Módulo 11
Dispositivos de Entrada-Salida (Pt. 2)

Organización de Computadoras
Depto. Cs. e Ing. de la Comp.
Universidad Nacional del Sur

Copyright

- Copyright © **2011-2023** A. G. Stankevicius
- Se asegura la libertad para copiar, distribuir y modificar este documento de acuerdo a los términos de la **GNU** Free Documentation License, Versión 1.2 o cualquiera posterior publicada por la Free Software Foundation, sin secciones invariantes ni textos de cubierta delantera o trasera
- Una copia de esta licencia está siempre disponible en la página **http://www.gnu.org/copyleft/fdl.html**
- La versión transparente de este documento puede ser obtenida de la siguiente dirección:

http://cs.uns.edu.ar/~ags/teaching

Contenidos

- El rol de los módulos de **E/S**
- Estructura interna de un módulo de **E/S**
- **E/S** programada
- **E/S** con interrupciones
- **E/S** con acceso directo a memoria
- Almacenamiento rotacional y de estado sólido
- Niveles de **RAID**

Operación prototípica

- La obtención de datos desde un dispositivo externo podría involucrar los siguientes pasos:
  - El procesador interroga al módulo de **E/S** para comprobar el estado del dispositivo
  - El módulo de **E/S** señaliza el estado del dispositivo
  - Si el dispositivo está preparado para operar, el procesador solicita la transferencia de datos
  - El módulo de **E/S** obtiene los datos del dispositivo
  - Finalmente, los datos son puestos por el módulo de **E/S** a disposición del procesador

E/S programada

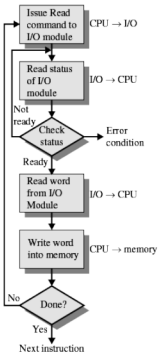
- Si algo se le puede criticar a esta operatoria es que el **CPU** parece quedar en todo momento atado al dispositivo de **E/S**
  - Como la velocidad del dispositivo suele ser inferior a la del procesador, estaremos desperdiciando capacidad de cómputo al quedar a la espera de que el dispositivo termine de resolver la última solicitud
- Este tipo de implementación se la denomina **E/S** programada, ya que el procesador ejecuta un programa que controla directamente al módulo de **E/S** (y por ende al dispositivo)

E/S programada

- Bajo esta modalidad de entrada/salida, el procesador realiza las siguientes tareas:
  - Sensa el estado del módulo
  - Envía los comandos de lectura/escritura
  - Transfiere desde/hacia el módulo los datos
- El procesador, al estar supervisando la tarea, debe esperar a que módulo y dispositivo completen el último requerimiento
  - Se dice que el procesador realiza un “busy waiting” (es decir, una “espera ocupada”).

E/S programada

- Analicemos en detalle la implementación de la operatoria "busy waiting":
  - El **CPU** solicita una operación de **E/S**
  - El módulo de **E/S** efectúa la operación
  - El módulo de **E/S** activa los bits de estado de manera acorde
  - El **CPU** monitorea periódicamente esos bits de estado
  - El módulo de **E/S** no puede informar al **CPU** de forma directa (¿por qué?)



Comandos de entrada/salida

- El **CPU** para enviar un comando de **E/S** a un cierto módulo debe suministrar su dirección
  - Esta dirección identifica unívocamente al módulo de **E/S** (y al dispositivo en caso de haber más de uno)
- Una vez identificado, el **CPU** envía el comando
  - Puede ser de control, indicándole al módulo qué hacer (por caso, encender el láser de una grabadora), o bien una consulta acerca del estado (por caso, se produjo algún error al realizar la última orden), o si no, una orden de lectura/escritura, instruyendo al módulo que transfiera los datos del buffer al dispositivo

Direccionamiento de dispositivos

- Bajo una **E/S** programada la transferencia de datos es muy parecida a un accesos a memoria: las instrucciones de **E/S** que el procesador trae de memoria durante su ciclo básico de ejecución se van mapeando en comandos de **E/S**
  - Cada dispositivo cuenta con un identificador único
  - Los instrucciones del **CPU** contienen ese identificador como argumento (el cual es, a los fines prácticos, una dirección de memoria)

