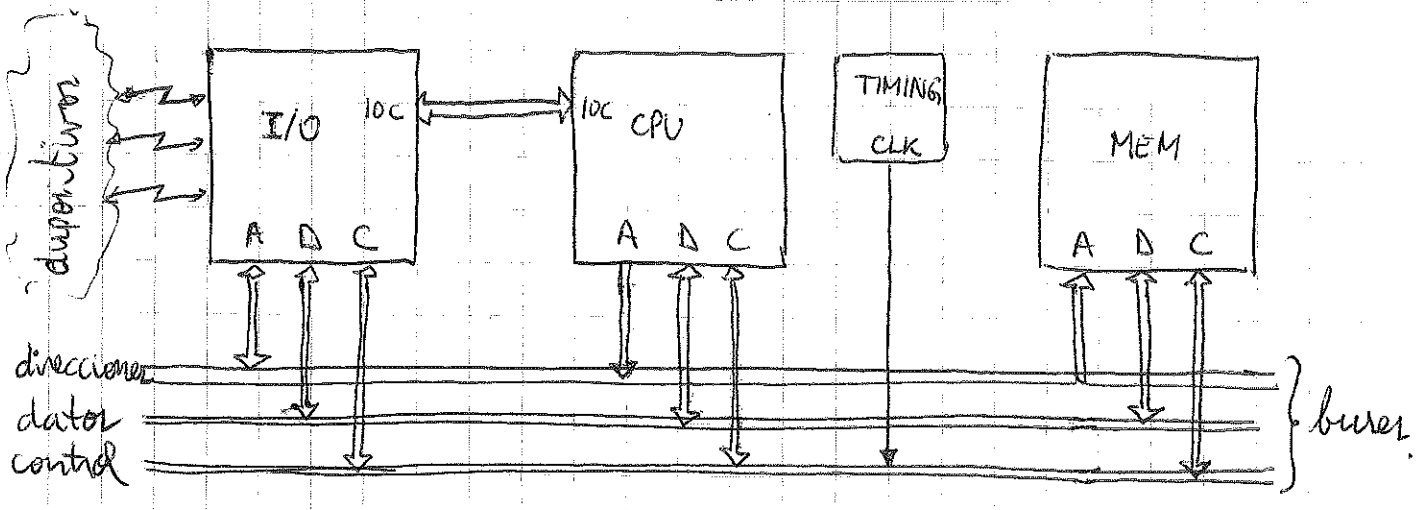
Mem 1

Arquitectura Física de M32



+ interfaz con los dispositivos.

+ realiza todos los cálculos
+ contiene los registros

+ contiene la memoria.

Toda la comunicación entre estos módulos se realiza a través de los buses. Hay 3 buses:

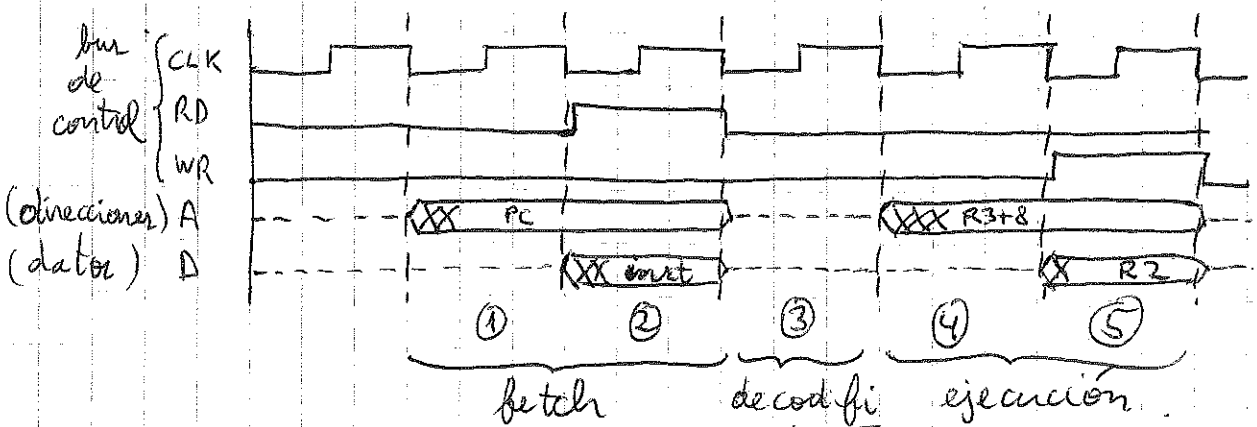
+ bus de datos: D31 - DØ
+ bus de direcciones: A31 - A2, BE3 - BEØ
+ bus de control: indica el tipo de operación a realizar (lectura o escritura):
  - RD: lectura
  - WR: escritura
  - CLK: el reloj.
  - WAIT: sirve para prolongar el acceso a la memoria

Mem 2

La CPU o procesador, es el intérprete del lenguaje de máquina. Todo el proceso de una instrucción se realiza internamente en la CPU. En la CPU están los registros (R0 ... R31), el contador de programa (PC) y registro de estado (SR).

Los programas y los datos se almacenan en la memoria.

Examinemos en un diagrama de tiempo la ejecución de la instrucción `ldw R2, [R3+8]`. Esta instrucción se codifica en una palabra de 32 bits en memoria...



Para ejecutar una instrucción, el procesador para por 3 fases:

Fetch: En esta fase se carga la instrucción que está en la memoria en un registro interno de la CPU.

Mem 3

La fase Fetch se ejecuta en 2 ciclos del reloj.

① La CPU coloca el PC (contador de programa) en el bus de direcciones.

② La CPU activa RD y mantiene el PC en el bus de direcciones.
La memoria responde a la señal RD colocando el contenido de la dirección especificada.

Decodificación: En esta fase el procesador examina la instrucción y decide como ejecutarla. Para esto sólo se necesita el ciclo ③.

Ejecución: En esta fase se ejecuta la instrucción. Esta fase depende del código de operación de la instrucción, cuyo contenido fue examinado en ③. Como la instrucción es un store, la CPU la ejecuta en 2 ciclos:

④ La CPU calcula la dirección como R3+8 y la coloca en el bus de direcciones.

