

DMA

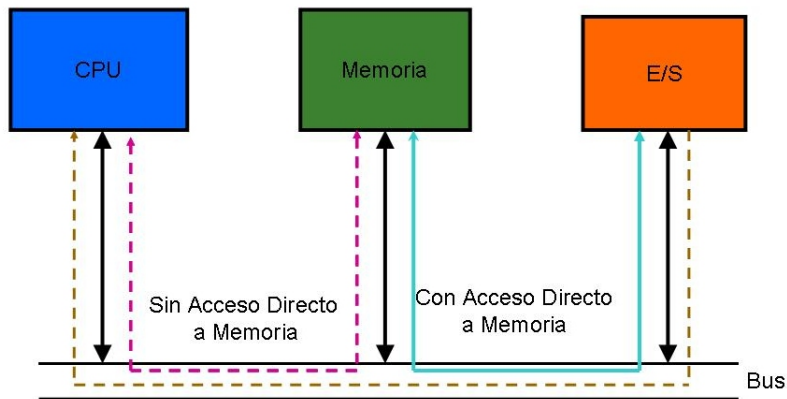
Juan Pablo Galeotti

8 de Noviembre de 2011

- ▶ E/S Por Polling.
- ▶ E/S Por Interrupciones.
- ▶ E/S Por DMA.

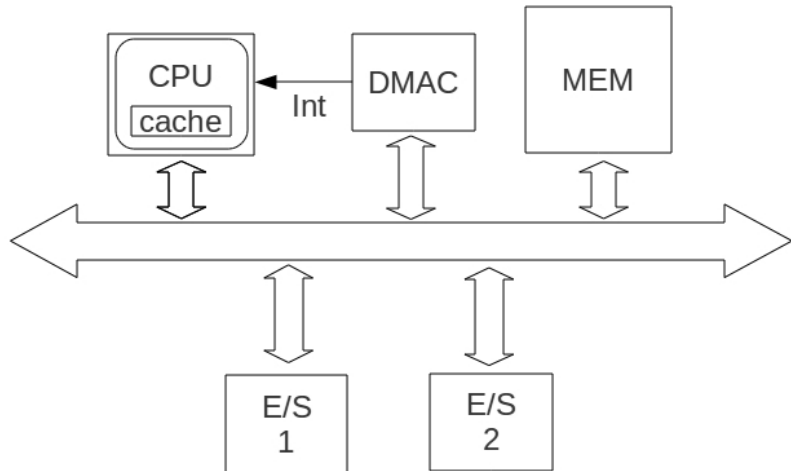
- ▶ **E/S por encuesta (Polling) o Programada:**
 - ▶ La CPU sondea periódicamente al dispositivo para ver cuál es su estado.
 - ▶ Sencillo pero claramente ineficiente.
- ▶ **E/S por interrupciones:**
 - ▶ Es el dispositivo quien establece el momento en que se realizará la transferencia de datos.
 - ▶ Cuando la CPU recibe notificación de la interrupción detiene el programa en ejecución, ejecuta la RAI y finalmente devuelve el control al programa.
- ▶ **E/S por acceso directo a memoria (DMA)**
 - ▶ Hoy!

Transferencias entre Módulos



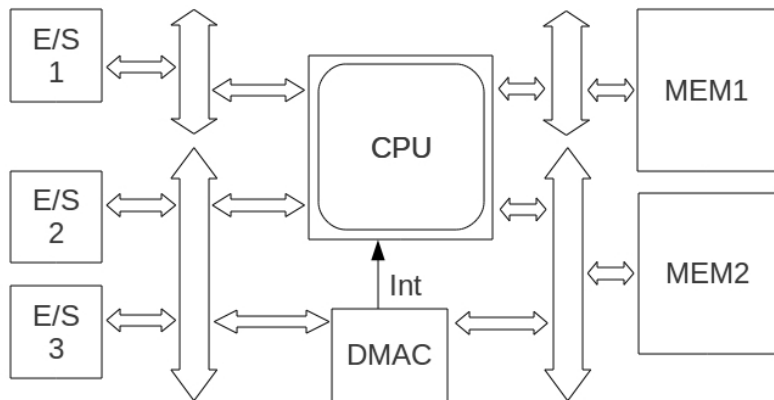
Ejercicio 1a.

