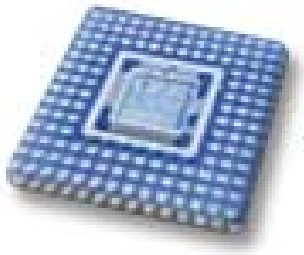
Arquitectura de Computadores

- Tema 1:
- CONEXIÓN DE PROCESADORES. BUSES



Departamento de
Arquitectura y
Tecnología de Computadores
UNIVERSIDAD DE SEVILLA

http:// www.atc.us.es

ÍNDICE (1)

- INTRODUCCIÓN: estructura de un computador y máquina de *Von Neumann*.

- INTRODUCCIÓN A LOS BUSES:
  - Características y tipos de señales
  - Clasificación de buses
  - Jerarquía de buses
  - Cronogramas ó Diagramas de Temporización

- LÍNEAS DE DATOS Y DIRECCIÓN
  - Organización de la memoria
  - Memorias ROM
  - Decodificación de espacios de memoria

ÍNDICE (2)

- LÍNEAS DE CONTROL
  - Señales de sincronización
  - Señales de arbitraje
  - Circuitería usada en la interfaz de bus
- EJEMPLOS DE CONEXIONES A PROCESADORES REALES:
  - La familia 80x86. El 8086.
  - La familia 68000. El 68000.
  - Circuitería adicional: el Reset y el Reloj.
  - Ejemplo de aplicación: diseño de un sistema de memoria para un 68000.
- EJEMPLO DE BUS REAL: EL BUS PCI

INTRODUCCIÓN (1)

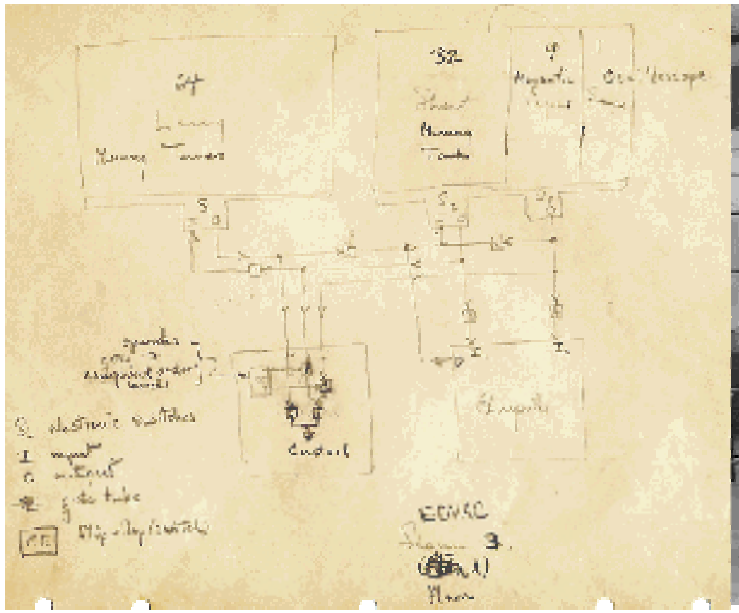
- En 1946 *John Louis Von Neumann* describió lo que son los fundamentos para la construcción y desarrollo de los todos los computadores.
- Instrucciones y datos se almacenan juntos en un medio uniforme: la memoria.
- Elemento en memoria ambiguo: ¿es dato o instrucción? Depende de cómo se opere o ejecute respectivamente.
- Presencia de registros internos: registro de instrucción y otro de datos.

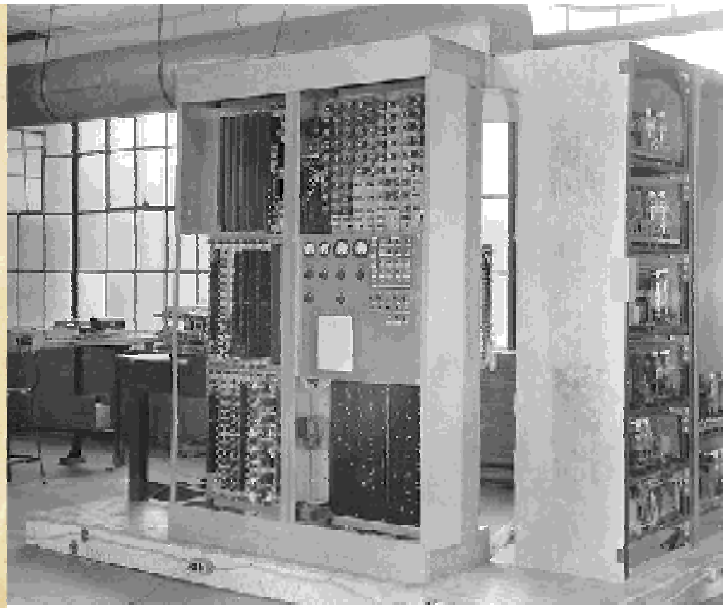
INTRODUCCIÓN (2)

- Ciclo repetitivo de pasos: localizar y ejecutar en secuencia las instrucciones de un programa.
- Con el modelo *Von Neumann* se construyeron: EDVAC (1949, primera con almacenamiento masivo memoria), EDSAC (1949, ordenador electrónico) y UNIVAC (1951, primer ordenador comercial y uso compilador).
- Tres componentes fundamentales: CPU ó procesador, memoria y sistema de Entrada/salida (E/S) de datos. ¿Bus el cuarto componente?

INTRODUCCIÓN (3)

EDVAC





Emplea 4000 bulbos y su capacidad de memoria es de 1024 palabras de 44 bits.

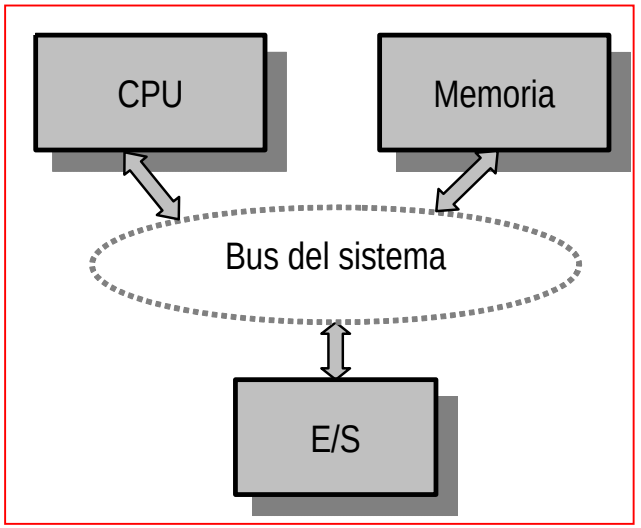
INTRODUCCIÓN (4)

- Elementos de un Sistema Basado en Computador:
  - **CPU ó PROCESADOR**: único elemento activo que:
    - Maneja datos e instrucciones.
    - Realiza operaciones: transferencia, aritméticas, lógicas, etc. (Posee un *Juego de Instrucciones*)
    - Se dice elemento activo porque recibe una serie de datos y genera otros diferentes.
  - **MEMORIA**: elemento (pasivo) que:
    - Almacena datos/instrucciones, sin alterarlos, en un conjunto de celdas, direccionables por el procesador, para lectura o escritura.
    - *Capacidad de direccionamiento* de una memoria: $M=2^n$.
    - La capacidad se define en Kilobytes (2^{10}), Megabytes (2^{20}), ó Gigabytes (2^{30}).
    - Mapa de memoria de una CPU: datos, programas y E/S.

INTRODUCCIÓN (5)

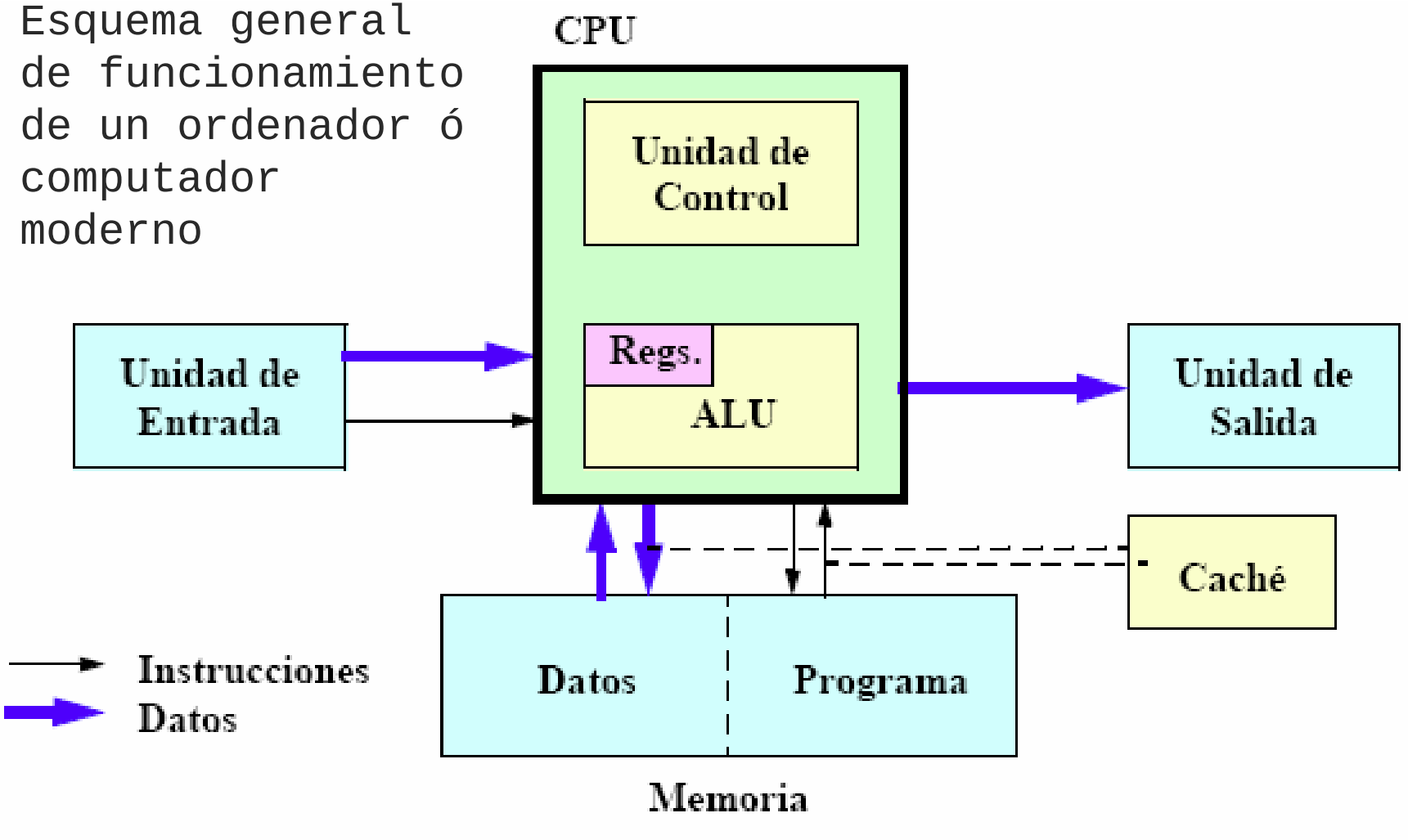
- **SUBSISTEMA DE ENTRADA/SALIDA (E/S):** elemento que comunica el computador con el exterior.
  - El computador ve el subsistema de E/S como un conjunto de posiciones de memoria sobre las que se puede escribir o leer datos.
- **BUS DEL SISTEMA (BUS): e**lemento que comunica la CPU, MEMORIA y E/S.

Arquitectura general de un ordenador ó computador moderno



INTRODUCCIÓN (6)

Esquema general de funcionamiento de un ordenador ó computador moderno



ÍNDICE (1)

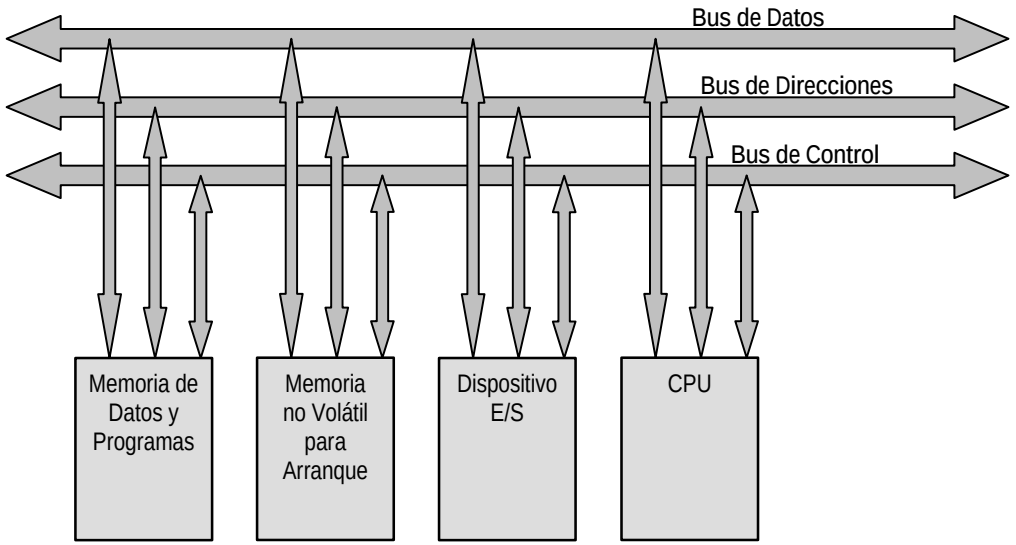
- INTRODUCCIÓN: estructura de un computador y máquina de *Von Neumann*.

