralelo con las operaciones de lectura de la cache**
    - » **Escrituras en paralelo con lecturas de la cache**
  - **Rendimiento similar al de una DRAM con cache externa mayor**

edram

- Memoria EDRAM de 4 megas
- Organización interna
  - 2 11 = 2048 filas
  - cada fila 2048 bits organizados en
    - » 512 bloque direccionados por 9 bits
    - » 4 bits cada bloque



DRAMS AVANZADAS
DRAM CACHE (CDRAM)

- Desarrollada por Mitsubishi
- Similar a la EDRAM
- Incluye una cache de mayor tamaño (16Kb)
- Esta cache se puede usar de dos modos
  - Como una cache normal contiene varias filas de 64 bits cada una
    - » modo efectivo para los usuales accesos aleatorios a la memeoria
  - Como buffer para proporcionar accesos serie a un bloque de datos , por ejemplo para refrescar una pantalla gráfica

DRAMS AVANZADAS
DRAM SINCRONA (SDRAM)

- A diferencia de las DRAM típicas que son asícronas esta es sícrona
- Reloj exterior que opera a la velocidad máxima del bus procesador/memoria,
  - Sin imponer estados de espera -->Síncrono de ciclo partido
- Accede por ráfagas
  - No lee byte a byte sino un conjunto de bytes consecutivos
    - » Para aprovechar los tiempos de establecimiento de dirección y precarga
  - En un registro se almacena el tamaño de las ráfagas
    - » No hace falta que lleguen todas las direcciones --> contador interno
  - Permite ajustar las latencias de espera entre la petición de una lectura y el comienzo de la transferencia de datos
- Arquitectura de doble banco (dos submatrices) que facilita el paralelismo
- Funciona bien para la transferencia de grandes bloque de datos tales como
  - Procesamiento de texto
  - Hojas de cálculo
  - Multimedia

DRAMS AVANZADAS
DRAM RAMBUS (RDRAM)

- Desarrollada por RAMBUS
- corazon de una DRAM estandar proporcionando una nueva interface
- Encapsulados vertical con todos los terminales en un lateral
- El chip intercambia información con el procesador por medio de 28 hilos con un tamaño inferior a 12 cm de longitud
- Asincrono de ciclo partido
- El bus puede direccionar hasta 320 chips de RDRAM a razón de 500 mbp (
  - Fuerte contraste con los 33 mbps de las DRAM asíncronas
- El bus especial de las RDRAM entrega direcciones e información de control utilizando un protocolo asíncrono orientado a bloques
- Las RDRAM obtienen peticiones de memoria a través de un bus de alta velocidad
- Cada petición contiene
  - La dirección deseada
  - Tipo de operación
  - El número de bytes de la operación

MEMORIAS EXTERNAS

TEMA 11

DISCOS MAGNETICOS

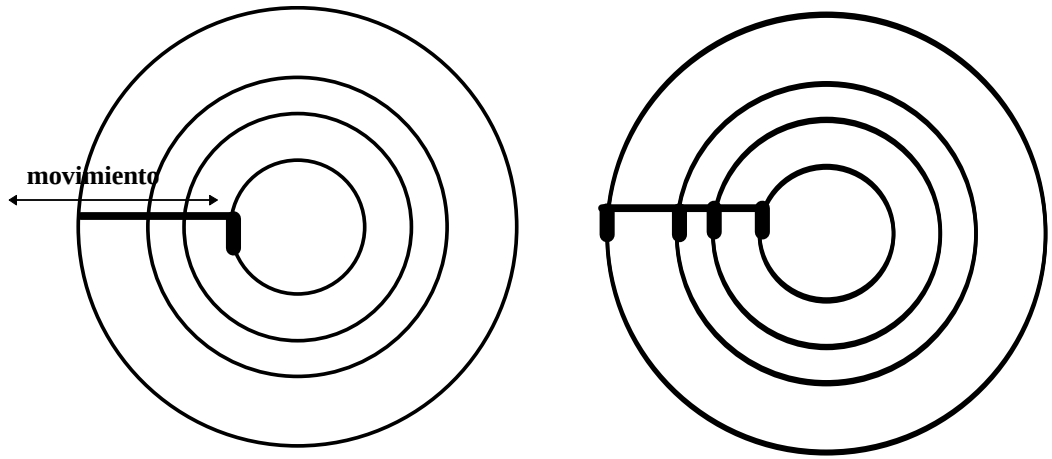
- Son los almacenamientos exteriores más rápidos
- Forma circular
- Material : metal o plástico recubierto de material magnético
- Los datos se escriben/leen mediante un rollo conductor llamado cabeza transductora
- Mecanismo de escritura:
  - Se basa en los campos magnéticos producidos por el flujo de corriente en el rollo conductor
  - Se generan patrones magnéticos en la superficie del disco
- Mecanismo de lectura
  - Los cambios de campo magnético producen cambios de corriente eléctrica en la cabeza

ORGANIZACIÓN DE LOS DISCOS

- **Pistas (Track):**
  - Anillos concéntricos de información llamados pistas
  - Todas las pistas contienen la misma cantidad de información
    - » Más densa en las pistas interiores que en las exteriores
    - » Esto simplifica la electrónica --> velocidad angular constante
  - Las pistas se separan por espacios llamados intertrack gap
- **Sector**
  - Cada pista se divide en zonas llamada sectores Menor cantidad de información que puede ser leida o escrita
  - Hay entre 10 - 100 sectores por pista
  - Puede ser de longitud fija o variable
  - Los sectores se separan por espacios llamados inter record gap
  - Identificación de un sector de una pista
    - » Se debe tener un punto inicial de comienzo de la pista
    - » Marcas de principio y fin de sector
    - » Esto implica un formateado del disco con información extra

CARACTERÍSTICAS

- Cabeza fija una cabeza de lectura escritura por cada pista
- Cabeza móvil
  - Solo tiene un transductor
  - Debe ser colocado sobre la pista a la que se desea acceder
- En relacióncon la distancia al disco existen tres mecanismos de cabeza:
  - Cabeza a distancia fija de la superficie
  - Cabeza en contacto con la superficie (floppy)
  - Cabeza móvil

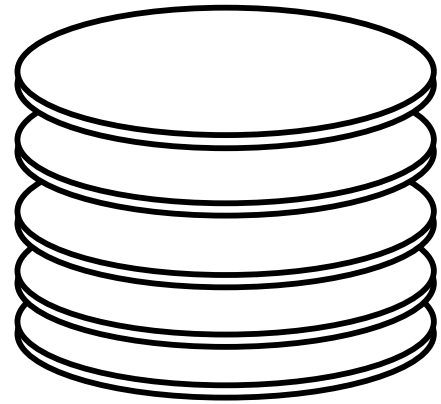


Densidad de datos

- Existe una relación entre la densidad de datos y la distancia entre la cabeza y la superficie
  - La cabeza debe generar o sentir el campo magnético con la suficiente intensidad como para poder leer o escribir
  - Cuanta más cantidad de información se desea almacenar en el disco más estrechas deben ser las pistas(para que entren más)
  - Para que las pistas sean estrechas los cabezales deben ser estrechos
  - Si los cabezales son muy estrechos deben estar muy próximos a la superficie para generar y sentir los campos
  - Cuanto más cercana la cabeza a la superficie más posibilidad de fallos debido a las irregularidades de la superficie

Disco duro

- Conjunto de discos magneticos (1-15)
- Posición fija
- Se escribe por ambas caras de cada uno de ellos
- Transductores para cada cara de cada disco
- Ventajas frente a los floppy
  - Mas grandes debido a que son rígidos
  - Control de los transductores más preciso
  - Razon de escritura/lectura porque giran más rápido
  - Incorporan mas de un disco
- Disco winchester
  - Disco precintado libre de contaminación
  - Cabeza muy cerca de la superficie alta densidad de datos



TIEMPOS DE ACCESO

- Para leer o escribir la cabeza debe encontrar primero la posición inicial de la operación
- Tiempo de búsqueda La selección de pista implica movimiento de cabeza
  - Fabricantes dan el máximo el mínimo y el promedio
    - » Promedio suma de todos los posibles tiempos de búsqueda /nº de ellos
- Latencia de rotación
  - Una vez alcanzado la pista se debe buscar el sector
  - Depende de la velocidad a la que giren
    - » 3600-7200 Rpm
  - Latencia promedio tiempo que se tarda en recorrer medio disco
    - » 8.3Mso 4.2 Ms
    - » Diametrtos de disco pequeños giran más rápido sin excesivo consumo
- Tiempo de acceso = tiempo de búsqueda más latencia de rotación
- Razón de transferencia
- Tiempo que se tarda en transferir un sector
- Función del tamño del sector, velocidad de rotación, desidad de grabado
- 1997 Entre 2 y 15MB/s

RAID

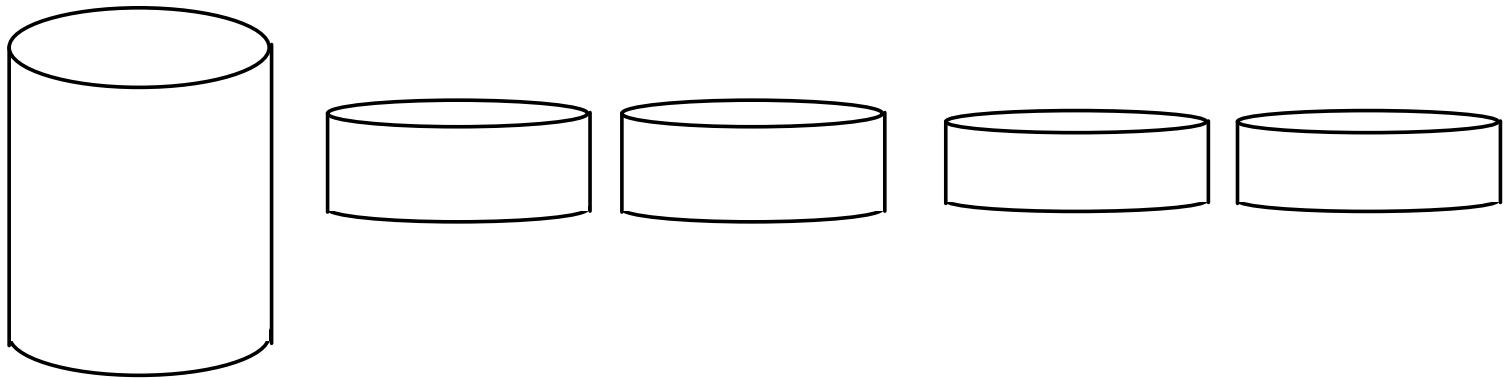
REDUNDANT ARRAY OF INDEPENDENT DISCK

- Arrays de discos que operan en paralelo e independientemente
- Paralelismo:
  - Aprovechamiento del concepto de paralelismo en la organización de discos
  - Múltiples discos permiten múltiples peticiones de entrada/salida
  - Una petición se realiza en paralelo si el bloque se distribuye en todos los discos
- Redundancia
  - Redundancia de la información para asegurar la fiabilidad del sistema
- Niveles:
  - Existen seis niveles del 0 al 5
  - No son niveles jerárquicos sino diferentes arquitectas
  - Características comunes:
  - Conjuntos de discos vistos por el sistema operativo como un único disco lógico
    - » Los datos se distribuyen a través del array de discos
    - » La capacidad redundante de los discos se utiliza para almacenar paridad de información

RAID

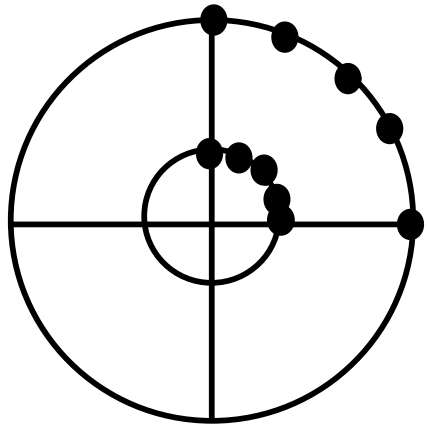
REDUNDANT ARRAY OF INDEPENDENT DISCK

- **Objetivo**
  - **Acortar distancias entre la velocidad del procesador y la velocidad electrodinámica de los discos**
- **Estrategias**
  - **Sustitución de un disco de gran tamaño por varios de tamaño menor**
  - **Distribuir datos de manera que se pueda acceder simultáneamente a varios discos mejorando el rendimiento de entrada salida,**
  - **Permite un sencillo aumento de la capacidad.**



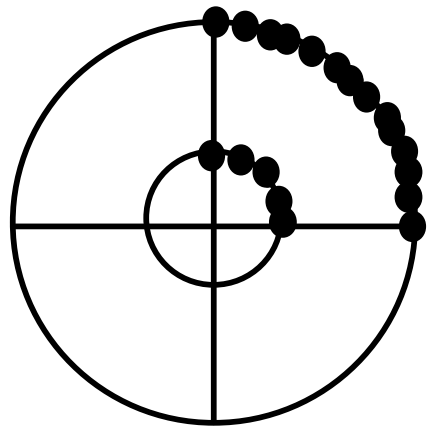
MEMORIA OPTICA
CDROM

- Material de construccion resina policarbonatos
- Recubierto de una superficie muy reflectiva como el aluminio
- Escritura
  - La información se guarda mediante pequeños agujeros en la superficie
  - Con un láser de alta densidad se crea el disco maestro
  - El resto por estampado
  - Se protege su superficie para evitar el deterioro
- Lectura
  - Mediante un láser de baja potencia
  - Recoge la variación de intensidad del rayo láser mediante un fotosensor y se convierte a señal digital
- Organización de la información
  - CAV (Constant Angular Velocity)
  - Similar a los discos magnéticos
  - Igual cantidad de información en todos los sectores
  - Velocidad de giro constante--> Vlineales diferentes
  - Poco usado desaprovecha espacio



MEMORIA OPTICA

- CLV (Constant Linear Velocity)
  - Densidad de información constante en todos los sectores
  - Hay que variar la velocidad de giro gira más lentamente en los exteriores
  - No existen varios track aislados sino uno solo en espiral
  - Capacidad aproximada 774.57Mb es decir aproximadamente 550 disquetes de 3,5
- Formato de bloque
  - SYNC.- Identifica el comienzo del bloque
  - Header.- Dirección del bloque y modos de la información
    - » Modo 0.- Sin datos
    - » Modo 1 con datos y código de error
    - » Modo 2 con datos sin código de error
- Ventajas
  - Mas información que el disco magnético
  - Más fácil de replicar la información
  - Móvil
- Desventajas
  - Solo lectura
  - Tiempos de acceso mayores que el magnético



LA MEMORIA CACHE

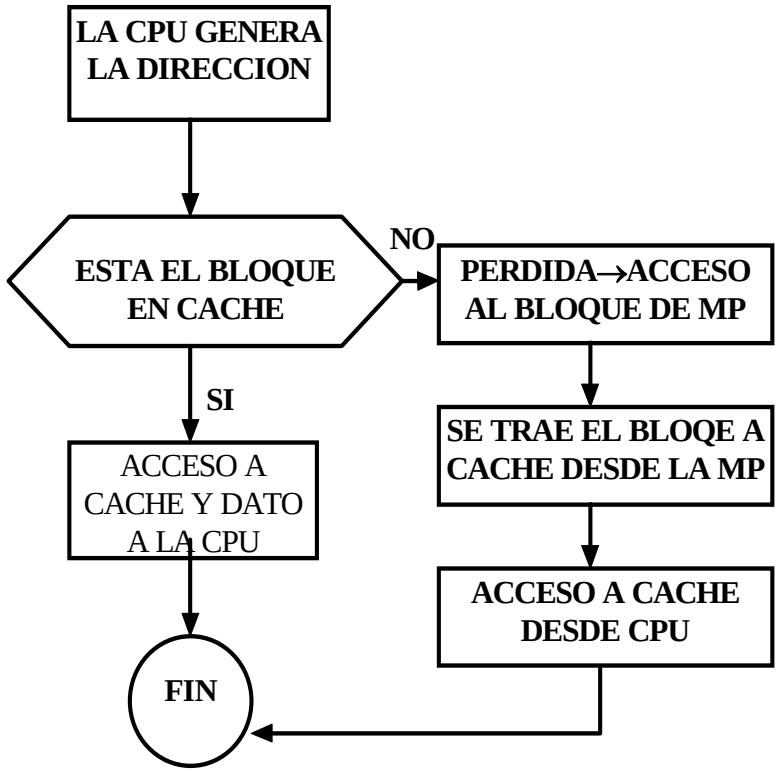
TEMA 12

INTRODUCCION

- La Memoria principal
  - Ventajas
    - » Es grande
    - » Es barata
  - Desventajas
    - » Demasiado lenta comparada con las velocidades de trabajo del Procesador
    - » Cuello de botella
- Solución
  - Proporcionar al sistema una velocidad parecida a la de la memoria más rápida
  - Proporcionar una memoria lo suficientemente grande al precio de la memorias semiconductoras más baratas
- Memoria cache
  - Memoria rápida y pequeña situada entre el procesador y la MP que almacena la información actualmente en uso.
- Primer computador IBM 360/85 (1969)
- Se basa en el principio de Localidad de referencia

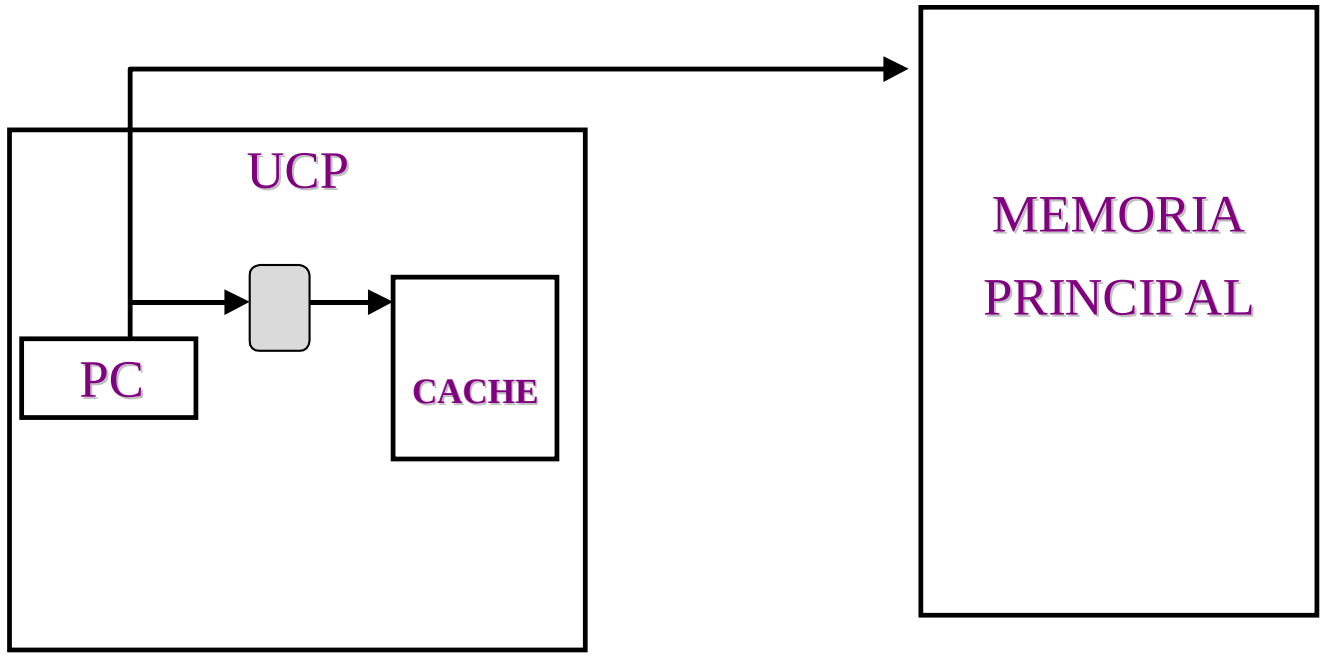
MODO DE OPERACIÓN

- La MP y la cache se dividen en bloque de igual tamaño
- En un T cualquiera un subconjunto de bloques de MP reside en cache
- Cuando la CPU intenta acceder a la información de MP comprueba si está en la cache
  - si está trabaja con el dato de la cache
  - Si no está lee un bloque de MP, Y lo copia en un la cache



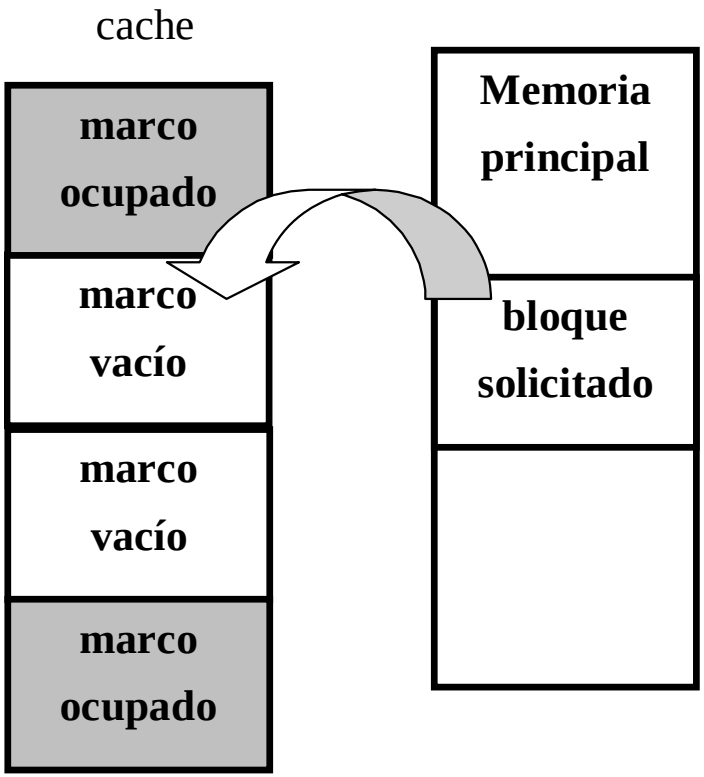
DIR CACHE VS DIR PRINCIPAL

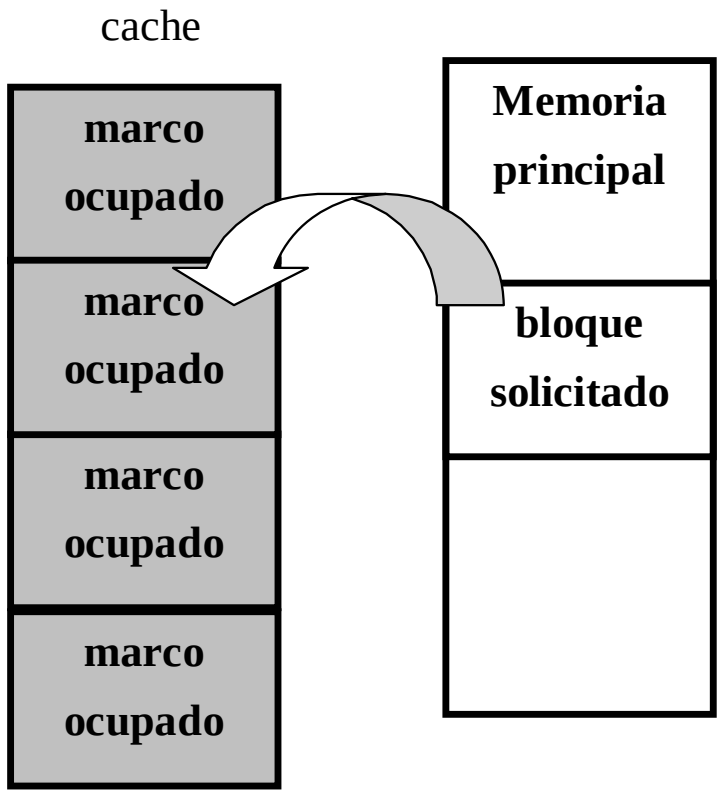
- **Problema:**
  - **la cache < principal --> un slot no puede ser ocupado permanentemente por un bloque**
- **Solución**
  - **Un slot incluye una etiqueta (TAG) que identifica el bloque que lo ocupa en el instante T**
  - **Como etiqueta se utilizan los bits más significativos de la dirección de memoria principal**



ELEMENTOS BÁSICOS DE DISEÑO

- politica de emplazamiento
- politicas de reemplazamiento
- politicas de actualización





POLITICAS DE EMPLAZAMIENTO

- Como la memoria principal es bastante mayor que la memoria cache hay que decidir donde se ubica un bloque de memoria principal en la memoria cache

- Existen tres políticas de emplazamiento:
  - Directa
  - Asociativa
  - Asociativa por conjuntos

- la elección de una política de emplazamiento condiciona la organización de la memoria cache

POLITICAS DE EMPLAZAMIENTO DIRECTO

MARCOS DE BLOQUE DE LA CACHE M =2^R

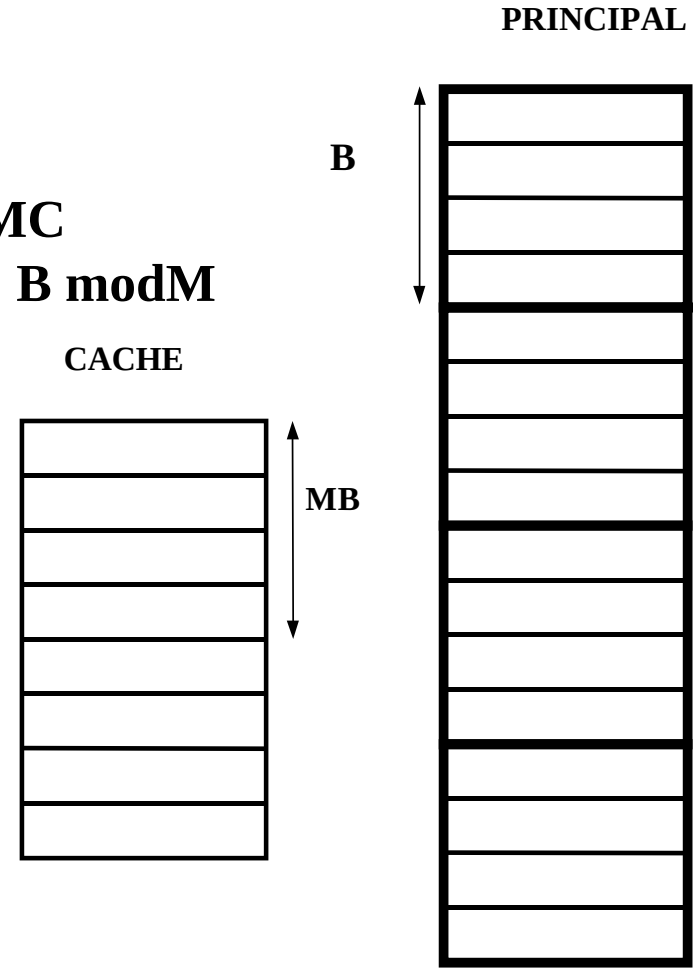
BLOQUE DE LA PRINCIPAL 2^s

PALABRAS POR BLOQUE 2^W

UN bloque de MP sólo puede ocupar un MB de la MC
El marco de bloque que ocupa un bloque B es MB= B modM

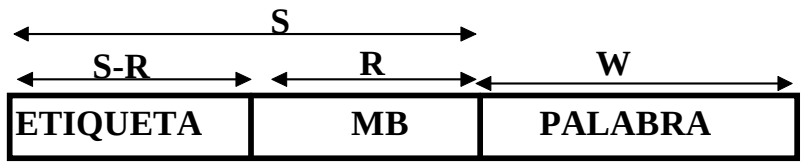
M=8
B=2--> MB=2
B=8-->MB=0
B=9--> MB=1
B=17-->MB=1
B=25-->MB=1

¿Como se cual es el B que ocupa el MB?



POLITICA DE EMPLAZAMIENTO DIRECTA

- **Implementación:**
  - **la dirección física que proporciona el procesador se interpreta de la siguiente manera**
    - » **MB.- Indica el número de bloque de la cache que ocupa el bloque**
    - » **PALABRA.- selecciona una palabra de las varias que tiene el bloque**
    - » **ETIQUETA .- se utiliza para comprobar que el marco de bloque esta ocupado por el bloque de MP que se desea acceder.**
  - **El Número de bloques de la cache es $M=2^R$**
  - **El número de bloques de la memoria principal 2^S**
  - **Bit de validez:**
    - » **Bit asociado a la etiqueta**
    - » **Sirve para indicar si la información contenida en el bloque es válida o no.**
    - » **Cuando un procesador arranca, la cache estará vacía y por lo tanto los campos de etiqueta no tendrán sentido.**



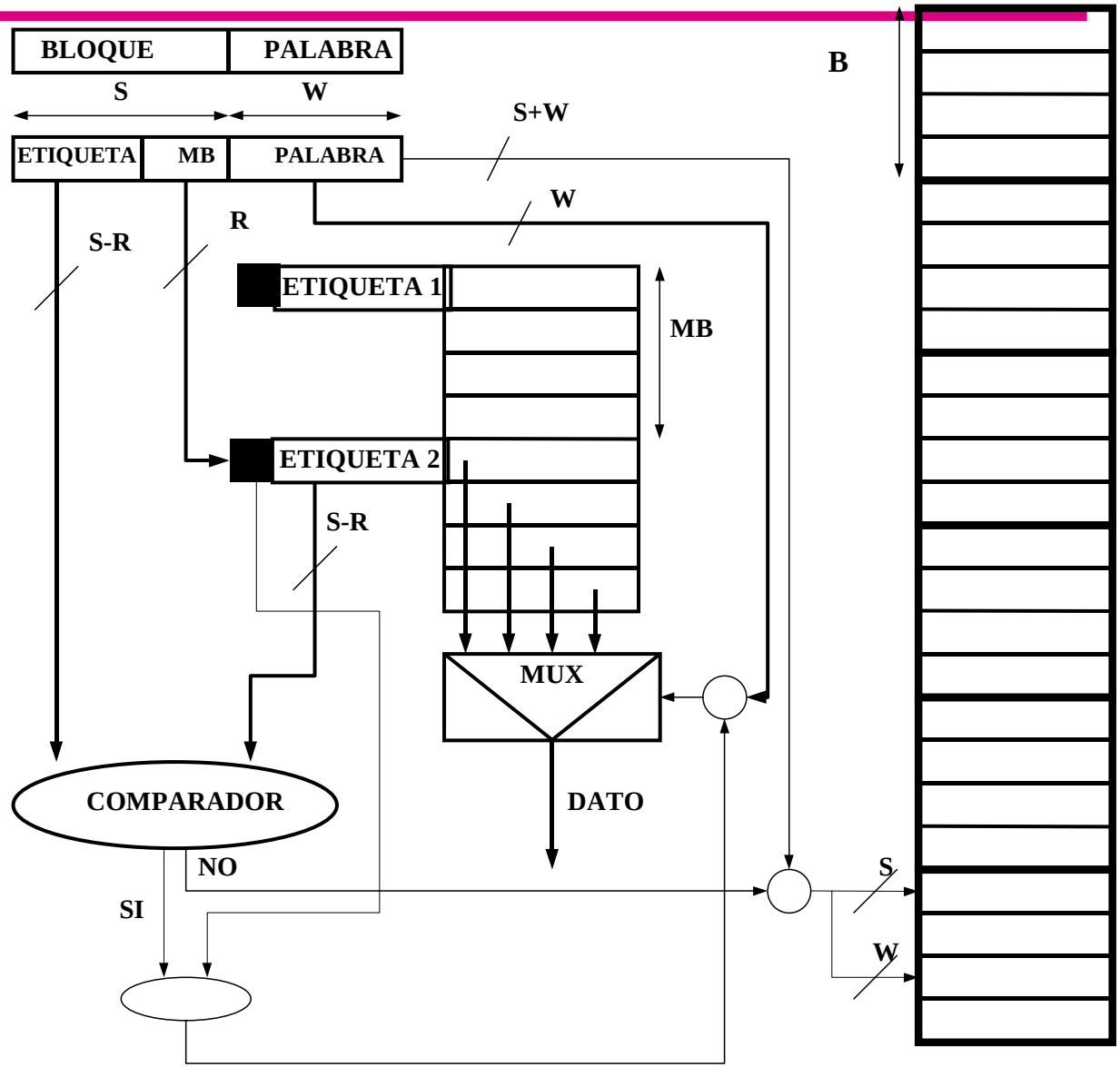
POLITICA DE EMPLAZAMIENTO DIRECTA

- **La forma de actuar es la siguiente:**
  - **Se identifica el bloque con MB accediendo de manera aleatoria**
  - **Se comprueba que es bloque deseado comparando la ETIQUETA de la dirección con la ETIQUETA de la memoria mediante un mecanismo similar al de una memoria asociativa**
    - **Si coinciden la etiquetas se realiza el acceso**
    - **Si no coinciden perdida de cache**
  - **Acceso aleatorio+comparación +lectura**

POLITICA DE EMPLAZAMIENTO DIRECTA

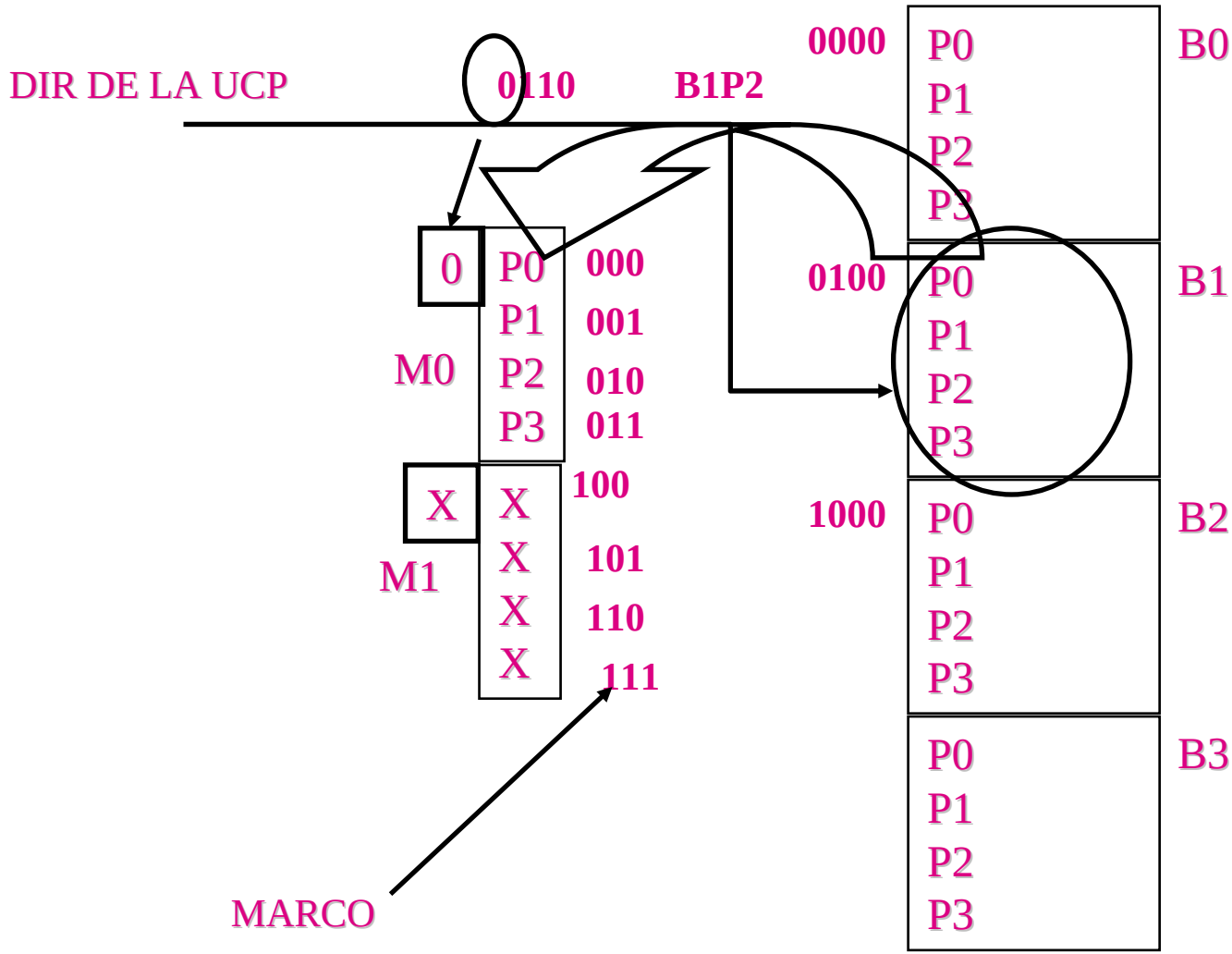
MP DE 16 PALABRAS
BLOQUES DE 4 PALABRAS
MP CON 4 BLOQUES
MC CON 2 MARCOS

0000	B0 P0	MB0
0001		MB0
0010		MB0
0011	B0 P3	MB0
0100	B1 P0	MB1
0110		MB1
0111	B1 P3	MB1
1000	B2 P0	MB1
1001		MB0
1010		MB0
1011	B2 P3	MB0
1100	B3 P0	MB0
1101		MB1
1110		MB1
1111	B3 P3	MB1



MP DE 16 PALABRAS
BLOQUES DE 4 PALABRAS
MP CON 4 BLOQUES
MC CON 2 MARCOS

0000	B0 P0	MB0
0001		MB0
0010		MB0
0011	B0 P3	MB0
0100	B1 P0	MB1
0110		MB1
0111	B1 P3	MB1
1000	B2 P0	MB1
1001		MB0
1010		MB0
1011	B2 P3	MB0
1100	B3 P0	MB0
1101		MB1
1110		MB1
1111	B3 P3	MB1



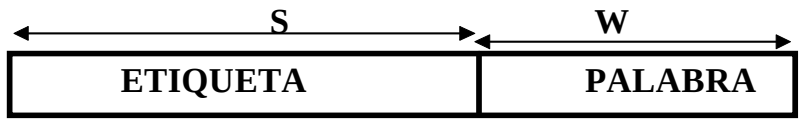
POLITICA DE EMPLAZAMIENTO DIRECTA

- **Ventajas:**
  - **Baja complejidad hardware**
  - **Barato**
  - **Alta velocidad de operación es decir el tiempo de cache es bajo**
  - **No necesita algoritmos de reemplazamiento**
  - **Poca sobrecarga de memoria --> etiquetas pequeñas**

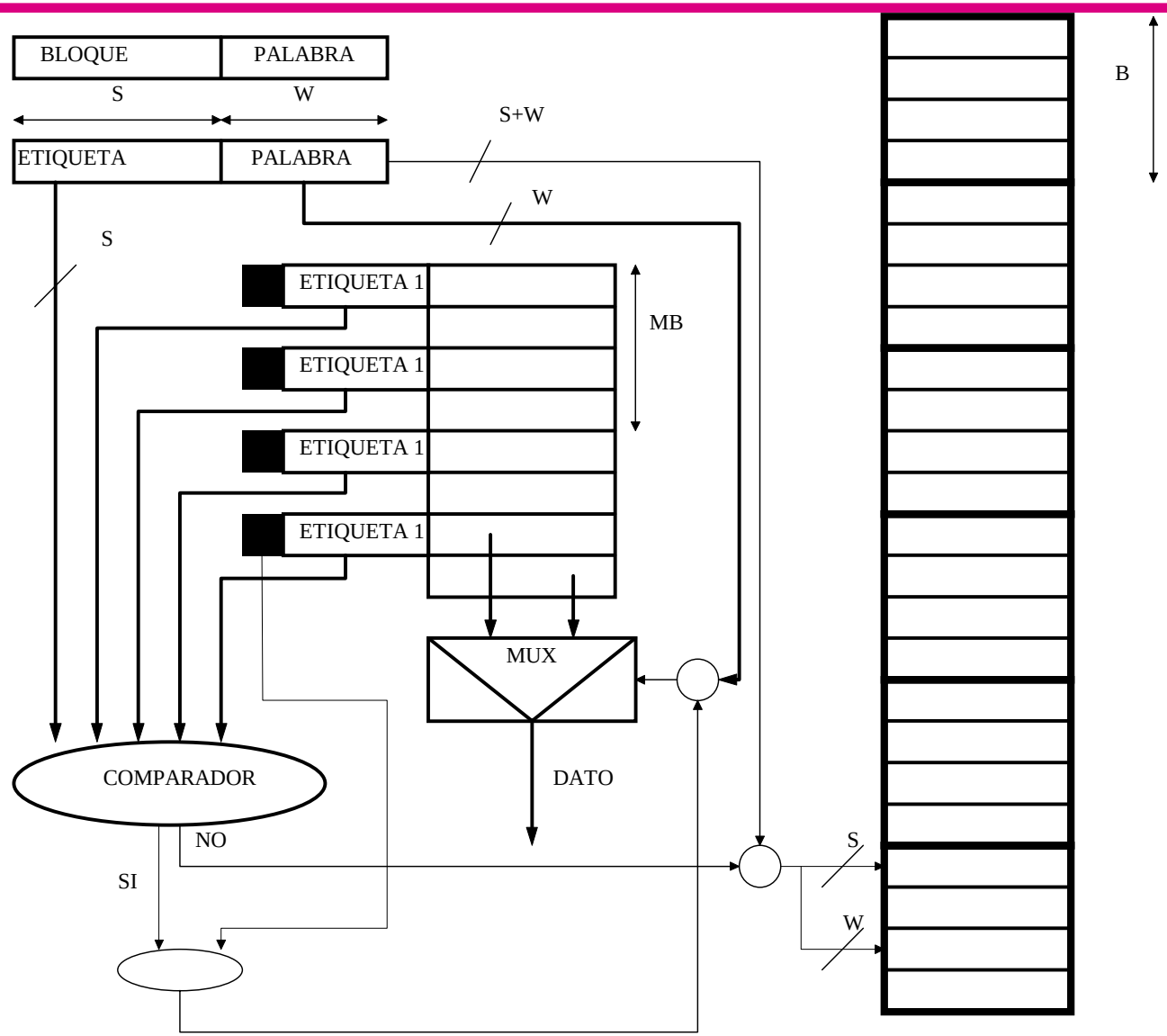
- **Desventajas**
  - **referencias repetidas de dos bloques diferentes a los que les corresponde el mismo marco, estos bloques tendrán que estar permanentemente saliendo de la cache**
    - » **tasa de fallos elevada**
  - **No se puede utilizar para solapar la traducción de direcciones virtuales**

POLÍTICA DE EMPLAZAMIENTO ASOCIATIVA

- Un bloque de la principal puede situarse en cualquier marco de bloque de la cache
- La cache interpreta la dirección de memoria de la siguiente manera
  - La etiqueta de la memoria cache contiene todos los bits que identifican al bloque de la memoria cache
- Se elimina el paso de identificación del marco aleatoria que tenía la ubicación directa
  - COMPARCION+LECTURA
- Para determinar si un bloque esta en la memoria cache su lógica debe comprobar simultáneamente las etiquetas de todos los bloques de memoria con la etiqueta de la dirección
  - Se podría ir comparando una a una
    - » Muy bajo rendimiento
    - » Más barato



POLITICA DE EMPLAZAMIENTO ASOCIATIVA



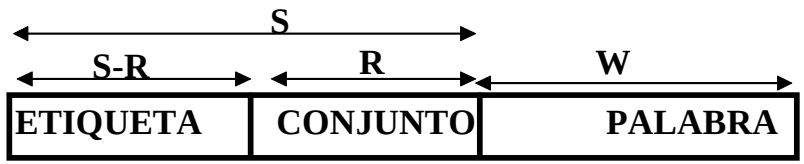
POLITICA DE EMPLAZAMIENTO ASOCIATIVA

- **Ventajas:**
  - Mas flexible que la ubicación directa
  - permite la libre elección del bloque que se reemplaza para introducir otro
    - » Tasa de fallos baja

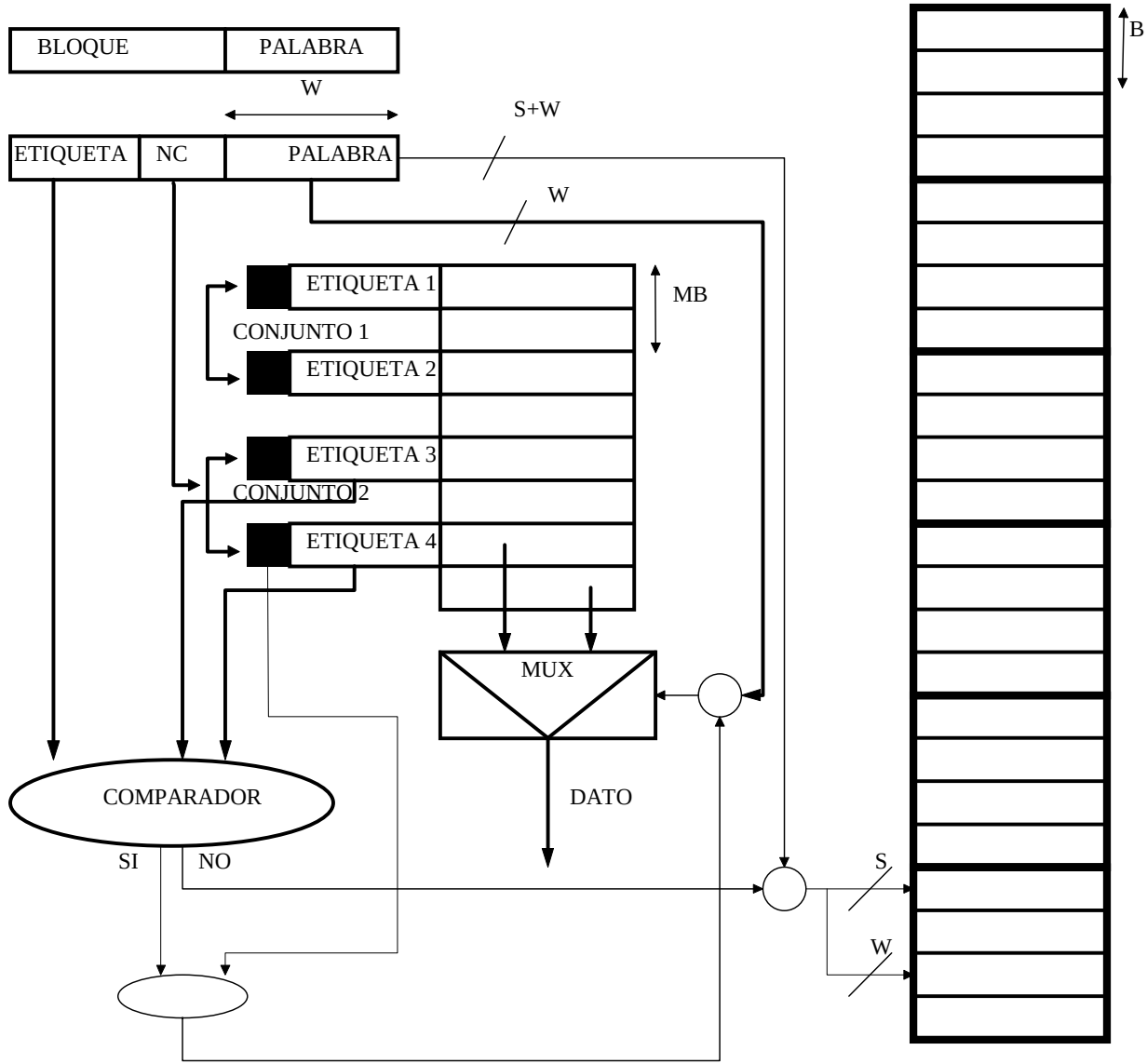
- **Desventajas**
  - Complejidad del circuito hw --> circuito comparador
  - La rapidez de identificación disminuye
    - » acceso asociativo más lento que el aleatorio
  - Sobrecarga de memoria:
    - » etiquetas muy largas que ocupan espacio de memoria
  - Más cara
  - Necesita algoritmos de reemplazamiento
    - » se deben implementar en HW para mantener la alta velocidad de operación

POLITICA DE EMPLAZAMIENTO ASOCIATIVA POR CONJUNTOS

- Es un compromiso entre el emplazamiento directo y el asociativo
- La memoria cache se divide en conjuntos de marcos
- Un bloque de memoria principal
  - se ubica siempre en el mismo conjunto de la cache
    - » dentro del conjunto se ubica en cualquiera de sus marcos
- Una memoria asociativa por conjuntos de N vías es aquella en la que cada conjunto contiene N marcos
- suponiendo
  - la memoria cache dividida en C conjuntos
  - la memoria principal en B bloques
  - el conjunto en el que se coloca el bloque es :
  - NC=MmodC
- La dirección de memoria física se interpreta de la siguiente manera:



POLITICA DE EMPLAZAMIENTO ASOCIATIVA POR CONJUNTOS



POLITICA DE EMPLAZAMIENTO

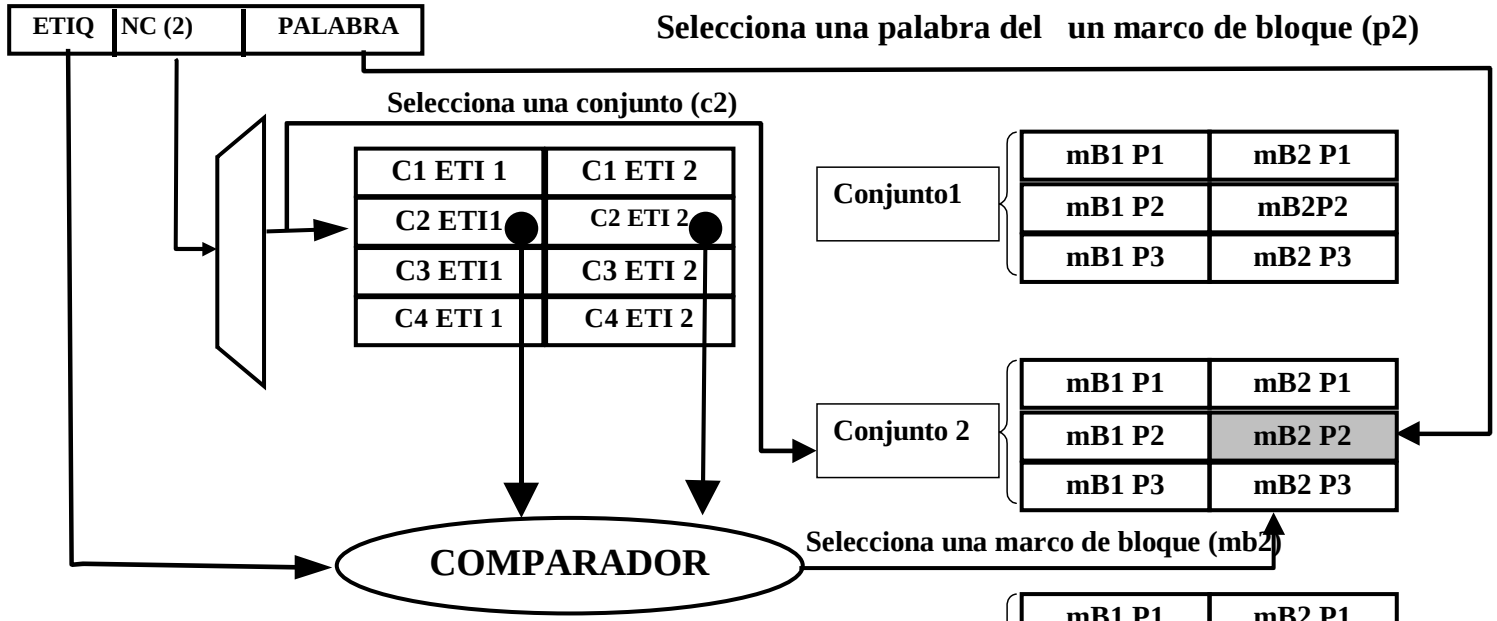
ASOCIATIVA POR CONJUNTOS

- **Complejidad hardware mediana**
  - **comparador menos complejo que la asociativa**
- **Rapidez en la identificación media**
- **Sobrecarga de memoria media**
- **Tasa de fallos baja**
- **Grado de asociatividad E=nºmarcos/nº conjuntos =M/C**
  - **coincide con el nº de vias**
  - **cuando M=C , E=1 emplazamiento directo**
  - **cuando C=1, E=M emplazamiento asociativo**

POLITICA DE EMPLAZAMIENTO ASOCIATIVA POR CONJUNTOS

- El número de comparadores viene dado por el numero de bloques de cada conjunto
  - el número de vías
- Paralelismo de acceso
  - En una memoria totalmente asociativa cada etiqueta necesita compararse antes de poder realizar el acceso
  - Una memoria RAM con las etiquetas de cada conjunto
  - Una memoria RAM por cada conjunto
    - » se accede en paralelo a cada bloque del conjunto
  - se puede
    - » acceder a la palabra de todos los bloque
    - » al tiempo que se comprueban las etiquetas
  - Cuando la etiqueta ha sido identificada , el bloque correspondiente se lee
  - En la escritura este acceso paralelo no se puede llevar a cabo

POLITICA DE EMPLAZAMIENTO ASOCIATIVA POR CONJUNTOS

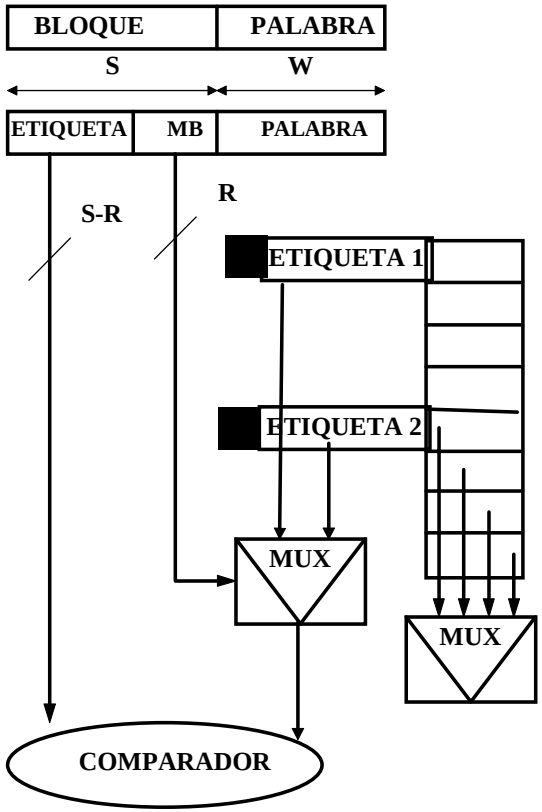


CONJUNTOS DE 2 MARCOS
MARCOS DE 3 PALABRAS

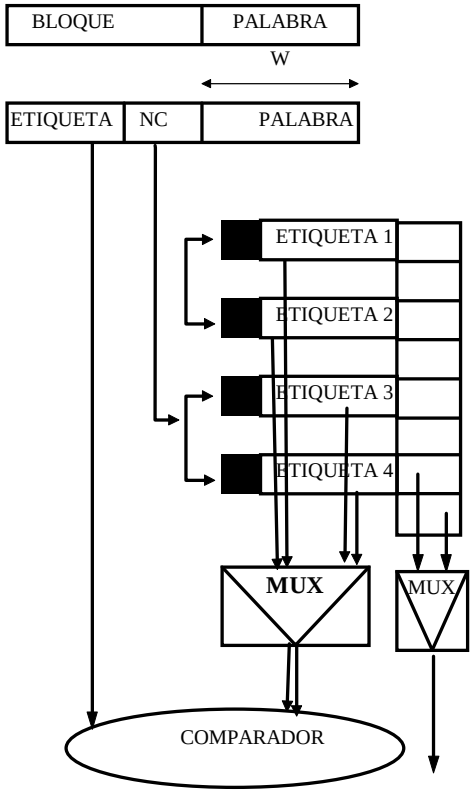
En paralelo la lectura de las palabras i de un conjunto j y la comprobación asociativa de las etiquetas del conjunto i

EMPLAZAMIENTOS

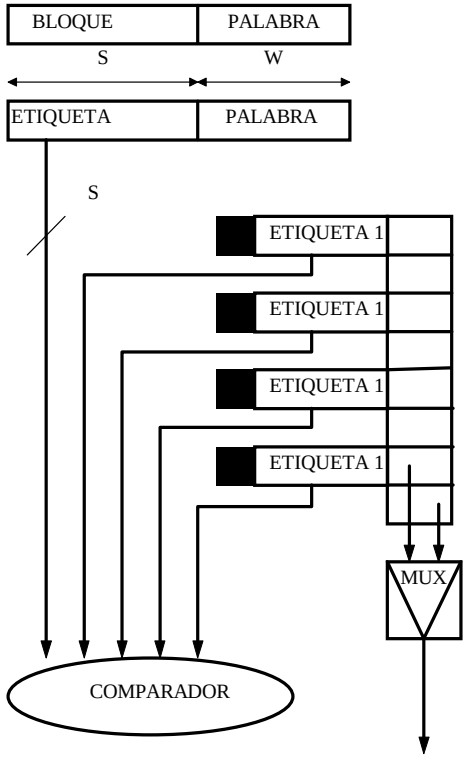
DIRECTO



ASOCIATIVO POR CONJUNTOS



ASOCIATIVO



TIPOS DE FALLO

- **Fallos iniciales (compulsor)**
  - Primer acceso a un bloque
  - Lógicamente no está en la cache todavía
  - También se llama de arranque en frío o de primera referencia
- **Fallos de capacidad (capacity)**
  - Debido a que la cache no puede conservar todos los bloques que se usan en un programa
  - Falta de localidad temporal
- **Fallos de conflicto (conflict)**
  - Debidos a la competencia por los marcos del conjunto en
  - Emplazamientos asociativo por conjuntos
  - Directo

PARÁMETROS DE DISEÑO Y RENDIMIENTO

- **Los parámetros de diseño de una memoria cache son:**
  - **El grado de asociatividad E=M/C**
  - **El tamaño de la memoria cache**
  - **El tamaño de los bloques**
- **Estos parámetros de diseño influyen en el**
  - **El parámetro de rendimiento**
  - **Coste HW**
- **EL PARÁMETRO DE RENDIMIENTO**
  - **T=Tcache+Tfallos*P**
    - » **Tcache tiempo de acceso de la cache que depende de la tecnología**
    - » **Tfallos.- Depende de la captura de la localidad**
    - » **P penalización el tiempo de acceso al siguiente nivel de la jerarquía**
- **El coste HW depende de la tecnología.**

PARÁMETROS DE DISEÑO Y RENDIMIENTO
GRADO DE ASOCIATIVIDAD

- Grado de asociatividad
  - E=nº de marcos/nº de conjuntos
    - » E=1--> directa
    - » E=nº marcos-->asociativa

- A mayor grado de asociatividad
  - Menor tasa de fallos
    - » los bloques no necesitan competir por el mismo marco
    - » politicas de reemplazamiento
  - Mayor tiempo de acceso a la cache
    - » más lentos los circuitos comparadores
  - Mayor coste de hardware (crece el número de comparadores)
- En general si la memoria cache es grande (128kb) las mejoras son poco notables
- la mejora es mayor al principio que al final

PARÁMETROS DE DISEÑO Y RENDIMIENTO
TAMAÑO DEL BLOQUE

- Inicialmente al crecer el tamaño del bloque (para tamaño de memoria cte)
  - Disminuye la tase de fallos debido a que se captura mejor la localidad espacial
- Al seguir aumentando
  - Vuelve a aumentar la tasa de fallos
    - » El número de bloque es menor y se producirá una gran competencia por esos bloques
      - Un bloque saldrá de la cache antes de que sean accedidas muchas de sus palabras-perdida de la localidad temporal-
  - Aumenta la penalización de un fallo
    - » La penalización de los fallos está determinada por el tiempo de búsqueda en el nivel inferior
      - tiene dos partes la latencia para la primera palabra
      - el tiempo de transferencia del bloque
        - Cuanto mayor es el bloque mayor es su tiempo de transferencia

PARÁMETROS DE DISEÑO Y RENDIMIENTO
TAMAÑO DEL BLOQUE

- **Solución**
  - **Early start o arranque anticipado:**
    - » **Reanudar la ejecución tan pronto como llegue la palabra pedida del bloque en lugar de esperara al bloque completo.**
    - » **Este mecanismo funciona mejor con instrucciones debido a que su carga es mas secuencial**
  - **Requested word first**
    - » **Se organiza la memoria para que la palabra necesitada sea siempre la primera**
    - » **A continuación se transfiere el resto del bloque y se vuelve al principio del bloque.**
  - **Memorias entrelazadas**

PARÁMETROS DE DISEÑO Y RENDIMIENTO

TAMAÑO DE LA MEMORIA CACHE

- Tasa de fallos disminuye con el tamaño de la memoria
  - Se captura mejor la localidad
- Disminución de la tasa de fallos de conflicto
  - Aumenta el número de conjuntos
- Disminución de la tasa de fallos de capacidad
  - Aumenta el tamaño de la memoria
- Llega un momento en que la mejora que se consigue no compensa con el precio del hw

PARÁMETROS DE DISEÑO Y RENDIMIENTO
CONCLUSIONES

- Grado de asociatividad
  - tasa de fallos
  - coste hw
- Tamaño de bloque
  - tasa de fallos
  - penalización
- Tamaño de la memoria cache
  - tasa de fallos
- Integración en el un CI de
  - procesador
  - cache
- limita el tamañode la cache y reduce los t acceso
- Tamaño de bloque pequeño y en aumento (4-6)
- Grado de asociatividad E pequeño y en disminución

POLITICAS DE REEMPLAZAMIENTO

- Cuando se intenta acceder a una palabra y esta no se encuentra en la caches, es necesario trasladar todo el bloque que la contiene a la cache.
- En el caso que no exista espacio en la memoria se debe seleccionar el bloque de la memoria cache que se quiere eliminar para introducir el nuevo.
- Estas politicas deben cumplir los siguientes requisitos:
  - Deben implementarse totalmente en Hardware
  - Se debe intentar que la selección se realice en el ciclo de memoria principal , durante el cual se está trayendo el nuevo bloque
  - Se debe intentar que el bloque reemplazado no se tenga que utilizar en el futuro.
    - » Este último requisito es el más difícil de cumplir puesto que no se conoce el comportamiento futuro de los programas

POLITICAS DE REEMPLAZAMIENTO

- **El espacio de reemplazamiento**
  - **Emplazamiento directo:**
    - » **Cada bloque tiene asignado su marco**
  - **Asociativo :**
    - » **Toda la cache**
  - **Asociativo por conjuntos:**
    - » **Los marcos del conjunto**

- **Algoritmos típicos son**
  - **FIFO**
  - **LFU (menos frecuentemente usado)**
  - **aleatorio**
  - **LRU**

POLITICAS DE REEMPLAZAMIENTO

- **FIFO**
  - Primero en entrar primero en salir
  - Se reemplaza el bloque del conjunto que lleve más tiempo
  - Round robin o técnica del buffer circular

- **MENOS FRECUENTEMENTE USADO**
  - Reemplaza el bloque menos referenciado
  - Se puede implementar asociado un contador a cada bloque

- **Estos dos no se usan demasiado costosos para los resultados que dan**

- **ALEATORIO**
  - El bloque a reemplazar se escoge aleatoriamente
  - Buenos resultados
  - Poco costosos

POLITICAS DE REEMPLAZAMIENTO

MENOS RECIENTEMENTE USADO (LRU)

- Se reemplaza el bloque menos recientemente usado
- Para conjuntos asociativos de dos bloque es fácil de implementar
  - Cada bloque incluye un bit de uso
  - Cuando un bloque se referencia su bit se pone a uno y el del otro bloque a cero
  - Se reemplaza el bloque que tiene su bit a cero
- Cuando el número de bloque es mayor en lugar de un bit se utiliza un contador llamado registro de edad.
  - Si se referencia el bloque j del conjunto el contador de j se pone a cero
  - El contador del resto de los bloques que tenían un valor inferior a j se incrementa.
  - Los contadores con valor superior no se incrementan
  - Se reemplaza el bloque que tenga el valor mayor.
    - » su contador se pone a cero para inicializar el nuevo bloque
    - » Se incrementa el resto en uno

POLITICAS DE REEMPLAZAMIENTO
CONCLUSIONES

- La selección aleatoria fácil de construir en HW mayor tasa de fallos
- LRU menor tasa de fallos
  - Se captura la localidad espacial
  - Problema cuando el nº de bloques aumenta la estrategia se hace muy cara
  - Para grado mayor que 4 costoso en tiempo y almacenamiento (Tactualización>tcache)
- LRU vs aleatoria
  - La diferencia en tasa de fallos es muy pequeña
    - » En una cache asociativa por conjunto de dos vías la tasa de fallos del algoritmo aleatorio es 1,1 veces mayor que el LRU
  - La diferencia crece con el grado de asociatividad
  - Disminuye con el aumentodel tamaño de la cache
- Comparamos con las políticas de emplazamiento el algoritmo de reemplazamiento tiene una influencia secundaria sobre el rendimiento del sistema,
    - » La cache totalmente asociativa es la mas sensible a los algoritmos de reemplazamiento

POLITICAS DE ACTUALIZACIÓN

- **las operaciones de lectura predominan sobre las de escritura.**
  - **Todos los accesos a instrucciones son lecturas**
  - **la mayoría de las instrucciones no escriben en memoria**
  - **Siguiendo el consejo de hacer siempre más rápido el caso común se deben optimizar las caches para mejorar las lecturas.**

- **ley Amdhal**
  - **el tiempo de mejora es proporcional al tiempo de ejecución no afectado por la mejora**
  - **no se deben despreciar las mejoras los accesos de escritura**

POLITICAS DE ACTUALIZACIÓN

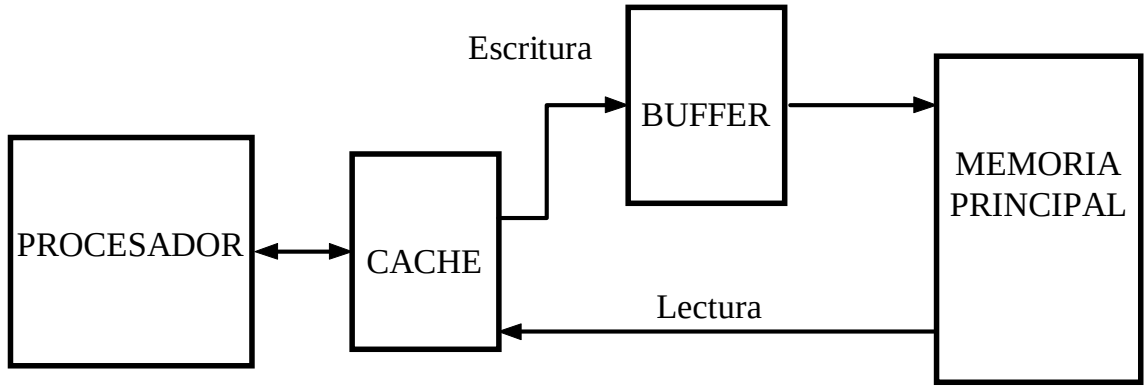
- **Deben tener en cuenta:**
  - La inconsistencia-- aciertos de escritura
  - sobreescritura de la cache-- fallos de lectura
  - Fallos de escritura
- **Inconsistencia:**
  - Si en una escritura solo modificamos el dato en la cache
    - » la memoria principal cache y tendrán datos diferentes.
  - Este problema se puede tratar mediante dos políticas diferentes:
    - » Escritura directa
    - » Postescritura
- **sobreescritura**
  - en función de la politica se machaca o no la información de la cache
- **Fallo de escritura**
  - Cuando se quiere actualizar un dato que no está en memoria cache.
  - Pueden tener dos estrategias diferentes:
    - » Con asignación de marco
    - » Sin asignación de marco

POLITICAS DE ACTUALIZACIÓN
ESCRITURA DIRECTA

- **Acierto de escritura:**
  - el dato a modificar está en cache
  - siempre se actualiza el dato simultáneamente
    - » la memoria principal
    - » la memoria cache
  - Este esquema lo utiliza la DECSTATION 31000
- **Fallos de lectura:**
  - supone un reemplazamiento
  - Si el bloque de cache no se ha modificado, ese bloque se puede sobreescribir con en nuevo bloque
  - Si el bloque ha sido modificado se puede sobreescribir porque tambien se ha modificado en MP
- **Ventaja**
  - No existe inconsistencia nunca
  - Manejo muy sencillo
  - Fallos de lectura son muy baratos porque nunca necesitan escrituras al nivel inferior
- **Desventaja:**
  - No produce buenos rendimientos en las escrituras debido a los accesos a MPrincipal

POLITICAS DE ACTUALIZACIÓN
ESCRITURA DIRECTA

- **Solución**
  - **Añadir un buffer de escritura que almacena el dato mientras que este espera ser escrito en MP**
  - **Una vez escrito el dato en la cache y en el buffer el procesador puede seguir la ejecución**
  - **El buffer de escritura puede acomodar un número fijo de palabras (1-10)**
  - **Si el buffer está lleno cuando se realiza una escritura desde el procesador el procesador debe detenerse hasta que haya una posición vacía.**
    - » **Si la frecuencia a que la memoria completa escrituras es inferior a la que las genera el procesador el buffer no sirve de nada.**



POLITICAS DE ACTUALIZACIÓN post escritura

- **Acierto de escritura:**
  - **Actualiza sólo en memoria cache**
  - **El bloque modificado se escribe solo en el nivel inferior cuando es sustituido, es decir cuando se produce un fallo**
- **Fallo de lectura**
  - **Cuando el bloque de cache a reemplazar no se ha modificado, ese bloque se puede sobreescribir sin problemas**
  - **Si el bloque se ha modificado antes de reemplazarlo hay que escribirlo en la mp**
  - **Para distinguir si se ha modificado o no utiliza un bit “poluto”(dirty)**
- **Ventajas:**
  - **Las palabras individuales las escribe el procesador a la velocidad de la cache**
  - **Múltiples escrituras en un bloque requieren sólo una escritura en el nivel más bajo de jerarquía**
  - **Se hace mejor uso del ancho de banda de comunicación con el nivel inferior de jerarquía**

POLITICAS DE ACTUALIZACIÓN
FALLOS DE ESCRITURA

- **Con asignación de marco**
  - **Cuando se produce el fallo**
  - **Se trae el bloque de la memoria principal al la cache**
  - **A continuación se actúa como si hubiera habido un acierto**
- **Sin asignación de marco el bloque**
  - **Se modifica directamente en la memoria principal sin cargarlo en la cache**
- **Ambas estrategias se podría aplicar a**
  - **la escritura directa**
  - **a la postescritura,**
- **pero en la práctica no es así.**
  - **La post escritura utiliza la asignación de marcos**
    - » **esperan que posteriores escrituras puedan ser capturadas por la cache**
  - **La escritura directa utiliza la no asignación de marco,**
    - » **en cualquier caso las siguientes escrituras también se llevaran inmediatamente a la memoria principal.**

PARÁMETROS DE DISEÑO Y RENDIMIENTO

- Para describir el efecto de las políticas de escritura en los tiempos medios de acceso a memoria vamos a suponer:
  - E: fracción de escrituras del sistema
  - Tc tiempo de ciclo de cache
  - Tm el tiempo de ciclo de memoria
  - Tb el tiempo de transferencia de bloque
- Sabiendo además que tc<tm

PARÁMETROS DE DISEÑO Y RENDIMIENTO

ESCRITURA DIRECTA CON ASIGNACIÓN DE MARCO

- El tiempo medio para completar una referencia cuando no hay almacenamiento intermedio viene dado por la siguiente expresión:

$$Tc + E \cdot (Tm - Tc) + Tfallos \cdot P$$

- Siendo
  - E la fracción de accesos de escritura (accesos conseguidos)
  - P la penalización
- Es una suma de tiempos promedios.
  - Tc el tiempo promedio de acceso a cache
  - E·(Tm-Tc) el tiempo promedio debido a las escrituras acertadas:
    - » Como la memoria principal y la cache se escriben simultáneamente el promedio se halla multiplicando la fracción de accesos de escritura por el tiempo total
    - » Tm-Tc es el tiempo adicional para escribir una palabra en la memoria principal.
    - » No se puede iniciar ninguna otra operación hasta que se ha acabado de escribir en la memoria principal
  - Tfallos·P hace referencia al tiempo que tarda en traerse un bloque a la memoria cache en el caso en que se produce un fallo de escritura o de lectura.

PARÁMETROS DE DISEÑO Y RENDIMIENTO

ESCRITURA DIRECTA SIN ASIGNACIÓN DE MARCO

- Tc+E·(Tm-Tc)+Tfallos·(1-E)·P
- El último termino difiere porque cuando hay fallos de escritura el bloque no se carga en memoria. En este caso se penaliza solo la tasa de fallos.
- (1-E) accesos que son lecturas
- Tfallo·(1-E) indica la tasa de fallos de lectura
  - son los únicos casos en los que hay penalización
    - » son los únicos casos en que se traen bloques a la cache

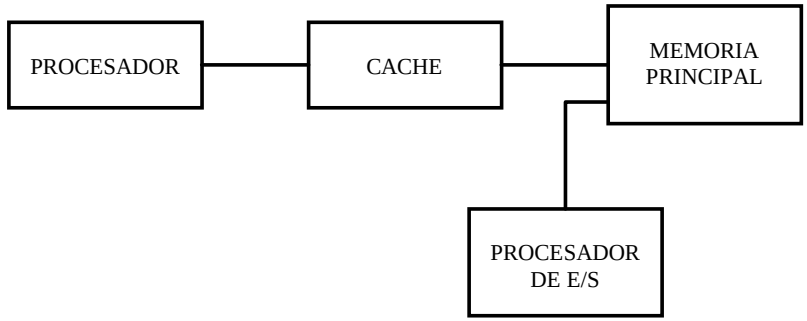
PARÁMETROS DE DISEÑO Y RENDIMIENTO

POST ESCRITURA CON ASIGNACIÓN DE MARCO DE BLOQUE

- En la postescritura
  - el bloque se escribe en memoria principal solo cuando se produce un fallo
  - cada vez que se produce un fallo se trae un bloque de memoria luego:
- Tc+Tfallos·P+Tfallos·P
- Un de los términos se debe a la escritura y el otro a la lectura, su`poniendolos iguales quedan
  - Tc+2Tfallos·P
- Esta política de escritura se podía mejorar utilizando un bit dirty
  - Tc+ Tfallos·P+Tfallos·P·Eb
    - » Donde Eb es la probabilidad de que un bloque reemplazado haya sido modificado.

coherencia entre la mp y la cache

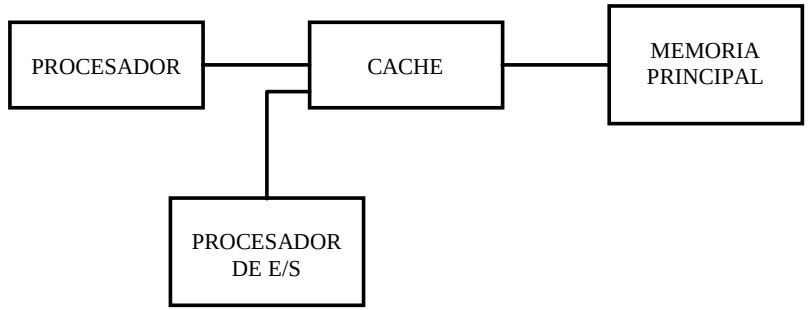
- **El que existan instantes de tiempo en los que el contenido de la memoria principal y el contenido de la memoria cache sean diferentes puede ocasionar problemas..**
- **UN PROCESADOR Y UNA CACHE**
- **Supongamos un controlador de DMA conectado directamente a la memoria principal:**
- **Suposiciones:**
- **El procesador de entrada salida escribe un dato directamente en la memoria principal. (Independientemente de la política de escritura aparece una inconsistencia entre la memoria principal y la cache.)**
- **una lectura directa de la memoria principal**
- **Inconsistencias:**
- **Con la política de escritura directa no**



- **Solución**
- conectar los procesadores de entrada salida directamente a la memoria cache

coherencia cache

- **Solución**
- **conectar los procesadores de entrada salida directamente a la memoria cache**
- **Desventaja:**
- **El inmenso trafico de entrada salida**
- **Las necesidades de sincronización de los dos puertos de la caches y del arbitraje producen importantes degradaciones del rendimiento.**



- **Otra solución:**
- La idea clave es asegurar que el procesador lea siempre el dato correcto y que cada dato escrito por el procesador de entrada/salida esté disponible en el procesador .
- Si el procesador de ES escribe un dato en la MP, este debe comprobar también si está en la cache.
- En el caso que dicho dato este en la cache debe invalidarlo de manera que si el procesador quiere leerlo, se deba recuperar de la memoria principal.
- Cuando el procesador de entrada/salida quiere realizar una lectura, comprueba si el dato ha sido modificado en la cache, en caso que así sea se escribe el dato correcto en la MP y se lee.

coherencia cache

- **VARIOS PROCESADORES CADA UNO CON SU CACHE**
- **Por otro lado en una estructura en al que varios procesadores ,cada uno de ellos con su cache, comparten bus y una memoria principal, aparece también el problema de la coherencia cache.**
- **Si se modifican datos en una cache, se invalida no solo la palabra correspondiente de la memoria principal, sino también la misma palabra en otras caches.. Incluso si se utiliza una política de escritura inmediata, el resto de las caches pueden contener datos no validos.**
- **Soluciones:**
- **Vigilancia del bus en escritura inmediata**
- **(bus watching with write through).(sloop bus o bus watcher)-**
- **Apropiado para sistemas con un solo bus.**
- **Cada controlador de cache comprueba las líneas de direcciones para detectar operaciones de escritura en memoria por parte de otros maestros del bus.**
- **Si otro maestro escribe en una posición de memoria compartida que también reside en la memoria cache, el controlador invalida la memoria cache.**

coherencia cache

- **Transparencia hardware**
- **se utiliza circuitería especial para asegurarse que todas las actualizaciones de memoria principal, vía cache, quedan reflejadas en todas las caches.**

- **Memoria excluida de cache - Not cacheable memory**
- **Solo una porción de memoria principal se comparte por más de un procesador, y esta se diseña como no transferible a cache.**
- **En un sistema de este tipo todos los accesos a memoria compartida son fallo de cache, porque la memoria compartida no se copia nunca a cache.**

mejoras al diseño básico

- Cache unificada frente a caches partidas
- Técnicas para disminuir la tasa de fallos
- Técnicas para ocultar y disminur la penalización de fallos
- Técnicas para disminuir los tiempos de acceso
- LA MEMORIA ENTRELAZADA

Cache unificada frente a caches partidas

- cache unificada la que tiene una sola memoria que almacena datos e instrucciones
- Se llama llama cache partida la que tiene una cache de datos y una cache de instrucciones.
- Ventajas de la cache unificada
- Para un tamaño dado mayor tasa de aciertos que la partida porque nivela automáticamente la carga de captura entre instrucciones y datos. Es decir si una ejecución implica más instrucciones que datos se cargara de instrucciones, en caso contrario se llenará de datos.
- Solo se necesita diseñar e implementar una cache
- La tendencia actual son las caches partidas.
- Máquinas superescalares, como el pentium o el powerpc, .
- Ventaja de la memoria partida
- Elimina la competición por la cache entre el procesador de instrucciones y la unidad de ejecución..
- Esto es importante en los sistemas segmentados :
- Normalmente el procesador captará instrucciones anticipadamente y llenará un buffer con las instrucciones que van a ejecutarse.
- Si el sistema tiene una cache unificada cuando la unidad de ejecución realiza un acceso a memoria para cargar y almacenar datos realiza una petición a la cache.
- Si al mismo tiempo el precaptador de instrucciones (que se explica más adelante) emite una petición de lectura de una instrucción a la cache, dicha petición será temporalmente

Técnicas para disminuir la tasa de fallos

- **Tamaño de bloques mayor***
- **Alto grado de asociatividad***
- **Caches víctimas**
- **Caches pseudoasociativas**
- **Prebúsqueda hw de instrucciones y datos**

- Cache víctima
- Se coloca una pequeña cache totalmente asociativa al lado de la cache
- Siempre que se sustituye un bloque se carga en el nivel más bajo de la jerarquía y en la cache víctima
- Siempre que se produce un fallo se comprueba en si el bloque deseado se encuentra en la cache víctima
- Si se encuentra el bloque de la cache y el de la cache víctima se intercambian
- Especialmente útiles para pequeñas caches de emplazamiento directo.
- Dependiendo del programa una memoria víctima de cuatro entradas elimina entre el 20% y el 95% de perdidas de una cache de 4K

Técnicas para disminuir la tasa de fallos

- Cache pseudo asociativa
- Se comporta de maneras diferentes si hay acierto o si hay fallo.
- Si hay un acierto se comporta como una memoria de emplazamiento directo
- Si hay un fallo se vuelve a buscar en otro marco de página.
- ¿En que marco ? Lo mas sencillo es simplemente cambiar el bit mas significativo de los bits de marco de página de la dirección física.
- Ejemplo:
- Suponiendo que la dirección inicial era 111111 11 11 primero se buscaría en el marco 11 y se comprobaría la etiqueta 11111. En caso de fallo se accedería al marco de bloque 01 y se comprobaría la etiqueta 11111.
- Estas caches tienen dos tiempos de acierto.
- Uno el normal cuando se accede a la primera como un emplazamiento directo
- pseudoacierto cuando el datos buscado se encuentra en el segundo acceso.

- Pre búsqueda hardware
- consiste en traer a la memoria cache bloques que todavía no ha demandado el procesador.
- OBL (One Block Loohahead)
- Se carga un bloque y el siguiente :
- Suele ser eficiente siempre que el tamaño del bloque sea pequeño.
- Existen dos técnicas diferentes:
- Se trae el bloque i+1 solo la primera vez que se referencia i
- Prebúsqueda sobre fallo se trae i+1 siempre que se produce un fallo sobre i
- Esta técnica se puede implementar de dos formas diferentes, o bien interna o bien externa a la cache.
- Por ejemplo el Alpha AXP 21064, carga el bloque referenciado en memoria cache y el otro en un buffer externo, si el bloque deseado se encuentra en este buffer se cancela la perdida de cache, se accede a este bloque y se produce una nueva prebúsqueda.
- Se ha demostrado que un buffer de un bloque para una cache de instrucciones de 4kb con bloque de 16bytes y con emplazamiento directo puede reducir la tasa de fallos de un 15 a un 25%. Con un buffer de 4 bloque se puede reducir un

Técnicas para ocultar y disminur la penalización de fallos

- ;;;;;;;;;;;;;;;;;Early Start.-
- Reanudar la ejecución tan pronto como llegue la palabra requerida del bloque
- Funciona mejora para caches de instrucciones debido a su secuencialidad

- Primera palabra requerida:
- La primera palabra que se transfiere es la requerida
- El resto del bloque se transfiere empezando con la palabra siguiente y acabando con la anterior
- Estas dos técnicas no minimizan la penalización pero la ocultan

- Dar preferencia a la lectura
- Suponer el método de la escritura directa con buffer intermedio.
- Suponer que se solicita una lectura.
- Puede ocurrir que el dato actualizado todavía no se haya escrito en MP y por lo tanto esté en el buffer.
- La solución más sencilla es detener la lectura hasta que se vacía el buffer. Aumenta mucho la penalización
- Solución, se comprueba si el dato está en el buffer si esta se recupera del buffer
- Si no está se da prioridad a la lectura del dato de la MP.

Técnicas para ocultar y disminur la penalización de fallos

- **Mejorar los tiempos de espera en la post escritura.**
- **Suponer que se tiene que reemplazar un bloque modificado (dirty bit)porque se ha producido un fallo de lectura.**
- **El proceso habitual es escribir todo el bloque en su posición de mp y a continuación realizar la lectura.**
- **Para dar prioridad a la lectura habría :**
- **Depositar el bloque en un buffer intermedio**
- **Se realiza la lectura**
- **Se realiza la escritura.**
- **Caches no bloqueantes**
- **La memoria cache continua proporcionando datos mientras gestiona un fallo de lectura**

- Caches de dos niveles
- Con el aumento de la densidad de integración es posible tener una memoria cache on chip
- La cache interna reduce la actividad del bus externo del procesadorreduce los tiempos de ejecución e incrementa las prestaciones globales del sistema
- Los accesos a las caches internas mucho más rápidos
- El bus permanece libre para realizar otras transferencias
- En la actualidad aparte de la cache interna los sistemas tienen también caches externas
- A esta estructura se le conoce como caches de dos niveles
- La cache de primer nivel afecta al ciclo de reloj de la CPU
- La cache de segundo nivel solo afecta a la penalización de perdida
- Esto tiene su importancia porque indica que muchas decisiones de diseño que no son aceptables en la cache de primer nivel por afectar al ciclo de reloj, si que lo son para la de segundo nivel.

Técnicas para disminuir los tiempos de acceso

- Los tiempos de accesos a la memoria cache son importantes porque afectan directamente al ciclo de reloj.
- Caches pequeñas y simples
- Cuanto más pequeño es el HW es más rápido
- Esto también ayuda a que se puedan incluir en el chip
- En cuanto a la simplicidad, los emplazamientos directos (que son los más sencillo de implementar) además permiten solapar la comprobación de la etiqueta con la transmisión del dato.
- Caches virtuales
- Se tratan en el tema de memoria virtual.

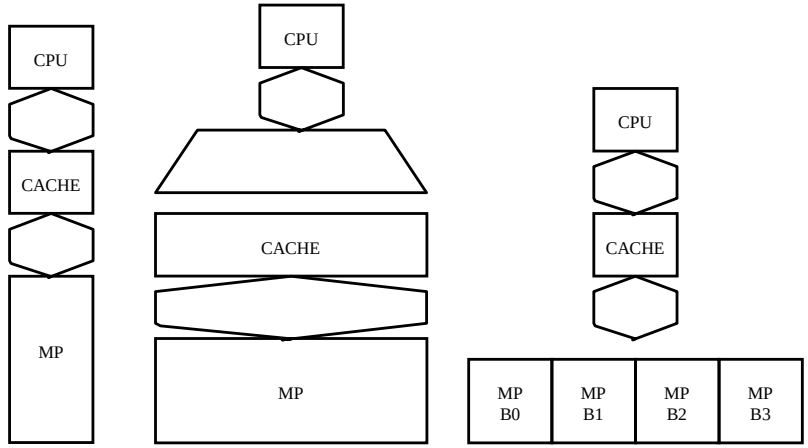
memoria entrelazada

- En el tiempo de penalización cuando se produce un fallo se pueden encontrar dos factores diferentes:
- La latencia
- El tiempo tardado en acceder a todo el bloque
- La latencia se debe principalmente la tecnología y mejora lentamente
- Si se quiere mejorar la penalización se debe intentar mejorara el tiempo de acceso al bloque
- Este tiempo de acceso se puede reducir si se consigue aumentar el ancho de banda de la memoria principal
- Solución
- Incrementar el tamaño de la memoria y del bus con lo que aumenta proporcionalmente la anchura de banda de memoria.
- El coste principal de esta mejora está en la mayor anchura del bus,
- Coste secundario está en los buffers adicionales de la memoria.

- **Otra solución:**
- **Organización de los bancos para leer o escribir múltiples palabras en un solo acceso , en lugar de necesitar un acceso para cada palabra.**
- **El ancho de cada pastilla sería el del bus y de la cache**
- **Al enviar la misma dirección a varios bancos les permite leer a todos simultáneamente.**
- **A esto se le llama memoria entrelazada**

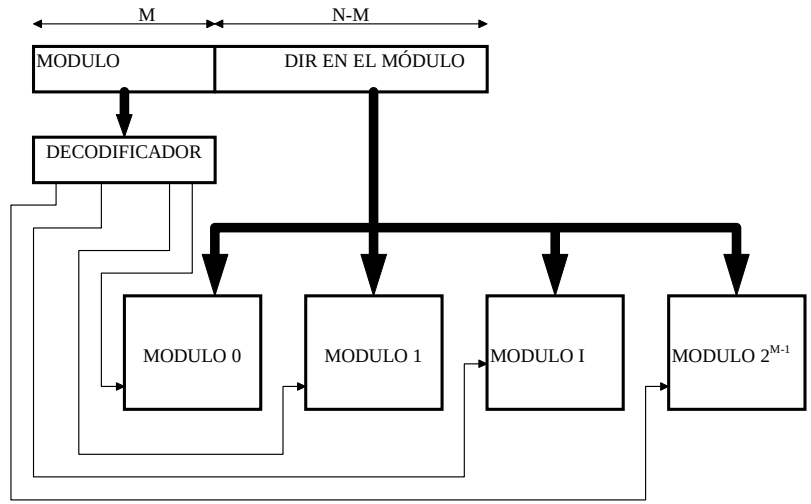
memoria entrelazada

- **ESTRUCTURA DE UNA MEMORIA ENTRELAZADA**
- **Una memoria principal de N palabras se divide en M módulos de N/M palabras cada uno de ellos**
- **Cada uno de los módulos puede trabajar en paralelo con el resto.**
- **Existen dos organizaciones :**
- **Entrelazamiento de orden alto**
- **Entrelazamiento de orden bajo**



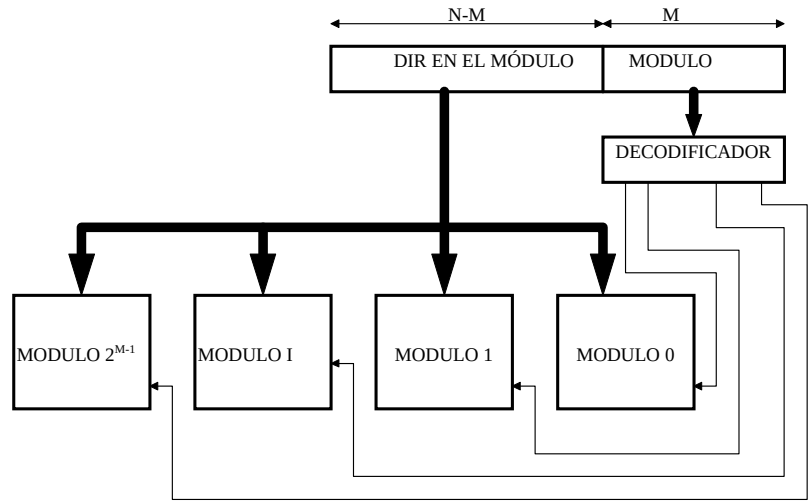
memoria entrelazada

- ENTRELAZAMIENTO DE ORDEN ALTO
- La memoria principal de 2N palabras de divide en 2M módulos cada uno de ellos con 2N-M palabras
- La forma de direccionar la y organizarla se puede ver en la siguiente figura:
- Cada módulo tiene 2N-M palabras consecutivas
- Con los bits más significativos de la dirección se selecciona el módulo. Por lo tanto direcciones consecutivas de memoria se almacenan en el mismo módulo.
- Con este entrelazamiento no se consigue paralelismo de acceso que es lo que se busca.



memoria entrelazada

- ENTRELAZAMIENTO DE ORDEN BAJO
- Con los bits mas significativos de la dirección se seleccionan las palabras
- Con los bits menos significativos se seleccionan los módulos
- Direcciones de memoria consecutivas corresponden a palabras almacenadas en la misma dirección de distintos módulos.



RENDIMIENTO DE LA CACHE

- El tiempo de CPU se divide en:
  - Ciclos de ejecución de programa
  - Ciclos que la CPU espera a la memoria
- Generalmente se suponen que los T_{ACCESO} a cache forman partes de los ciclos de ejecución de programa:
- T_{CPU}= (ciclos de ejecución + ciclos de detención de memoria)*T_{CICLO}
- Ciclos de detención de memoria
  - Se deben a fallos de cache
  - Ciclos de detención = ciclos de lectura + ciclos de escritura
  - Ciclos de lectura
    - » Para calculara los ciclos de detención de lectura (fallos de lecturas) se aplica la siguiente expresión:

Nº de lecturas * %tasa de fallo de lecturas*penalización de fallos de lectura.