⑤ La CPU mantiene la dirección en el bus y activa la señal WR. Al mismo tiempo coloca R2 en el bus de datos. La memoria modifica el contenido de la dirección especificada con el dato que aparece en el bus.

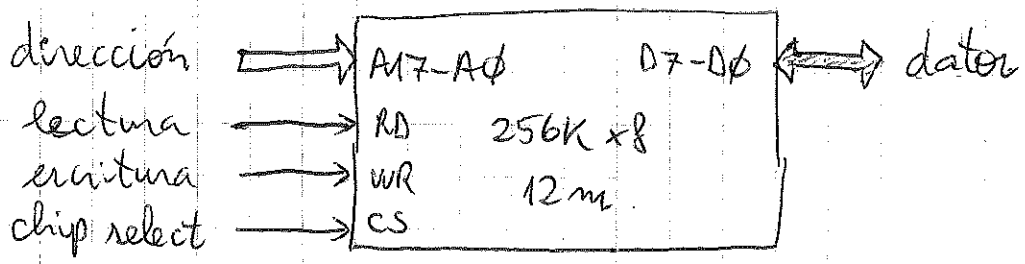
Mem 4

La Memoria

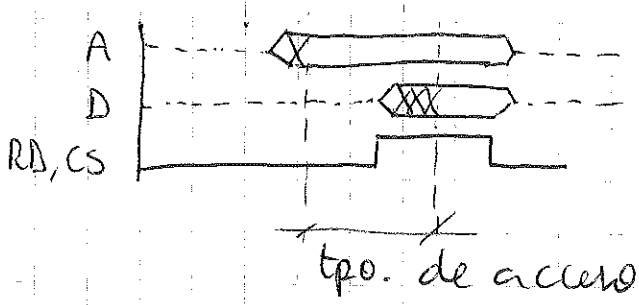
Existen varios tipos de memoria: SRAM, DRAM, EDO RAM, ROM, EPROM, Flash y otras.

(a) Static Random Access Memory: memoria estática de acceso aleatorio. Son rápidas, se borran si se apagan, necesitan 4 o 6 transistores por bit.

Se etiquetan como nº de palabras x ancho de palabra y tiempo de acceso.



Comportamiento en un diagrama de tiempo:

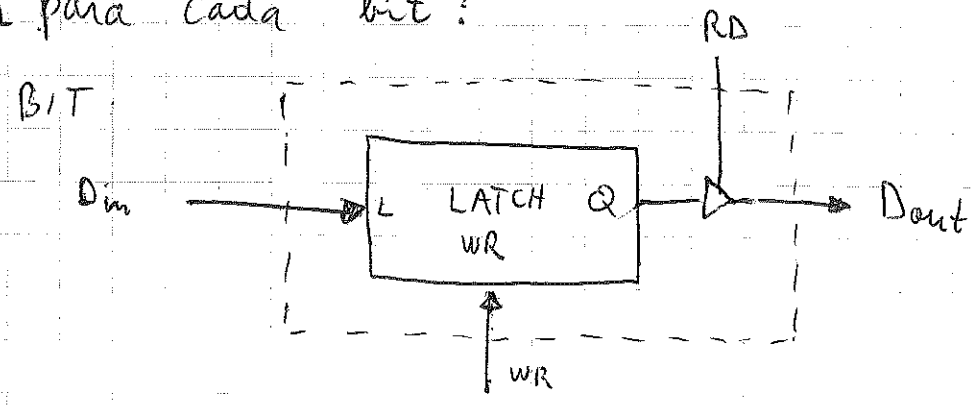


El tiempo de acceso es el tiempo transcurrido desde que se coloca una dirección en A hasta que la memoria entrega el dato.

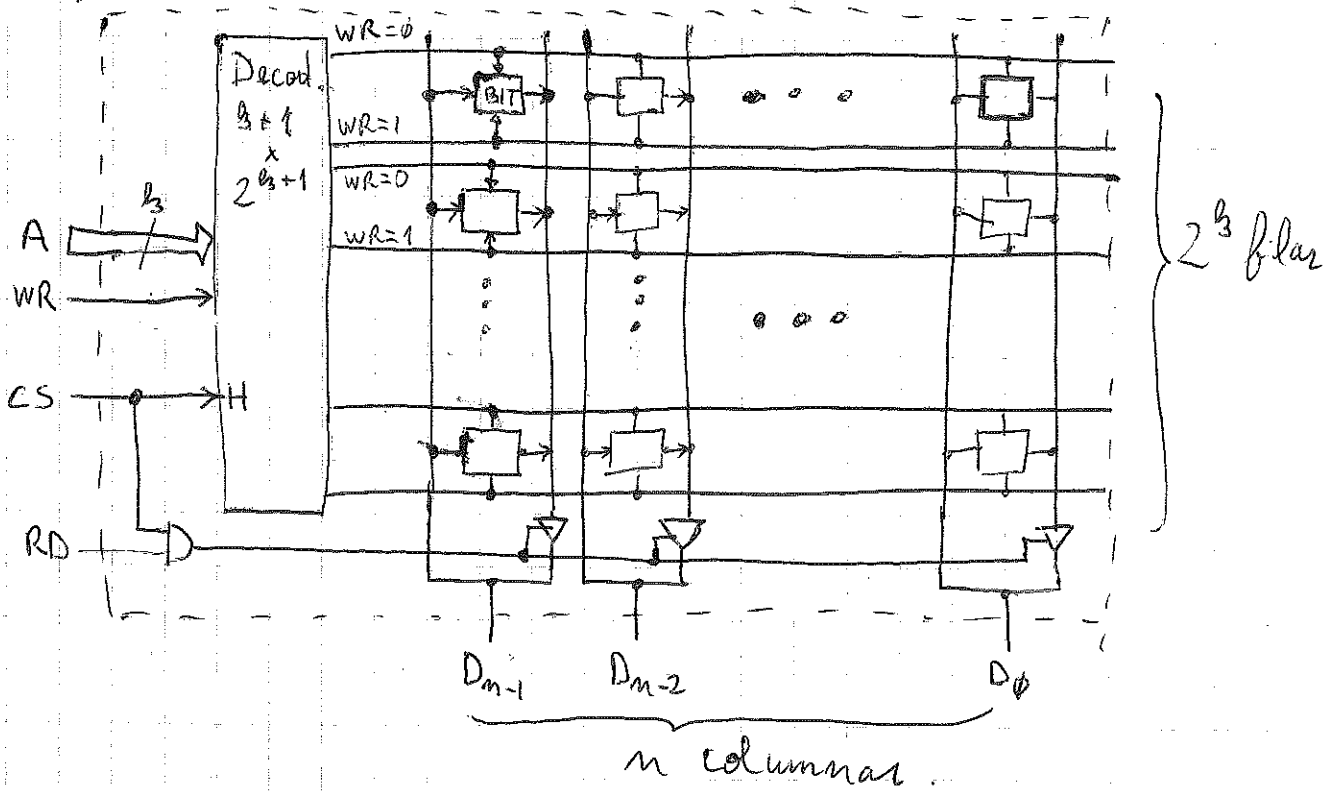
Mem 5

Implementación de un chip de memoria estática

Este tipo de chip se puede implementar usando latches para cada bit:



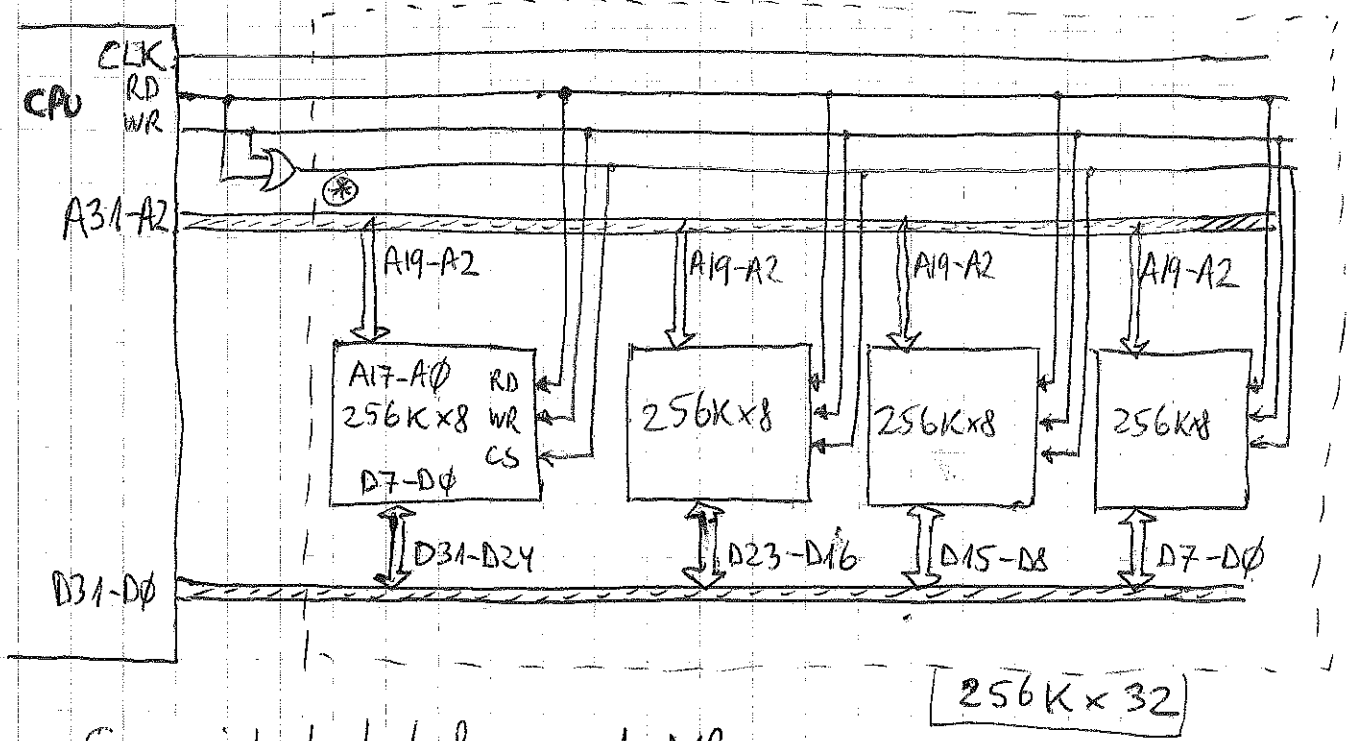
Se usa un decodificador para seleccionar la palabra que se va a escribir o leer.



Para conectar este chip a la CPU, hay que resolver varios problemas:

Mem 6

(i) Tamaño de palabra del bus de datos (32) y el ancho de la memoria (8) no coinciden.
Por lo tanto hay que colocar varios chips en paralelo: (todos los chips se activan al mismo tiempo).



Capacidad total: 1 MB.

Problema: las direcciones Ø, 1MB, 2MB, ... son sinónimos. Si se escribe x en la dirección Ø, también se leerá x en las direcciones 1MB, 2MB, 3MB, ...

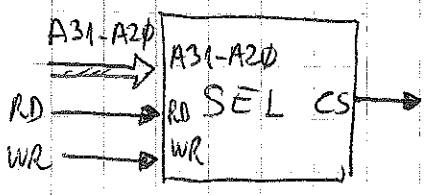
(ii) La memoria solo debe accesarse en el rango de direcciones [Ø, 1MB[. Pero:

$$a \in [Ø, \underbrace{1MB}_{2^{20}}[\iff a = ØØ\ldots Ø\,??\ldots?$$

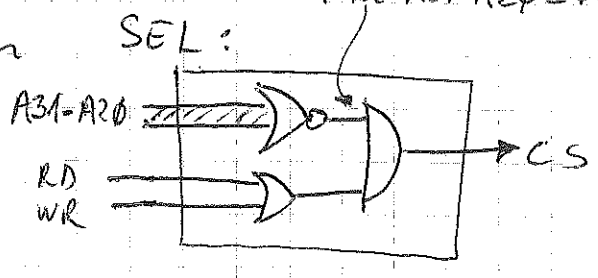
A31 A30 A29 A19 ... AØ

Mem 7

Para limitar el rango de acceso, usamos la línea CS (chip select) de la memoria. Un chip se accesa solo si CS está en 1. Si éste está en ϕ el chip no participa en un acceso. CS se calcula como

