Estructura de un microcontrolador

Sistemas con Microprocesadores

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

http://microprocesadores.unt.edu.ar/procesadores

Cronograma

Actividad	Inicio	Descripción	Fin
Presentación	19/08	Reglamento de la Materia	✔
Tema 1	19/08	Estructura de las computadoras	✔
Tema 2	26/08	Proyecto con un microcontrolador	✔
Tema 3	30/08	Descripción funcional de microprocesador	✔
Tema 4	13/09	Programación en lenguaje ensablador	✔
Tema 5	25/09	Descripción general de un microcontrolador	✔
Tema 6	27/09	Estructura general de microcontrolador	←
Parcial	09/10	Primer examen parcial	
Tema 7	14/10	Sistema de Interrupciones	
Tema 8	21/10	Entradas y salidas digitales	
Tema 9	28/10	Entrada/salida con perifericos	
Tema 10	06/11	Temporizadores	
Proyectos	25/11	Seminarios de Proyectos	
Parcial	04/12	Segundo examen parcial	

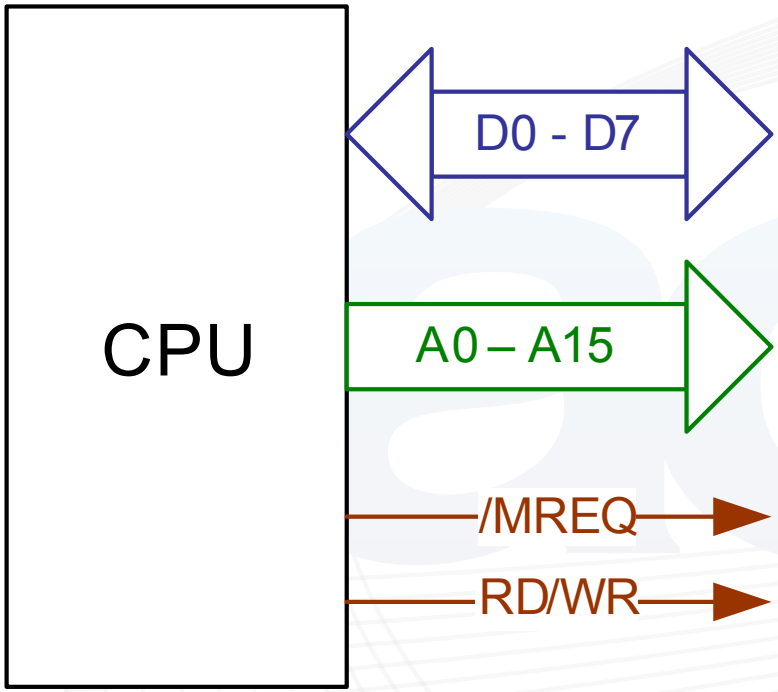
Desarrollo Conceptual

- Se empleará un Microprocesador de 8 bits, llamado CPU08
- Data bus – 8 bits.
- Address bus – 16 bits.
- ¿Máx cantidad de memoria posible?

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.

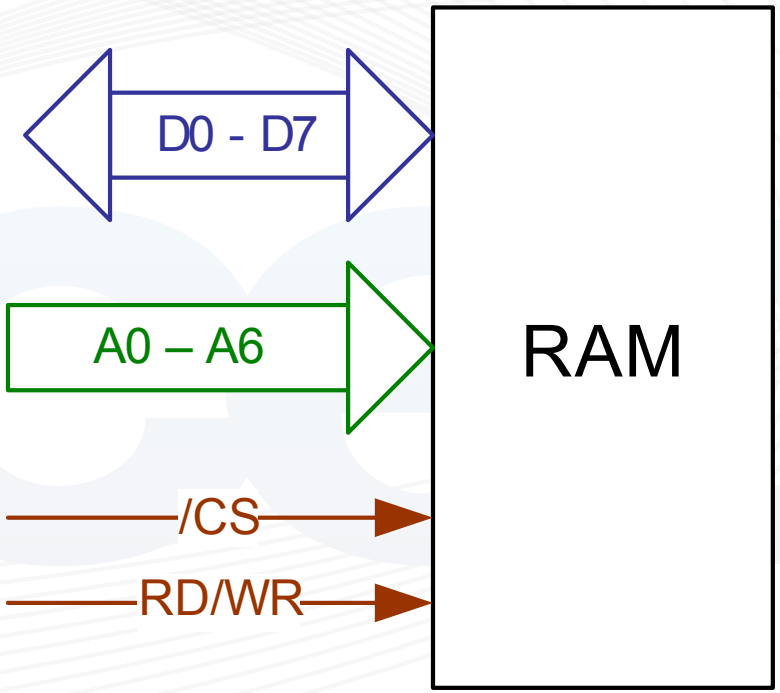
Entradas y Salidas del Procesador



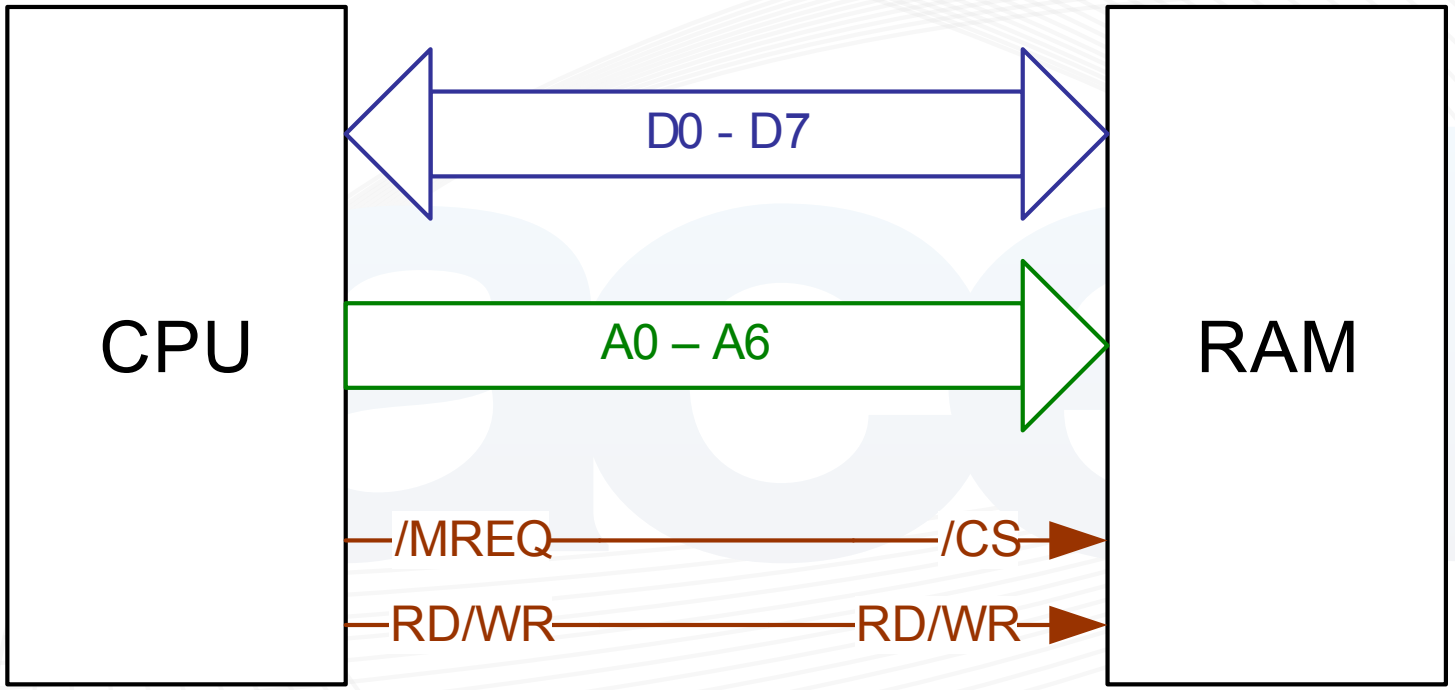
Entradas y Salidas de la Memoria

- 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.