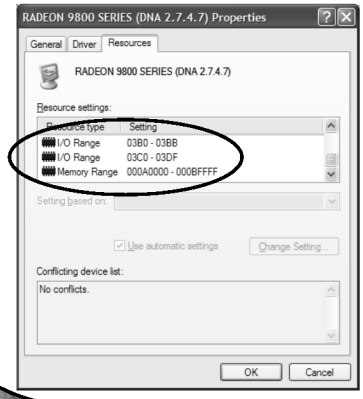
Direccionamiento de dispositivos

- A la hora de direccionar a un dispositivo de **E/S**, se han ensayado principalmente dos variantes
- Una posibilidad es hacer uso de una **E/S** mapeada en memoria principal, es decir, que los dispositivos y la memoria compartan el mismo espacio de direccionamiento
  - La **E/S** es indistinguible de un acceso convencional a memoria (por caso, la **E/S** en la **OCUNS**)
  - Como ventaja esto implica que no hace falta contar con instrucciones especiales para **E/S**

Direccionamiento de dispositivos

- La segunda alternativa consiste en aislar el espacio de direccionamiento para **E/S**
  - En otras palabras, el espacio de direccionamiento de la memoria principal es independiente del espacio de direccionamiento de los dispositivos de **E/S**
  - Nótese que esto implica contar con líneas adicionales (es decir, más pines en el **CPU**) para poder hacer referencia a esos objetos
  - A su vez, hace falta contar con instrucciones específicas para **E/S**, distintas de las anteriores de acceso a memoria

Ejemplo



Concepto de interrupción

- La interrupción es un mecanismo que permite indicarle al procesador que se produjo un evento que requiere atención inmediata
  - Para poder atender este evento es necesario interrumpir la ejecución del programa en curso
  - El estado del programa en ejecución debe ser guardado para poder ser retomado posteriormente
  - Al producirse una interrupción se ejecuta usualmente un pequeño programa de propósito específico a la interrupción actual denominado manejador de la interrupción (interruption handler)

Concepto de interrupción

- Las interrupciones fueron concebidas para atender principalmente excepción aritméticas y señalizar eventos de tiempo real
- Con el paso del tiempo, la alta flexibilidad que brinda este mecanismo permitió que se lo aplique a otras situaciones, tales como:
  - Comunicación con dispositivos de **E/S**
  - Invocación de servicios del sistema operativo
  - Resolución de los faltantes de páginas (page faults)

Taxonomía

- Las interrupciones se pueden clasificar de acuerdo a las siguientes dimensiones:
  - Sincrónicas vs. asincrónicas
  - Solicitadas por el usuario vs. forzadas
  - Enmascarables vs. no enmascarables
  - Entre-instrucciones vs. intra-instrucciones
  - Recuperable vs. insalvable

Atención de una interrupción

- A continuación repasaremos los pasos que se llevan adelante para atender una interrupción generada por un dispositivo de **E/S**:
  1) Un cierto dispositivo señaliza un interrupción
  2) El procesador finaliza la ejecución de la instrucción en curso (esto hace más fácil guardar el contexto)
  3) Antes de ejecutar la próxima instrucción revisa si se produjo una interrupción; al detectar que hubo una, envía una señal de **ACK** al dispositivo para informarle que ya puede bajar la señal de interrupción

Atención de una interrupción

- Continúa:
  4) El procesador se prepara para transferir el control al manejador; para esto guarda en la pila del sistema la palabra de estado del procesador y la dirección de la próxima instrucción a ser ejecutada
  5) Apelando al mecanismo del cual se disponga en la arquitectura, se determina la dirección de comienzo de la rutina que manejará la interrupción
  6) Esta rutina suele comenzar por guardar en la pila del sistema los distintos registros que van a ser modificados durante la atención de la interrupción