- INTRODUCCIÓN A LOS BUSES:
  - Características y tipos de señales
  - Clasificación de buses
  - Jerarquía de buses
  - Cronogramas ó Diagramas de Temporización

- LÍNEAS DE DATOS Y DIRECCIÓN
  - Organización de la memoria
  - Memorias ROM
  - Decodificación de espacios de memoria

INTRODUCCIÓN A LOS BUSES

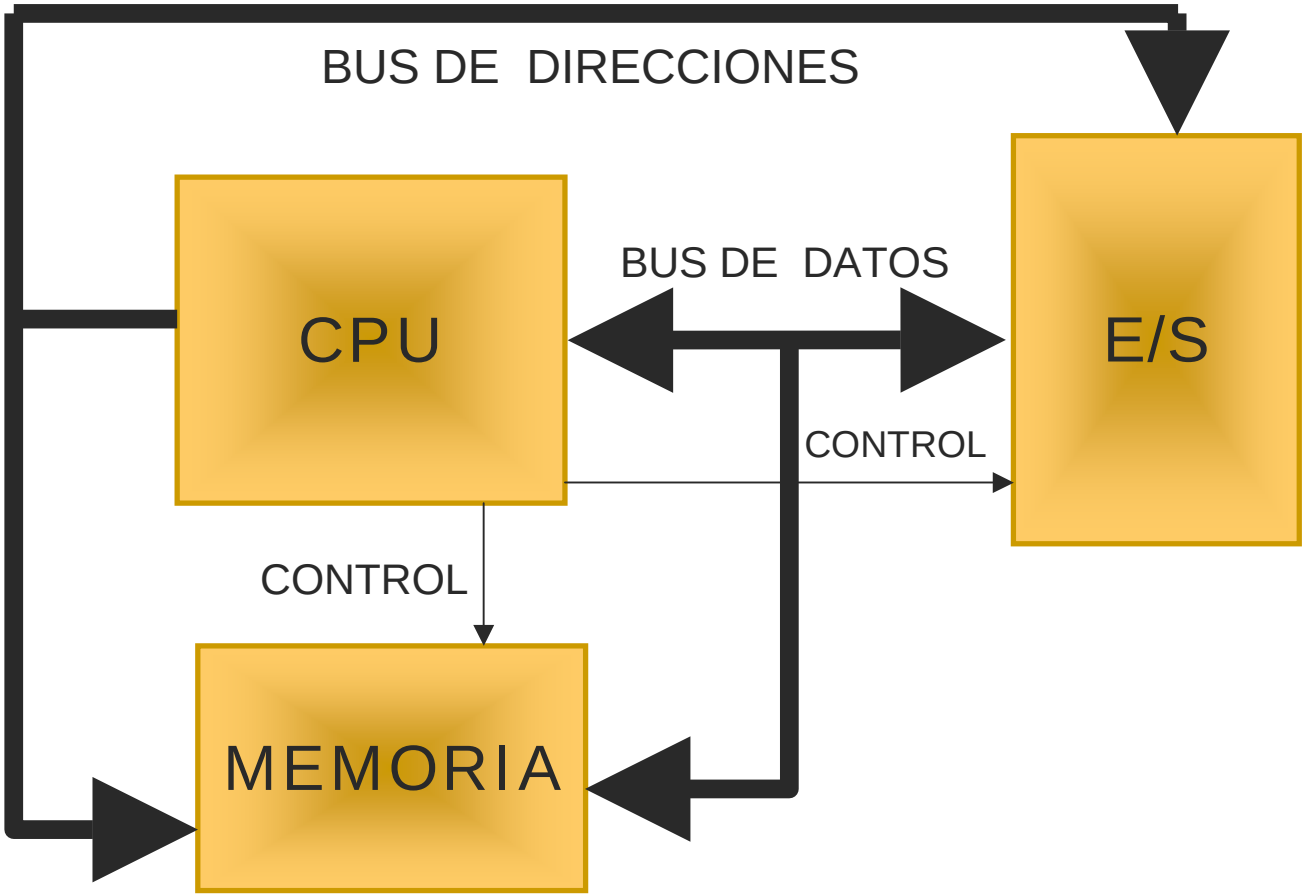
- Los componentes de un computador (CPU, Memoria, E/S) se conectan entre sí mediante un conjunto de líneas que transmiten señales con funciones específicas.
- Tres tipos de señales que constituyen un bus: direcciones (de memoria o E/S), datos y control.



Características y tipos de señales de buses (1)

- Los buses se componen de líneas eléctricas que transmiten un “0” (cero voltios) o un “1” (más de cero voltios).
- Líneas/bus de datos: camino para transferir datos entre el resto de componentes de un computador. Su anchura (número de líneas eléctricas) suele ser una potencia de dos ($8=2^3$, $16=2^4$, $32=2^5$, $64=2^6$, ...).
- Líneas/bus de direcciones: designan la posición/dirección de los datos. Son salidas de la CPU/procesador y determinan capacidad de direccionamiento.
- Líneas/bus de control: controlan el acceso y uso de las líneas/buses anteriores.

Características y tipos de señales de buses (2)



Clasificación de buses

- Tipos de buses:
  - SERIE y PARALELO: los primeros transmiten bit a bit y los segundos varios bits a la vez.
  - MULTIPLEXADOS y NO MULTIPLEXADOS ó DEDICADOS: los multiplexados realizan diferentes funciones en función de las necesidades del momento.

    Ejemplo: bus compartido para direcciones y datos → ahorro en Hardware y por lo tanto en costes.
  - CENTRALIZADOS y DISTRIBUIDOS (*arbitración*): necesidad de determinar qué elemento transmite y cuál recibe. Generalmente existe arbitración centralizada por la CPU ó procesador.
  - SÍNCRONOS y ASÍNCRONOS (*temporización*): cómo ocurren los diferentes eventos (comienzo, fin, ...) implicados en la transmisión de información. Utilización de una señal de reloj (comunicación síncrona) ó unas *líneas de protocolo* (comunicación asíncrona).

Jerarquía de buses (1)

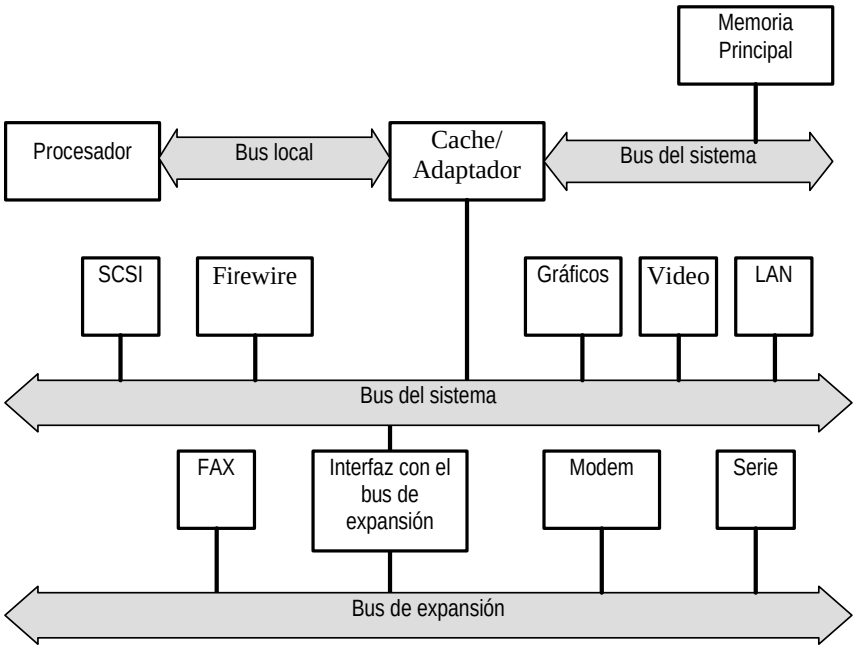
- Compatibilidad entre buses:

  sólo si son eléctricamente idénticos. Las características de los diferentes tipos de buses deben estar *normalizadas*. Ejemplo: bus PCI, AGP, USB, FireWire...
- Antiguamente sólo existía un bus principal que lo conectaba todo: bus del sistema.
- Actualmente existe un conjunto de buses conectados entre sí y formando una jerarquía.
- Facilita la mejora del rendimiento de todo el computador al agrupar dentro de los diferentes tipos de buses aquellos componentes del ordenador que tienen aproximadamente la misma velocidad de transmisión de la información.

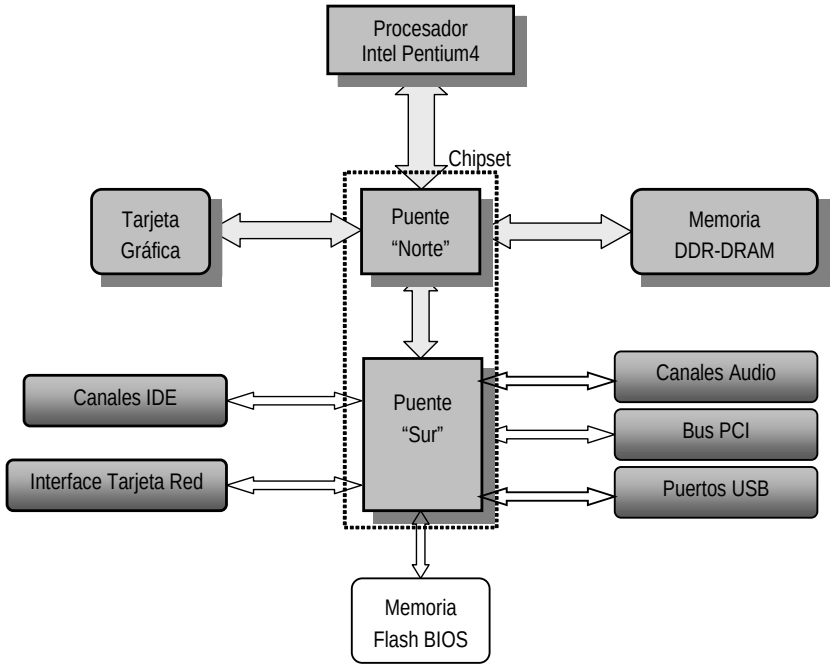
Jerarquía de buses (2)

- Contra más lejos de la CPU, buses más lentos y normalmente de menos líneas de datos.
- Varios tipos de buses en función de su posición dentro de la jerarquía:
  - Bus de CPU ó “bus local” del procesador: elementos más rápidos tales como la memoria caché.
  - Bus local ó bus del sistema (*Front Side Bus*): conecta elementos tales como la memoria principal o dispositivos rápidos (por ejemplo AGP).
  - Bus de expansión y/o E/S: PCI, USB, ATA, SCSI, ...

Jerarquía de buses (3)



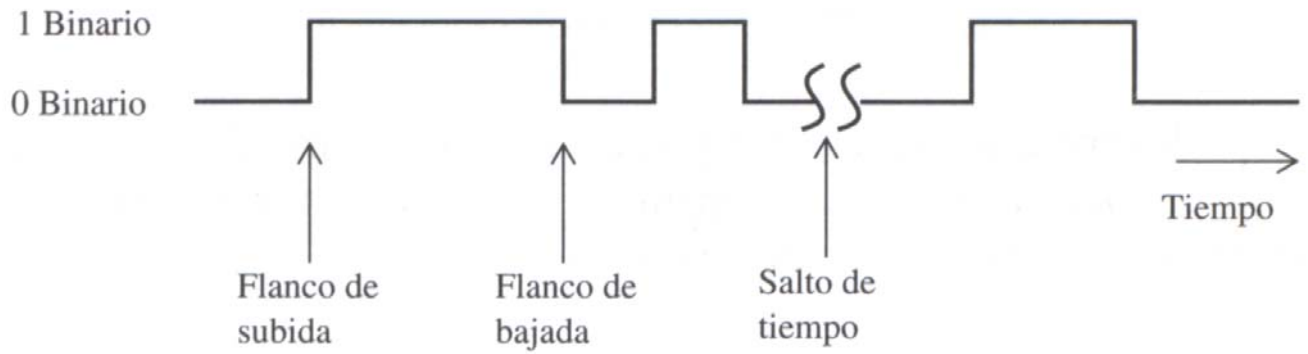
Esquema típico de jerarquía de buses en un ordenador. Los buses de arriba son los más rápidos y el bus de expansión el más lento.



Esquema de conexión de componentes en un PC. El chip "Puente Sur" agrupa los buses más lentos y el "Puente Norte" los más rápidos.

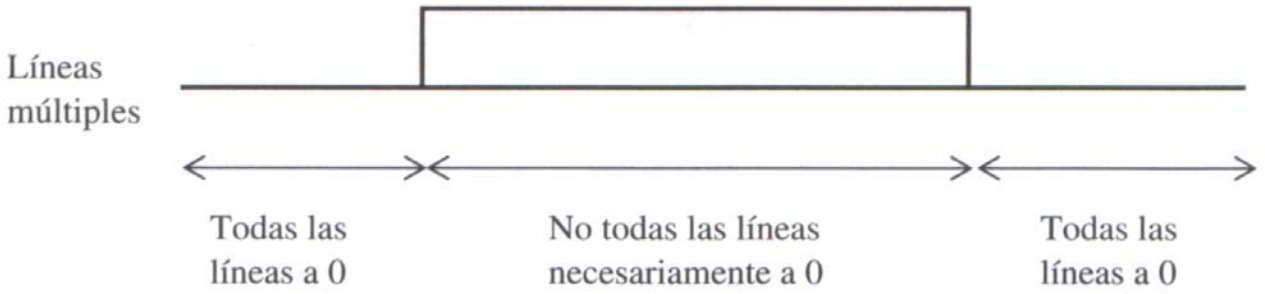
Cronogramas o Diagramas de Temporización (1)

- Muestran cómo ocurren las secuencias de acciones y la relación causa-efecto entre diferentes sucesos → Muestran comunicación entre dos dispositivos conectados a través de un bus.
- Las líneas de un bus tienen dos niveles de señal o tensión → 0 ó 1.



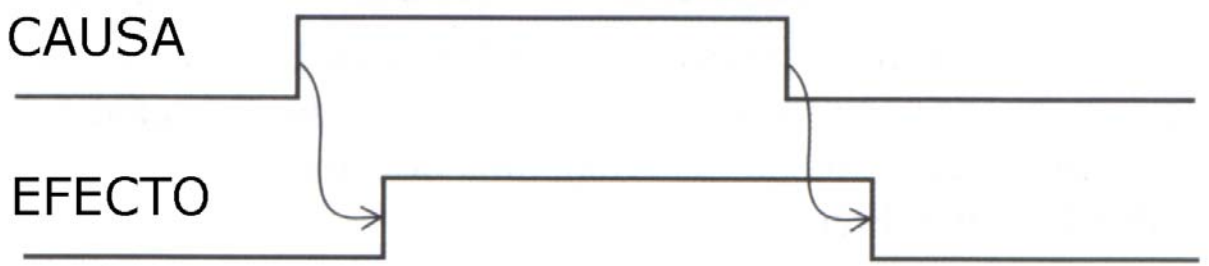
Cronogramas o Diagramas de Temporización (2)

- Flanco subida: transición señal 0 a 1.
- Flanco bajada: transición señal 1 a 0.
- Se requiere cierto tiempo para estabilizar las transiciones de las señales (zona metaestabilidad).
- Señal triestado: 0, 1 o alta impedancia (desconectada).
- Varias señales pueden formar grupos y se representan como una línea compuesta.



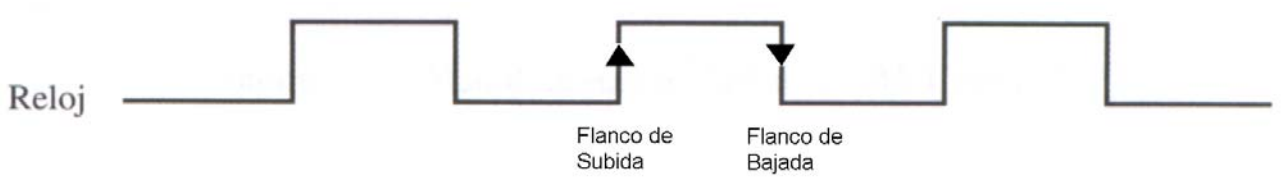
Cronogramas o Diagramas de Temporización (3)

- La transición de una señal en un dispositivo puede dar lugar a transiciones en las señales de otros dispositivos.

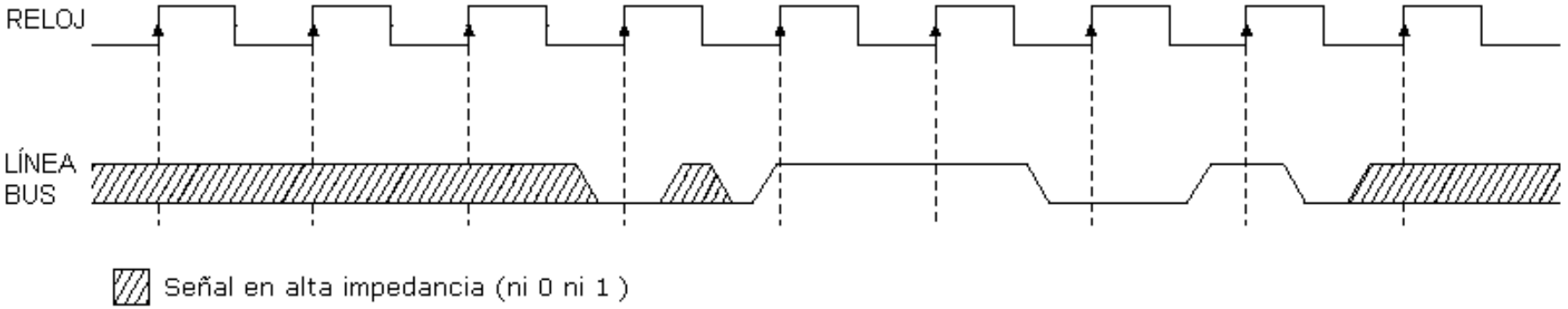


- En los buses denominados síncronos existe una señal de reloj que sirve para sincronizar sucesos.
- Es una señal periódica y repetitiva.

