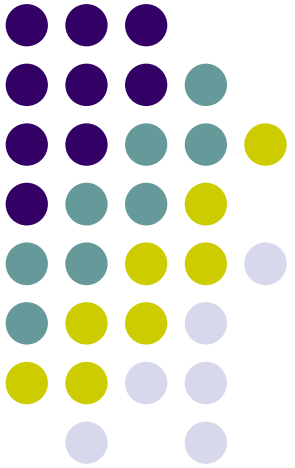
Tema 2

Gestión de Entrada/Salida



Bibliografía:

- Stallings, W (cap 6)
- Patterson, D.A. & Hennessy, J.L. (cap 8)

Indice

- Modelos de gestión de la E/S
- E/S mediante programa
- E/S mediante interrupciones
  - Gestión de las interrupciones
  - Las excepciones en MIPS
  - La rutina de atención
- Acceso directo a memoria
  - Inconvenientes de la E/S por interrupciones
  - Funcionamiento del DMA
  - Controladores de DMA

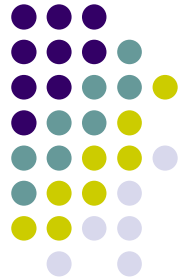
Modelos de gestión de la E/S

- En las fases de gestión de los procesos de E/S vimos que había que fijarse:
  - Sincronización
  - Transferencia de la información
- Estos procesos podían realizarse por software o por
- Da lugar a tres modelos de gestión:
  - E/S realizada por programa (encuesta/polling)
  - Interrupciones (sin DMA)
  - Acceso Directo a Memoria (interrupciones del DMAC)

	Polling	Interrupciones	DMA
Sincronización	SW-CPU	HW.	HW.
Transferencia	SW-CPU	SW-CPU	HW.

ATC UC

E/S mediante programa



- El procesador es el responsable de realizar todos los pasos de la operación:
  - Gestión
  - Sincronización
  - Transferencia de datos
- Para hacer una orden de E/S se ejecuta un programa que controla todas las acciones de la operación de E/S:
  - Manda la orden al controlador E/S
  - El controlador E/S realizará la operación y al finalizar activará los bits adecuados del registro de estado
  - El procesador debe comprobar periódicamente el estado del controlador hasta que la operación finalice
  - Cuando la operación ha finalizado se realiza la transferencia de la información

E/S mediante programa

- El proceso de esperar a que el controlador de E/S finalice la operación es el proceso de **Sincronización por Encuesta**
  - **Encuesta por estado**: Comprobar si se puede realizar la operación en un bit del registro de estado del controlador
    - Encuesta continua:

```
leer registro de estado
mientras no preparado hacer
        leer registro de estado
```

    - Encuesta periódica:

```
leer registro de estado
mientras no preparado hacer
        otra serie de acciones
        /* o esperar un cierto tiempo*/
        leer registro de estado
```

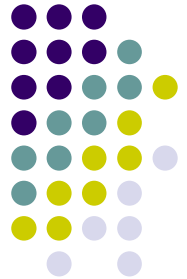
ATC UC

E/S mediante programa

- El procesador adapta su velocidad a la del dispositivo
- Los datos se intercambian entre el procesador y el controlador de E/S
- Las ventajas de este método son:
  - El código desarrollado es fácil de entender
  - No requiere hardware adicional
  - Se introduce poco overhead en el código
- Los inconvenientes de este método son:
  - La CPU no hace trabajo útil durante el bucle de espera
  - Consume mucho tiempo y mantiene ocupado al procesador innecesariamente
  - La dinámica del programa se detiene completamente durante la operación de E/S
  - Existen dificultades para atender a varios periféricos al mismo tiempo

E/S mediante programa

Ejemplo



- Ejemplo de programación: se desea realizar la lectura de un sector de un disco que está conectado por medio de un controlador con las siguientes características:
  - El controlador tiene 3 registros, se encuentran ubicados a partir de la dirección 0xbf900000 y cada uno ocupa 1 byte
  - El **registro de estado** indica:
    - Bit 0: Controlador está en operación.
    - Bit 1: Dato disponible.
    - Bit 2: Error de lectura.
  - Si en el **registro de comandos** se escribe el valor
    - 0x00 → Orden de lectura
    - 0x10 → El registro dato indica la pista
    - 0x20 → El registro dato indica el sector
  - Un **registro de datos** en donde se depositará cada byte del sector leído
  - Un sector ocupa 256 bytes y debe depositarse en la variable buffer

ATC UC

E/S mediante programa

Ejemplo

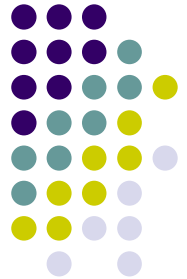
- Para realizar una de lectura de un sector del disco, se requieren los siguientes pasos, en los que los puntos de sincronización están en:
  - detectar que el controlador esté listo
  - detectar disponibilidad de carácter

Detectar que el controlador está listo.
Enviar al registro de comandos:
    el nº de pista, el nº de sector y la orden de lectura.
Mientras queden caracteres por recibir hacer
    detectar disponibilidad del carácter
    leer carácter y guardarlo
Leer el registro de estado para ver si la operación ha ido bien.
Si hay error volver a empezar.

ATC UC

E/S mediante programa

Ejemplo

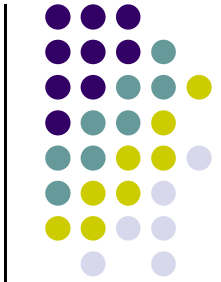


- Los controladores deben inicializarse, supongamos que el controlador ya está inicializado para actuar por ENCUESTA.
- Código:

```
#  Función  : leer_sector      Programador: XXXX XXXX XXXX
#  Fecha realización: 1/dic/2005
#  Ultima modificación: 15/dic/2005
#################################################################
# Descripción funcional: Función lee un sector de un disco
# sincronizándose por encuesta
#################################################################
# Parámetros: a0 → dirección de la variable buffer
#                  a1 → Número de pista
#                  a2 → Número de sector
#################################################################
```

E/S mediante programa

Ejemplo



```
#  registros del controlador
            .equ        R_status, 0xbf900000
            .equ        R_datos, 0xbf900001
            .equ        R_comando, 0xbf900002

# Código de la subrutina
Leer_sector:
# Detectar que el controlador está listo.
            la          $t0, R_status
Espera_dis:
            lb          $t1, 0($t0)
            andi        $t1, $t1, 0x01
            beq         $t1, $zero, Espera_dis
# Enviar al registro de comandos:
# el nº de pista, el nº de sector y la orden de lectura.
            la          $t0, R_datos
            la          $t1, R_comando
            sb          $a1, 0($t0)          # se guarda el número de pista
            li          $t2, 0x10            # comando de número de pista
            sb          $t2, 0($t1)
            sb          $a2, 0($t0)          # se guarda el número de sector
            li          $t2, 0x20            # comando de número de sector
            sb          $t2, 0($t1)
            li          $t2, 0x00            # comando de lectura
            sb          $t2, 0($t1)
```

E/S mediante programa

Ejemplo

```
# Mientras queden caracteres por recibir hacer
            li      $t3, 0x0100         # número de caracteres
Mientras:   beq     $t3, $zero, Fin_mientras
# detectar disponibilidad del carácter
            la      $t0, R_status
Espera_car:
            lb      $t1, 0($t0)
            andi    $t1, $t1, 0x02
            beq     $t1, $zero, Espera_car
# leer carácter y guardarlo
            la      $t0, R_datos
            lb      $t1, 0($t0)
            sb      $t1, 0($a0)
            addi    $a0, $a0, 1         # incremento el puntero buffer
            addi    $t3, $t3, -1        # decrementa contador
            b       Mientras
# Leer el registro de estado para ver si la operación ha ido bien.
Fin_mientras:
            la      $t0, R_status
            lb      $t1, 0($t0)
# Si hay error volver a empezar.
            andi    $t1, $t1, 0x04
            bne     $t1, $zero, leer_sector
# fin de la subrutina
#el sector será procesado en el programa principal
            jr      $ra
```

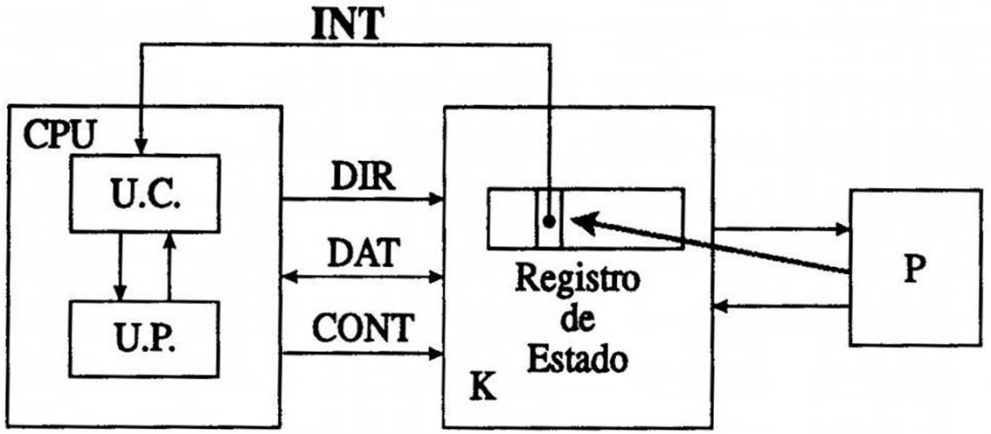
E/S mediante Interrupciones

- **Interrupción**: suceso asíncrono que se produce como consecuencia de un evento externo, que hay que tratar mediante la ejecución de una rutina
  - Esto significa que las interrupciones pueden usarse como una forma de sincronizar los procesos de E/S
  - Una interrupción está ligada a una acción de un dispositivo
  - Cada interrupción deberá tener asociada la rutina que lleve a cabo la acción requerida por el dispositivo