Entradas y Salidas de la Memoria

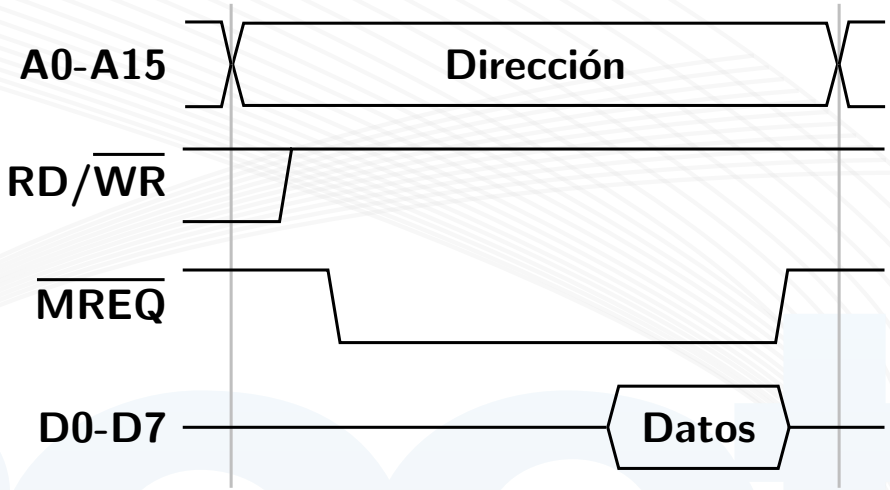


¿Como se conectan?

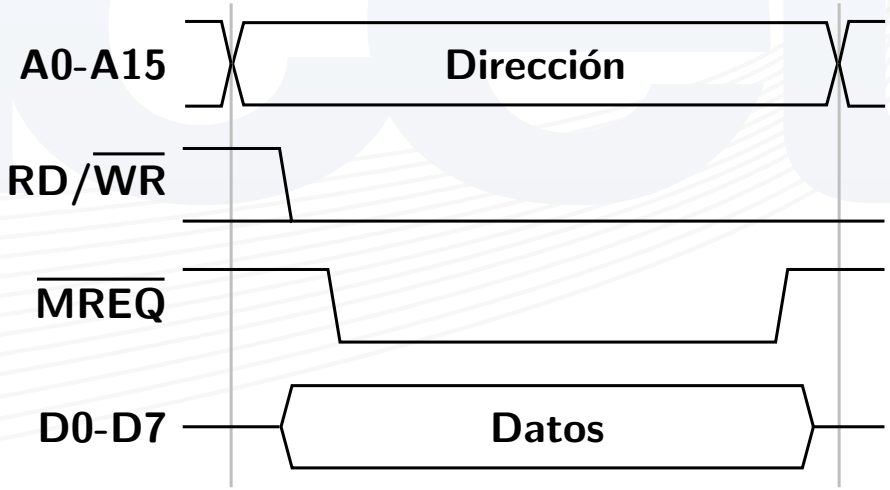


Diagramas de tiempos típicos

Lectura



Escritura



Mapas de Memoria

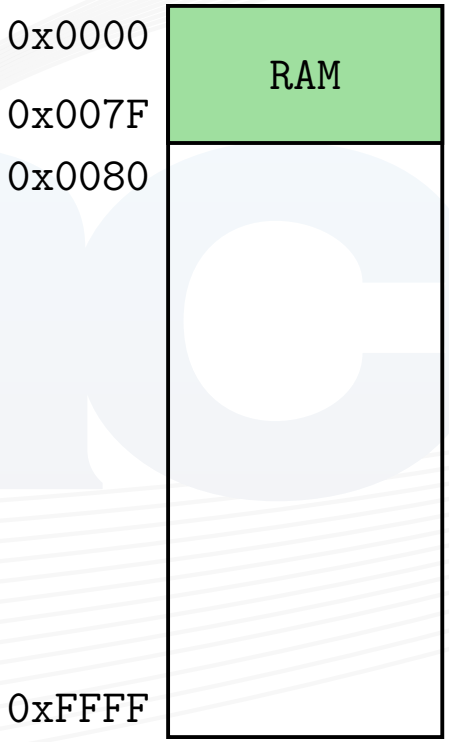
- Se llama espacio de direcciones al conjunto de todas las direcciones posibles a las que podría acceder un CPU.
- En el caso del CPU08 es de 65.536 o 64Ki direcciones.
- No siempre se usa la totalidad del espacio disponible, puede no hacer falta.
- Los mapas de memoria muestran las partes del espacio 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

¿Que sucede al acceder a la dirección 0x0000?

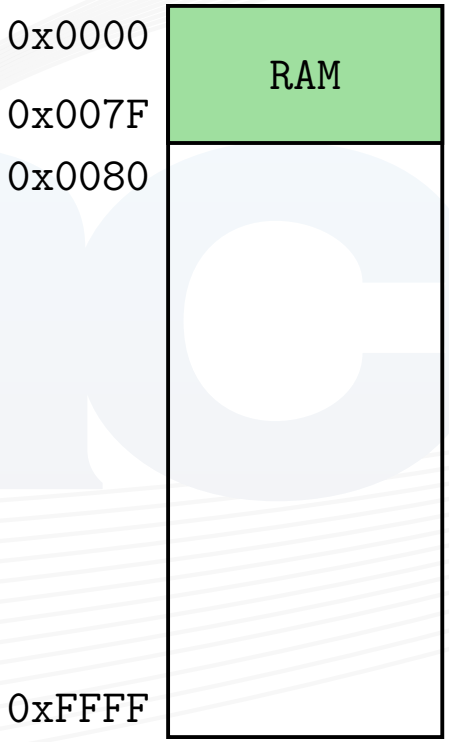
Ejemplo de Aplicación

¿Que sucede al acceder a la dirección 0x0000?



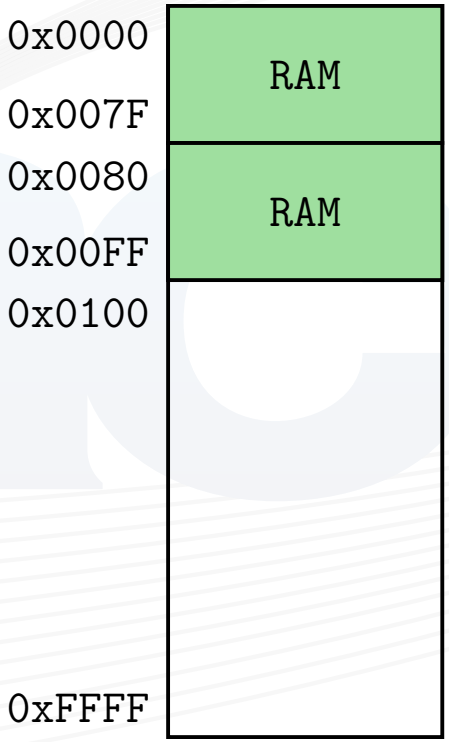
Ejemplo de Aplicación

¿Y al acceder a la dirección 0x0080?



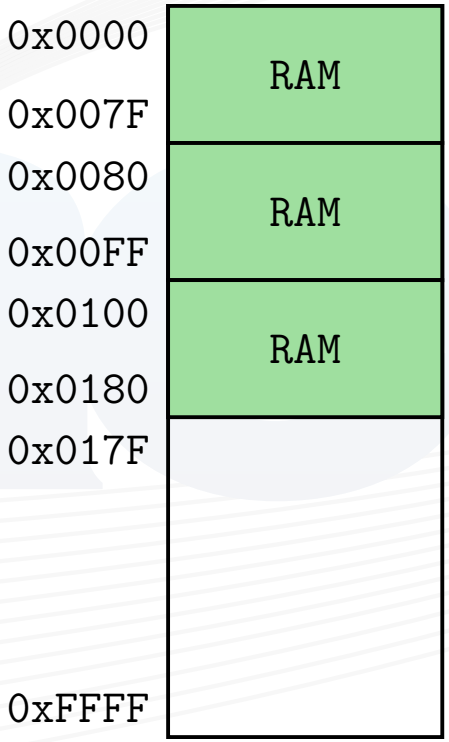
Ejemplo de Aplicación

A7 a A15 no están conectadas a la RAM



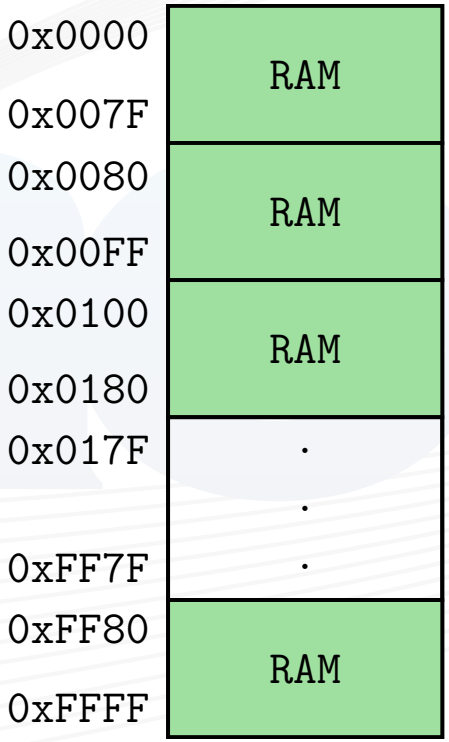
Ejemplo de Aplicación

¿Y en la dirección 0x0100?