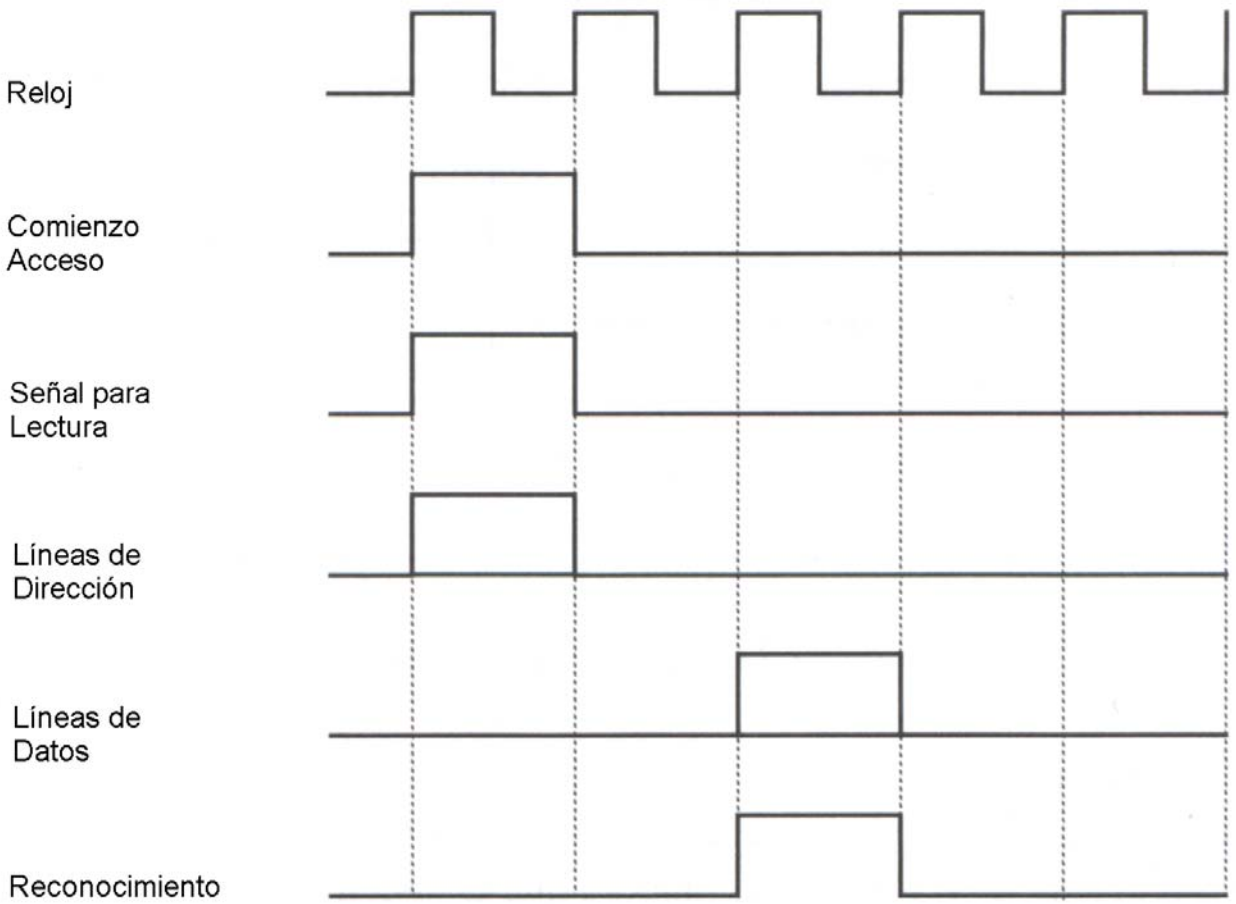
Cronogramas o Diagramas de Temporización (4)



- Generalmente en los buses síncronos se lee el valor de una señal a la subida o a la bajada de la señal de reloj.



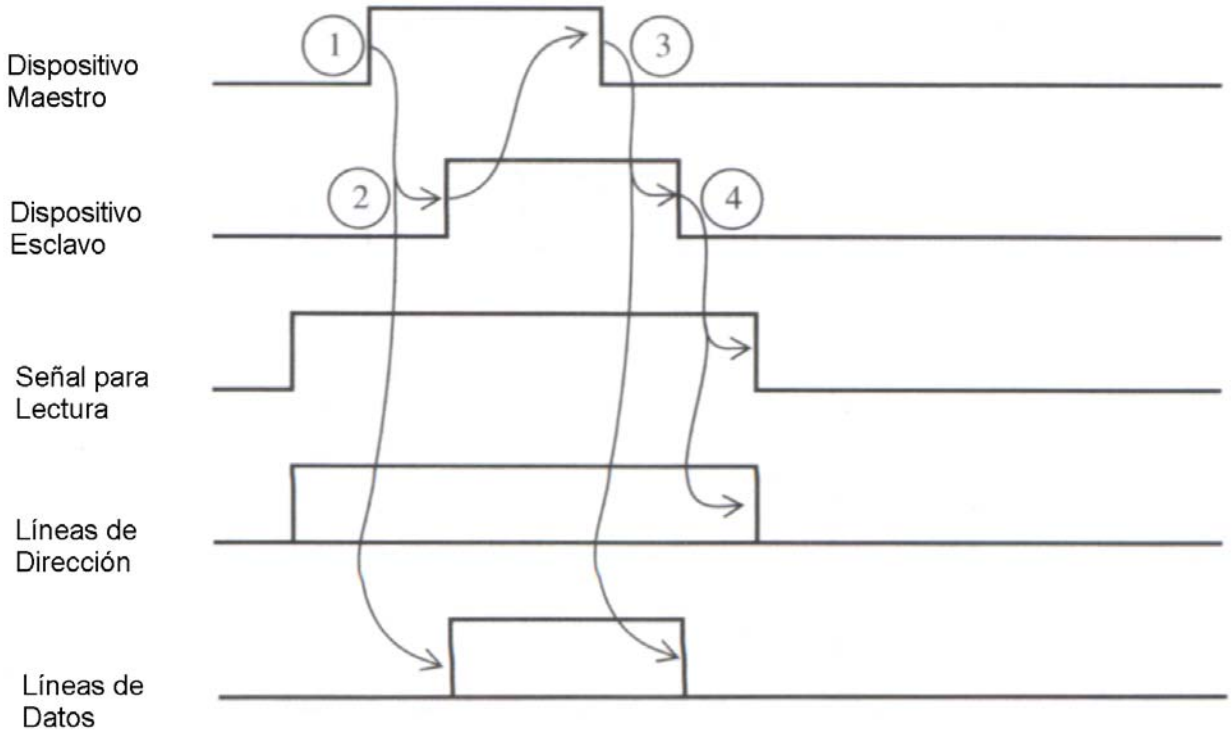
Cronogramas o Diagramas de Temporización (5)



- Esquema cronograma para una operación de lectura síncrona.
- La CPU emite una señal de lectura y coloca dirección en bus direcciones.
- Cada suceso ocupa un ciclo de reloj.

Cronogramas o Diagramas de Temporización (6)

• Esquema Cronograma para una operación de lectura asíncrona.



• 1) El dispositivo *Maestro* solicita acceso: indica hay dirección y señales de control válidas.

• 2) El *Esclavo* realiza la tarea pedida y activa su señal al terminar.

• 3) El *Maestro* coge el dato válido y lo almacena y desactiva señal.

• 4) El *Esclavo* detecta que el Maestro ha desactivado su señal y desactiva la suya.

ÍNDICE (1)

- INTRODUCCIÓN: estructura de un computador y máquina de *Von Neumann*.

- INTRODUCCIÓN A LOS BUSES:
  - Características y tipos de señales
  - Clasificación de buses
  - Jerarquía de buses
  - Cronogramas ó Diagramas de Temporización

- LÍNEAS DE DATOS Y DIRECCIÓN
  - Organización de la memoria
  - Memorias ROM
  - Decodificación de espacios de memoria

LÍNEAS DE DATOS Y DIRECCIÓN (1)

- Línea de datos únicas realmente necesarias.
- Líneas dirección y control necesarias para correcto funcionamiento. Ejemplo: bus serie.
- Ancho de banda, *data rate* ó bits por unidad de tiempo: generalmente expresado en bits/segundo. Ejemplo: bus PCI de 64 bits a 66MHZ → 64 bits * $66*10^6$ = 4224 Mbps = 528 MB/s
- Lo normal es que los buses conectados a la CPU sean del tamaño de palabra del procesador.

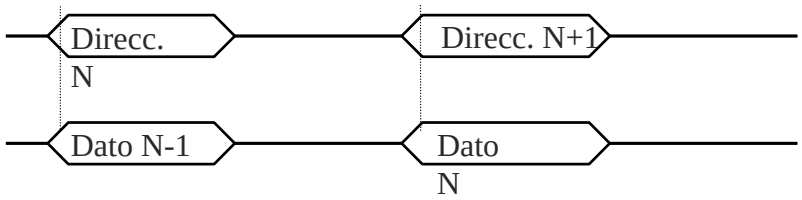
LÍNEAS DE DATOS Y DIRECCIÓN (2)

- Buses más pequeños: menor patillaje, opción de bajo coste.
- Buses más grandes: aumento del ancho de banda. Usados en procesadores superescalares que son capaces de ejecutar más de una instrucción por ciclo.

- Líneas de dirección: indican localización dato para L/E y determinan número posiciones accesibles. Ejemplo: en el 68000 hay 24 bits direcciones → 2^{24} bytes = 16 MB.
- Es posible dividiendo la fase de direcciones en dos ciclos/partes (o más), acceder a más posiciones. Ejemplo: bus PCI.

LÍNEAS DE DATOS Y DIRECCIÓN (3)

- A veces se realiza el multiplexado en el tiempo: se comparten los buses de datos y direcciones a través de las mismas líneas.
  - Ventajas: se disminuye el patillaje del procesador y así se ahorran costes.
  - Desventajas: se necesitan señales externas para diferenciar ambos buses y circuitería para separar ambos valores. Es más lento.
- Los buses NO multiplexados son más rápidos al poder indicar direcciones y recibir datos a la vez.



LÍNEAS DE DATOS Y DIRECCIÓN (4)

- Actualmente una misma dirección provoca varios accesos (consecutivos). Por ello ventaja buses NO multiplexados desaparece. Ejemplo: acceso en modo ráfaga del bus PCI.
- La memoria es más lenta que el procesador → ciclos partidos ó *split transactions*: la CPU una vez transmitida la dirección del dato libera el bus. Luego espera a que la memoria o el periférico de E/S ponga el resultado en un *buffer* o comunique dato disponible a través del bus de control.

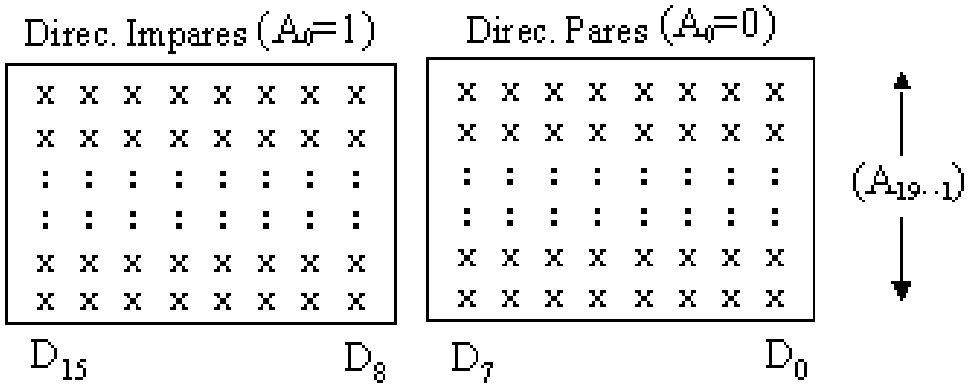
LÍNEAS DE DATOS Y DIRECCIÓN (5)

- Ventaja: aumenta utilización del bus → otros dispositivos lo pueden usar.
- Desventajas: el dato no se lee cuando está disponible, se compiten dos veces por el bus, ...

ORGANIZACIÓN DE LA MEMORIA

- En la memoria principal (memoria RAM) y secundaria la unidad mínima de información suele ser el *byte*.
- La palabra del procesador se compone a su vez de varios *bytes*.

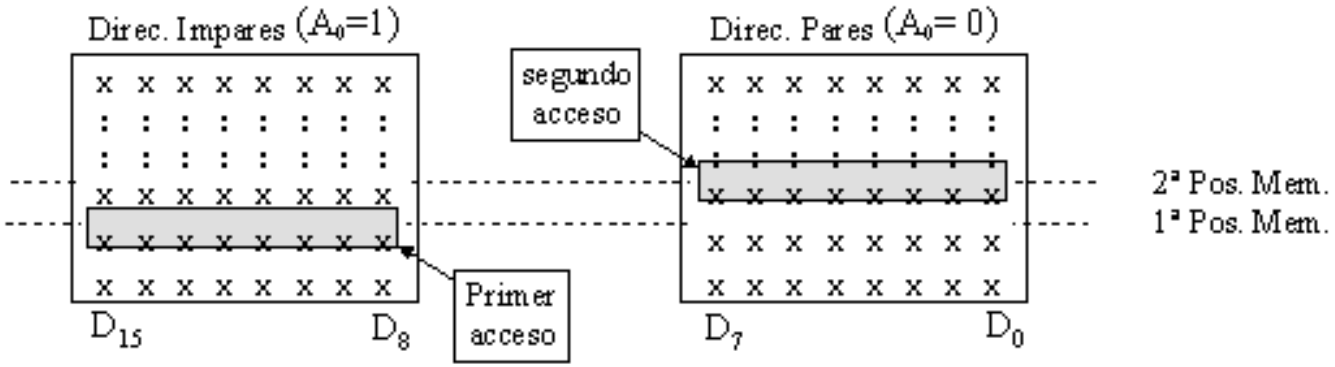
Organización de la memoria (1)



En el esquema se ve un posible esquema de memoria para un 80286 con tamaño de palabra de 16 bits: dos bancos de 1 *byte* y 20 líneas de dirección (A_0 ...A_{19}). Con el bit A_0 se selecciona el banco: *byte* par o *byte* impar de la palabra.