ATC UC

E/S mediante Interrupciones

- Con interrupciones, el hardware del controlador ayuda en el proceso de sincronización
  - El procesador envía la orden de E/S al controlador y se despreocupa, mientras continúa haciendo otro trabajo
  - El controlador E/S realizará la operación y al finalizar activará los bits adecuados del registro de estado
  - El hardware asociado al controlador, al cambiar los bits de estado, activará la señal de interrupción para avisar a la CPU
  - La CPU detecta el cambio en esta señal y realiza las acciones necesarias para que se ejecute la rutina correspondiente que finaliza la operación de E/S



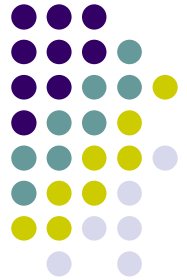
E/S mediante Interrupciones

- Los programas de E/S deben dividirse en dos partes y entre estas partes actúa el hardware
  - Inicio de la operación de E/S (software):
    - Se solicita al controlador de E/S una operación de E/S
  - Actuación del hardware:
    - El controlador completa la operación
    - Se genera la señal de interrupción y se transmite a la CPU
    - La CPU determina si se acepta la petición de interrupción
  - Software de gestión de las interrupciones (S.O)
    - Se determina la fuente de la interrupción
    - Se salta a la rutina de servicio
  - Rutina de servicio (software)
    - Salva el estado del programa interrumpido
    - Realiza las acciones correspondientes a la interrupción
      - Finalización de la operación de E/S
    - Recupera el estado y finaliza la rutina

ATC UC

E/S mediante Interrupciones

Hardware de interrupción

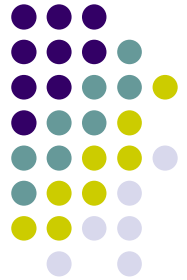


- El controlador posee una señal de petición de interrupción
- Esta señal se activa al finalizar una operación de E/S
  - Suele activarse al tiempo que se activa algún bit del registro de estado (sobre el que se hace la encuesta en el caso de E/S por programa)
  - Debe programarse el controlador para que genere esta señal
  - La programación se hace en el proceso de inicialización del controlador, usualmente se hace mediante el registro de control
- Esta señal se transmite a la CPU por medio del bus de sistema

ATC UC

E/S mediante Interrupciones

Hardware de interrupción



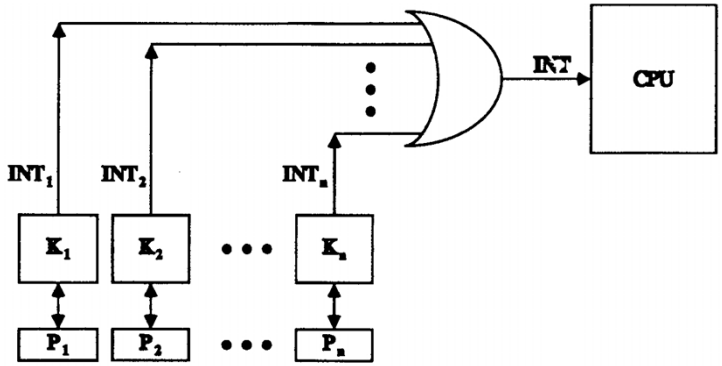
- La CPU dispone de señales para recibir las peticiones de interrupción:
  - El procesador no acepta interrupciones mientras está ejecutando una instrucción.
  - Entre instrucciones comprueba el de estado de estas líneas para determinar que interrupciones hay pendientes
- Casos de estudio:
  - Número de líneas de petición de interrupción que posee la CPU
  - Aceptación de la interrupción:
    - Mecanismos de enmascaramiento
    - Mecanismos de priorización

ATC UC

E/S mediante Interrupciones

Hardware de interrupción

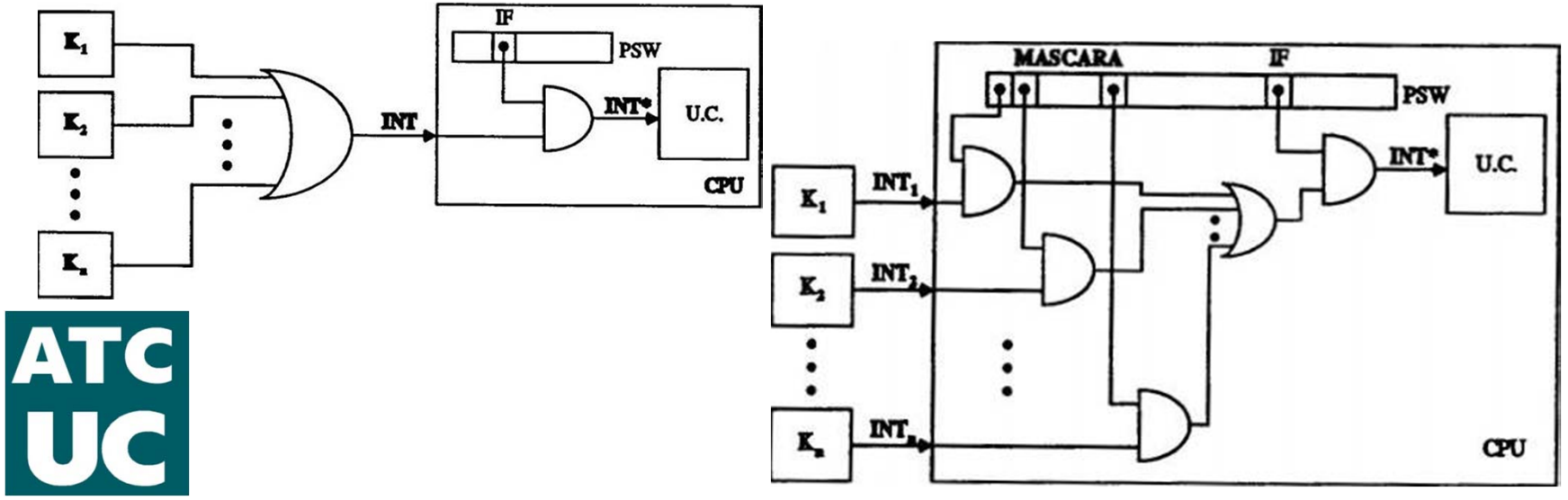
- La CPU puede poseer una o varias líneas de petición de interrupción:
  - Si posee una línea:
    - Todos los controladores se conectan a una única línea
  - Si posee varias líneas:
    - No se puede usar una línea por periférico
      - Más periféricos que líneas
    - Se utilizan varias líneas para indicar diferentes prioridades
  - Cada línea soporta varios periféricos:
    - Problemas para identificar el origen de la interrupción
    - No hay forma hardware de priorizar los dispositivos



E/S mediante Interrupciones

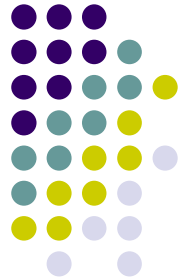
Hardware de interrupción

- Se debe poder decidir que interrupciones serán atendidas y cuáles se dejarán pendientes
  - Esto se hace mediante máscaras y se conoce como '**enmascarar**' peticiones
    - Las peticiones se pueden enmascarar de forma individual o colectiva
    - Usualmente se realiza en el registro de estado del procesador
  - Los procesadores poseen entradas no enmascarables (RESET, BERR, NMI), que hay que atenderlas inmediatamente



E/S mediante Interrupciones

Hardware de interrupción

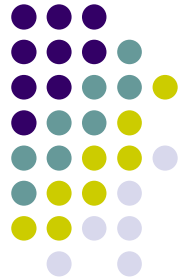


- Cuando varios periféricos solicitan a la vez una interrupción, la CPU debe determinar a cuál atiende primero
  - La prioridad puede establecerse
    - Asignada a cada línea de entrada
    - Durante el proceso de reconocimiento
    - Por medio de un controlador de interrupciones

ATC UC

E/S mediante Interrupciones

Hardware de interrupción



- Proceso de reconocimiento: es el mecanismo por el que se determina que dispositivo ha solicitado una interrupción
- Se puede realizar:
  - Por software: cuando la CPU decide atender una interrupción, no analiza la causa sino que salta a una rutina de carácter general que determina mediante encuesta el dispositivo que provocó la interrupción
  - Por hardware: la CPU analiza la causa de la interrupción saltando a la rutina especifica de la interrupción solicitada
- En ambos casos deben tratarse los aspectos de prioridad

ATC UC

E/S mediante Interrupciones

Hardware de interrupción

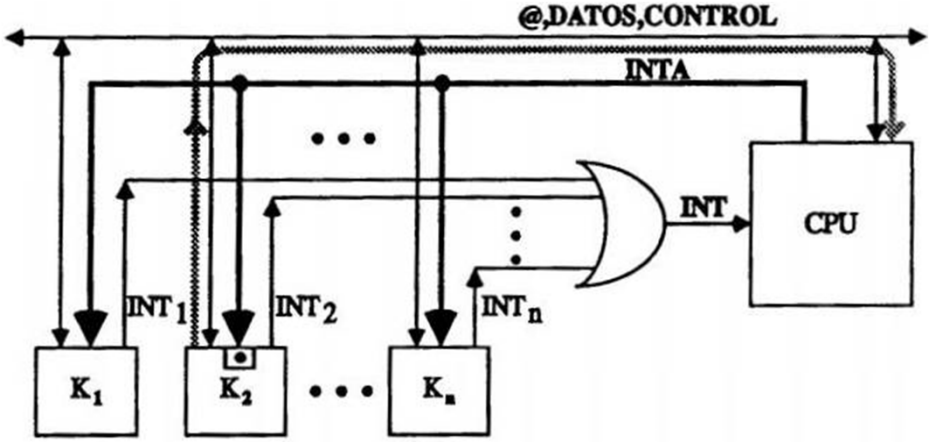
- Identificación del origen de la interrupción
  - Por software (gestor de interrupciones → S.O.)
    - Se ejecuta una rutina de carácter general que determina mediante encuesta que dispositivo ha provocado la interrupción
    - Habitualmente se comprueba el registro de estado
    - Problema: la consulta software consume mucho tiempo
    - Soluciona los dos problemas
      - Identificación - por encuesta
      - Prioridad - orden de la encuesta