RENDIMIENTO DE LA CACHE

- **Ciclos de escritura**
  - **Las paradas por escritura son algo más complejas:**
  - **Por ejemplo para una estrategia de escritura directa existen dos fuentes de paradas diferentes.**
    - » **Los fallos de escritura**
    - » **Paradas de buffer de escritura cuando los buffer están llenos**

Ciclos de parada de escritura = escrituras de un programa*tasa de fallos de escritura*penalización de fallos) + paradas de buffer

RENDIMIENTO DE LA CACHE

- Fijando las siguientes condiciones:
- Las penalizaciones de escritura y lectura son iguales
- Las paradas de buffer despreciables
- Las tasa de fallos de lectura y escritura se suman en la tasa de fallos de accesos se obtiene la siguiente expresión
- Ciclos parada= accesos a memoria por programa*tasa de fallos*penalización
- Esta misma expresión se puede reordenar de la sigiente manera:
- Ciclos parada = I’s por programa*fallos por I*penalizacion de fallos
- CPIfinal=CPIinical+ Nº de acceso* tasa de fallos *penalizacion

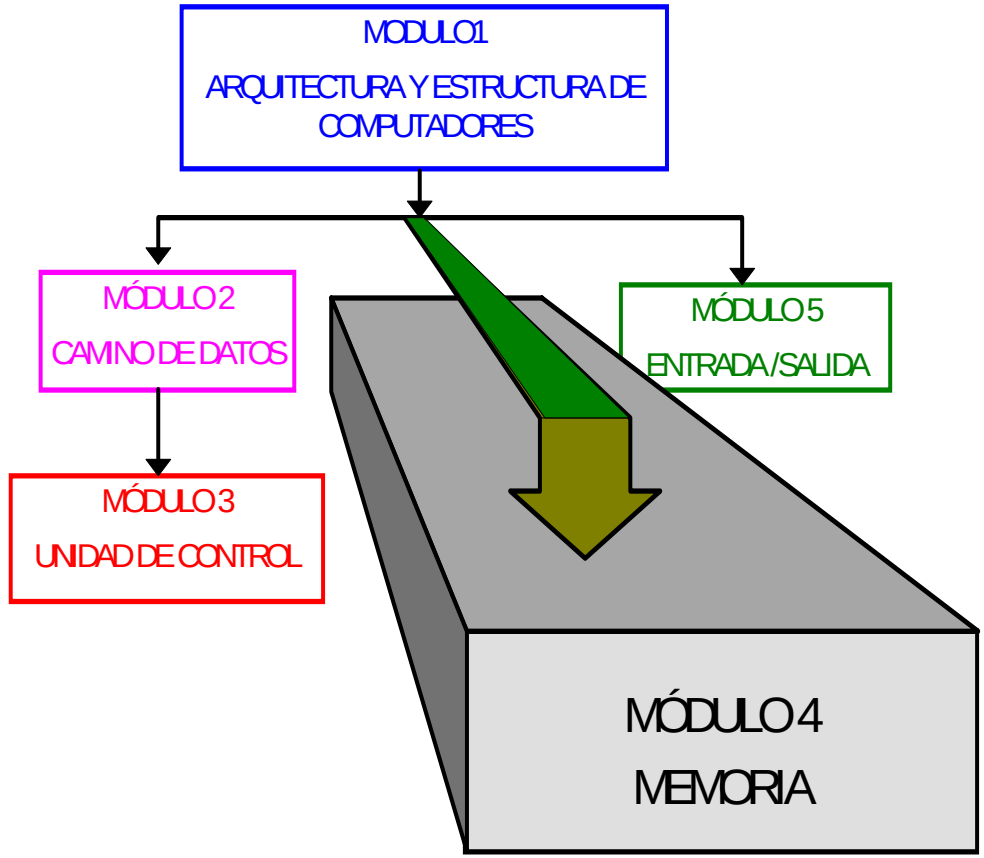
RENDIMIENTO DE LA CACHE

- **Efecto de las mejoras de frecuencia**
  - **las penalizaciones se incrementan cuando la máquina se hace más rápida porque se nota mas la diferencia entre el acceso a cache y el fallo.**
- **Efecto en las mejoras de COI**
  - **cuanto menor sea el número de ciclos por instrucción de la máquina , mayor será el impacto de los ciclos de parada en el rendimiento**
- **La importancia del rendimiento de la cache para CPU con un bajo nº de ciclos por instrucción y elevada frecuencia de trabajo es mayor.**
- **Otra expresión que se suele utilizar es tiempo de ciclos promedio de una cache es:**
- **t=tcache +tfallo*penalización**

LA MEMORIA VIRTUAL

LA MEMORIA VIRTUAL

- **INTRODUCCIÓN**
- **LA MEMORIA REAL**
  - UN USUARIO
  - MULTIPROGRAMACIÓN
- **MEMORIA VIRTUAL**
  - FUNDAMENTOS
  - PAGINADA
    - » DIRECTA
    - » ASOCIATIVA
    - » TLB
  - SEGMENTADA
  - PAGINADA SEGMENTADA
  - ESTRATEGIAS DE ADMINISTRACIÓN
- **MEMORIA VIRTUAL VS MEMORIA CACHE**



OBJETIVOS

- NECESIDAD DE LA MEMORIA VIRTUAL
- APRENDIZAJE DE TECNICAS PARA IMPLEMENTAR LA MEMORIA VIRTUAL
- ELEMENTOS DE DISEÑO
- SIMILITUDES Y DIFERENCIAS ENTRE LA MEMORIA VIRTUAL
- LA MEMORIA CACHE DIRECCIÓN VIRTUAL Y MEMORIA CACHE

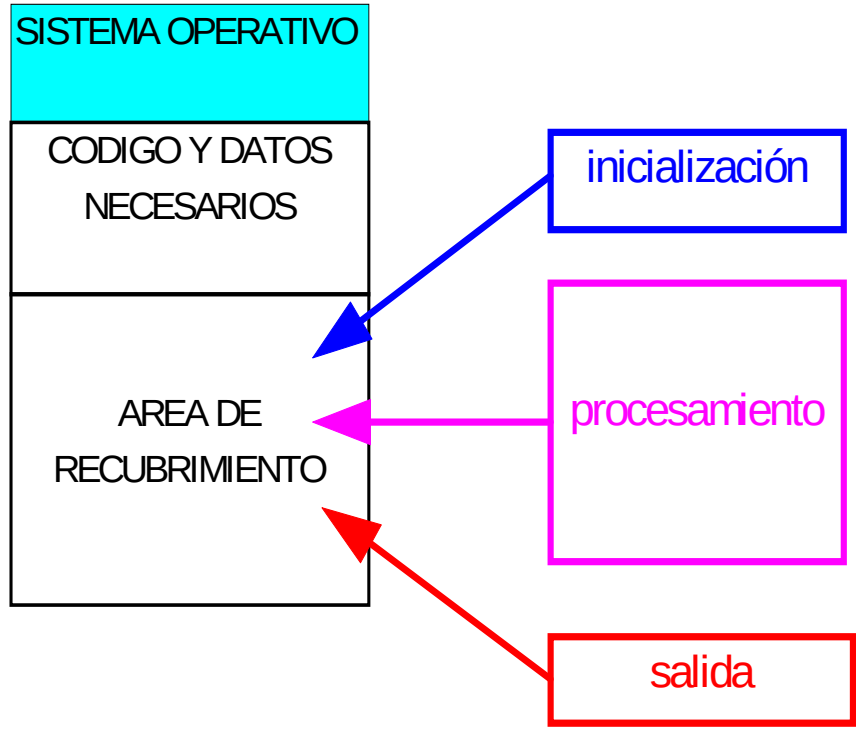
INTRODUCCIÓN

- **LOS SISTEMAS DE COMPUTADORES NECESITAN CADA VEZ MÁS MEMORIA:**
  - EVITAR INACTIVIDAD DE LA CPU
  - PROGRAMAS CADA VEZ MÁS GRANDES
- **PROBLEMA**
  - MP CARA
  - MS LENTA
- **BASANDOSE**
  - VARIEDAD DE CARACTERÍSTICAS DELA MEMORIAS
  - COMPORTAMIENTO PREVISIBLE DE LOS ACCESO S A MEMORIA
- **SE PUEDEN ORGANIZAR LOS DATOS EN DOS NIVELES DE MEMORIA:**
  - RÁPIDA Y PEQUEÑA MP
  - GRANDE LENTA MS
- **OBJETIVO: % ACCESOS MP>%ACCESOS MS**
- **CON ESTO SE CONSIGUE**
  - TAMAÑO MS
  - VELOCIDAD MP

ORGANIZACIÓN DE LA MEMORIA REAL

UN ÚNICO USUARIO

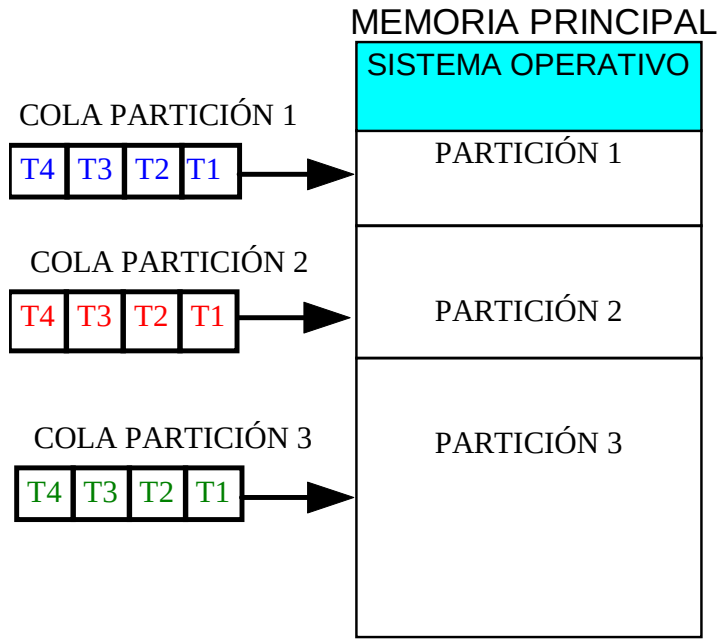
- **PROBLEMA 1**
  - PROGRAMAS MAS LARGOS QUE LA MP
- **SOLUCIÓN1 OVERLAY :**
  - DIVIDIR EL PROGRAMA EN RECUBRIMIENTOS MANUALMENTE
  - TODOS LOS MÓDULO SE CARGAN EN LA MISMA REGIÓN DE LA MEMORIA
  - UN PROGRAMA PRINCIPAL QUE SE ENCARGA DEL INTERCAMBIO DE LOS MÓDULOS
  - TAREA COMPLEJA, LARGA Y TEDIOSA
- **PROBLEMA2**
  - DESAPROVECHAMIENTO DE LOS TIEMPOS MUERTOS DE CPU
- **SOLUCIÓN2 MULTIPROGRAMACIÓN**
  - VARIOS USUARIOS COMPITEN POR LOS RECURSOS DEL SISTEMA



ORGANIZACIÓN DE LA MEMORIA REAL

MULTIPROGRAMACIÓN DE PARTICIÓN FIJA Y CARGA ABSOLUTA

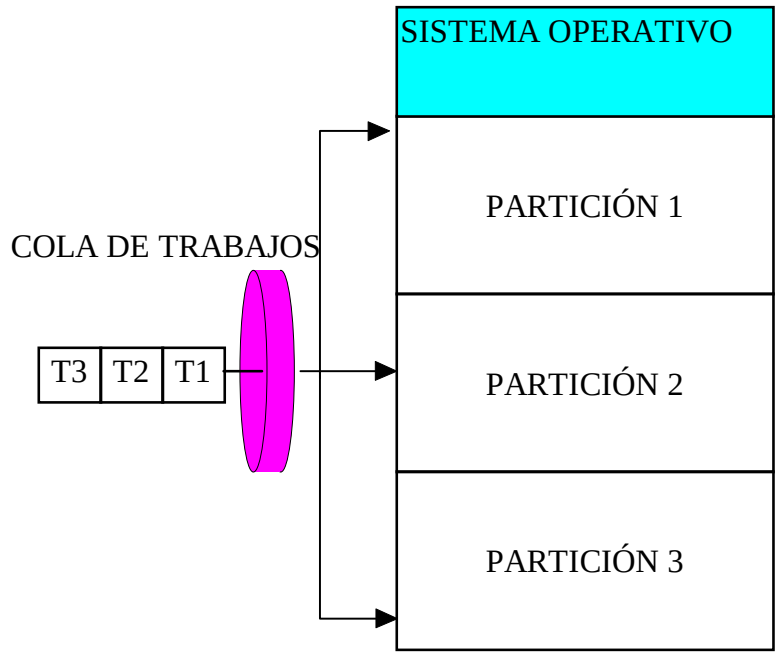
- **SE DIVIDE LA MP EN REGIONES DE TAMAÑO FIJOS**
- **UN TRABAJO SE COLOCA ENTERO EN UNA PARTICIÓN**
- **LA GESTIÓN LA REALIZA EL PROGRAMADOR**
- **PROBLEMA**
  - TRABAJO MAYOR QUE PARTICIÓN- SOL OVERLAY
  - FRAGMENTACIÓN USO INEFICIENTE DE LA MEMORIA
    - » FRAGMENTACIÓN INTERNA MALGASTA ESPACIO DE LA PARTICIÓN
- **UNA POSIBLE SOLUCIÓN A AMBOS PROBLEMAS PARTICIONES DE TAMAÑO DISTINTO.**
- **CARGA ABSOLUTA**
  - UN TRABAJO SIEMPRE LA MISMA PARTICIÓN
  - **INCONVENIENTE:**
    - » SI UN TRABAJO LISTO PARA SU EJECUCIÓN Y SU PARTICIÓN OCUPADA DEBE ESPERA
  - **SOLUCION:**
    - » CARGA REUBICABLE



ORGANIZACIÓN DE LA MEMORIA REAL

MULTIPROGRAMACIÓN DE PARTICIÓN FIJA Y CARGA REUBICABLE

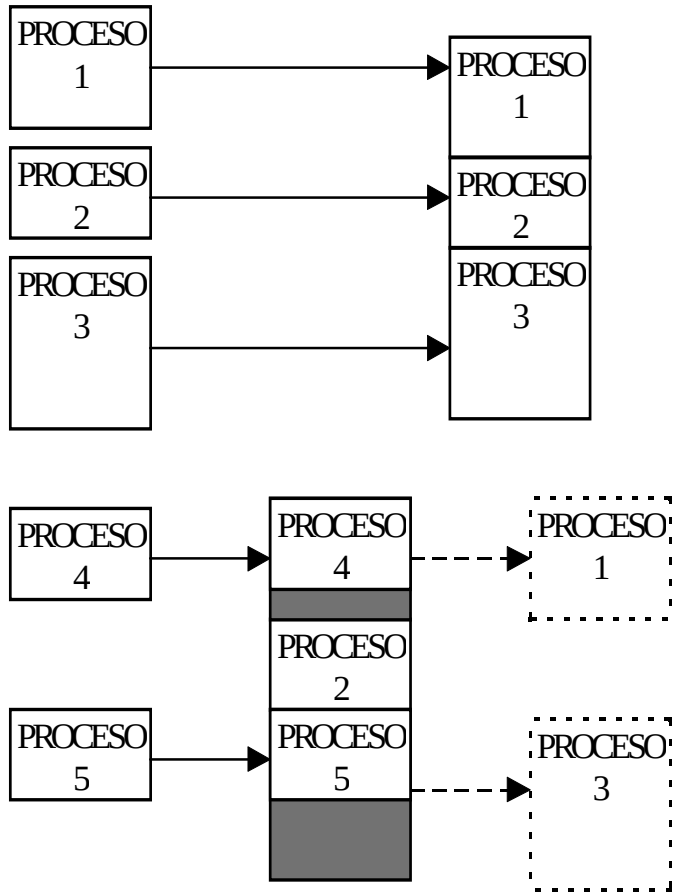
- **CARGA REUBICABLE**
  - LOS TRABAJOS SE EJECUTAN EN CUALQUIER PARTICIÓN QUE ESTÉ LIBRE
  - CONCEPTO DE DIR LÓGICA Y DIRECCIÓN FÍSICA
  - MECANISMO DE TRADUCCIÓN
- **ESTRATEGIAS DE UBICACIÓN:**
  - PRIMERO DISPONIBLE
  - MEJOR AJUSTE
  - PEOR AJUSTE
- **INCONVENIENTES**
  - FRAGMENTACIÓN
  - NÚMERO DE PROCESOS EN MEMORIA ES FIJO.
- **VENTAJAS**
  - SENCILLO DE IMPLEMENTAR
  - POCA SOBRE CARGA DEL SO
- **SO DE IBM EL OS/MFT (MULTIPROGRAMACIÓN CON Nº FIJO DE TAREAS)**
- **PROBLEMAS DE LA PARTICIÓN FIJA:**
  - FRAGMENTACIÓN INTERNA
  - Nº DE PROCESOS FIJO



ORGANIZACIÓN DE LA MEMORIA REAL

MULTIPROGRAMACIÓN DE PARTICIÓN DINÁMICA

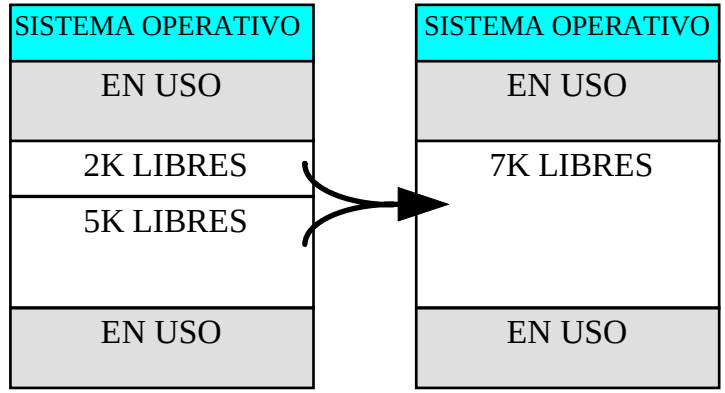
- **LAS PARTICIONES SE CREAN DINÁMICAMENTE**
- **LOS TRABAJOS OCUPAN TODO EL ESPACIO QUE NECESITAN**
- **Nº PARTICIONES VARIABLE --> Nº DE PROCESO VARIABLE**
- **INICIALMENTE OPTIMIZAN LA MEMORIA**
- **FRAGMENTACIÓN EXTERNA A LAS PARTICIONES**
- **VENTAJAS:**
  - DESAPARECE LA FRAGMENTACIÓN INTERNA
  - USO MÁS EFICIENTE DE LA MEMORIA
- **DESVENTAJAS**
  - INEFICIENCIA DEL PROCESADOR DEBIDO A LA NECESIDAD DE COMPACTAR LA MEMORIA PARA EVITAR LA FRAGMENTACIÓN EXTERNA
- **OS/MVT**
  - MULTIPROGRAMACIÓN CON NÚMERO VARIABLE DE TAREAS



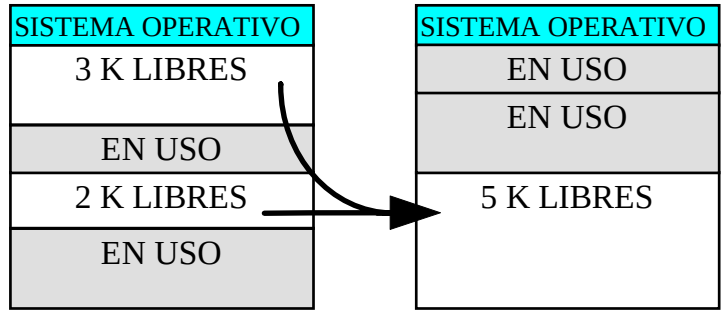
ORGANIZACIÓN DE MEMORIA REAL
MULTIPROGRAMACIÓN DE PARTICIÓN DINÁMICA

SOLUCIÓN A LA FRAGMENTACIÓN

- **AGRUPAR ESPACIOS LIBRES CONSECUTIVOS**
- **COMPACTAR ESPACIOS LIBRES PASAR TODAS LAS ÁREAS LIBRES A UN EXTREMO**
- **INCONVENIENTES:**
  - CONSUMO DE RECURSOS
  - DETENCIÓN DEL SISTEMA
  - REUBICACIÓN DE TRABAJOS
  - SI LOS TRABAJOS PEQUEÑOS LA COMPACTACIÓN CON FRECUENCIA



a) agrupar



b) compactar

ORGANIZACIÓN DE LA MEMORIA REAL

PAGINACIÓN Y SEGMENTACIÓN

PAGINACIÓN SIMPLE

- LA MEMORIA SE DIVIDE EN MARCOS
- LOS PROCESOS SE DIVIDEN EN PAGINAS DE IGUAL TAMAÑO
- TODO EL PROCESO EN MEMORIA
- PAGINA SE SITÚAN EN MARCOS NO NECESARIAMENTE CONTIGUOS
- SOLUCIONA LA FRAGMENTACIÓN EXTERNA
- FRAGMENTACIÓN INTERNA EN LA ÚLTIMA PÁGINA DEL PROCESO
- DIRECCIÓN LÓGICA SE COMPONE DE Nº DE PAGINA Y DESPLAZAMIENTO
- TABLA DE PÁGINAS
- TRASPARENTE AL PROGRAMADOR

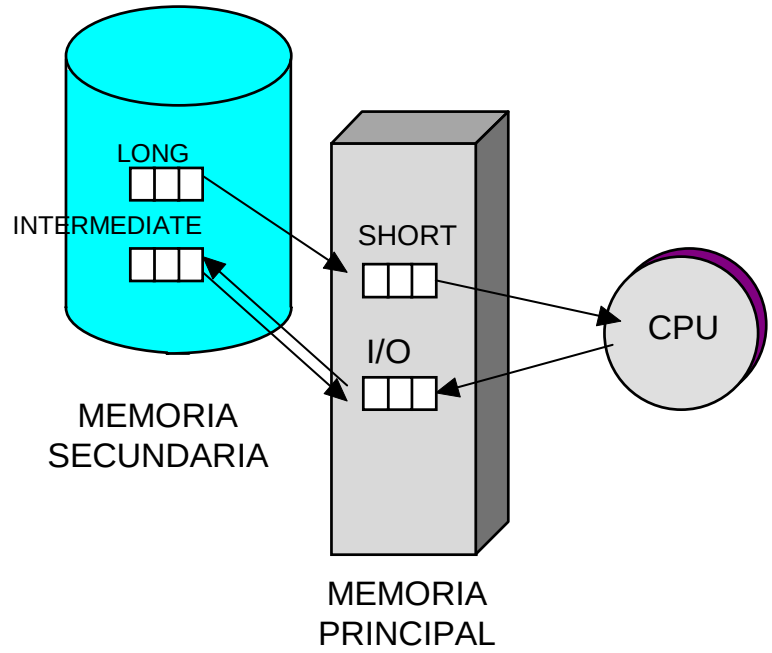
SEGMENTACIÓN SIMPLE

- PROGRAMA SE DIVIDE EN SEGMENTOS DE TAMAÑO VARIABLE
- SE CARGAN DINÁMICAMENTE EN MEMORIA
- LOS SEGMENTOS NO TIENE QUE ESTAR CONTIGUOS
- DIRECCIÓN LÓGICA Nº SEGMENTO MAS DESPLAZAMIENTO
- TABLA DE SEGMENTOS
- NO TIENE FRAGMENTACIÓN INTERNA
- FRAGMENTACIÓN EXTERNA PEQUEÑA
- VISIBLE COMODIDAD PARA ORGANIZAR PROGRAMAS Y DATOS

ORGANIZACIÓN DE MEMORIA REAL

MULTIPROGRAMACIÓN CON INTERCAMBIO

- **HASTA EL MOMENTO UNA CARACTERÍSTICA ES QUE LOS TRABAJOS EN MEMORIA HASTA QUE SE FINALIZA**
- **UN TRABAJO SE EJECUTA HASTA QUE NO PUEDE CONTINUAR**
- **ALMACENA EN LA LISTA DE E/S**
- **PROBLEMA:**
  - COMOE/S LENTA:
  - TODOS LOS PROCESOS COLA E/S
- **SOLUCION:**
  - NUEVA COLA EN MS
  - INTERCAMBIO DE PROCESOS ENTEROS,
- **INCONVENIENTE:**
  - SUPONE OPERACIONES DE ENTRADA/SALIDA



MEMORIA VIRTUAL
DEFINICIONES

- **Sistema de almacenamiento jerárquico de dos niveles administrado por el sistema operativo que da al usuario la sensación de tener un espacio de direcciones superior al que realmente tiene**

- **OBJETIVO Velocidad de acceso similar a la de la memoria principal con un coste por bit y tamaño similar al de la memoria secundaria**

- **COMO FUNCIONA**
  - Se divide la memoria real en marcos de bloques
  - Se divide un proceso en bloques
  - Se carga en la memoria real un pequeño conjunto de bloques del proceso
  - El resto de bloques a memoria secundaria
  - Se van trayendo nuevos bloques a la mp según se van necesitando

- **NECESIDADES.- MECANISMO DE TRADUCCION DE DIRECCIONES Y ESTRATEGIAS DE ADMINISTRACIÓN**

- **FALLO DE PAGINA.- CUANDO SE INTENTA ACCEDER A UNA PÁGINA QUE NO ESTÁ EN MEMORIA PRINCIPAL**

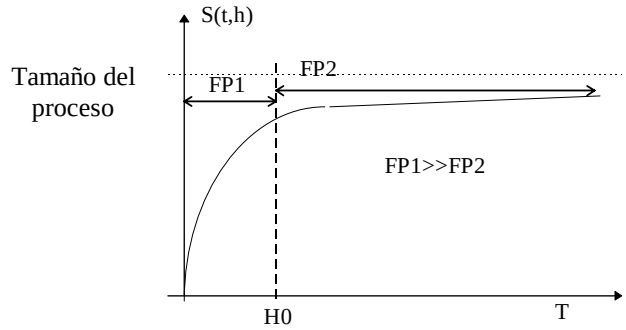
- **LOS FALLOS DE PÁGINA LOS GESTIONA EL SISTEMA OPERATIVO**

MEMORIA VIRTUAL
FUNDAMENTOS

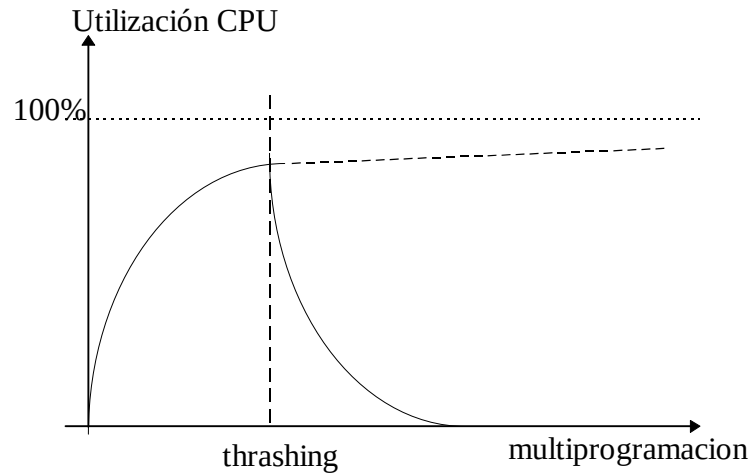
- **SECUENCIA DE REFERENCIAS A MEMORIA NO ES ALEATORIA**
- **PRINCIPIO DE LOCALIDAD DURANTE UN INTERVALO DE TIEMPO UN PROGRAMA TIENDE A AGRUPAR SUS REFERENCIAS A MEMORIA EN UNA PEQUEÑA PORCIÓN DEL ESPACIO DE DIRECCIONES DISPONIBLE.**
- **LA LOCALIDAD TIENE DOS COMPONENTES:**
  - **TEMPORAL** EN UN FUTURO CERCANO SE REFERENCIAN OBJETOS REFERENCIADOS EN EL PASADO RECIENTE
    - » BUCLES
    - » SE VE CONDICIONADA POR EL NÚMERO DE BLOQUES QUE PUEDE HABER EN MP
  - **ESPACIAL** EN EL FUTURO CERCANO SE REFERENCIAN OBJETOS CERCANOS AL LOS ÚLTIMOS OBJETOS REFERENCIADOS
    - » EJECUCIÓN SECUENCIAL DE LOS PROGRAMAS
    - » ESTRUCTURAS DE DATOS
    - » IMPORTANTE PARA DETERMINAR LOS TAMAÑOS DE LOS BLOQUES

MEMORIA VIRTUAL
FUNDAMENTOS

- **CONJUNTOS DE TRABAJO**
  - CONJUNTO DE PÁGINAS DEL PROCESO QUE PUEDEN SER REFERENCIADAS EN UN INTERVALO DE TIEMPO
  - w(T, H) T INSTANTE DE TIEMPO, H NUMERO DE REFERENCIAS
- **SI UN PROGRAMA TIENE TODO SU CONJUNTO DE TRABAJO EN MP OCURRIRÁN POCOS FALLOS DE PÁGINA SEGÚN AVANZA LAS COMPUTACIÓN**
- **LOS FALLOS SE INCREMENTAN NOTABLEMENTE SI LOS CONJUNTOS DE TRABAJO NO ESTÁN DISPONIBLES**
- **EL ESFUERZO DE ADMINISTRACIÓN DEBE DIRIGIRSE A QUE TODO EL CONJUNTO DE TRABAJO ESTE EN MEMORIA**
- **SI ESTO NO SE CUMPLE THRASHING**
- **CUANTO MÁS FALLOS DE PÁGINA MÁS SE ALEJA LOS TIEMPOS DE ACCESO DE LOS DE LA MP**
- **IMPORTANCIA EN LA MULTIPROGRAMACIÓN**



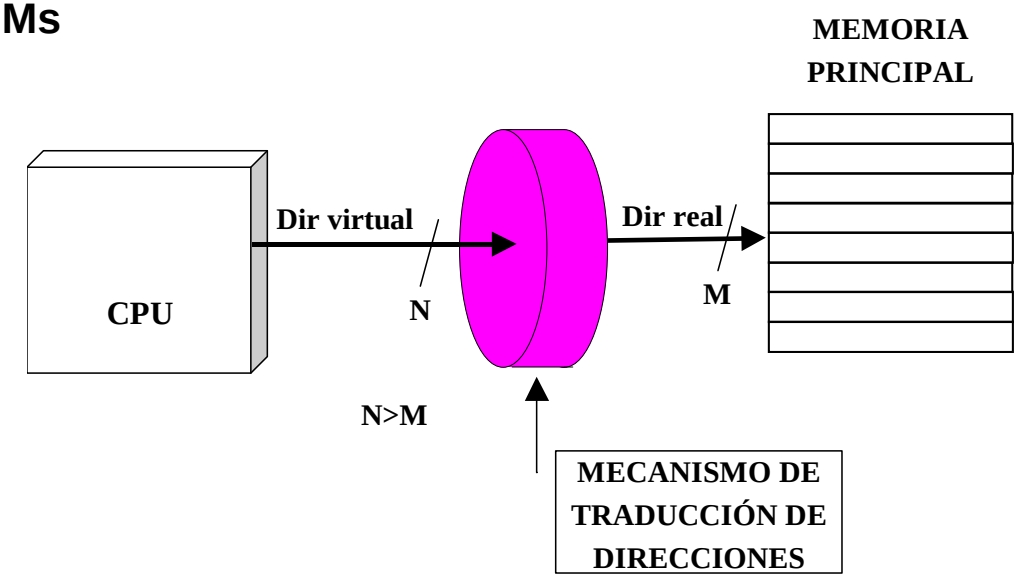
h→ nº bloques del conjunto de trabajo
h0 → tamaño de conjunto de trabajo óptimo
FP Nº de fallos de página



MEMORIA VIRTUAL

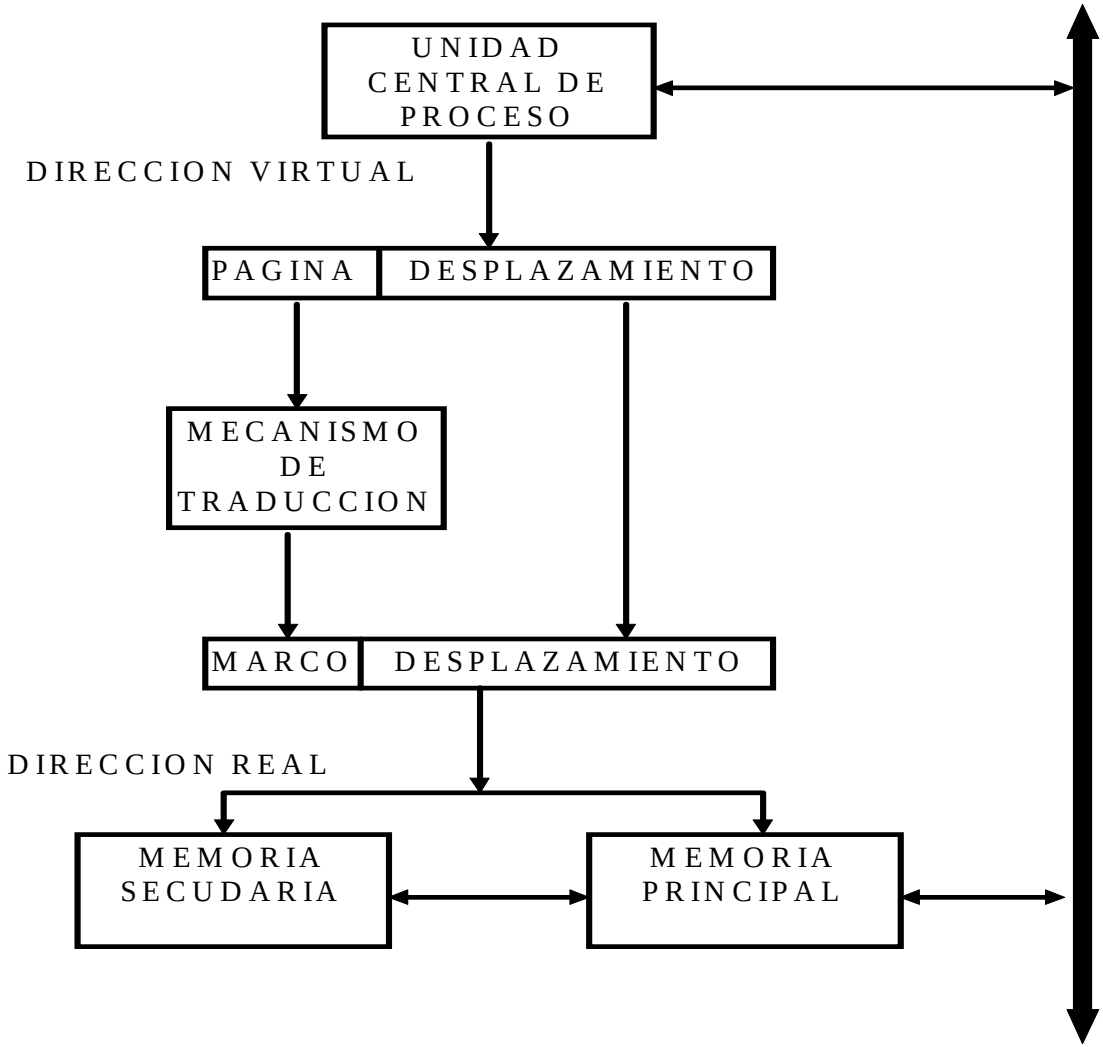
MECANISMOS DE TRADUCCIÓN DE DIRECCIONES

- **Espacio de direcciones del proceso(virtual) >> espacio de direcciones MP**
- **Mecanismo de Traducción Dinámica**
  - por bloques
    - » cuanto mas pequeño el bloque mayor cantidad de información, mayor número de procesos
    - » cuanto más grande el bloque menos información menor número de procesos
- **se implementa mediante tablas de páginas**
- **La dirección virtual unas veces se traduce a una dirección de MP y otras a una dirección de Ms**



MEMORIA VIRTUAL

MECANISMOS DE TRADUCCIÓN DE DIRECCIONES

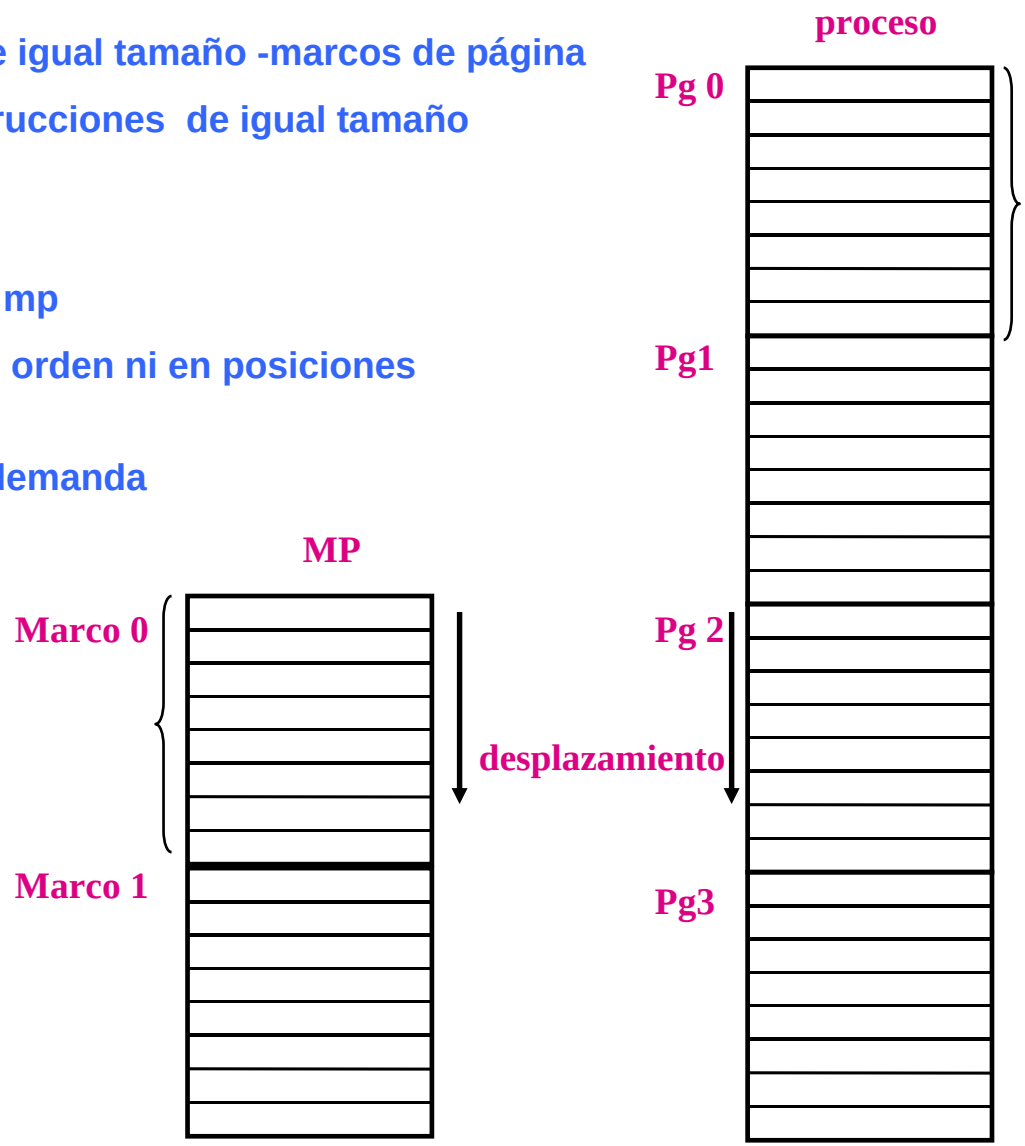


MEMORIA VIRTUAL
RENDIMIENTO

- **T= (Nº CICLOS DE MP+ CICLOS DETENCIÓN DE MP)xT CICLO**
  - CICLO DETENCIÓN = CICLOSLECTURA(ms) + CICLOSESCRITURA(ms)
  - CICLOSLECTURA(ms)= Nº LECTURASx%TASA DE FALLOSxPENALIZACIÓN(en nª ciclos)
- **PENALIZACIÓN:**
  - TIEMPO QUE SE TARDA EN LEER O ESCRIBIR UN BLOQUE EN MS
  - 10.000 1.000.000 INSTRUCCIONES
- **PARA INCREMENTAR EL RENDIMIENTO**
  - REDUCIR LA PENALIZACIÓN
  - REDUCIR AL MÁXIMO LOS FALLOS DE PÁGINA
- **REDUCCIÓN DE LA TASA DE FALLOS:**
  - SI SE PUEDE COLOCAR CUALQUIER PÁGINA VIRTUAL EN CUALQUIER MARCO SE PUEDE SUSTITUIR CUALQUIER PAGINA
  - SO IMPLEMENTA ALGORITMOS SOFISTICADOS DE REEMPLAZAMIENTO QUE MINIMICEN LOS FALLOS
- **OTRA CONSECUENCIA**
  - SUSTITUCION DE PROCESOS

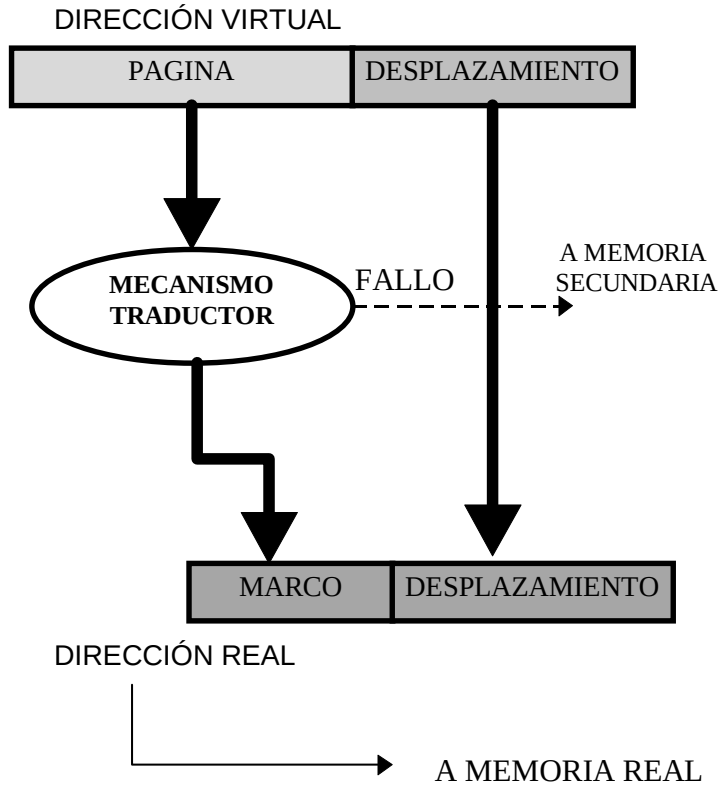
MEMORIA VIRTUAL
PAGINADA

- La mp se divide en conjuntos de posiciones de igual tamaño -marcos de página
- Los procesos se dividen en conjuntos de instrucciones de igual tamaño llamados páginas
- |PAGINAS|=|MARCOS DE PAGINA|
- un proceso solo guarda algunas paginas en la mp
- las páginas del proceso no tienen que estar en orden ni en posiciones consecutivas
- Las páginas se trae traen a MP desde MS por demanda
- DI VIRTUAL=(p,d) Y DIR FÍSICA (m,d)



MEMORIA VIRTUAL
PAGINADA

- **Se necesita un mecanismo de traducción**
  - Convierte dir virtuales en dir MP y en dir de MS
- **Tablas de pagina**
  - Relaciona las direcciones virtuales con las reales
  - la crea el SO cuando se activa un proceso por primera vez
  - Tantas tablas como procesos
- **Mecanismo de traducción:**
  - Directa
  - Asociativa
  - Mixta



MEMORIA VIRTUAL
PAGINADA DIRECTA

- **LA TABLA DE PÁGINAS (TP) TIENE UNA ENTRADA POR CADA PÁGINA VIRTUAL**
- **UNA ENTRADA A LA TABLA CONTIENE EL MARCO DE PAGINA (MP) EN QUE SE GUARDA LA PAGINA**
- **REG. BASE LA POSICIÓN INICIAL DE LA TP**
- **Nº PÁGINA VIRTUAL ES UN DESPLAZAMIENTO RESPECTO AL REG BASE**
- **TP GRANDE**
  - EN MEMORIA PRINCIPAL
  - LENTA
- **TP PEQUEÑA:**
  - CONJUNTO DE REGISTROS
  - RÁPIDA
- **PROCESOS GRANDES TABLAS GRANDES. SOLUCIONES**
  - TABLA DE DOS NIVELES
  - TABLA DE PÁGINAS INVERSA

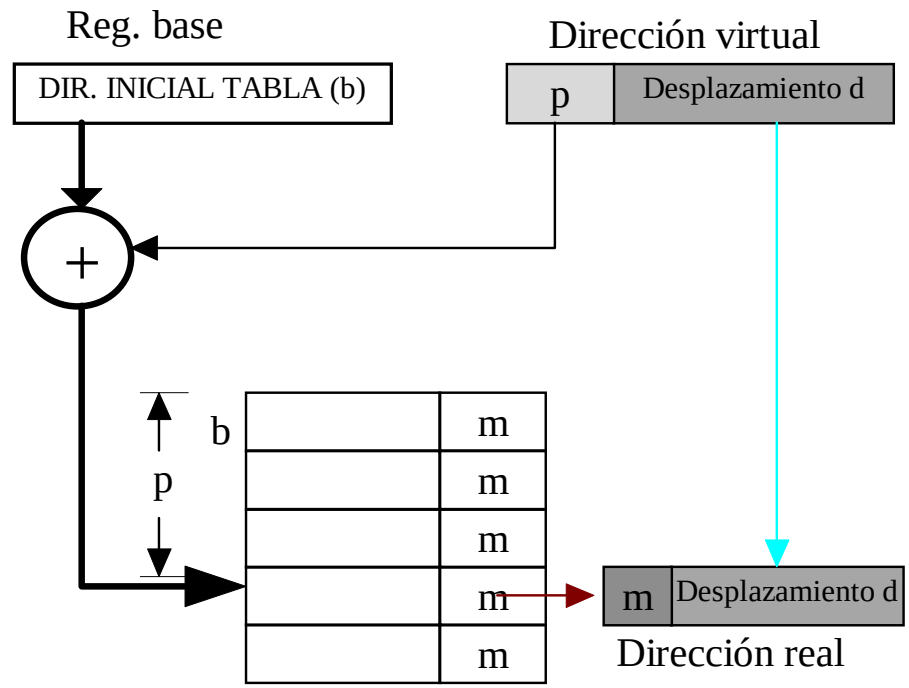
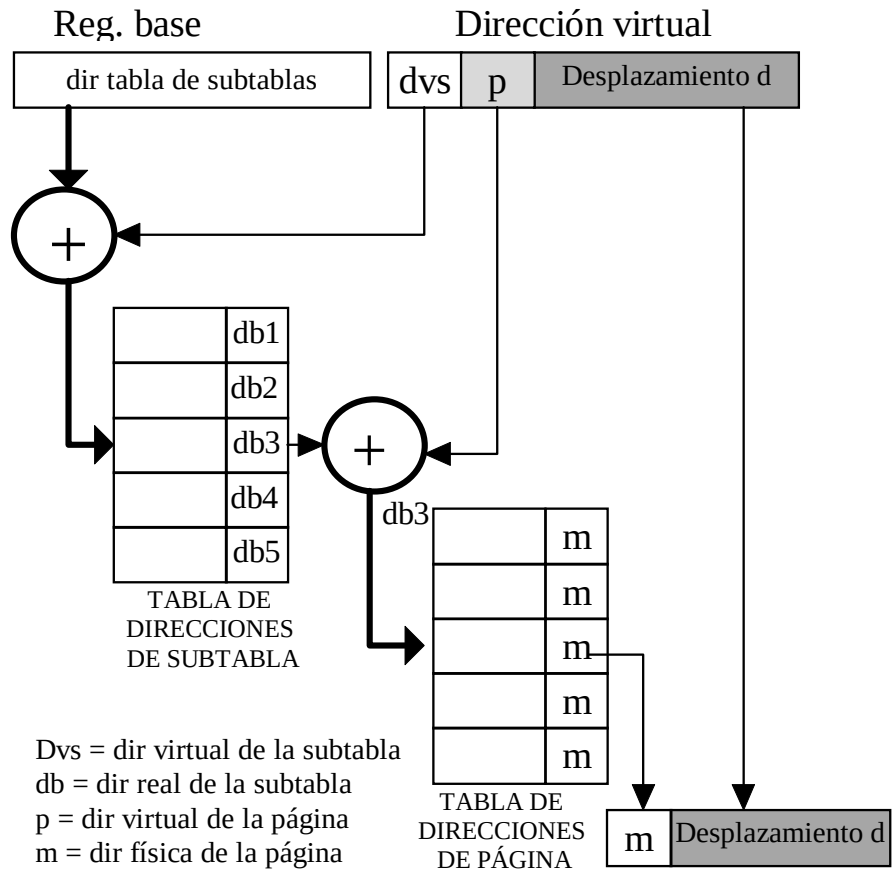


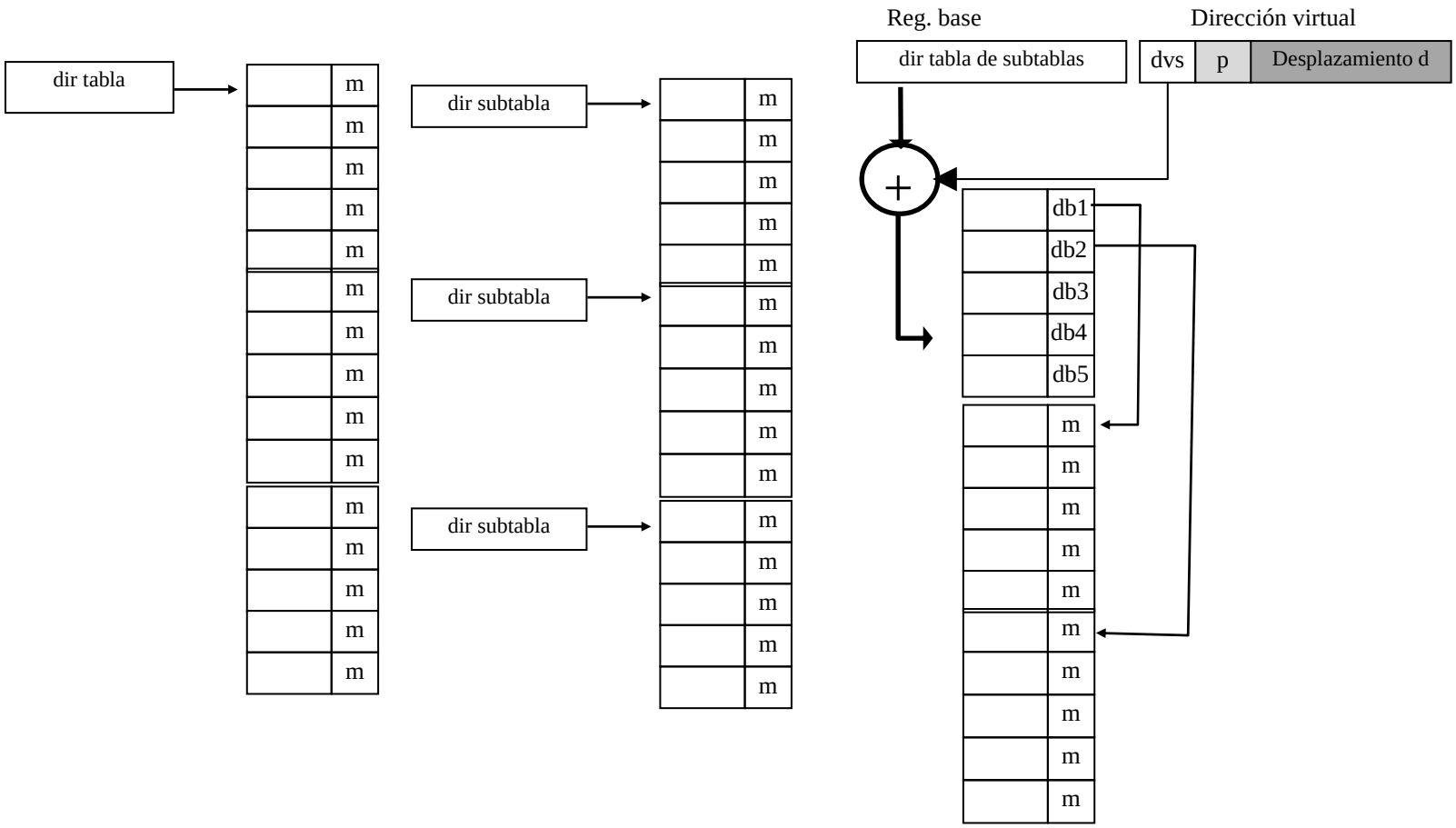
Tabla de página directa

MEMORIA VIRTUAL
PAGINADA DE DOS NIVELES

- **LA TABLA DE PÁGINAS SE DIVIDE EN SUBTABLAS QUE PUEDEN ESTAR EN MEMORIA PRINCIPAL O SECUNDARIA --> TABLA VIRTUAL.**
- **NO ES NECESARIO QUE LAS SUBTABLAS ESTÉN EN POSICIONES CONSECUTIVAS EN MEMORIA**
- **ES NECESARIA UNA DIRECCIÓN BASE PARA CADA SUBTABLA**
  - TABLA DE DIRECCIONES DE SUBTABLA
- **LA DIRECCIÓN DE PÁGINA VIRTUAL SE DIVIDE EN DOS**
  - DVS DIRECCIÓN VIRTUAL DE SUBTABLA
  - P DIRECCIÓN DE LA PÁGINA EN LA SUBTABLA
- **PENTIUM**

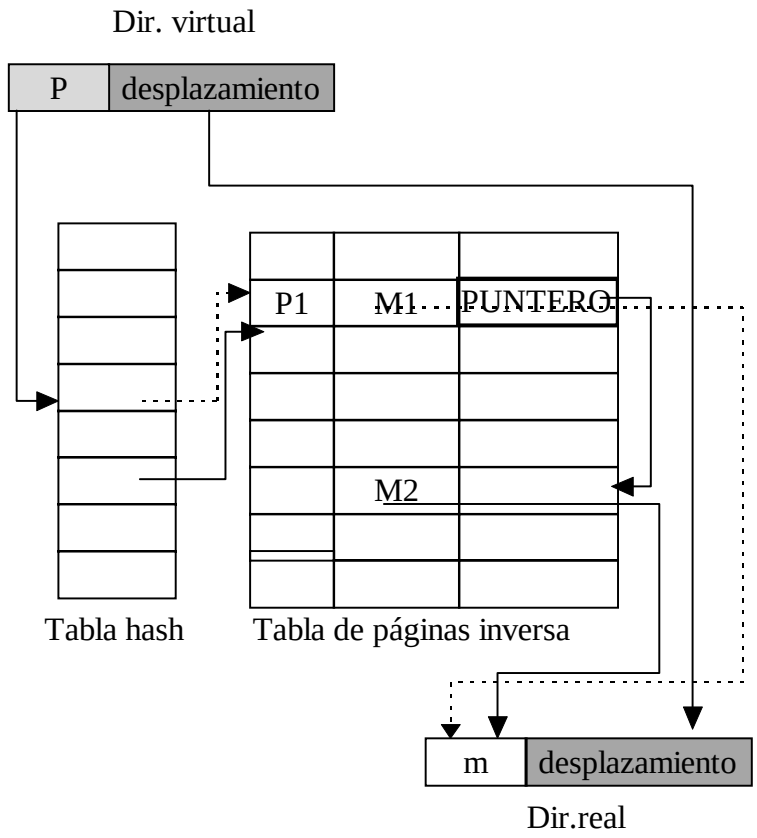


MEMORIA VIRTUAL
PAGINADA DE DOS NIVELES



MEMORIA VIRTUAL
PAGINADA INVERSA

- EN LUGAR DE UNA ENTRADA POR PÁGINA VIRTUAL TIENE UNA ENTRADA POR MARCO DE PÁGINA
- VENTAJA- TAMAÑO DE LA TABLA CONSTANTE INDEPENDIENTEMENTE DEL TAMAÑO DEL PROCESO
- LA PÁGINA VIRTUAL SE ASOCIA A LA PÁGINA INVERSA A TRAVÉS DE UNA FUNCIÓN HASH
- EN ESTA TÉCNICA MÁS DE UNA PÁGINA VIRTUAL PUEDE APUNTAR A LA MISMA POSICIÓN DE LA TABLA
- SOLUCIÓN UNA TÉCNICA DE ENCADENAMIENTO.
- POWERPC



MEMORIA VIRTUAL PAGINADA
FALLO DE PÁGINA

- **UNA ENTRADA A LA TABLA DE PAGINA CONTIENE INFORMACIÓN PARA GESTIONAR LOS FALLOS DE MEMORIA**
- **ESTA INFORMACIÓN PUEDE SER**
  - **V** BIT VÁLIDO
    - » V=1 EXISTE LA PÁGINA Y PUEDE ESTAR EN MP O MS
    - » V=0 PAGINA NULA TIENE QUE CREARSE
  - **M** BIT DE MEMORIA DE DISCO
    - » M=1 PAGINA ACTIVA EN MEMORIA PRINCIPAL
    - » M=0 PAGINA NO NULA EN MEMORIA SECUNDARIA
  - **DMP** DIRECCIÓN DEL MARCO
  - **LEX** - TIPOS DE ACCESO PERMITIDOS
  - **P** INDICA LA PRIVACIDAD. UNA MISMA PÁGINA PUEDE PERTENECER A VARIOS PROCESOS.