- Alineamiento: un procesador accede a posiciones de memoria pares (A_0 = 0) para optimizar los accesos a memoria. Estos accesos suponen acceso en paralelo de los bancos de memoria, lo cual, evita dos accesos a memoria para leer una palabra de procesador. Ejemplo:

Organización de la memoria (2)



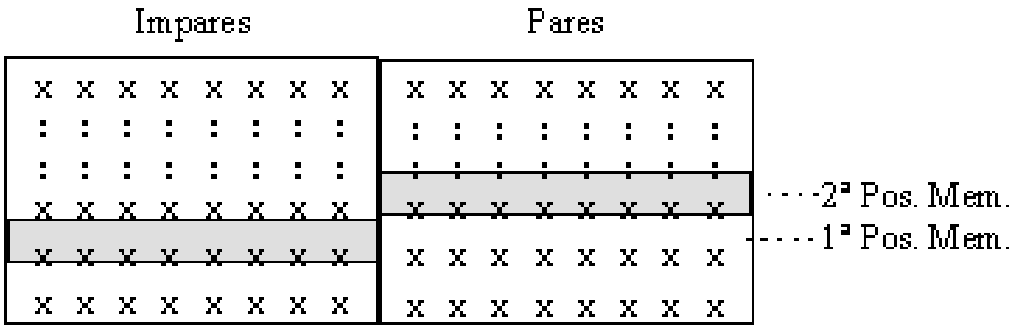
En el esquema el procesador accede a una “palabra impar” de 16 bits. Eso implica acceder primero a el banco 1 de memoria y después al banco 0.

- El alineamiento es importante para acceder a palabras de una forma eficiente. Los microprocesadores luego pueden acceder a un *byte* dentro de la palabra. Ejemplo: señal BHE* en el procesador 8086.

	(A_0)	BHE*
Byte par	0	No act.
Byte impar	1	Activa
Word (par)	0	Activa

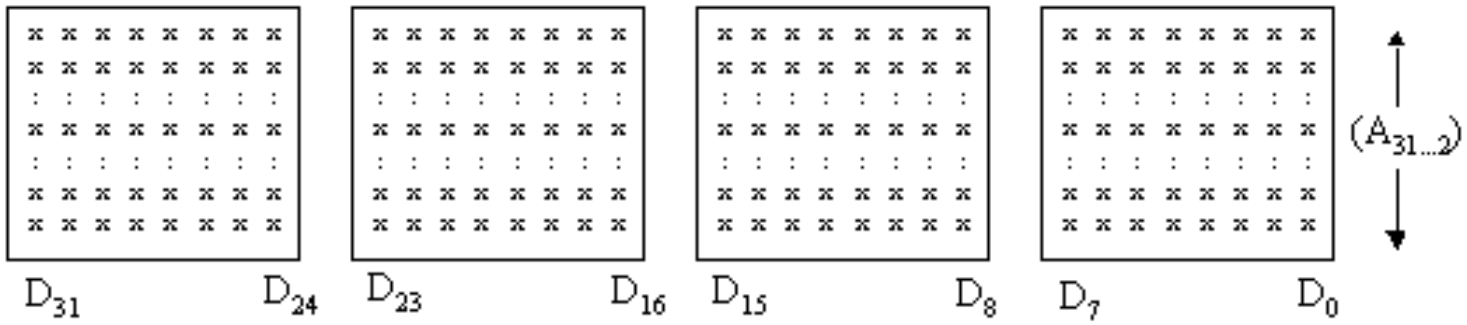
- Algunos microprocesadores como el 68000 exigen alineamiento de palabras.

Organización de la memoria (3)



El siguiente alineamiento de palabras en memoria no se permite en un procesador de la familia 68000. El objetivo es que al leer una posición par de memoria se lea una palabra completa.

- Estas formas de direccionamiento se pueden generalizar a más bancos. Ejemplo: el 80586 (Pentium) tiene palabras de 32 bits. Un posible esquema de memoria sería:



Organización de la memoria (4)

- Aunque el programador trabaja con 2^{32} y puede direccionar con los bits A31.. A0, externamente se utilizan A31.. A2 como dirección, de cada palabra de 32 bits, y los dos últimos bits activan las señales de control /BE0-/BE3 que controlan el banco donde se accede.

- El Motorola 68000 tiene 16 bits para datos, en dos bancos de un *byte*, y 23 bits para direcciones de manera que A23.. A1 apuntan a 2^{23} palabras ($\equiv 2^{24}$ *bytes*). A0 no existe. A cambio tiene dos señales, UDS* y LDS* que sirven para acceder a uno y/o otro banco.

Memorias ROM (1)

MEMORIAS ROM

- Memorias no volátiles. Borrado/escritura de datos “costosa”.
- Varios tipos en función de su forma de borrado: ROM,PROM, EPROM, EEPROM, Flash EEPROM (utilizada para guardar la BIOS en PCs actuales).
- Diferentes capacidades (Nº posiciones x Bits por posición): 1Kx8, 16Kx8, 32Kx8, 64Kx8, ...
- Buses de datos y direcciones.

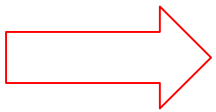
Memorias ROM (2)

- Señales de control:
  - CS* ó CE* (*Chip Select ó Chip Enable*): la memoria empieza a trabajar decodificando la dirección.
  - OE* (*Output Enable*): indica ciclo de lectura
  - WE* (*Write Enable*): indica ciclo de escritura. Incompatible con OE*.

Conjunto de pines del chip 62256P-10 (32Kx8).

•Bus de direcciones (A_0 – A_{14}): $2^{15} = 2^5$Ks = 32Ks

•Bus de datos (I/O_1 – I/O_8): 8 bits = 1 *byte*



1	A14	VCC	28
2	A12	$\overline{WE}$	27
3	A7	A13	26
4	A6	A8	25
5	A5	A9	24
6	A4	A11	23
7	A3	$\overline{OE}$	22
8	A2	A10	21
9	A1	$\overline{CS}$	20
10	A0	$I/O8$	19
11	$I/O1$	$I/O7$	18
12	$I/O2$	$I/O6$	17
13	$I/O3$	$I/O5$	16
14	GND	$I/O4$	15

Memorias ROM (3)

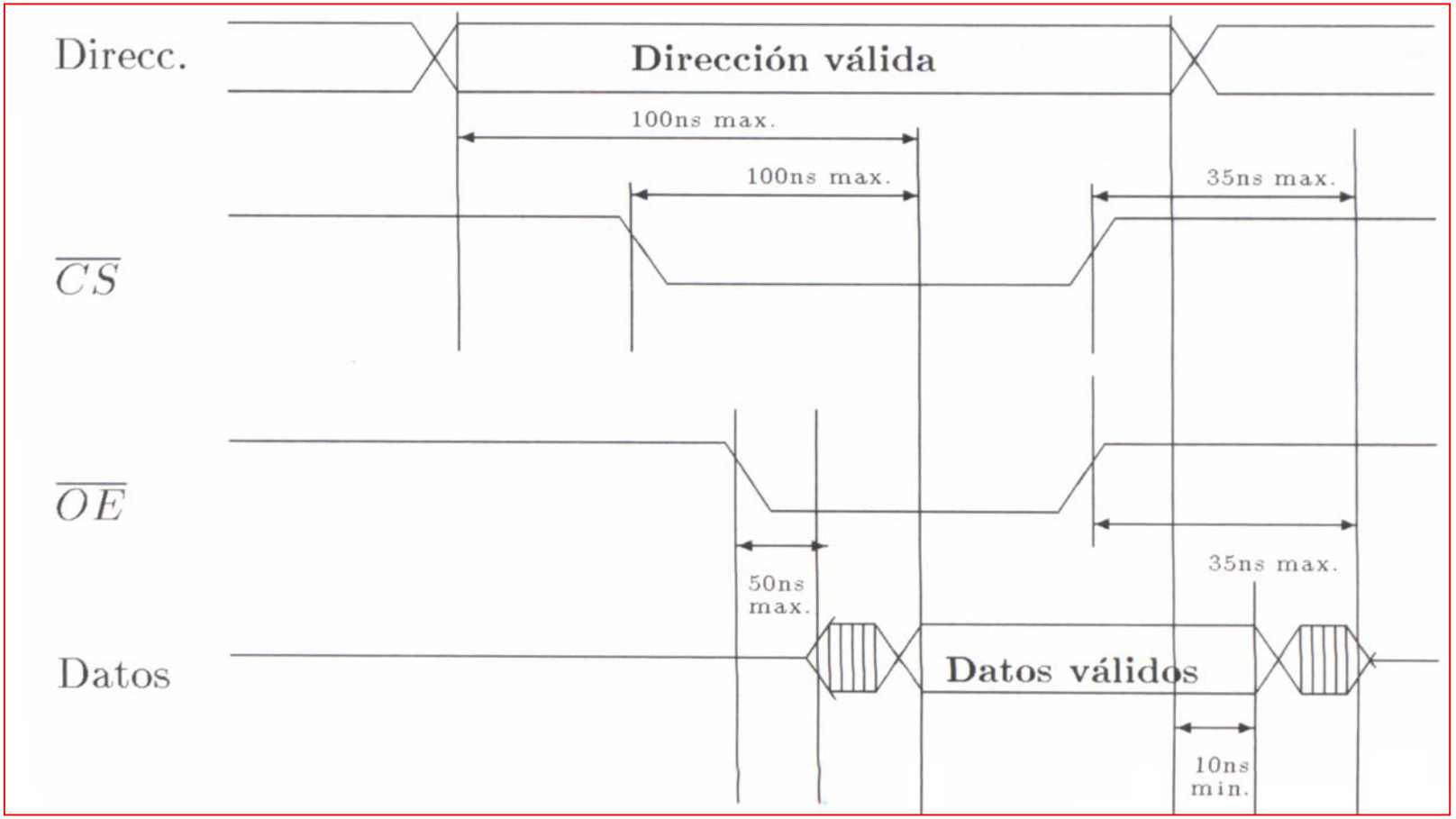


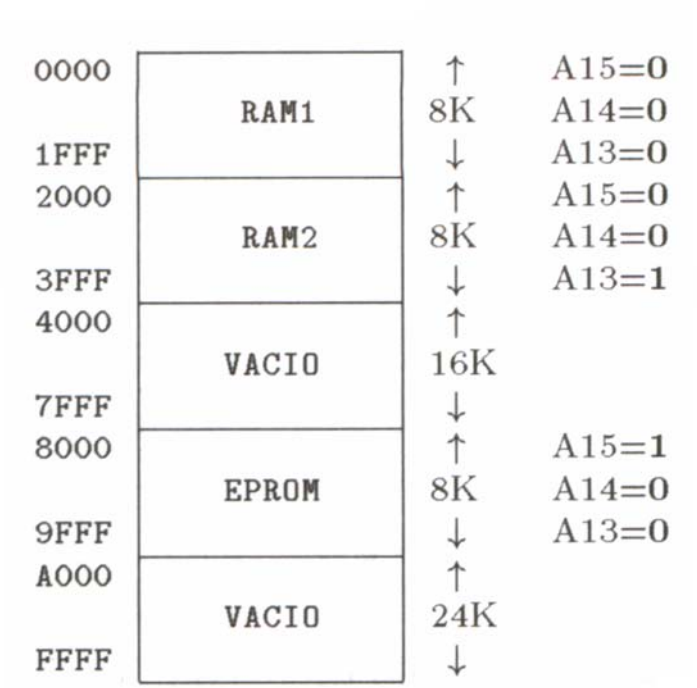
Diagrama de ciclo de lectura simplificado de memoria 62256P-10 (32Kx8)

Decodificación de espacios de memoria (1)

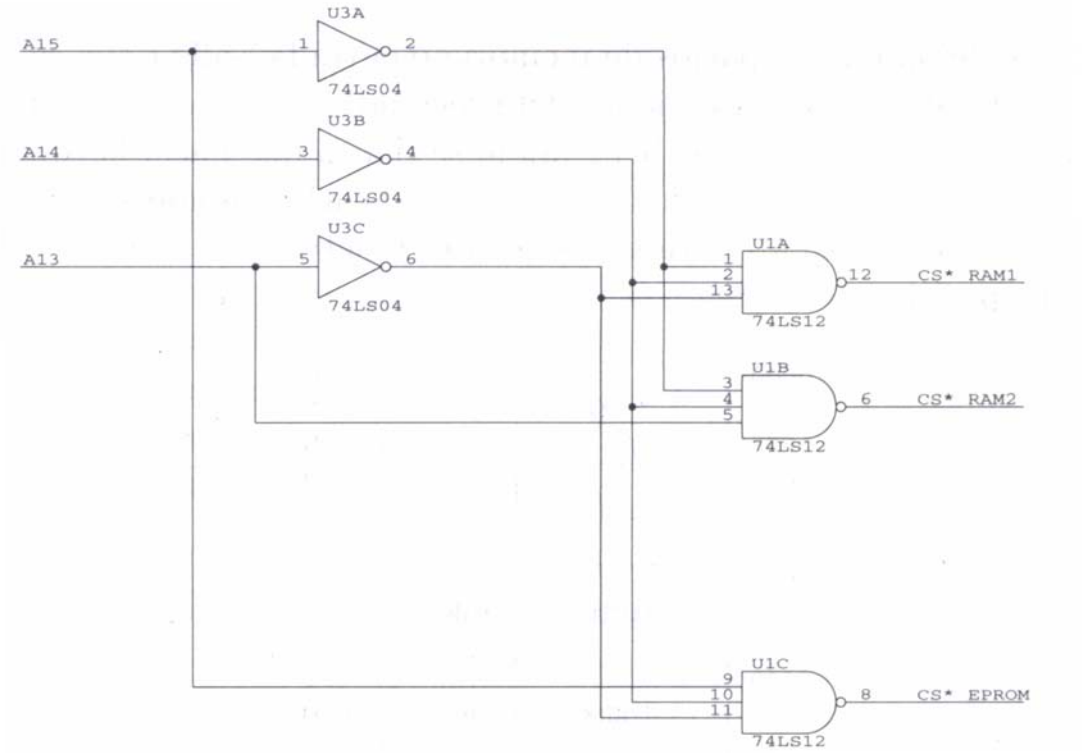
DECODIFICACIÓN DE ESPACIOS DE MEMORIA

- La memoria total que el procesador puede direccionar se divide en varios espacios de memoria.
- Cada espacio de memoria puede tener asociado un determinado tipo de memoria (uno o varios *chips* de memoria tipo ROM ó RAM).
- Las direcciones que envía el procesador tienen que ser decodificadas para activar unos chips de memoria u otros.
- Las activaciones de memoria (en los *chips*) se realizan mediante la señal CS* (*chip select*).

Decodificación de espacios de memoria (2)



Distribución de espacios de memoria en el espacio de direcciones de un procesador.



Decodificación total del espacio de direcciones.

Decodificación de espacios de memoria (3)

- Dos tipos de decodificación de direcciones:
  - Decodificación total: cada posición de memoria (*byte*, palabra, ...) sólo puede ser decodificada por una dirección del procesador. Usada en sistemas de propósito general.
  - Decodificación parcial: una posición de memoria puede ser accedida mediante varias direcciones que envía el procesador. Sencilla pero no utiliza toda la capacidad de la memoria.
- Para hacer la decodificación se pueden emplear decodificadores. En realidad se suelen emplear PALs (*Programmable Array Logic*), PLAs (*Programmable Logic Array*) o circuitos integrados específicos.