Atención de una interrupción

- Continúa:
  7) El manejador lleva adelante todas las acciones que requiera para completar la atención de la interrupción en curso
  8) Al terminar la atención de la interrupción, se procede a restaurar los valores de los registros que fueran oportunamente guardados en la pila del sistema
  9) Finalmente, se restaura la palabra de estado del procesador y la dirección de la próxima instrucción a ser ejecutada (es decir, continua la ejecución del programa interrumpido al principio)

E/S con interrupciones

- La **E/S** con interrupciones fue concebida como una alternativa superadora a la **E/S** programada
  - La clave de este tipo de **E/S** consiste en evitar que el **CPU** entre en el "busy waiting"
  - A tal efecto, la idea es que el **CPU** envíe un comando de **E/S** para inmediatamente dedicarse a atender otros asuntos y recién cuando el comando de **E/S** finalice su ejecución se interrumpirá al **CPU** para anoticiarlo de la novedad

E/S con interrupciones

- La operatoria básica de una **E/S** con interrupciones abarca las siguientes etapas:
  - El **CPU** envía un comando, por caso un **READ**
  - El módulo de **E/S** procede a leer los datos desde el dispositivo mientras el **CPU** atiende otros asuntos
  - Una vez finalizada la operación, el módulo de **E/S** interrumpe al **CPU**
  - El **CPU** solicita los datos obtenidos del dispositivo
  - El módulo de **E/S** finalmente los entrega

E/S con interrupciones

- La **E/S** con interrupciones constituye sin duda una mejora por sobre la **E/S** programada
  - El **CPU** no está más "atado" al dispositivo de **E/S**, ya que entre el envío de un comando y la recepción de los datos queda libre para atender a otros procesos
- No obstante, el **CPU** sigue interviniendo en la transferencia de la totalidad de los bytes enviados/recibidos
  - Más aún, cada vez que se interrumpe al **CPU** se debe llevar adelante <u>un costoso cambio de contexto</u>

Cambio de contexto

You
Por qué se dice que el cambio de contexto al atender un interrupción resulta costoso?

ChatGPT
El cambio de contexto se refiere al proceso en el que una CPU (Unidad Central de Procesamiento) o un programa cambia su estado activo de ejecución de una tarea a otra. Cuando se produce una interrupción, es necesario realizar un cambio de contexto para atender la interrupción y luego volver a la tarea original. Este cambio de contexto puede ser costoso en términos de tiempo y recursos por varias razones:

1. **Guardado y restauración de registros:** Antes de cambiar a una tarea diferente, la CPU debe guardar el estado actual de la tarea en ejecución, incluyendo los valores de los registros y otros datos relacionados. Cuando regresa a la tarea original, debe restaurar este estado. Este proceso consume tiempo y recursos.
2. **Flush de caché:** El cambio de contexto a menudo implica la limpieza de la caché de la CPU, ya que la nueva tarea puede tener diferentes datos y código que pueden no estar presentes en la caché. Esto lleva a una pérdida de eficiencia, ya que la caché debe llenarse nuevamente cuando se vuelve a la tarea original.
3. **Invalidate TLB (Buffer de Traducción de Direcciones):** La TLB almacena las traducciones de direcciones virtuales a direcciones físicas. Al cambiar de contexto, es posible que sea necesario invalidar algunas entradas de la TLB, lo que agrega una sobrecarga adicional.
4. **Operaciones de gestión de interrupciones:** Atender la interrupción en sí misma implica llevar a cabo ciertas operaciones de gestión de interrupciones, como determinar la causa de la interrupción, realizar alguna acción específica y luego volver a la tarea original.
5. **Tiempo de latencia:** Durante el cambio de contexto, la CPU está ocupada realizando operaciones relacionadas con la interrupción, lo que significa que no está ejecutando instrucciones de la tarea original. Esto introduce un tiempo de latencia en la ejecución de la tarea, lo que puede afectar el rendimiento del sistema.