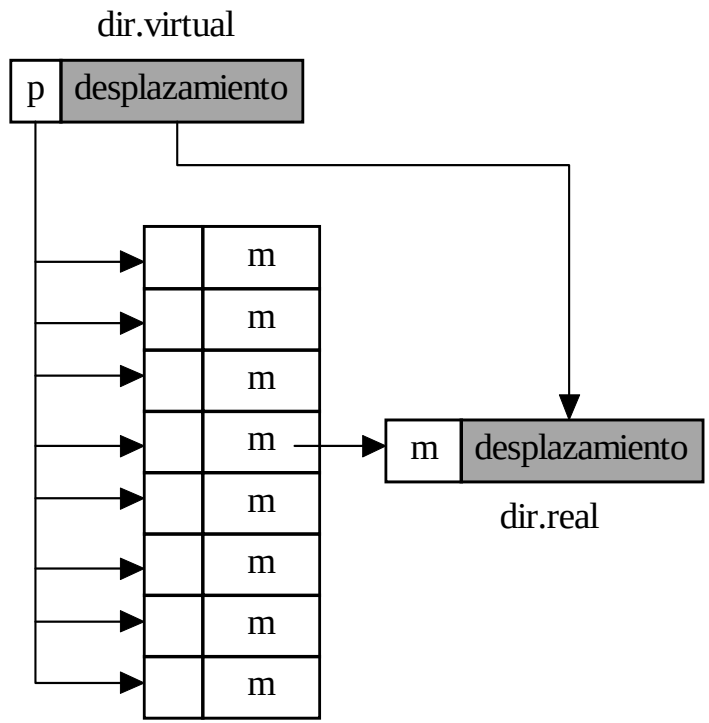
MEMORIA VIRTUAL

INCONVENIENTES DE LA PAGINACIÓN DIRECTA

- **COMO LAS TABLAS GRANDES SE IMPLEMENTAN EN MEMORIA PRINCIPAL**
  - CADA LECTURA DE UNA INSTRUCCIÓN SUPONE DOS LECTURAS DE MEMORIA
    - 1ºPARA OBTENER EL MARCO
    - 2º PARA LEER LA INSTRUCCIÓN
- **LOS PROGRAMAS SE EJECUTA A LA MITAD DE SU VELOCIDAD**
- **ESTO SE HACE INTOLERABLE**
- **SOLUCIÓN- PAGINACIÓN ASOCIATIVA**

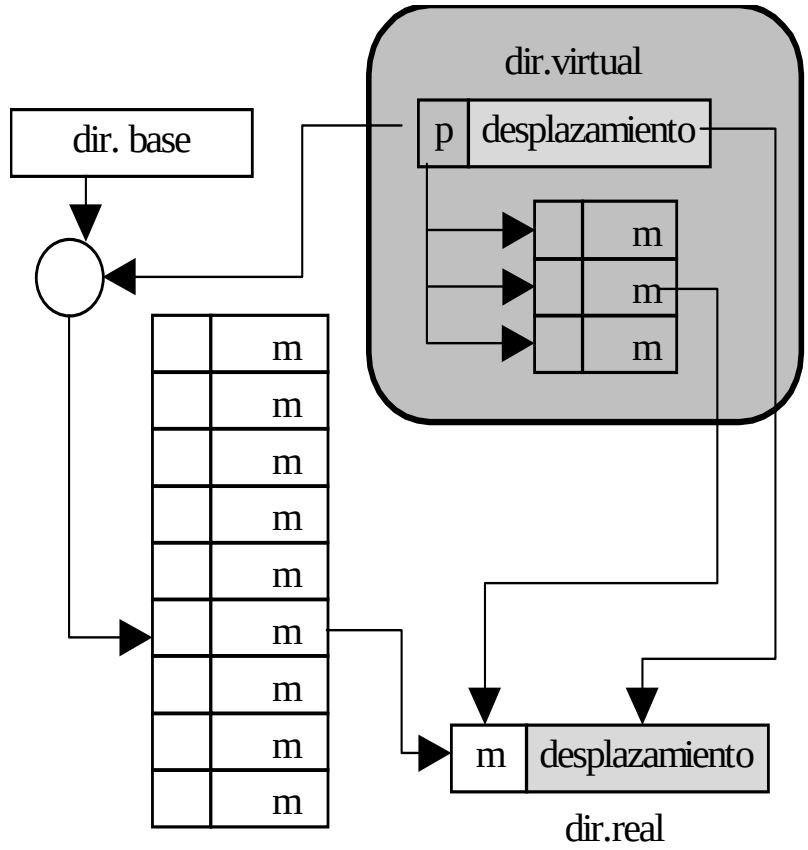
MEMORIA VIRTUAL
PAGINADA ASOCIATIVA

- **SE IMPLEMENTA LA TABLA CON UNA MEMORIA ASOCIATIVA**
- **MUY RÁPIDA UN ORDEN DE MAGNITUD MÁS RÁPIDO**
- **SE COMPRUEBAN TODAS LAS POSICIONES DE LA TABLA SIMULTÁNEAMENTE**
- **DE LOS DOS ACCESOS NECESARIOS EL PRIMERO ES MUY RÁPIDO**
- **PROBLEMAS**
  - ES MUY COSTOSO
  - SI ES GRANDE SE DEGRADA EL RENDIMIENTO
- **SOLUCIÓN**
  - DIRECTA -ASOCIATIVA O TRANSLATION LOOKASIDE BUFFER (TLB)



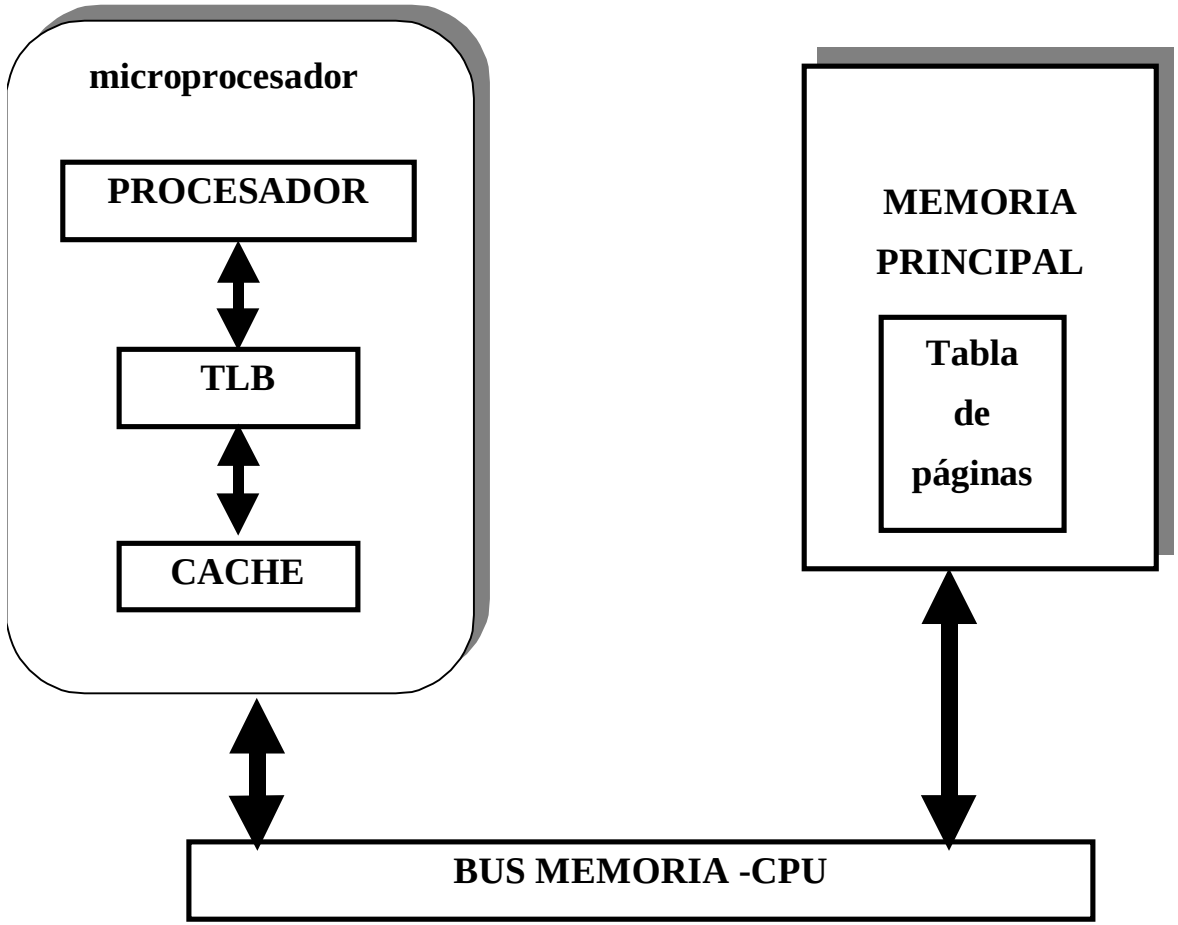
MEMORIA VIRTUAL
PAGINADA DIRECTA ASOCIATIVA (TLB)

- **TRANSLATION LOOKASIDE BUFFER**
- **DOS TABLAS DE PÁGINA**
  - UNA DIRECTA
  - ASOCIATIVA
- **LA TABLA DIRECTA CONTIENE TODAS LAS PÁGINAS VIRTUALES**
- **LA TABLA ASOCIATIVA**
  - información de las páginas más recientemente accedidas
  - aprovecha el principio de localidad
- **EFECTO CACHE**
- **MODO DE FUNCIONAMIENTO:**
  - Primero se accede a la asociativa
  - Si no está la información de la página buscada se accede a la tabla directa



MEMORIA VIRTUAL

PAGINADA DIRECTA ASOCIATIVA (TLB



MEMORIA VIRTUAL
FALLO DE TLB

- **HAY QUE DETERMINAR**

  - **FALLO DE TLB**--
    - » luego el dato dato en la memoria principal
    - » hay que realizar acceso a la tabla directa en MP
    - » guardar la información en la TLB
    - » de nuevo se accede al tlb que ahora contiene la información

  - **FALLO DE PÁGINA**
    - » dato en la memoria secundaria
    - » se trae la página de MS a MP
    - » se actualiza la tabla de direcciones
    - » se actualiza el tlb
    - » se accede de nuevo a la información

MEMORIA VIRTUAL
FALLO DE TLB

- **INFORMACIÓN DE LA ENTRADA DE TLB:**
  - DIR DE MP
  - BIT DE ESCRITURA QUE INDICA SI SE MODIFICO LA PAGINA
  - BIT DE REFERENCIA PARA IMPLEMENTAR EL ALGORITMO DE REEMPLAZAMIENTO
- **FALLOS DE TLB MAS FRECUENTES QUE LOS DE PÁGINA**
  - MECANISMO DE REEMPLAZAMINETO SENCILLO -- ALEATORIO
  - POST ESCRITURA DE LAS ENTRADAS DEL TLB EN LA TABLA DE PAGINAS DIRECTA

MEMORIA VIRTUAL PAGINADA

TAMAÑO DE PAGINA

- **COMPROMISO ENTRE:**
  - NUMERO DE PALABRAS
  - NUMERO DE TABLAS DE PÁGINA
  - TAMAÑO MEDIO DE ENTIDADES LÓGICAS DEL PROCESO

- **PAGINA GRANDE:**
  - TABLAS DE PÁGINA MÁS PEQUEÑAS
  - SE APROVECHA DE LA LOCALIDAD ESPACIAL
  - MAYOR FRAGMENTACIÓN INTERNA
  - MAYOR LENTITUD EN CARGAR LAS NUEVAS PÁGINAS

- **PAGINAS PEQUEÑAS:**
  - TABLAS DE PÁGINAS MAYORES
  - ACCESO DE MEMORIA ASOCIATIVA LIMITADO
  - LOCALIDAD TEMPORAL

MEMORIA VIRTUAL SEGMENTADA

- **BLOQUES DE TAMAÑO VARIABLE**
- **EL SEGMENTO SE AJUSTA A LAS ESTRUCTURAS DE PROGRAMA**
- **LAS INSTRUCCIONES DEL SEGMENTO EN POSICIONES CONSECUTIVAS**
- **LA SEGMENTACIÓN ES VISIBLE AL USUARIO**
- **SIRVE PARA**
  - ORGANIZAR PROGRAMAS Y DATOS
  - ASOCIAR PRIVILEGIOS Y ATRIBUTOS A PROGRAMAS
- **EXPLOTA LA TÉCNICA DE MODULARIDAD DE LOS PROGRAMAS ESTRUCTURADOS**
- **RUTINAS DE TRANSFERENCIA MÁS COMPLEJAS QUE LAS PAGINADAS**
- **MECANISMOS DE DIRECCIONAMIENTO:**
  - DIRECTA
  - ASOCIATIVA Y
  - MIXTA

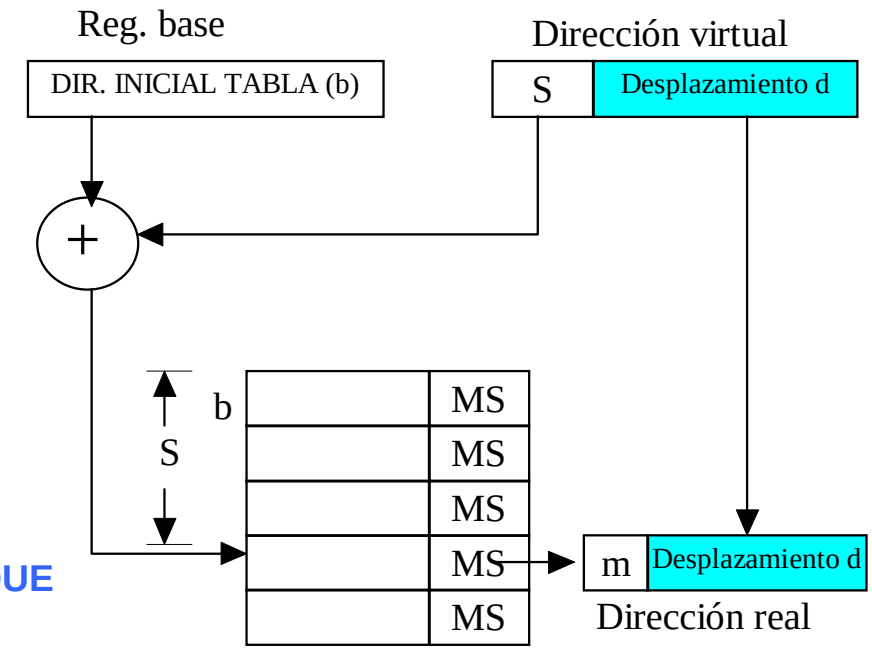
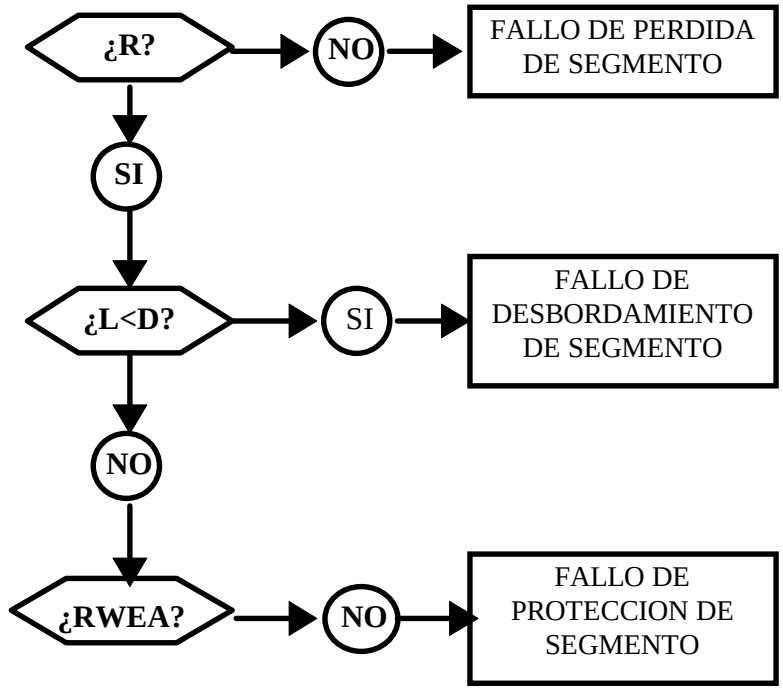


Tabla de segmentos directa

MEMORIA VIRTUAL SEGMENTADA

FALLO DE PERDIDA DE SEGMENTO SEGMENTOS

- **INFORMACIÓN PARA GESTIONAR LOS FALLOS DE SEGMENTO**
- **R BIT DE RESIDENCIA DEL SEGMENTO**
  - SI R=0 EL SEGMENTO NO ESTA EN MEMORIA REAL-
- **L LONGITUD DEL SEGMENTO**
  - PARA DESCODIFICAR LA DIRECCIÓN HAY QUE COMPROBAR SI EL DESPLAZAMIENTO DE LA DIRECCIÓN VIRTUAL ES MENOR QUE LA LONGITUD DEL SEGMENTO :
- **RWEA BITS DE PROTECCIÓN**
- **A DIRECCIÓN DE ALMACENAMIENTO SECUNDARIO**



MEMORIA VIRTUAL
SEGMENTADA- PAGINADA

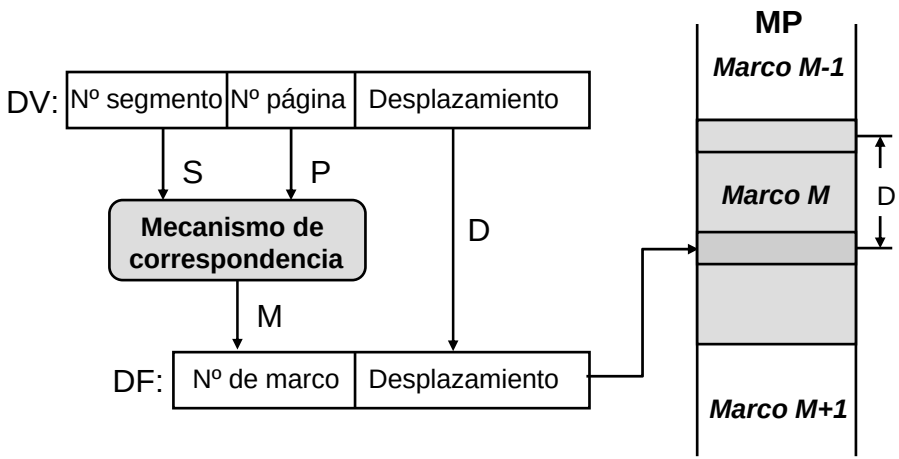
- **Un proceso se descompone en segmentos**
  - los segmentos no necesitan ser contiguos
  - no es necesario que todos los segmentos de un proceso estén en MP
    - » fallo de segmento

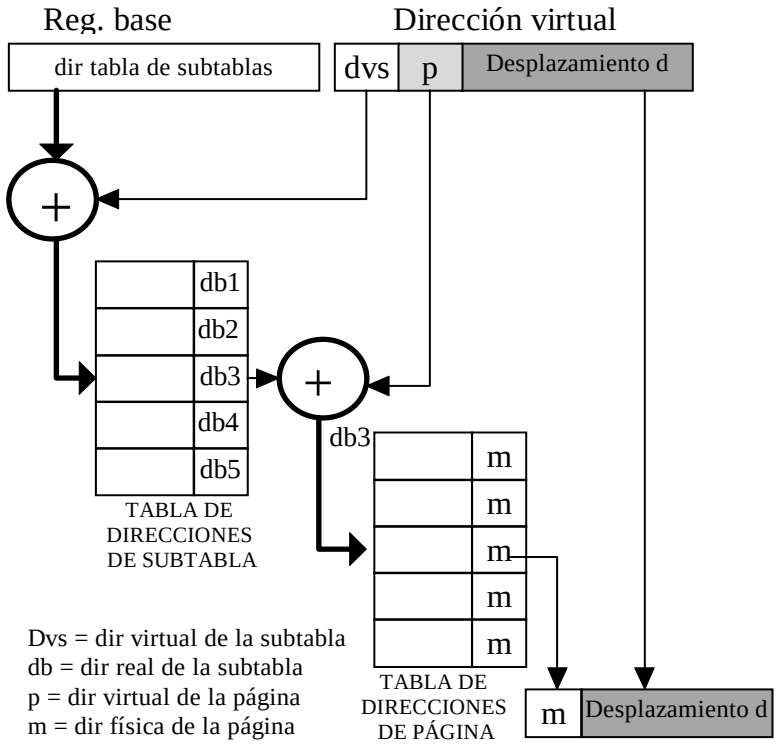
- **Un segmento se descompone en páginas**
  - Tamaño del segmento múltiplo del tamaño de pagina
  - Las paginas del mismo segmento no necesitan ser contiguas
  - no es necesario que todas las páginas estén en MP
    - » fallo de página

¿Cómo se implementa esto?

MEMORIA VIRTUAL
SEGMENTADA- PAGINADA

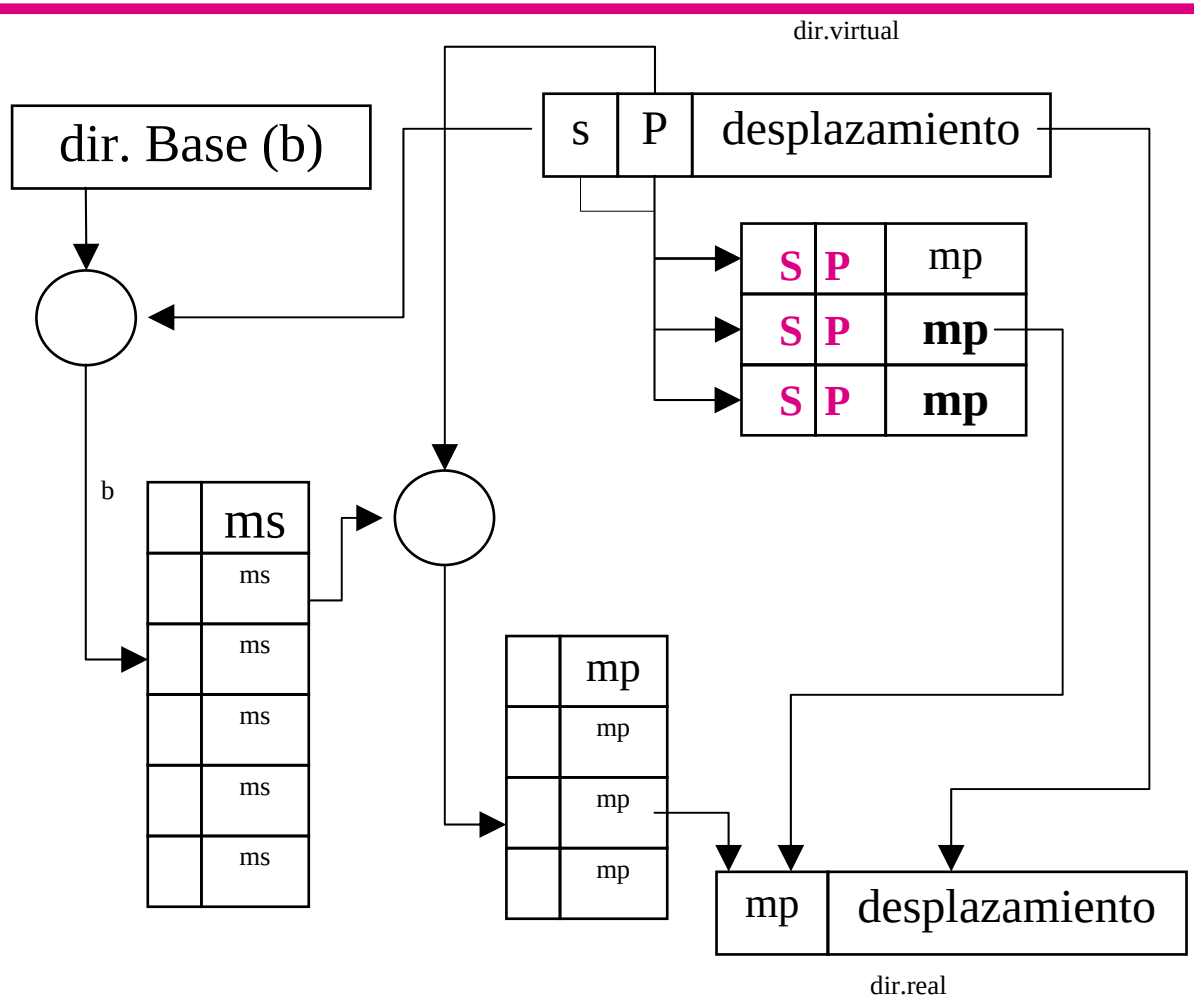
- **Dirección virtual compuesta de tres elementos (s,p,d)**
  - S número de segmento
    - » 2^S segmentos por proceso
  - P número de página del segmento
    - » 2^P paginas por segmento
  - D desplazamiento relativo a la página
    - » 2^D palabras por pagina





MEMORIA VIRTUAL

SEGMENTADA- PAGINADAcon tlb



Las páginas recientemente referenciadas tienen entradas en un almacenamiento asociativo por los campos s y p

MEMORIA VIRTUAL
paginada-segmentada

- **La tabla de segmentos está siempre en MP**
- **Las tablas de páginas pueden estar o no en MP**
- **Un segmento no esta en MP si su tabla de páginas no está en MP**
  - fallo de segmento o de tabla de páginas
  - hay que tener en cuenta la longitud del segmento por que el tamaño de la tabla de pgs varia con el segmento
- **En realidad el tratamiento se hace página a página**
- **Conceptualmente idéntico a los dos niveles, con la diferencia de que al poder ser los segmentos de tamaño variable sus tablas de páginas también son variables**

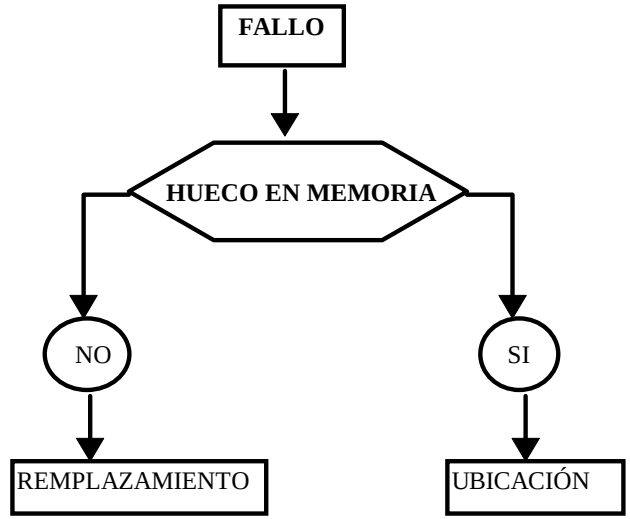
MEMORIA VIRTUAL

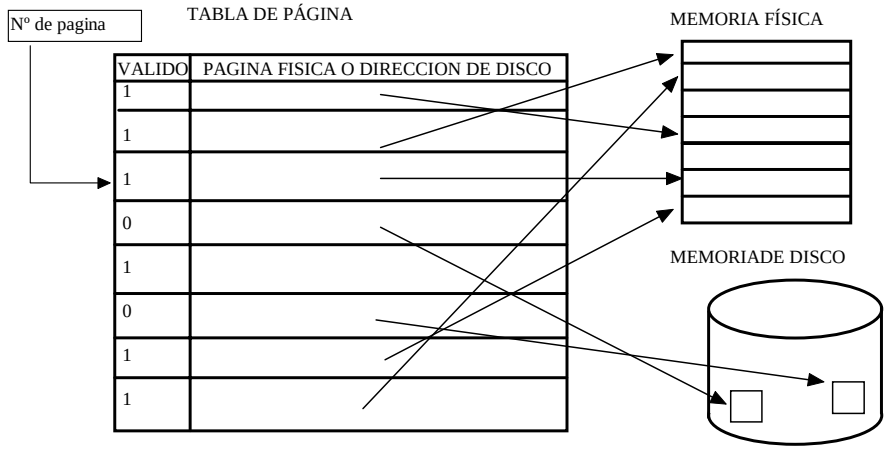
ESCRITURA

- LAS ESCRITURAS EN LA MEMORIA SECUNDARIA NECESITAN MILES DE CICLOS
- LA ESCRITURA DIRECTA ES IMPRACTICABLE
- POST ESCRITURA
- LAS ESCRITURAS INDIVIDUALES SE ALMACENAN EN LA PÁGINA
- CUANDO SE ELIMINA LA PÁGINA SE ESCRIBE EN EL NIVEL INFERIOR
- SE PUEDE MEJORAR INDICANDO SI LAS PÁGINAS NECESITAN ESCRITURA

MEMORIA VIRTUAL
GESTIÓN DE FALLOS DE REFERENCIA

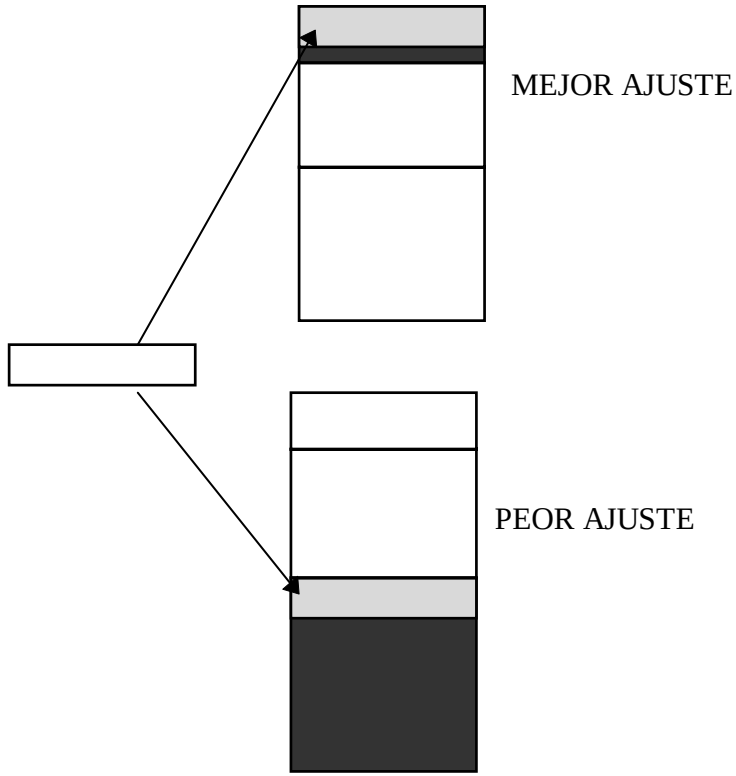
- **FALLO DE REFERENCIA OCURRE CUANDO SE REFERENCIA A UN OBJETO QUE NO ESTÁ EN MEMORIA REAL**
- **SOLUCIÓN LLAMADA AL SISTEMA OPERATIVO PARA QUE TRAIGA UN NUEVO OBJETO**
- **GESTIÓN DE FALLOS:**
  - ESTRATEGIAS DE UBICACIÓN
  - ESTRATEGIAS DE REMPLAZAMIENTO
  - ESTRATEGIAS DE BÚSQUEDA
- **CUANDO SE PRODUCE UN FALLO DE REFERENCIA LO PRIMERO ES COMPROBAR SI EXISTEN ELEMENTOS EN LA LISTA DE HUECOS**
  - SI EXISTEN ESTRATEGIA DE UBICACIÓN
  - SI NO EXISTEN ESTRATEGIAS DE REMPLAZAMIENTO





MEMORIA VIRTUAL
ESTRATEGIAS DE UBICACIÓN

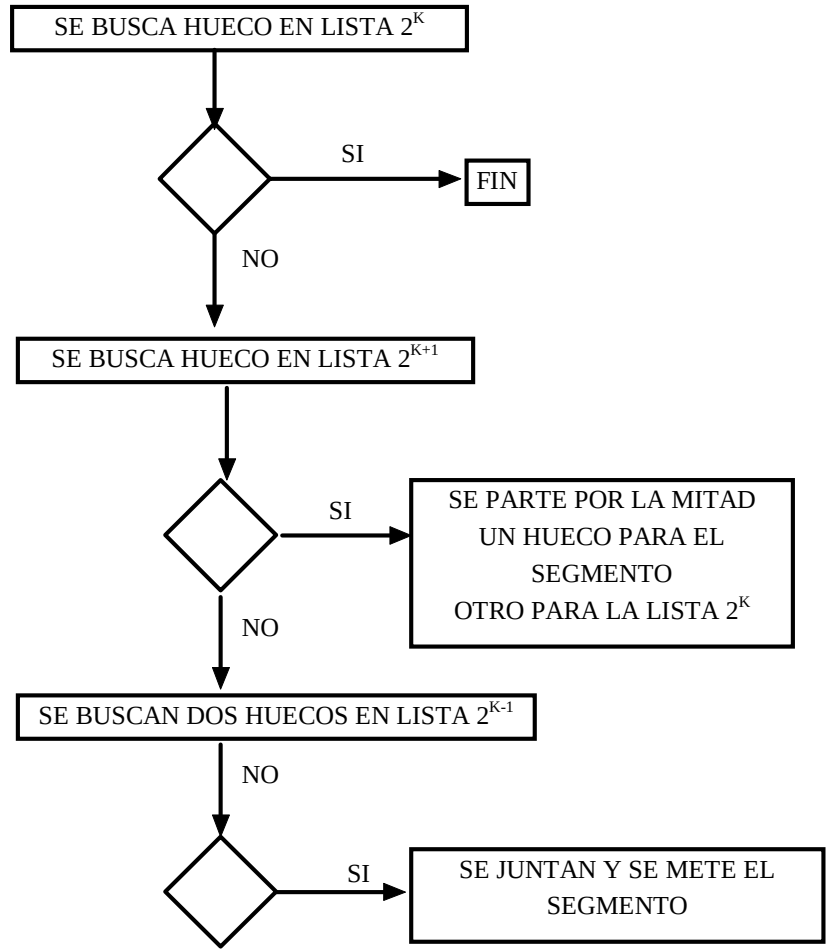
- **EN LOS PAGINADOS ES IRRELEVANTE**
  - TODOS EL MISMO TAMAÑO
- **EN LOS SEGMENTADOS:**
  - PRIMERO DISPONIBLE
  - MEJOR AJUSTE
  - PERO AJUSTE
  - BUDDY
- **PRIMERO DISPONIBLE**
  - PRIMER ESPACIO LIBRE
  - FÁCIL DE IMPLEMENTAR
  - LISTA DE HUECOS LIBRES
- **MEJOR AJUSTE**
  - EL HUECO QUE ENCAJA MEJOR
  - LISTA ASCENDENTE DE HUECOS
- **PERO AJUSTE**
  - LISTA DESCENDENTE DE HUECOS



MEMORIA VIRTUAL

ESTRATEGIAS DE UBICACIÓN

- **BUDDY**
  - N LISTAS DE HUECOS DE TAMAÑOS DIFERENTES
    - » LISTA1 HUECOS DE TAMAÑO 2^1
    - » LISTAN HUECOS DE TAMAÑO 2^N
- **UN HUECO SE PUEDE PARTIR EN DOS DE IGUAL TAMAÑO**
- **DOS HUECOS SE PUEDEN UNIR PARA FORMAR UNO**
- **ALGORITMO**
  - SI HAY SITIO EN LA LISTA K SE OCUPA
  - SI NO LO HAY SE BUSCA EN LA LISTA K+1
    - » SI HAY SE PARTE POR R LA MITAD .UN HUECO PARA EL SEGMENTO Y OTRO PARA LA LISTA
    - » SI NO SE BUSCAN DOS HUECO EN LA LISTA K-1 Y SE UNEN



MEMORIA VIRTUAL
ESTRATEGIAS DE REMPLAZAMIENTO

- **SE APLICAN CUANDO NO HAY HUECOS EN LA MEMORIA PRINCIPAL**
- **OBJETIVO: MINIMIZAR LA TASA DE FALLOS**
- **REPOSICIÓN AL AZAR**
  - POCA SOBRECARGA
  - NO DISCRIMINA NINGÚN USUARIO
  - TODAS LAS PG LA MISMA PROBABILIDAD DE SER DESPLAZADAS
- **PRIMERO EN ENTRAR PRIMERO EN SALIR**
  - MARCA DE TIEMPO EN CADA PÁGINA
  - EL PROBLEMA ES QUE SE PUEDEN REMPLAZAR PÁGINAS MUY USADAS
- **MENOS RECIENTEMENTE USADA**
  - LA QUE LLEVA MÁS TIEMPO SIN USARSE
  - FALLO EN LOS CICLOS LARGOS
- **MENOS FRECUENTEMENTE USADA**
  - FALLO PUEDE SER LA ULTIMA REFERENCIADA

MEMORIA VIRTUAL
ESTRATEGIAS DE REMPLAZAMIENTO

- **NO USADA RECIENTEMENTE**
  - LAS PÁGINAS NO USADAS RECIENTEMENTE TIENEN POCA PROBABILIDAD DE USARSE EN UN FUTURO PRÓXIMO
  - CADA PÁGINA CON DOS BITS :REFERENCIADO (RE) Y MODIFICADO (MO)
  - INICIALMENTE TODOS A 0
  - AL REFERENCIARSE RE= 1
  - CUANDO SE MODIFICA MO= 1
  - CUANDO LA PÁGINA SE REMPLAZA SE BUSCA EN ESTE ORDEN
    - » NO REFERENCIADO NO MODIFICADO
    - » NO REFERENCIADO MODIFICADO
    - » REFERENCIADO NO MODIFICADO
    - » REFERENCIADO MODIFICADO
  - PELIGRO LLEGA UN MOMENTO QUE TODOS RE=1 SE PIERDE LA CAPACIDAD DE DECISIÓN
  - SOLUCIÓN PERIÓDICAMENTE RE=0

MEMORIA VIRTUAL
ESTRATEGIAS DE BÚSQUEDA

PAGINACIÓN POR DEMANDA

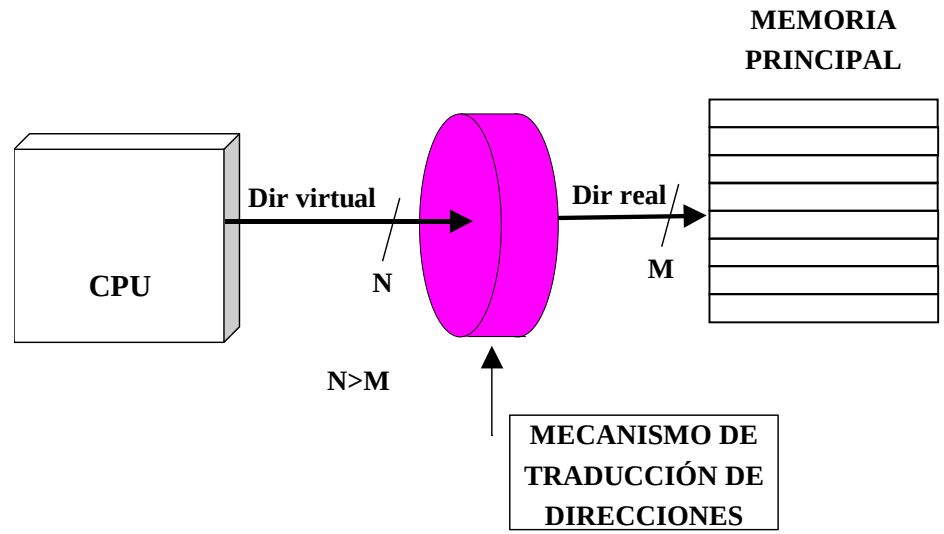
- LAS PÁGINAS SE TRAEN CUANDO SE REFERENCIAN EXPLÍCITAMENTE
- EL SISTEMA SE SOBRE CARGA POCO
- DESVENTAJAS
  - EL PROCESO ACUMULA SUS PAGINAS UNA A UNA
  - CADA VEZ QUE SE PRODUCE UN FALLO DEBE TRAE UNA PÁGINA A MEMORIA PRINCIPAL
  - CUANTO MÁS PÁGINAS EN MP MÁS LARGA LA ESPERA DEBIDO A QUE CADA VEZ MÁS PROCESOS EN ESPERA

PAGINACIÓN ANTICIPADA

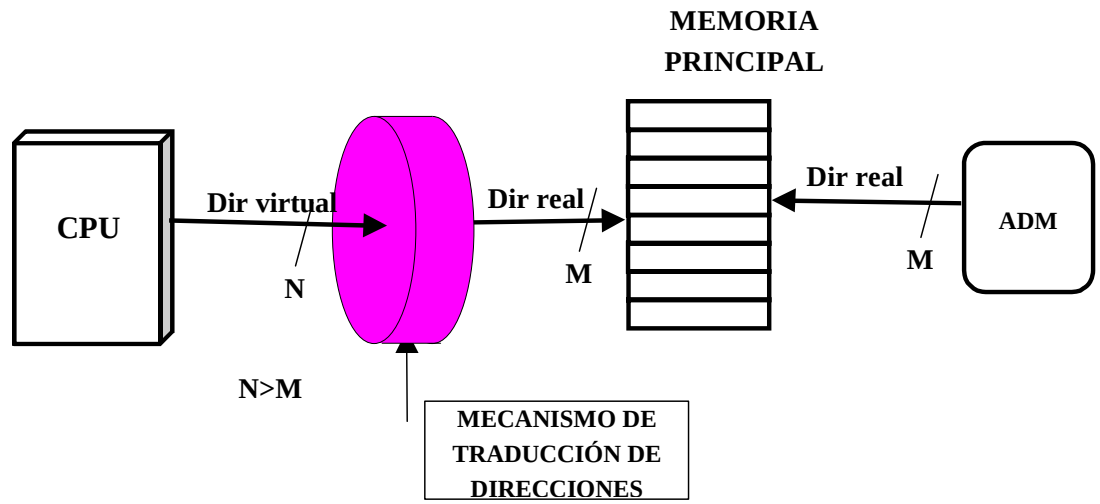
- INTENTA PREDECIR LAS PÁGINAS QUE UN PROCESO VA A NECESITAR
- Y LAS PRECARGA CUANDO HAY ESPACIO DISPONIBLE
- ASPECTOS:
  - CUANDO HACER LA PREBÚSQUEDA
  - QUÉ PÁGINAS DEBEN PREBUSCARSE
  - QUE ESTADO DE REMPLAZAMIENTO SE LES DEBE ASIGNAR
- SI LAS DECISIONES SON ACERTADA PUEDE REDUCIR NOTABLEMENTE EL TIEMPO DE EJECUCIÓN DELPROGRAMA
- OBL
  - ONE BLOCK AHEAD
  - UNA PÁGINA Y LA SIGUIENTE
  - SE BASA EN LA LOCALIDAD ESPACIAL
  - FUNCIONA BIEN CON BUENA SECUENCIALIDAD DE DATOS

MEMORIA VIRTUAL Y EL ADM

- **SIN DMA**
  - Todos los acceso a mp a través del mecanismo de traducción de direcciones

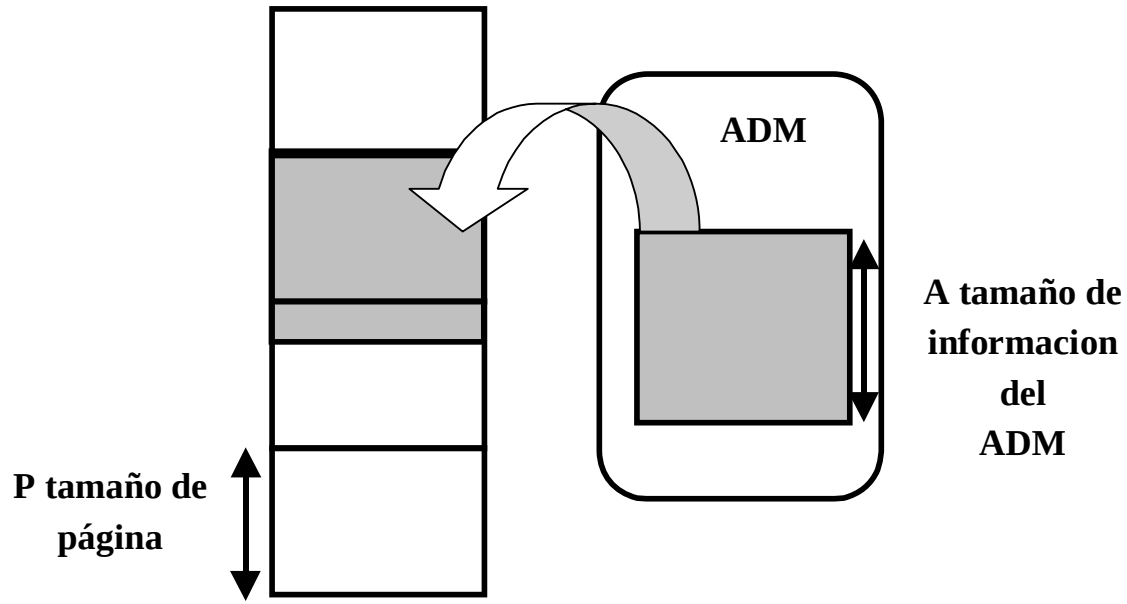


- **CON DMA**
  - Aparece una camino de acceso a mp alternativo



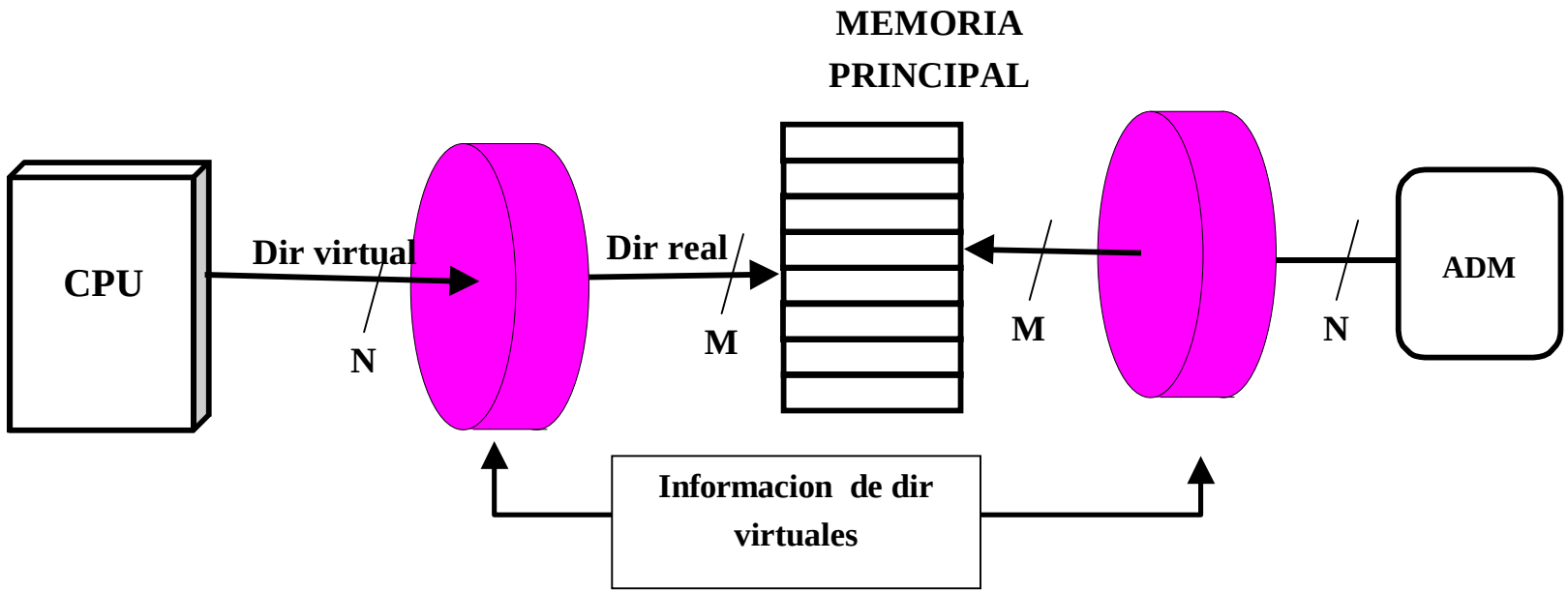
MEMORIA VIRTUAL Y EL ADM

- **DMA DE DIRECCIÓN FÍSICA**
  - TAMAÑOS DE TRANSFERENCIA SUPERIORES A UNA PÁGINA
  - SOLUCIÓN NO PERMITIR TAMAÑOS MAYORES