ÍNDICE (2)

- LÍNEAS DE CONTROL
  - Señales de sincronización
  - Señales de arbitraje
  - Circuitería usada en la interfaz de bus
- EJEMPLOS DE CONEXIONES A PROCESADORES REALES:
  - La familia 80x86. El 8086.
  - La familia 68000. El 68000.
  - Circuitería adicional: el Reset y el Reloj.
  - Ejemplo de aplicación: diseño de un sistema de memoria para un 68000.
- EJEMPLO DE BUS REAL: EL BUS PCI

LÍNEAS DE CONTROL

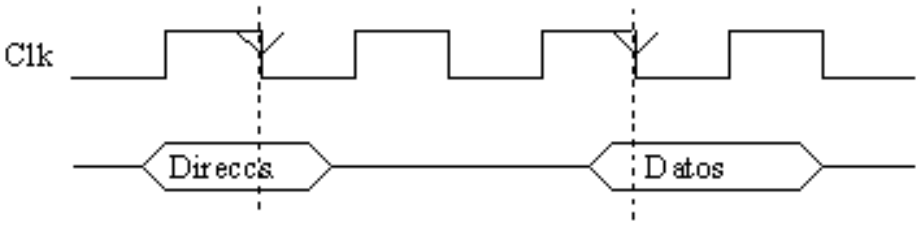
IDENTIFICADOR DE CICLO DE BUS

- Ciclo de bus: periodo de tiempo en el que el procesador realiza un acceso exterior (no registros internos) a memoria o E/S.
- Tiene que haber líneas de control dentro del bus de control que distinga el tipo de acceso: lectura, escritura, memoria, E/S, instrucciones, datos, interrupciones, ...
- Dos posibilidades para activación de señales:
  - El procesador activa directamente las señales de control. Ejemplo: 80x86, señales rd*, wr*, M/IO, ...
  - El procesador indica su estatus y una circuitería externa activa señales de control. Ejemplo: 68000.

Señales de sincronización (1)

SEÑALES DE SINCRONIZACIÓN

- Un ciclo de bus tiene varios pasos → indicados por señales de control para que los eventos ocurran en el orden correcto.
- Tres tipos de sincronización:
  - Sistemas SÍNCRONOS:
    - Gobernados por una señal de reloj periódica.
    - Indica cuando leer, escribir, cuantos ciclos hay que mantener una señal, ...

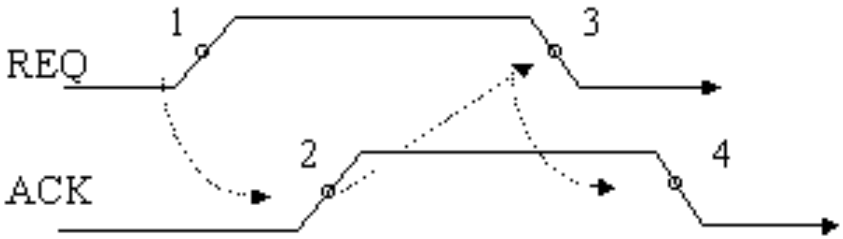


Señales de sincronización (2)

- Típico de buses locales o de sistema.
- Requiere que todos los dispositivos (memoria, CPU, periféricos de E/S) tengan la misma señal de reloj. En frecuencias altas el retraso de propagación de la señal entre los diferentes dispositivos alejados puede provocar desfases temporales.
- Ejemplo de sistema/bus síncrono: el bus PCI.

- Sistemas ASÍNCRONOS:
  - Utilizan señales del bus de control que sincronizan los diferentes pasos en la comunicación.
  - Ejemplo: protocolo de cuatro eventos ó *handshake* (apretón de manos).

Señales de sincronización (3)

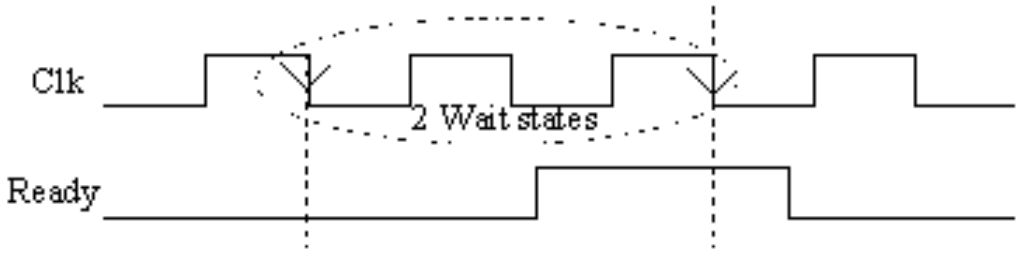


1: Request (CPU) ≡ dirección válida
2: ACK (Mem) ≡ acuse de recibo
3: CPU Lee ≡ Fin REQ.
4: FIN.

- Sistemas SEMISÍNCRONOS, MIXTOS ó de BUS SÍNCRONO:
  - Buses síncronos a los que se les añaden señales asíncronas para comunicarse con dispositivos lentos.
  - Los dispositivos rápidos pueden tener que esperar ciertos ciclos de reloj hasta la activación de alguna señal por parte del dispositivo lento.

Señales de sincronización (4)

- A estos tiempos de espera se les llama estados de espera (*wait states*).
- Ejemplo: el procesador 68000 y la señal DTACK* (*Data Acknowledgement*).

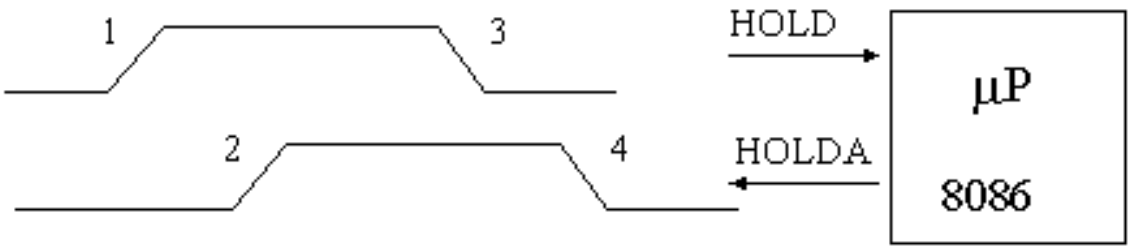


SEÑALES DE ARBITRAJE

- ¿Cómo y quién controla el acceso al bus?

Señales de arbitraje (1)

- Un dispositivo *Master* (Maestro) puede controlar el bus mientras que un dispositivo *Slave* (Esclavo) interpreta las señales del *Master*.
- Pueden existir varios dispositivos *Master*.
- Ejemplo: controlador de DMA (*Slave*) en arquitecturas 80x86 que realiza transferencias entre la memoria y los periféricos de E/S. Pide permiso de utilización del bus del sistema a la CPU (*Master*) mediante la señal HOLD. La CPU concede permiso con señal HOLDA.



1) El DMA pide permiso.

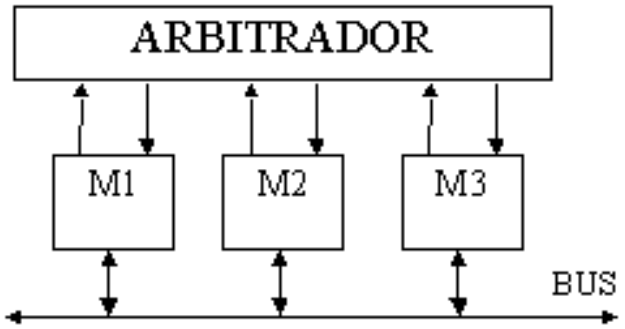
2) El µP da permiso.

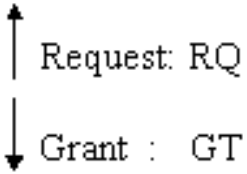
3) El DMA devuelve el control.

4) El µP retoma el control.

Señales de arbitraje (2)

- La arbitración de un bus puede ser centralizada o distribuida.
- En la arbitración centralizada un dispositivo hace de arbitrador y la arbitración puede ser en paralelo.



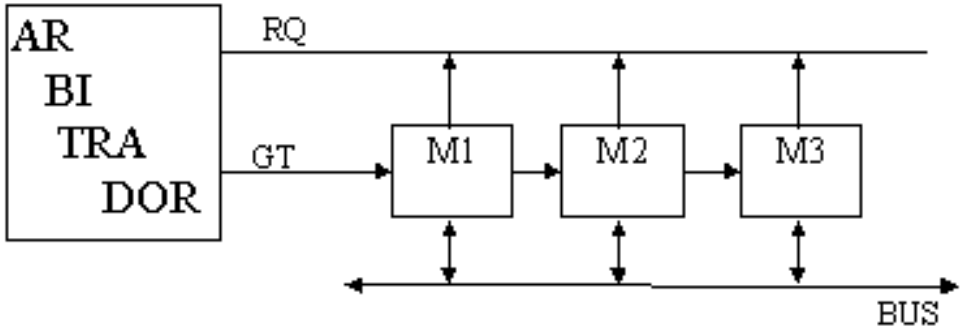


Ante varias peticiones RQ se activa una señal de concesión GT.
Sistema de concesión variado: rotación, prioridades, ...

Ejemplo: bus PCI.

Señales de arbitraje (3)

- Configuración en *Daisy Chain* (cadena de margaritas): la prioridad viene determinada por la proximidad del *Slave* al arbitrador.



- En la arbitración distribuida todos los *Master* se ponen de acuerdo en ver quién transmite. Parecido a una red local.

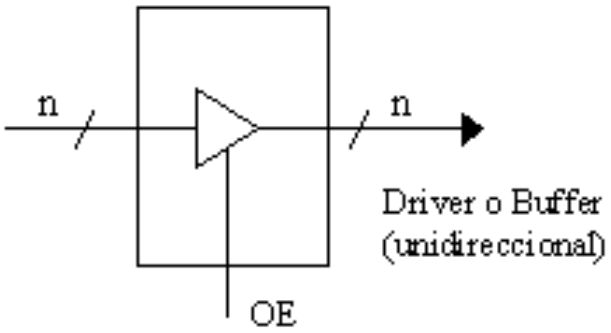
Señales de arbitraje (4)

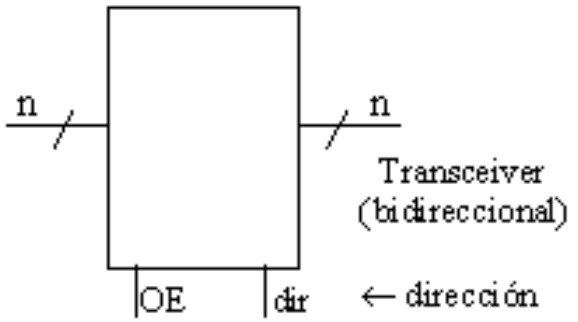
- Los nodos (dispositivos *Master*) envían un identificador al bus y posteriormente monitorizan el estado del bus. Si lo que dejaron en el bus difiere de lo posteriormente leído, han perdido la arbitración.
- Ejemplos: bus SCSI y CAN (*Controller Area Network*).
- Si se utilizan líneas de arbitración diferentes a las de datos es posible transmitir datos mientras se realiza la transmisión de datos y aprovechar mejor el ancho de banda del bus.

Circuitería usada en la interfaz de bus (1)

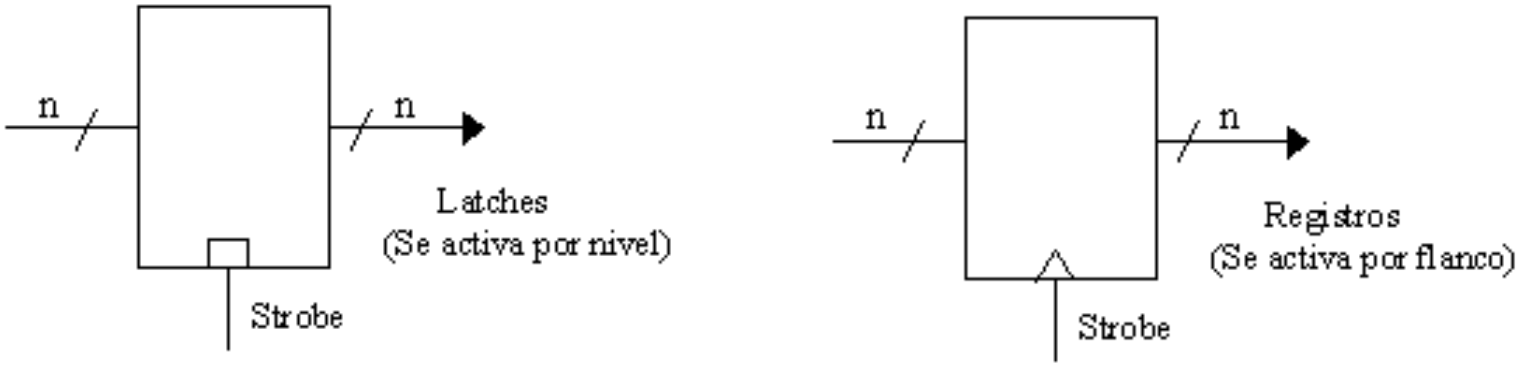
CIRCUITERÍA USADA EN LA INTERFAZ DE BUS

- Circuitería usada para conectar dispositivos (memoria ó E/S) a la CPU:
  - Dispositivos combinacionales (no conservan estado) → puertas lógicas (AND, OR, ...), inversores lógicos, multiplexores, decodificadores, ...
  - Dispositivos secuenciales (conservan estado) → *buffers*, *transceivers*, *latches*, *registros*, ...





Circuitería usada en la interfaz de bus (2)



- En estos últimos, la salida está conectada de manera que siempre está a la vista el contenido: bien lo almacenado o la nueva entrada. También suele haber un OE* (*Output Enable*).

ÍNDICE (2)

- LÍNEAS DE CONTROL
  - Señales de sincronización
  - Señales de arbitraje
  - Circuitería usada en la interfaz de bus
- EJEMPLOS DE CONEXIONES A PROCESADORES REALES:
  - La familia 80x86. El 8086.
  - La familia 68000. El 68000.
  - Circuitería adicional: el Reset y el Reloj.
  - Ejemplo de aplicación: diseño de un sistema de memoria para un 68000.
- EJEMPLO DE BUS REAL: EL BUS PCI

EJEMPLOS DE CONEXIONES A PROCESADORES REALES

LA FAMILIA 80x86

- En los primeros 80x86 había 20 líneas de dirección. En los Pentium hay 32 → puede direccionar hasta 4 GBs de memoria RAM.
- El bus de datos: 16 líneas (8086, 80186, 80286), 32 líneas (80386, 80486) y 64 líneas (Pentium).
- El bus de control: diversas líneas.
- Como ejemplo se va a describir el 8086: 20 líneas de direcciones y 16 líneas de datos (palabra de procesador de 16 bits).
- Posiciones pares en un banco e impares en otro (selección con A_0).

El procesador 8086 (1)

- Con las líneas A_{19} ... A_1 se accede a las posiciones dentro de cada banco de memoria.
- Con la señal BHE* se accede a cualquier *byte* dentro de la palabra ó a la palabra entera.
- Señales de control del 8086 (y familia 80x86):
  - RD*, WR*: Lectura o escritura.
  - M/IO*: Indica si se accede a una dirección de memoria o de entrada/salida.
  - ALE (*Address Latch Enable*). Sirve para capturar valor de la dirección en un *Latch*. El bus es multiplexado, el valor de dirección que aparezca en el bus será sustituido a continuación por el valor del dato.

El procesador 8086 (2)

- READY. Es una señal para la sincronización con otros dispositivos como, por ejemplo, la memoria. Equivalente a DTACK* del 68000.