ATC UC

E/S mediante Interrupciones

Hardware de interrupción

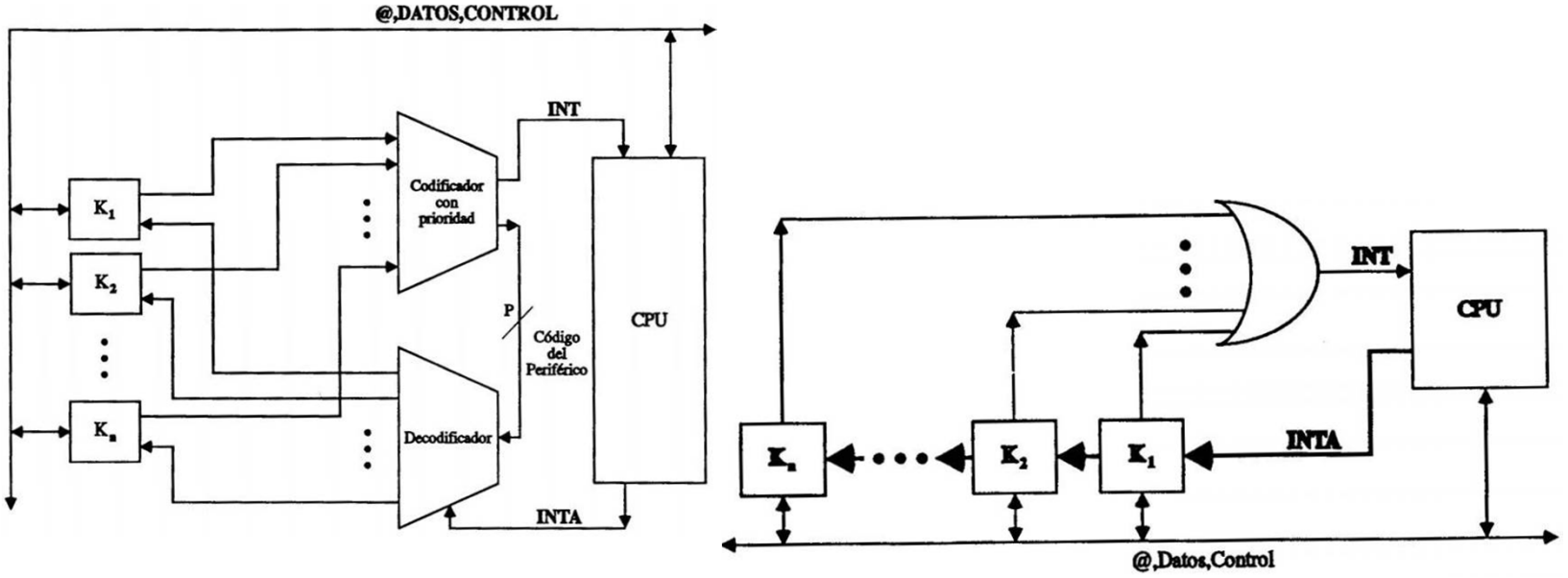
- Por hardware
  - Deben existir señales de reconocimiento (INTA)
  - El dispositivo proporciona información que será empleada para determinar la rutina a ejecutar
    - La propia dirección de la rutina
    - El código de una instrucción
    - Un identificador (número de vector)
  - Este método requiere hardware adicional para tratar las prioridades



E/S mediante Interrupciones

Hardware de interrupción

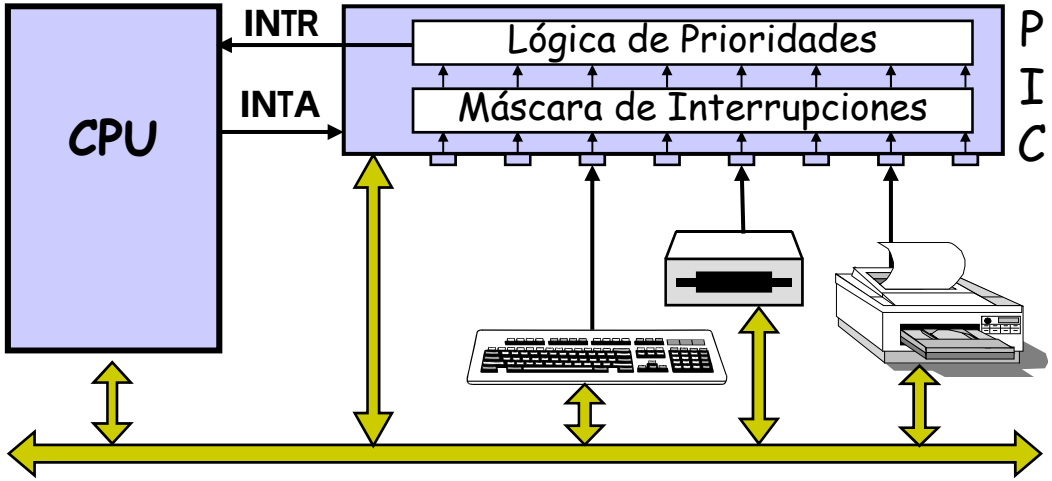
- Técnicas de establecimiento de prioridades en la detección por hardware:
  - Empleo de un codificador/decodificador
  - Mecanismo “Daisy Chain”.



E/S mediante Interrupciones

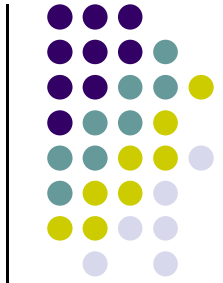
Hardware de interrupción

- Existen en el mercado unos dispositivos hardware que se encargan de la gestión de las interrupciones:
- Se denominan **Controladores de Interrupción**
  - Cada entrada del controlador de interrupciones está asignada a un solo dispositivo
  - Gestionan las prioridades: el controlador puede ser programado para asignar a cada dispositivo una prioridad diferente (pueden incluir prioridades dinámicas)
  - Gestionan el enmascaramiento de cada dispositivo
  - Se encargan de asignar un vector de interrupción a cada dispositivo



E/S mediante Interrupciones

Software: la rutina de atención



- La rutina de atención es el código que se encarga de completar la operación de E/S.
- Se ejecuta como una subrutina normal, pero debe encargarse de ciertas tareas de gestión:
  - **Salvaguarda del programa interrumpido**: el programa que se estaba ejecutando cuando se produjo la interrupción debe poder seguir ejecutándose al terminar ésta, esto implica que toda la información relativa a su ejecución debe ser guardada para poder recuperarla cuando finalice la rutina
    - Cierta información suele ser almacenada por la CPU durante el proceso de detección (el PC, PSW y algunos registros)
    - Los registros que se modifiquen durante la ejecución de la rutina de atención se deben salvar al principio de la rutina de atención
    - El gestor de interrupciones (S.O.) , en caso de existir, se suele encargar de salvar los registros.

ATC UC

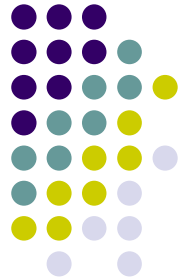
E/S mediante Interrupciones

Software: la rutina de atención

- Debe garantizar que se **desactive la petición de interrupción**, ya que sino al finalizar la atención volverá a producirse
  - Esta tarea puede ser automática o que se encargue de ella el controlador de interrupciones
- La finalización de la rutina suele hacerse con una instrucción especial que restaure el valor de los registros salvados automáticamente
- Además debe tenerse en cuenta que la rutina de atención no es una subrutina y que no puede recibir parámetros ni devolver resultados
  - Todo intercambio de información debe hacerse mediante el empleo de **variables globales**

ATC UC

E/S mediante Interrupciones

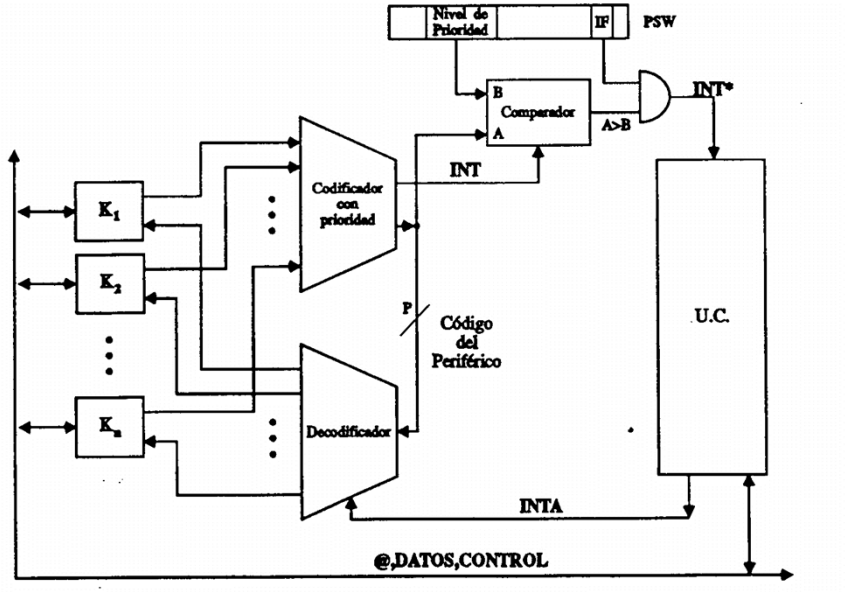


- ¿Qué ocurre cuando se produce una interrupción mientras se está atendiendo otra?
  - Nivel de interrupciones único:
    - Una vez que se inicia la ejecución de una rutina de interrupción debe continuar hasta el final sin que le CPU pueda aceptar otra interrupción.
  - Interrupciones multinivel:
    - Se puede atender peticiones de interrupción anidadas.
    - A cada causa de interrupción se le asigna un nivel de prioridad.
    - Una interrupción sólo se atiende si su nivel de prioridad es superior al de la interrupción en curso.