MEMORIA VIRTUAL Y EL ADM

- **DIRECCIONES VIRTUALES**
  - EL DMA TIENE UN PEQUEÑO NÚMERO DE ENTRADAS DE MEMORIA QUE PROPORCIONA EL MECANISMO DE TRADUCCIÓN DE DIRECCIONES PARA LA TRANSFERENCIA
  - EL SISTEMA OPERATIVO ES EL ENCARGADO DE SUMINISTRARLAS CUANDO SE INICIA UNA I/O



MEMORIA VIRTUAL Y MEMORIA CACHE

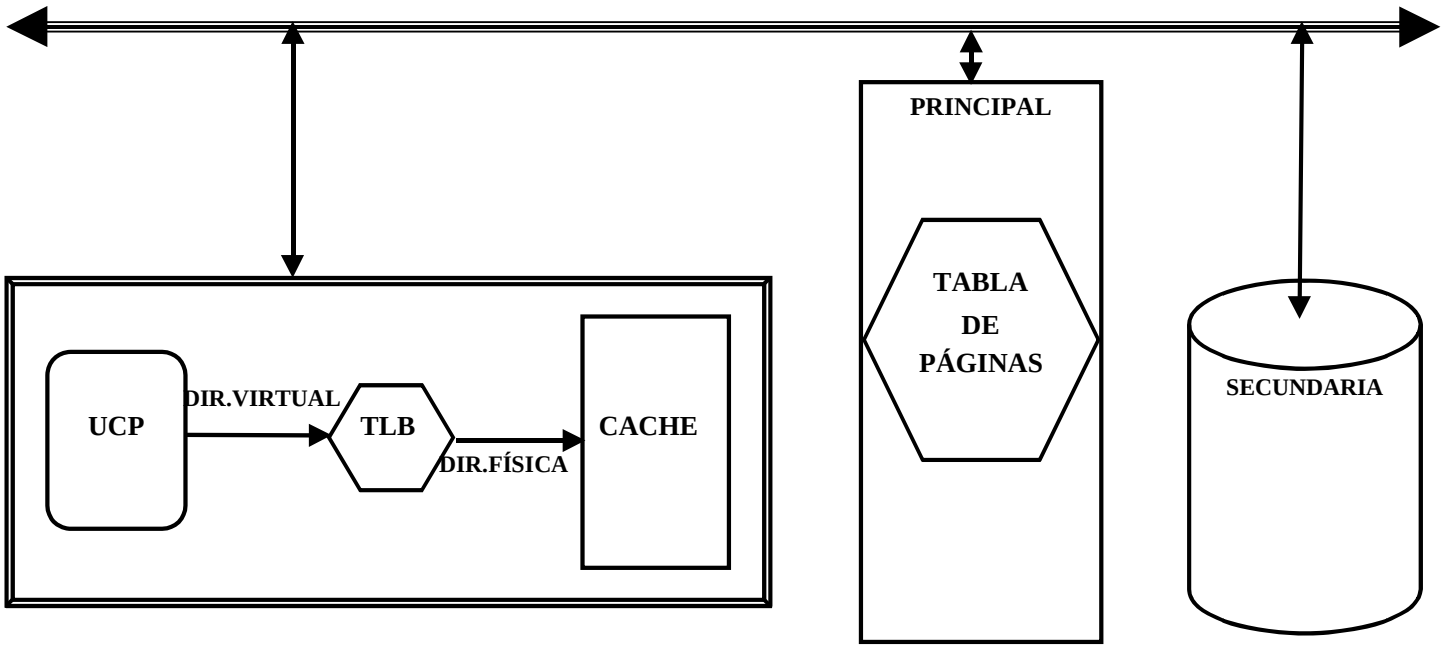
COINCIDENCIAS Y DIFERENCIAS

- **GESTIONAN DIFERENTES NIVELES DE JERARQUÍA**
- **LOS PRINCIPIOS COINCIDEN**
  - GUARDAR LOS BLOQUES ACTIVOS EN LA MEMORIA DE MAYOR VELOCIDAD
  - GUARDAR LOS INACTIVOS EN LA MEMORIA MAS LENTA
  - SI EL ALGORITMO TIENE ÉXITO:
    - » EL RENDIMIENTO SE APROXIMA AL DE LA MEMORIA MÁS RÁPIDA
    - » EL COSTE A LA MAS BARATA
- **DIFERENCIA EN LA PENALIZACIÓN DE LOS FALLOS**
  - PERDIDA DE CACHE UNA PENALIZACIÓN DE 4 A 20 VECES EL TIEMPO DE ACCESO A LA CACHE
  - FALLO DE PÁGINA COSTER ENTRE 1000 Y10000 VECES EL ACCESO A UNA MEMORI A PRINCIPAL
  - ESTA PENALIZACIÓN AUMENTA: MEMORIAS SEMICONDUCTIORAS MÁS RÁPIDAS MIENTRAS EL ACCESO A DISCO CONSTANTE
- **EFECTO SOBRE LAS ESTRATEGIAS DE ADMINISTRACIÓN**
  - EN CACHE
    - » ALGORITMO HW
    - » NO SE CAMBIA DE TAREA
  - EN VIRTUAL
    - » ALGORITMOS SW
    - » CAMBIO DE TAREA

DIRECCIÓN VIRTUAL Y MEMORIA CACHE

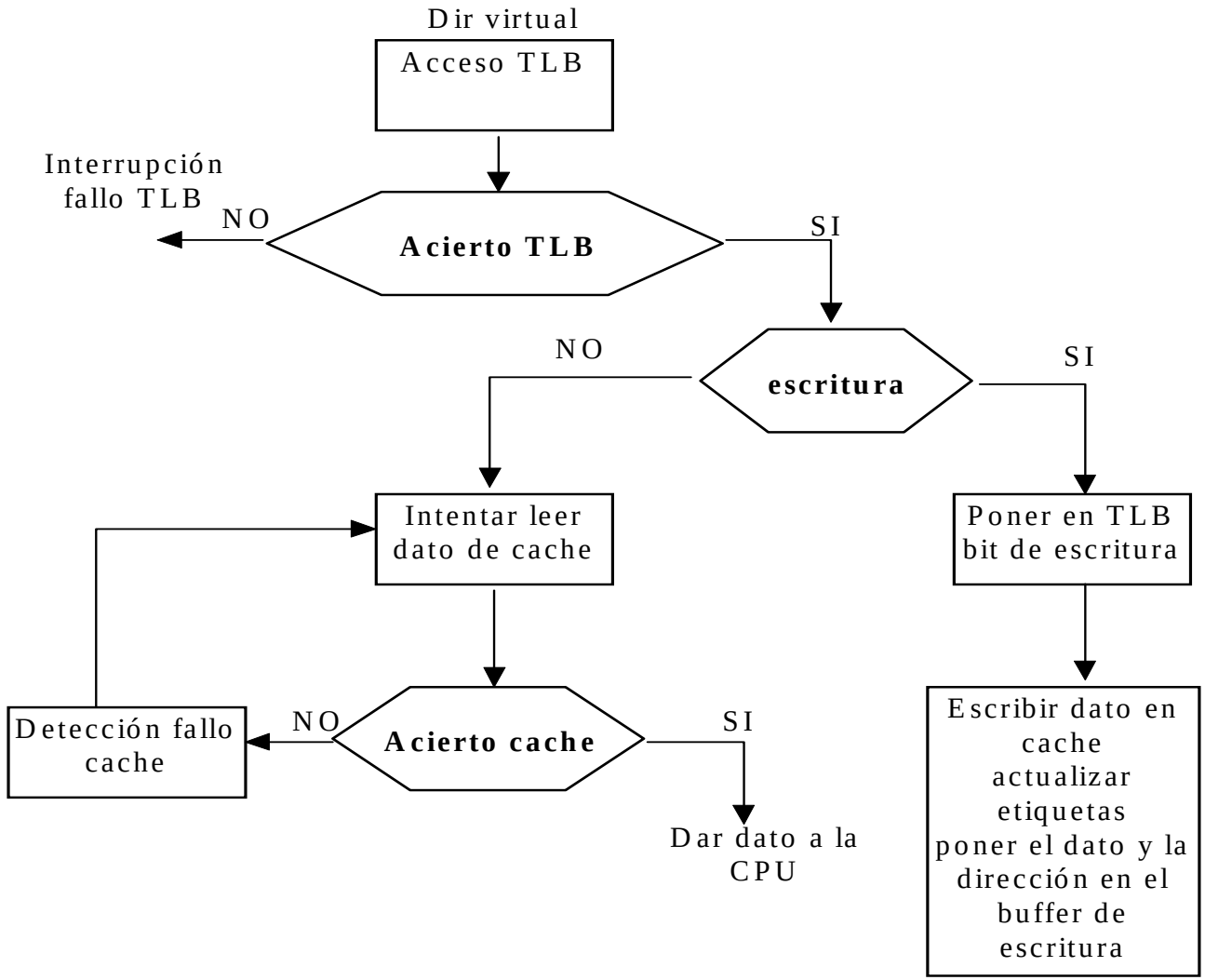
CACHE FÍSICA

- PERMITE QUE EN LA MEMORIA CACHE CONVIVAN VARIOS CONTEXTOS ES DECIR VARIOS PROCESOS ACTIVOS.
- RETARDOS EN LOS ACCESOS A CACHE PUESTO QUE HAY QUE TRADUCIR LA DIRECCION

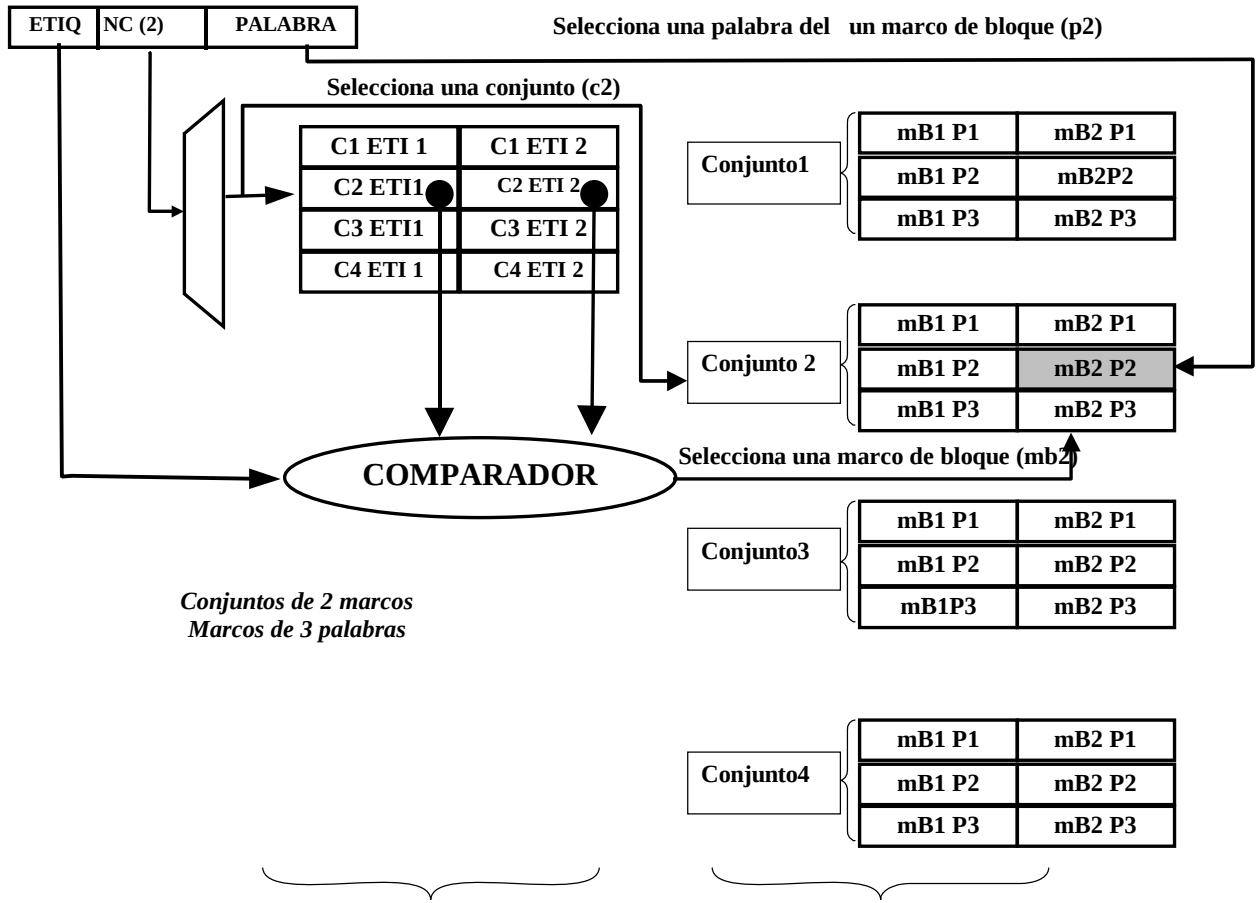


DIRECCIÓN VIRTUAL Y MEMORIA CACHE

CACHE FÍSICA



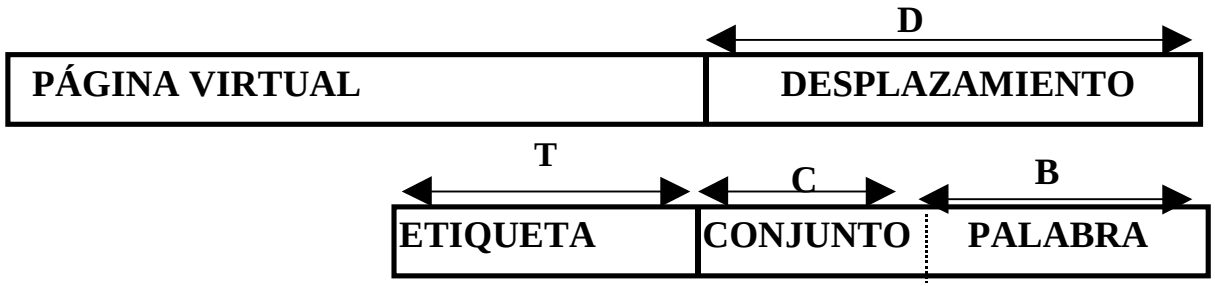
Cache virtual
Cache asociativa por conjuntos



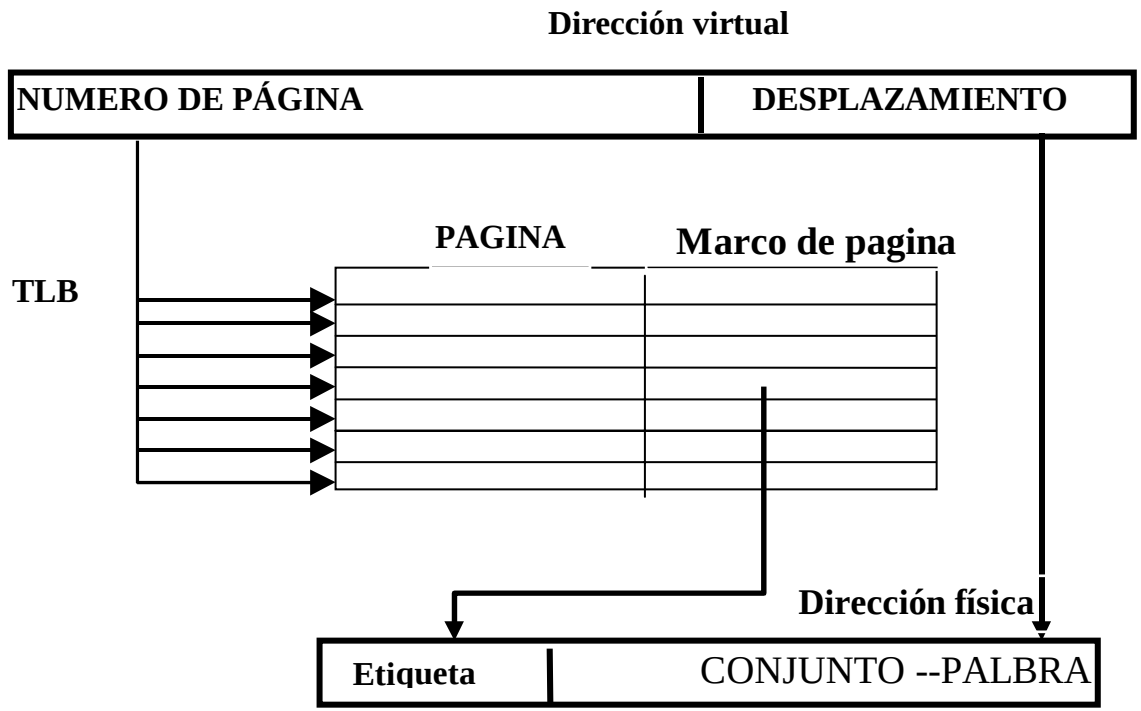
DIRECCIÓN VIRTUAL Y MEMORIA CACHE

CACHE VIRTUAL

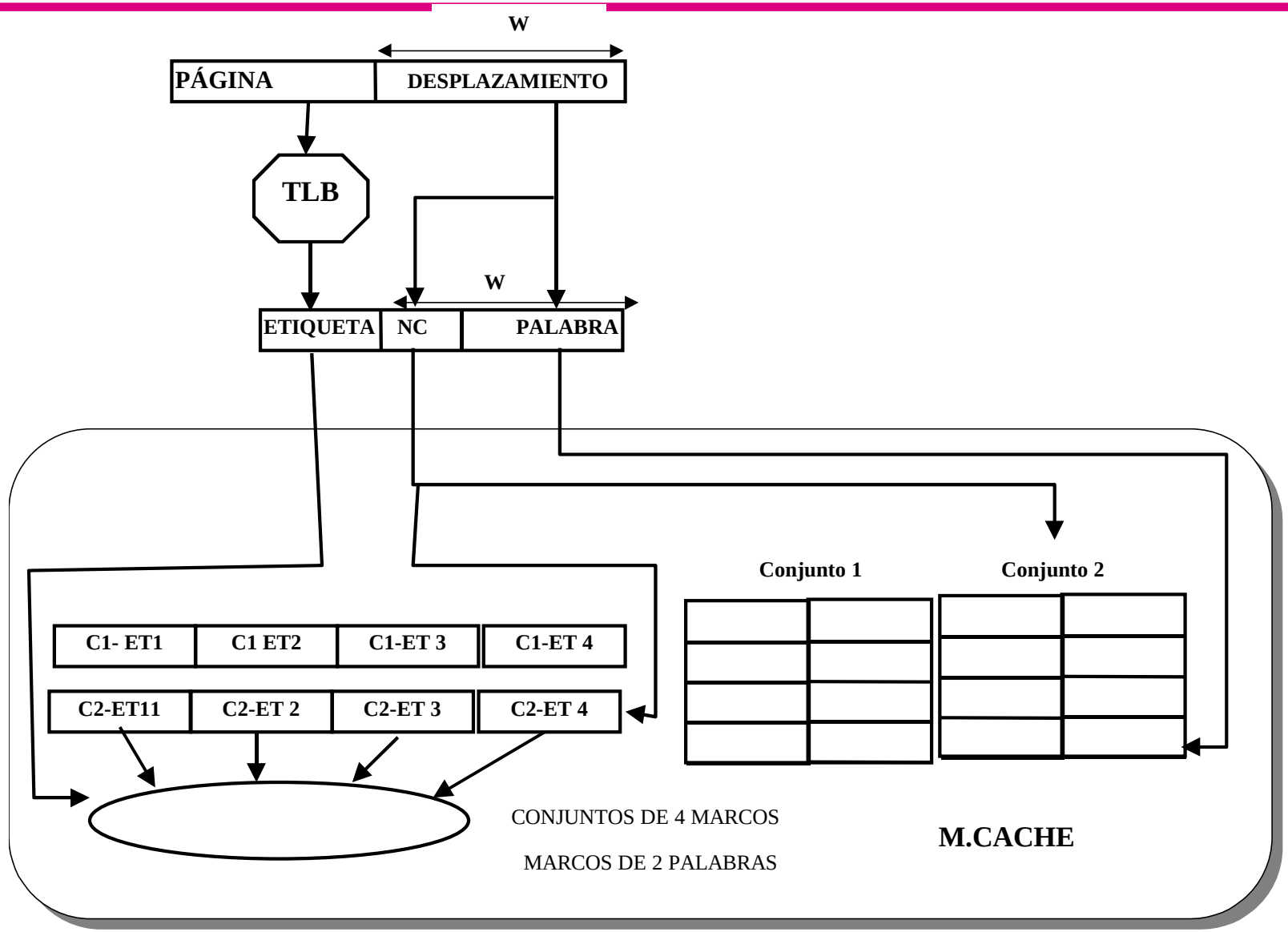
- **DIRECCIONAMIENTO VIRTUAL PARCIAL**
- **SE REALIZA SIMULTÁNEAMENTE**
  - La búsqueda del bloque en la cache
  - La traducción virtual
- **MODO DE OPERACIÓN**
  - Mientras se traduce la página a través de la tabla de direcciones
    - » Convierte página virtual en etiqueta con la que comparar en la cache
  - El desplazamiento de la dirección virtual (cte en la dirección real) se utiliza para seleccionar
    - » El conjunto
    - » la misma palabra de cada marco de bloque del conjunto
- **LA CONDICION ES D=C+B**



CACHE VIRTUAL

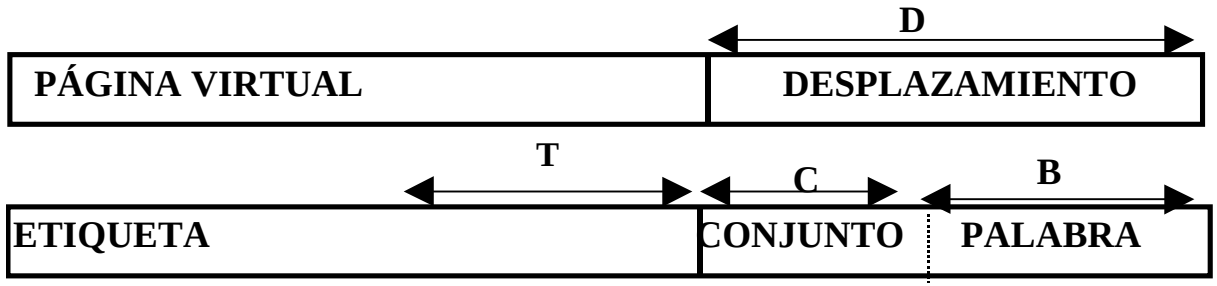


CACHE VIRTUAL PARCIAL

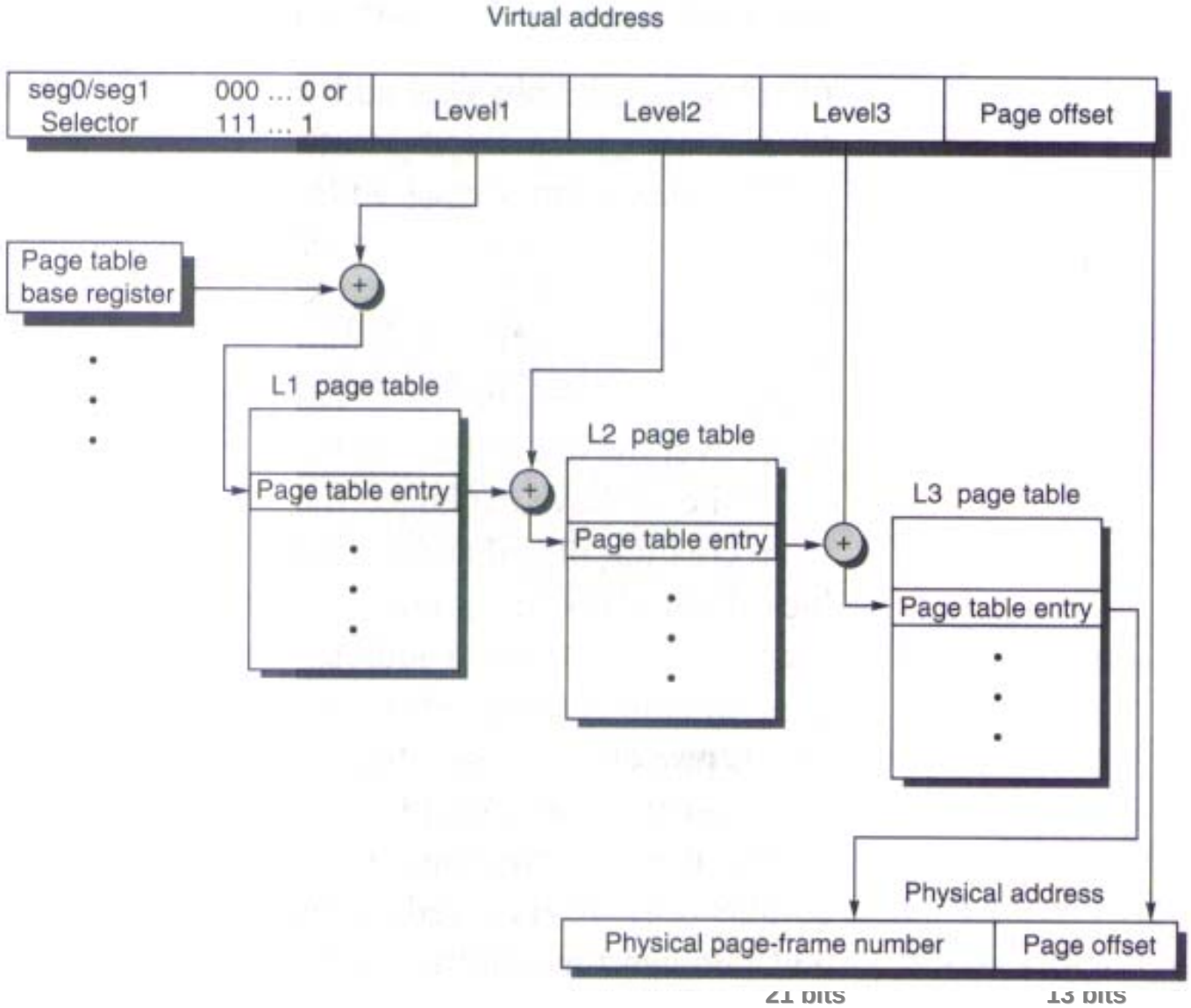


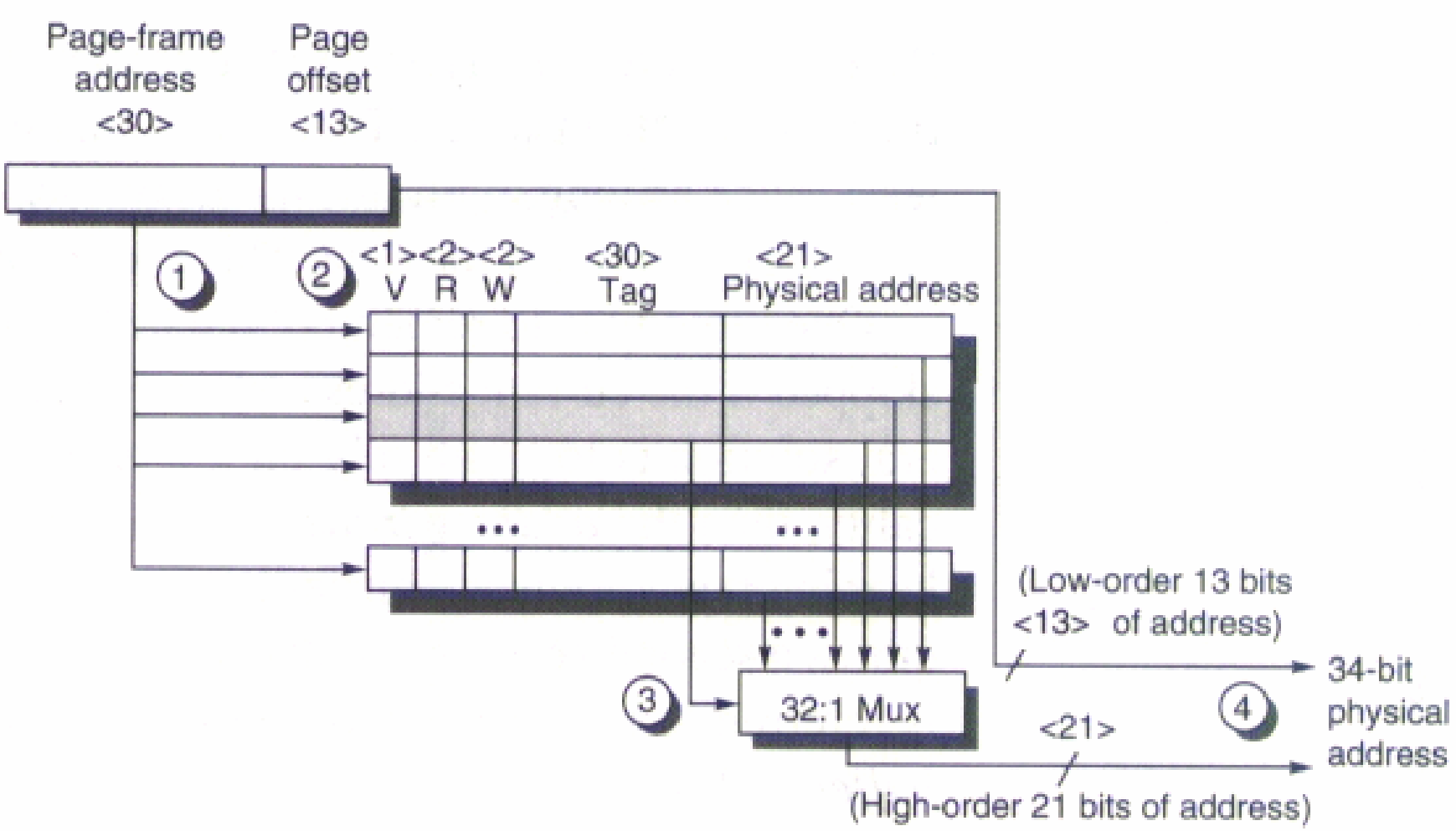
CACHE VIRTUAL TOTAL

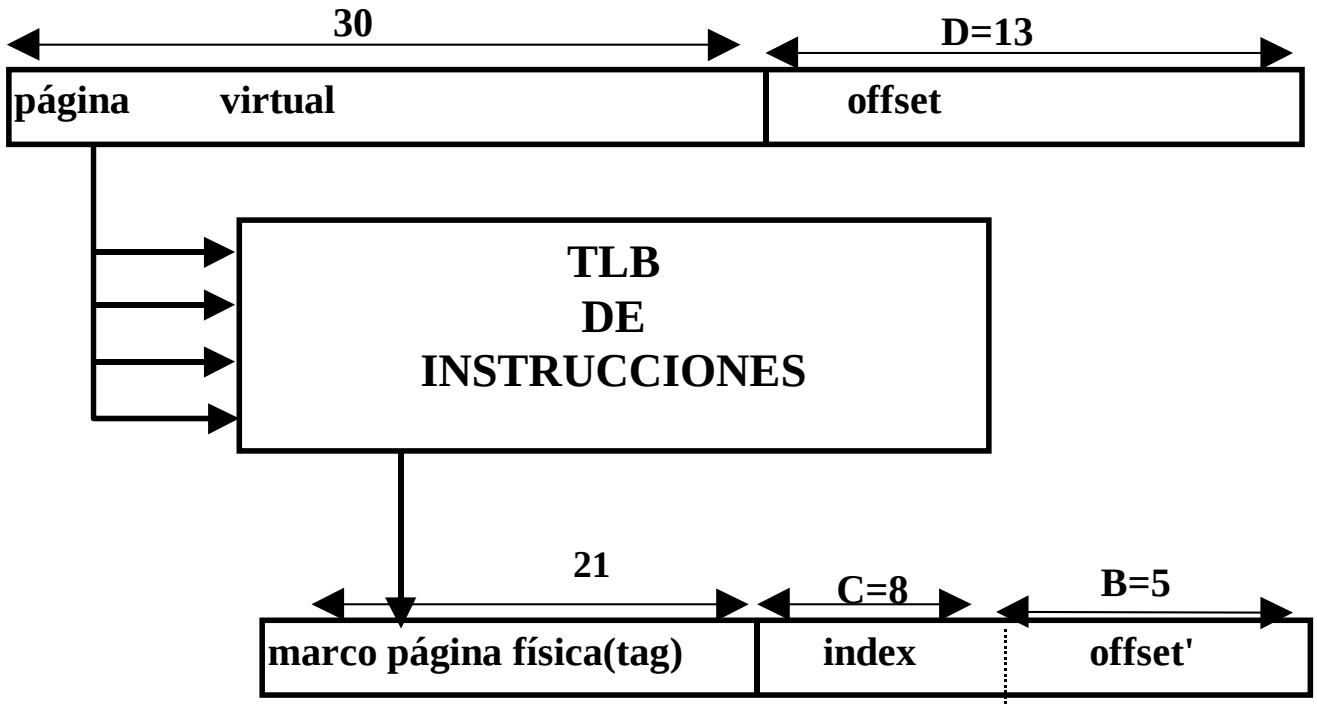
- **se utilizan las páginas proporcionadas por la dirección virtual como tag de la dirección de cache.**
- **los tamaños de dirección virtual y cache deben coincidir.**
- **Problemas:**
  - » Como pueden existir procesos diferentes con la misma dirección virtual, --> la misma dirección virtual tenga diferentes direcciones físicas.
- **solución**
  - es realizar un vaciado de la cache cada vez que se produce un cambio de contexto.
- **Inconveniente.-**
  - perdida de tiempo, degradación del rendimiento.
- **Otra solución es aumentar el tamaño del tag de la dirección de cache añadiéndole un identificador de proceso (PID).**



- Problema de los sinónimos (o alias):
  - Aparece cuando varios procesos comparten información,
  - Puede existir el mismo dato, en diferentes bloques de cache, cada uno de ellos para un proceso diferente.
  - **Problemas** de coherencia de cache, cuando un proceso actualiza un dato, y el resto de los procesos no se da cuenta.
  - **Solución** es evitar que existan varias copias del mismo dato mediante la detección de los sinónimos cuando se produzca.
  - Cuando la dirección virtual se traduce a dirección física a través del TLB es fácil evitar que existan sinónimos, con solo dar las misma dirección física a las diferentes direcciones virtuales,
  - esto no es posible si la cache utiliza directamente la dirección virtual.
  - La única solución es realizar la traducción física en paralelo al acceso a la cache.


Virtual address
seg0/seg1 Selector
000 ... 0 or 111 ... 1
Level1
Level2
Level3
Page offset
Page table base register
L1 page table
Page table entry
L2 page table
Page table entry
L3 page table
Page table entry
Physical address
Physical page-frame number
Page offset
21 bits
13 bits


Page-frame address <30>
Page offset <13>
1
2
<1><2><2>
V R W
<30>
Tag
<21>
Physical address
(Low-order 13 bits <13> of address)
34-bit physical address
3
32:1 Mux
4
<21>
(High-order 21 bits of address)


30
D=13
página virtual
offset
TLB
DE
INSTRUCCIONES
21
C=8
B=5
marco página física(tag)
index
offset'

LA ENTRADA-SALIDA

TEMA 14

INTRODUCCIÓN

- **Problema:**
  - **La conexión directa de la CPU los periféricos disminuye el rendimiento**
- **Solución:**
  - **Se utilizan unos interfaces**
  - **La CPU envía la información a los interfaces y estos se conectan con los periféricos.**
- **Inconveniente**
  - **La CPU debe estar en constante conexión con los interfaces de entrada/salida**
    - » **Perdida de ciclos de CPU**
- **Solución:**
  - **Crear dispositivos cada vez mas sofisticados que se encargan de conectarse con los interfaces**
  - **El caso límite en que se añaden procesadores de entrada/salida con su propia memoria**
  - **Gracias a esto se descarga de trabajo a la CPU para que siga trabajando en los suyo sin preocuparse de las entradas/salidas.**

INTRODUCCIÓN

- **Diferencias entre los sistemas de entrada/salida y los procesadores:**
  - **Los diseñadores de procesadores se centran en mejoras del rendimiento**
- **Los sistemas de entrada/salida se deben tener en cuenta:**
  - **Expandibilidad del sistema**
  - **Recuperación de fallos**
  - **Diversidad de periféricos**
  - **Finalidad del sistema**

MEDIDAS DEL RENDIMIENTO

- El modo de evaluar el rendimiento de la entrada/salida depende de la aplicación del sistema
  - Productividad
  - Tiempo de respuesta
  - Ambas a la vez
- La productividad
  - El ancho de banda
  - Dos orientaciones diferentes:
    - » Total de datos que se pueden intercambiar en un determinado tiempo
    - » Total de operaciones de entrada salida en un determinado tiempo
    - » Por ejemplo:
      - En cálculos científicos una sola E/S maneja gran cantidad de datos.
        - En este caso lo que cuenta es la anchura de banda de la transferencia
      - Bases de datos gran cantidad de pequeños accesos independientes entre sí
- El tiempo de respuesta del computador:
  - Y el sistema de entrada salida con latencia más baja logrará el mejor tiempo de respuesta
  - Ej: monousurario
- Un gran nº de aplicaciones necesitan alta productividad y corto tiempo de respuesta.
  - EJ cajeros automáticos.
  - Cuanto tarda cada tarea y cuantas tareas simultáneamente pueden procesarse

MEDIDAS DEL RENDIMIENTO

- El Tiempo de respuesta y la productividad son inversamente proporcionales
  - Para conseguir pequeños tiempos de respuesta
    - » Suele procesarse las entradas salidas en cuanto llega la petición
      - Se pierde productividad
  - Para optimizar la productividad
    - » Suelen agruparse las peticiones de entradsalida próximas,
      - Con lo que se aumentan los tiempos de respuesta

MODULOS DE ENTRADA/SALIDA

- Se conecta a un bus del sistema
- Controla uno o mas periféricos
- Encargado de llevar el peso de la transferencia en diversos grados
- Contiene la lógica necesaria para permitir la comunicación entre el periférico y el bus
- ¿Porque no se conecta un periférico directamente al bus del sistema:
  - Diversidad de periféricos.
    - » Habría que incorporar demasiada lógica a la CPU
  - La velocidad de transferencia de periféricos mucho menor que la de la CPU
  - Diferencia de formatos
- El módulo de E/S debe tener una interfaz interna con la CPU y la Memoria y una interfaz externa con los periféricos

MODULOS DE ENTRADA/SALIDA

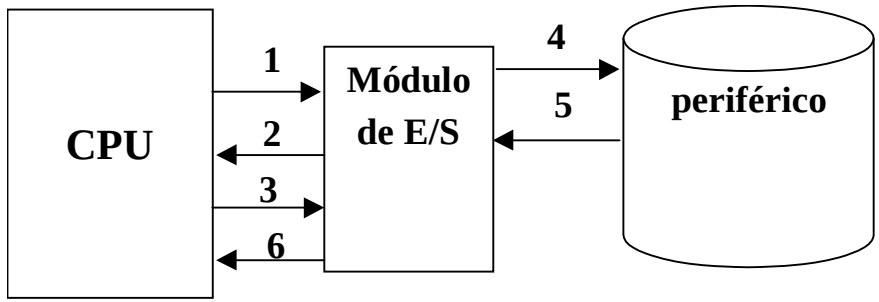
- **FUNCIONES DEL MÓDULO:**
  - **Comunicación con la CPU**
  - **Comunicación con los dispositivos**
  - **Control y temporización**
  - **Almacenamiento temporal de datos**
  - **Detección de errores**

MODULOS DE ENTRADA/SALIDA

- **Comunicación del módulo de e/s con la CPU**
  - Descodificación de ordenes acepta ordenes de la CPU
  - Intercambian datos
  - Intercambian información de estado. Puesto que los periféricos son lentos es importante conocer su estado
  - Reconocimiento de dirección un módulo puede reconocer una única dirección para cada periférico.

MODULOS DE ENTRADA/SALIDA
Control y temporización

- 1. La CPU pregunta al módulo de ES por el estado del periférico
- 2. El módulo responde a la CPU
- 3. Si el periférico esta operativo y preparado la CPU solicita la transferencia al módulo
- 4. El módulo de E/S solicita un dato del periférico
- 5. El periférico envía el dato al módulo
- 6. Los datos se transfieren del módulo a la CPU
- Cada interacción entre CPU y módulo requiere un arbitraje del bus.

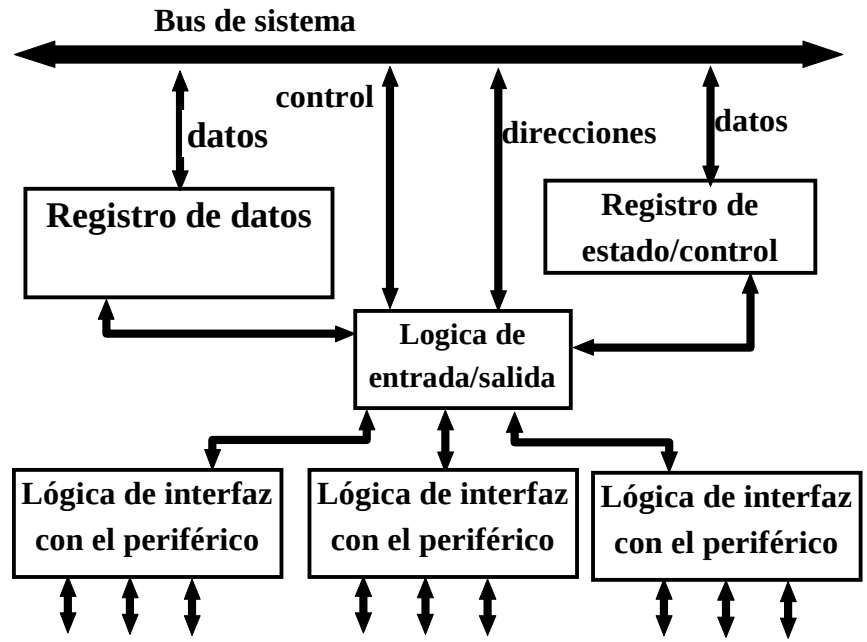


MODULOS DE ENTRADA/SALIDA
Almacenamiento temporal de datos (buffer)

- Utilizada para sincronizar las velocidades de la CPU y de los periféricos.
- Los datos se envían en ráfagas al periférico,
  - para aprovechar la velocidad de comunicación del procesador,
- se almacenan temporalmente en el módulo de entrada/salida
- se envían al periférico a la velocidad de este.
- Cuando los datos se envían del periférico a la unidad central de proceso los datos se almacenan en el buffer
  - para no mantener la memoria ocupada en transferencias demasiado lentas.