Dado el esquema siguiente, ¿Bajo que condiciones el DMAC no interfiere con el CPU?



Ejercicio 1b.

Dado el esquema siguiente, ¿Bajo que condiciones el DMAC no interfiere con el CPU?



Ejercicio 2

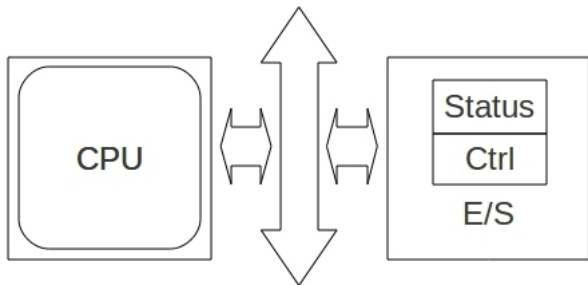
El siguiente protocolo de *handshaking* permite a un amo (por ejemplo: CPU) ordenar la realización de una acción a un esclavo (por ejemplo: dispositivo de E/S). Para el mismo, el esclavo debe contar con 2 registros de E/S: CTRL y STATUS.

1. El **amo** escribe el valor 0xFF en el registro CTRL
2. El **esclavo** reconoce el valor 0xFF en su registro CTRL, a continuación, escribe el valor 0xAC en el registro STATUS.
3. El **esclavo** realiza la acción para la cual fue invocado, una vez que finaliza escribe 0x00 en el registro CTRL y escribe 0x00 en STATUS
4. El **amo** reconoce el valor 0x00 en el registro STATUS.

Ejercicio 2

Suponiendo que el amo es un procesador ORGA1 y el esclavo es un dispositivo de E/S.

- ▶ Escribir en lenguaje ensamblador ORGA1 un programa para la CPU que obedezca este protocolo.
- ▶ Describir en pseudocódigo el comportamiento del dispositivo de E/S.
- ▶ ¿Cuál es el valor final de los registros CTRL y STATUS.?



Ejercicio 3

Escribir en pseudocódigo la rutina de una Controladora de tarjetas de memoria SD, dicha controladora posee los siguientes registros de E/S de 16 bits (Previamente cargados):

- ▶ `SD_IO_ADDRESS`: dirección de memoria (en el espacio de direccionamiento del dispositivo) que se desea leer/escribir.
- ▶ `SD_CTRL`: `0x00FF` para leer, `0x00AA` para escribir.
- ▶ `SD_STATUS`: `0x0001` El dispositivo se encuentra ocupado, `0x0000` en caso contrario.
- ▶ `SD_DATA`: Buffer donde se almacena el dato a escribir o el dato leído.

Asumir que se cuenta con las primitivas `readData()` que lee la posición de la tarjeta referida en `SD_IO_ADDRESS` y la guarda en `SD_DATA` y `writeData()` que escribe lo que esta en `SD_DATA` en la posición de la tarjeta de memoria referida en `SD_IO_ADDRESS`. Como el acceso a la memoria de la tarjeta es lento la primitiva `is_ready()` indica la finalización de la operación pedida.

Ejercicio 4

Escribir en pseudocódigo la rutina del DMAC que se encarga de transferir los datos entre la tarjeta SD del ejercicio anterior (tanto en el caso de lectura como de escritura). El DMAC posee un conjunto de registros de 16 bits, (previamente cargados) con la siguiente información:

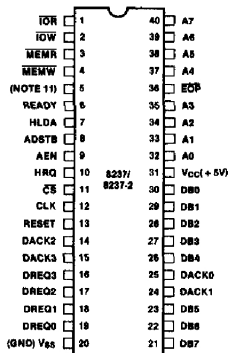
- ▶ `DMA_IO_ADDRESS`: dirección de memoria (en el espacio de direccionamiento del dispositivo) que se desea leer/escribir.
- ▶ `DMA_MEM_ADDRESS`: dirección de memoria (en el espacio de direccionamiento de la memoria principal) que se desea leer/escribir.
- ▶ `DMA_SIZE`: Cantidad de datos a transferir.
- ▶ `DMA_CTRL`:
 - ▶ Bit menos significativo = 1 en caso de escritura (memoria a dispositivo), y 0 en caso contrario.
 - ▶ Bit más significativo = 1 indica que ya se han cargado todos los datos necesarios, y se puede iniciar la transmisión.