Ejemplo de Aplicación

Y así hasta el final de espacio...

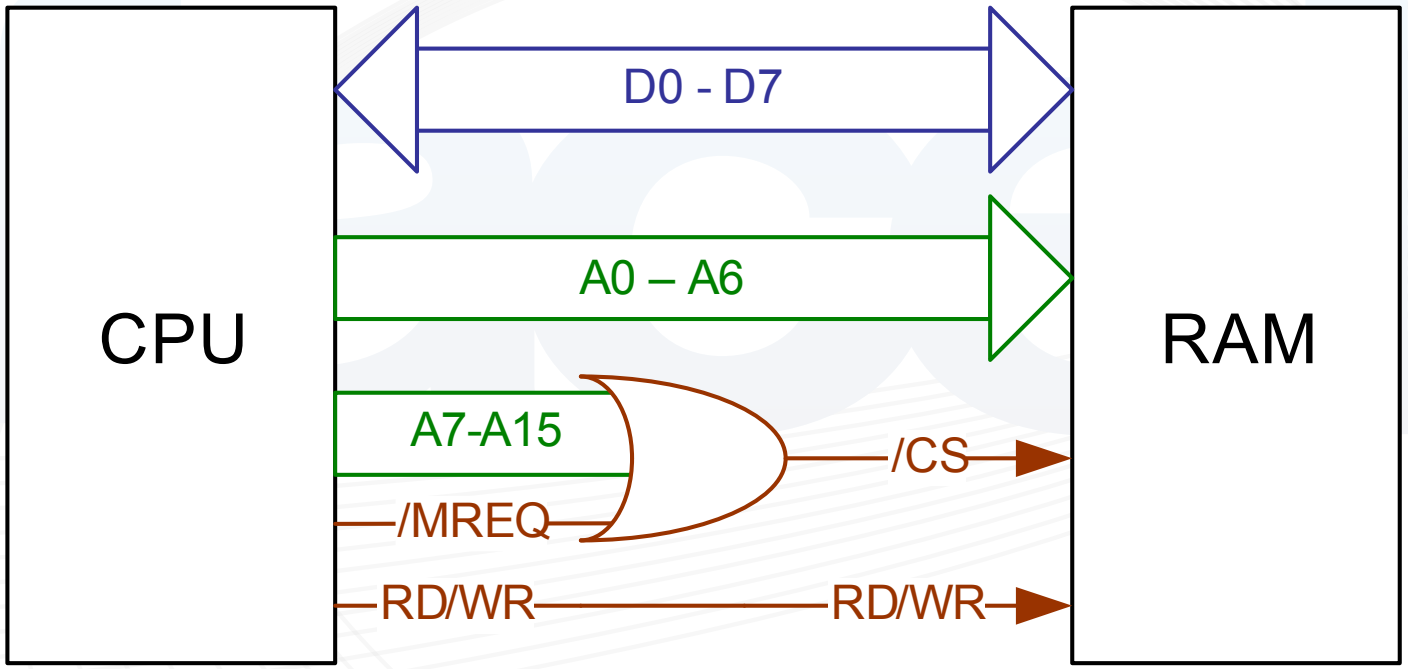


Conexión Redundante (100%)

- 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 0x0000 a 0x007F – sin redundancia…

Conexión sin Redundancia

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



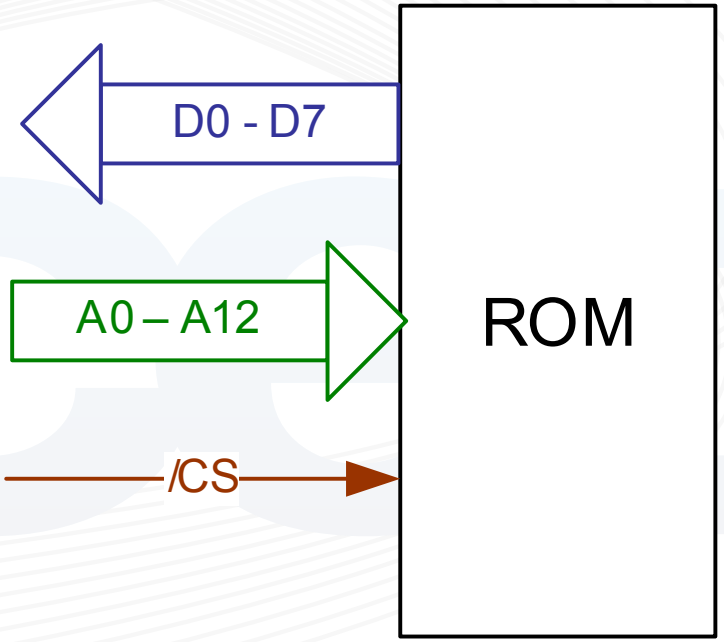
Conexión sin Redundancia

- 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.

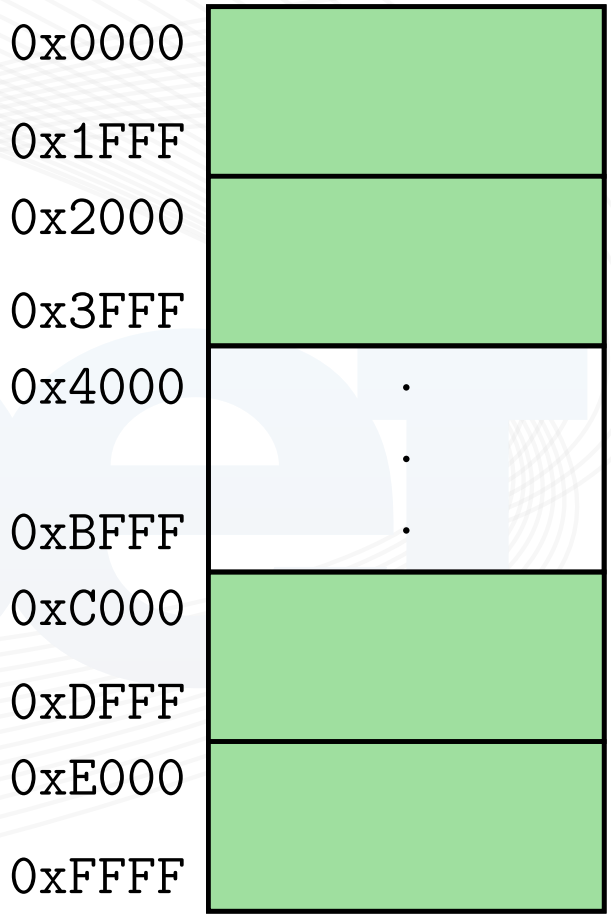
- Conexiones que tiene una memoria ROM de 8K x 8:
  - Líneas de Datos: Depende de cuántos bits se almacena en cada dirección, en nuestro caso 8 líneas, de D0 a D7.
  - Líneas de Direcciones: Depende de cuántas direcciones tenga la memoria, en nuestro caso 13 líneas, de A0 a A12.
  - Selección: Una línea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