MÓDULO DE ENTRADA/SALIDA
Estructura del modulo

- Se conecta al computador a través del bus de sistema.
- Los datos que se envían o reciben se almacenan temporalmente en un registro interno
- El bloque de lógica
  - Interactúa con la CPU
  - Genera las señales de control de los dispositivos internos
- Debe ser capaz de generar y reconocer las direcciones de los tres dispositivos.
- Lógica especifica de interfaz con cada dispositivo
- Un módulo permite que la CPU vea una amplia gama de dispositivos de una forma simplificada.
- La ES debe ocultar
  - Los detalles de temporización
  - Formatos
  - Electromecánica de los dispositivos externos
- Para que la CPU funcione en término de ordenes de lectura escritura y de abrir y cerrar ficheros



TIPOS DE MÓDULOS DE ENTRADA/SALIDA

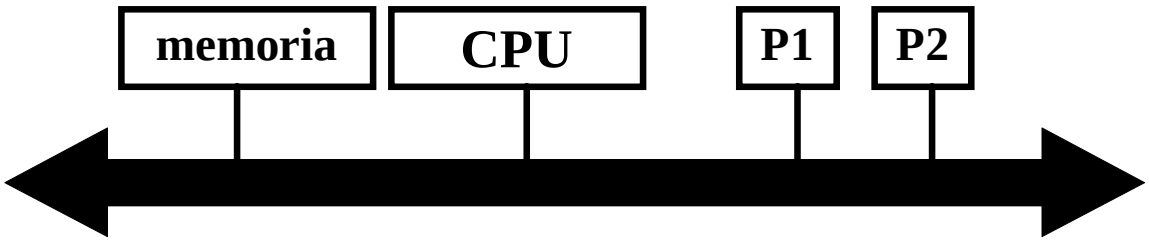
- **PROCESADOR DE ES O CANAL DE ES**
  - **Modulo de ES que se encarga de la mayoría de los detalles de procesamiento, presentando a la CPU una interfaz de alto nivel.**
  - **Aparecen normalmente en grandes computadores**
- **CONTROLADOR DE DISPOSITIVO**
  - **Un módulo simple que requiera un control detallado se llama controlador de dispositivo de E/S**
  - **En microcomputadores**

ORGANIZACIÓN DE LA ENTRADA/SALIDA

- **Existen dos organizaciones:**
  - **E/S localizada en memoria**
  - **ES aislada o independiente**

ORGANIZACIÓN DE LA ENTRADA/SALIDA
LOCALIZADA EN MEMORIA

- El computador considera las direcciones de Entrada/salida como direcciones de memoria
- No diferencia entre Memoria principal y la entrada/salida.
- Generalmente la ES se agrupa en una zona bien definida de la memoria principal.
- ¿Como se realiza el direccionamiento?
  - Con P bits se direccionan 2^P periféricos diferentes.
  - A estas 2^P direcciones se le llama mapa de entrada salida.
- Las operaciones de entrada salida son operaciones de movimiento de datos
  - un operando es un registro de la CPUo una posición de memoria y
  - el otro operando es un registro del módulo de entrada/salida.
- La dirección se descodifica para convertirla en una señal de selección del dispositivo.
  - Esto se puede hacer de dos formas:
    - » Un solo descodificador que genera las 2^P señales de selección
    - » Un reconocedor de dirección en cada puerto de entrada/salida



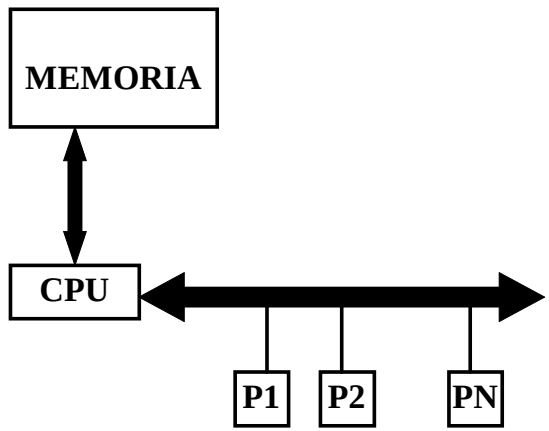
ORGANIZACIÓN DE LA ENTRADA/SALIDA
LOCALIZADA EN MEMORIA

- **modo 1.-**
  - relación biunivoca.
  - A cada dirección le corresponde un periférico
  - a cada periférico le corresponde una sola dirección
- **modo2**
  - a cada dirección le corresponde un solo periférico
  - a cada periférico le corresponde 32 direcciones
- **modo3 .-**
  - Cada periférico tiene 128 direcciones
  - una dirección direcciona varios periféricos (tanto como 1's incluya)
- **Las dos últimas soluciones**
  - desaprovechan la capacidad de direccionamiento,
  - si no existen más periféricos reducen el coste total del sistema
- **La ultima solución**
  - no necesita Descodificador
    - » luego es la más económica.
  - Desventaja
    - » si se da por error una dirección que selecciona varios periféricos, se produce una colisión en el bus

Direccion	modo 1	modo2	modo3
0	0000 0000	xxxxx 000	xxxxxxx1
1	0000 0001	xxxxx 001	xxxxxx1x
2	0000 0010	xxxxx 010	xxxxx1xx
3	0000 0011	xxxxx 011	xxxx1xxx
4	0000 0100	xxxxx 100	xxx1xxxx
5	0000 0101	xxxxx 101	xx1xxxxx
6	0000 0110	xxxxx 110	x1xxxxxx
7	0000 0111	xxxxx 111	1xxxxxxx

ORGANIZACIÓN DE LA ENTRADA/SALIDA
INDEPENDIENTE DE MEMORIA

- Mayor rigidez
- Operaciones específicas de entrada salida
- Mayor complejidad del diseñado de la cpu
- Facilita la protección de la operaciones de entrada/salida
- Menor variedad de modos de direccionamiento,
- Mapa de entrada/salida mas pequeño



TECNICAS DE ENTRADA/SALIDA

- Existen tres técnicas de entrada/salida
  - Programada
  - Interrupciones
  - Acceso directo a memoria

TECNICAS DE ENTRADA/SALIDA

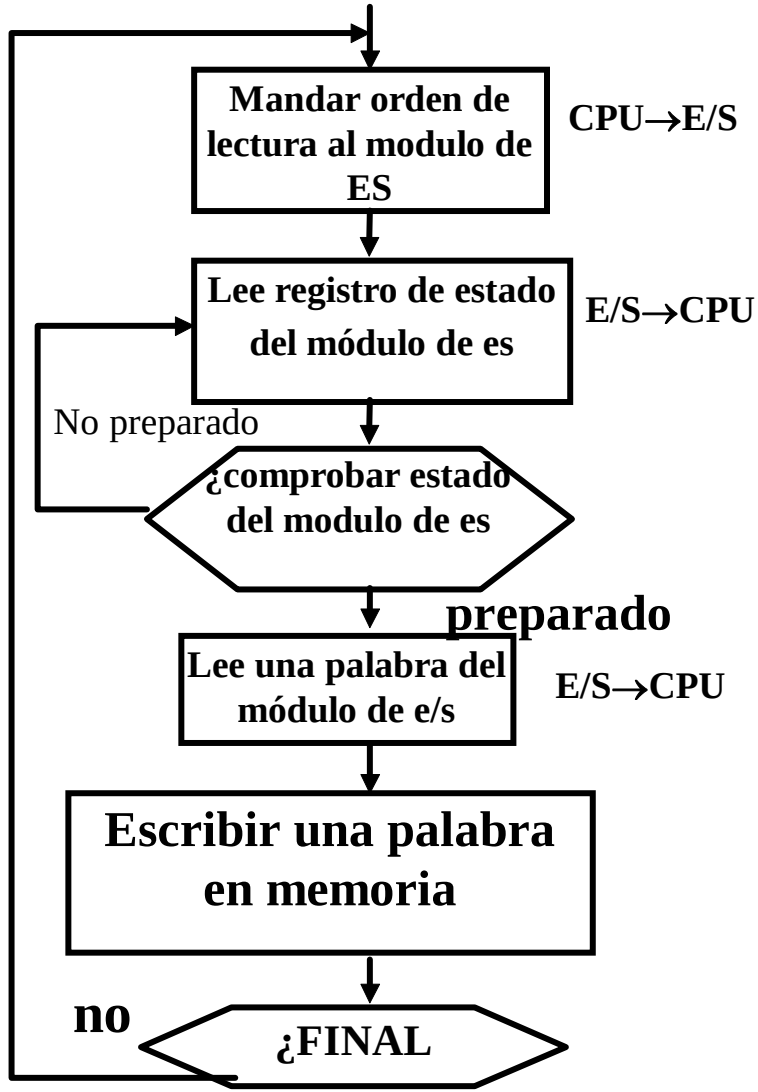
ENTRADA/SALIDA PROGRAMADA

- Tambien llamado Escrutineo Polling
- Comprobación periódica de los bits de estado de los periféricos para ver si están listos para realizar la siguiente operación
- Es la forma más simple para comunicar un dispositivo de entrada/salida y el procesador
- Modo de operación:
  - El dispositivo pone la información en el registro y modifica el bit de estado
  - El procesador comprueba que el bit de estado se ha activado
  - Si el bit está activo recoge la información del registro
- El procesador tiene totalmente el control y hace todo el trabajo
- En la actualidad el único dispositivo que se accede por escrutinio es el ratón

TECNICAS DE ENTRADA/SALIDA

ENTRADA/SALIDA PROGRAMADA

- **Inconvenientes:**
  - **Perdida de ciclos de CPU**
    - » **Durante el tiempo de espera la CPU no hace trabajo útil**
    - » **Con periféricos lentos el bucle puede repetirse miles de veces**
  - **Dificultades para atender a varios periféricos simultaneamente**
  - **Existen tareas que no pueden esperar a que acabe el ciclo de espera.**
- **Una solución parcial**
  - **Es la limitación del tiempo de espera del bucle**



TECNICAS DE ENTRADA/SALIDA
LAS INTERRUPCIONES

- **Definición:**
  - **Bifurcación externa al programa en ejecución provocada por una señal exterior a la CPU**
- **Esta señal llega a través de una línea llamada de petición de interrupción**
- **El modo general de operación:**
  - **La CPU selecciona el módulo de E/S y la operación que quiere realizar**
  - **A continuación se desentiende de la operación y se pone a trabajar en algún otro proceso**
  - **Cuando el módulo de E/S acaba la operación, interrumpe a la CPU para indicarle que ya está lista para intercambiar información con ella**
  - **La CPU realiza la transferencia de información**
  - **La CPU continua con el trabajo interrumpido**

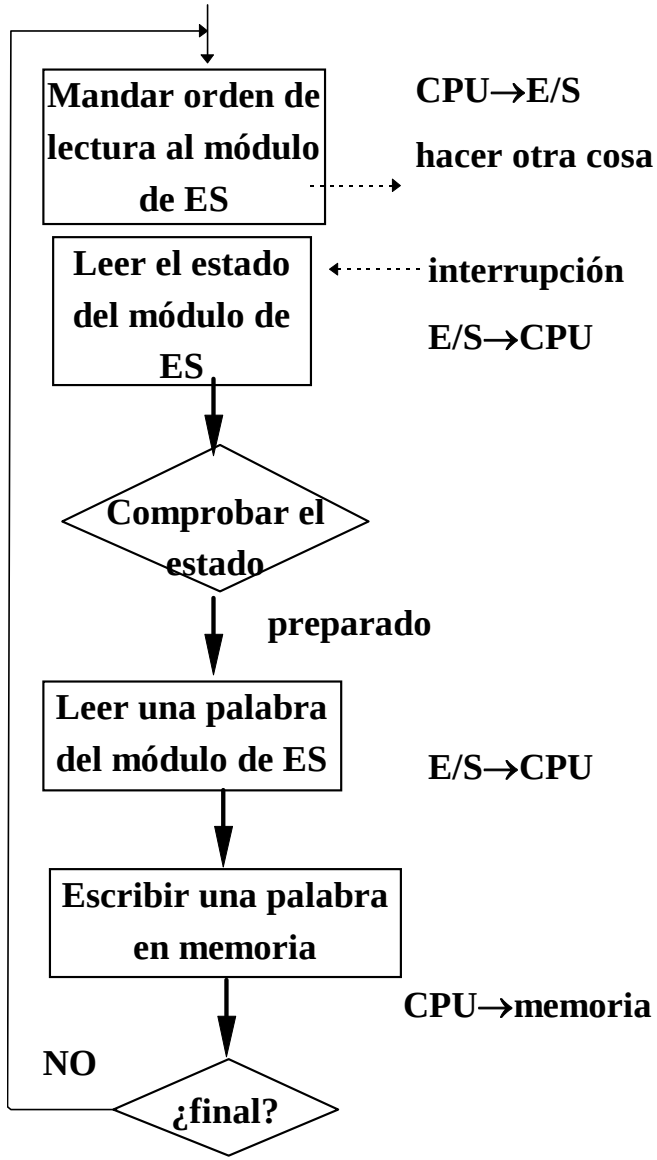
TECNICAS DE ENTRADA/SALIDA
LAS INTERRUPCIONES

- **Características**
  - **Las interrupciones hacen el tratamiento byte a byte**
    - » **Para mover 8 bytes se realizan 8 interrupciones--> 8 paradas de la CPU**
  - **Una interrupción de E/S es asíncrona**
- **Ventaja**
  - **Eliminan el ciclo de espera del procesador entre cada entrada de byte**
- **Las interrupciones resultan particularmente útiles en**
  - **Sistemas operativos**
  - **En los procesos de tiempo real**

TECNICAS DE ENTRADA/SALIDA
LAS INTERRUPCIONES

- **Rutina de servicio de interrupción**
  - **Es la que se ejecuta en respuesta a una interrupción**
  - **Similar a la una subrutina --> rompe la secuencia de ejecución del programa**
  - **Diferente --> no se sabe cuando se llevará a cabo**
  - **Puede interferir el proceso que se esté ejecutando por lo tanto hay que guardar:**
    - **El registro de estado del procesador**
    - **Los registros accesibles por programa**
  - **La CPU informa al dispositivo que su solicitud ha sido reconocida para que el dispositivo pueda eliminar la señal de petición de interrupción**

EJEMPLO DE OPERACIÓN DE ENTRADA/SALIDA MEDIANTE INTERRUPCIONES



INTERRUPCIONES

- **Inconvenientes de la E/S mediante interrupciones**
  - **Sigue realizando la transferencia a través de la CPU**
  - **Durante la transferencia la CPU queda bloqueada**
    - » **La transferencia se hace byte a byte**
      - **Debe haber una interrupción por cada byte que se transmite**
      - **Las subrutinas de tratamiento de interrupción incluyen gran cantidad de instrucciones preparatoria para que la transacción se realice sin problemas**
  - **Si el bloque de datos es muy grande, la CPU puede quedar bloqueada casi permanentemente**
- **Ventaja:**
  - **La interrupción alivia al procesador de la espera**

GESTION DE INTERRUPCIONES

- Una petición de interrupción tiene como efecto inmediato que la CPU suspenda la ejecución del programa e inicie otro.
- La aceptación de una interrupción se realiza al final de una instrucción
  - Última microinstruccion
  - Consultando el bit correspondiente
  - En las instrucciones que son muy largas se consulta en diversos puntos de la instrucción
- La CPU tiene capacidad para habilitar y deshabilitar interrupciones
  - Supongamos que las interrupciones no se pudieran deshabilitar:
    - » La petición de interrupción queda activa hasta que no llegue una señala de aceptación de interrupción
    - » Se salta a la rutina de interrupción
    - » Al final de cada instrucción de la subrutina se comprueba si existen interrupciones
    - » Como la señal de petición sigue activa se detecta la misma interrupción
      - efecto: se entra en un bucle infinito.

GESTIÓN DE INTERRUPCIONES

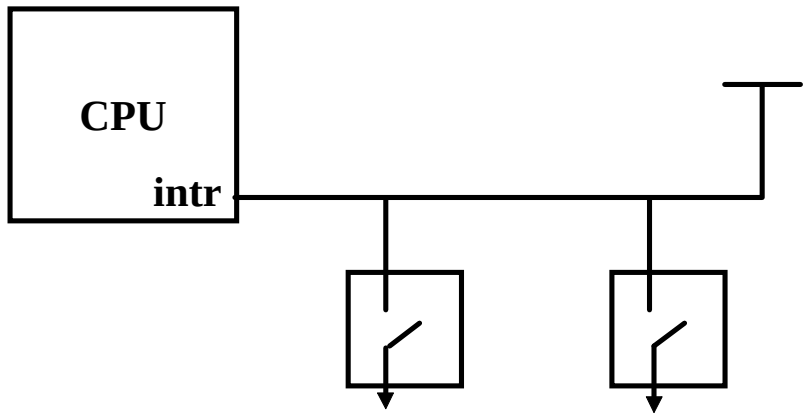
- La secuencia correcta de eventos es la siguiente:
  - El dispositivo solicita la interrupción
  - La CPU interrumpe el programa
  - Se deshabilitan las interrupciones
  - Se informa al dispositivo que se ha reconocido su solicitud
  - Se realiza la acción solicitada
  - Se habilitan las interrupciones
  - Se reanuda el programa interrumpido
- Existen tres formas de deshabilitar las interrupciones:
  - La CPU deshabilita durante la ejecución de la primera instrucción de la rutina
  - Se deshabilitan de manera automática
  - La señal de petición de interrupción funciona por flanco no por nivel

TIPO DE SISTEMAS DE INTERRUPCIONES:
DIRECCIONAMIENTOS Y PRIORIDADES

- **Direccionamiento:**
  - **Modo de seleccionar un dispositivo cuando hay varios que producen interrupciones**
  - **Existe dos técnicas:**
    - » **Encuesta**
    - » **Vector**
- **Prioridades**
  - **Varios dispositivos piden simultáneamente la interrupción**
    - » **Se debe decidir cual se atiende primero.**
  - **Existen dos técnicas ligadas al tipo de direccionamiento**
    - » **Hardware**
    - » **Software**
- **Tipos de sistemas**
  - **Con una línea de interrupción**
  - **Con varias línea de interrupción**

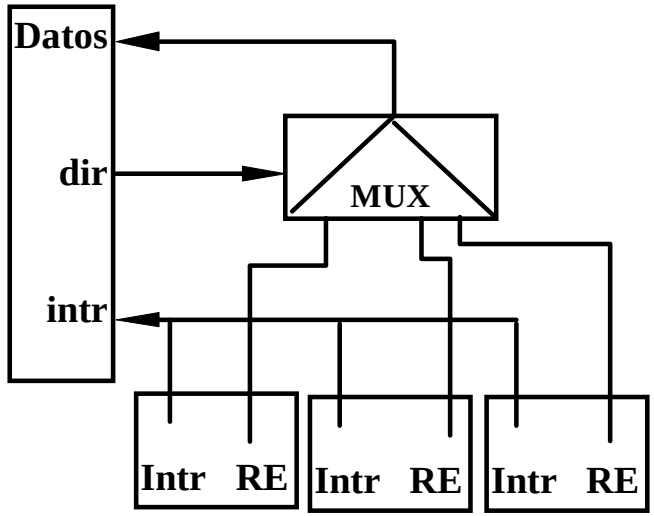
SISTEMAS DE UNA LINEA DE INTERRUPCIÓN

- Todos los periféricos comparten una única línea de interrupción
- Cuando un periférico hace una petición pone la línea a cero
- Interrupción = not(intr1)not(intr2)not(intr3)
- Identificación de la fuente. Existen dos métodos
  - Polling
  - Vector



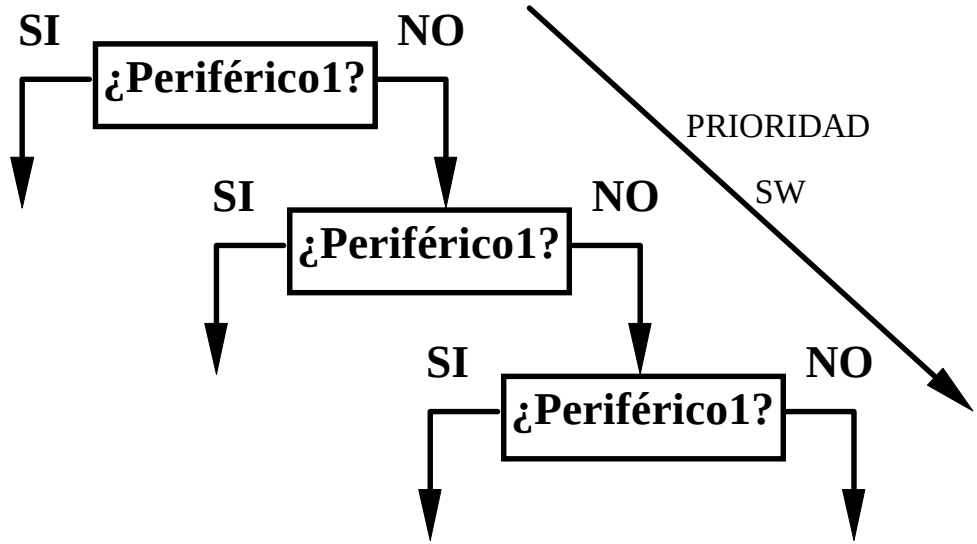
SISTEMAS DE UNA LINEA DE INTERRUPCIÓN
POLLING

- Se examina el registro de estado de cada periférico
- El orden lo determina la subrutina de interrupción
  - La prioridad es software
- Modo de operación:
  - Llega una INTR a la CPU
  - La CPU ejecuta la subrutina de interrupción
  - Se envian direcciones sucesivas por la línea dir
    - » Para cada dirección selecciona un dispositivo
    - » Para cada dispositivo se lee el registro de estado



SISTEMAS DE UNA LINEA DE INTERRUPCIÓN
POLLING

- **Desventaja:**
  - **El tiempo que se tarda en consultar el registro de cada módulo de interrupción**
  - **Si el periférico n-esimo interrumpe mucho se realizan muchas consultan hasta llegar a él lo que implica que la CPU no trabaja en profundidad**

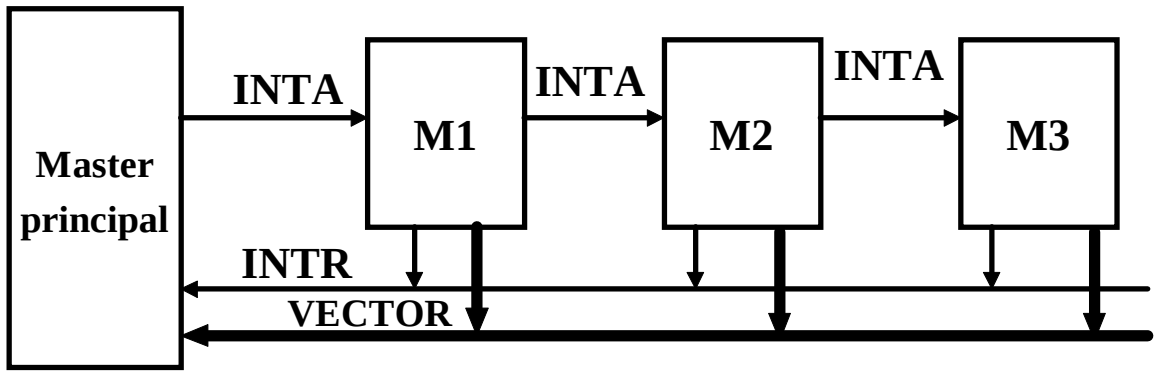


SISTEMAS DE UNA LINEA DE INTERRUPCIÓN VECTOR

- El dispositivo se identifica dando algun tipo de información:
  - el número de vector
  - la dirección de memoria de la subrutina
  - la dirección de memoria que almacena la dirección de la subrutina (dir indirecto)
- Se llama vector a la dirección de la subrutina.
- Las direcciones no tienen porque ser completas,
  - disminuye la complejidad del interfaz
  - disminuye la capacidad de direccionamiento
- Autovector cuando el periférico no da la información explícitamente de manera que el periférico tiene una dirección de memoria asignada previamente.

SISTEMAS DE UNA LINEA DE INTERRUPCIÓN VECTOR

- ¿Como se determina la prioridad?.
  - Se suele utilizar un protocolo de daisychain:
  - El periférico que provoca las interrupción pone la línea INTR a 0
  - La CPU activa la señal INTA
  - Los periféricos que no han pedido interrupciones dejan pasar la señal INTA
  - El periférico que ha interrumpido impide el paso de las señal INTA y envía su código a la CPU

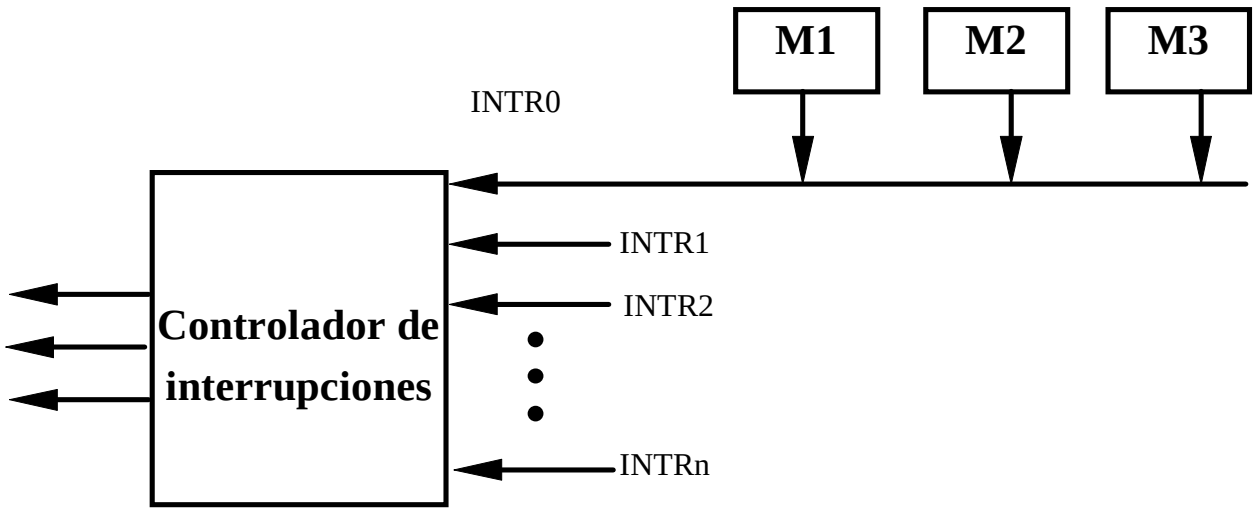


SISTEMAS DE UNA LINEA DE INTERRUPCIÓN VECTOR

- La prioridad es Hardware,
  - depende de la posición del periférico en la cadena:
- Problema
  - Cuando el periférico da directamente la dirección de la rutina de interrupción el sistema es demasiado rígido
    - » obligado a guardar siempre en la misma dirección de memoria la subruitna
- solución
  - utilizar un direccionamiento indirecto
  - El sistema es más fiable
  - En algunas ocasiones el vector de interrupción contiene valores para el registro de estado de la máquina.
    - » para modificar el nivel de prioridades
    - » para deshabilita las interrupciones posteriores.

SISTEMAS DE INTERRUPCIÓN MULTILINEA

- Cuando existen varias líneas de petición de interrupción
- En cada línea pueden estar conectados varios periféricos
- La forma de direccionar es sencilla.
  - La activación de la línea de interrupción en si misma indica la dirección
- Prioridades
  - las peticiones simultáneas en líneas diferentes se resuelve mediante un codificador de prioridades.
  - La peticiones simultáneas en la misma línea como se ha estudiado ya.



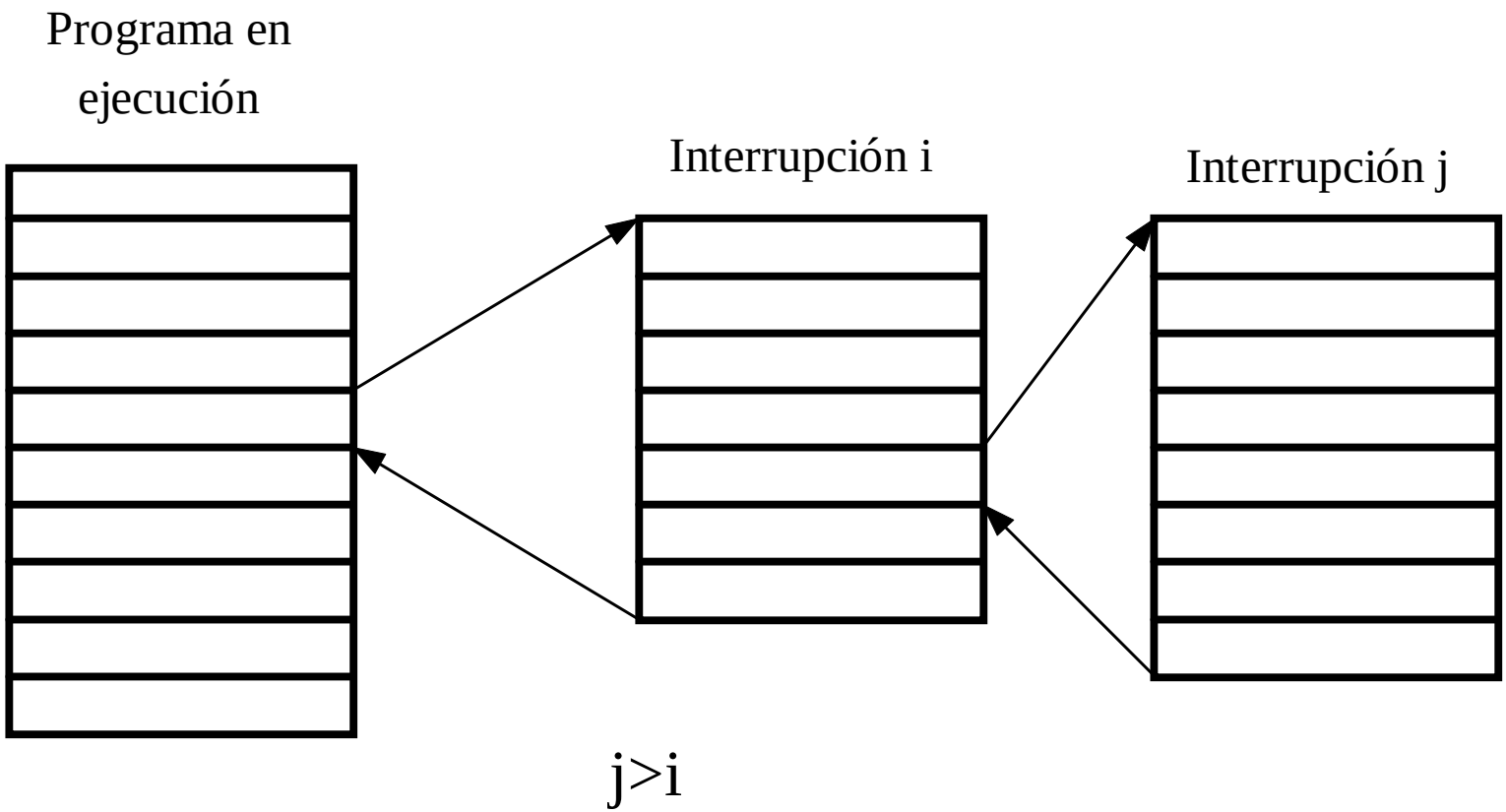
JERARQUÍA DE INTERRUPCIONES

- ¿Qué ocurre cuando se está tratando una interrupción y llega otra?
- Existen dos métodos complementarios
  - Interrupciones anidadas
  - Enmascaramiento de interrupciones
- Dependen de la prioridad de las interrupciones:

JERARQUÍA DE INTERRUPCIONES
INTERRUPCIONES ANIDADAS

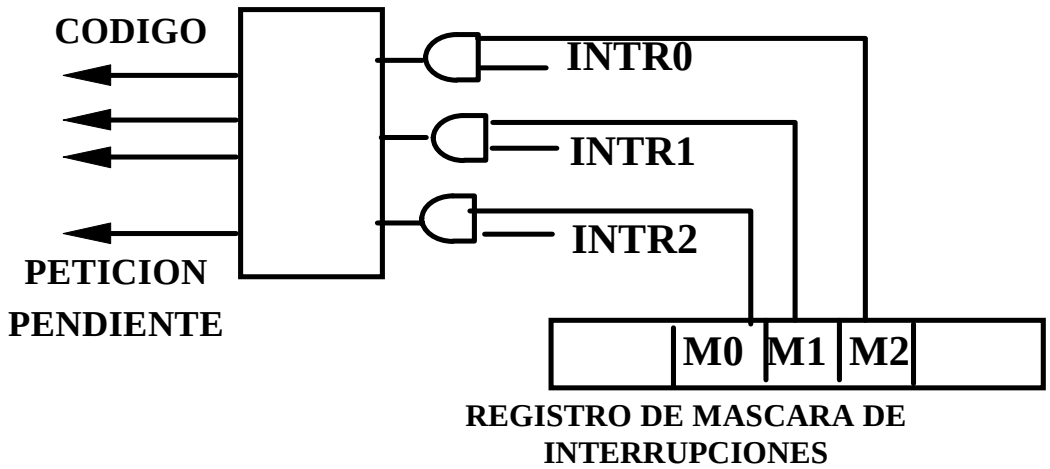
- Suponer una jerarquía de interrupciones I1, I2, I3,…, In
- Mayor prioridad la de mayor índice
- Modo de operación de las interrupciones anidadas
  - Se está procesando a interrupción Ii
  - LLega una petición de interrupción Ij
  - Si j>i se interrumpe la rutina de Ii para ejecutarse las Ij
- Problema
  - Lo primero que hace el tratamiento de una interrupción es deshabilitar las interrupciones para evitar entrar en bucles infinitos
  - Existen interrupciones que no pueden esperara hasta que se acaba de procesar otra interrupción
    - » ej la señal de reloj del computador
- Solución:
  - Pueden interrumpir las interrupciones de mayor nivel de prioridad
- Normalmente se asigna cierto nivel de prioridad a la CPU (programa en ejecución)
  - La prioridad de la CPU es parte del registro de estado y se puede controlar por programa

JERARQUÍA DE INTERRUPCIONES
INTERRUPCIONES ANIDADAS

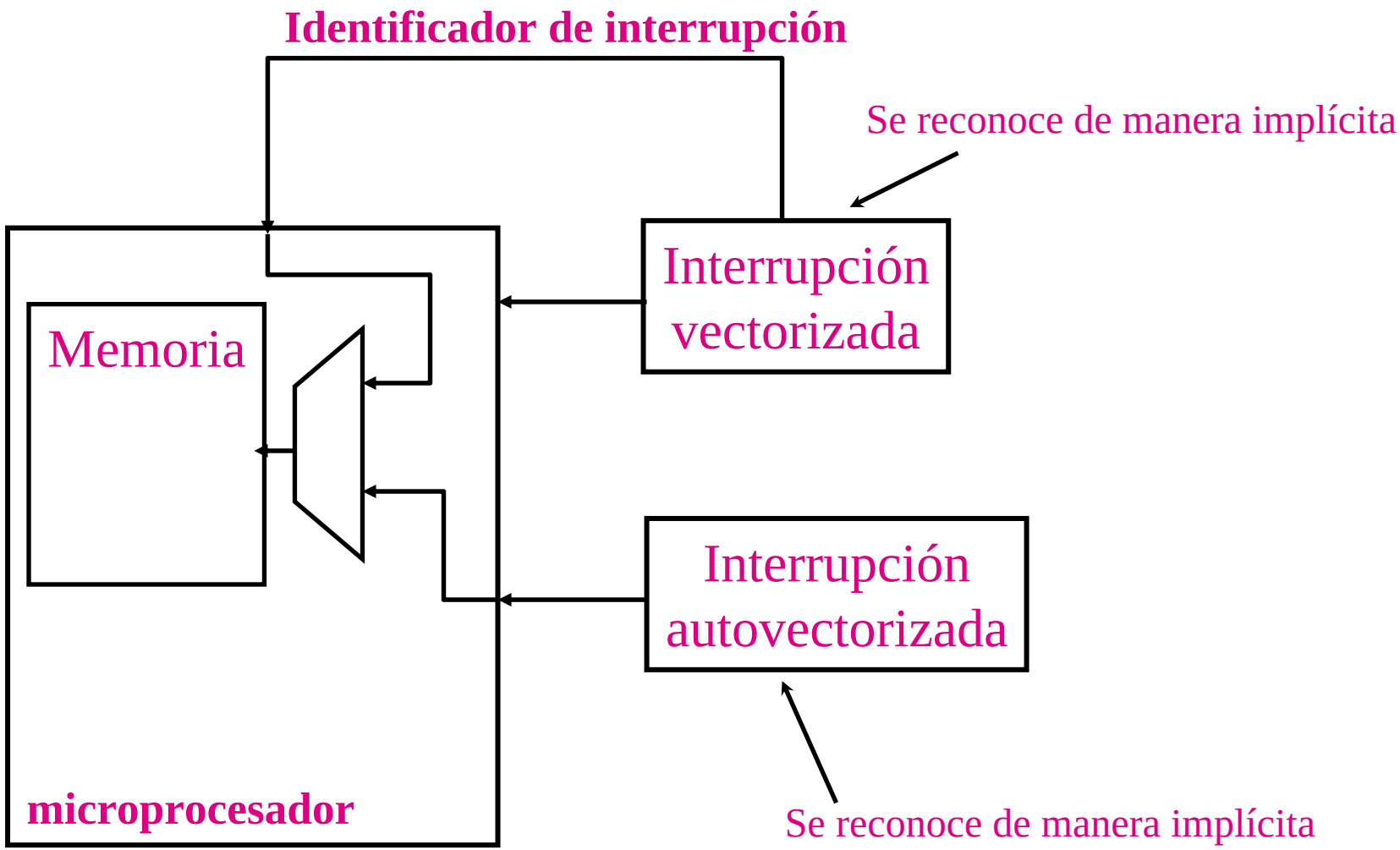


ENMASCARAMIENTO DE INTERRUPCIONES

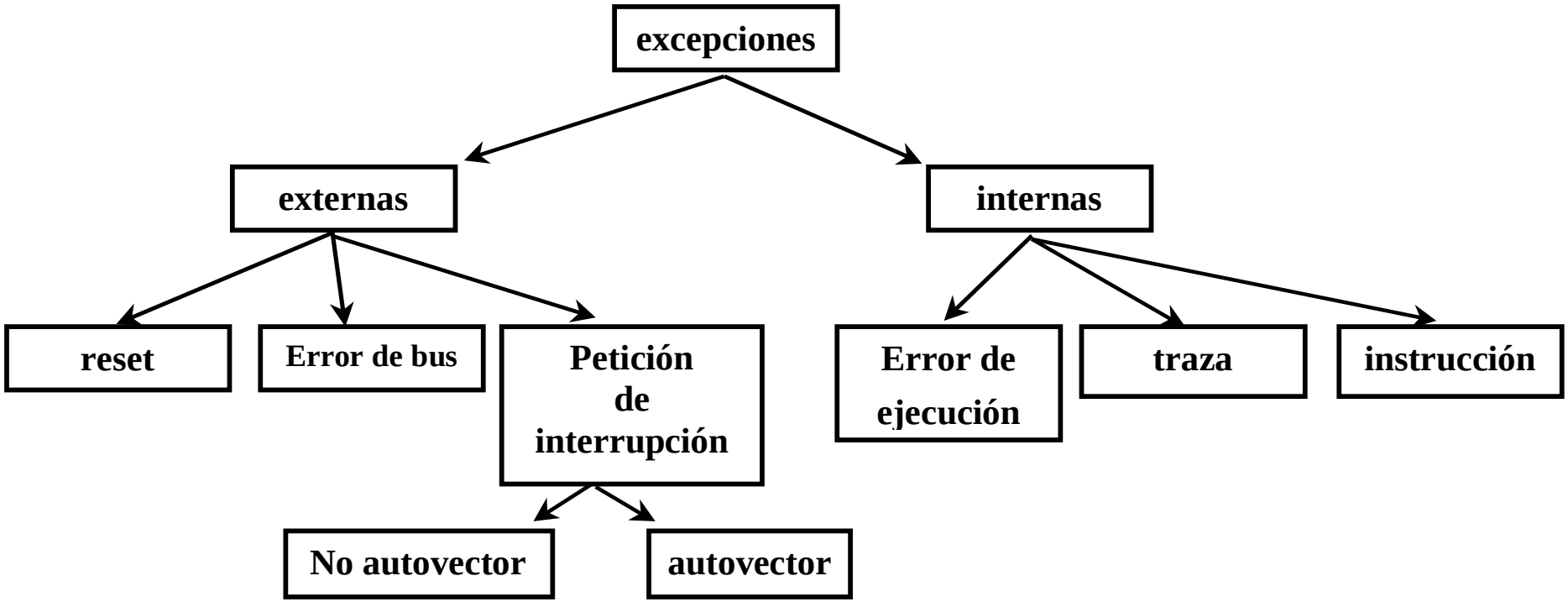
- Existe la capacidad de habilitar o deshabilitar interrupciones de manera selectiva
- Capacitación de dos tipos:
  - Hardware
  - Software
- Hardware:
  - Cada periférico o grupo de periféricos tiene un biestable de capacitación
  - Este biestable puede estar incorporado al interfaz de dispositivo como un bit del registro de condición.
- Software:
  - En la rutina de identificación de la fuente se detecta la interrupción pero no se trata



Interrupción vectorizada v.s. autovectorizada

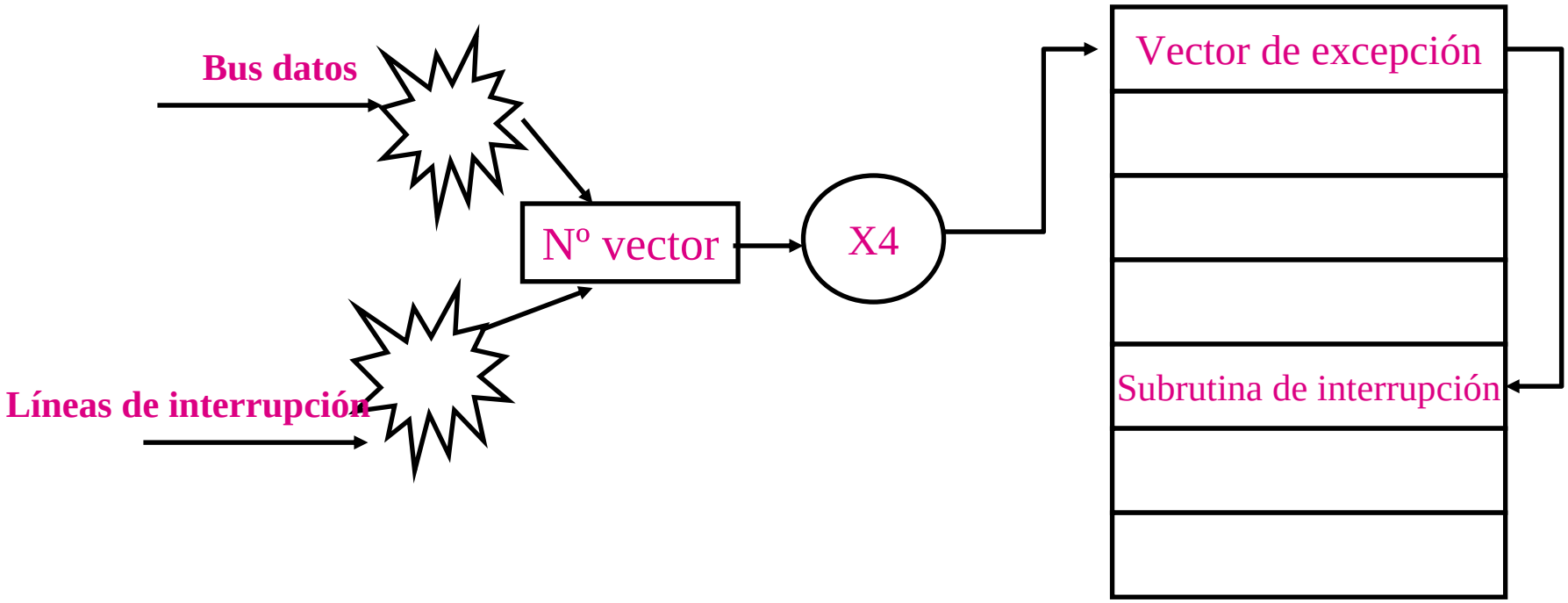


Las interrupciones del 68000



Las interrupciones del 68000

- El numero de vector identifica la interrupción
- Cada número de vector tiene una posición de memoria asociada
- La posición de memoria viene dada por la expresión:
  - posición de memoria=nº de vector*4
- En cada posición de memoria contiene la dirección de memoria en la que se encuentra la subrutina de tratamiento de interrupción



Las interrupciones del 68000

Numero vector	dirección		asignación
	decimal	hexadeci	
2	8	008	Error de bus
3	12	00c	Error de dirección
4	16	010	Instrucción ilegal
5	20	014	División por 0
7	28	01c	Instrucción trap
9	36	024	traza
15	60	03c	Vector de instrucción no inicializado
25	100	064	Autovector nivel 1
26	104	068	Autovector nivel 2
27	108	06c	Autovector nivel 3
28	112	070	Autovector nivel 4
29	116	074	Autovector nivel 5
30	120	078	Autovector nivel 6
31	124	07c	Autovector nivel 7
64-255	256-1020	100-3fc	Vectores de interrupción de usuario

Las interrupciones del 68000
gestión de las interrupciones

. Copia del registro SR en un registro interno. Activa el bit S del registro SR para pasar a modo supervisor. Desactiva el modo traza poniendo el bit T a cero. Actualiza el valor de la máscara I2,I1,I0.

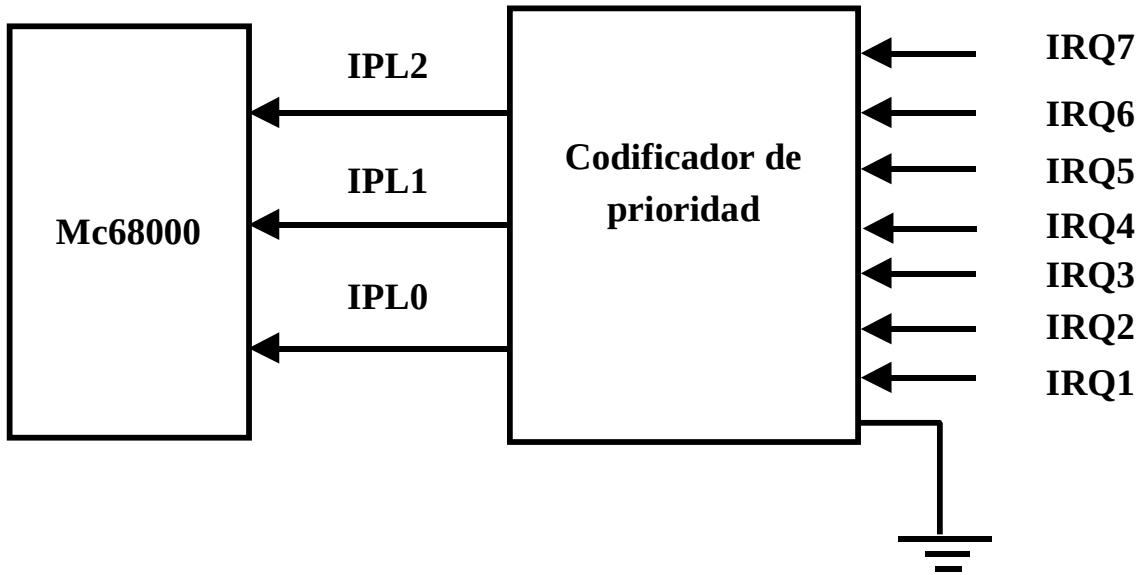
. Determina el número de vector de interrupción. Si es una interrupción de usuario se realiza una búsqueda llamada reconocimiento de interrupción. A partir del número de vector encontrado se genera la dirección del vector de excepción.

.Salvar el valor actual del contador de programa en la pila. A continuación se salva el valor de SR que se había guardado en el primer paso. Como se está en modo supervisión se usará el puntero de pila SSP

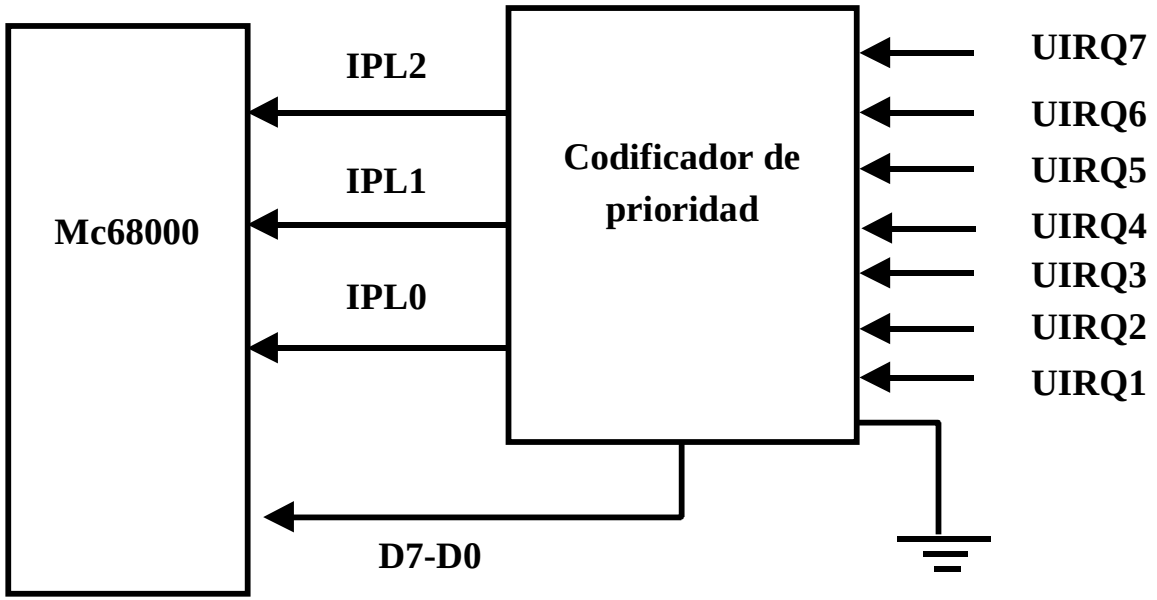
.Se carga en PC el valor de la dirección contenida vector de excepción.

.La última instrucción de la rutina de tratamiento de interrupción debe ser la RTE similar a las de retorno de subrutina.