Para su resolución dispone de las siguientes Primitivas:

- ▶ $requestInt()$ = Para solicitar la interrupción de fin de transmisión.
- ▶ $writeMem(ADDRESS, DATA)$ = Escribe en memoria principal lo que está en $DATA$.
- ▶ $readMem(ADDRESS)$ = Devuelve el dato que se encuentra en la posición $ADDRESS$ de la memoria principal.
- ▶ $writelIO(ADDRESS, DATA)$ = Escribe en el registro de E/S $ADDRESS$ del dispositivo lo que está en $DATA$.
- ▶ $readIO(ADDRESS)$ = Devuelve el dato que se encuentra en el registro de E/S $ADDRESS$ del dispositivo.

Escenario *Casi Real*

El Controlador 8237 es el DMAC clásico de las PC's, tiene varios "canales", alguno de los cuales se pueden configurar para usarse con un dispositivo y otros son fijos, la disquetera por ejemplo usa el canal 2.



Supongamos que se quiere leer un sector específico de un disquete y transferirlo hacia la memoria, la disquetera se posiciona para leer/escribir el sector y se configuran los siguientes parámetros en el DMAC:

- ▶ Dirección base de memoria destino/fuente.
- ▶ Número de bytes a transferir.
- ▶ El offset dentro del destino.
- ▶ El tipo de operación DMA (Lectura/Escritura).

Escenario *Casi Real*

Para realizar esto se tienen que setear los siguientes registros del DMAC:

- ▶ Registro de Modo (8 bits)
- ▶ Registro de Pagina (16 bits)
- ▶ Registro de Offset (16 bits)
- ▶ Registro de tamaño de bloque (16 bits)

Mode Register

7	6	5	4	3	2	1	0
MODE		INC	AI	TYPE		CHANNEL	

- Bits 6 and 7 are used to select the transfer mode: 00b = Demand mode, 01b = Single mode, 10b = Block mode, 11b = Cascade mode
- Setting INC selects *address decrement*, clearing INC selects *address increment*
- Setting AI enables auto-initialization
- Bits 2 and 3 are used to select the transfer type: 00b = Verify, 01b = Write to memory, 10b = Read from memory, 11b = Undefined
- Bits 0 and 1 are used to select the channel: 00b = channel 0, 01b = channel 1 10b = channel 2 and 11b = channel 3

Registro de Pagina: Dirección de memoria en donde está el buffer DMA.

Registro de Offset: Offset inicial del buffer usado en la transacción.



Registro de tamaño de bloque: Tamaño del bloque a ser transferido.

Ejercicio 5

En una computadora ORGA1 se ha conectado un controlador DMA. El acceso a cada uno de los registros del controlador está mapeado a memoria del siguiente modo:

DEVICE_ADDRESS	MEM_ADDRESS	SIZE	STATUS
0xFFF1	0xFFF2	0xFFF3	0xFFF4

Se quiere transferir desde la posición 0x0045 de la cinta hasta la 0x013A. Estos datos se deben guardar a partir de la posición 0xA142 de la memoria principal. El bit menos significativo del registro STATUS contiene un 1 en caso de escritura, y un 0 en caso contrario. Para indicarle al DMAC que ya se cargaron todos los datos y que puede iniciar la transmisión, se setea en 1 el bit más significativo del registro STATUS. Escribir un programa en *assembler* que acceda a una cinta para transferir datos via DMA.

-  Computer Organization and Architecture: Designing for Performance, 8/E William Stallings.
ISBN-10: 0136073735. ISBN-13: 9780136073734 Publisher: Prentice Hall. Copyright: 2010
-  Null, Linda. The essentials of computer organization and architecture / Linda Null, Julia Lobur. p. cm. ISBN 0-7637-0444-X