Entradas y Salidas de la Memoria



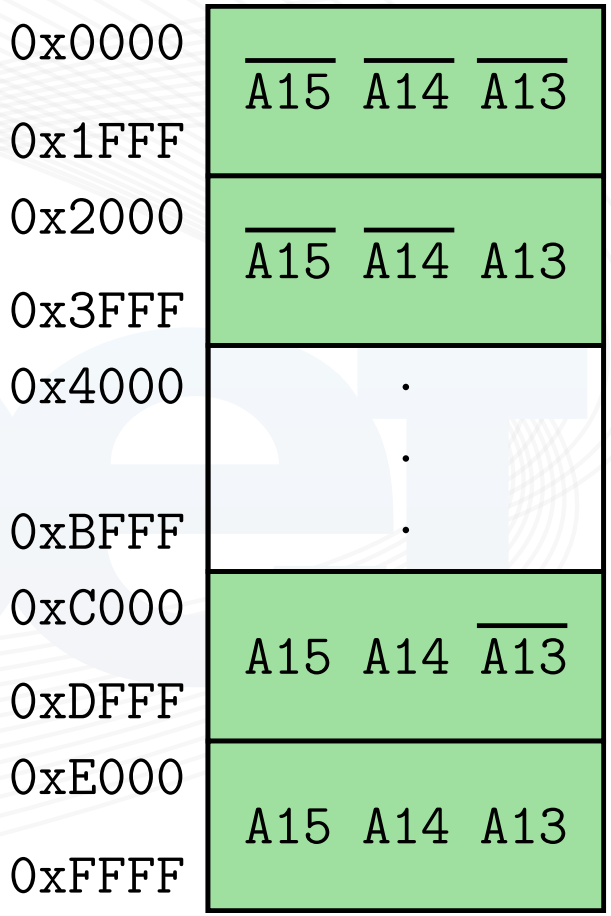
División del Mapa de Memoria

- Dado que nuestro ROM es de 8K, dividimos el mapa de memoria en fracciones de 8K.



División del Mapa de Memoria

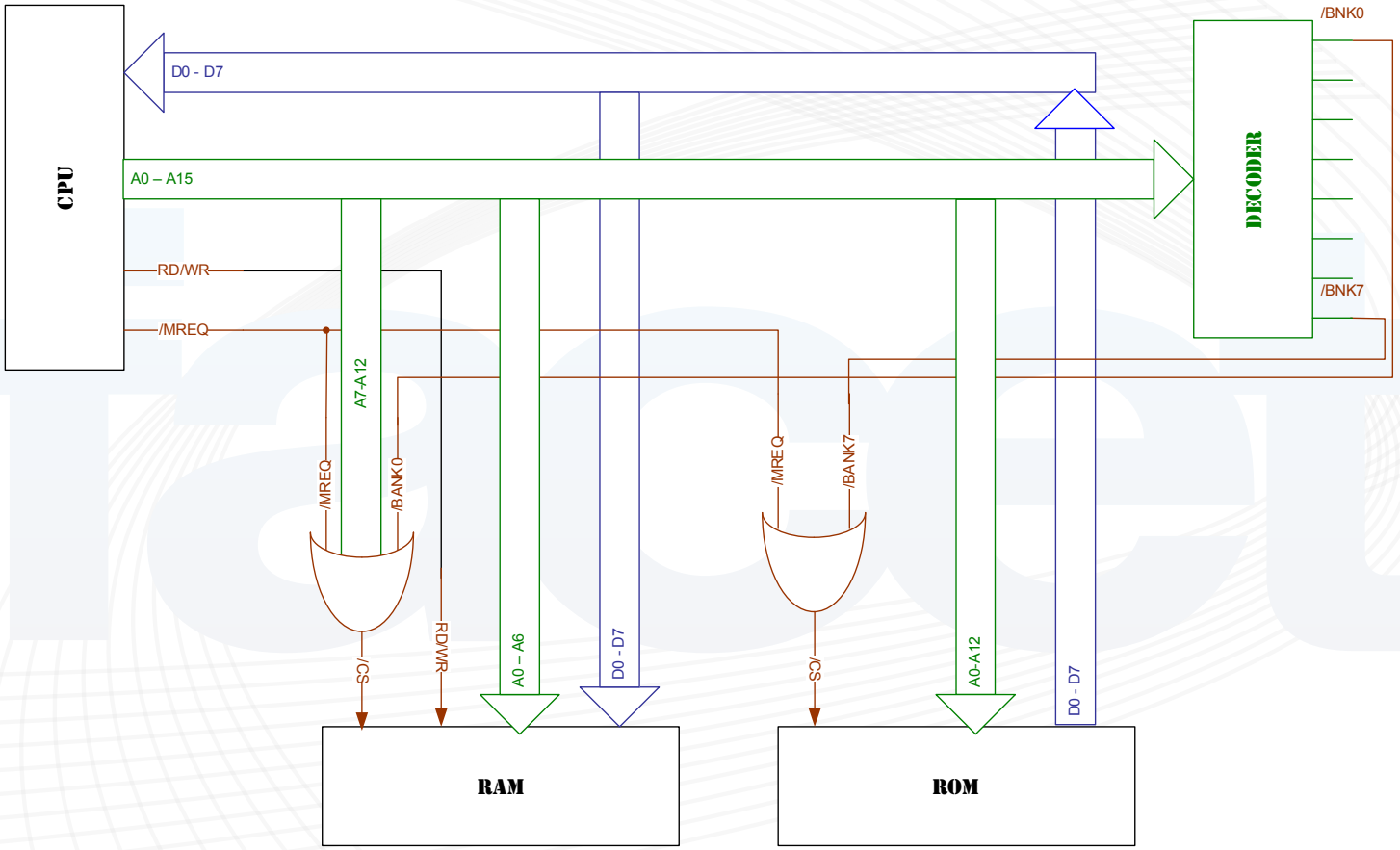
- Dado que nuestro ROM es de 8K, dividimos el mapa de memoria en fracciones de 8K.
- Cada fracción se caracteriza por una combinación de valores de las 3 líneas superiores de direcciones