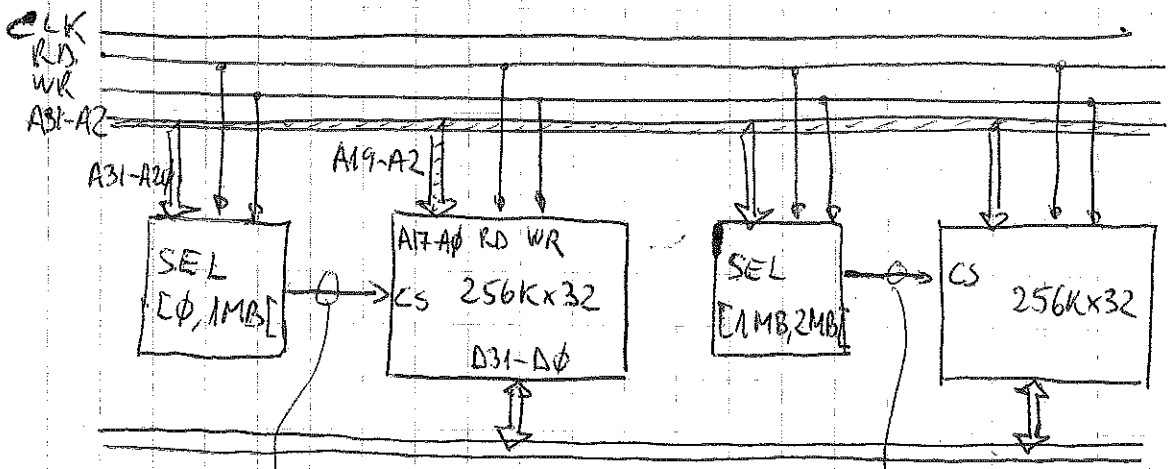


con SEL:



Ahora, si se lee la dirección 1 MB se lee basura, puesto que CS permanece en ϕ.
$\Rightarrow$ la memoria no entrega nada en el bus de datos.

(iii) La memoria puede ser insuficiente. Se coloca un módulo de 1 MB en las direcciones $[\phi, 1MB[$ y otro módulo en $[1MB, 2MB[$.



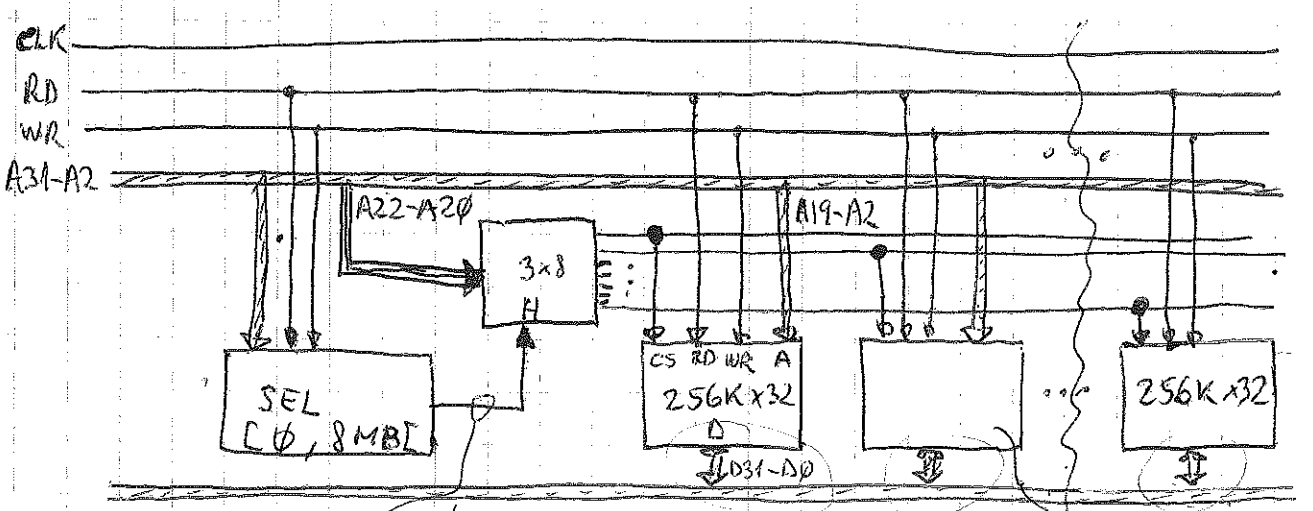
1 si (RD+WR) y $A \in [\phi, 1MB[$
i.e. $A = \phi\phi \ldots \phi\phi$?? ... ?
$\in$ A2φ

1 si (RD+WR) y $A \in [1, 2MB[$
i.e. $A = \phi\phi \ldots \phi 1$?? ... ?
A2φ

Observe que estos módulos nunca se activan al mismo tiempo.

Ejercicio: Interfaz de memoria de 8 MB ubicada en $[\phi, 8MB[$.

Sol: Se colocan 8 bancos de 1 MB y se activan por medio de un decodificador.



$1 \leftrightarrow (RD+WR) \cdot a \in [\phi, 8MB[$.

i.e. $a = \phi\phi\ldots\phi\ \underline{b\,b\,b}\ \underline{?\,?\ldots?}$

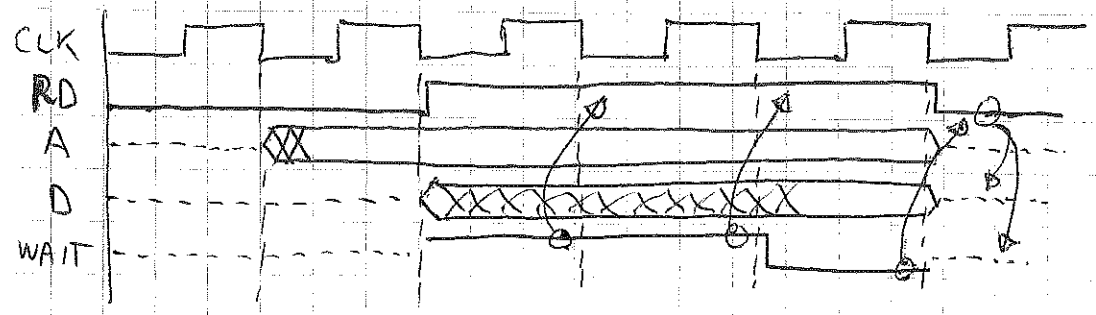
nº del banco $\in [\phi, 7]$ palabra dentro del banco

(iw) La memoria puede tener un tiempo de acceso muy extenso $\Rightarrow$ hay que agregar por ejemplo 2 ciclos de espera (wait states).