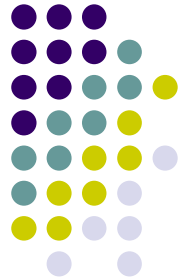
ATC UC

E/S mediante interrupciones

- Gestión de Interrupciones multinivel
  - Dependiendo de la CPU la gestión se realiza por software o hardware
  - Las soluciones software (complejas) se implementan en el gestor de interrupciones (S.O.)
  - Hay diversas soluciones hardware, por ejemplo el codificador con prioridad:



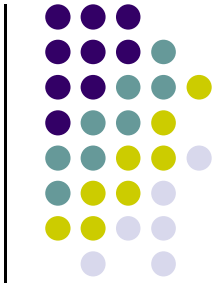
Excepciones en MIPS



- En los procesadores MIPS (y en la mayoría de procesadores) todas las excepciones, incluidas las interrupciones, reciben un tratamiento idéntico.
- Son excepciones:
  - Fallos en el subsistema de memoria.
  - Errores del programa.
  - RESET.
  - TRAPS y llamadas al sistema.
  - Las interrupciones

ATC UC

Excepciones en MIPS



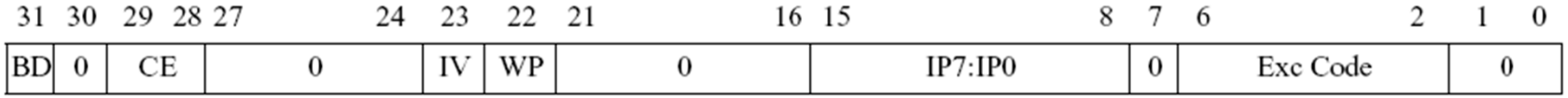
- Cuando el procesador decide atender una excepción:
  - Guarda la dirección de retorno en el registro **EPC** (coprc. 0)
  - Cambia el modo de operación a “kernel” (máxima prioridad).
  - Deshabilita las interrupciones en el registro **SR**.
  - Indica la razón de la excepción en el registro **CAUSE**.
  - Si se trata de una excepción de fallo de memoria indica la dirección de fallo en el registro **BadVaddr**.
- El control de programa salta a una dirección fija, función de la excepción y del bit **BEV** del **SR**:

Exception type	*Entry point*			
	SR(BEV)==0		*SR(BEV)==1*	
	Program	*Physical*	*Program*	*Physical*
Reset, NMI			0xBFC0 0000	0x1FC0 0000
TLB refill, 32-bit task	0x8000 0000	0x0	0xBFC0 0200	0x1FC0 0200
XTLB refill, 64-bit task	0x8000 0080	0x80	0xBFC0 0280	0x1FC0 0280
Cache error (R4x00 and later)	0xA000 0100	0x100	0xBFC0 0300	0x1FC0 0300
Interrupt (some QED CPUs only)	0x8000 0200	0x200	0xBFC0 0400	0x1FC0 0400
All other exceptions	0x8000 0180	0x180	0xBFC0 0380	0x1FC0 0380

ATC UC

Excepciones en MIPS

- El registro CAUSE:

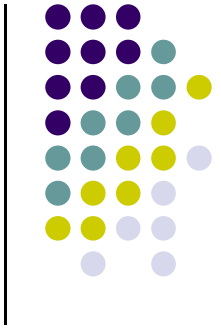


- El campo **Exc Code** indica la fuente de la excepción. Algunos valores son:

Exception Code Value		Mnemonic	Description
Decimal	Hexadecimal		
0	16#00	Int	Interrupt
1	16#01	Mod	TLB modification exception
2	16#02	TLBL	TLB exception (load or instruction fetch)
3	16#03	TLBS	TLB exception (store)
4	16#04	AdEL	Address error exception (load or instruction fetch)
6	16#06	IBE	Bus error exception (instruction fetch)
8	16#08	Sys	Syscall exception
9	16#09	Bp	Breakpoint exception

ATC UC

Excepciones en MIPS



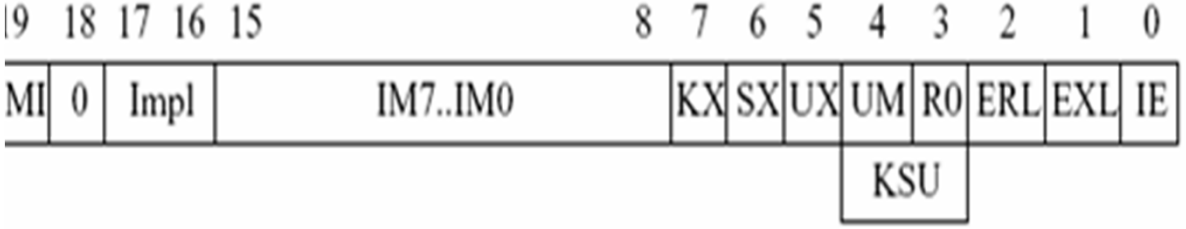
- Retorno de la rutina de atención:
  - Cuando finaliza la rutina de atención debe volverse al programa interrumpido manteniendo su modo de operación.
  - La instrucción **rfe** (restore from exception) restaura el estado anterior.
  - Debe ejecutarse como la instrucción siguiente a la vuelta de la excepción (usando el salto retardado).

```
mfc0    k0,C0_EPC
nop
j       k0
rfe
```

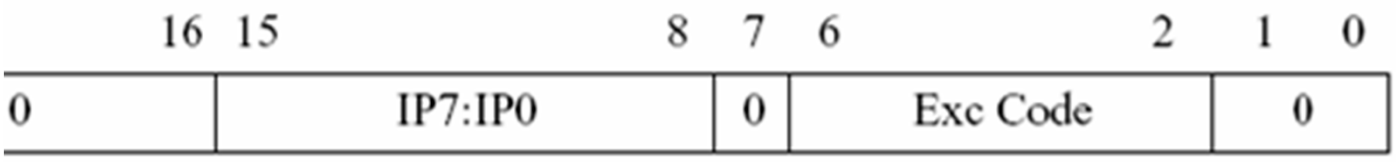
- En los microprocesadores posteriores a MIPS III existe una instrucción **eret** que realiza todas estas acciones.

Las interrupciones en MIPS

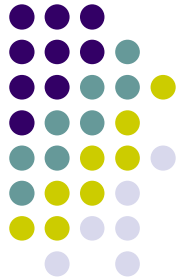
- Los microprocesadores MIPS disponen de 8 fuentes de interrupción.
- Pueden enmascararse de forma individual en los bits **IM7..IM0** del SR.
- El Bit **IE** del SR habilita de forma global todas las interrupciones.



- Para determinar la fuente que ha provocado la interrupción se miran los bits **IP7..IP0** del registro CAUSE:



Las interrupciones en MIPS

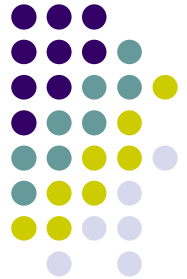


- Los procesadores MIPS no implementan mecanismos de excepciones anidadas:
  - La rutina de servicio haga una copia del **SR** y del **EPC**.
  - Debe tenerse cuidado con los registros “reservados” para la gestión de las excepciones *k0* y *k1*.
- En los procesadores MIPS todas las interrupciones tienen la misma prioridad:
  - La gestión de las prioridades puede hacerse por software.

ATC UC

E/S mediante interrupciones

Ejemplo



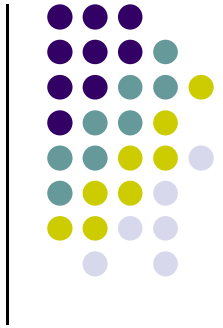
- **Volvemos al ejemplo del disco**.
- Si la sincronización se realiza por interrupciones:
  - El controlador tenga disponible un carácter del sector activará la señal de interrupción.
  - El **programa principal**, realizará los siguientes pasos:

Esperar que el controlador esté listo.
Programar el controlador de disco:
Enviar el nº de pista y el nº de sector.
Enviar la orden de lectura.
Aviso al SO que debemos ESPERAR.
Procesar sector leído.

ATC UC

E/S mediante interrupciones

Ejemplo



- **Rutina de atención**, se activa para recoger cada carácter:

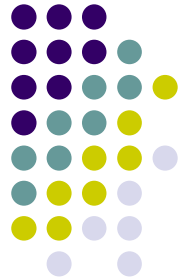
```
Recoger el dato y almacenarlo
Si no quedan más datos entonces
    Leer estado
    Si se ha producido un error entonces
                Esperar a que el controlador esté listo
                Volver a programar el controlador (reintentar)
                        Enviar el nº de pista y el nº de sector
                        Enviar la orden de lectura
    Sino Indicar al SO que a finalizado la ESPERA
```

  - Supondremos:
    - Que existe una rutina de inicialización que programa el controlador para que actúe por interrupciones.
    - Que existe una rutina que determina la fuente y llama a la rutina de atención como una subrutina.
    - Que la interrupción se considera atendida al leer el dato.