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

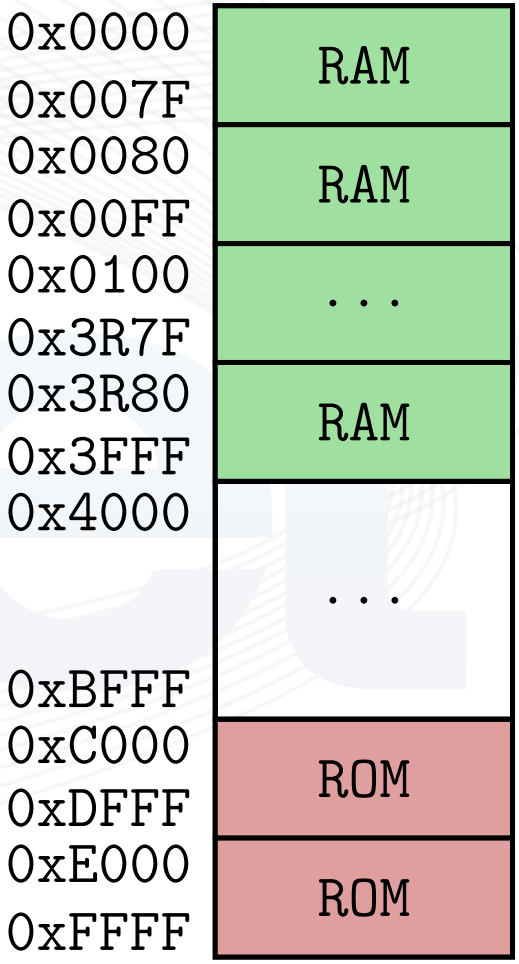


Posibles mejoras

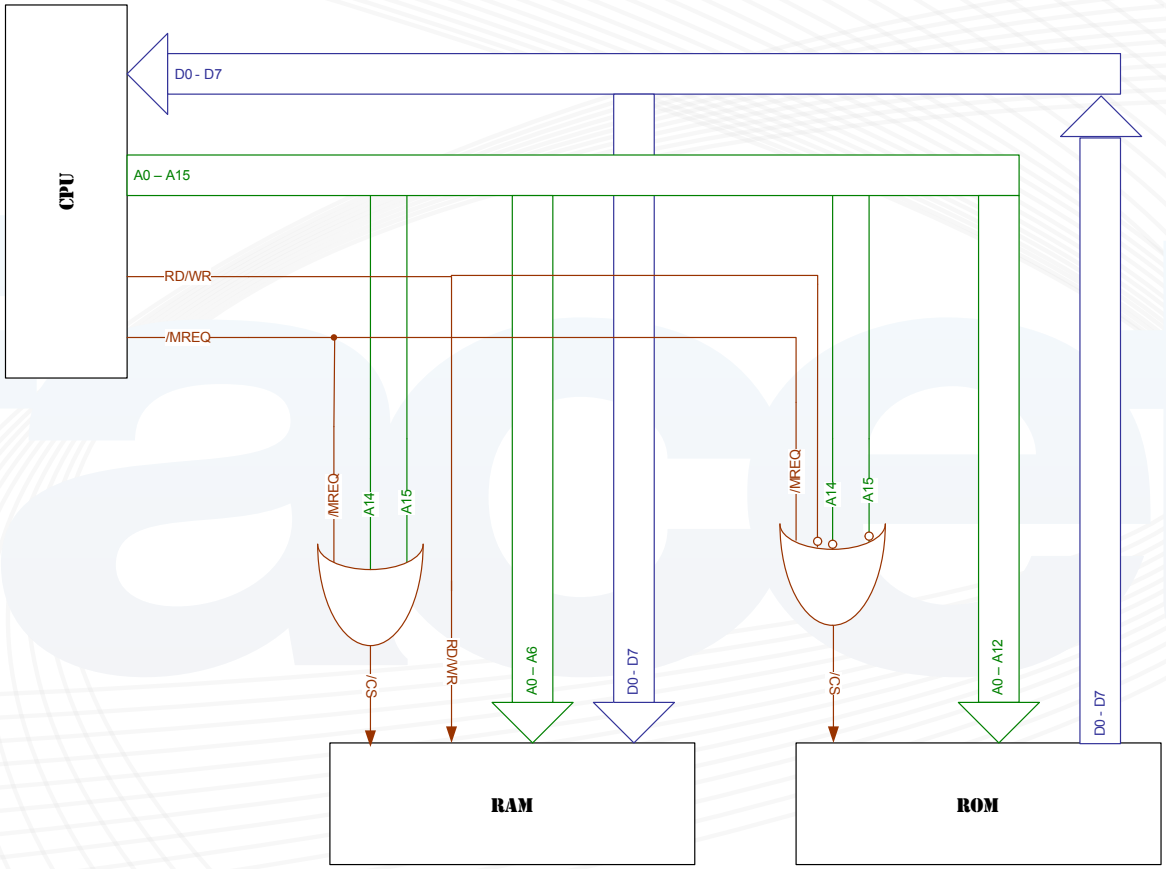
- /MREQ, mejor como habilitación del DECO
- Un error de programa quemaría el CPU y la ROM.
  - ¿Qué instrucción provoca esto?
  - ¿Forma de Evitarlo?

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



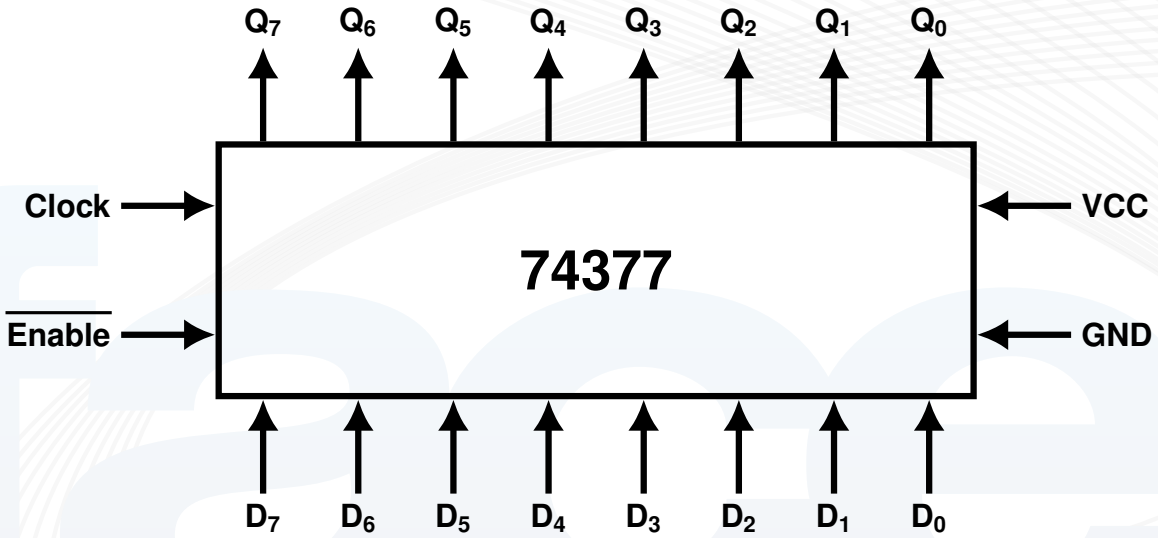
Conexión con Redundancia



Conexión con Redundancia

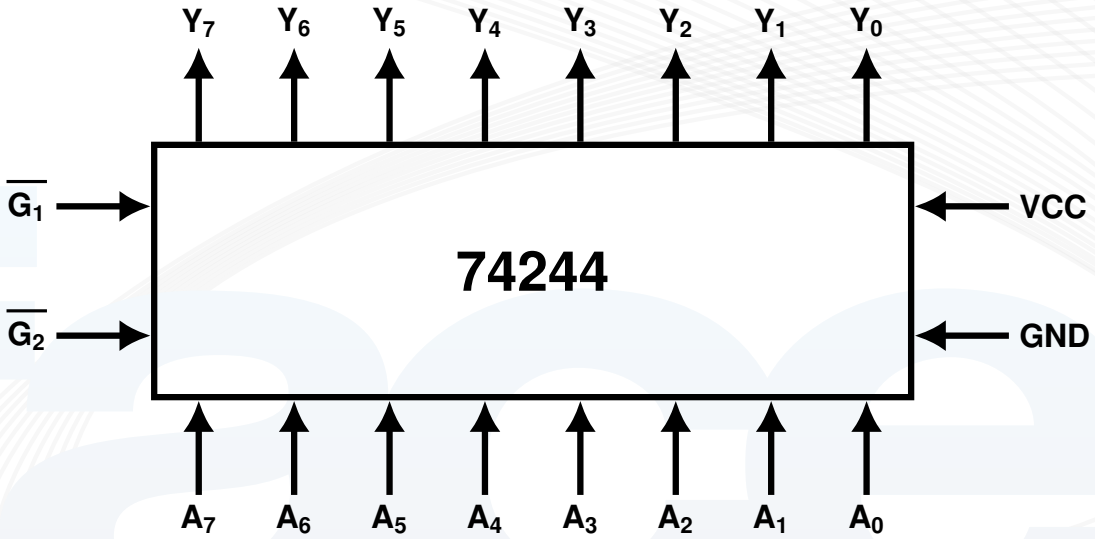
- Memoria Conectada Real = MR
- Memoria Ocupada en el Mapa = MO
- ¿Cuánto vale cada una en ej. anterior?
- En general
  - MR <= MO
  - La igualdad es cuando no hay redundancia.
  - La redundancia permite economizar compuertas.
  - Con la seguridad de que no se requiere el espacio redundante en el futuro.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