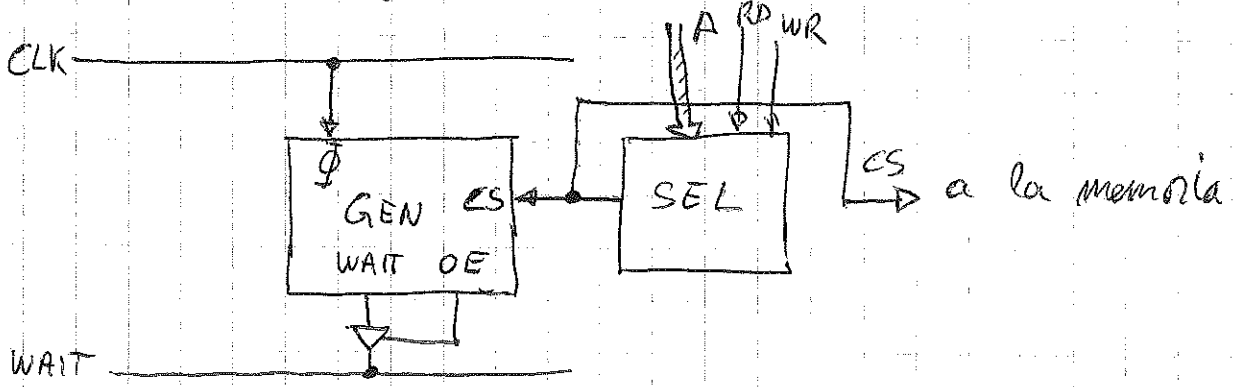
Cuando la CPU está en el 2do ciclo de una lectura o escritura y detecta que la línea WAIT está en 1 en el pulso de bajada del reloj, repite el 2do ciclo hasta que

Mem 9

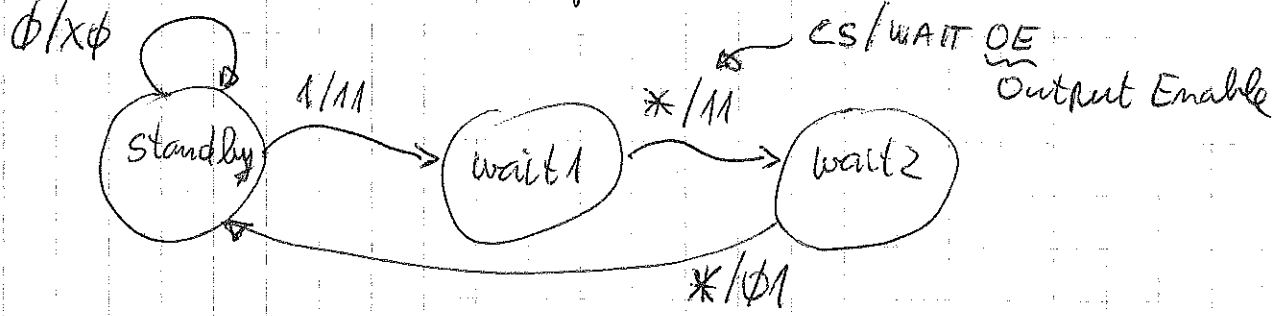
WAIT se ponga en ϕ.



Para activar la línea WAIT se coloca un circuito generador de ciclos de espera



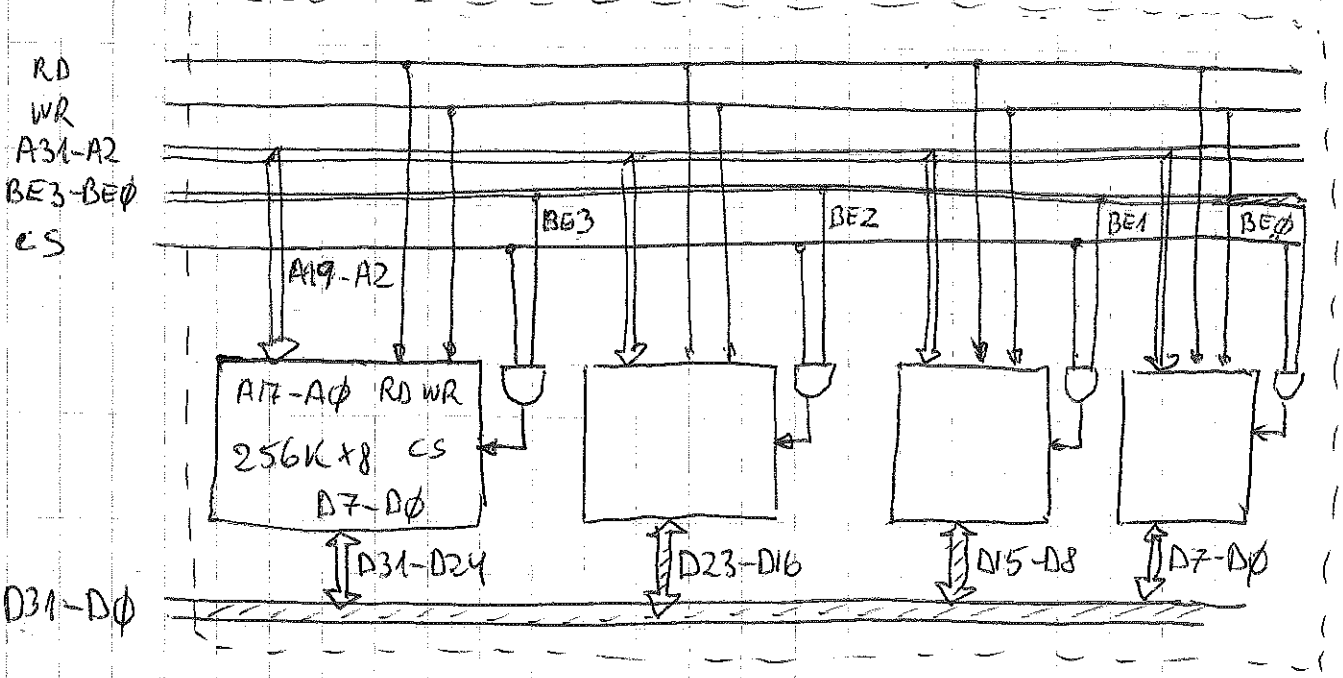
GEN es un circuito secuencial que examina la misma línea CS que a la memoria. Cuando detecta un 1 en un ciclo, coloca WAIT en 1 por 2 ciclos y luego ϕ en el tercer ciclo. Cuando CS es ϕ WAIT queda en tristate. El diagrama de estados para GEN es:



Es importante que WAIT permanezca en tristate cuando esta memoria no es accesada.
La gracia de un bus es que se pueden conectar varios módulos de memoria que funcionan en forma independiente, pero que reaccionan (se activan, se seleccionan) en rangos disjuntos. En este caso cada módulo tiene su propio generador de ciclos de espera y se conectan a la misma línea WAIT. El corto circuito se evita porque cada módulo lleva su línea WAIT a tristate cuando se realiza un acceso a una dirección que no le pertenece.

(v) Modificar un byte o una media palabra.
Para leer 1 byte se puede leer la palabra completa y extraer de ahí el byte que interesa. Esto no penaliza en nada el tiempo de ejecución de la instrucción "load byte" en comparación con la instrucción "load word".
En cambio para escribir 1 byte, es necesario leer primero la palabra completa, modificar el byte de interés y luego escribir la palabra completa. Esto se debe a que la interfaz de memoria que hemos visto no permite modificar solo 1 byte de una palabra de 32 bits. Esto sí penaliza los "store byte" puesto que requieren ahora 2 accesos a la memoria.

Esto se soluciona utilizando las señales BE3-BE0 (Byte Enable) que provee la CPU, que indican cuáles son los bytes que se modifican durante una escritura en memoria. Entonces, es necesario modificar el módulo 256K × 32 (1 MB) visto:



El CS de cada chip sólo se activa si la línea BE correspondiente está en 1. De esta forma se puede escribir selectivamente 1 byte o una media palabra sin modificar el resto.

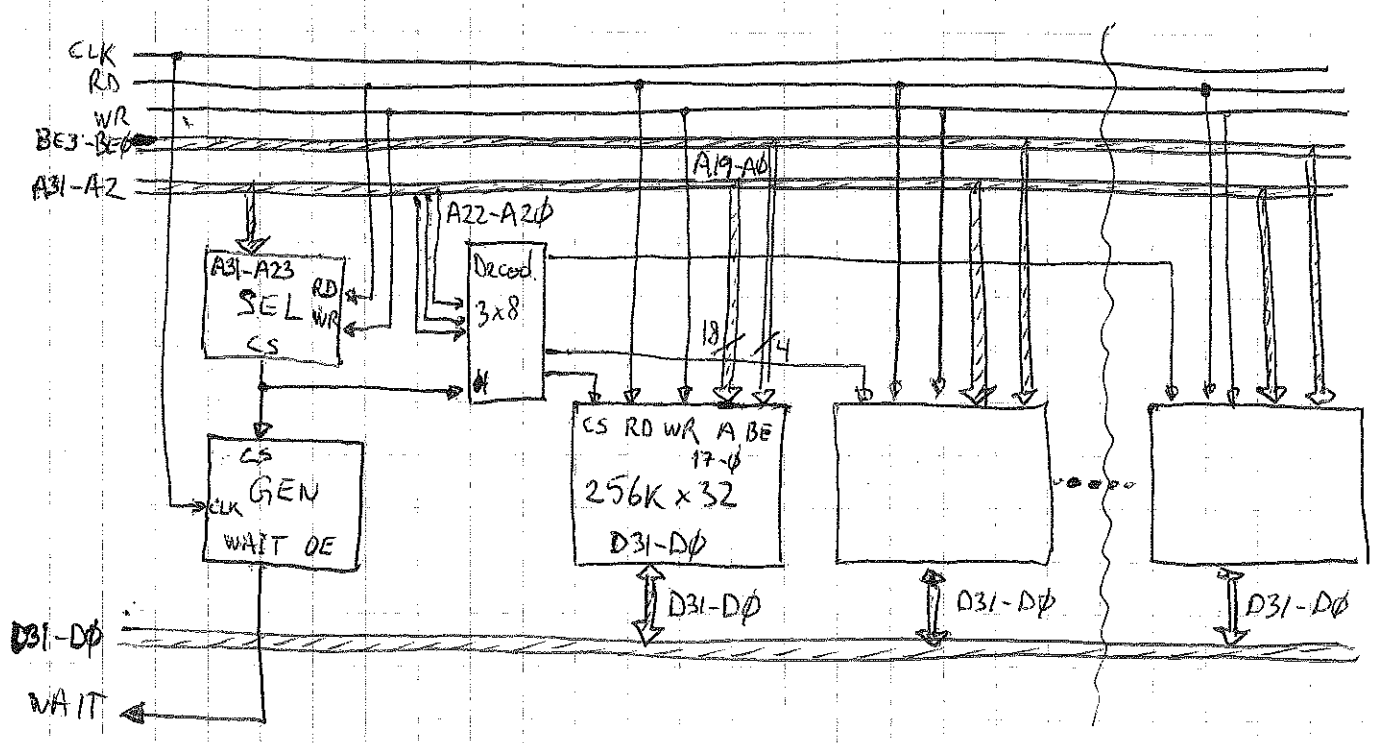
Nota importante: en este esquema cuando la CPU escribe un byte en memoria, debe colocarlo en el lugar adecuado en el bus de data.

Mem 12

		BE3	BE2	BE1	BE0	¿datos en?
↑tb- R1, [R0+0]	⇒	1	0	0	0	D31-D24
↑tb- R1, [R0+1]	⇒	0	1	0	0	D23-D16
" +2	⇒	0	0	1	0	D15-D8
" +3	⇒	0	0	0	1	D7-D0
↑th R1, [R0+0]	⇒	1	1	0	0	D31-D16
" +2	⇒	0	0	1	1	D15-D0

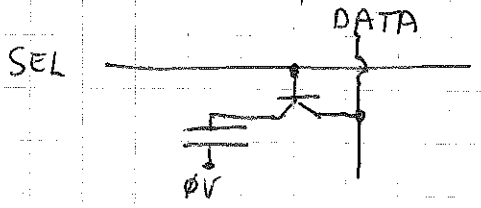
La ubicación de los datos en el bus depende del hecho de que esta arquitectura es big-endian. ¿En donde irían los datos en una máquina little-endian?

Ahora estamos en condiciones de examinar la forma final de una interfaz de 8MB usando el nuevo módulo de 1MB · 256K x 32 (con línea BE) y con ciclos de espera



(b) Dynamic RAM o DRAM

Son las más usadas debido a su bajo costo por bit. Se implementan con un condensador para almacenar 1 bit y un transistor para seleccionarlo:



Para escribir el bit se coloca SEL en 1 y un ϕ o 1 en DATA, lo que descarga o carga el condensador. Cuando SEL = ϕ, el condensador se mantiene aislado, por lo que mantiene su carga por algunos milisegundos. Para leer el dato se coloca SEL en 1. Si el condensador está cargado, se descarga generando una débil corriente que es detectada como 1. Esto significa que la lectura es destructiva: después de leer un bit hay que reescribirlo para no perder los 1s.

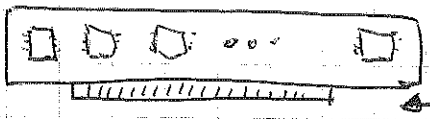
Las desventajas de este tipo de memoria son:

(i) Son lentas: tiempo de acceso de 60-70 ns (vs 10-20 para las estáticas).

(ii) Cada 4 milisegundos hay que reescribirlas completamente. Esto se debe a que los condensadores se descargan lentamente, aún cuando no se leen.

Mem 14

Las DRAMs se venden en chips o en SIMMs (Single Inline Memory Module). Cada SIMM contiene de 2 a 36 chips dependiendo de la capacidad del SIMM.



30 o 72 pines (que incluyen datos, direcciones y control)

Los SIMM de 30 pines tienen un bus de datos de 8 bits o 9 si incluyen paridad.
Los SIMM de 72 pines tienen un bus de datos de 32 bits o 36 si incluyen paridad.

La paridad es un bit adicional redundante que se calcula como el XOR de 8 bits de datos. La paridad permite detectar errores de almacenamiento en la memoria dinámica, producidos por la descarga de un condensador.

Durante la lectura se verifica que la paridad almacenada coincide con la paridad calculada a partir de los 8 bits de datos. Si no coincide se genera una interrupción por "parity error". La paridad siempre detecta errores de un solo bit en un byte.

Una motherboard tiene slots especiales para SIMMs de 30 pines u otro tipo de slot para SIMM de 72 pines, pero ambos tipos de SIMM son incompatibles.

Mem 15

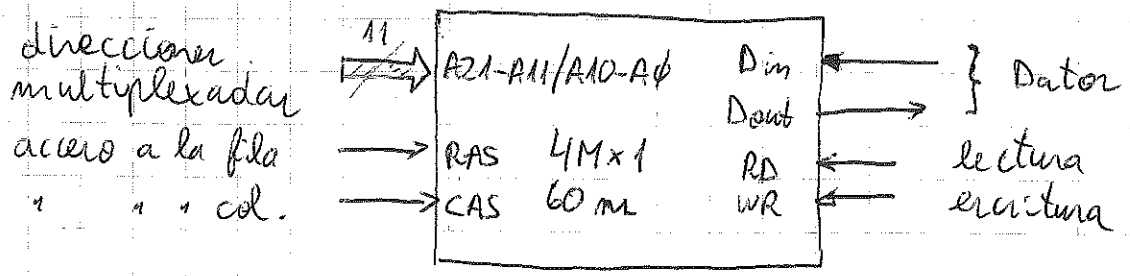
Los SIMM con paridad son compatibles con los SIMM sin paridad, pero si el SIMM es sin paridad la motherboard debe ofrecer un mecanismo para inhibir el chequeo de la paridad.

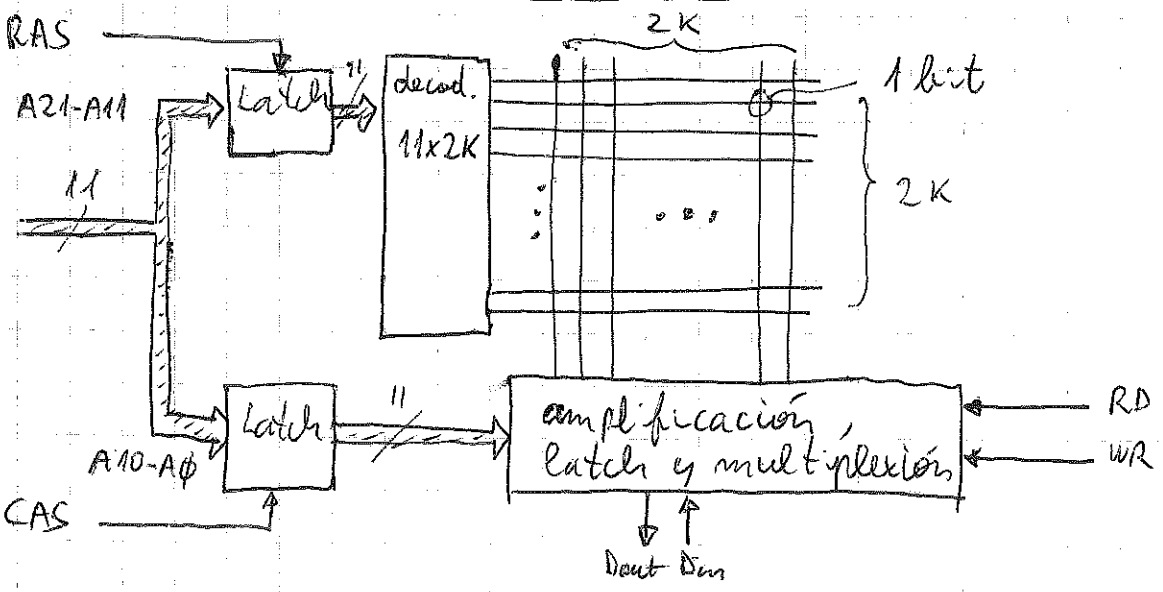
Los SIMM se agregan en grupos para completar el ancho del bus del procesador. Por ejemplo una motherboard para una 386SX (bus de dato de 16 bits) necesita 2 SIMMs de 30 pines. Si se necesita más memoria hay que agregar otros 2 SIMM. Una 486 requiere grupos de a 4 SIMM de 30 pines o 1 SIMM de 72 pines. (si la motherboard ofrece el tipo de slot adecuado), ya que la 486 tiene un bus de dato de 32 bits. Un pentium usa grupos de a 2 SIMM de 72 pines debido a que su bus de dato es de 64 bits.