ATC UC

E/S mediante interrupciones

Ejemplo



- Vamos a llamar a la rutina TAREAS mientras se recibe todo el sector por interrupciones.
- Código de la rutina de lectura:

```
#  Función  : leer_sector          Programador: XXXX XXXX XXXX
#  Fecha realización: 1/dic/2005
#  Ultima modificación: 15/dic/2005
#########################################################
# Descripción funcional: Función inicia la lectura de un sector de
# un disco. Espera que se complete la lectura  llamando a TAREAS
#########################################################
# Parámetros: a0 → dirección de la variable buffer
#                 a1 → Número de pista
#                 a2 → Número de sector
#########################################################
# Usa las variable GLOBALES: SECTOR, PISTA, BUFFER_INI,
# BUFFER y CONTADOR para que sean usadas
# por la rutina de atención.
```

E/S mediante interrupciones

Ejemplo

```
#  registros del controlador
            .equ        R_status, 0xbf900000
            .equ        R_datos, 0xbf900001
            .equ        R_comando, 0xbf900002

# Código de la subrutina
Leer_sector:
# Detectar que el controlador está listo.
            la          $t0, R_status
Espera_dis:
            lb          $t1, 0($t0)
            andi        $t1, $t1, 0x01
            beq         $t1, $zero, Espera_dis

# Inicializar las variables globales para la rutina de Interrupción
            la          $t3, SECTOR
            sb          $a2, 0($t3)
            la          $t3, PISTA
            sb          $a1, 0($t3)
            la          $t3, BUFFER_INI
            sw          $a0, 0($t3)
            la          $t3, BUFFER
            sw          $a0, 0($t3)
            la          $t3, CONTADOR
            sb          $zero, 0($t3)          # número de caracteres
```

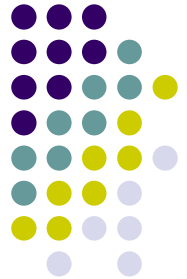
E/S mediante interrupciones

Ejemplo

```
# Enviar al registro de comandos:
# el nº de pista, el nº de sector y la orden de lectura.
        la      $t0, R_datos
        la      $t1, R_comando
        sb      $a1, 0($t0)         # se guarda el número de pista
        li      $t2, 0x10           # comando de número de pista
        sb      $t2, 0($t1)
        sb      $a2, 0($t0)         # se guarda el número de sector
        li      $t2, 0x20           # comando de número de sector
        sb      $t2, 0($t1)
        li      $t2, 0x00           # comando de lectura
        sb      $t2, 0($t1)

# Supondremos que la llamada al SO que debemos esperar es:
        li      $v0,25
        syscall

# fin de la subrutina
# el sector será procesado en el programa principal
        jr      $ra
```

E/S mediante interrupciones

Ejemplo



- Código de la Rutina de Atención:

```
#  Función  : Atencion_Disco    Programador: XXXX XXXX XXXX
#  Fecha realización: 1/dic/2005
#  Ultima modificación: 15/dic/2005
#######################################################
# Descripción funcional: Esta rutina se activa cada vez que el
# controlador de disco haya recibido un byte. La rutina guarda el
# byte en el buffer y comprueba si es el último, en este caso debe
# comprobar si ha habido errores.
#
# Esta subrutina es llamada por el gestor de interrupciones.
#######################################################
# Usa las variable GLOBALES: SECTOR, PISTA, BUFFER_INI,
# BUFFER y CONTADOR.
```

ATC UC

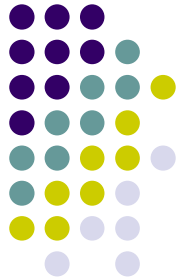
E/S mediante interrupciones

Ejemplo

```
# Código de la subrutina
Atencion_Disco:
# Salvar los registros que se usen en el stack ($t0-$t3).
        addiu   $sp, $sp,  -16
        sw      $t0, 0($sp)
        sw      $t1, 4($sp)
        sw      $t2, 8($sp)
        sw      $t3, 12($sp)
# leer la posición del buffer y la cantidad de datos leidos
        la      $t0, BUFFER
        lw      $t2,0($t0)
        la      $t0, CONTADOR
        lb      $t3,0($t0)
# leer carácter y guardarlo
        la      $t0, R_datos
        lb      $t1, 0($t0)
        sb      $t1, 0($t2)
        addi    $t2, $t2, 1             # incremento el puntero buffer
        addi    $t3, $t3, 1             # incrementa contador
# Guardar la posición del buffer y la cantidad de datos leidos
        la      $t0, BUFFER
        sw      $t2,0($t0)
        la      $t0, CONTADOR
        sb      $t3,0($t0)
```

E/S mediante interrupciones

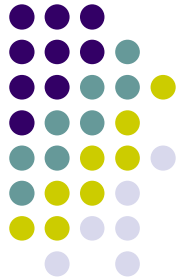
Ejemplo



```
#Ha habido errores hay que reintentar
# Comprobar si quedan mas caracteres
            li      $t0,0x0100
            bne     $t0,$t3,Fin_Rutina
# Leer el registro de estado para ver si la operación ha ido bien.
            la      $t0, R_status
            lb      $t1, 0($t0)
# Si hay error Reintentar.
            andi    $t1, $t1, 0x04
            bne     $t1, $zero, Repetir
# Todo Correcto.
# Supondremos que la llamada al SO que ha finalizado la espera es:
            li      $v0,26
            syscall
            b       Fin_Rutina
Repetir:
# Detectar que el controlador está listo.
            la      $t0, R_status
Espera_dis:
            lb      $t1, 0($t0)
            andi    $t1, $t1, 0x01
            beq     $t1, $zero, Espera_dis
```

E/S mediante interrupciones

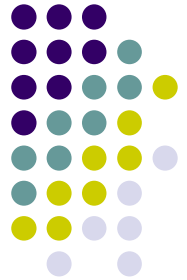
Ejemplo



```
# Se reinicia la variable BUFFER
        la      $t3, BUFFER_INI
        lw      $t2, 0($t3)
        la      $t3, BUFFER
        sw      $t2, 0($t3)
# Enviar al registro de comandos:
# el nº de pista, el nº de sector y la orden de lectura.
        la      $t0, R_datos
        la      $t1, R_comando
        la      $t3, PISTA
        lb      $t2, 0($t3)
        sb      $t2, 0($t0)         # se guarda el número de pista
        li      $t2, 0x10           # comando de número de pista
        sb      $t2, 0($t1)
        la      $t3, SECTOR
        lb      $t2, 0($t3)
        sb      $t2, 0($t0)         # se guarda el número de sector
        li      $t2, 0x20           # comando de número de sector
        sb      $t2, 0($t1)
        li      $t2, 0x00           # comando de lectura
        sb      $t2, 0($t1)
```

E/S mediante interrupciones

Ejemplo

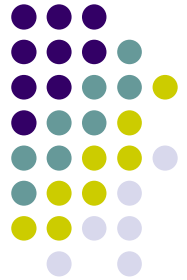


```
# fin de la subrutina
Fin_Rutina:
# Recuperar los registros salvados del stack ($t0-$t3).
        lw      $t0, 0($sp)
        lw      $t1, 4($sp)
        lw      $t2, 8($sp)
        lw      $t3, 12($sp)
        addiu   $sp, $sp,  16

# Vuelta al gestor de interrupciones
        jr      $ra
```

ATC UC

E/S mediante interrupciones

Ejemplo

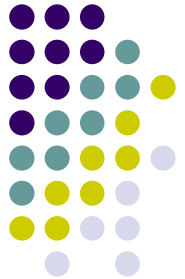


- Ejemplo de rutina gestora de Excepciones.
- Supondremos que la interrupción de disco está en la línea 3.
- Esta rutina debe estar en la posición 0x8000 0200 para gestionar las interrupciones.

```
#  Función  : Gestor_Int          Programador: XXXX XXXX XXXX
#  Fecha realización: 1/dic/2005
#  Ultima modificación: 15/dic/2005
#############################################################
# Descripción funcional: Esta rutina determina la causa de una
# excepción (interrupción) y salta a la rutina de atención
# correspondiente.
#############################################################
# Usa los registros k0 y k1 reservados para el gestor de
# interrupciones
```

E/S mediante interrupciones

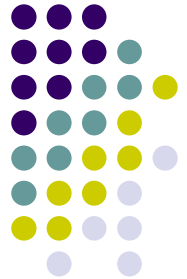
Ejemplo



```
# Constantes del coprocesador
            .equ        C0_SR, 0x0C
            .equ        C0_CAUSE, 0x0D
            .equ        C0_EPC, 0x0E

# Código de la rutina
Gestor_Int :
# leer el registro CAUSE del coprocesador 0
            mfc0        $k0,C0_CAUSE
# Comprobar los bits de Exc Code
# No es necesario en el caso de interrupciones
            andi        $k1,$k0,0x07C       # los bits del 2 al 6
            beq         $k1,$zero,          Interrupción
# ahora se comprueban otras excepciones

# Si es una interrupción
# Comprobar los bits de IP para saltar a la rutina correspondiente
Interrupcion:
# ahora se comprueban todas las líneas (solo compruebo la linea 3)
            andi        $k1,$k0,0x0800      # el bit 11 corresponde a la linea 3
            beq         $k1,$zero,          Linea_3
```

E/S mediante interrupciones

Ejemplo



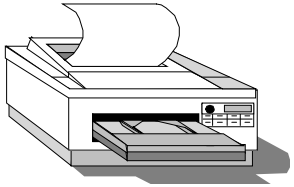
```
# Si es la interrupción de la línea 3
# La dirección de la rutina la gestiona el SO y suele estar en una tabla
Linea_3:
        la      $k0, Atencion_Disco
        jal     $k0

# Retorno de la excepción
# Podría hacerse con eret
        mfc0    $k0, C0_EPC
        j       $k0
        rfe
```

Acceso Directo a Memoria

- ¿Porqué DMA?. Al transferir grandes cantidades de datos la CPU ocupa mucho tiempo en las transferencias.
  - Si la sincronización es por encuesta…
    se pierde mucho tiempo en las esperas
  - Si la sincronización es por interrupción…
    existe un gran overhead en la gestión.
- Casos reales: Si suponemos que en la gestión de interrupciones se consumen 100 ciclos por Byte en un computador moderno (2 GHz)
  - Un dispositivo lento (impresora/teclado)
    transmite a un ritmo máximo de pocos KB/s
  - Un dispositivo medio (pendrive)
    puede alcanzar las decenas de MB/s
  - Un dispositivo rápido (disco duro)
    alcanza los cientos de MB/s