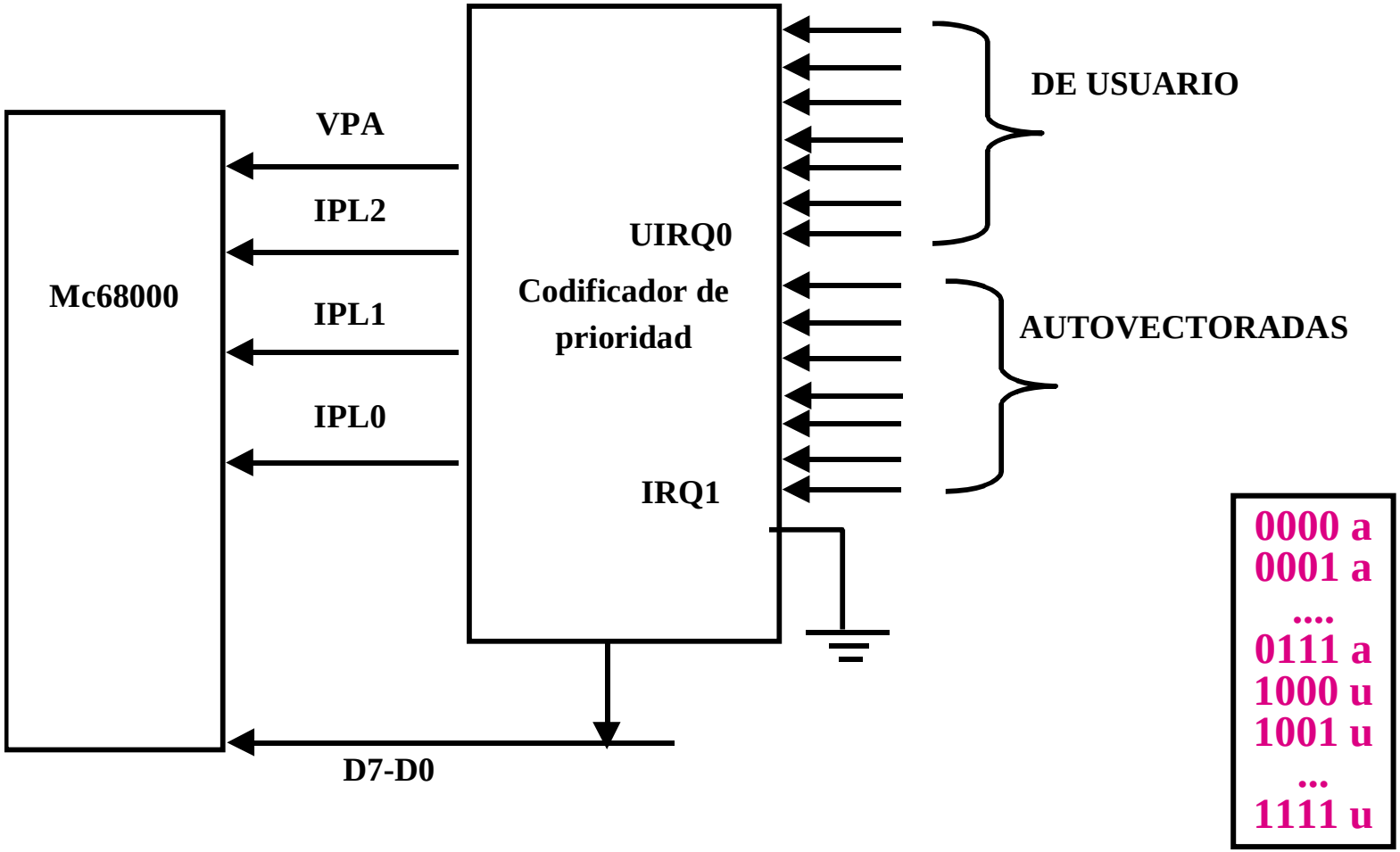
Las interrupciones del 68000
Interrupciones autovectorizadas



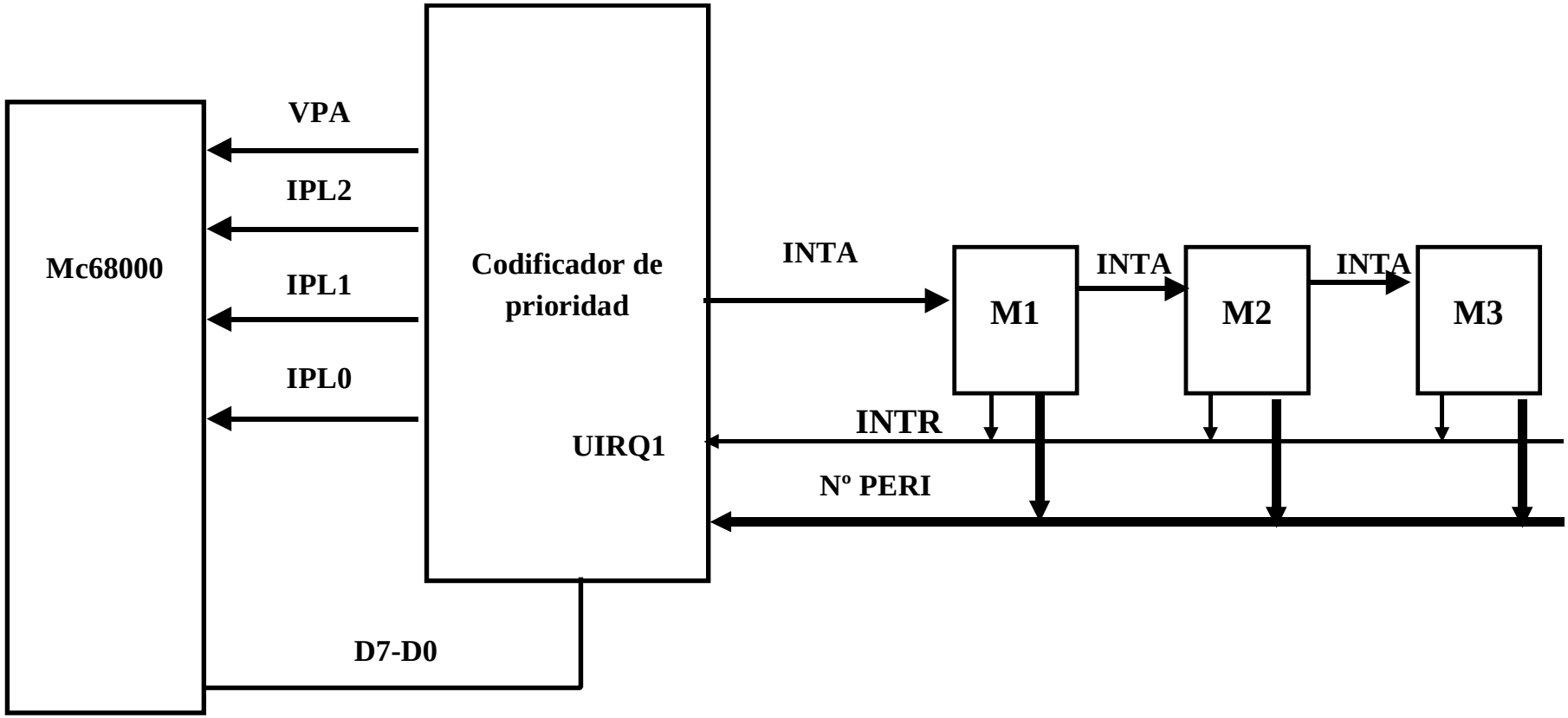
Las interrupciones del 68000
Interrupciones de usuario



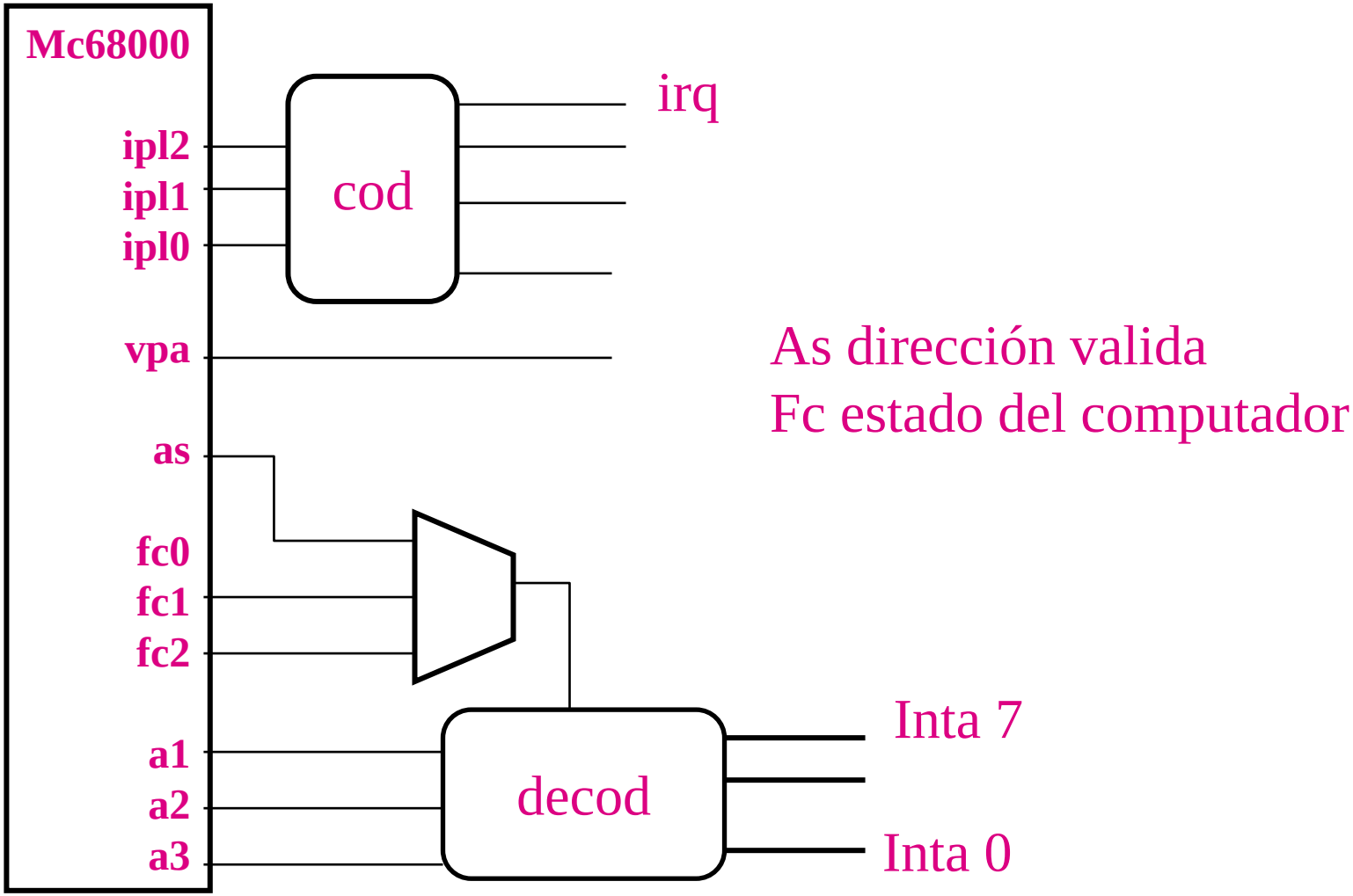
Las interrupciones del 68000



Las interrupciones del 68000

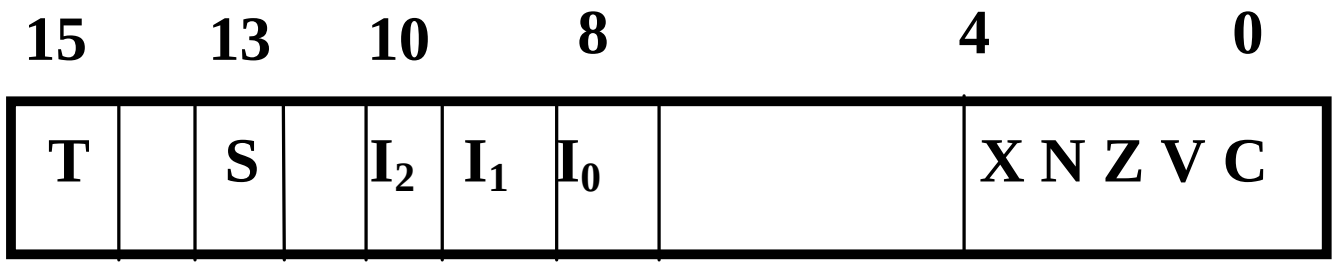


Problema 68000 no tiene línea de INTA para cada nivel de interrupción



As dirección valida
Fc estado del computador

enmascaramiento



La mascara descapacita interrupciones de prioridad igual o inferior a la codificada
$I_2I_1I_0$=000 capacita interrupciones de cualquier tipo

Se recomiendo no modificar la máscara escribiendo directamente sobre el registro
ANDI #F8FF,SR
ff8f=1111-1111-1000-111

ACCESO DIRECTO A MEMORIA

ADM

- **Problemas en las técnicas de escrutinio e interrupciones**
  - **sólo son útiles para transferencias de pequeño ancho de banda.**
  - **El peso de la transacción la sigue llevando la CPU.**
    - » **pérdida de ciclos útiles de CPU**
  - **En intercambios de información entre dispositivos de alta velocidad y la CPU pueden degradar enormemente el sistema**
  - **Ej: los discos duros con ancho de banda grande**
    - » **grandes bloque que pueden contener ciento o miles de bytes**

ACCESO DIRECTO A MEMORIA

ADM

- **Solución**
  - **Acceso directo a memoria**
  - **El controlador del dispositivo comunica directamente con la memoria sin involucrar al procesador.**
  - **Objetivo:**
    - » **Descargar de trabajo al procesador**
  - **El dispositivo utiliza las interrupciones para comunicarle al procesador que ha acabado la transferencia o para comunicarle un error**

ACCESO DIRECTO A MEMORIA

MODO DE OPERACIÓN

- Cuando la CPU desea leer o escribir un bloque de datos, envía un comando al controlador de DMA indicándole :
  - Tipo de operación
  - Dirección del módulo de entrada/salida
  - Primera posición de la memoria a la que se desea acceder
  - Nº de palabras a leer o escribir
- A continuación la CPU reanuda su trabajo y
  - el controlador de DMA el encargado de manejar la ES
- Cuando la transferencia ha terminado el controlador de DMA envía una señal de interrupción a la CPU

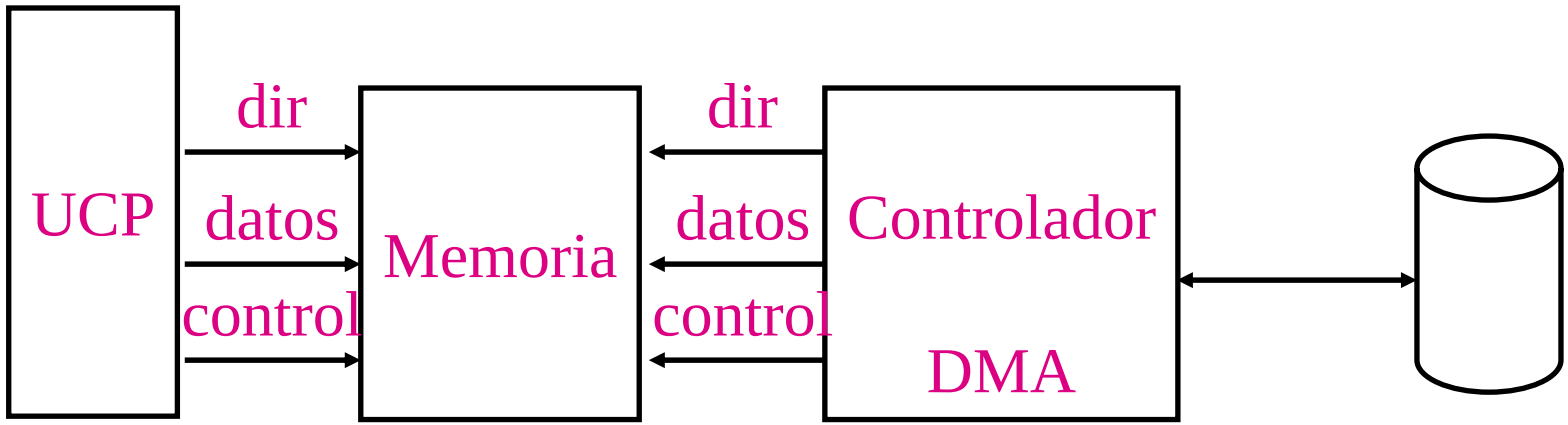
La CPU solo interviene al principio y al final de la operación

ACCESO DIRECTO A MEMORIA

- **TÉCNICAS DE IMPLEMENTACION DE ADM**
  - **Memoria multipuerto**
  - **Robo de ciclo**

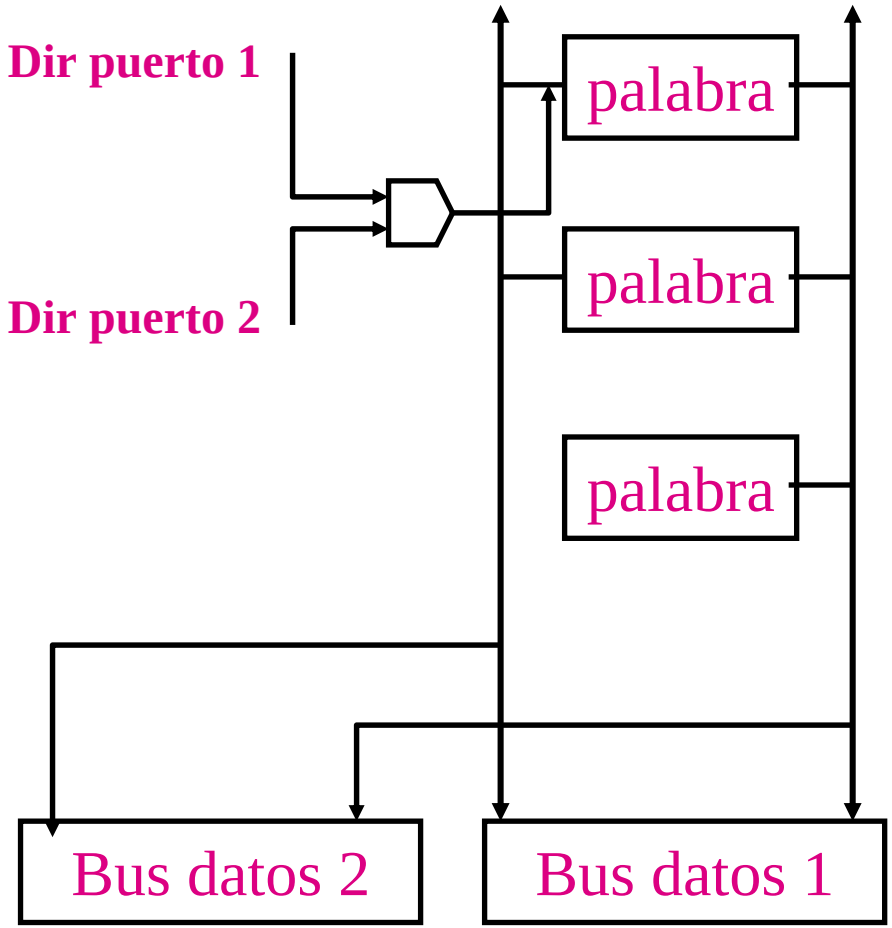
ACCESO DIRECTO A MEMORIA
MEMORIA MULTIPUERTO

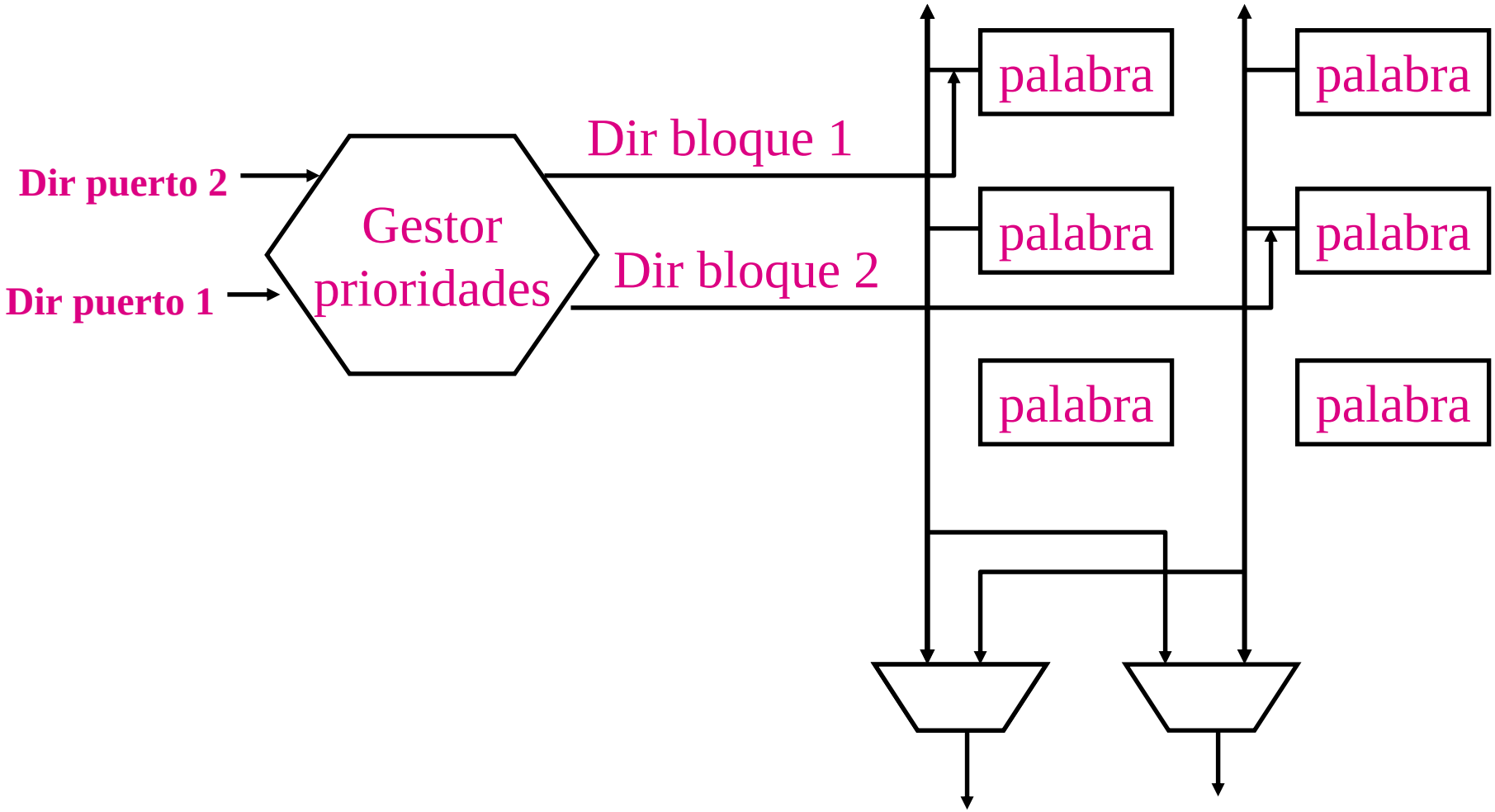
- **Memoria con dos puertos**
  - **Cada puerto puede realizar lecturas y escrituras sobre direcciones diferentes**
  - **Un puerto se conecta con la CPU**
  - **El otro se conecta con el periférico a través del controlador DMA**
  - **La memoria suele dividirse en bloques que permiten el acceso paralelo**
- **Modo de operación:**
  - **Los periféricos obtienen la memoria principal sin intervención de la CPU**
  - **Las peticiones de acceso se tratan en paralelo mientras no dirección en el mismo bloque**



ACCESO DIRECTO A MEMORIA
MEMORIA MULTIPUERTO

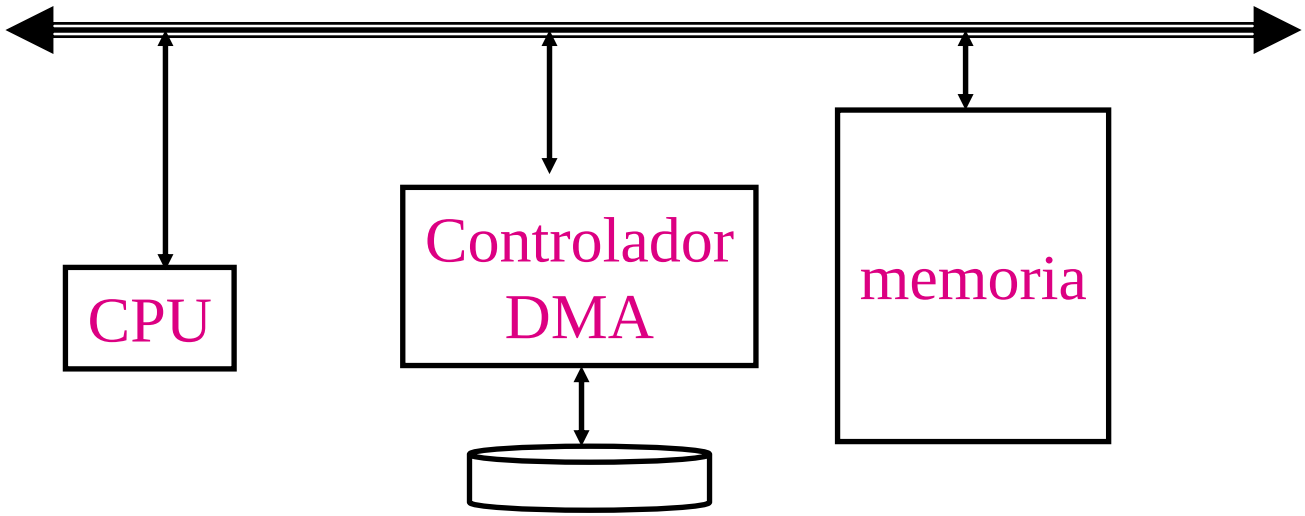
- **Inconveniente:**
  - Elevado coste de la memoria multipuerto
    - » Lógica para cada puerto
    - » Dispositivo de gestión de prioridades
  - El controlador de DMA debe generar señales de
    - » Dirección de memoria
    - » Dato
    - » Señal de L/E
    - » Inicio de ciclo
  - La memoria debe contestar con señales
    - » Datos
    - » Fin de ciclo que realiza la sincronización




Dir puerto 2
Dir puerto 1
Gestor prioridades
Dir bloque 1
Dir bloque 2
palabra
palabra
palabra
palabra
palabra
palabra

ACCESO DIRECTO A MEMORIA
ROBO DE CICLO

- La memoria sólo tiene un puerto al que deben acceder el DMA y la CPU
- Comparten bus
- ventaja
  - Sistema más económico
- Inconveniente
  - La CPU y el DMA deben ponerse de acuerdo para obtener el control del bus
  - En realidad es la CPU la que cede el control al DMA
  - más lento
  - perdidas de ciclo de la CPU aunque estas se limitan al arbitraje



ACCESO DIRECTO A MEMORIA

ROBO DE CICLO-modo operación

- El procesador envía al DMA los datos
  - » Dir memoria, nº palabras,tipo operación
  - » El DMA actúa como esclavo
    - ◆ El procesador controla la DMA como un periférico más
- El procesador retorna a sus tareas mientras el DMA prepara la transferencia
- Cuando el DMA necesita acceder a la memoria pide al arbitro el control del bus.
  - » Actúa como maestro (señal PETBUS)
- El arbitro que es el procesador concede el control
  - » DMA empieza a actuar como master
- Cuando el DMA completa la operación de E/S avisa al procesador con una interrupción
- El procesador reconoce y trata la interrupción

ACCESO DIRECTO A MEMORIA
ROBO DE CICLO

- La CPU solo acepta robos de ciclo cuando esta al final de alguna de las fases de la instrucción
  - El robo de ciclo puede aceptarse en medio de una instrucción
  - No modifica el estado del procesador solo lo para
  - El periférico debe esperar

ACCESO DIRECTO A MEMORIA
TIPOS DE ROBO DE CICLO

- **Existen tres tipos de robo de ciclo**
  - **sencillo, ráfagas, transparente**

- **Robo de ciclo sencillo**
  - **transmisión byte a byte**
  - **la CPU le concede el ciclo cuando le viene bien**
  - **modo de operación:**
    - » **Toma el control en un ciclo sencillo**
    - » **Transmite una palabra**
    - » **Devuelve el control de bus al procesador**
    - » **Solicita nuevamente el bus a sí hasta que se acaba la operación**

ACCESO DIRECTO A MEMORIA
TIPOS DE ROBO DE CICLO

- **Ráfagas:**
  - **NO Libera el bus hasta que acaba la transferencia de todo el bloque**
  - **Ventaja**
    - » **Alta velocidad de transferencia**
  - **Inconveniente**
    - » **Puede llegar a parar al procesador si el bloque es muy grande**
  - **Se utiliza para entradas sincronas de alta velocidad**
    - » **Velocidades próximas a las de la MP**
    - » **EJ lectura de cintas magnéticas**

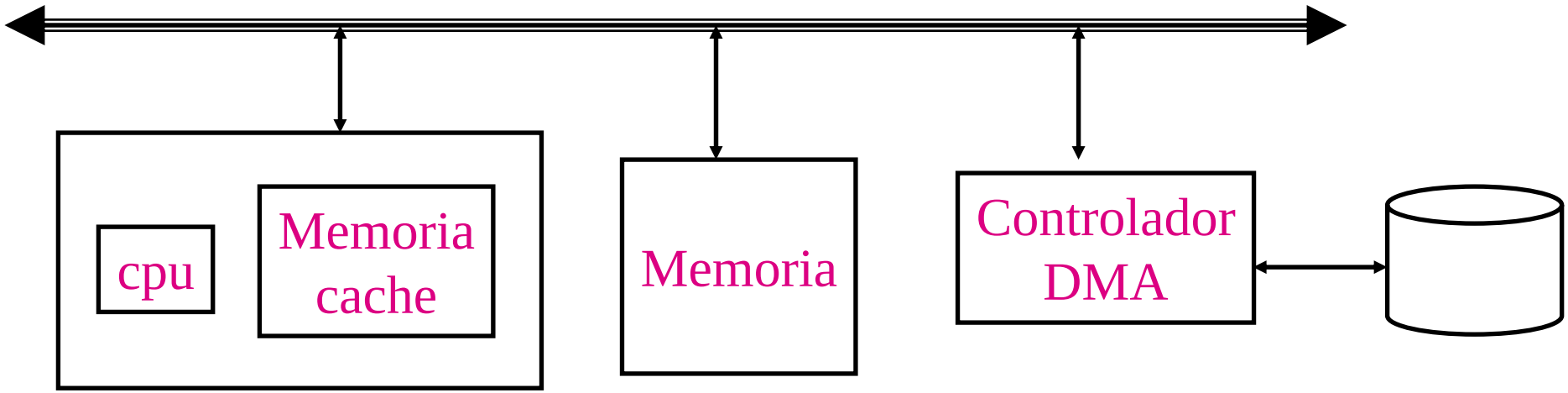
ACCESO DIRECTO A MEMORIA

TIPOS DE ROBO DE CICLO

- **Inconveniente los dos casos anteriores**
  - **degradación del rendimiento del procesador**
- **Solución:**
  - **Tener en cuenta que el procesador no necesita usar el bus en todas las fases de ejecución de una instrucción**
- **Transparente:**
  - **ADM accede al bus solo cuando no lo usa la CPU**
  - **La velocidad de trabajo de la CPU no se ve alterada**

ACCESO DIRECTO A MEMORIA

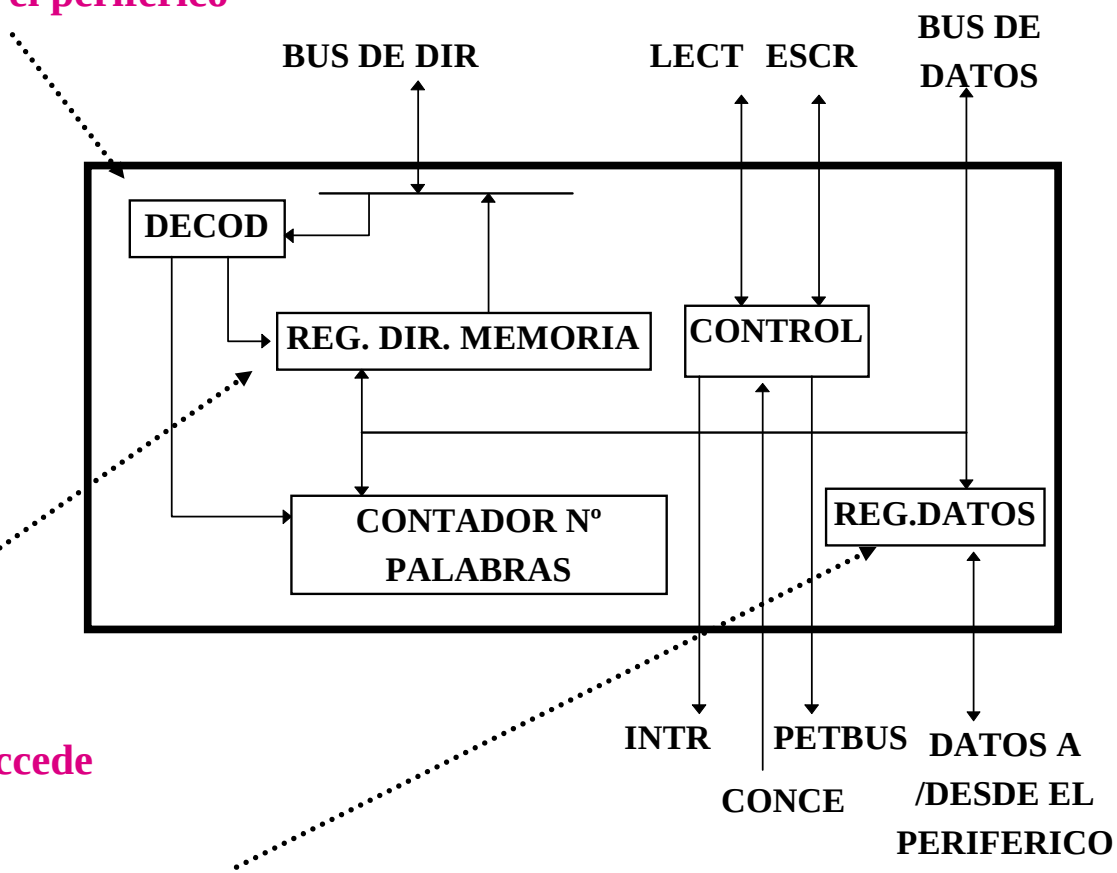
- **Problema:**
  - **Transferencias lentas**
- **Solución:**
  - **Utilizar caches**
  - **El procesador evitar los accesos directos a la memoria principal la mayor parte del tiempo dejando libre el bus para el ADM**



ESTRUCTURA DE UN CONTROLADOR DE DMA

- **DMA como esclavo**
  - Dir entrada
  - Datos de entrada
- **DMA como maestro**
  - Las direcciones salen del registro
  - Señales de control de LE
- **Líneas de arbitraje del bus**

Selecciona el periférico

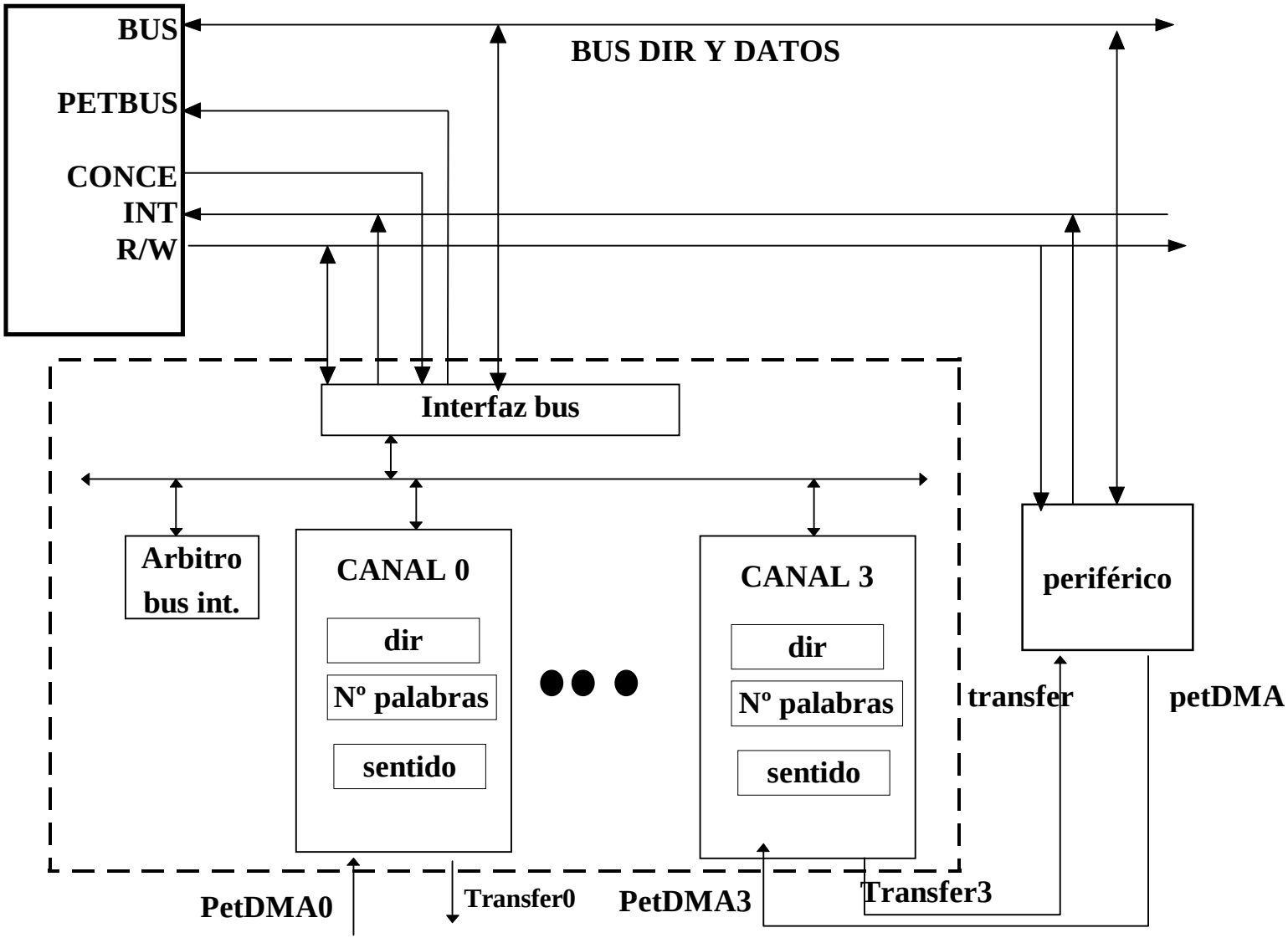


Dir de memoria a partir de la cual se accede

Guarda información de entrada y salida

ESTRUCTURA DE DMA

CUATRO CANALES INDEPENDIENTES



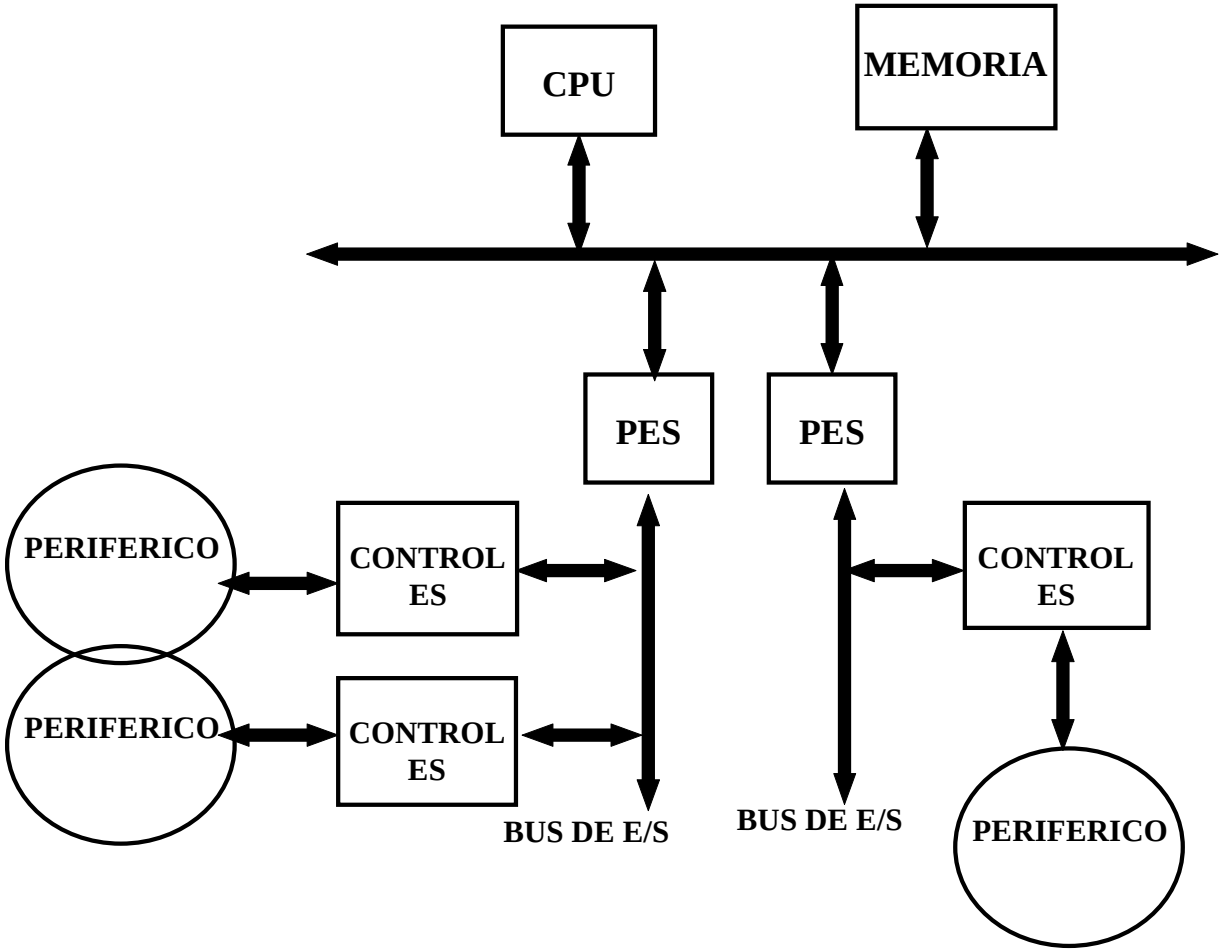
Controlador de dma mc68440/mc68450

- Pg 362

PROCESADORES DE ENTRADA/SALIDA

- Razones económicas en los CPU de los grandes sistemas justificaron la introducción de procesadores auxiliares, dedicados a E/S
- Objetivo:
  - Permitir a la CPU una dedicación máxima en las tareas de computación
- Definición
  - PE/S es un procesador con un repertorio limitado, especializado en la realización de operaciones de E/S
  - Es supervisado por un procesador central
  - Ejecutan programas de E/S almacenados en la memoria MP
    - » En la MP coexisten:
      - Programas y datos del procesador
      - Programas del PES con formatos máquina diferentes
- (Nota a los procesadores en IBM se les llama canales)

PROCESADORES DE ENTRADA/SALIDA
configuración

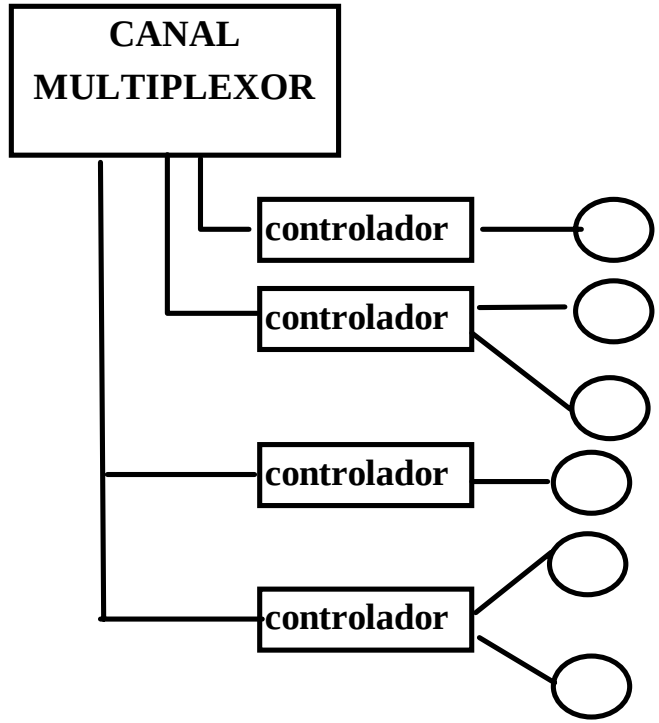


PROCESADORES DE ENTRADA/SALIDA

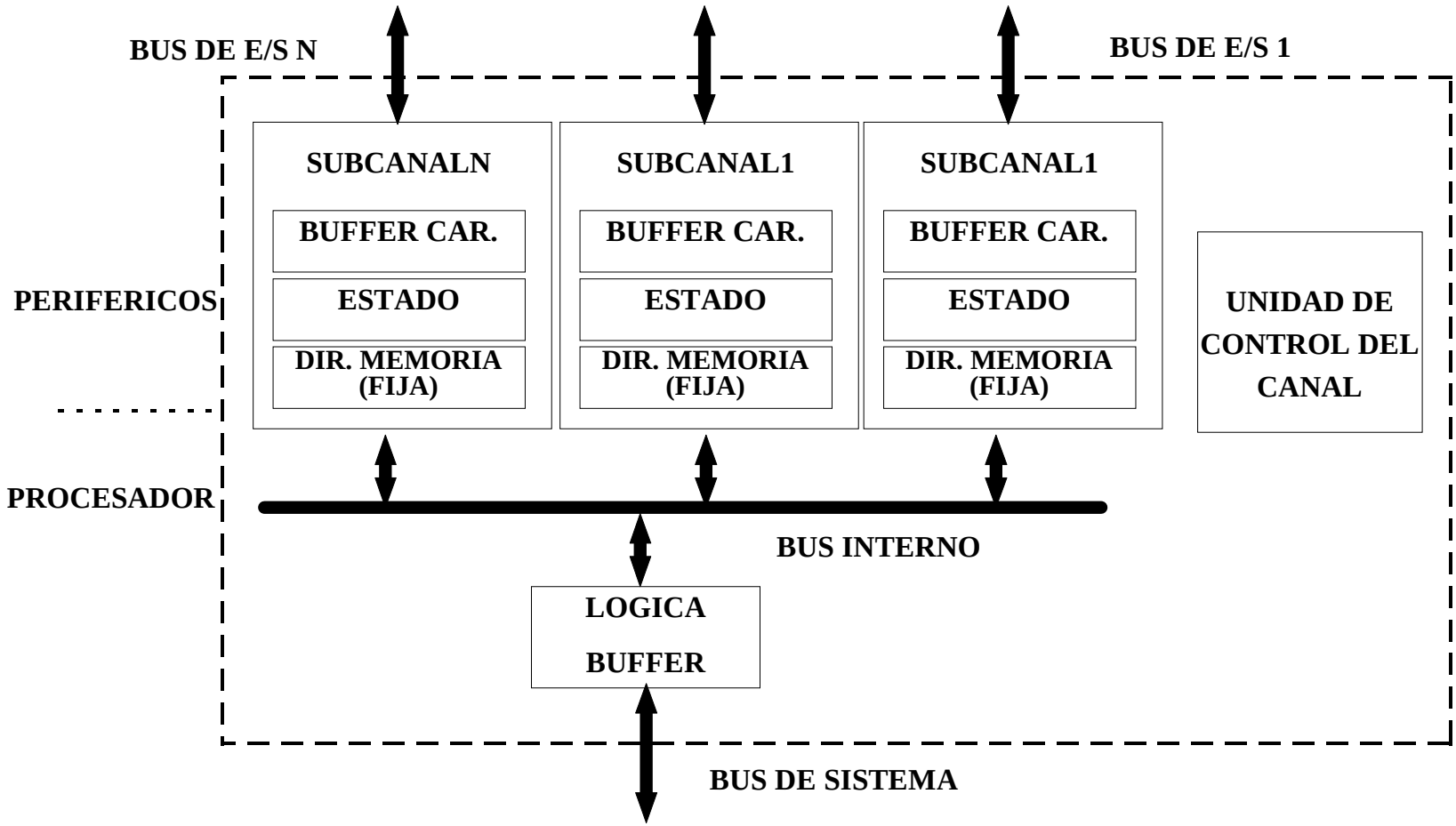
- **Tipos de procesador de entrada salida:**
  - **Multiplexor**
  - **Selector**
  - **Multiplexor por bloques**

PROCESADORES DE ENTRADA/SALIDA
multiplexor

- Intercambia datos con el procesador central a una velocidad mucho mayor que los periféricos
- Funcionamiento
  - atiende alternativamente a los periféricos durante cortos periodos de tiempo
- periféricos de velocidad media baja
- EJ: como las terminales de impresora

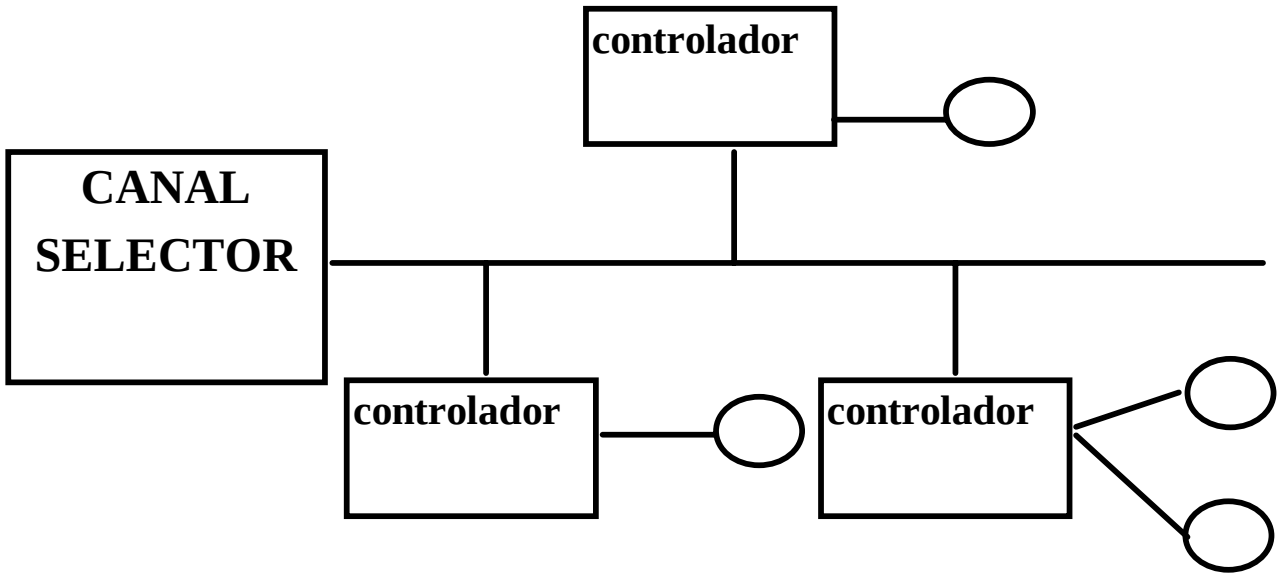


PROCESADORES DE ENTRADA/SALIDA
multiplexor

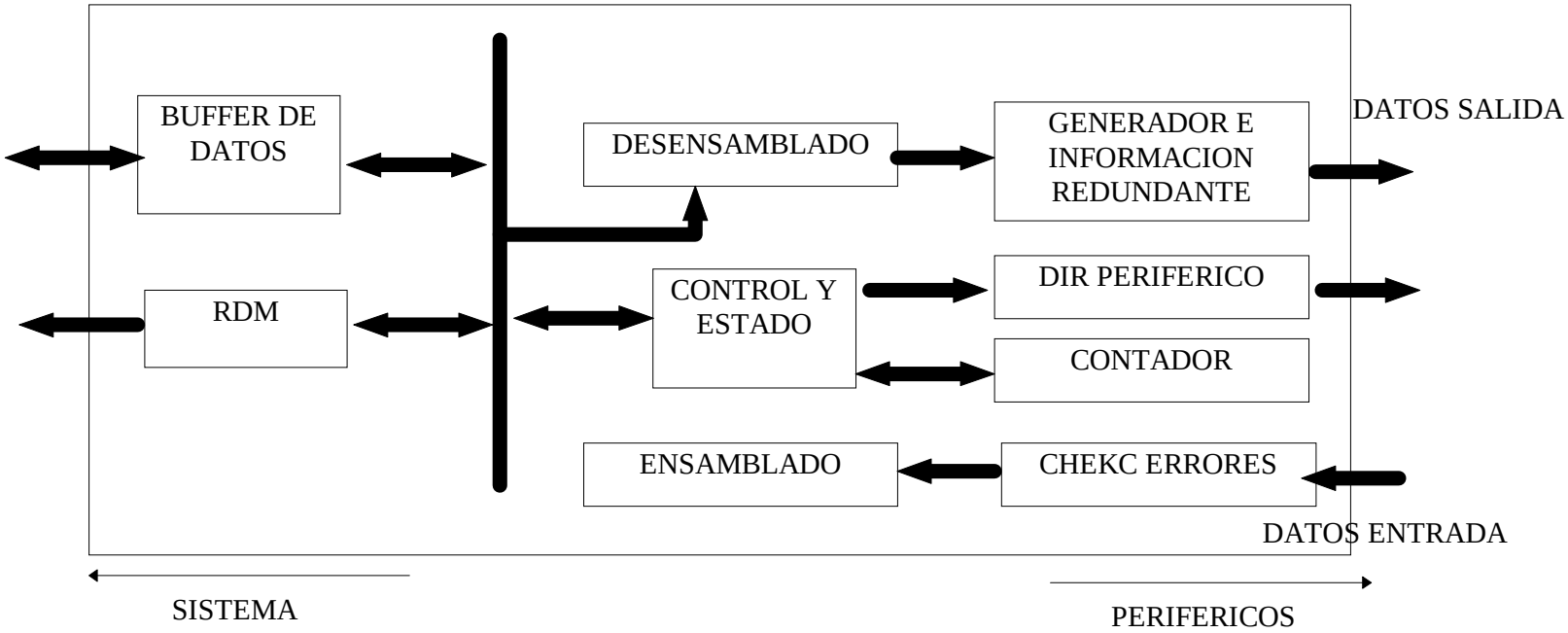


PROCESADORES DE ENTRADA/SALIDA
selector

- Controla varios dispositivos de alta velocidad
- Funcionamiento,
  - cuando comienza una transferencia con un periférico la completa antes de atender a otro.
- A diferencia del multiplexor, mantiene los parámetros de la transferencia en registros internos:



PROCESADORES DE ENTRADA/SALIDA
selector



PROCESADORES DE ENTRADA/SALIDA

MULTIPLEXOR POR BLOQUES

- Combina las características del canal multiplexor y del selector
- Funcionamiento:
  - Comunicación multiplexada con varios dispositivos bloque a bloque
- Ventajas.
  - Aprovecha las fases mecánicas de los dispositivos de alta velocidad para atender otras peticiones

INTERFACES DE ENTRADA/SALIDA

- **Misión de los circuitos de interfaces de ES:**
  - **Conversión de formatos de datos de periféricos a datos aceptables por el procesador**
  - **Sincronización de transferencias**
- **Factores que dificultan el diseño de interfaces son:**
  - **Diferencias de formatos CPU- periféricos**
  - **Diferencias de velocidad**
  - **Diferencias de niveles eléctricos**

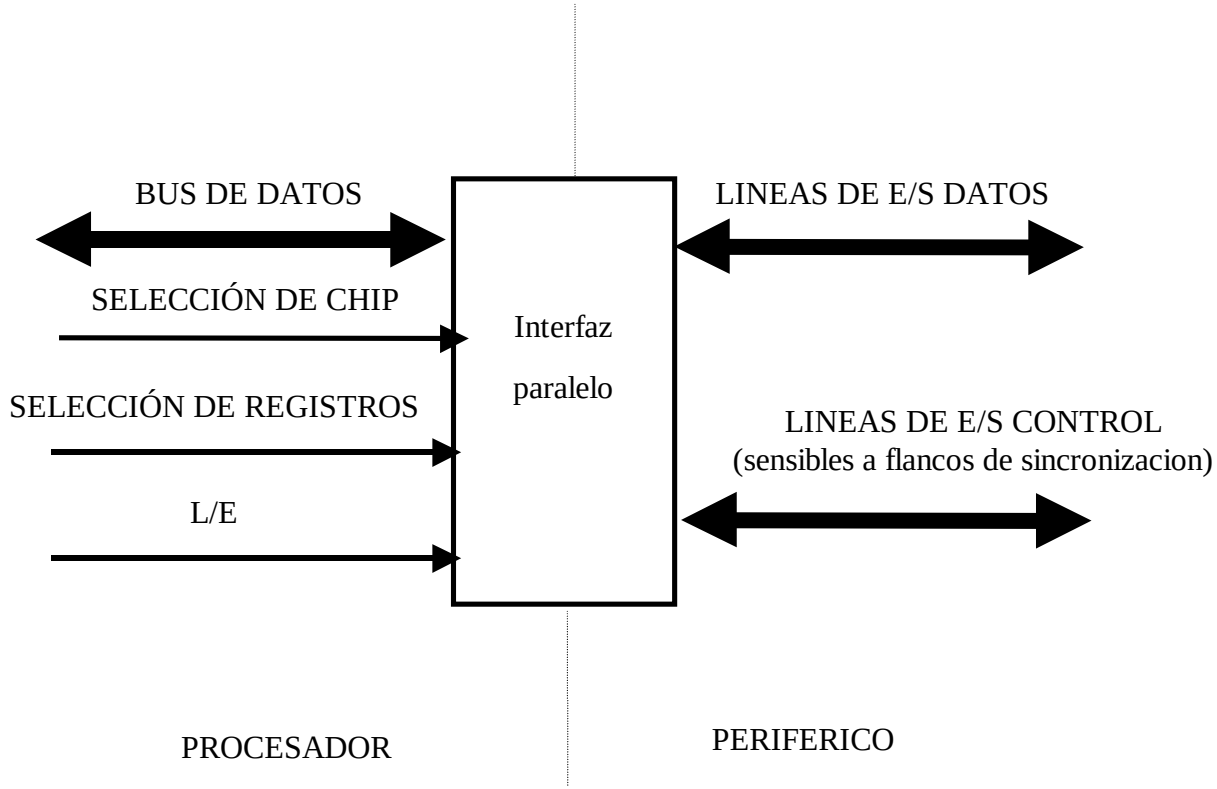
<table>
<tr><td colspan="3"></td><td>TRATAMIENTO DE LOS DATOS EN EL INTERFASE</td></tr>
<tr><td rowspan="3">FORMATO DE DATOS EN LOS PERIFERICOS</td><td colspan="2">ANALOGICOS</td><td>E: CONV A/D
S: CONV D/A</td></tr>
<tr><td colspan="2">DIGITALES</td><td rowspan="2">E: CONV S/P
S:CONV P/S</td></tr>
<tr><td>SERIE</td><td>PARALELO</td></tr>
</table>

INTERFACE PROGRAMABLES

- **Motivo de su aparición**
  - **coste tiempo y complejidad implicados en el diseño de interfaces específicos para cada periférico**
- **Solución:**
  - **CI de interfaces Programables para adaptarlos a una aplicación concreta**
  - **Se programan escribiendo en registros**
- **Ventajas:**
  - **Flexibilidad**
    - » **utilización en diversos entornos alterando el contenido de sus registros**
  - **Fiabilidad**
    - » **reducción del nº de componentes**
  - **Efecto colateral provoca la estandarización de periféricos**

INTERFASE DE E/S PARALELO

- **Misión**
  - **Acoplo de velocidades de transferencia**
  - **Sincronización entre el procesador y el periférico**



MC6821 (PIA)

PIA (Peripheral Interface Adapter)

- **Interfaz paralelo de 8 bit que puede conectarse al MC68000 a través de un bus síncrono para periféricos del mc6800**
- **2 puertos de entrada/salida paralelos**

∗ORA registro de datos

∗emisor y receptor de datos hacia o desde las líneas exteriores.