Cuidado con el tiempo de acceso: un SIMM de 60 ns se puede colocar en una motherboard diseñada para SIMM de 70 ns, pero no viceversa. Si uno se equivoca el error no ocurrirá necesariamente de inmediato. Las motherboards ofrecen típicamente la posibilidad de agregar ciclos de espera (wait-states) si se colocan SIMM muy lentos, pero esto penaliza la eficiencia de ejecución.

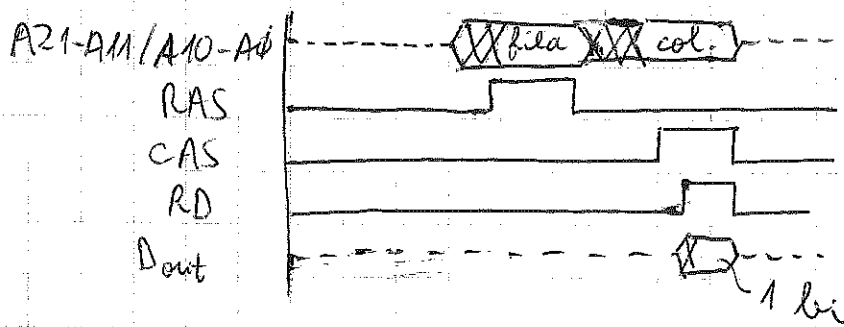
Mem 16

Los chips que vienen en un SIMM se organizan como una matriz de bits. Los más usados son los chips de 1 Mb, 4 Mb y 16 Mb.





El acceso a la memoria dinámica es más complejo que el de la memoria estática debido a la multiplexión del bus de direcciones.



Durante el acceso a la fila se lee toda la fila de condensadores. Como la lectura es destructiva (es decir descarga los condensadores), la fila se reescribe completamente a final del acceso. En el acceso a la columna se selecciona un bit dentro de la fila.

Por otra parte, cada fila debe ser reescrita cada 4 milseg para que no se borre. Esto se logra con un acceso a la fila sin acceso a la columna. De esto se encarga el hardware en forma transparente para el software.

También por cada acceso a una fila se pueden acceder 4 columnas dentro de esa misma fila lo que permite un acceso secuencial más eficiente que el acceso aleatorio. Este tipo de acceso se denomina "page mode".

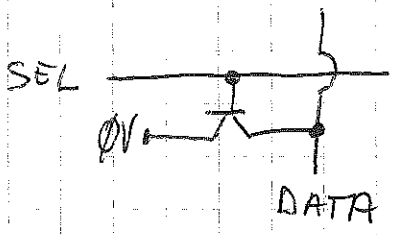
(c) EDORAM o Extended Data Out RAM

Esta es un variante de la memoria dinámica. Son compatibles con las DRAMs, pero a mismo tiempo de acceso pueden ser accesadas secuencialmente en forma más eficiente. Pero para aprovechar esta capacidad la motherboard tiene que ser rediseñada.

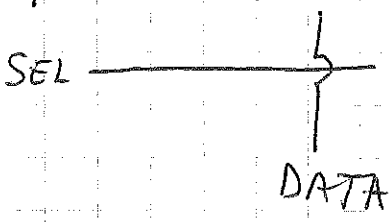
(d) Read Only Memory o ROM.

Vienen grabadas de fábrica y no se pueden escribir. No se borran al apagarlas. Son lentas. Se usan para almacenar el programa de bootstrap (el cargador del sistema operativo) o en control automático/embarcado. Son relativamente económicas pues requieren de a lo más 1 transistor por bit:

1 binario:

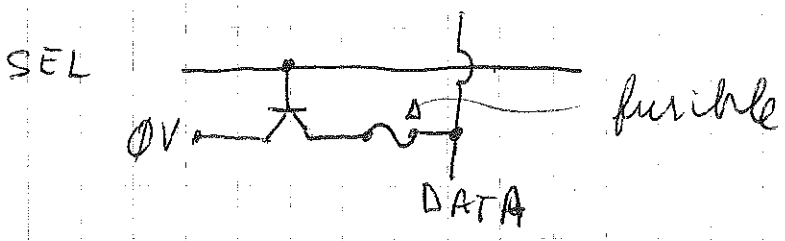


0 binario:



(e) PROM o Programmable ROM:

Es una ROM que se programa en dispositivos especiales. Cada bit se implementa con un transistor y un fusible:

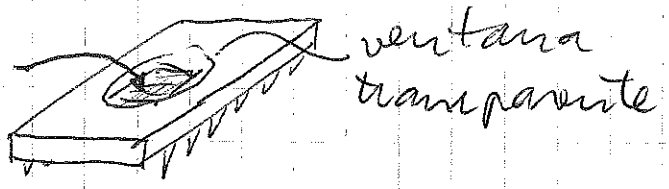


Los 0's se pueden cambiar por 1's aplicando una intensidad fuerte de corriente para quemar el fusible. Pero una vez que se quema el fusible, no se puede regresar a 0.

Luego de grabarlas, se instalan en el computador, el que no puede modificarlas (en dicha forma como ROMs). La ventaja sobre las ROMs es que no es necesario pedirle al fabricante que las grabe.

(f) EPROM o Erasable PROM.

Son similares a las PROM, pero el fusible se puede restituir con luz ultravioleta aplicada en una ventana transparente ubicada sobre el silicio:



La luz solar contiene luz ultravioleta de modo que una vez que se borra su contenido se debe tapar la ventana, para que no se borre.

(g) Flash

Este es un tipo de memoria que no se borra al apagarla, pero que se puede escribir en el mismo computador, aunque no tan eficientemente como una DRAM o SRAM. Se pueden cambiar bits por 1s fácilmente, pero se borran por filas completas. Son apropiadas para implementar "discos" de estado sólido o para programas de bootstrap.