ATC UC

Acceso Directo a Memoria

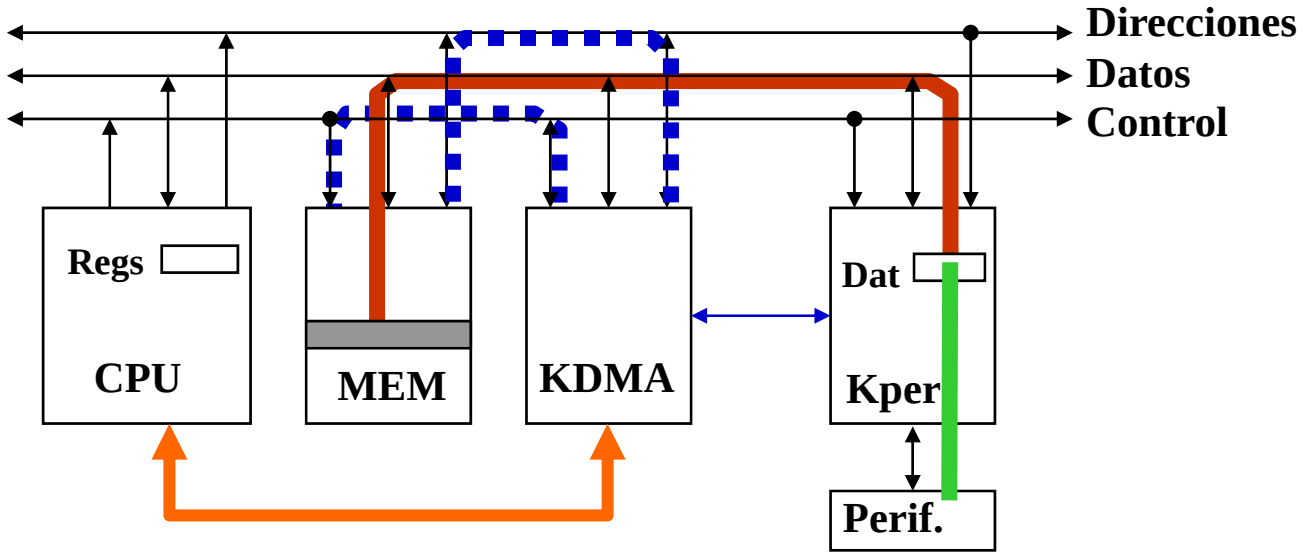


- En el computador:
  - Tiempo de ciclo 2 Ghz → $0.5\ 10^{-9}$s
  - Tiempo en la rutina de interrupción (100 ciclos) → $50\ 10^{-9}$s
- La impresora (500cps):
  - Tiempo entre operaciones de E/S: 500 B/s → $2\ 10^{-3}$s
  - La impresora ocupa el 0,0025% del tiempo de CPU
- Pendrive (20 MB/s):
  - Tiempo entre operaciones de E/S: $20\ 10^{6}$ B/s → $50\ 10^{-9}$s
  - El pendrive ocupa el 100% del tiempo de CPU
- El disco duro (300 MB/s):
  - Tiempo entre operaciones de E/S: $300\ 10^{6}$ B/s → $3.33\ 10^{-9}$s
  - **El disco duro ocupa el 1501% del tiempo de CPU**

ATC UC

Acceso Directo a Memoria

- Acceso Directo a Memoria (DMA):
  - Técnica de gestión de E/S para tratar grandes volúmenes de datos y dispositivos de alta velocidad.
  - Transferencia de datos entre periféricos y memoria sin intervención de la CPU.
  - Necesita un controlador de DMA conectado al bus de sistema.
  - Sólo existe sincronización al finalizar el bloque

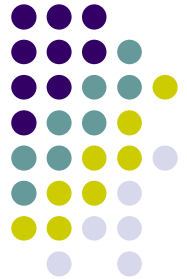


Acceso Directo a Memoria

- El controlador DMA es capaz asumir todo el control del sistema para realizar las operaciones de E/S.
- Necesita dicho control para transmitir datos a y desde, memoria a través del bus de sistema.
- El controlador DMA debe utilizar el bus sólo cuando el procesador no lo necesita o forzar al procesador a que suspenda temporalmente su funcionamiento.

ATC UC

Acceso Directo a Memoria



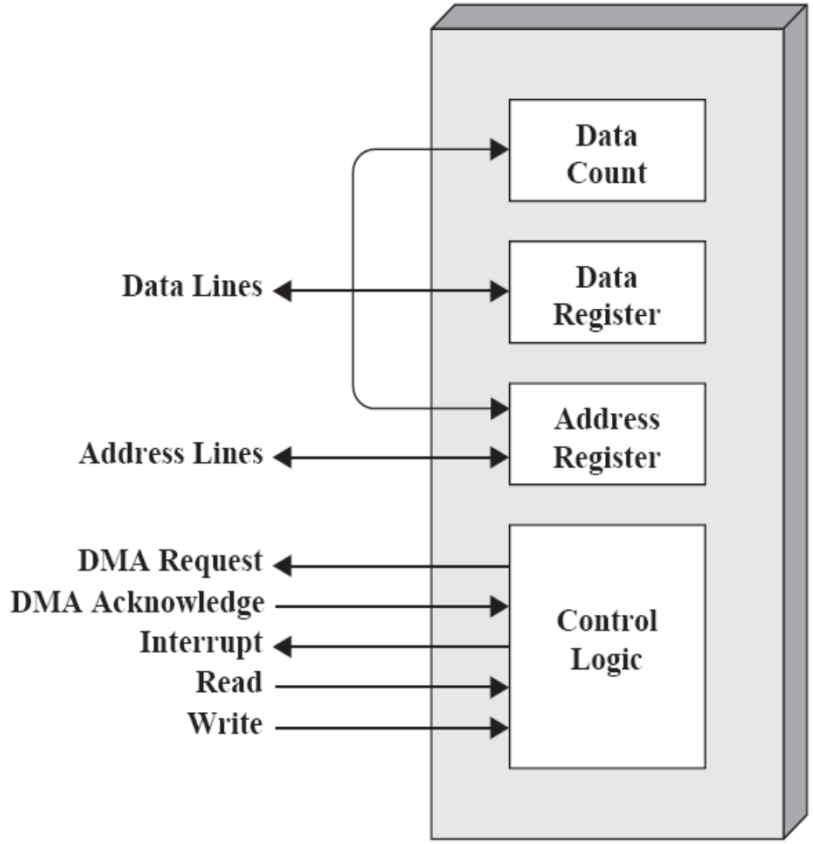
- El DMA ayuda en las operaciones de E/S tanto en el proceso de sincronización como de transferencia de la información
  - El procesador programa el controlador de DMA para que acepte las peticiones del controlador, envía la orden de E/S al controlador y se despreocupa, mientras continúa haciendo otro trabajo
  - **El controlador E/S realizará la operación y al finalizar avisa al DMA**
  - **El DMA realiza la transferencia a memoria y comprueba si se ha transmitido todo el bloque**
  - **Si ha transmitido todo el bloque activa la señal de interrupción**
  - La CPU detecta el cambio en esta señal y realiza las acciones necesarias para que se ejecute la rutina correspondiente que finaliza la operación de E/S
- Igual que en el caso de las interrupciones, las operaciones de sincronización y transferencia son realizadas por **hardware**
- Requiere una **programación inicial del controlador de DMA**

ATC UC

Acceso Directo a Memoria

hardware

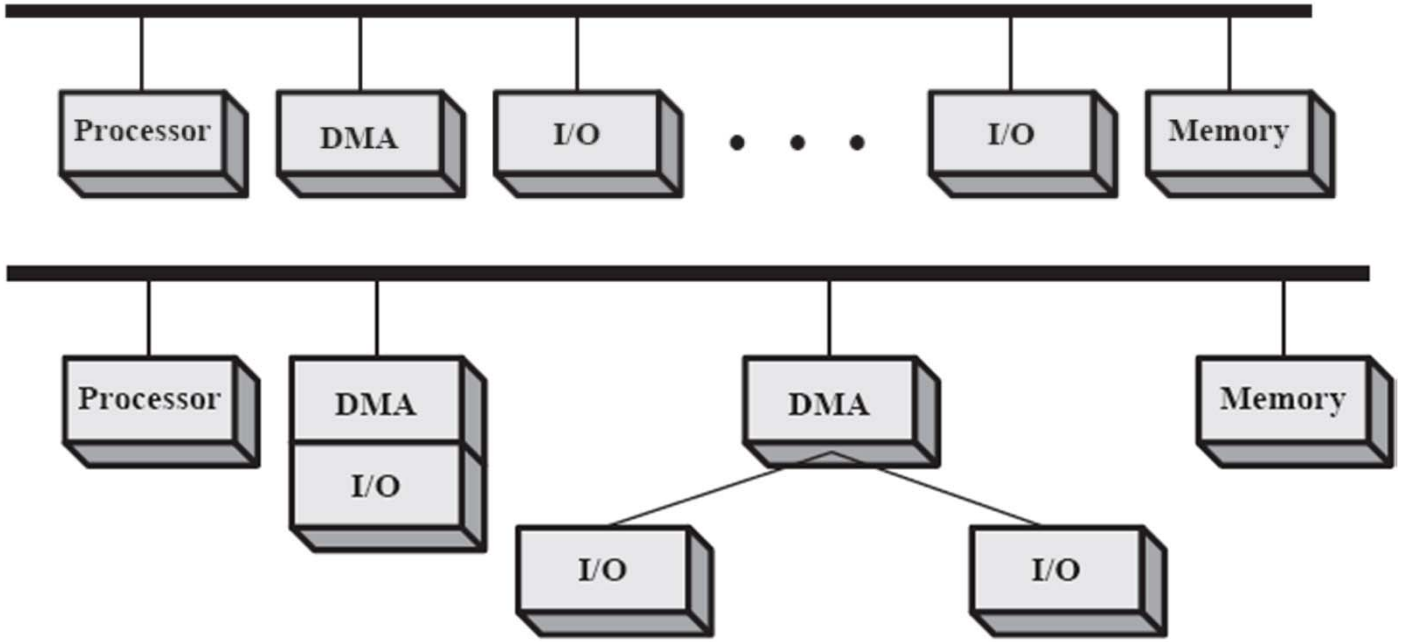
- El controlador de DMA
  - Registros típicos:
    - Contador de datos a transmitir
    - Dirección de los datos
    - Control
    - Status
  - Líneas
    - Petición/reconocimiento de DMA
    - Petición/reconocimiento de bus
    - Las típicas de un bus