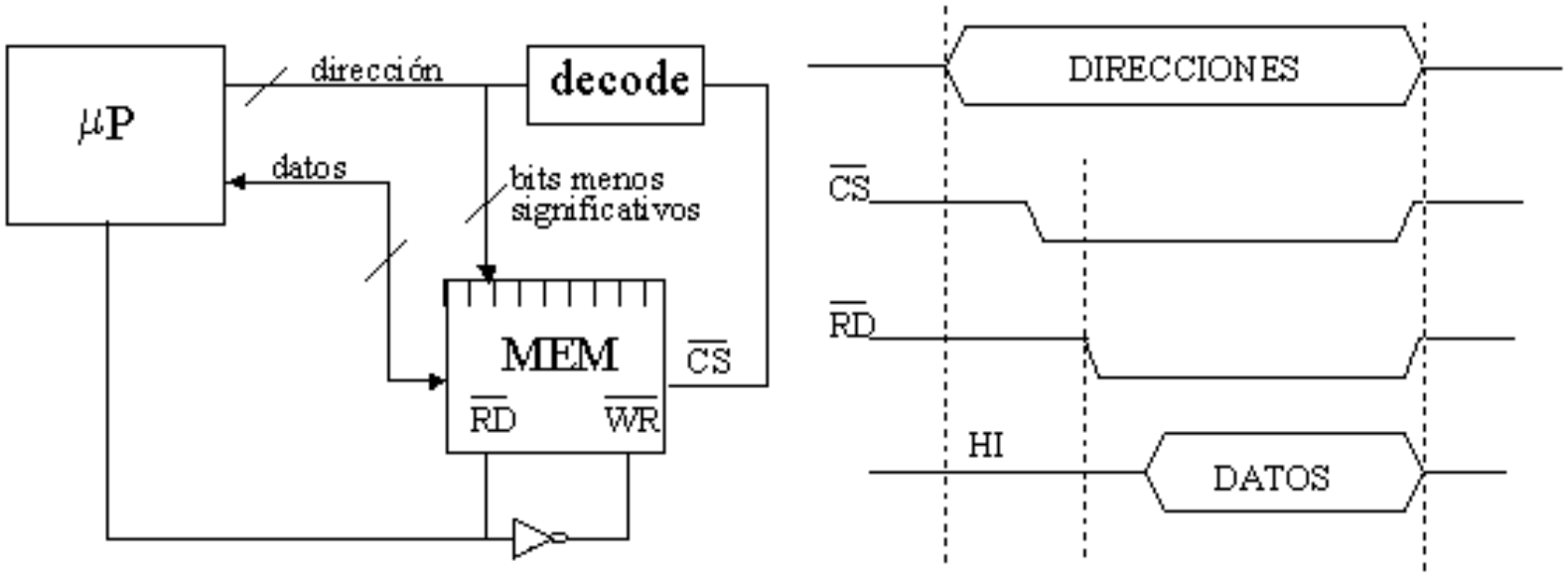
- Cada ciclo de bus tiene 4 ciclos de reloj:
  - T1: Dirección, ALE (se almacena la dirección en los *latches*), M/IO*.
  - T2: Desaparece valor de dirección; Bus en alta impedancia; Comienza RD*.
  - T3: Lectura del dato a la bajada del ciclo. Para ello tiene que estar *Ready* en nivel alto. Pueden existir estados de espera. *Ready*=0 significa no preparado.
  - T4: FIN. Se desactivan todas las señales de control y finaliza el ciclo

El procesador 8086 (3)

- En el caso de ciclo de bus para escritura hay algunas diferencias:
  - La CPU pone el dato.
  - En T2 desaparece el valor de dirección, y se sustituye por el dato. Se activa WR*.
  - La escritura se realiza en T4, a la subida de WR*. Decisión de diseño: retardar lo más posible la escritura.
- El 8086 tiene una cola de instrucciones donde se van almacenando conforme se van leyendo.

El procesador 8086 (4)

(Ejemplo conexión y acceso con procesador genérico)

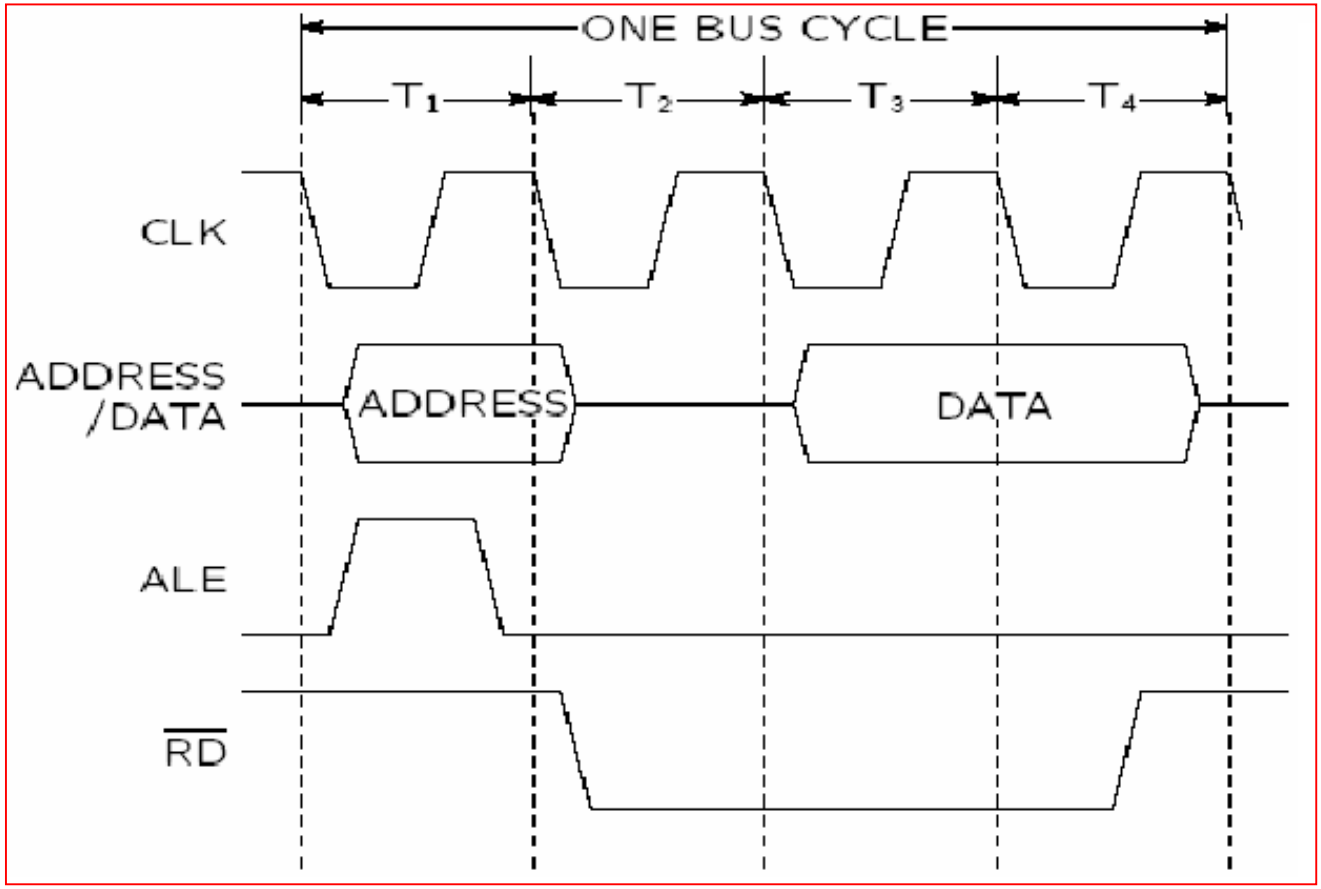


- En el esquema anterior se utilizan las líneas más significativas del bus de direcciones (bits más significativos), para activar el/los chip/s de memoria a los que se va a acceder → Señal CS* ó *Chip Select*.

El procesador 8086 (5)

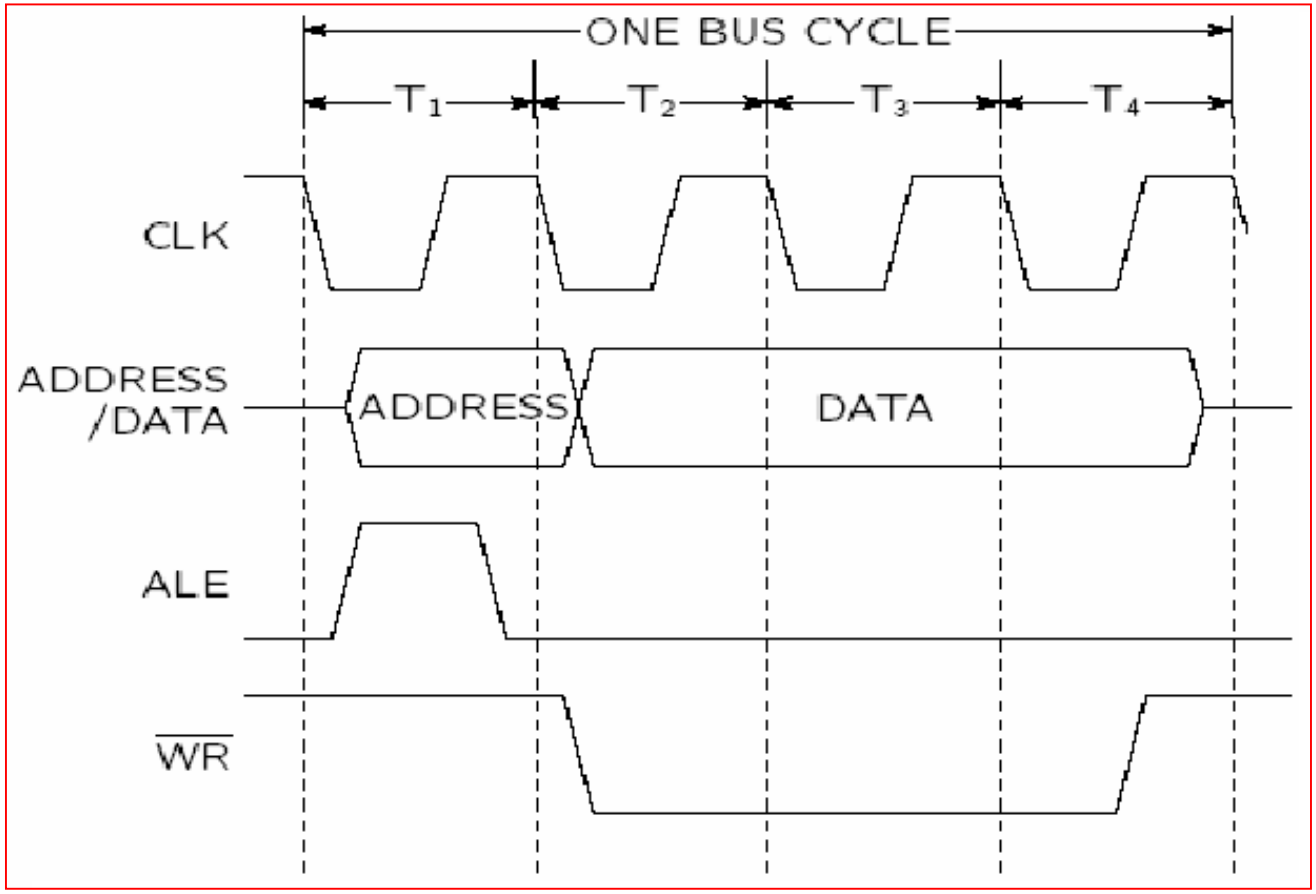
- Las líneas/bits menos significativos del bus de direcciones se usan como direcciones dentro del/ de los chip/s seleccionado/s.
- Si no hay acceso a datos, el bus de datos está en alta impedancia (HI).
- La señal RD* puede usarse en muchas memorias como indicador de *Output Enable*. Aunque puede parecer redundante (bastaría con WR*=0 para escritura y WR*=1 para lectura) así se permite que en lectura no se pongan los datos hasta bien avanzado el ciclo de bus.

El procesador 8086 (6)



Ciclo de lectura de un 8088/8086 simplificado

El procesador 8086 (7)



Ciclo de escritura de un 8088/8086 simplificado

EJEMPLOS DE CONEXIONES A PROCESADORES REALES

LA FAMILIA 68000

- El *Motorola* 68000 (y familia) fue utilizado en los primeros ordenadores *Macintosh*.
- Encapsulado en chips de 64 pines. Incluye 23 líneas de direcciones (A_1 – A_{23}) y 16 líneas de datos (D_0 – D_{15}).
- La memoria se divide en dos bancos de un *byte* cada uno. Con las señales UDS* y LDS* se puede seleccionar la parte alta y baja de una palabra de 16 bits respectivamente → como máximo se pueden direccionar 2^{24} bytes = 16MB.

El procesador 68000 (1)

- Señales de control:
  - AS* (*Addres Strobe*): indica que la dirección está preparada ó es válida.
  - R/W*: indica que el acceso es de lectura (1) o escritura (0).
  - DTACK*: indica al 68000 que la información está disponible (en D_0 – D_{15}) en el caso de ciclo bus para lectura ó que han sido tomados del bus de datos (en D_0 – D_{15}) en el caso de ciclo bus para escritura. Permite comunicación asíncrona. Señal similar a *Ready* en 8086.
  - Reset*: inicialización de controladores conectados al 68000.

El procesador 68000 (2)

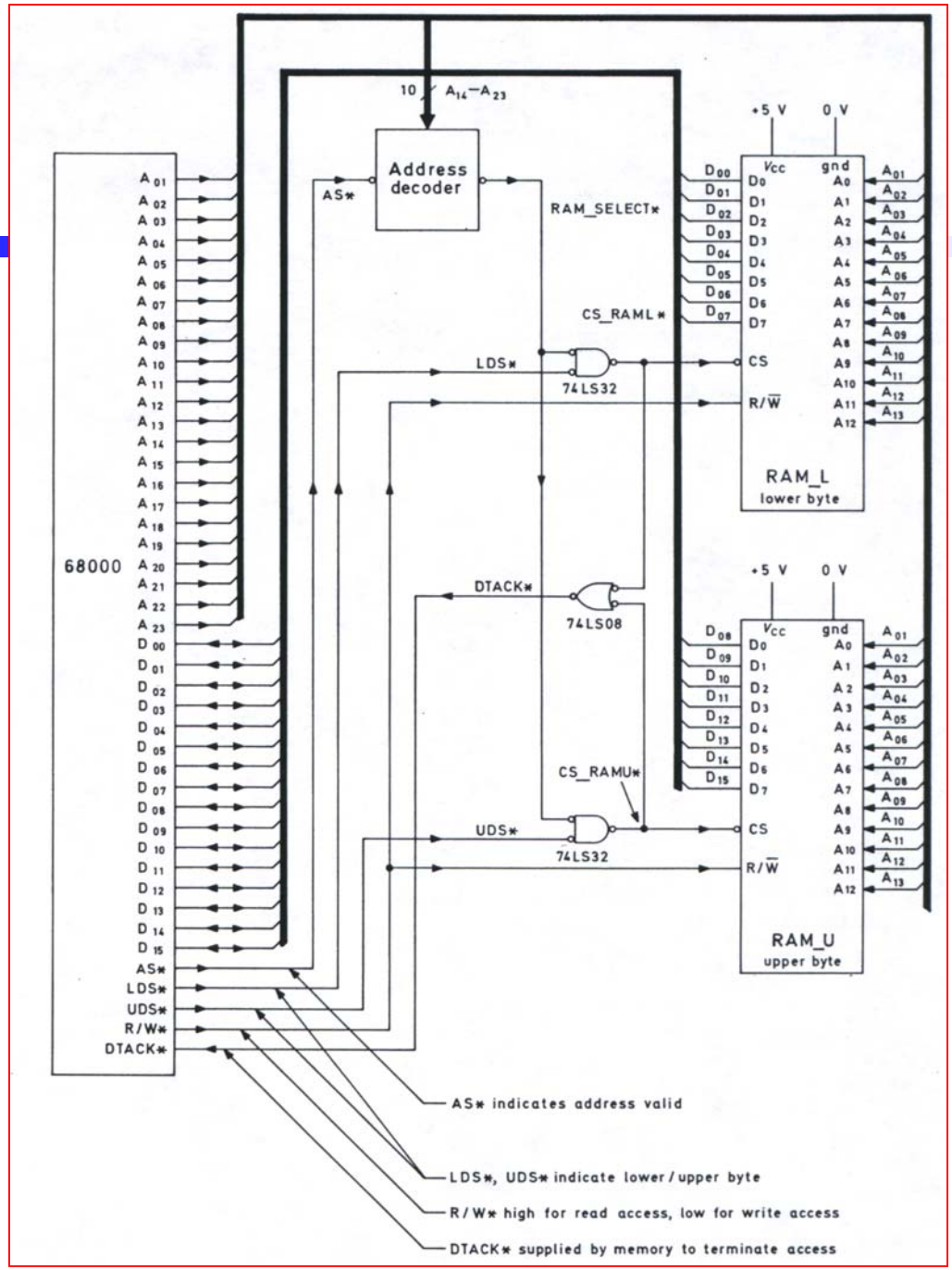
- Halt*: detección de errores al tratar errores. El procesador se queda parado y lo indica con la señal *Halt*.
- BERR* (*Bus Error*): se le indica a la CPU (el 68000) que aborte el ciclo de bus actual. Activada entre AS* y DTACK*.