Presentación de un Registro



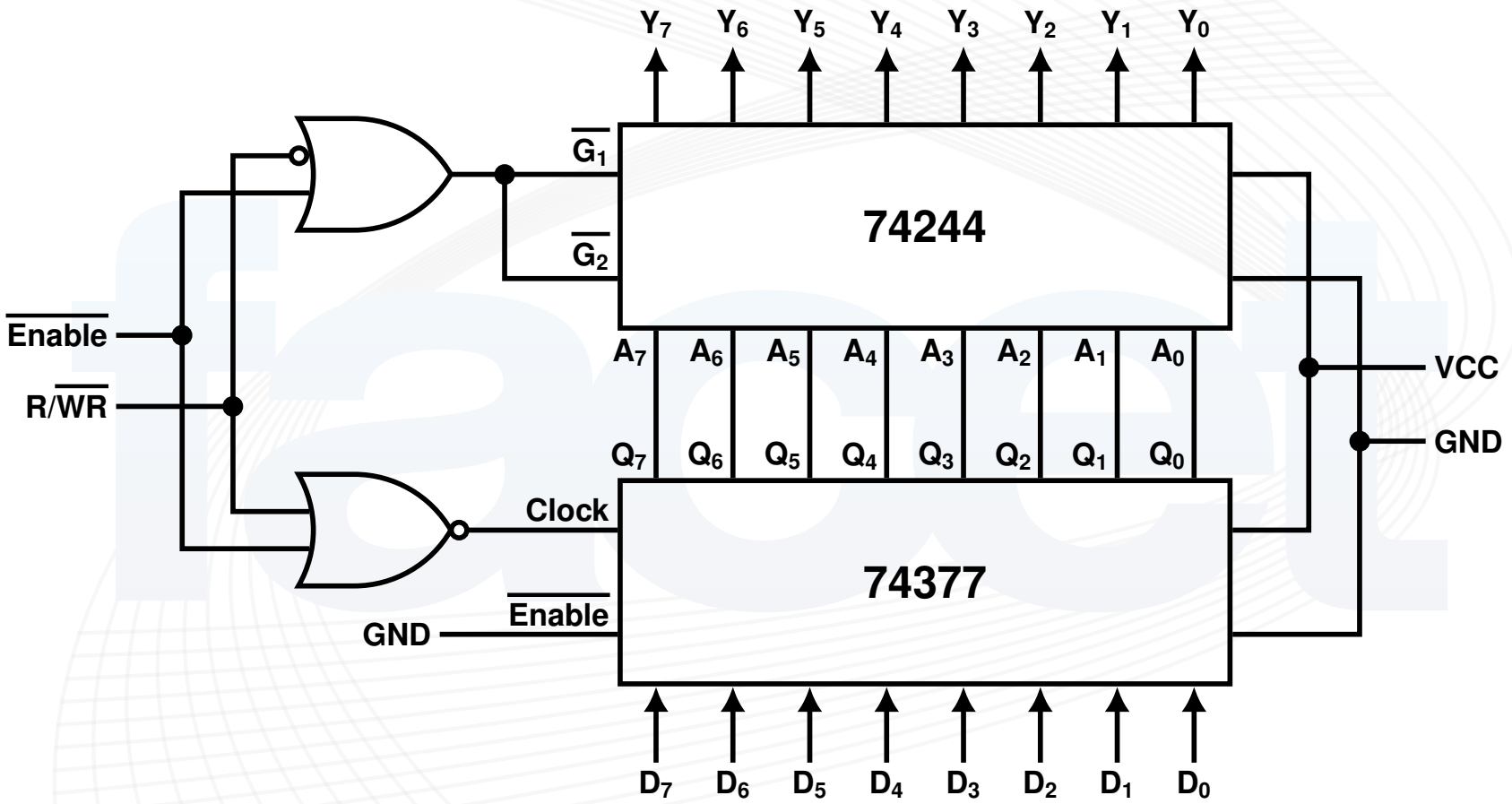
- Almacena los valores de las entradas D_i cuando se produce un flanco ascendente del Clock y se activa Enable.

Presentación de un Buffer 3State

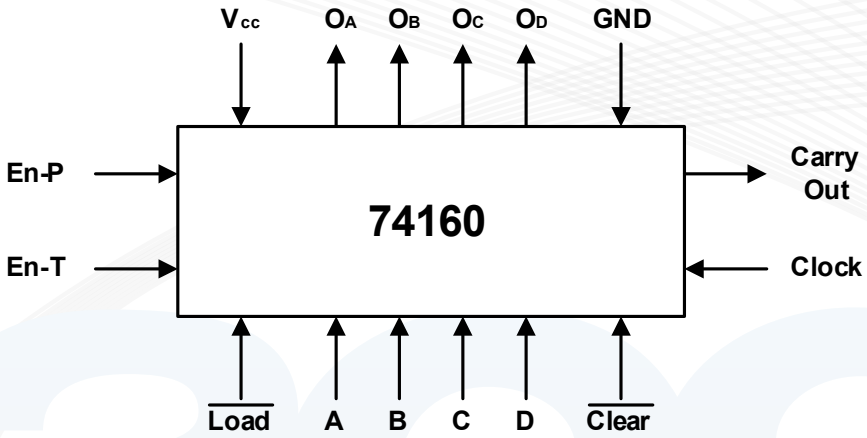


- Cuando los Gates están en alto las salidas permanentes en alta impedancia, y cuando están en bajo las salidas toman el mismo valor de las entradas correspondientes

Conexión de un registro al Bus



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 al CPU

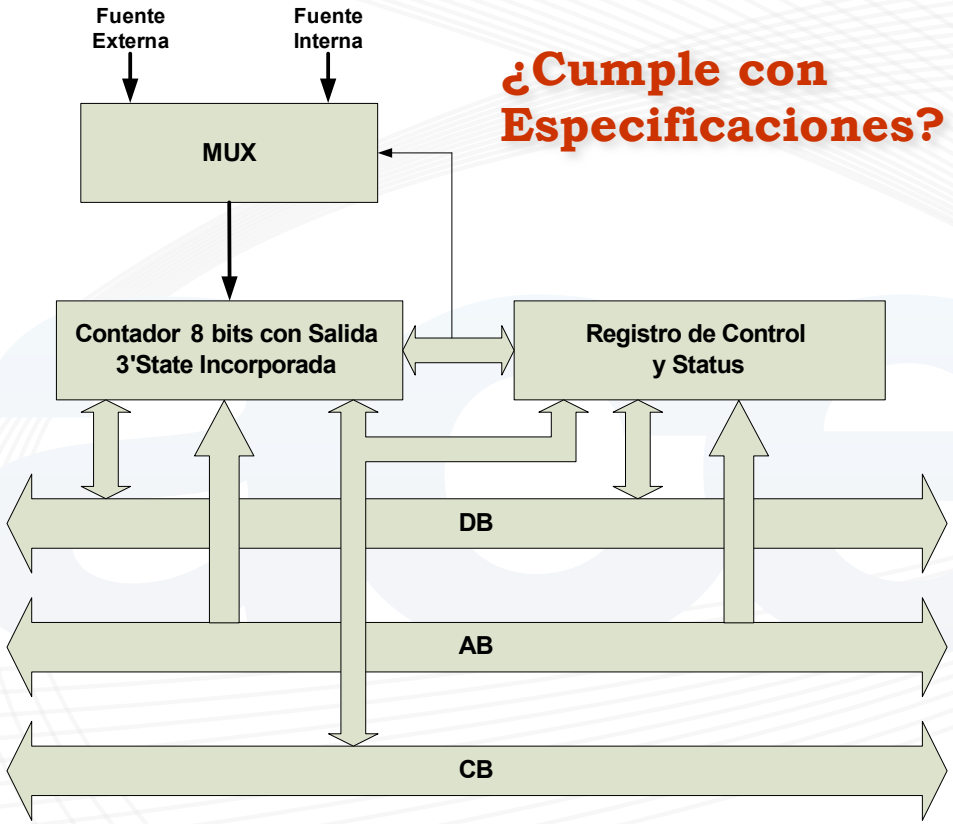
Problema: Se requiere que el CPU sea capaz de:

- Cargar un valor inicial en contador.
- Habilitar la cuenta a partir de un clock externo o interno.
- Inhabilitar la cuenta.
- Leer un valor.
- Poner el contador a cero.
- Con Overflow del Contador
  - Indicar este evento en un bit de estado.
  - Pedir INT al CPU (IRQ – Int. Request).
  - Con una señal ACK cesa pedido de INT y estado vuelve a normal.
- Enable/Disable IRQ solo para este contador.

Del Problema a los Requisitos.

- El CPU tiene una entrada /IRQ
- Para conectar el Contador se requiere una interfase con buffers.
- Se requiere un Registro de Control y Estado (RCS).
  - Para dar señales de control al Contador.
  - Un bit es un FF para informar de OVF=1.
- El registro adicional debe ser accedido por el CPU.
- Además un MUX que seleccione el clock externo o interno.