Acceso Directo a Memoria

hardware

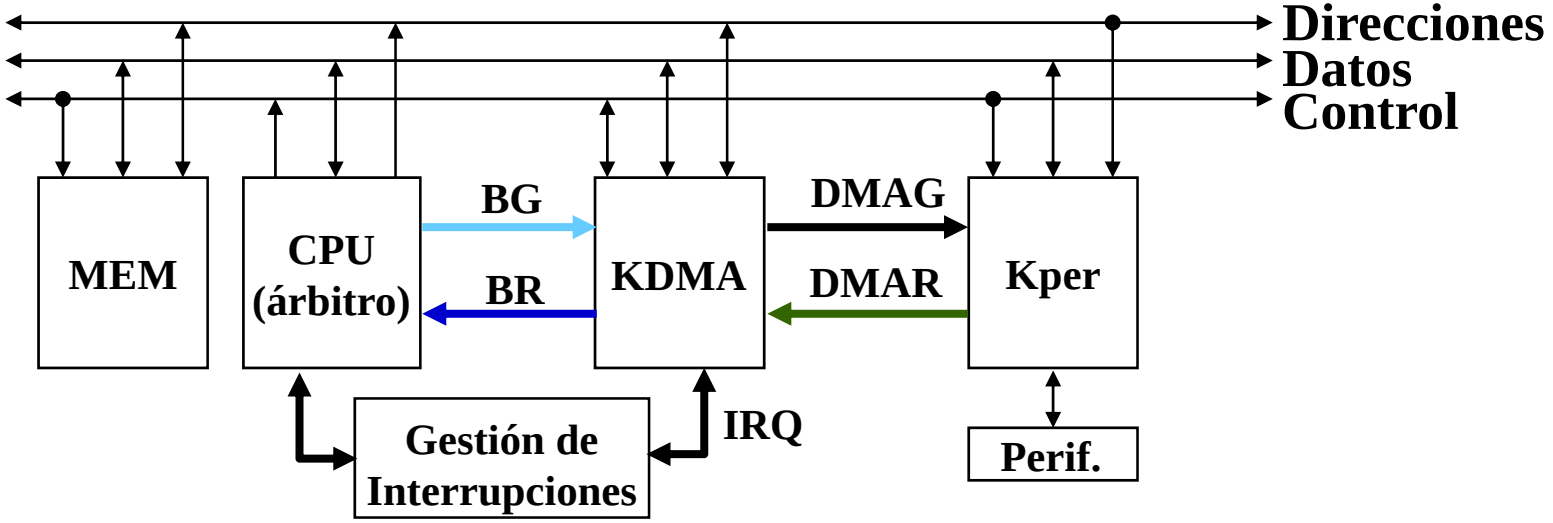
- Conexión del controlador de DMA
  - La estructura más común de conexionado es usando un único bus
  - Todos los módulos conectados al mismo bus de sistema, en estos casos pueden darse varios tipos de arquitectura



Acceso Directo a Memoria

hardware

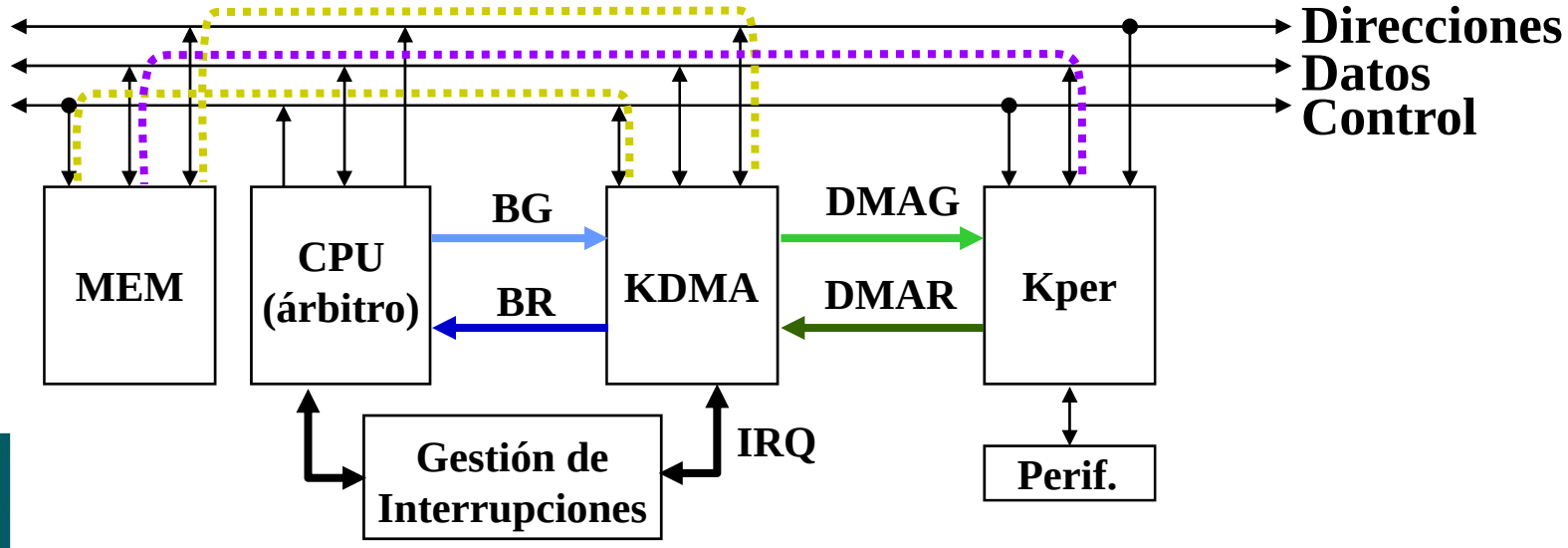
- Operaciones con DMA:
  - Transferencia de datos
    - El KPER avisa al KDMA de la disponibilidad de un dato (DMAR)
    - El KDMA solicita el BUS (BR)
    - El árbitro (en este caso la CPU) concede el BUS (BG)



Acceso Directo a Memoria

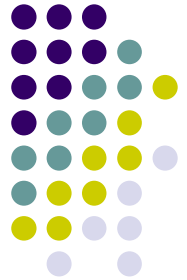
hardware

- Transferencia de datos
  - El KDMA activa las señales de control y avisa al KPER para que deposite el dato sobre el bus (DMAG)
  - Al finalizar la transferencia se desactivan las señales



Acceso Directo a Memoria

hardware

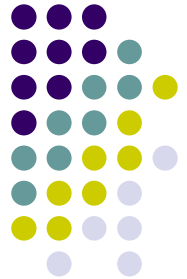


- Esquemas de transferencia por DMA:
  - **Robo de ciclo**:
    - Se basa en usar uno o más ciclos de CPU por cada palabra (o dato) que se transfiera (de ahí el nombre).
    - Se consigue una alta disponibilidad del bus del sistema para la CPU
    - La transferencia de los datos será considerablemente lenta.
    - Este método es el que se usa habitualmente ya que la interferencia con la CPU es muy baja.

ATC UC

Acceso Directo a Memoria

hardware

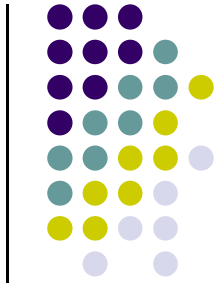


- Esquemas de transferencia por DMA:
  - **DMA por ráfagas**:
    - Consiste en enviar el bloque de datos solicitado mediante una ráfaga, ocupando el bus del sistema hasta finalizar la transmisión
    - Se consigue la máxima velocidad
    - La CPU no podrá usar el bus durante todo ese tiempo, por lo que permanecería inactiva.
  - **DMA transparente:**
    - Se trata de usar el bus del sistema cuando se tiene certeza de que la CPU no lo necesita, como por ejemplo en aquellas fases del proceso de ejecución de las instrucciones donde nunca se usa ya que la CPU realiza tareas internas (v. g. fase de decodificación de la instrucción).
    - De esta manera el DMA permanecerá transparente para la CPU y la transferencia se hará sin obstaculizar la relación CPU-bus del sistema.
    - Como desventaja, la velocidad de transferencia es la más baja posible.

ATC UC

Acceso Directo a Memoria

Programación de DMA



- Cuando se va a realizar una operación de E/S que requiera el uso de un DMA, éste debe programarse indicando:
  - Información de **control**
    - Tipo de operación (lectura/escritura)
    - Esquema de transferencia (Robo de ciclo/ráfagas)
    - Otra información /interrupción
  - El registro de **dirección de los datos** deberá tener la posición inicial de memoria del bloque
  - El **contador de datos a transmitir** deberá tener el número de palabras que hay que leer o escribir
- Al finalizar la transmisión el registro de **status** nos indicará si se ha producido algún error durante la transmisión

ATC UC

Acceso Directo a Memoria

Ejemplo

- **Volvemos al ejemplo del disco**.
- Haciendo la transferencia por DMA
  - Sincronización por interrupciones:
    - El DMA activará la señal de interrupción al terminar de leer un bloque.
  - El **programa principal**, realizará los siguientes pasos:

Esperar la disponibilidad de KDMA y KDISCO
Programar KDMA
    Programar la dirección de memoria y el número de datos
    Programar el modo de operación y activar el controlador
Programar KDISCO
    Enviar el nº de pista y el nº de sector
    Enviar la orden de lectura
Aviso al SO que debemos ESPERAR.
Procesar sector leído.