∗Su dirección depende del contenido del registro DDRX

∗DDRA, Registro de dirección de datos

∗ Cada bit indica como se comporta el bit correspondiente del ORX

∗CA1,CA2 líneas de control

∗para examinar el estado de los dispositivos periféricos

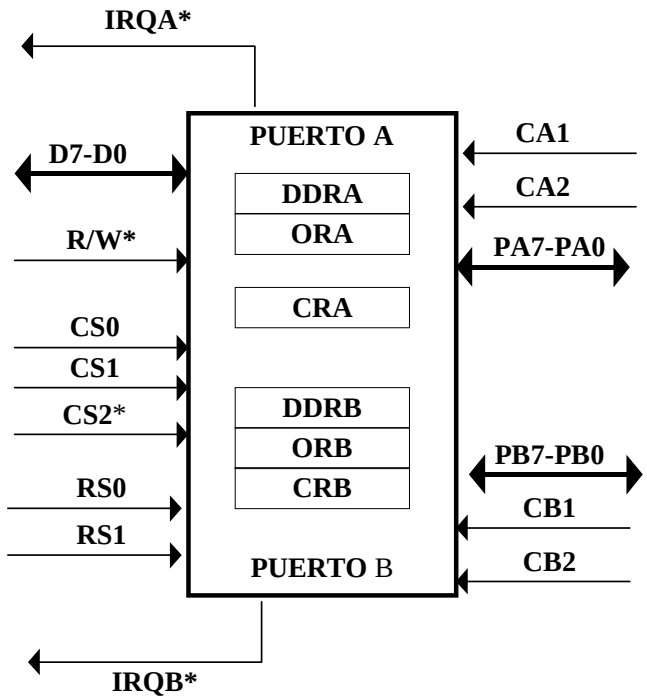
∗generar señales de control sobre los mismos.

∗ CRA, Registro de control contiene información de estado.

∗Determina el funcionamiento de las líneas CA1 y CA2,

∗Si actúan como entradas o como salida

∗ su capacidad para generar peticiones de interrupciones

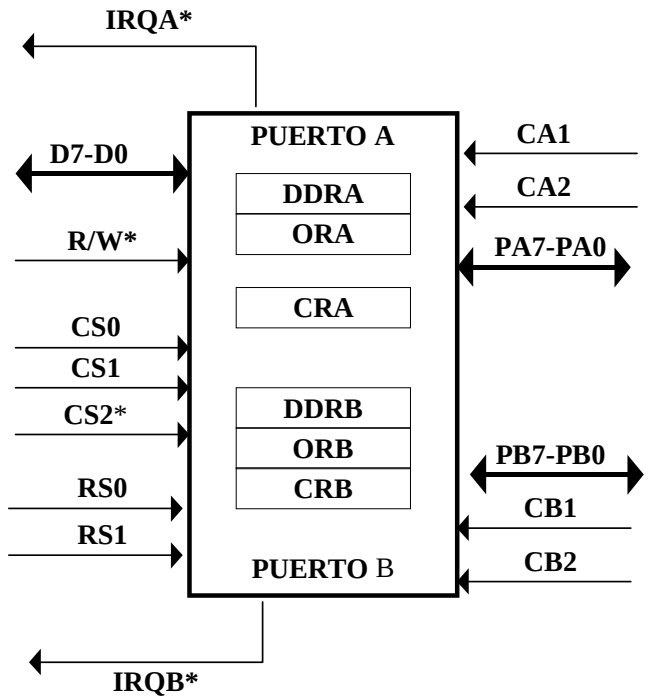


MC6821 (PIA)

PIA (Peripheral Interface Adapter)

- **PROGRAMACIÓN**
  - **RS1 selecciona el acceso al puerto A o B**
  - **RS0 indica si se accede a uno de los registros de datos (OR o DDR) o al de control CRx**
  - **El acceso a ORX o DDRX viene determinado por el bit 2 de CR**
    - » **Cada vez que se quiera modificar OR o DDR se debe modificar previamente cr**
- **SELECCIÓN DEL INTERFAZ**
  - **CS0,CS1 YCS2**

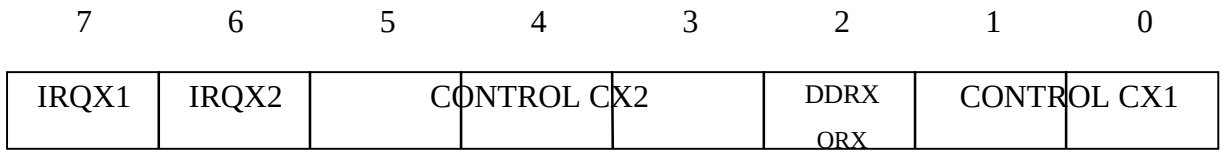
RS1	RS0	CRA2	CRB2	REGISTRO SELECCIONADO
0	0	0	X	DDRA
0	0	1	X	ORA
0	1	X	X	CRA
1	0	X	0	DDRB
1	0	X	1	ORB
1	1	X	X	CRB



MC6821 (PIA)

PIA (Peripheral Interface Adapter)

- **Registro de datos ORX:**
  - **Cada bit va asociado a una línea (PA0-PA7)**
  - **se programa como de E(0)/S(1) escribiendo DDRA.**
- **Registro de control CRA**
  - **<7(CA1),6(CA2)> flags que se activan al detectar la transición programada por la líneas CA1 y CA2**
- **<5 ,3> programan las líneas CA2.**
  - ⇒ **El 5 si es indica si es de entrada (0) o de salida (1).**
  - ⇒ **Si es de entrada el 4 determina el tipo de transición. Positiva(1) o negativa(0).**
  - ⇒ **El 3 capacita la petición de interrupción**
- **El bit 2 determina la selección de acceso a DDRA/ORA**
- **bits 0 y 1 programanCA1.**
  - ⇒ **El bit 1 determina si la transición que detecta el flag es positiva o negativa**
  - ⇒ **El bit 0 capacita la interrupción.**

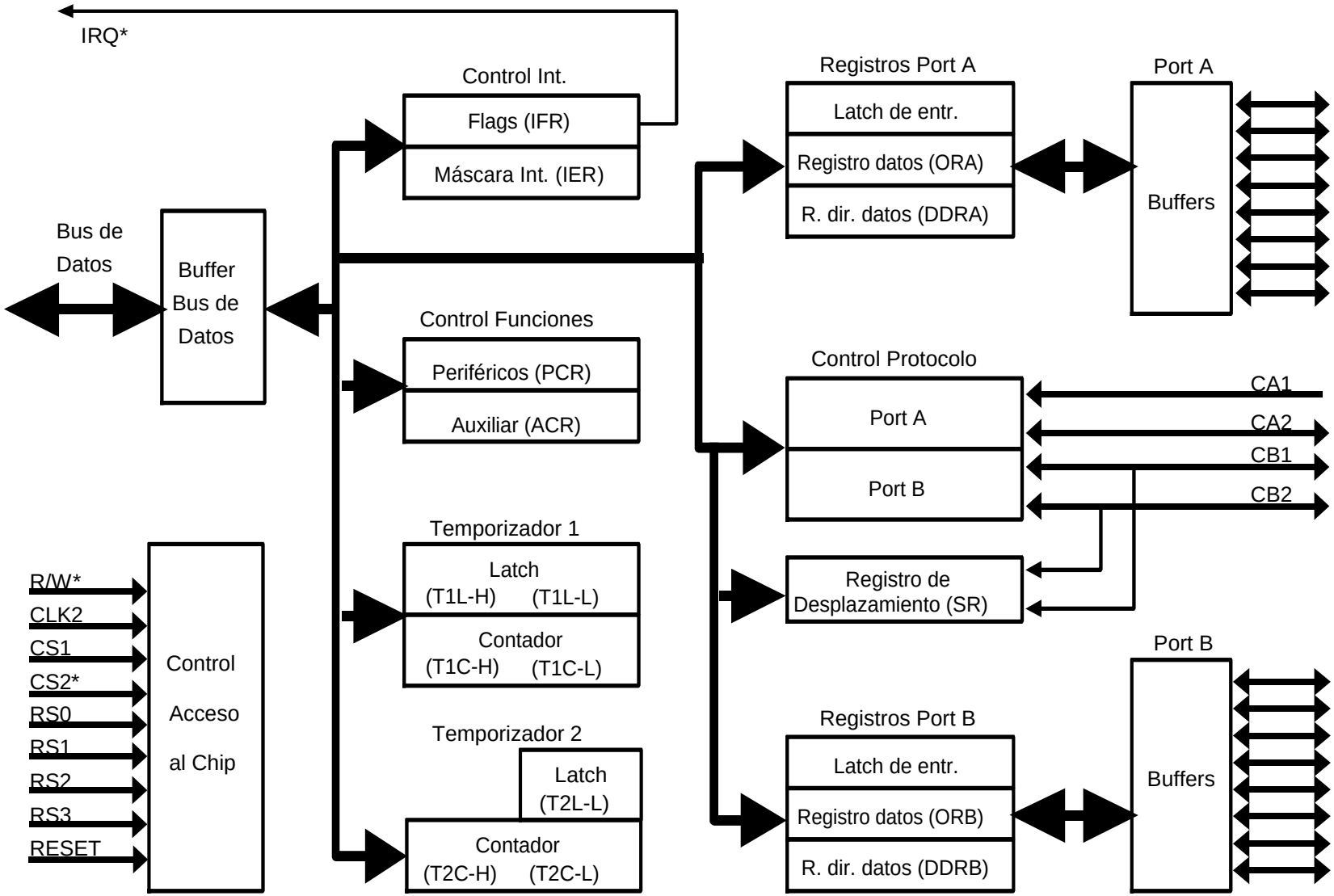


REGISTRO DE CONTROL CRX

VIA R6522

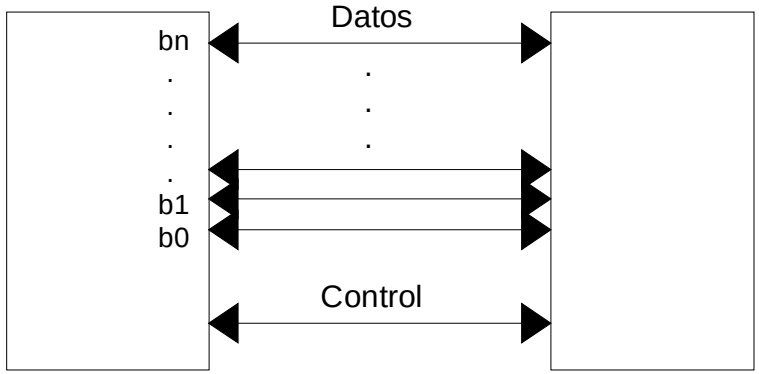
- Más completo que la PIA
- Dos puertos de E/S de 8 líneas similar a la PIA
- Dos contadores temporizadores (T1 y T2)
  - generar retardos y pulsos
  - detección de pulsos
  - generación de interrupciones
- registro desplazamiento
  - conversiones serie-paralelo
- registro de flags de 8 bits
  - permite detectar la ocurrencia de determinados eventos
- Circuitería lógica para el tratamiento de interrupciones

VIA R6522

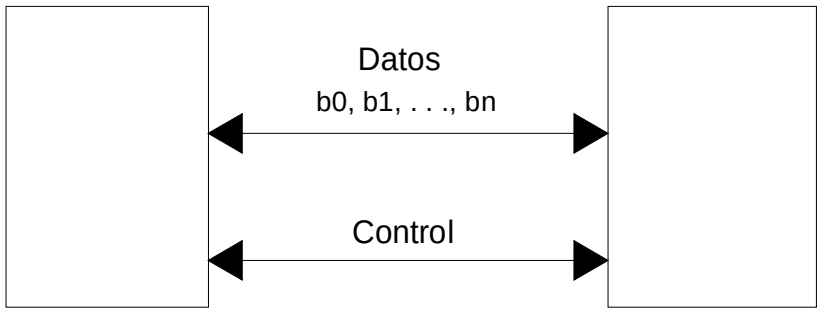

IRQ*
Control Int.
Flags (IFR)
Máscara Int. (IER)
Registros Port A
Latch de entr.
Registro datos (ORA)
R. dir. datos (DDRA)
Port A
Buffers
Bus de Datos
Buffer Bus de Datos
Control Funciones
Periféricos (PCR)
Auxiliar (ACR)
Control Protocolo
Port A
Port B
CA1
CA2
CB1
CB2
Temporizador 1
Latch (T1L-H) (T1L-L)
Contador (T1C-H) (T1C-L)
Registro de Desplazamiento (SR)
R/W*
CLK2
CS1
CS2*
RS0
RS1
RS2
RS3
RESET
Control Acceso al Chip
Temporizador 2
Latch (T2L-L)
Contador (T2C-H) (T2C-L)
Port B
Registros Port B
Latch de entr.
Registro datos (ORB)
R. dir. datos (DDRB)
Buffers

ENTRADA/SALIDA SERIE

- La transmisión paralela envía datos a distancias cortas rápidamente
- Para distancias medias o grandes,
  - el elevado coste
- En la transmisión serie
  - solo una línea de datos
  - viajan los bitas en instantes sucesivos
  - el tiempo asignado a un bit determina la frecuencia de transmisión
  - Biis/sg = Baudios
  - El receptor debe muestrear la línea con una frecuencia igual a la de transmisión para que no haya errores
  - a la sincronización de la recepción se le llama sincronización del bit
  - existen dos tipos sincronización:
    - » asíncrona
    - » síncrona



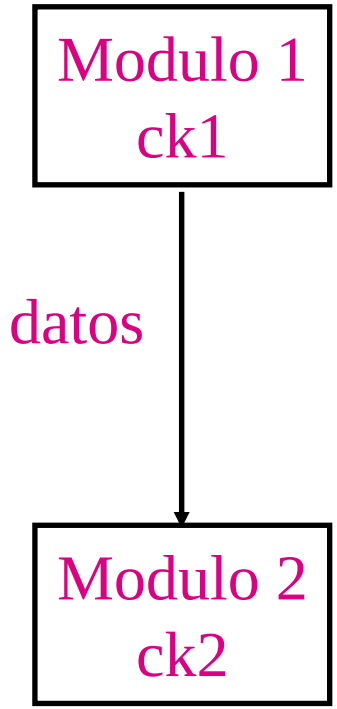
a)

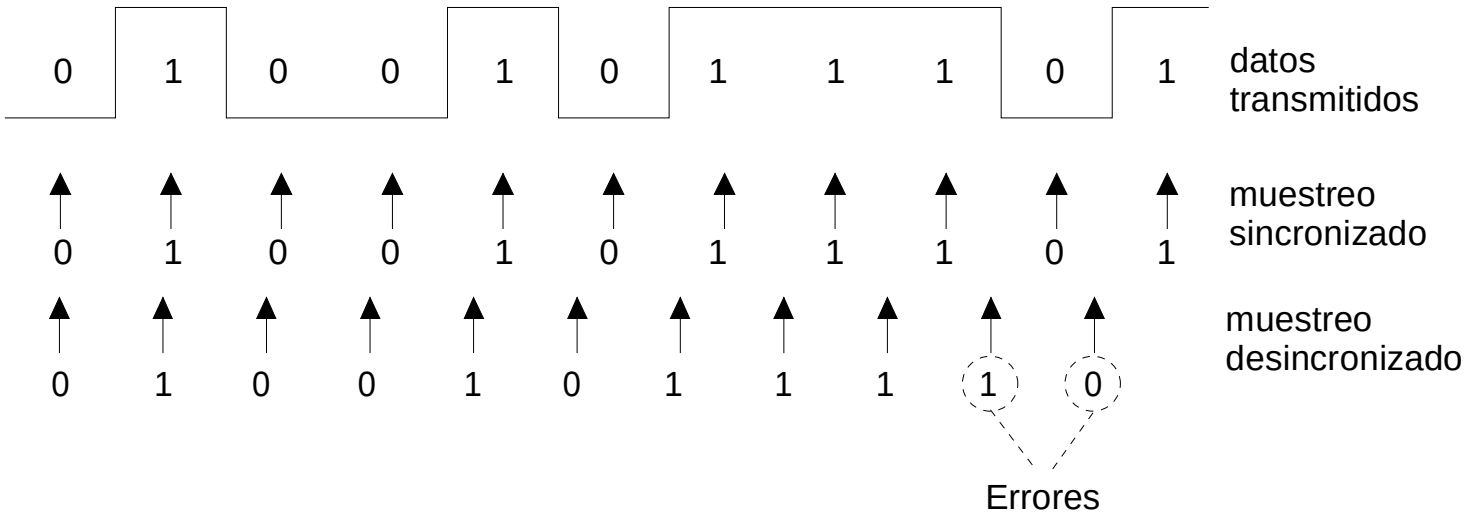


b)

ENTRADA/SALIDA SERIE
TRANSMISIÓN ASÍNCRONA

- Utilizan relojes diferentes el emisor y el receptor
- Estos relojes de la misma frecuencia en el emisor y el receptor.
  - Siempre acaba introduciendo desincronizacion al cabo de unos pocos bits.
- solo se utiliza para la transmisión de 10 o 12 bits que representan caracteres
- El siguiente carácter se transmite un tiempo arbitrario después y necesita de una nueva sincronización
- Los bit de cada carácter se envía con frecuencia fija
- Frecuencia inferior a 20.000baudios





ENTRADA/SALIDA SERIE SÍNCRONA

- **Utilizan la misma señal de reloj para la emisión y la recepción**
- **El problema**
  - » **es imposible enviar una señal de reloj a distancia sin que se produzcan distorsiones.**
- **solución**
  - » **codificar las señales de reloj en los bits de datos que se transmiten.**
- **permite mantener la sincronización durante la transmisión de bloques de bits de tamaño considerable**
- **permitiendo mayores velocidades de transmisión de la información.**

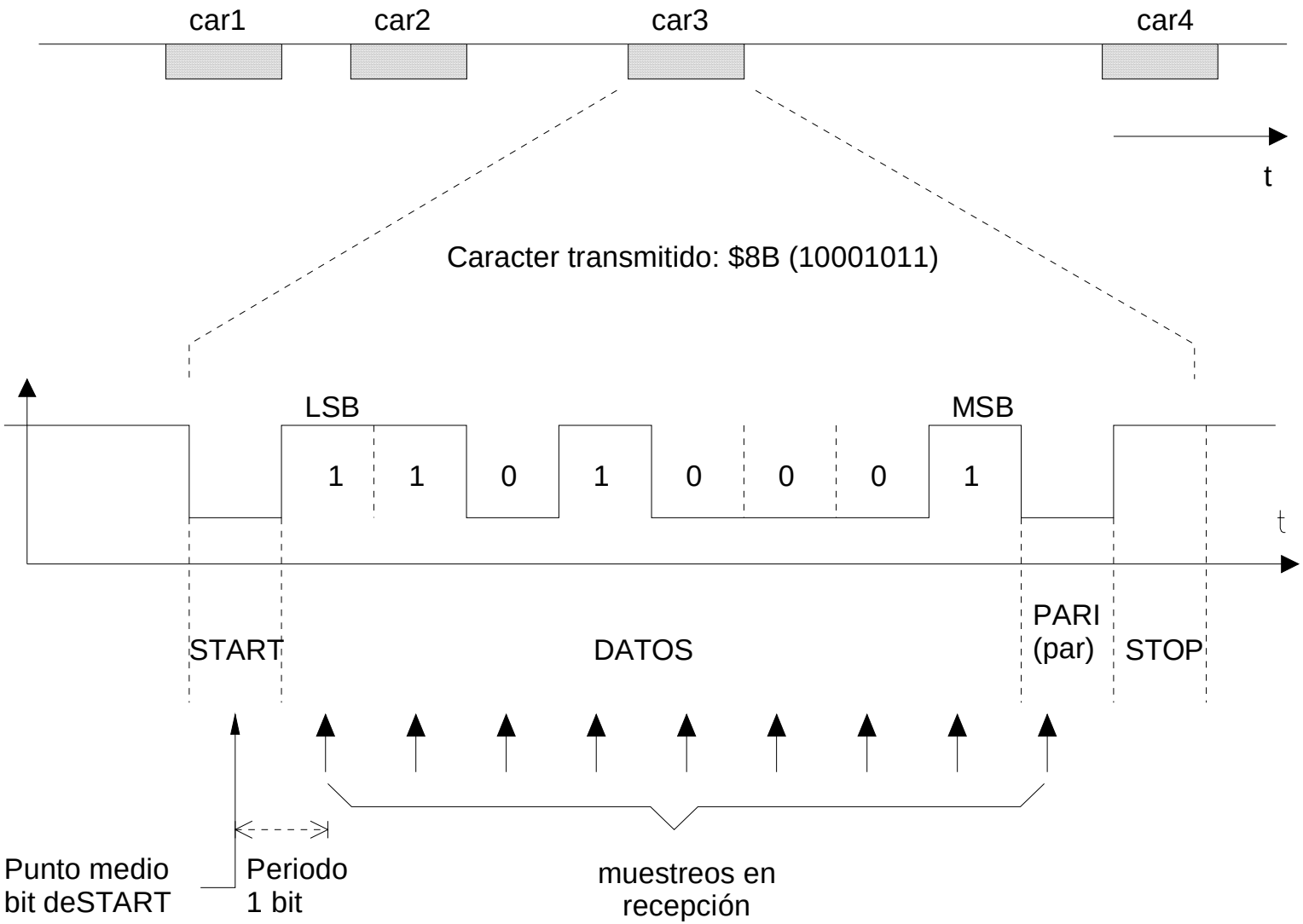
ENTRADA/SALIDA SERIE
TRANSMISION ASINCRONA

- Orientada a la transmisión de caracteres
- los interfaces añaden bits extras para
  - garantizar la sincronización
  - facilitar la detección de errores

- FORMATO DEL CARÁCTER
  - Un bit START siempre a 0
  - De 5 a 8 bits para representar el dato dependiendo del código empleado
  - Un bit de paridad, que sigue al ultimo bit de datos y que ayuda a detectar errores simples.
  - Bit de parada (STOP)que consiste en uno o varios bits a 1
- Después del bit de parada la línea queda a 1 de esta forma se detecta con facilidad el bit de arranque.

ENTRADA/SALIDA SERIE

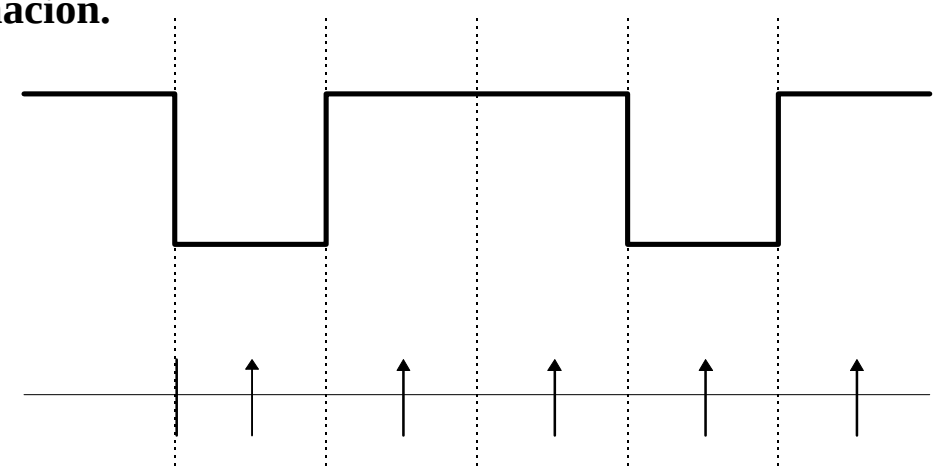
TRANSMISION ASINCRONA



ENTRADA/SALIDA SERIE

TRANSMISION ASINCRONA

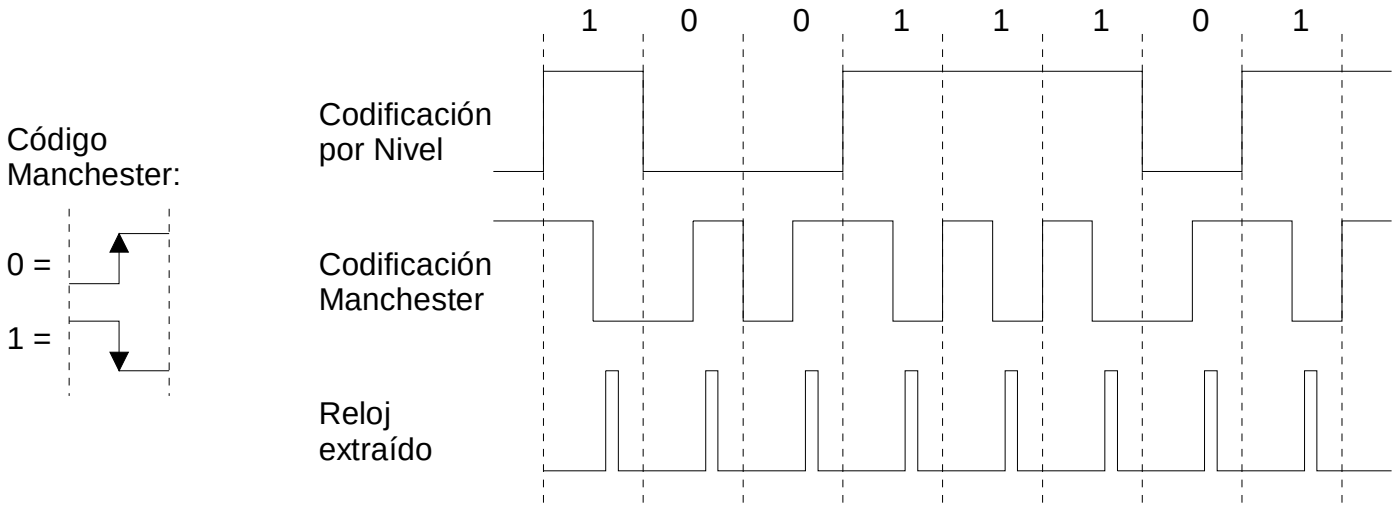
- **SINCRONIZACIÓN EMISOR RECEPTOR**
- **Problema:**
  - **Emisor y receptor pueden tener relojes cuyas frecuencias no sean exactamente idénticas**
- **Solución**
  - **El receptor usa un reloj múltiplo de la frecuencia de transmisión**
  - **Detecta el centro del bit start y muestrea cada 16 pulsos, con lo que el muestreo coincide siempre aproximadamente en el centro de cada bit de información.**



ENTRADA/SALIDA SERIE

TRANSMISION SINCRONA

- La señal de reloj incluida en el propio dato
- orientado a la transmisión de grandes bloques de información
- Eliminación de los bits START y STOP aumenta la anchura de banda disponible para datos
- facilidades para incorporar información de control e información para detección de errores
- El tratamiento del reloj permite transmitir más rápidamente a grandes distancias.



ENTRADA/SALIDA SERIE

TRANSMISION SINCRONA

- **La transmisión sincrona a su vez puede ser:**
  - **orientada a byte**
  - **orientada a bit**
- **En los dos casos es necesario añadir información a los datos:**
  - **indicar donde empieza o termina un byte**
  - **detección de errores**

SYNC	SYNC	SOH	Cabecera	STX	Bloque datos	ETX	Checksum

BISYNC--> ORIENTADO A CARACTER

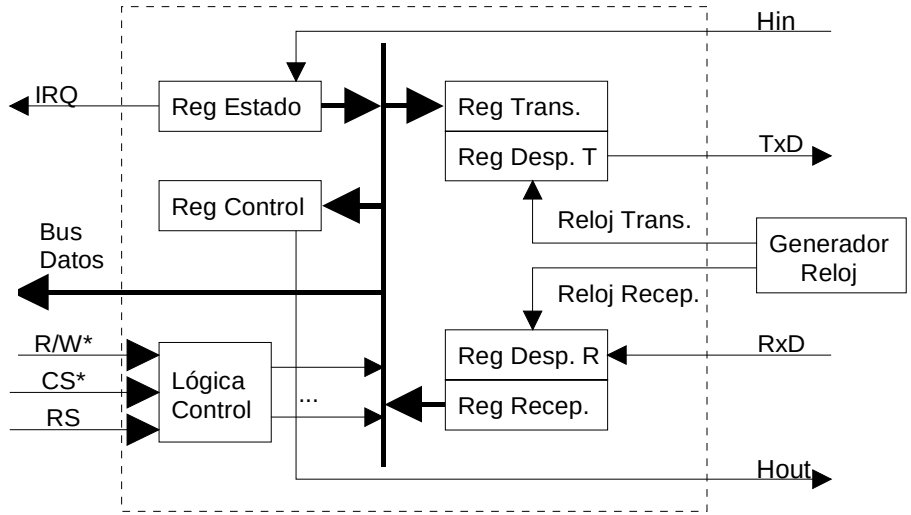
01111110	Dirección	Control	Bloque datos	Checksum	01111110

HDLC--> ORIENTADO ABIT

ENTRADA/SALIDA SERIE

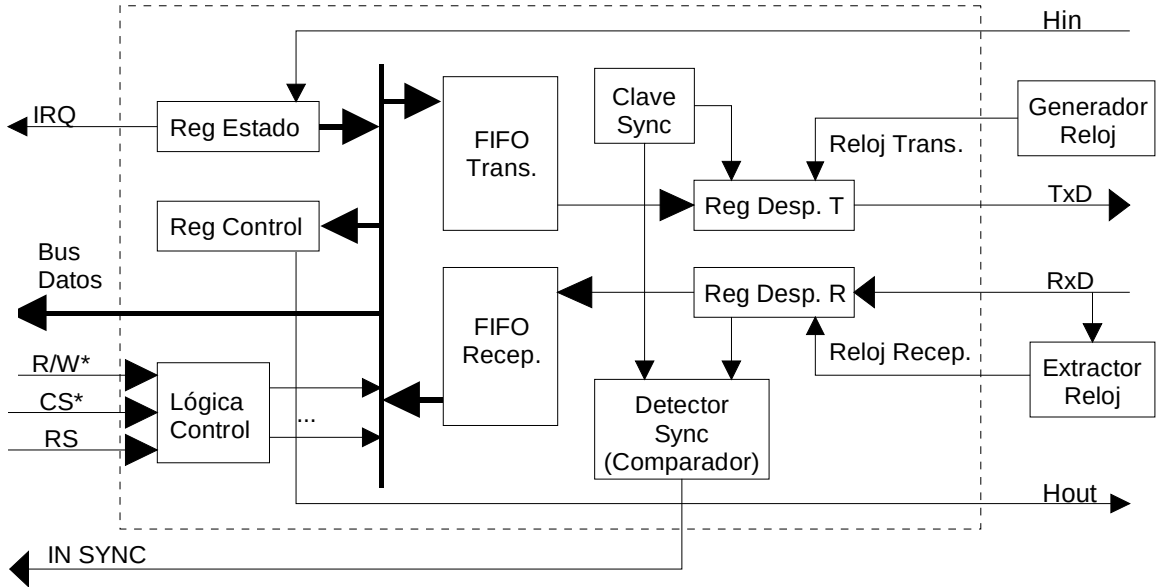
ASINCRONO

Un registro de 8 bits para almacenar los datos



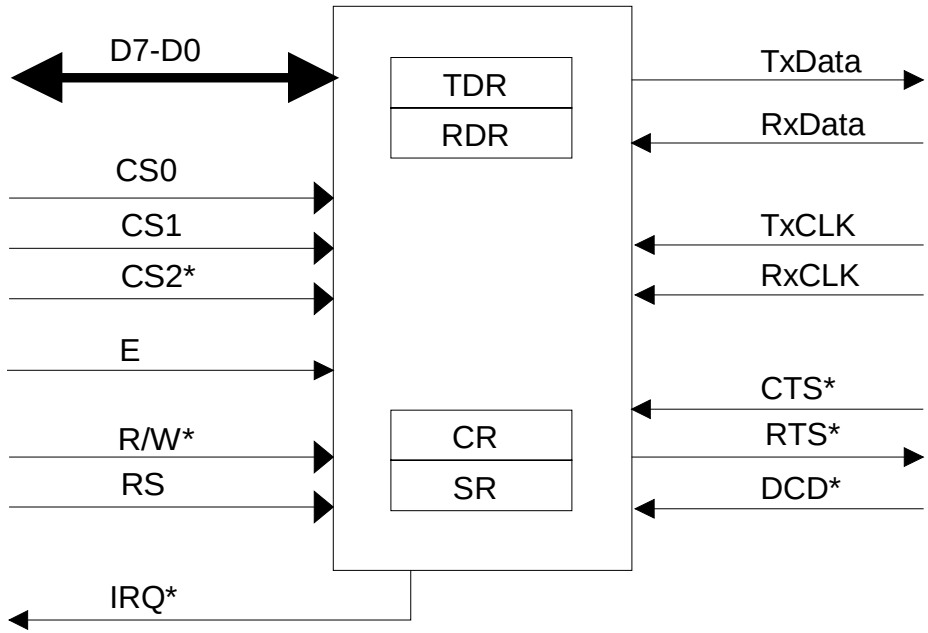
SINCRONO

extractor del reloj
cola FIFO para almacenar los bloques



MC6850 HACIA
Asynchronous Communications Interface Adapter

- 4 registros de 8 bits
- seleccionables con R/w y RS
- TDR registro de transmisión de datos
  - solo escritura SR=1
- RDR Registro de recepción de datos
  - solo lectura SR=1
- CR Regsitro de control
  - programa el funcionamiento
  - las interrupciones
  - solo escritura
- Registro de estado
  - flags de información
  - solo lectura
- Señales
  - RTS solicitud de transmisión (salida)
  - CTS permiso para transmitir concedido (entrada)
  - DCD detección de dato de entrada (entrada)



MC6850 HACIA
Asynchronous Communications Interface Adapter

- **CONTROL DE INTERRUPCIONES**
  - **CR<7> capacitación de interrupciones de recepción**
    - » **SR<0> indica si el registro de recepción RDR lleno**
  - **CR<6,5> Capacitación de interrupciones de transmisión**
    - » **SR<1> Indica reg de transmisión vacío**

Bit	7	6	5	4	3	2	1	0
CR	Capacita Interrupción por SR0	Control Transmisión		Selección Formato			Selección Frecuencia	
SR	IRQ*	Error paridad	Solapam. recepción	Error formato	CTS*	DCD*	TDR vacío	RDR lleno

CR6	CR5	Función
0	0	RTS* a baja, descapacita interrupciones por TDR vacío (SR1)
0	1	RTS* a baja, capacita interrupciones por TDR vacío (SR1)
1	0	RTS* a alta, descapacita interrupciones por TDR vacío (SR1)
1	1	RTS* a baja, descapacita interrupciones por TDR vacío (SR1), y pone TxD a baja ("break")

MC6850 HACIA
Asynchronous Communications Interface Adapter

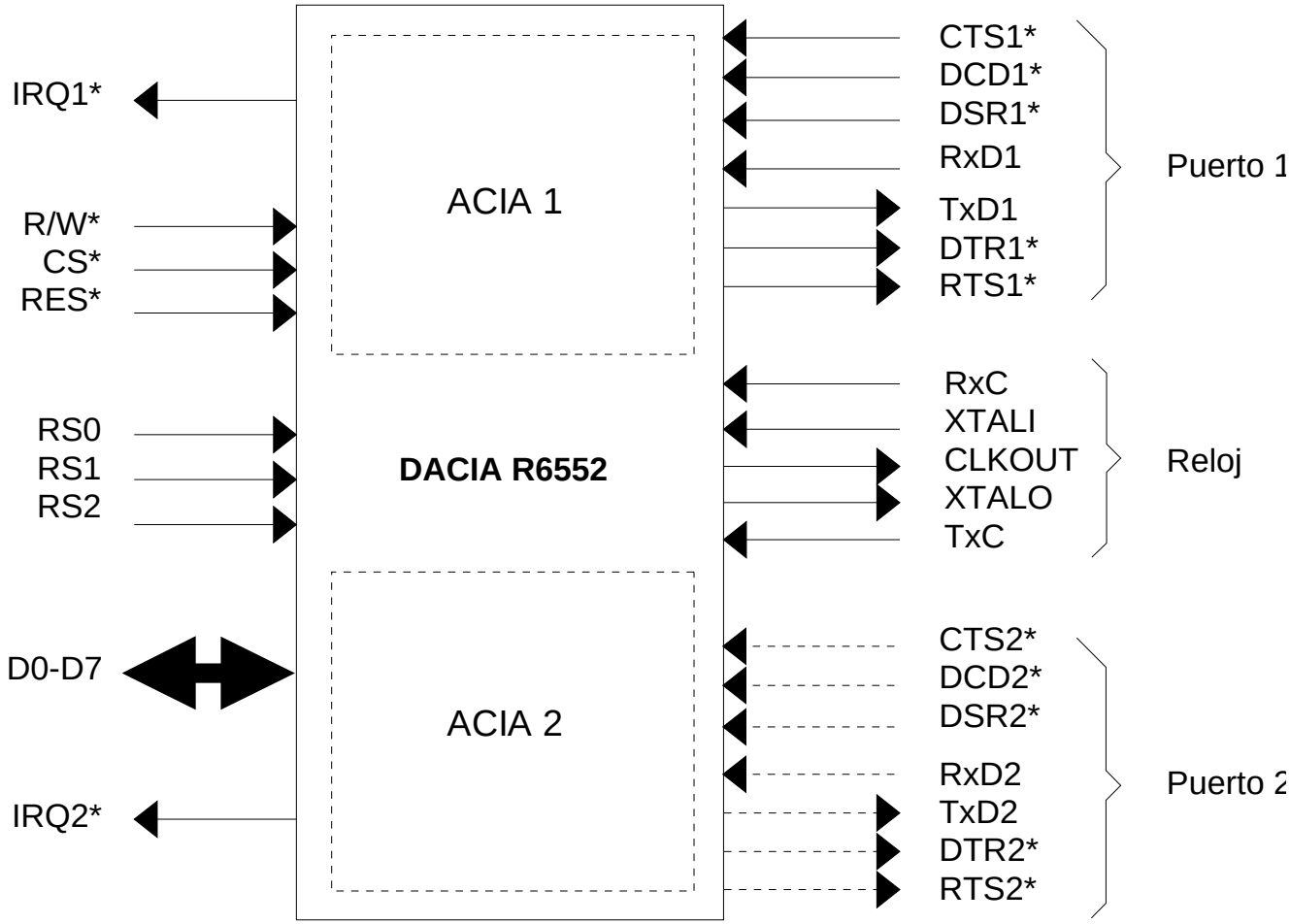
- **Selección de formato y frecuencia de transmisión**
  - **CR<4:2> formato de los datos**
  - **CR<1:0> frecuencia**

CR4	CR3	CR2	Función
0	0	0	7 bits datos + paridad par + 2 bits stop
0	0	1	7 bits datos + paridad impar + 2 bits stop
0	1	0	7 bits datos + paridad par + 1 bit stop
0	1	1	7 bits datos + paridad impar + 1 bit stop
0	0	0	8 bits datos + no paridad + 2 bits stop
0	0	1	8 bits datos + no paridad + 1 bit stop
0	1	0	8 bits datos + paridad par + 1 bit stop
0	1	1	8 bits datos + paridad impar + 1 bit stop

CR1	CR0	Función
0	0	División por 1
0	1	División por 16
1	0	División por 64
1	1	Reset ACIA

R6552 DACIA

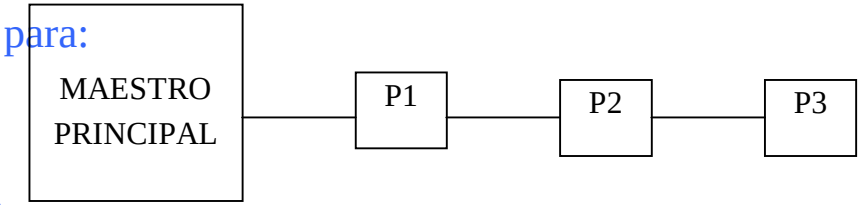
Dual Asynchronous Communications Interface Adapter



- **CONFIGURACIONES DE CONEXIÓN**
- **Existen dos formas de conectar los módulos de entrada salida y los dispositivos externos:**
- **Punto a punto**
- **Multipunto**
- **CONFIGURACIÓN PUNTO A PUNTO**
- **Línea dedicada entre el módulo de entradasalida y el periférico**
- **Se utiliza en pequeños sistemas como PC o estaciones de trabajo.**
- **Ej teclado, impresores, modems**
- **Ejemplo típico es el eia-232**
- **MULTIPUNTO**
- **una línea conecta un modulo de entrada salida con un conjunto de periféricos**
- **generalmente se utiliza para dispositivos de almacenamiento masivo como cintas o discos magnéticos y para los dispositivos multimedia, cdroms vídeo audio.**
- **De hecho se comporta como un bus externo. Los protocolos y arbitrajes son similares a los ya vistos en el capítulo de buses.**
- Ejemplo SCSI y p1394

;;;;;;;;;;;;;;;;;EJEMPLO DE BUS ENTRADA/SALIDA :SCSI

- ;;;;;;;;;;;;;;;;Small computer system interface
- Inicialmente Macintosh (1984)
- Actualmente Macintosh, sistemas compatibles pc y estaciones de trabajo
- Interfaz estándar para:
- CDROM,
- Equipos de audio,



- Para almacenamientos masivos
- Interfaz paralelo con 8,16 o 32 líneas de datos
- Aunque se suele nombrar como bus:
- ;;;;;;;;;;;;;;;;Todos los sistemas funcionan independientemente y se puede intercambiar datos entre ellos y con el host
- La longitud del cable es de 25 metros
- Se pueden conectar un totla de 8 dispositivos

EJEMPLO DE BUS DE ES SERIE: P1394

- ;;;;;;;;;;;;;;;;Bus serie de alto rendimiento
- Alta velocidad
- Bajo coste
- Sencillo de implementar
- Lo usan sistemas electrónicos que no son computadores: cámaras digitales, VCR, televisores
- Transporta imágenes digitalizadas.
- **Ventajas de la transmisión serie:**
- Menos hilos
- Menos pines que se pueden doblar y romper
- Evita transferencias entre hilos
- Evita problemas de sincronización entre las señales

- **Configuración**:
- daisy chain con 63 dispositivos conectados
- usando puentes entre diferentes buses se pueden conectar hasta 1022 dispositivos
- permite conectar y desconectar dispositivos sin tener que desconectar el sistema o reconfigurarlo
- Configuración automática
- No necesita terminadores
- No tiene que ser un daisy chai estricto puede utilizar una estructura de árbol
- Especifica tres capas de protocolo para estandarizar la forma en que el host actúa con los periféricos.