Conexión de un Contador

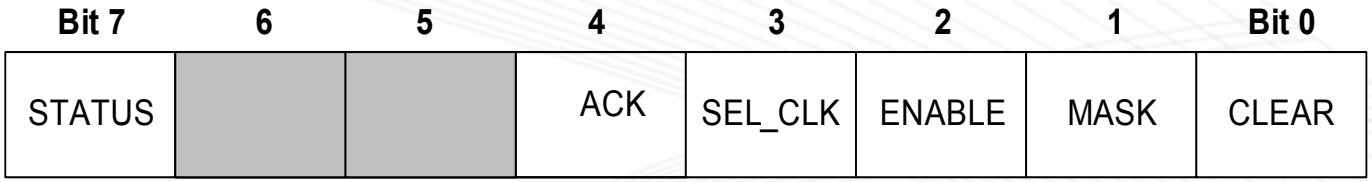


¿Cumple con Especificaciones?

Contador

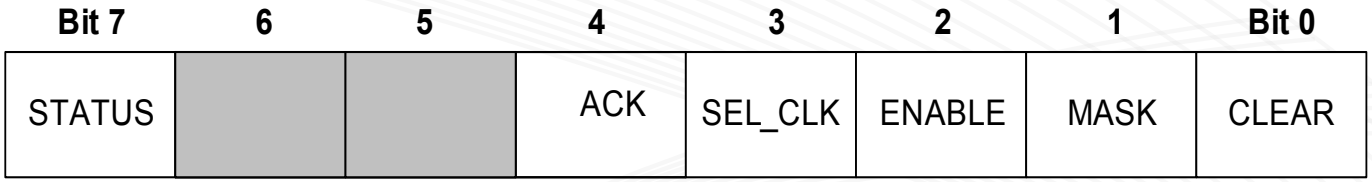
- ¿Cuántas Direcciones Ocupa?
  - Reg de Control y Status – 1 dirección.
  - Contador – 1 dirección.
- ¿Cuántos Bits Tiene el Reg. de Control y Status?
  - Bits: CLR, ENABLE, CLK_SEL, STATUS, MASK, ACK (6 bits).
    - Por conveniencia Status es un FF separado.
  - Se puede operar sobre el contador y el registro con operaciones de lectura y escritura de memoria.

Registro de Status y Control



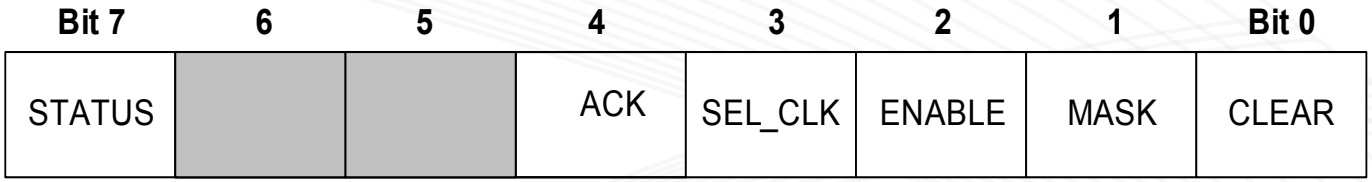
- ACK: bit de reconocimiento del evento OVF.
  - ACK=1 → Status=0 (conectado a CLEAR de Status).
- SEL_CLK: bit de selección de clock
  - SEL_CLK = 0 → El contador opera con un clock interno.
  - SEL_CLK = 1 → El contador opera con un clock externo.
- STATUS: bit de status de overflow
  - STATUS = 0 → No ocurrió overflow en el contador.
  - STATUS = 1 → Ocurrió un overflow en el contador.

Registro de Status y Control



- CLEAR: bit de borrado
  - CLEAR = 0 → El contador mantiene su valor.
  - CLEAR = 1 → El contador se borra asincronicamente.
- MASK: bit de máscara de interrupción por overflow
  - MASK = 1 → Las interrupciones están inhabilitadas.
  - MASK = 0 → Las interrupciones están habilitadas.
- ENABLE: bit de habilitación de cuenta.
  - ENABLE = 0 → El contador está detenido
  - ENABLE = 1 → El contador está habilitado y contando.

Registro de Status y Control

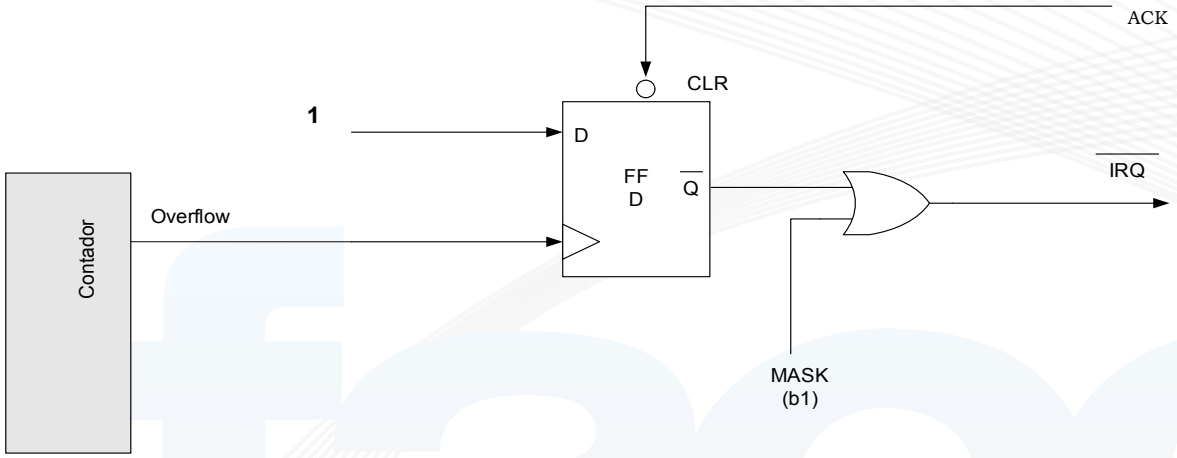


- Los bits ACK, SEL_CLK, ENABLE, MASK y CLEAR son de solo escritura; STATUS es de solo lectura.
- Los bits 5 y 6 no están implementados.

Lógica para IRQ.

- Un FF-D=1 (Status) cuando OVF=1
- Si bit en RCS: MASK=0 permite pedir INT.
- El FF-D puede chequearse por
  - Polling por Progr. Principal.
  - Pedido de Interrupción.
- Cuando el CPU reacciona al evento escribe ACK=1:
  - Quita el pedido de INT.
  - Pone FF-D=0 (para que registre un próx OVF).
  - A continuación se debe escribir ACK=0, ¿por qué?
- ACK es un bit RCS conectado al CLR FF.

Detección del overflow



D	/CLR	CLK	Q
0	1	Asc.	0
1	1	Asc.	1
X	0	X	0

- Para mantener la señal IRQ cuando hay overflow: FF“D”.
  - Un FF mantiene la señal para la interrupción y funciona como STATUS.
  - Para el resto de los bits se usa un registro.

La Imagen Completa.