- Señales de control del bus:
  - BR* (*Bus Request*): petición de uso del bus por otro dispositivo. Por ejemplo un controlador de DMA.
  - BG* (*Bus Grant*): concesión del uso del bus por parte del 68000.
  - BGACK* (*Bus Grant Ackowledge*): confirmación de recepción de señal BG*.

El procesador 68000 (3)

- Señales de Control de Interrupciones: usadas para trabajar en modo asíncrono con dispositivos de E/S e indicar estados internos 68000.
  - IPL0*, IPL1*, IPL2* (*Interrupt Level*) : sirven para indicar prioridades de controladores de E/S que solicitan interrupción (realización de una operación de E/S de datos).
  - FC0*, FC1*, FC2*: todas a 1 indican reconocimiento (*Acknowledge*) de la petición de interrupción para un controlador de E/S. También sirven para indicar el estado interno del 68000.

Ejemplo de comunicación simple entre el 68000 y la memoria



El procesador 68000 (4)

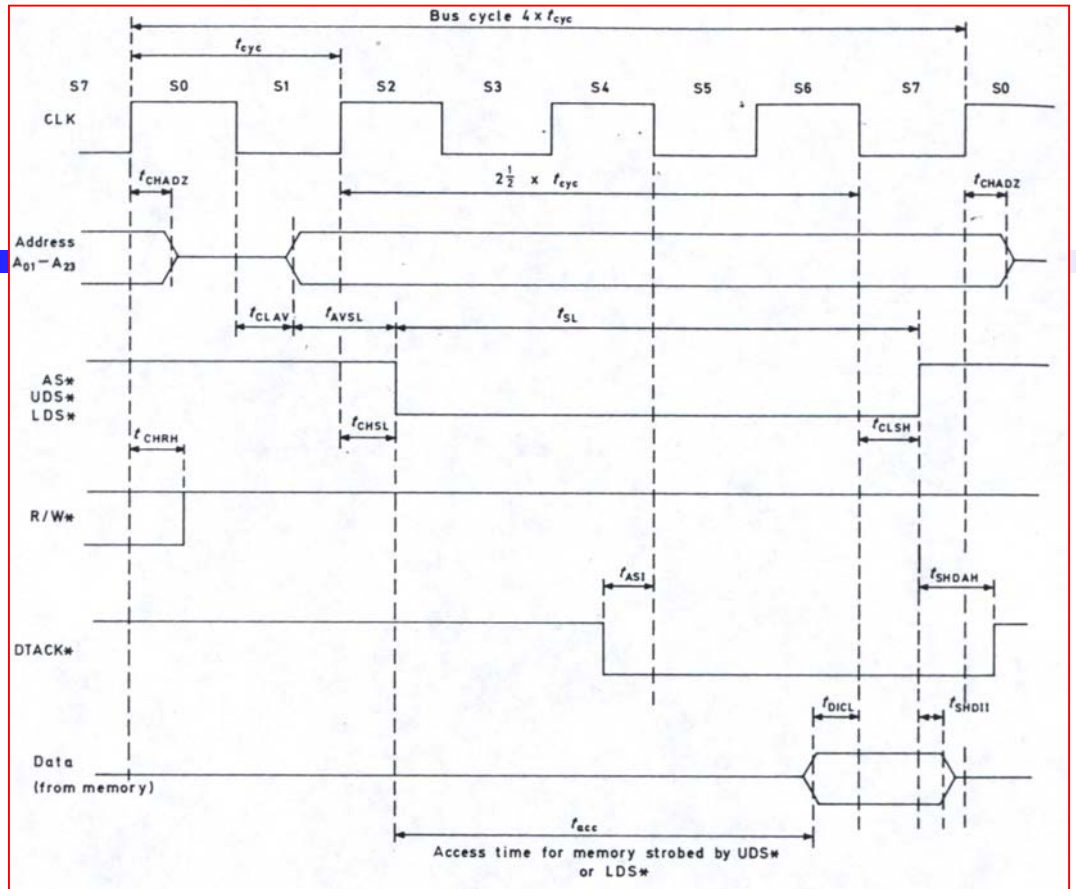
- El ciclo de bus tarda 4 ciclos de reloj divididos en 8 semiciclos (S_0 ... S_7):
  - S_0: activación de R/W. Dirección en alta impedancia.
  - S_1: colocación de la dirección válida.
  - S_2: Se activan AS*, UDS*, LDS*.
  - S_4: si se activa DTACK*, para adaptar el proceso a memorias más lentas, indica a la CPU que todo va bien. Si no se activa, quiere decir que hay que esperar a la memoria repitiendo S4.

El procesador 68000 (5)

- S_6: A la bajada de este semiciclo, el procesador leerá lo que haya en el bus de datos. Los espacios de tiempo intermedios (S3, S5) en los que no se hace nada, son necesarios para respetar los tiempos que las memorias invierten en los accesos y para asegurar la estabilidad de la señal en el momento de la lectura.

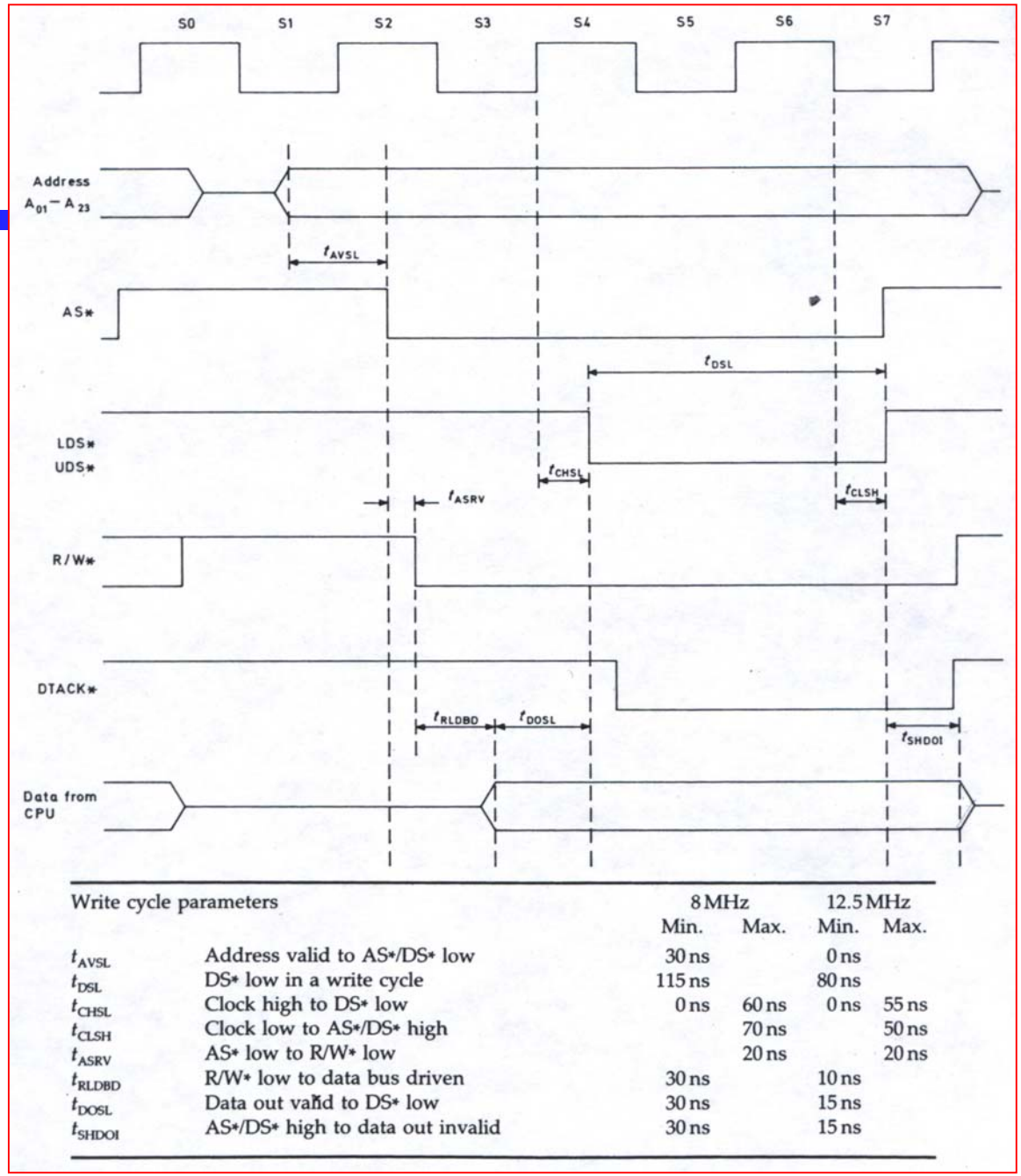
- Las señales al pasar de un estado a otro pasan por una zona de *metaestabilidad* → Respetar unos tiempos de transición y mantenimiento de las señales.

Ciclo de lectura del 68000



Read cycle parameters		8 Mhz		12.5 MHz	
		Min.	Max.	Min.	Max.
t_{CHADZ}	Clock high to address float	0 ns	80 ns	0 ns	60 ns
t_{CLAV}	Clock low to address valid		70 ns		55 ns
t_{AVSL}	Address valid to AS*, DS* asserted	30 ns		0 ns	
t_{SL}	AS*, DS* asserted	240 ns		160 ns	
t_{CHSL}	Clock high to AS*, DS* asserted	0 ns	60 ns	0 ns	55 ns
t_{CLSH}	Clock low to AS*,DS* negated		70 ns		50 ns
t_{SHDAH}	AS*, DS* negated to DTACK* negated	0 ns	245 ns	0 ns	150 ns
t_{CHRH}	Clock high to R/W* high	0 ns	70 ns	0 ns	60 ns
t_{DICL}	Data in to clock low (data setup)	15 ns		10 ns	
t_{SHDII}	DS* negated to data invalid (data hold)	0 ns		0 ns	
t_{ASI}	DTACK* low to clock low setup time	20 ns		20 ns	

Ciclo de escritura del 68000



Write cycle parameters		8 MHz		12.5 MHz	
		Min.	Max.	Min.	Max.
t_{AVSL}	Address valid to AS*/DS* low	30 ns		0 ns	
t_{DSL}	DS* low in a write cycle	115 ns		80 ns	
t_{CHSL}	Clock high to DS* low	0 ns	60 ns	0 ns	55 ns
t_{CLSH}	Clock low to AS*/DS* high		70 ns		50 ns
t_{ASRV}	AS* low to R/W* low		20 ns		20 ns
t_{RLDBD}	R/W* low to data bus driven	30 ns		10 ns	
t_{DOSL}	Data out valid to DS* low	30 ns		15 ns	
t_{SHDOI}	AS*/DS* high to data out invalid	30 ns		15 ns	

El procesador 68000 (6)

- Tiempo de *Setup*: en el caso de acceso por lectura, necesidad de que el dato o la señal esté un tiempo antes (t_{DICL} en el 68000). En el caso de acceso por escritura es el tiempo que debe permanecer el dato en el bus hasta desactivar señal CS (t_{DOSL} + t_{DSL} en el 68000).
- Tiempo de *Hold*: necesidad de que el dato o la señal permanezca estable un tiempo una vez desactivado el CS (*chip select*). Durante S_7 en el 68000 estos tiempos corresponden: t_{SHDII} en lectura y t_{SHDOI} en escritura.
- El tiempo de acceso (t_{acc}) en lectura se puede considerar como el tiempo hasta obtener el dato, desde activación de CS o bien desde que la dirección es válida.

EJEMPLOS DE CONEXIONES A PROCESADORES REALES

CIRCUITERÍA ADICIONAL: EL *RESET* Y EL RELOJ

- El *Reset*: señal común a todos los procesadores que los lleva a un estado inicial.
- Dos tipos: el del usuario (accionado con un botón) y el de arranque (accionado al encenderse el sistema).
- Cada vez que se acciona se salta a una dirección de memoria fija. En esas posiciones de memoria hay que colocar memoria no volátil. En el 8086 la dirección es la FFFF0h y en el 68000 la 0h.

Circuitería adicional: el *Reset* y el Reloj

- Las líneas de dirección y memoria permanecen en alta impedancia, y las líneas de control inactivas.
- El reloj: La onda cuadrada que produce el cristal hay que adaptarla a nuestro sistema, que exigirá una frecuencia máxima, frecuencia mínima, un tiempo máximo de subida, un *duty cycle* (proporción entre el tiempo que el ciclo está en alta y el periodo), sincronización con otras señales, etc
  - Un ciclo de reloj incluye dos estados periódicos: uno con valor 0 y otro con valor 1.

Ejemplo de aplicación: diseño de un sistema de memoria

DISEÑO DE UN SISTEMA DE MEMORIA PARA UN 68000

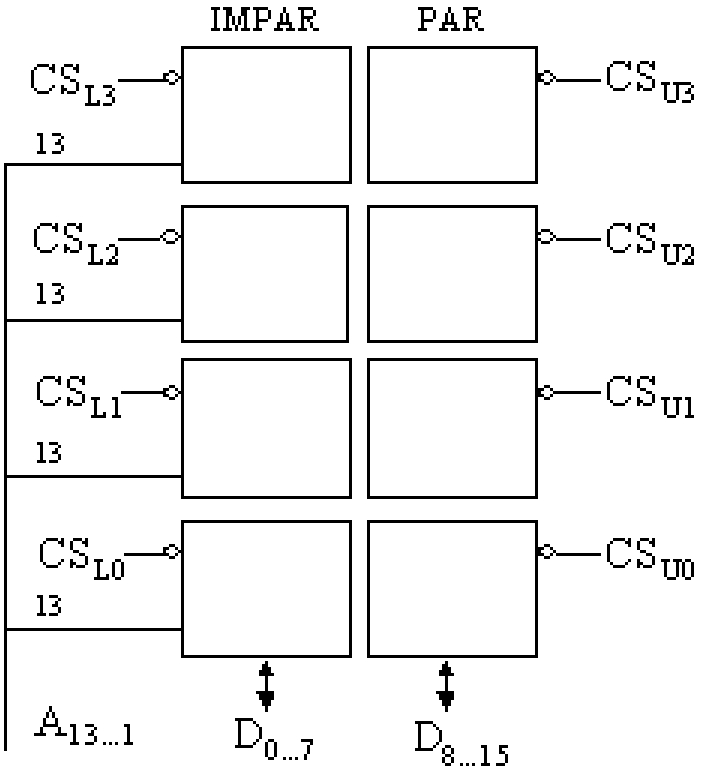
- Memoria de 64Kbytes de EPROM con chips de 8Kx8.

