Conexión de memoria y dispositivos periféricos

Electrónica IV

Mg.Ing. Esteban Volentini (evolentini@herrera.unt.edu.ar)

https://facetvirtual.facet.unt.edu.ar/course/view.php?id=165

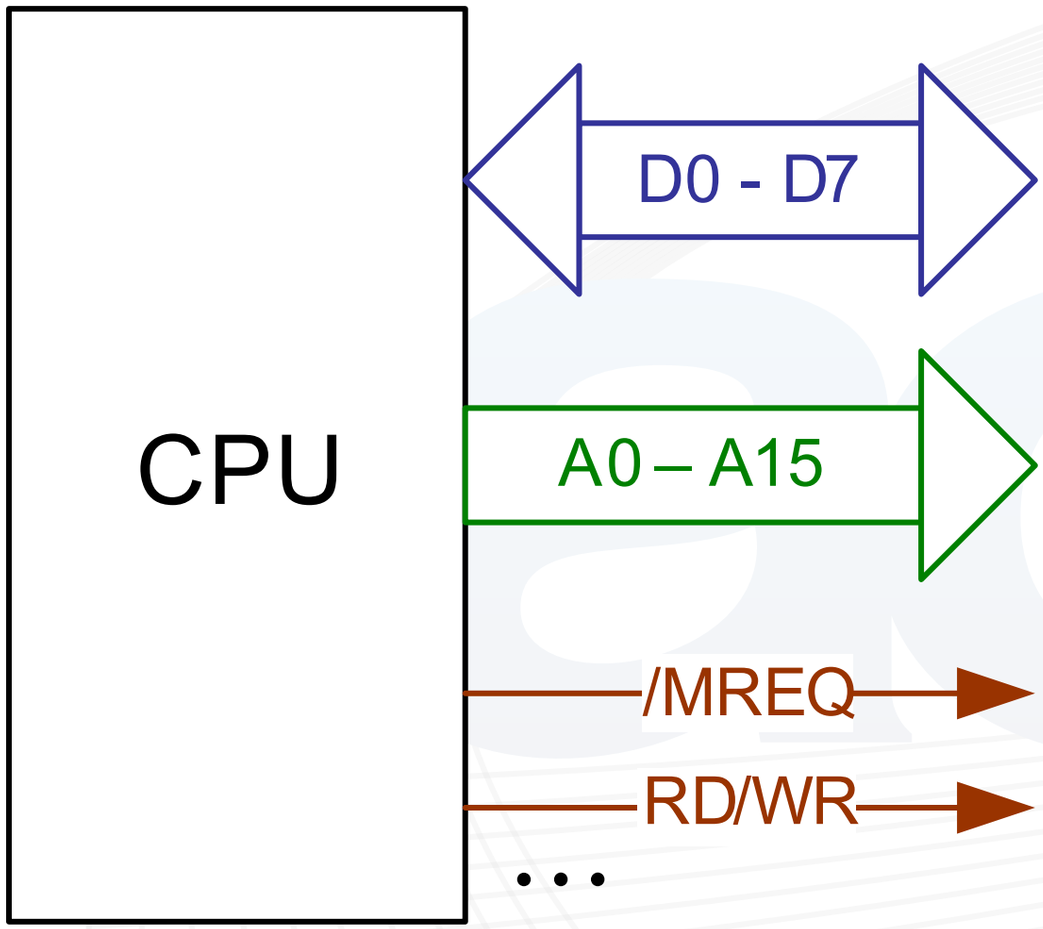
Desarrollo Conceptual

- Se empleará un Microprocesador de 8 bits, llamado CPU08
- Data bus – 8 bits.
- Address bus – 16 bits.
- ¿Máx cantidad de memoria posible?
- Representación 0x0F=$0F

Conexiones internas del CPU08

- Dentro del µC el CPU08 se vincula con memoria y con los dispositivos de E/S a través de tres buses
- Bus de Direcciones: Esta formado por 16 líneas (A0 a A15) y permite seleccionar 64K Direcciones.
- Bus de Datos: Esta formado por 8 líneas bidireccionales (D0 a D7) y permite intercambiar información.
- Bus de Control: Esta formado por varias líneas independientes y sirve para enviar y recibir señales de control y de estado.

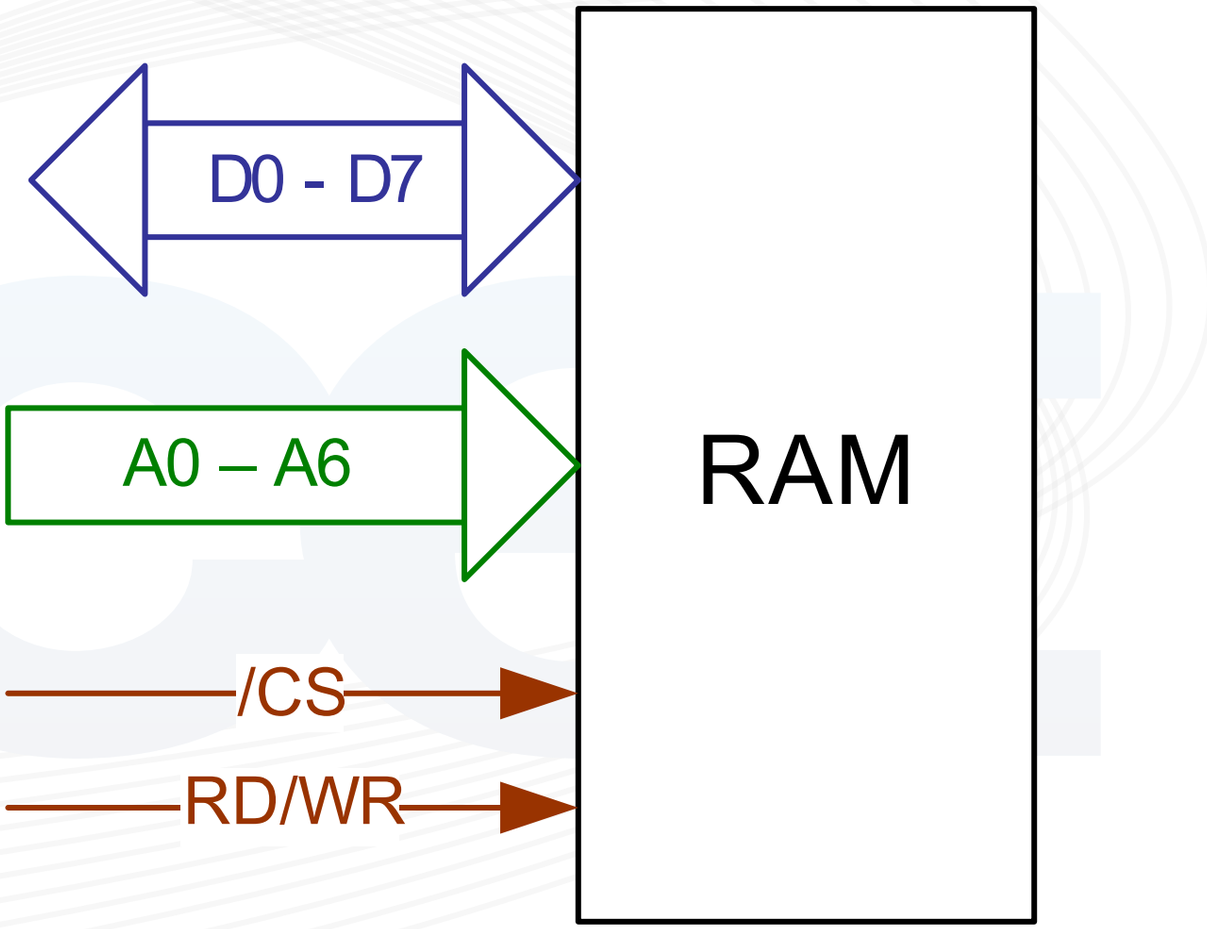
Entradas y Salidas del Procesador



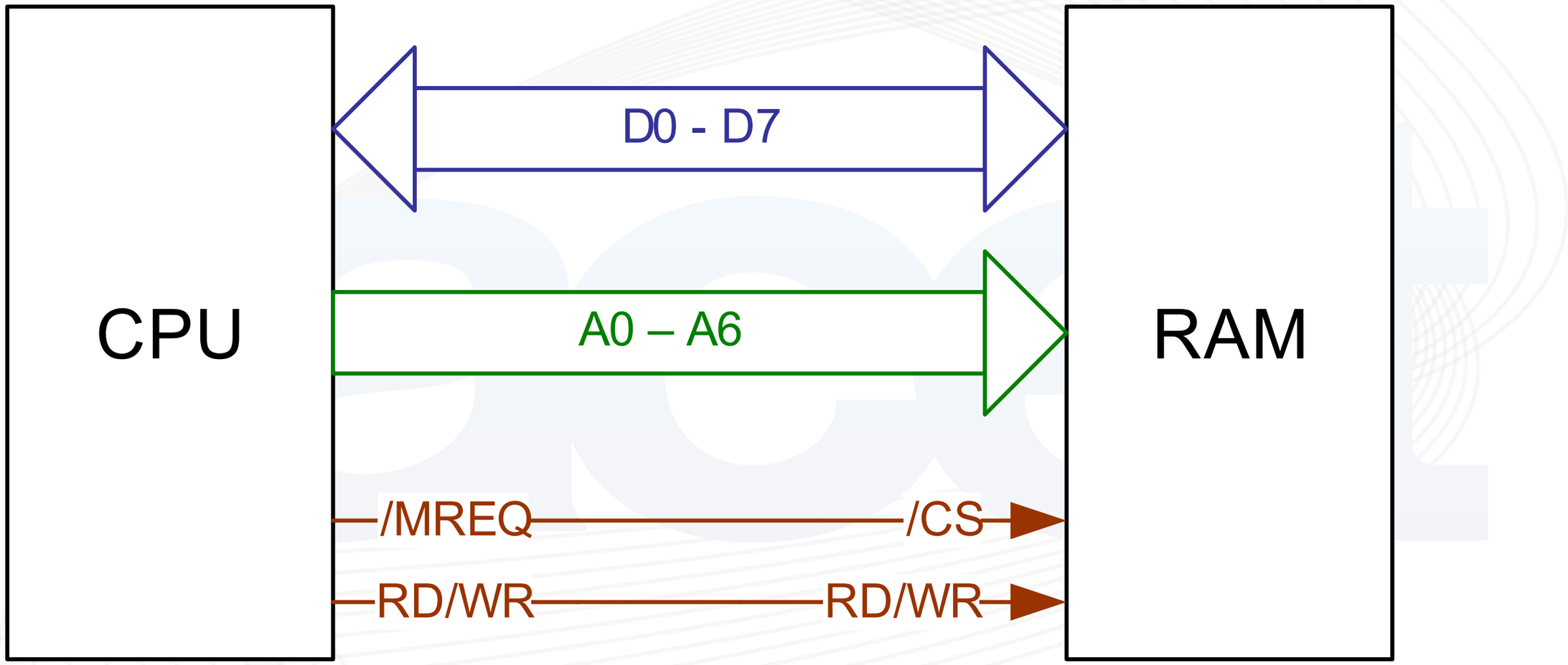
Entradas y Salidas de la Memoria

- Ahora revisemos las conexiones que tiene una memoria RAM de 128 x 8
  - Líneas de Datos: Depende de cuantos bits se almacena en cada dirección, en nuestro caso 8 líneas, de D0 a D7.
  - Líneas de Direcciones: Depende de cuantas direcciones tenga la memoria, en nuestro caso 7 líneas, de A0 a A6.
  - Lectura o Escritura: Una línea que indica el tipo de operación a realizar
  - Selección: Una línea que habilita la operación del chip. ¿Para qué sirve?

Entradas y Salidas de la Memoria

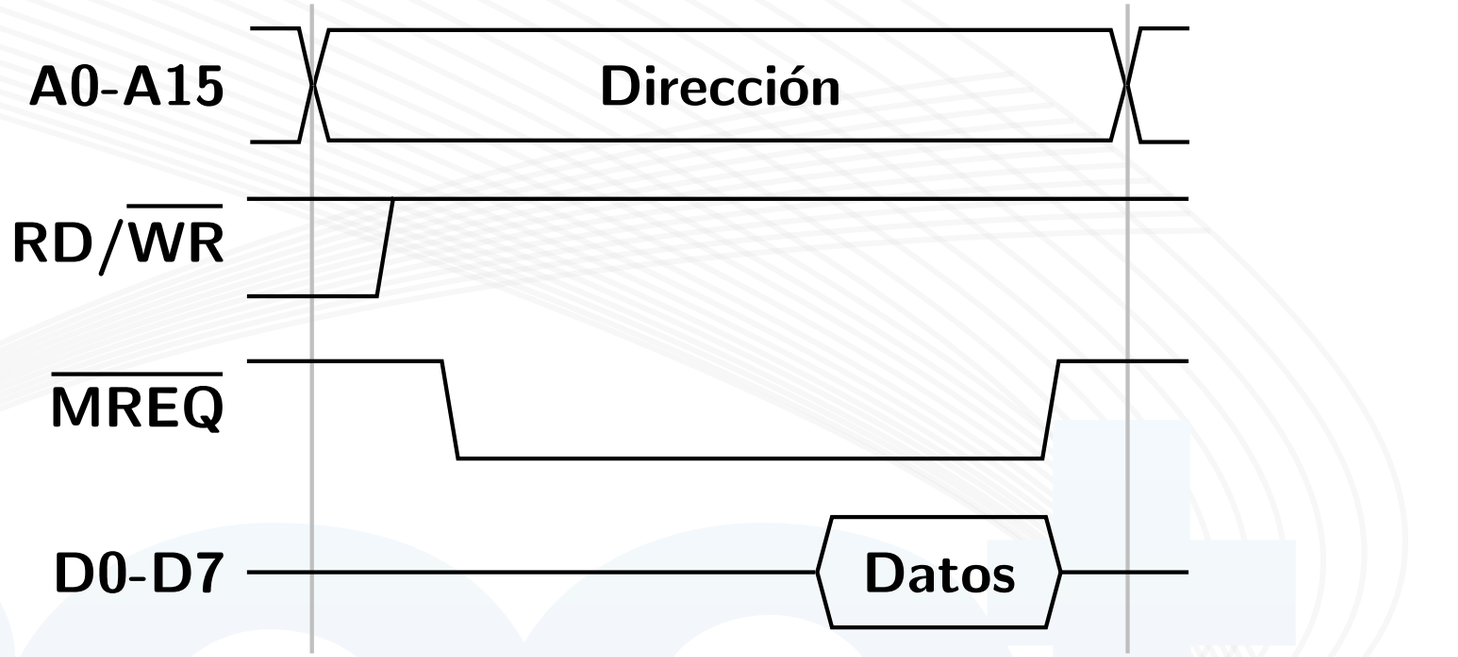


Como se conectan?

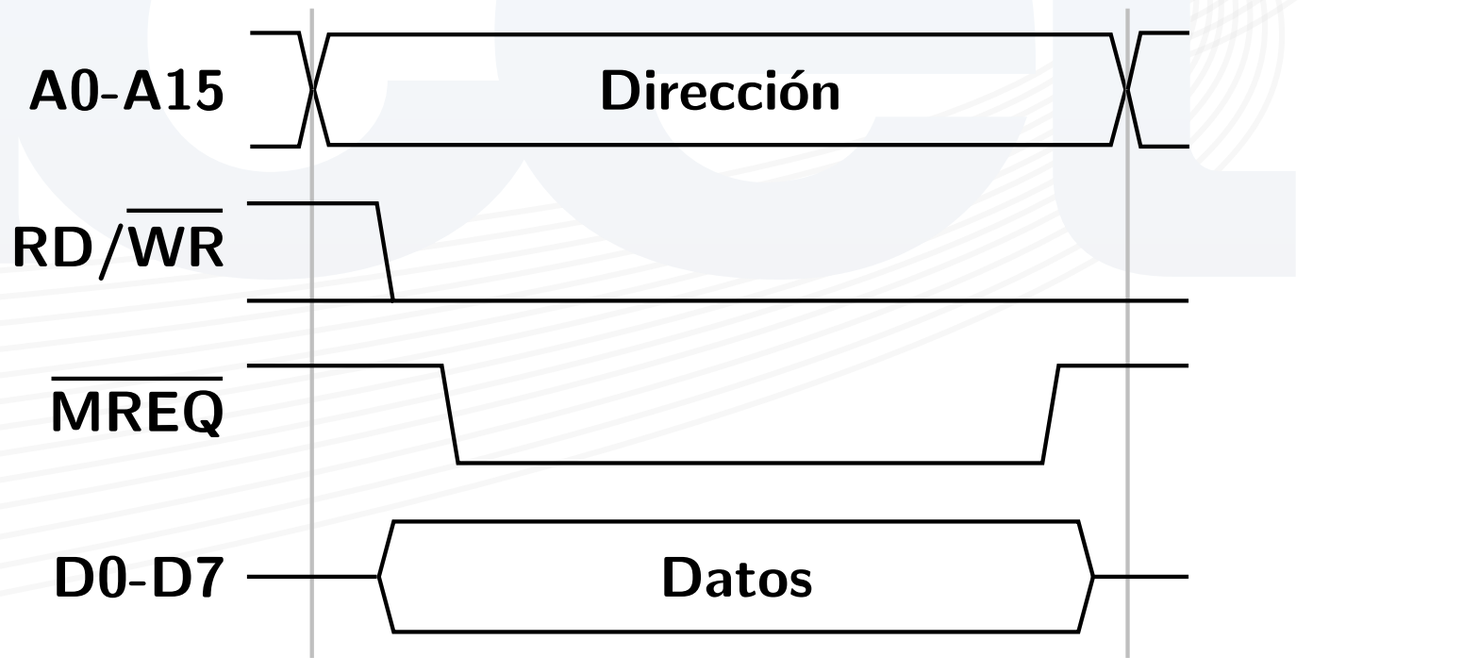


Diagramas de tiempos típicos

Lectura



Escritura



Mapas de Memoria

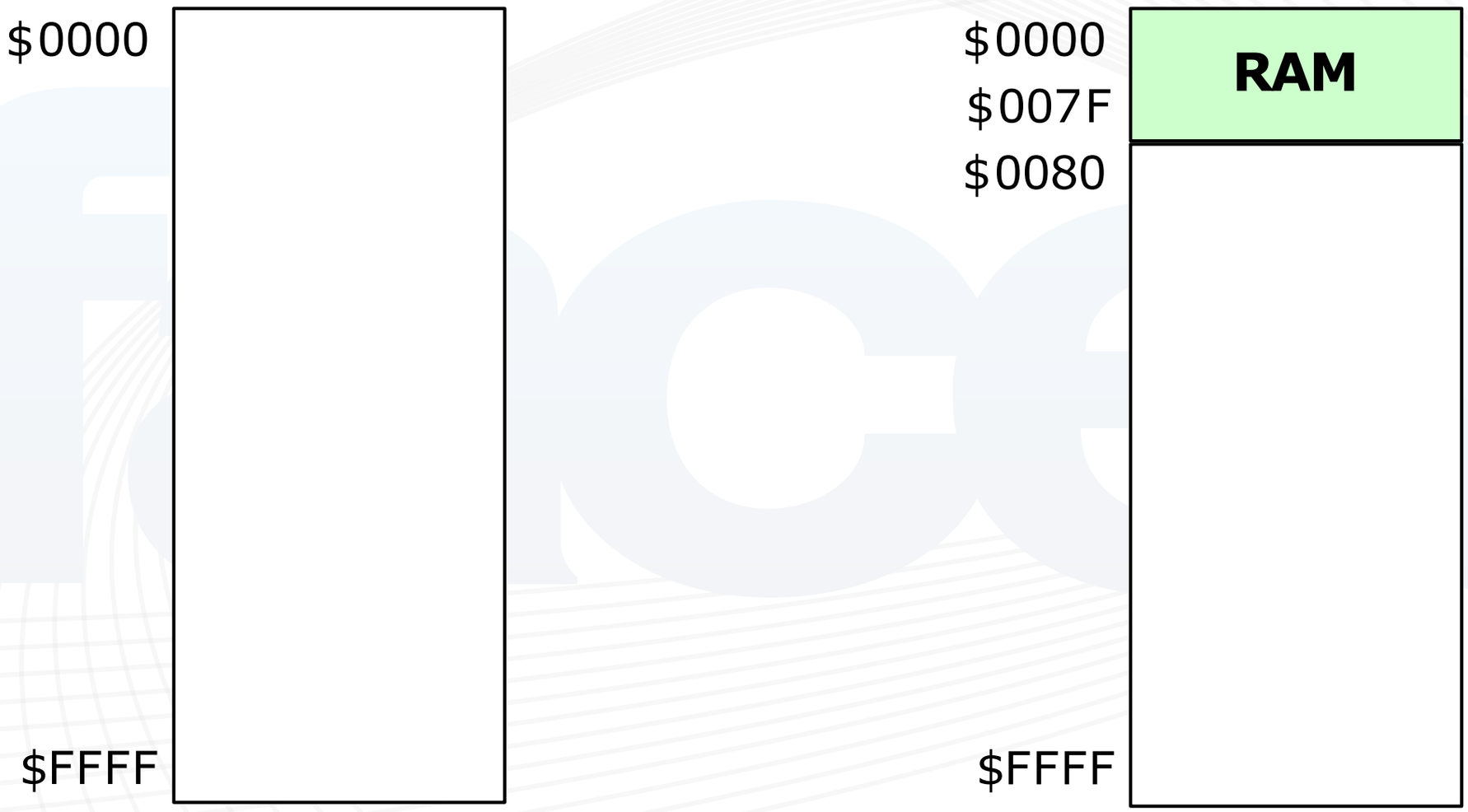
- Se llama espacio de direcciones al conjunto de todas las direcciones posibles a las que podría acceder un CPU.
- En el caso del CPU08 es de 65.536 o 64K direcciones.
- No siempre se usa la totalidad del espacio disponible, puede no hacer falta.
- Los mapas de memoria muestran las partes del espacio de direcciones a las que se conecta Memoria Física.
- En nuestro ejemplo anterior podemos representar en un mapa de memoria en qué direcciones está conectada la RAM.

Ejemplo de Aplicación

Ejemplo de Aplicación

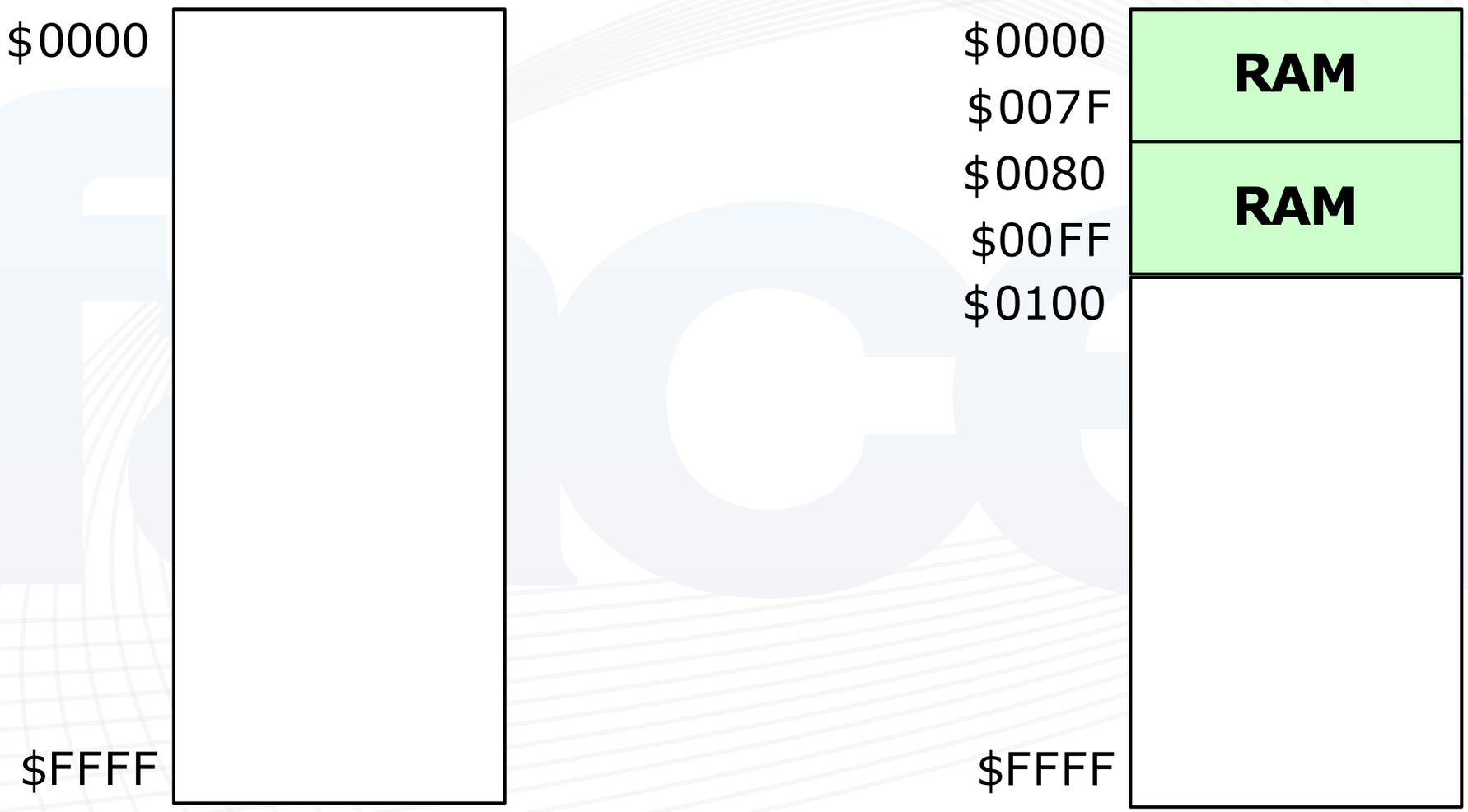
Pero A7-A15 son indiferentes, no están conectados a la RAM

¿qué hay en la dirección $0080?

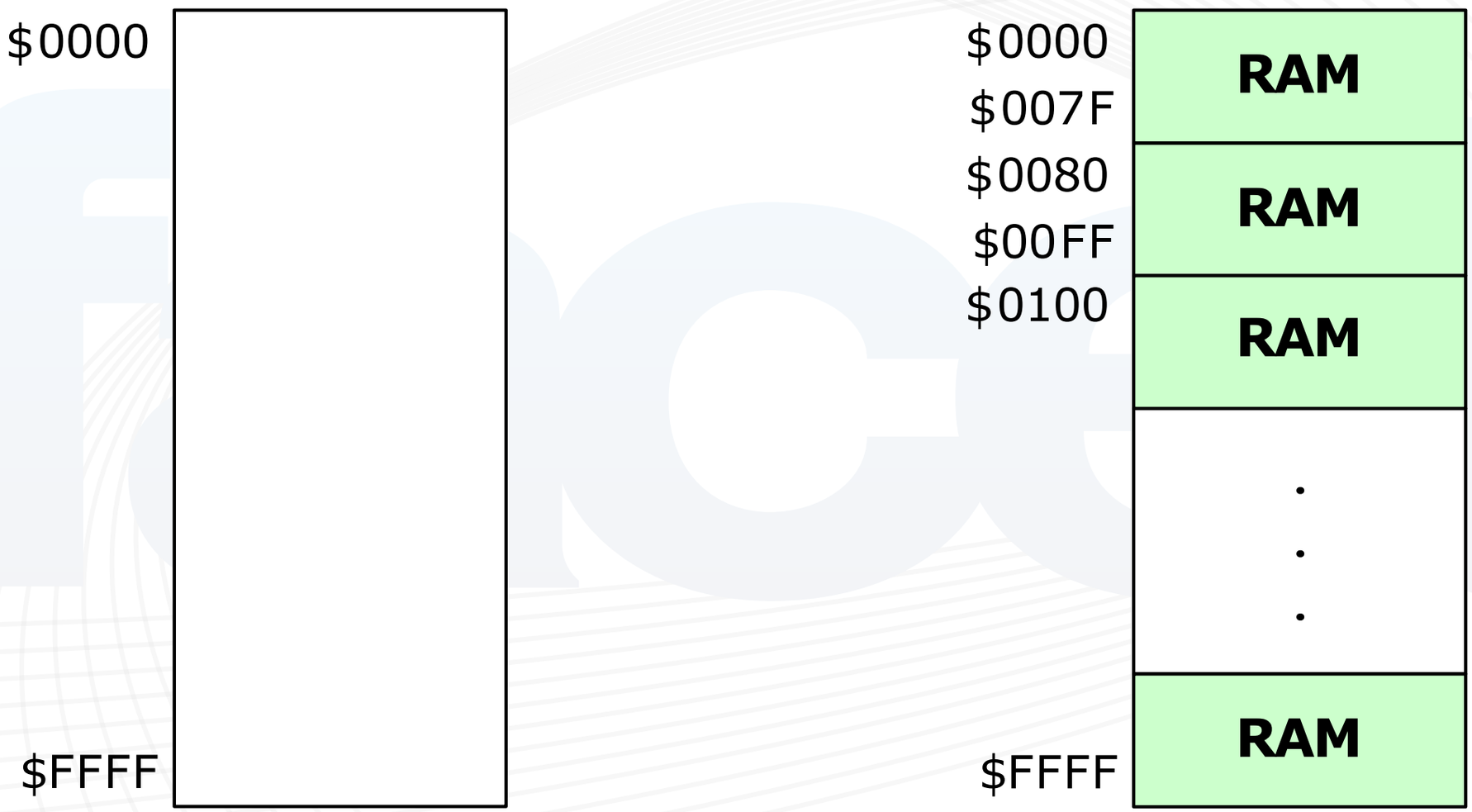


Ejemplo de Aplicación

Y así sucesivamente en $0100…



Ejemplo de Aplicación

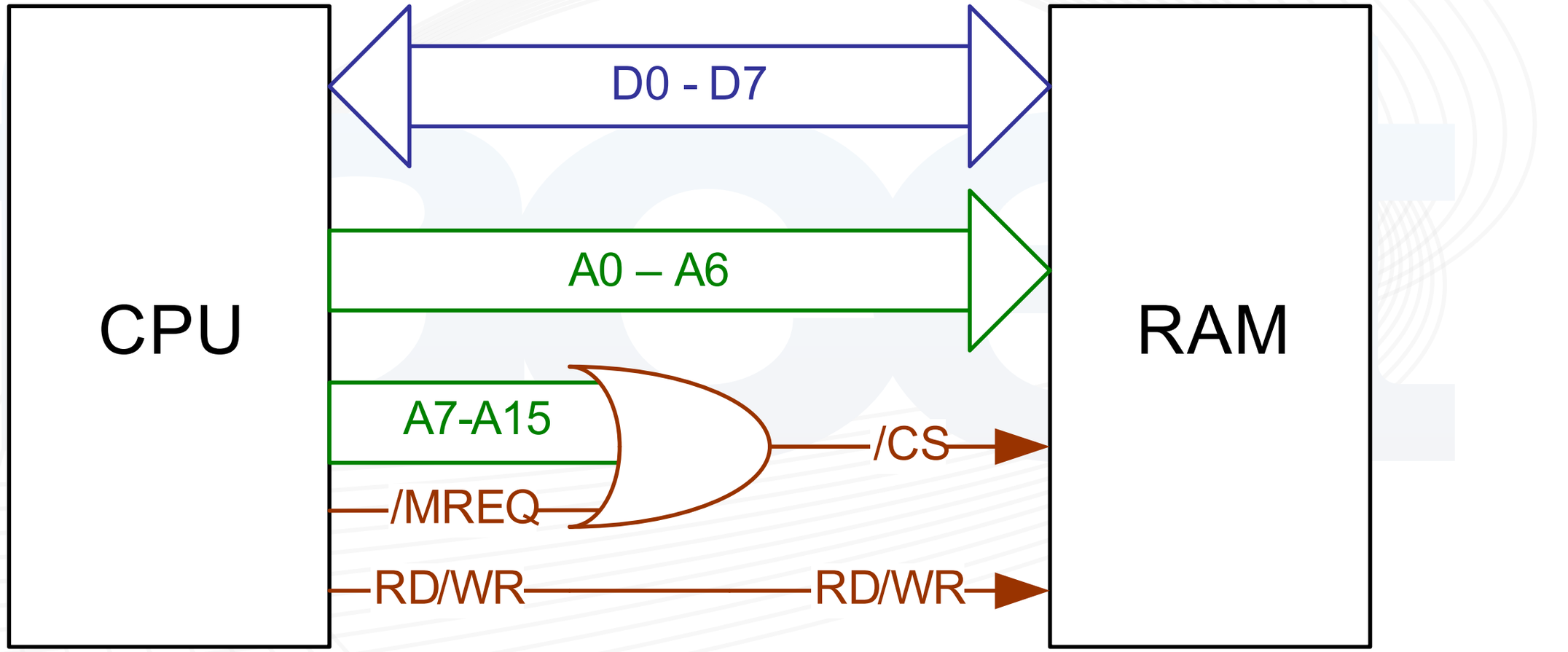


Conexión Redundante (100%)

- Este tipo de conexión se llama **redundante**.
- Ventaja: simple. ¿Por qué?
- Desventaja: ocupa todo el mapa de memoria e impide conectar más memoria en el futuro.
- Si se desea que la RAM ocupe la primera parte del mapa de memoria de $0000 a $007F – **sin redundancia**…

Conexión sin Redundancia

En general los chips de RAM o ROM traen varios pins CS. ¿Por qué?



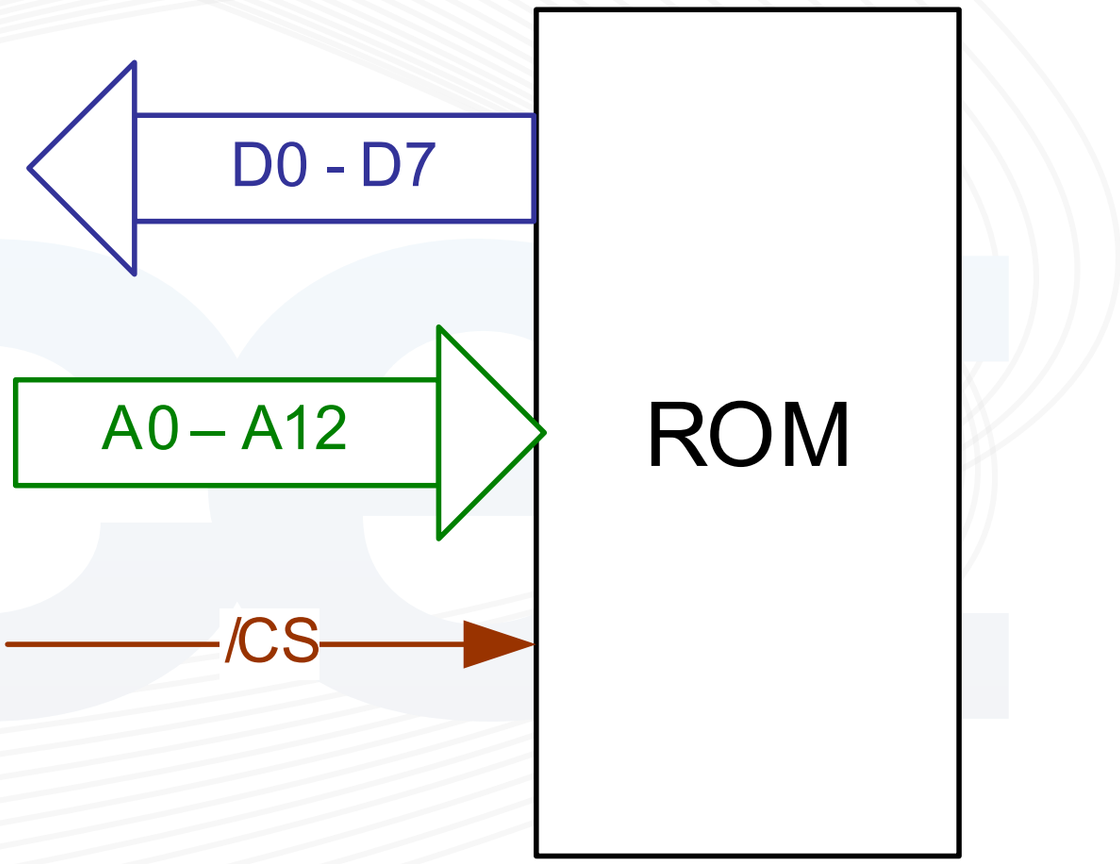
Conexión sin Redundancia

- En este tipo de conexión asignamos un mínimo posible del mapa de memoria.
- Ventaja: óptimo aprovechamiento del mismo.
- Desventajas
  - Mayor cantidad de componentes
  - Tiempo de propagación a través de las compuertas.

ROM 8K x 8 – Presentación.

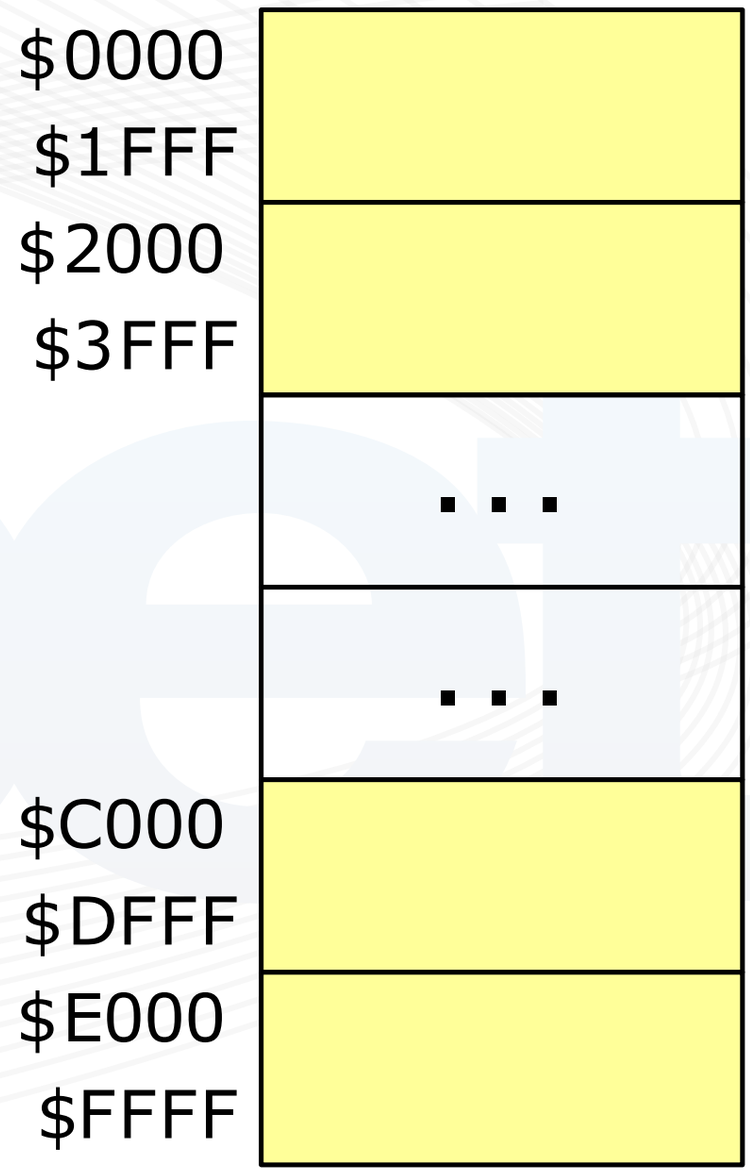
- Conexiones que tiene una memoria ROM de 8K x 8:
  - Líneas de Datos: Depende de cuántos bits se almacena en cada dirección, en nuestro caso 8 líneas, de D0 a D7.
  - Líneas de Direcciones: Depende de cuántas direcciones tenga la memoria, en nuestro caso 13 líneas, de A0 a A12.
  - Selección: Una línea que habilita la operación del chip.

Entradas y Salidas de la Memoria



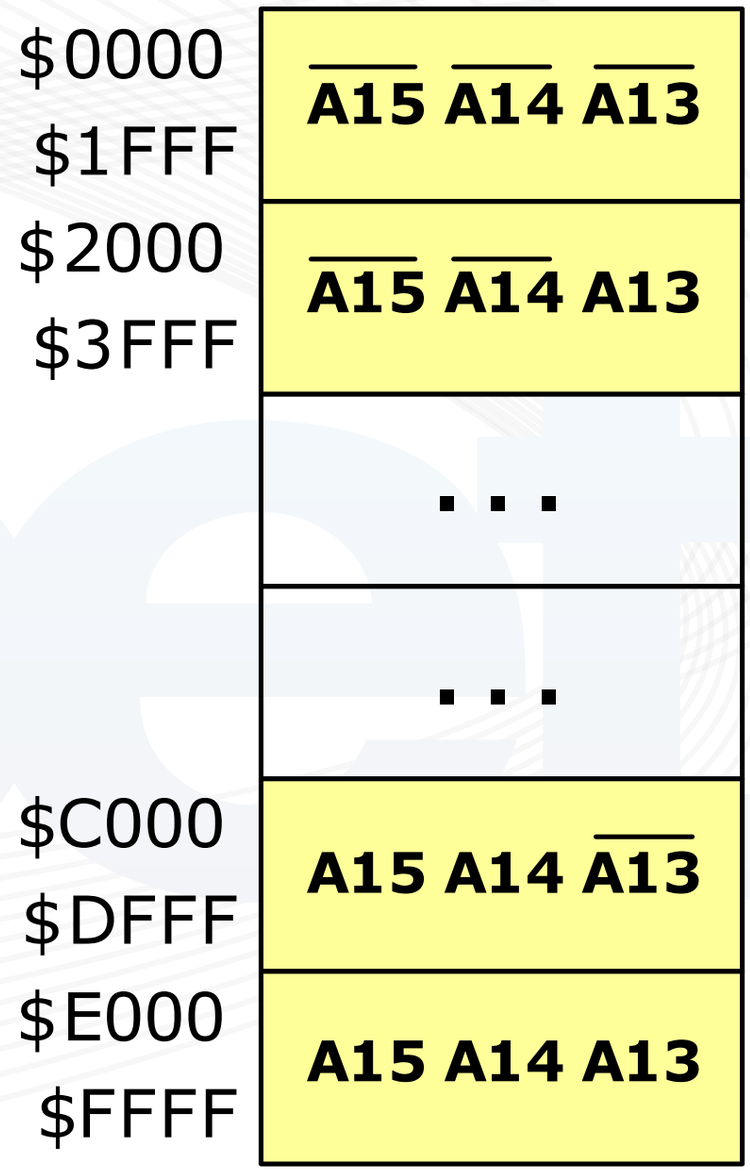
División del Mapa de Memoria

- Dado que nuestro ROM es de 8K, dividimos el mapa de memoria en fracciones de 8K.
- Cada fracción se caracteriza por una combinación de valores de las 3 líneas superiores de direcciones



División del Mapa de Memoria

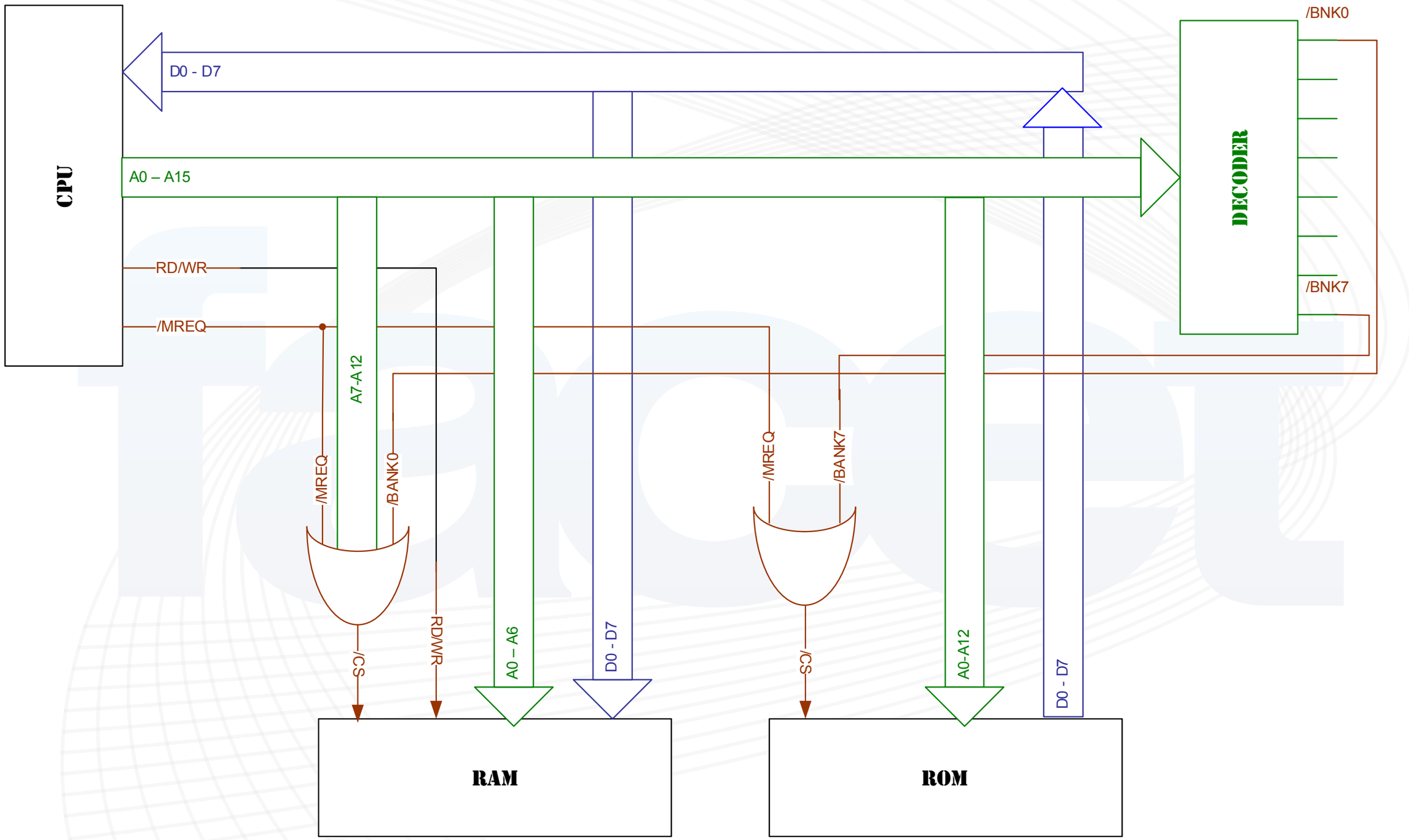
- Dado que nuestro ROM es de 8K, dividimos el mapa de memoria en fracciones de 8K.
- Cada fracción se caracteriza por una combinación de valores de las 3 líneas superiores de direcciones



División del Mapa de Memoria

- De acuerdo a la zona en la que queremos ubicar el Chip, elegimos la combinación de líneas de direcciones que selecciona al mismo.
- Una opción es utilizar compuertas y negadores para sintetizar la función de selección.
- Otra opción es utilizar un decodificador de 3 a 8 líneas para generar 8 líneas de selección, una para cada espacio del mapa.

Conexión sin redundancia

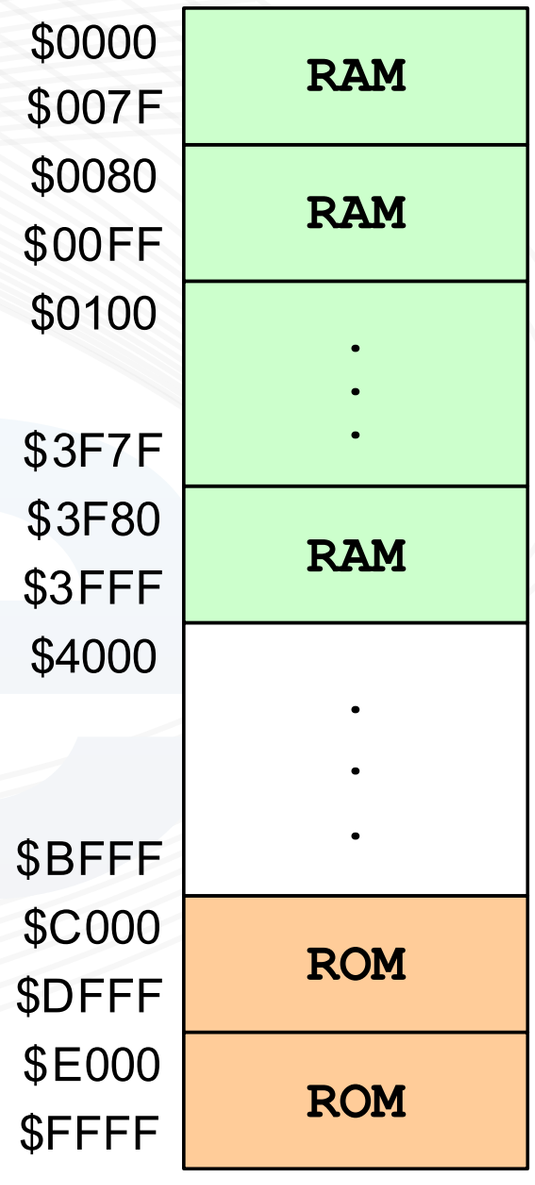


Posibles mejoras

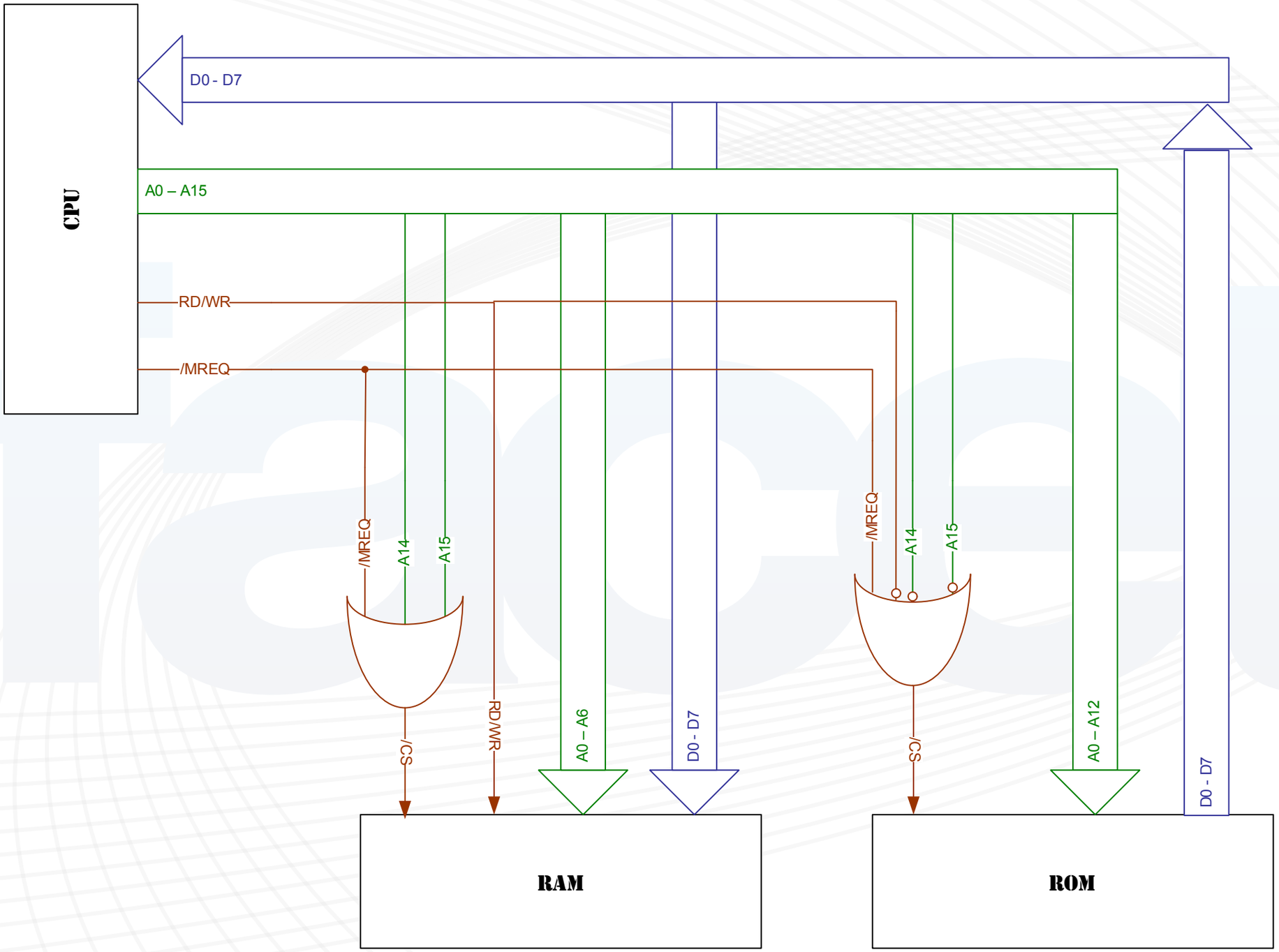
- /MREQ, mejor como habilitación del DECO
- Un error de programa quemaría el CPU y la ROM.
  - ¿Qué instrucción provoca esto?
  - ¿Forma de Evitarlo?

Conexión con Redundancia

- Conectar un chip RAM y un chip ROM.
- Se desea que una mitad del mapa quede libre para futuras ampliaciones.
- Dibuje el mapa de memoria resultante.
- RAM a partir de $0000, ROM al final del mapa – para este Micro.
- Dibuje el esquema del circuito completo.
  - A15, A14 definen las dos áreas:
    - 00 ⇒ RAM, 11 ⇒ ROM



Conexión con Redundancia



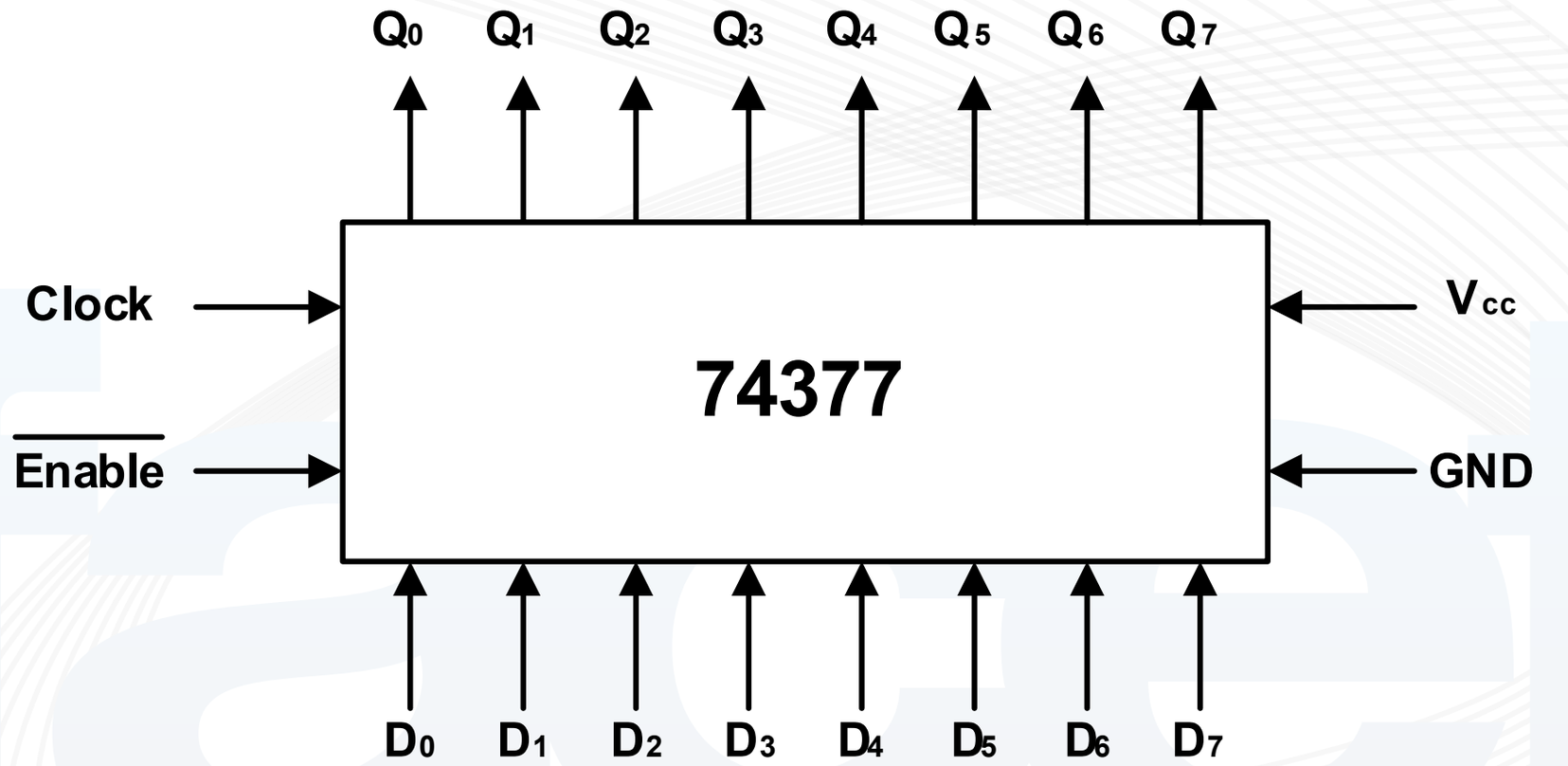
Conexión con Redundancia

- Memoria Conectada Real = MR
- Memoria Ocupada en el Mapa = MO
- ¿Cuánto vale cada una en ej. anterior?
- En general
  - MR <= MO
  - La igualdad es cuando no hay redundancia.
  - La redundancia permite economizar compuertas.
  - Con la seguridad de que no se requiere el espacio redundante en el futuro.
- ATENCION:
  - No superponer nuevos chips con MO. ¿Por qué?

Salidas digitales en un CPU

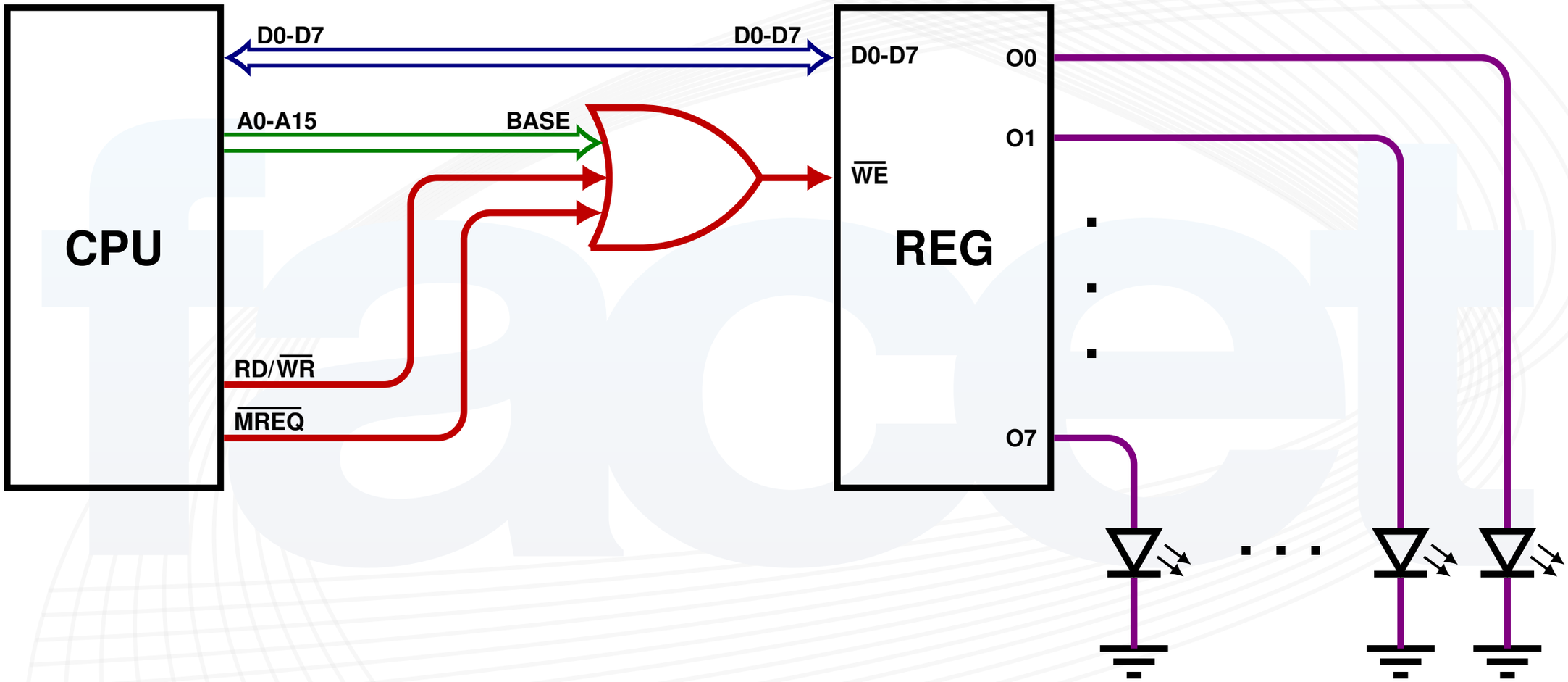
- Se propone agregar un conjunto de salidas digitales de propósito general
  - Por ejemplo para controlar los display de 7 segmentos en el reloj del proyecto integrador
- Se puede utilizar un registro tipo D conectado al bus
  - Se puede considerar como una memoria de un byte, es decir que ocupa solo 1 dirección
  - Se debe asignar una dirección de memoria a este registro con una lógica de selección en una dirección arbitraria *BASE*
- Las salidas del registro se conectan a los elementos que se desean controlar

Presentación de un registro



- Las salidas Q_i toman el valor de las entradas D_i cuando la señal /Enable está activa (en bajo) y se se produce un flanco ascendente en la señal Clock

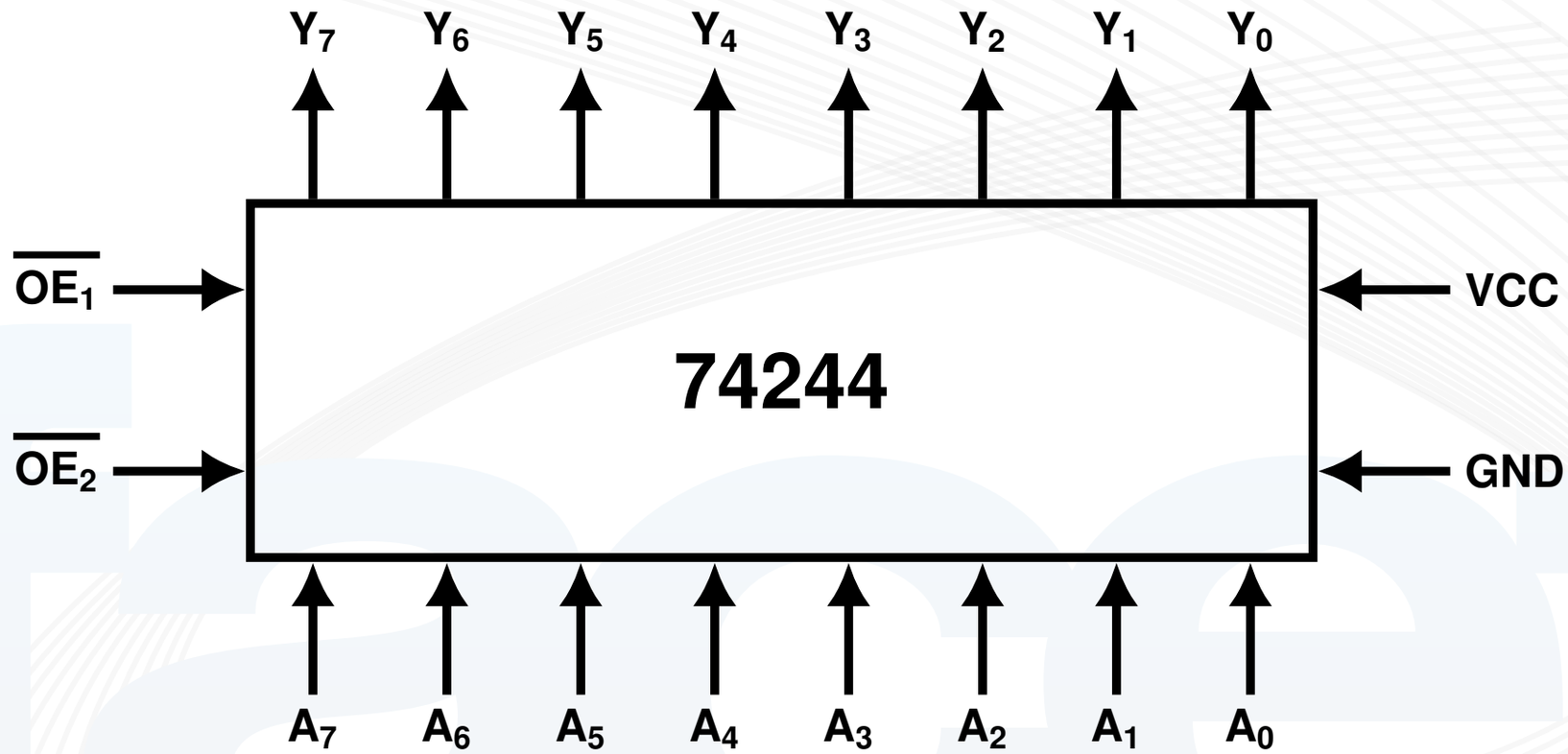
Salidas digitales en un CPU



Salidas digitales en un CPU

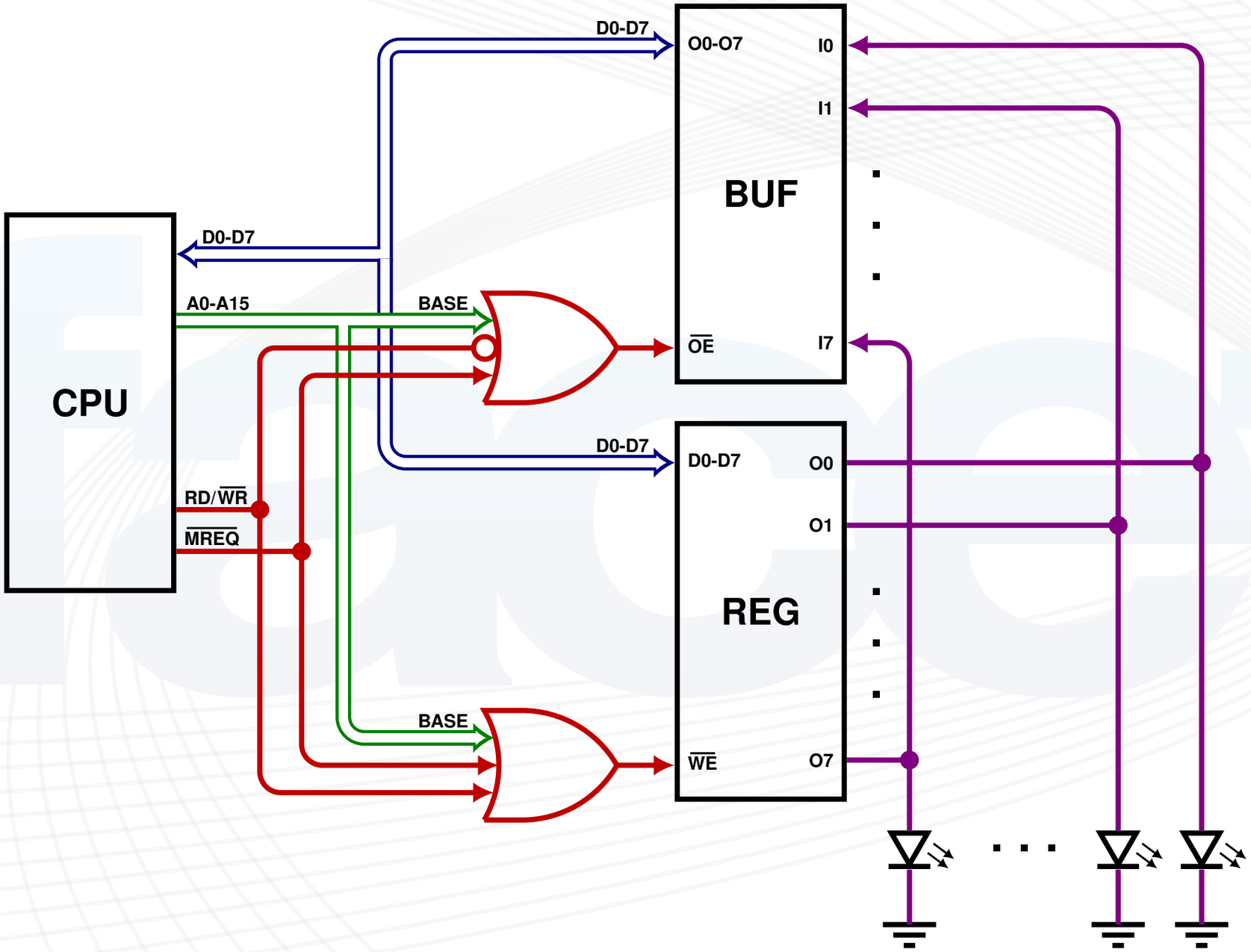
- Esta conexión permite controlar el estado de los leds escribiendo el valor adecuado en la dirección *BASE*
  - En este caso los leds se prenden al escribir un 1 en el bit correspondiente
  - La corriente de salida del registro debe ser suficiente para alimentar a la carga
- Sin embargo no es posible leer el estado de una salida
  - Al realizar una lectura de la dirección *BASE* no se activa ningún dispositivo y el valor resultante es indeterminado
- Esto es un problema bastante significativo cuando se desea cambiar el estado de un led sin afectar el resto
  - Sería necesario disponer de una variable para recordar el estado

Presentación de un buffer 3state



- Las salidas Q_i toman el valor de las entradas D_i cuando la señal /OE está activa (en bajo)
- Cuando la señal /OE está inactiva (en alto) las salidas permanecen Q_i en alta impedancia (desconectadas)

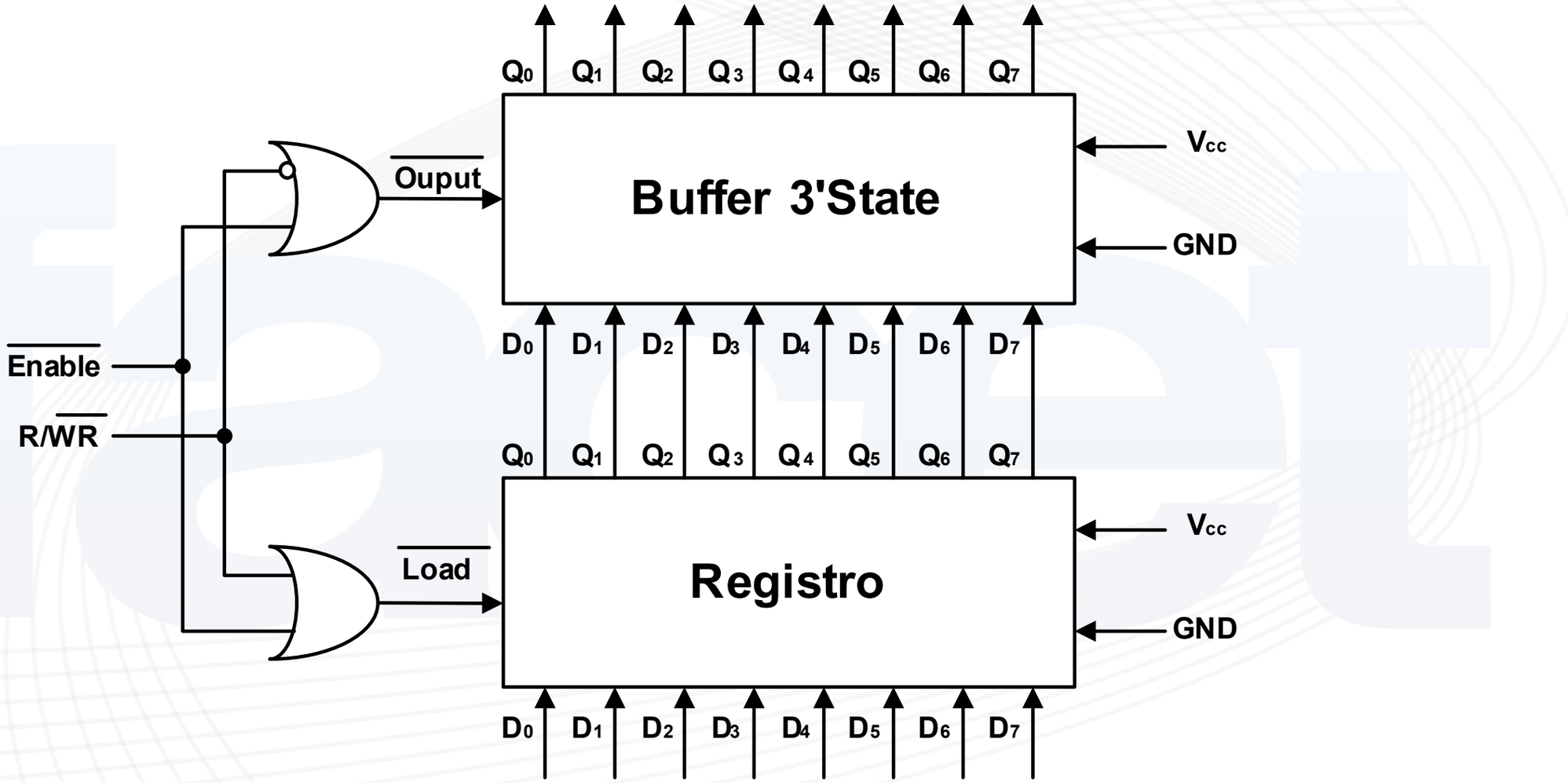
Salidas digitales en un CPU



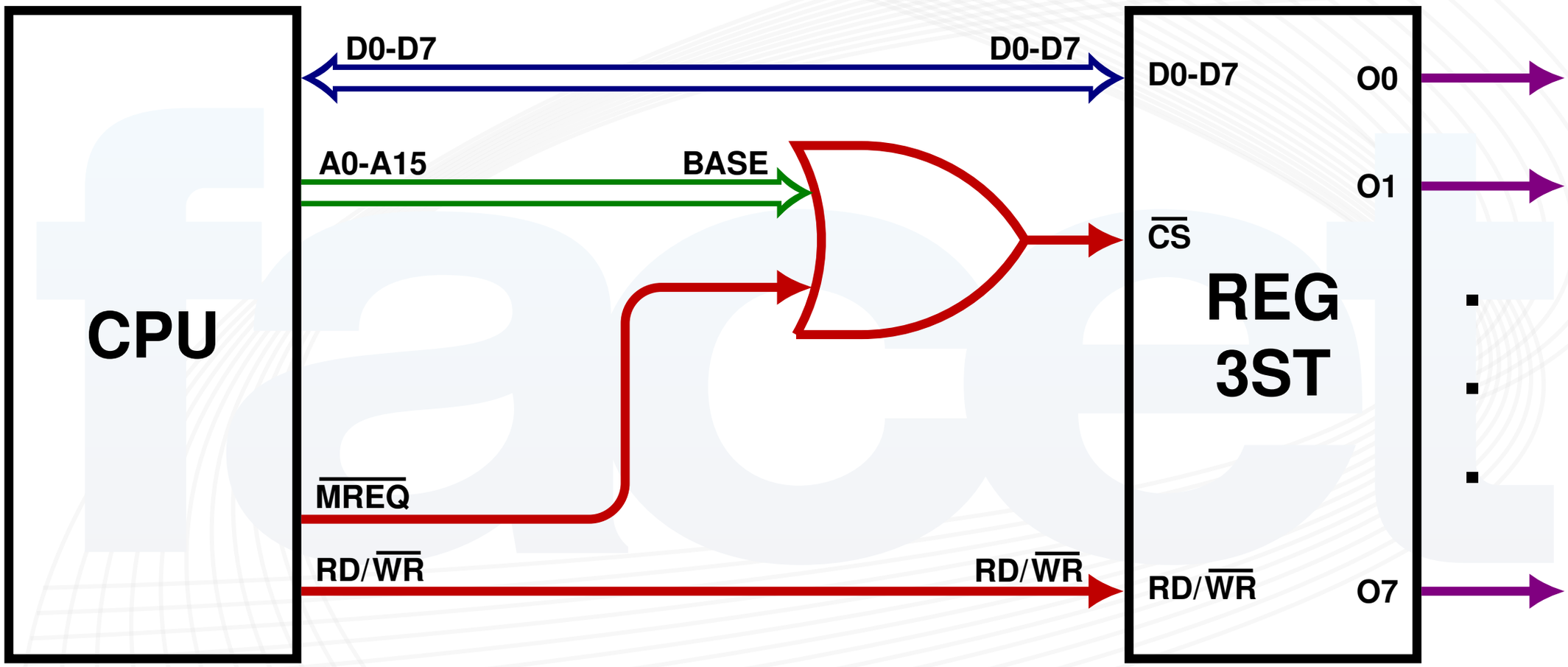
Salidas digitales en un CPU

- Se utiliza un registro con un buffer 3-state
- Permite escribir y leer el valor del registro como una dirección de memoria normal
  - Mismos componentes que una memoria RAM estática
- Pero el contenido del los flip-flips que forman las celdas de almacenamiento están conectados a un circuito externo
  - Se puede controlar el circuito externo en función de lo que se escriben en esos flips-flops

Conexión de un registro al Bus



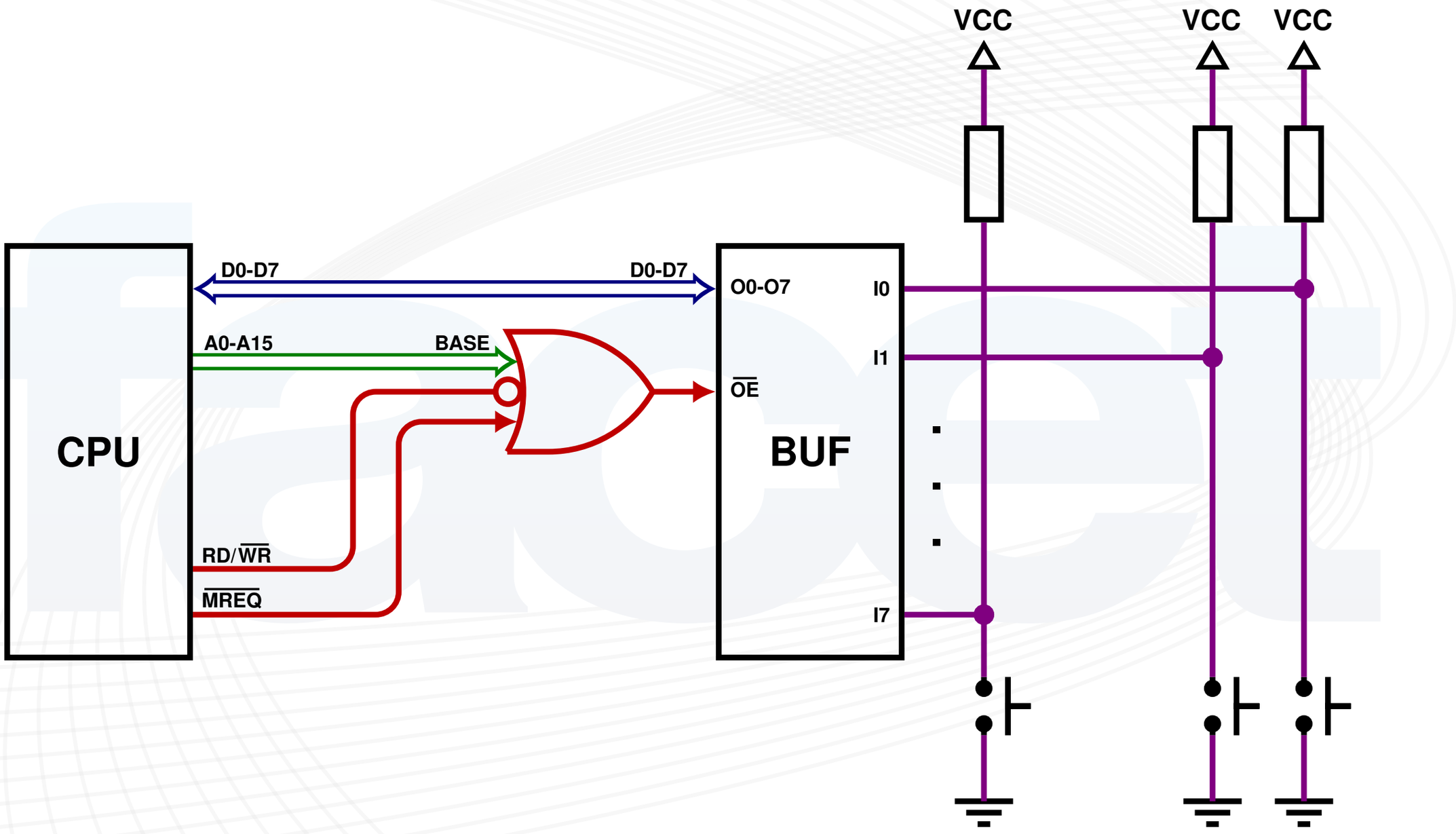
Salidas digitales en un CPU



Entradas digitales en un CPU

- Para leer el valor de entradas digitales desde un CPU se puede usar el bus de datos
- Sin embargo para conectar un dispositivo al bus es necesario que:
  - El dispositivo tenga salidas con capacidad de alta impedancia (3 State)
  - Las salidas deben permanecer en alta impedancia hasta que se realice una lectura en una dirección específica
- La solución es utilizar un buffer 3-State que se habilita en una dirección específica *BASE*

Entradas digitales en un CPU



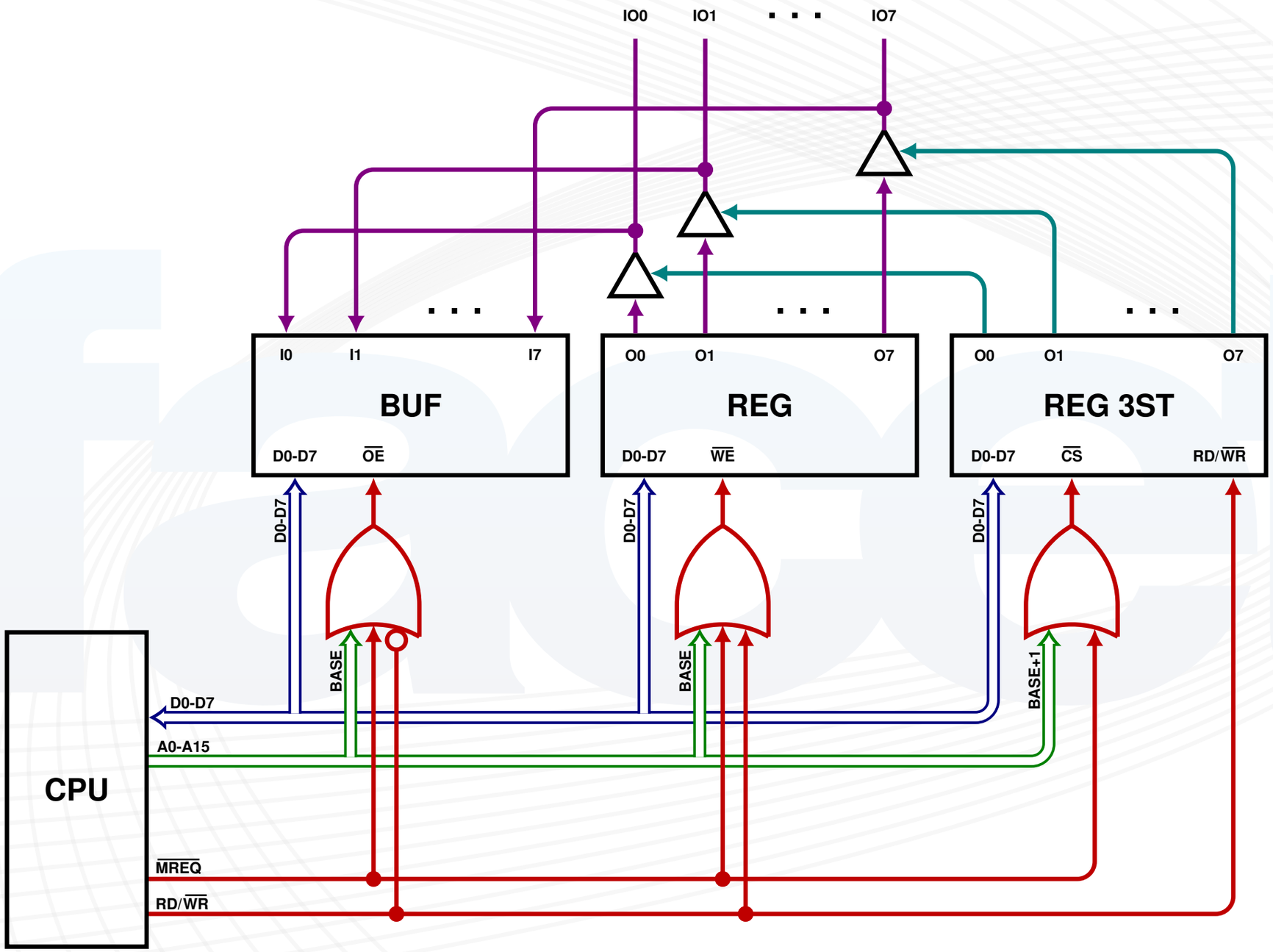
Entradas digitales en un CPU

- La entrada del buffer 3State debe tener siempre un valor definido
  - Cuando se conectan botones se requieren resistencias de pull-up (o de pull-down)
- Esta resistencia fija el valor de entrada cuando el botón está en reposo
  - Y limita la corriente cuanto el botón está activo
- Lo mismo sucede si se utiliza un dispositivo electrónico con salida a colector abierto

Entradas/Salidas digitales (GPIO)

- Cada linea se configura como entrada o salida en forma independiente
- Es una combinación de los dos circuitos anteriores
  - Se deben agregar buffers para conectar o desconectar las salidas de los registros de los terminales
  - Se debe agregar un registro para controlar estos buffers
- Este segundo registro es el que permite configurar la dirección como salida o entrada

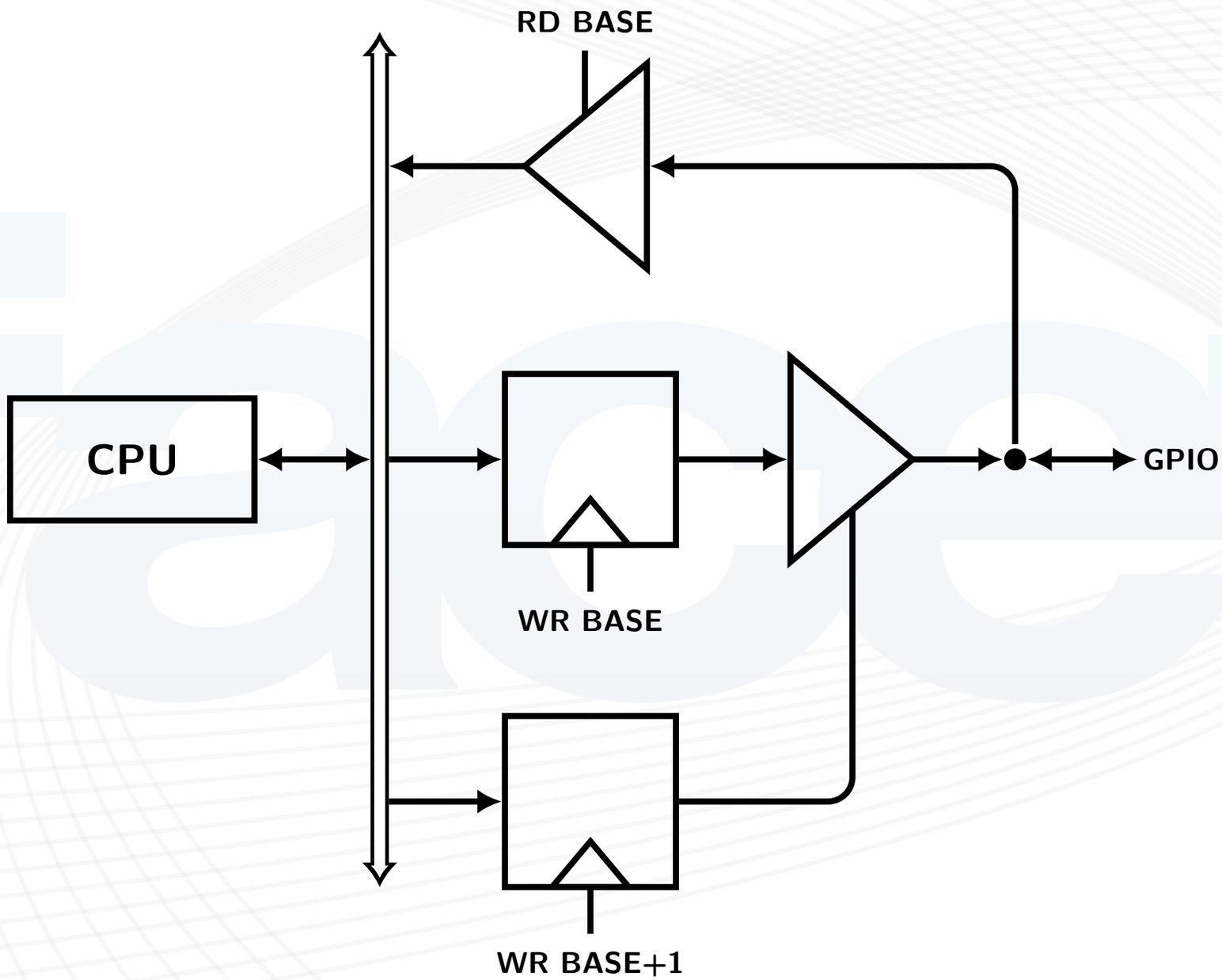
Entradas/Salidas digitales (GPIO)



Entradas/Salidas digitales (GPIO)

- Este dispositivo ocupa dos direcciones de memoria
- En la dirección BASE opera con las entradas o salidas
  - Se puede escribir el estado de las salidas
  - Se puede leer el estado de las entradas o de las salidas
- En la dirección BASE+1se configura la dirección de cada linea como entrada o salida
  - Si se escribe un 0 la línea opera como entrada
  - Si se escribe un 1 la línea opera como salida
  - Tambien se puede leer la configuración actual de las líneas

GPIO simplificado



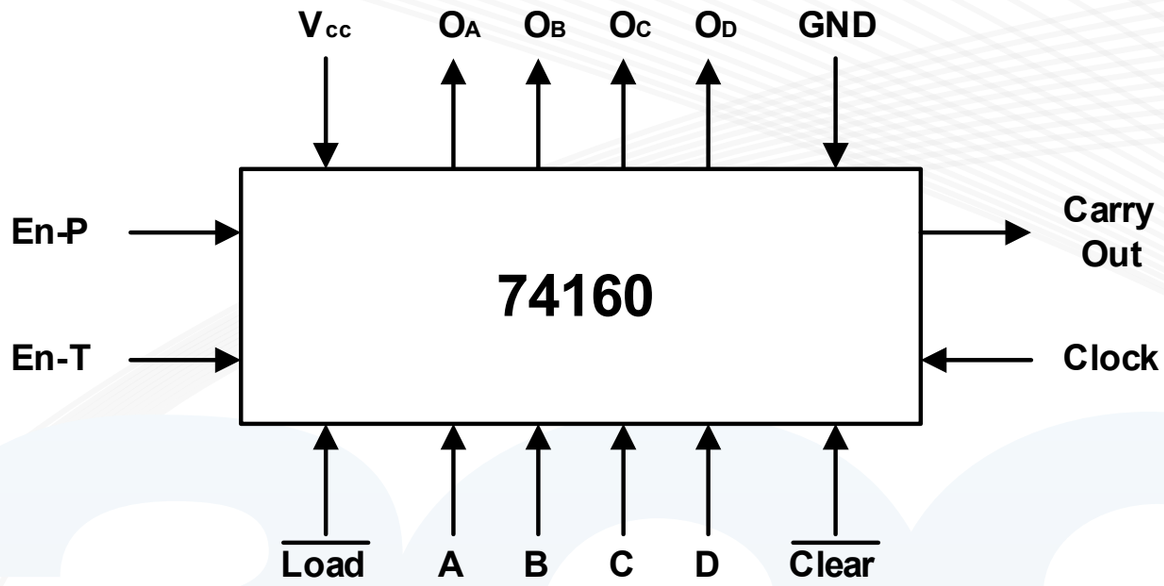
Dispositivos en un CPU

- Con entradas y salidas digitales se puede controlar cualquier dispositivo
- Por ejemplo se desea conectar un contador con las siguiente funcionalidades:
  - Seleccionar la fuente de reloj entre dos opciones
  - Iniciar o suspender la cuenta
  - Leer el valor actual de la cuenta
  - Cargar un valor inicial en el contador
  - Volver la cuenta a cero

Conexión de un contador a un CPU

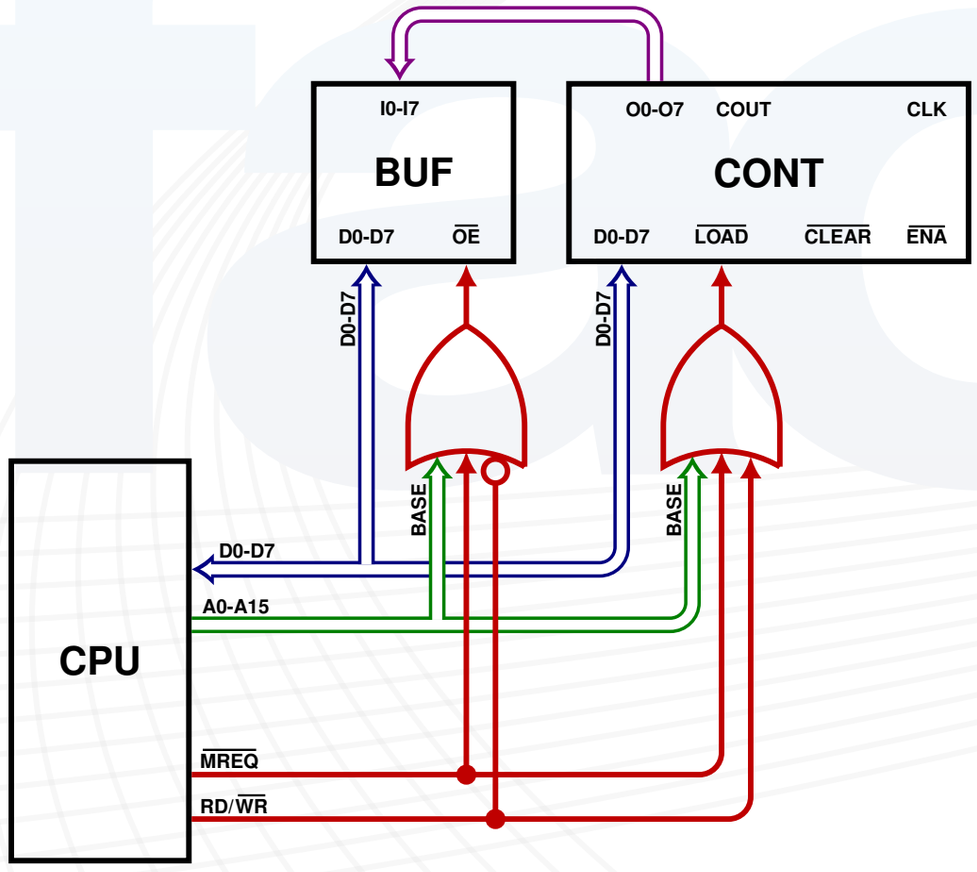
- La lectura y escritura de un valor en el contador es igual que en un registro
  - Se debe generar la señal de carga a partir de la decodificación de una dirección de memoria *BASE*
  - Se debe utilizar un buffer 3State para que las salidas del contador se lean desde el bus
    - Este buffer se habilita con la decodificación de la misma dirección *BASE* utilizada para la carga del contador
- Para el software es como una dirección de memoria normal que puede leerse y escribirse

Presentación de un Contador.



- Carga Paralelo
- Especificaciones para 74160, 61, 62, 63
  - Clear/Load sincrónico o asincrónico
  - Clock: flanco ascendente o descendente
- ¿Cómo se arma un contador de 8 bits?

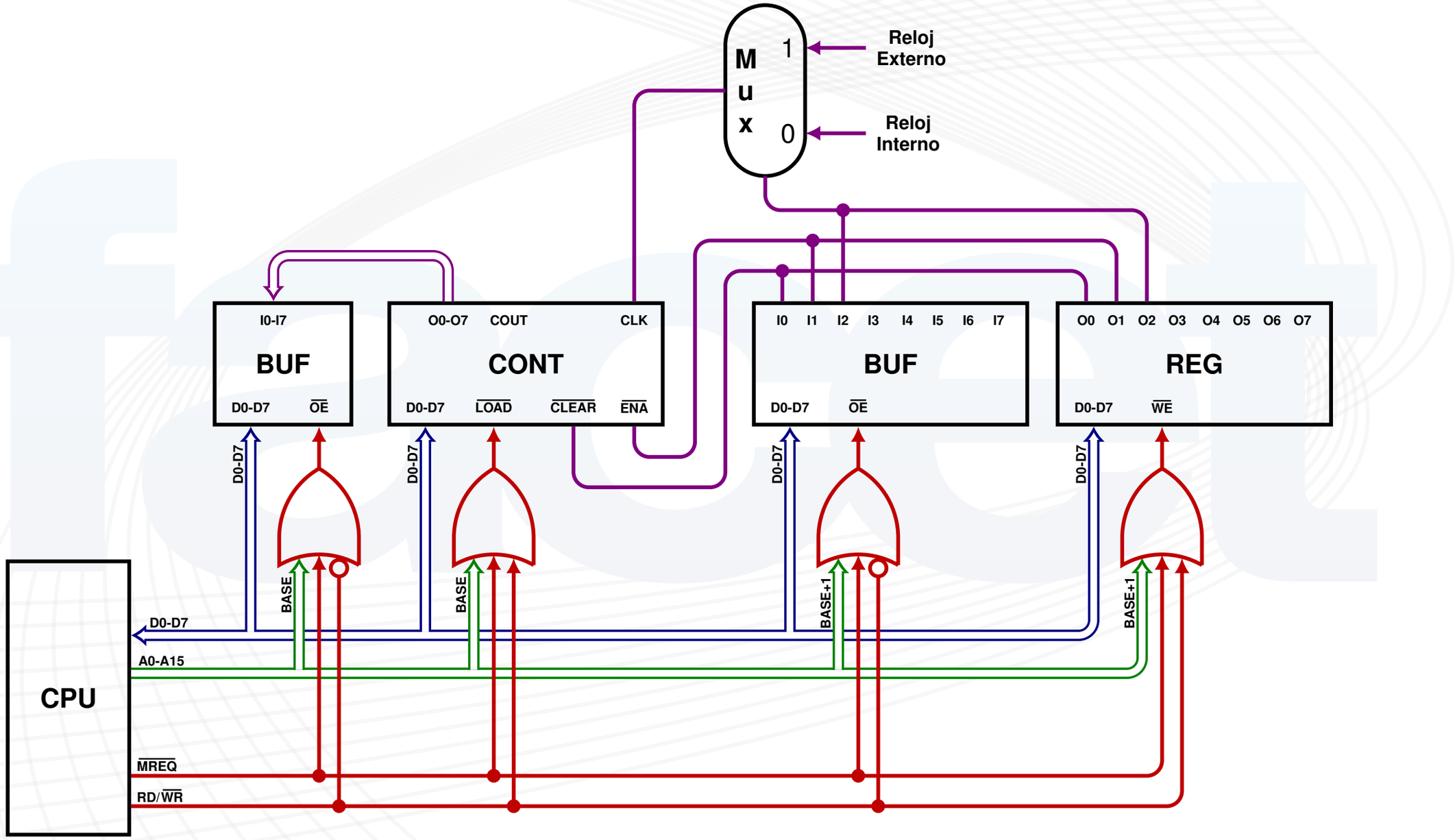
Conexión de un contador a un CPU



Conexión de un contador a un CPU

- Para manipular las señales de control del contador se necesitará un registro adicional
  - Este registro se mapea en una segunda dirección de memoria *BASE + 1*
- Se utilizarán señales de este registro para activar las señales de borrado y habilitación del contador
- Se agregará un multiplexor para seleccionar la fuente de reloj del contador
  - Se controla tambien con una señal del nuevo registro
- Al registro se le agrega un buffer para poder leerlo

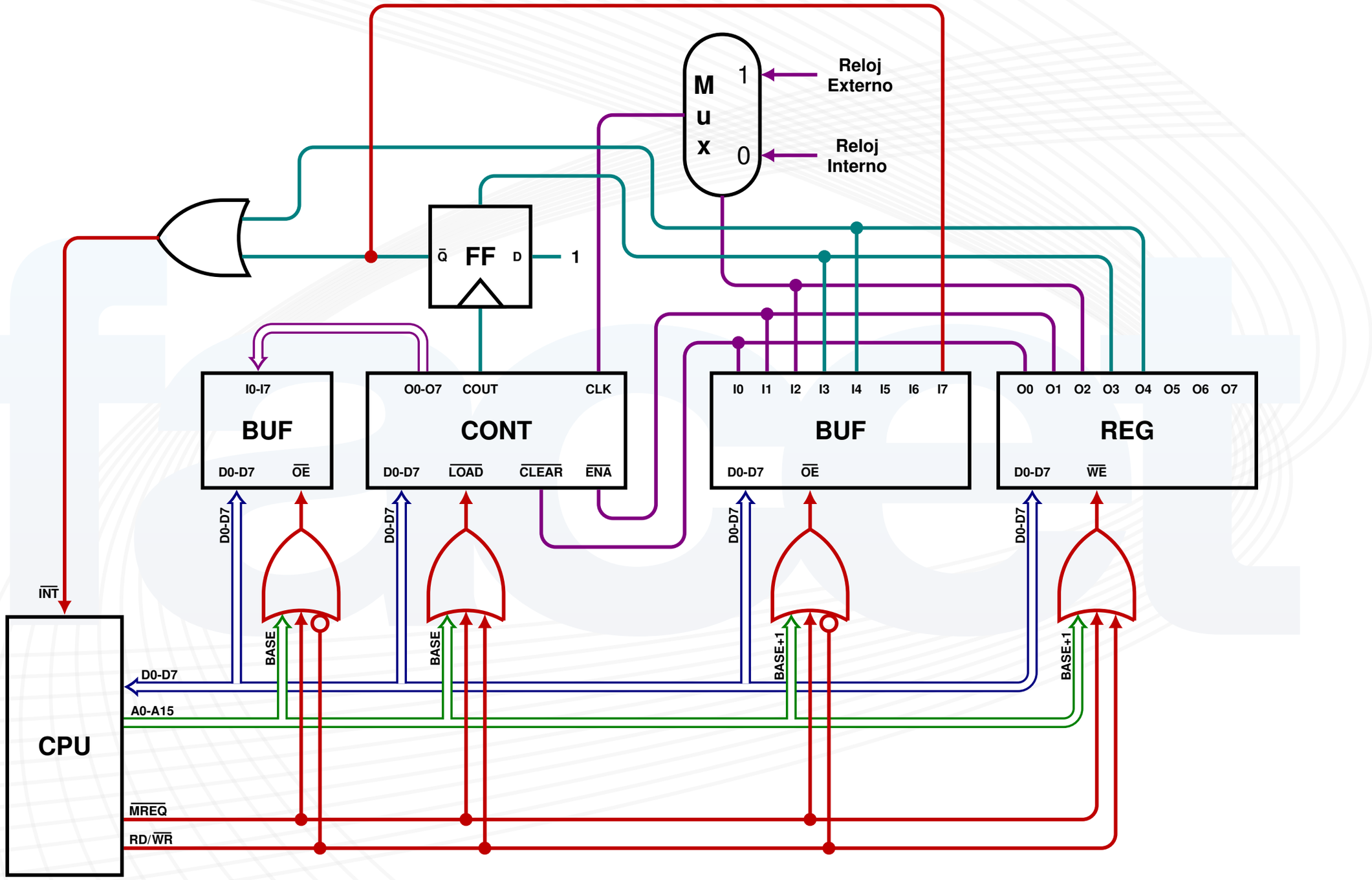
Conexión de un contador a un CPU



Conexión de un contador a un CPU

- Agregar al contador la capacidad para generar un pedido de interrupción en el CPU
  - El procesador tiene una entrada específica para este propósito
- El pedido de interrupción se debe generar cuando el contador alcance la cuenta máxima
  - Se utiliza la señal carry out del contador
  - Se debe memorizar hasta que el pedido sea atendido
- Se debe poder consultar si hay una interrupción pendiente, borrarlo y enmascararlo por software

Conexión de un contador a un CPU



Conexión de un contador a un CPU

- El contador ocupa dos direcciones de memoria
  - En la dirección BASE se encuentra el registro de datos
  - En la dirección BASE + 1 se encuentra el registro de control y estado
- Se puede operar sobre el contador con operaciones de lectura y escritura en estas direcciones
  - Sin embargo conocer estas direcciones no es suficiente para escribir un programa adecuado
  - Es necesario conocer la función de cada uno de los bits del registro de control y estado

Registro de control y estado

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STATUS	N/I	N/I	MASK	ACK	CLOCK	ENABLE	CLEAR

- CLEAR: bit de borrado
  - CLEAR = 0 → El contador mantiene su valor
  - CLEAR = 1 → El contador se borra asincronicamente
- ENABLE: bit de habilitación de cuenta
  - ENABLE = 0 → El contador está detenido
  - ENABLE = 1 → El contador está habilitado y contando
- SEL_CLK: bit de selección de clock
  - SEL_CLK = 0 → El contador opera con un clock interno
  - SEL_CLK = 1 → El contador opera con un clock externo

Registro de control y estado

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STATUS	N/I	N/I	MASK	ACK	CLOCK	ENABLE	CLEAR

- ACK: bit de reconocimiento del evento de overflow
  - ACK=1 → Borra el pedido bit de STATUS y el pedido de interrupción
  - ACK=0 → Se memoriza el evento de overflow en el bit de STATUS
- MASK: bit de máscara de interrupción por overflow
  - MASK = 1 → Las interrupciones están inhabilitadas
  - MASK = 0 → Las interrupciones están habilitadas
- STATUS: bit de status de overflow
  - STATUS = 0 → No ocurrió overflow en el contador
  - STATUS = 1 → Ocurrió un overflow en el contador

Conclusiones

- Se conectaron dispositivos de entrada/salida al procesador como si fueran memoria
  - Para el programa no hay diferencia
- Un dispositivo tiene más de un tipo de registro
  - Registros de datos, de control y de estado
- Se necesita conocer la dirección de memoria y la función de cada bit de estos registros
  - El mapa de memoria tiene mas detalle cuando se trata de los dispositivos de entrada/salida