- Conectar un Registro y un Contador a un sistema que requiere RAM de 128 Bytes y ROM de 8K.
- Dibujar el mapa de memoria.
- Dibujar el Circuito.
- Escribir las siguientes rutinas:
  - Arrancar el contador (fuente externa).
  - Parar el contador y leerlo.
- RCS, CONTADOR y TIEMPO
- Como ejercicio programa ISA ARM

Rutinas de manejo del contador

```
/*      RCS                 Dirección del registro
        CONT                Dirección del contador
        TIEMPO              Resultado contador en RAM        */

START:  LDR  R0,=RCS        // Apunto al registro de control
        MOV  R1,#0x01
        STRB R1,[R0]        // Borrar contador y deshabilitar reloj
        MOV  R1,#0x0C
        STRB R2,[R0]        // Reloj externo habilitado
        BX   LR

STOP:   LDR  R0,=RCS        // Apunto al registro de control
        MOV  R1,#0x08
        STRB R1,[R0]        // Reloj externo deshabilitado
        LDR  R0,=CONT       // Apunto al contador
        LDRB R1,[R0]        // Leo del valor del contador
        LDR  R0,=TIEMPO     // Apunto a la variable resultado
        STRB R1,[R0]        // Almaceno el valor en la variable
        BX   LR
```

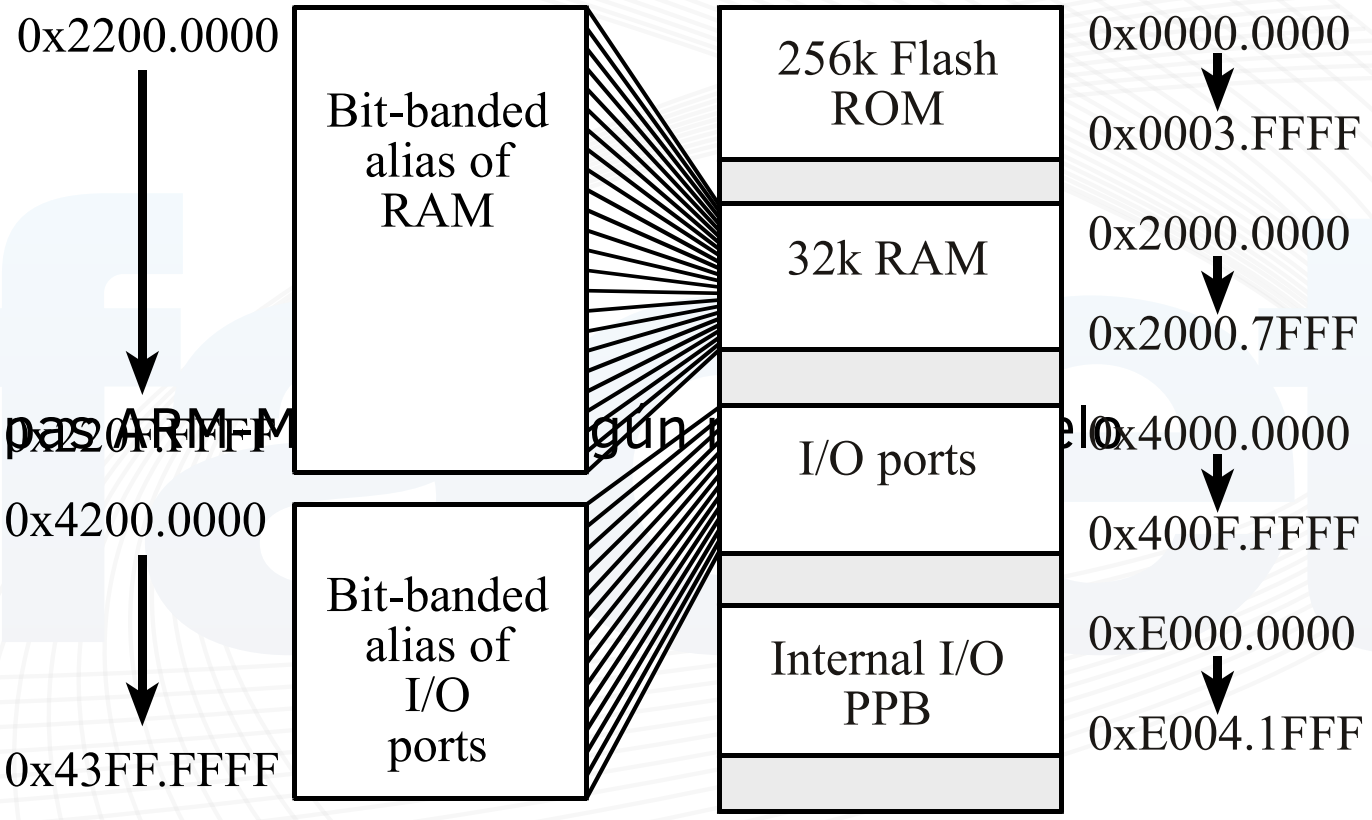
E/S mapeada en Memoria

- Hemos conectado dispositivos de E/S como si fueran memoria para el CPU.
  - Hemos empleado parte del mapa de memoria para E/S.
  - Todas las instrucciones del CPU que acceden a memoria son válidas.
  - En ARM no hay otra alternativa.
- Existen CPUs con mapa de E/S separado e Instrucciones separadas.
  - ¿Ventajas y Desventajas?.

Bit-Banding

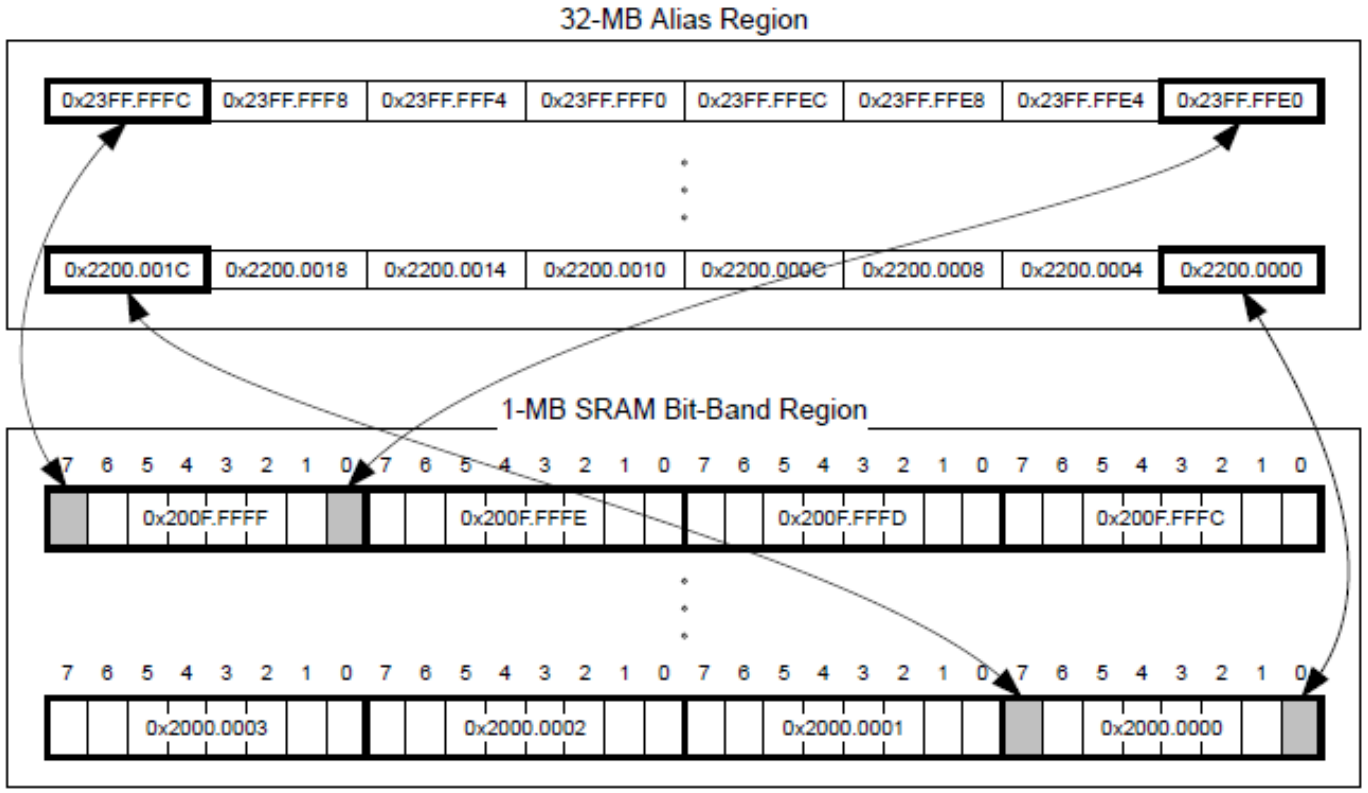
- Se usan direcciones que “contienen” un solo bit de un byte determinado en memoria.
- Estas direcciones no direccionan nada físicamente.
- La interfase del Bus del Sistema
  - Detecta los accesos a zona de bits, y mapea hacia la zona de la palabra a que pertenecen.
  - En lecturas entrega el bit como LSB.
  - Escritura en una única operación indivisible.

Ej: Bit-Banding RAM & I/O Ports



- Mapas ARM-M según modelo

Direccionamiento Bit-Banding



Se usa tambien en algunos puertos E/S