ATC UC

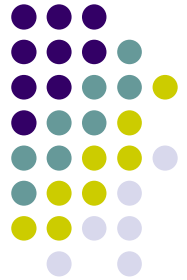
Acceso Directo a Memoria
Ejemplo

- Sincronización por **interrupciones** (Cont).
  - **Rutina de atención**, se activa al finalizar la transmisión:

```
Leer el registro de estado
Si no ha ido bien entonces
    Esperar la disponibilidad de KDMA y KDISCO
    Programar de nuevo el KDMA
        Programar la dirección de memoria y el
            número de datos
        Programar el modo de operación
    Programar de nuevo el KDISCO
        enviar el nº de pista y el nº de sector
        enviar la orden de lectura
Sino Indicar al SO que a finalizado la ESPERA
```

ATC UC

Acceso Directo a Memoria
Ejemplo

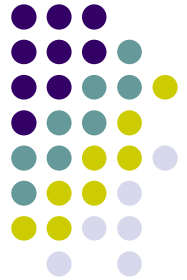


- Supongamos:
  - Que el controlador de DMA tiene cuatro registros:
    - Dirección de memoria (**DMA_MEM**) de tamaño 32 bits y está en la posición 0xbfA00000.
    - Cantidad de datos (**DMA_NDAT**) de tamaño 32 bits y está en la posición 0xbfA00004.
    - Control (**DMA_CONT**) donde se programa el modo de operación y se activa el controlador. Está en la posición 0xbfA00008, ocupa un byte y en nuestro caso debe tomar el valor 0x0E3.
    - Status (**DMA_ST**): El bit 0 indica si el dispositivo está en uso y el bit 1 indica que ha habido algún error en la transferencia. Está en la posición 0xbfA00009, ocupa un byte. La interrupción del DMA se considera atendida al leer este registro.
  - Que existe una rutina de inicialización que programa el controlador de disco para que actúe por DMA.
  - Que existe una rutina de gestión de las interrupciones que determina la fuente y llama a la rutina de atención como una subrutina.

ATC UC

Acceso Directo a Memoria

Ejemplo

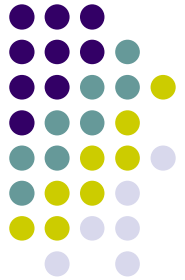


- Vamos a llamar a la rutina TAREAS mientras se recibe todo el sector por DMA.
- Código de la rutina de lectura:

```
#  Función  : leer_sector          Programador: XXXX XXXX XXXX
#  Fecha realización: 1/dic/2005
#  Ultima modificación: 15/dic/2005
#############################################################
# Descripción funcional: Función inicia la lectura de un sector de
# un disco. Espera que se complete la lectura  llamando a TAREAS
#############################################################
# Parámetros: a0 → dirección de la variable buffer
#                 a1 → Número de pista
#                 a2 → Número de sector
#############################################################
# Usa las variable GLOBALES: SECTOR, PISTA y
# BUFFER_INI para que sean usadas por la rutina de atención.
```

Acceso Directo a Memoria

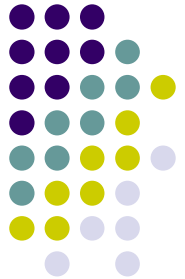
Ejemplo



```
#  registros del controlador de disco
            .equ        R_status, 0xbf900000
            .equ        R_datos, 0xbf900001
            .equ        R_comando, 0xbf900002
#  registros del controlador de DMA
            .equ        DMA_MEM, 0xbfA00000
            .equ        DMA_NDAT, 0xbfA00004
            .equ        DMA_CONT, 0xbfA00008
            .equ        DMA_ST, 0xbfA00009

# Código de la subrutina
Leer_sector:
# Detectar que los controladores estén listos.
            la          $t0, R_status
            la          $t1, DMA_ST
Espera_dis:
            lb          $t2, 0($t0)
            lb           $t3, 0($t1)
            andi        $t2, $t2, 0x01
            andi        $t3, $t3, 0x01
            andi        $t2, $t2, $t3
            beq         $t2, $zero, Espera_dis
```

Acceso Directo a Memoria

Ejemplo



```
# Inicializar las variables globales para la rutina de Interrupción
        la      $t3, SECTOR
        sb      $a2, 0($t3)
        la      $t3, PISTA
        sb      $a1, 0($t3)
        la      $t3, BUFFER
        sw      $a0, 0($t3)

# Programar el controlador de DMA
        la      $t0, DMA_MEM
        sw      $a0, 0($t0)         # Se guarda la dirección del buffer
        la      $t0, DMA_NDAT
        li      $t1, 0x0100         # Tamaño de un sector
        sw      $t1, 0($t0)
        la      $t0, DMA_CONT
        li      $t1, 0x00           # comando de programacion
        sb      $t1, 0($t0)         # del DMA
```

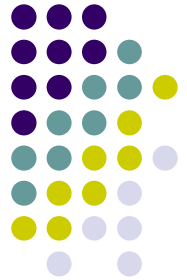
Acceso Directo a Memoria

Ejemplo

```
# Programar el controlador de disco. Enviar al registro de comandos:
# el nº de pista, el nº de sector y la orden de lectura.
        la      $t0, R_datos
        la      $t1, R_comando
        sb      $a1, 0($t0)         # se guarda el número de pista
        li      $t2, 0x10           # comando de número de pista
        sb      $t2, 0($t1)
        sb      $a2, 0($t0)         # se guarda el número de sector
        li      $t2, 0x20           # comando de número de sector
        sb      $t2, 0($t1)
        li      $t2, 0x00           # comando de lectura
        sb      $t2, 0($t1)

# Supondremos que la llamada al SO que debemos esperar es:
        li      $v0,25
        syscall

# fin de la subrutina
# el sector será procesado en el programa principal
        jr      $ra
```

Acceso Directo a Memoria

Ejemplo



- Código de la Rutina de Atención:

```
#  Función  : Atencion_DMA      Programador: XXXX XXXX XXXX
#  Fecha realización: 1/dic/2005
#  Ultima modificación: 15/dic/2005
#########################################################
# Descripción funcional: Esta rutina se activa cada vez que el
# controlador de DMA ha leído un sector. La rutina comprueba si
# ha habido errores y en caso de haberlos reintenta.
#
# Esta subrutina es llamada por el gestor de interrupciones.
#########################################################
# Usa las variable GLOBALES: SECTOR, PISTA y BUFFER.
```

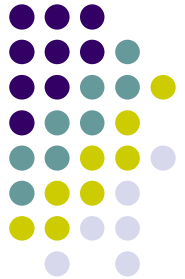
Acceso Directo a Memoria

Ejemplo

```
# Código de la subrutina
Atencion_DMA:
# Salvar los registros que se usen en el stack ($t0-$t3).
        addiu   $sp, $sp,  -16
        sw      $t0, 0($sp)
        sw      $t1, 4($sp)
        sw      $t2, 8($sp)
        sw      $t3, 12($sp)
# Leer el registro de estado del DMA para ver si la operación ha ido bien.
        la      $t0,DMA_ST
        lb      $t1, 0($t0)
# Si hay error Reintentar.
        andi    $t1, $t1, 0x02
        bne     $t1, $zero, Repetir
# Leer el registro de estado del disco para ver si la operación ha ido bien.
        la      $t0, R_status
        lb      $t1, 0($t0)
# Si hay error Reintentar.
        andi    $t1, $t1, 0x04
        bne     $t1, $zero, Repetir
# Todo Correcto.
# Supondremos que la llamada al SO que ha finalizado la espera es:
        li      $v0,26
        syscall
        b       Fin_Rutina
```

Acceso Directo a Memoria

Ejemplo



```
#Ha habido errores hay que reintentar
Repetir:
# Detectar que los controladores estén listos.
            la      $t0, R_status
            la      $t1, DMA_ST
Espera_dis:
            lb      $t2, 0($t0)
            lb       $t3, 0($t1)
            andi    $t2, $t2, 0x01
            andi    $t3, $t3, 0x01
            andi    $t2, $t2, $t3
            beq     $t2, $zero, Espera_dis

# Programar el controlador de DMA
            la      $t0, DMA_MEM
            la      $t3, BUFFER
            lw      $t2, 0($t3)
            sw      $t2, 0($t0)         # Se guarda la dirección del buffer
            la      $t0, DMA_NDAT
            li      $t1, 0x0100         # Tamaño de un sector
            sw      $t1, 0($t0)
            la      $t0, DMA_CONT
            li      $t1, 0x00           # comando de programacion
            sb      $t1, 0($t0)         # del DMA
```

Acceso Directo a Memoria

Ejemplo

```
# Enviar al registro de comandos:
# el nº de pista, el nº de sector y la orden de lectura.
            la      $t0, R_datos
            la      $t1, R_comando
            la      $t3, PISTA
            lb      $t2, 0($t3)
            sb      $t2, 0($t0)          # se guarda el número de pista
            li      $t2, 0x10            # comando de número de pista
            sb      $t2, 0($t1)
            la      $t3, SECTOR
            lb      $t2, 0($t3)
            sb      $t2, 0($t0)          # se guarda el número de sector
            li      $t2, 0x20            # comando de número de sector
            sb      $t2, 0($t1)
            li      $t2, 0x00            # comando de lectura
            sb      $t2, 0($t1)
# fin de la subrutina
Fin_Rutina:
# Recuperar los registros salvados del stack ($t0-$t3).
            lw      $t0, 0($sp)
            lw      $t1, 4($sp)
            lw      $t2, 8($sp)
            lw      $t3, 12($sp)
            addiu   $sp, $sp,  16
# Vuelta al gestor de interrupciones
            jr      $ra
```