En resumen, el cambio de contexto es costoso porque implica una serie de operaciones que consumen tiempo y recursos. En sistemas en tiempo real o en situaciones donde se busca una alta eficiencia y baja latencia, minimizar la frecuencia de los cambios de contexto puede ser crucial.

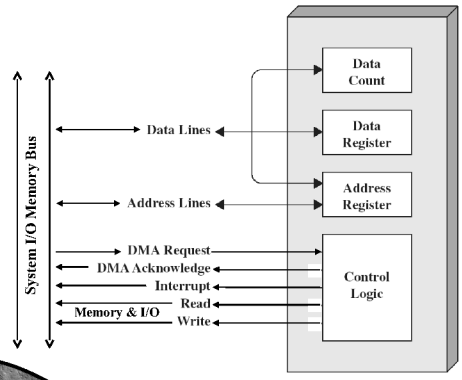
E/S con DMA

- La **E/S** con interrupciones parece mejorar el desempeño por sobre la **E/S** programada, si bien también se paga un costo oculto:
  - La **E/S** programada alcanza una elevada tasa de transferencia a instancias de mantener cautivo al **CPU**, el cual no podrá atender otros requerimientos
  - La **E/S** con interrupciones libera al procesador de ese uso exclusivo, pero sacrificando parte de la tasa de transferencia, pues cada byte leído tiene que pasar por las manos del **CPU**, lo que a su vez requiere de un costoso cambio de contexto

E/S con DMA

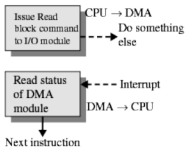
- Para capitalizar lo mejor de los dos escenarios se concibió la técnica de **E/S** con **DMA** (Direct Memory Access)
  - Esta técnica consiste en agregar un módulo **DMA** adicional al bus del sistema el cual asume la responsabilidad de gestionar la **E/S**
  - Este módulo debe usar el bus sólo cuando el **CPU** no lo esté usando o bien debe forzar al **CPU** a que difiera temporalmente su uso
  - Esta última variante es la más frecuente y se la denomina cycle stealing (robo de ciclos)

Diagrama del módulo DMA



Operatoria del módulo DMA

- El **CPU** informa al módulo **DMA**:
  - Tipo de operación (**READ/WRITE**)
  - Dirección del dispositivo
  - Dirección de memoria de comienzo
  - Cantidad de información a transferir
- Luego, el **CPU** trabaja en otra cosa
  - El módulo **DMA** se encarga de la transferencia (una palabra a la vez) desde o hacia memoria
- Por último, el módulo **DMA** interrumpe al **CPU**



Robo de ciclos

- Para transferir cada dato, el módulo **DMA** debe tomar el bus por un ciclo
- Transfiere de a una palabra por vez
- El robo de ciclos no es una interrupción:
  - El **CPU** no hace cambio de contexto, etc.
  - El **CPU** se suspende justo antes de acceder al bus
- Si bien esto ralentiza al **CPU**, no lo es tanto como cuando está a cargo de la **E/S** (esto es, bajo la **E/S** programa o con interrupciones)

Canales de E/S

- Los dispositivos de **E/S** son cada vez más sofisticados y requieren mayores tasas de transferencia
- La **E/S** con **DMA** parece ser la solución más adecuada, si bien el procesador todavía debe ser interrumpido una vez por cada bloque de datos transferido (al finalizar la misma)
- Para poder mejorar el desempeño de la **E/S** con **DMA** sólo resta hacer uso de procesadores de **E/S** de propósito específico

Canales de E/S

- Los canales de **E/S** cuentan con la capacidad de ejecutar instrucciones de **E/S**
  - También se los conocen como procesadores de **E/S**
- Los canales de **E/S** cuentan con un control completo sobre las operaciones de **E/S**
  - Por ejemplo, las placas controladoras de video
- La intención es que el **CPU** instruya al canal de **E/S** para hacer una transferencia y el canal de **E/S** realice la totalidad de la misma

¿Preguntas?