1º) Circuitería de Decodificación

- Las 23 líneas de direcciones del 68000 (A_1 ... A_{23}) se dividen de la siguiente manera:
  - A_1 ... A_{13} : direccionamiento interno de cada chip de 8Kx8 (véase que $2^{13} = 2^3.2^{10} = 8K$)
  - A_{14} y A_{15} : selección de los cuatro niveles de bancos (en total 8 chips de 8Kx8).
  - A_{16} ... A_{23} : selección de la zona de memoria donde están los 64K.

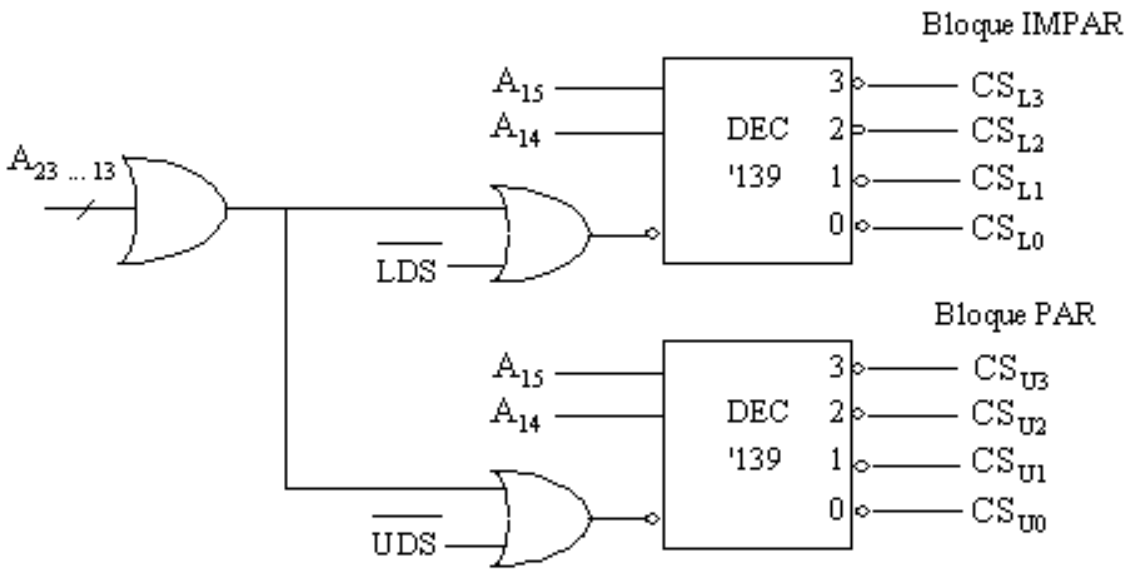
Diseño de un sistema de memoria para un 68000 (1)

- Se sitúan los 64Ks en la parte baja de memoria porque allí empieza a direccionar el 68000 en caso de *reset*. Esto quiere decir que A_{16} ... A_{23} = 00h.
- Con las señales LDS* y UDS* se selecciona bloque par o impar.
- La decodificación de las direcciones es una decodificación completa.



Los subíndices de los CS significan
U ≡ Upper y L ≡ Lower
(El 68000 sigue el modo Big Endian)

Diseño de un sistema de memoria para un 68000 (2)



Circuito de decodificación de direcciones del anterior esquema.

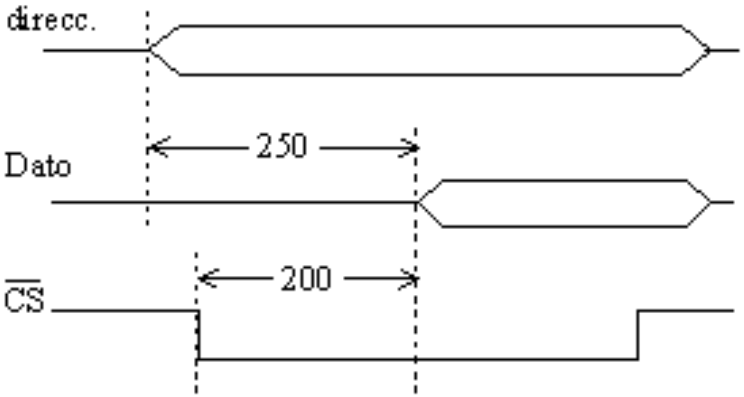
- Solución con <u>decodificación parcial o incompleta</u>: si se tuvieran por ejemplo otros 64Ks de memoria RAM, bastaría con permitir mediante la línea A_{23} cualquier dirección a la mitad baja para la EPROM ($A_{23} = 0$) y cualquier dirección a la mitad alta para la RAM ($A_{23} = 1$).

Diseño de un sistema de memoria para un 68000 (3)

2º) Cálculo Tiempos Acceso a Memoria y Estados de Espera

- Si el 68000 funciona a una frecuencia de 12,5 MHz (T = 80 ns), averiguar los tiempos de espera necesarios suponiendo los siguientes tiempos de lectura para las memorias EPROM de 8Ks:
  - A) Necesidad de dirección estable de 250 ns.
  - B) Tiempo desde dirección válida hasta dato válido de 250 ns.
  - C) Tiempo de acceso desde CS*: 200 ns.

Diseño de un sistema de memoria para un 68000 (4)



Esquema cronograma de lectura de un chip de memoria EPROM de 8Ks

- Hay que comprobar atendiendo a los tiempos de un ciclo de lectura de un 68000, que los tiempos mínimos correspondientes al acceso para lectura a la memoria EPROM, se van a respetar. En caso contrario → estados de espera.
- Tres casos a estudiar: a) duración mínima de dirección estable (250 ns), b) tiempo mínimo de dirección estable (250 ns) y c) tiempo mínimo de decodificación de las direcciones para CS* (200 ns).

Diseño de un sistema de memoria para un 68000 (5)

- A) La duración mínima de dirección estable:

$$TDmin. = 3.5\ T - T_{CLAV} + T_{CHADZ}.$$

Tomando los casos más desfavorables (valor máximo T_{CLAV} y mínimo T_{CHADZ}),

$$TDmin. = 3.5*80 – 55 + 0 = 225\ ns.$$

No llega a los 250 ns requeridos → Introducir un estado (ciclo) de espera, ya que 225ns + 80 ns >250 ns

- B) El tiempo mínimo desde dirección estable hasta dato válido:

$$TDDmin. = 3\ T - T_{CLAV} - T_{DICL}$$

Y el caso más desfavorable (valores $T_{CLAV} - T_{DICL}$ máximos),

$$TDDmin. = 3\ *80 - 55 - 10 = 175\ ns$$

Diseño de un sistema de memoria para un 68000 (6)

No llega a los 250 ns requeridos → Introducir un estado (ciclo) de espera, ya que 175ns + 80 ns = 255 ns >250 ns

- C) En el tiempo mínimo de decodificación de las direcciones para CS* hay que tener en cuenta, que viendo el esquema de decodificación de direcciones y el cronograma de ciclo de lectura de un 68000, la señal más tardía (desfavorable) es la AS*/UDS*/LDS*.

$$TDec. = 2.5\ T - T_{DICL} - T_{CHSL}$$

Tomando los casos más desfavorables (valores máximos de T_{DICL} y T_{CHSL}),

$$TDDmin. = 2.5 * 80 - 55 - 10 = 135\ ns$$

Para hacer que supere los 200 ns necesarios, hace falta un ciclo/estado de espera: 80 + 135 = 215 > 200 ns

Diseño de un sistema de memoria para un 68000 (7)

- En el anterior caso no se han tenido en cuenta los retrasos derivados de la circuitería (puertas lógicas y decodificadores). Si se supusiese un retraso de 16 ns (retraso máximo típico de circuitos LS) y teniendo en cuenta los dos elementos utilizados en la decodificación de direcciones (puerta OR + decodificador):

$$TDec. = 2.5\ T - T_{DICL} - T_{CHSL} - 2*16 = 103\ ns$$

Esto supondría dos tiempos/ciclos de espera, ya que:

$$2*80 + 103 = 263 > 200\ ns$$

- En este ejemplo se va a despreciar el retraso por circuitería.

Diseño de un sistema de memoria para un 68000 (8)

- Ahora supongamos los siguientes datos de la EPROM para escritura:
  - A) Pulso mínimo de escritura: 50 ns
  - B) Tiempo de *Setup-Hold*: 20 ns
- En base al cronograma que muestra el ciclo de escritura de un 68000, ¿es necesario algún estado/ciclo de espera?
  - A) Tiempo de escritura T_{DSL}= 80 ns > 50 ns y por lo tanto al 68000 le sobran 30 ns.
  - B1) El *Setup* es $T_{DSL} + T_{DOSL}$ = 80 + 15 = 95 ns > 20 ns y también está sobrado de tiempo.

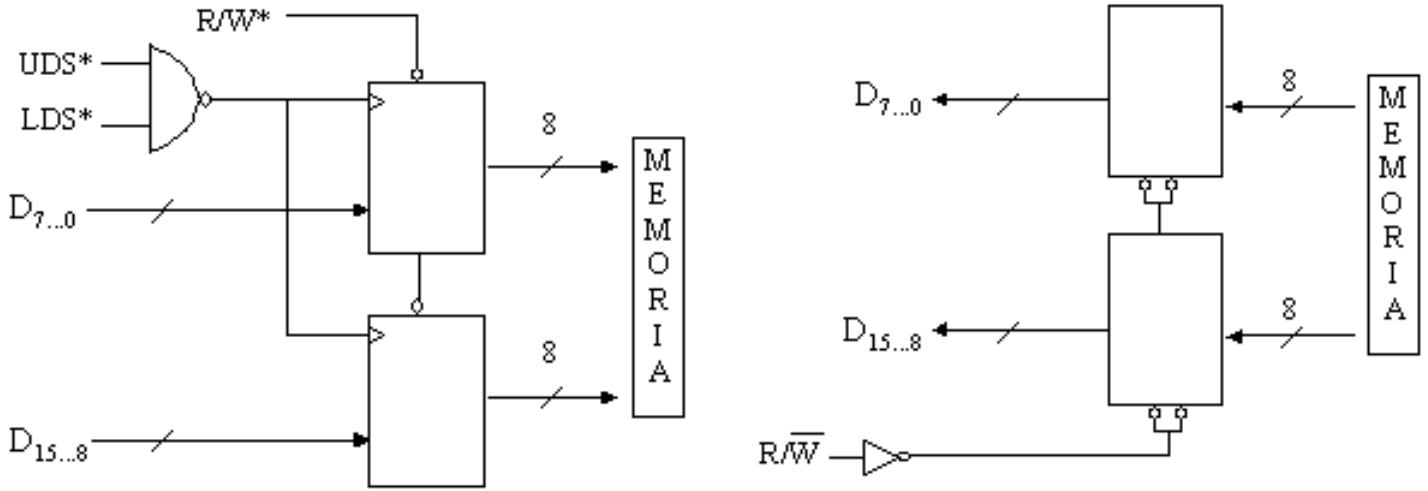
Diseño de un sistema de memoria para un 68000 (9)

- B2) Por su parte el *Hold* de datos (tiempo que se debe esperar y mantener el dato después de desactivar AS* ó LDS*/UDS* antes de hacer un siguiente acceso) es también 20 ns. Sin embargo, observando los tiempos en el ciclo de escritura del 68000, ese tiempo es T_{SHDOI} = 15 ns y por lo tanto no es suficiente.

- Esto último obliga a guardar en un *latch* el dato para que cuando se desactive AS* ó LDS*/UDS*, el dato siga a la entrada de la memoria EPROM.

  Insertar estados de espera no soluciona el problema.

Diseño de un sistema de memoria para un 68000 (10)



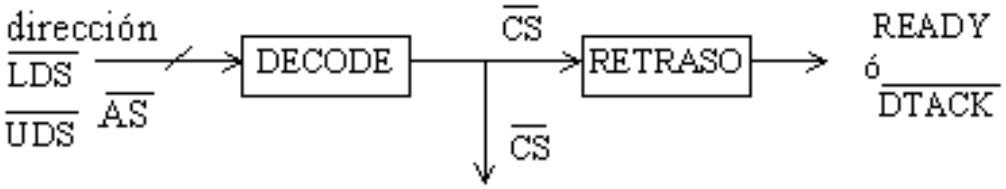
- La inserción de nueva circuitería puede afectar al cálculo anterior de tiempos.

Diseño de un sistema de memoria para un 68000 (11)

3º) Circuitería de Generación de Estados de Espera

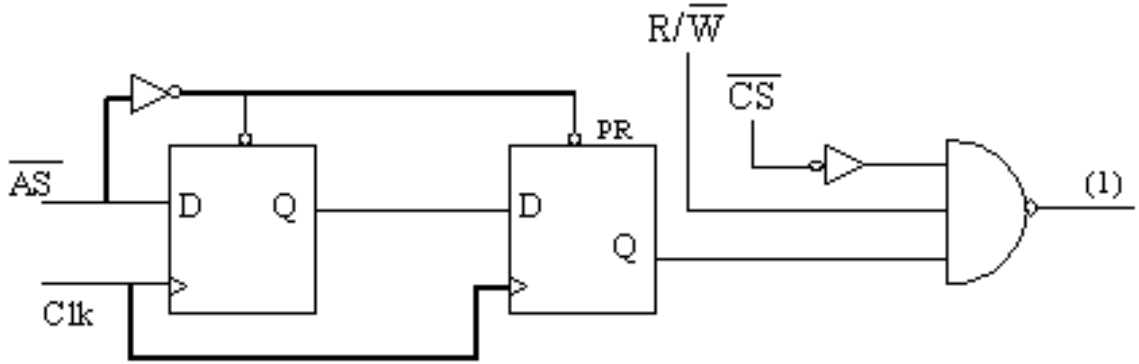
- Para generar el estado de espera se utilizará la señal AS*. Se pueden utilizar cualquiera de las señales del 68000 involucradas en el acceso a memoria: CS*, UDS*, ...
- El objetivo es retrasar la recepción de la señal DTACK* en el 68000, una vez se ha accedido a los datos de la memoria EPROM, para que mantenga por más tiempo activadas las direcciones (A_0 – A_{23}) y el resto de señales de control (AS*, UDS*, ...).

Diseño de un sistema de memoria para un 68000 (12)



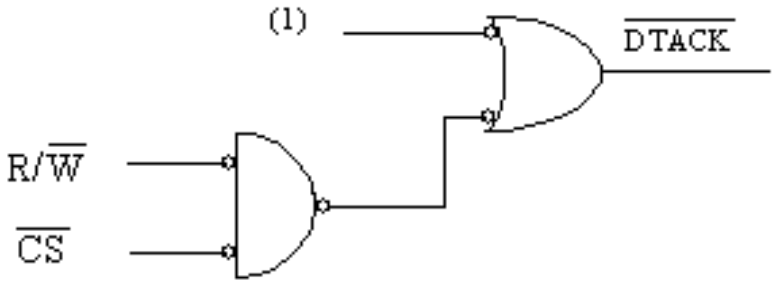
Esquema general del circuito de inserción de estados de espera

- Para implementar el circuito de retraso para accesos de lectura se pueden emplear baterías de *Flip-Flops*:



Diseño de un sistema de memoria para un 68000 (13)

- En escritura no se necesitan estados de espera. El circuito anterior se completaría con:



- La salida del anterior circuito queda retrasada según se muestra en la figura siguiente: