Lezione 15

Driver delle periferiche

Sistemi operativi

26 giugno 2012

Marco Cesati

System Programming Research Group
Università degli Studi di Roma Tor Vergata

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Di cosa parliamo in questa lezione?

Driver ed interfaccie delle periferiche di I/O

1. Le periferiche di I/O
2. Le interfaccie di I/O
3. Driver basati su polling
4. Driver basati su interruzioni
5. Driver basati su DMA

Driver delle periferiche

Marco Cesati

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Periferiche

Con il termine *dispositivi periferici* (o brevemente *periferiche*) si intendono i componenti del calcolatore che realizzano funzioni aggiuntive rispetto alla semplice esecuzione dei programmi

La definizione non tragga in inganno: il ruolo delle periferiche in un calcolatore è di vitale importanza! Ad esempio, sono periferiche:

- i dispositivi di memorizzazione di massa come i dischi rigidi
- tutti i dispositivi che realizzano i sistemi di ingresso e uscita
- componenti hardware inclusi all'interno del calcolatore, ad esempio l'orologio interno del calcolatore (Real Time Clock)

In pratica, spesso si confonde il significato del termine **periferica** con quello di **dispositivo di I/O**: in fondo, lo scopo di ogni periferica è scambiare segnali con il processore centrale

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Bus di sistema

Il metodo più semplice per collegare al processore i dispositivi periferici consiste nell'utilizzare un singolo bus di sistema

Il *bus* è un fascio di linee di comunicazione, ciascuna in grado di trasportare un bit alla volta, che collega due o più sistemi o unità funzionali

Un tipico bus è costituito da 50–100 linee di comunicazione che trasportano:

- i dati, cioè i segnali da trasferire
- gli indirizzi di celle di memoria centrale o di periferiche in cui i dati vanno letti o scritti
- segnali vari di controllo e gestione

Spesso si parla di *bus dati*, *bus indirizzi* e *bus controllo* per indicare le varie porzioni del bus di sistema

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Spazio di indirizzamento di I/O

Ogni periferica connessa ad un bus deve essere dotata di uno o più *indirizzi*, ossia identificatori numerici che consentono di selezionare senza ambiguità la periferica come sorgente o destinazione di un trasferimento di dati sul bus

Si definisce *spazio di indirizzamento di I/O* l'insieme dei valori numerici che possono essere potenzialmente utilizzati per selezionare una periferica collegata ad un bus

Esistono fondamentalmente due tipi di indirizzi di I/O:

- indirizzi di I/O *isolati*: il processore è dotato di istruzioni macchina specifiche per trasferire dati con le periferiche, e lo spazio di indirizzamento di I/O è separato ed indipendente rispetto allo spazio di indirizzamento della memoria centrale
- indirizzi di I/O *mappati in memoria* (*memory mapped I/O*): il processore può utilizzare le stesse istruzioni macchina sia per accedere alla memoria centrale che per trasferire dati con le periferiche; memoria e periferiche condividono lo stesso spazio di indirizzamento

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

La gerarchia di interfaccie di una periferica

Ciascuna periferica fa uso di diversi livelli di interfaccie che la collegano al resto del sistema di calcolo

- L'interfaccia hardware è realizzata dai circuiti hardware della periferica e consente il collegamento agli altri circuiti del calcolatore elettronico
- L'interfaccia logica è realizzata dai circuiti hardware della periferica e consente la sua programmazione mediante una sequenza ben definita di operazioni
- L'interfaccia software è realizzata dal SO per consentire l'uso della periferica da parte dei programmi del kernel e delle applicazioni utente

Driver di una periferica

I programmi del SO che pilotano la periferica e ne consentono il normale funzionamento; possono essere considerati una parte della sua interfaccia software

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Interfaccia hardware di una periferica

Ogni dispositivo periferico è collegato al bus tramite un circuito chiamato *interfaccia hardware*

L'interfaccia hardware di un dispositivo svolge tutte le funzioni di *adattamento* necessarie per collegare la periferica al bus, ad esempio:

- conversione di segnali elettrici
- collegamento di dispositivi con temporizzazioni differenti
- realizzazione di una interfaccia logica per programmare la periferica

In che cosa consiste l'interfaccia logica di una periferica?

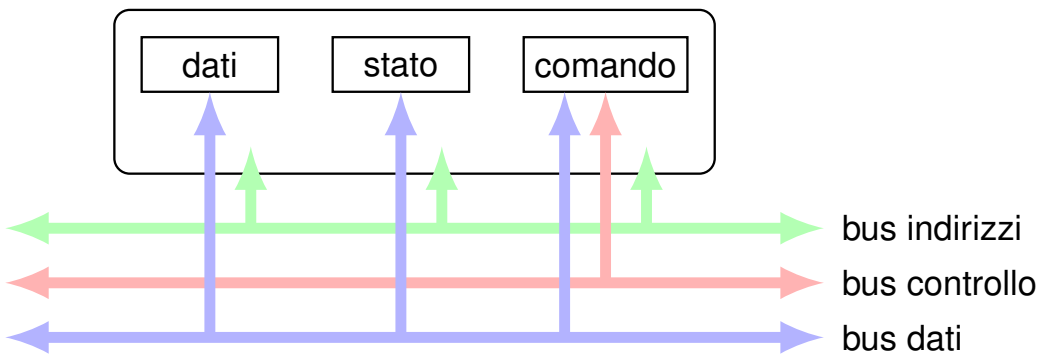
Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Interfaccia logica di una periferica

Dal punto di vista logico, l'interfaccia di un dispositivo permette di trasferire dati in alcuni *registri di interfaccia* (o *porte*):

- uno o più *registri di dati* per la memorizzazione dei dati
- uno o più *registri di comando* per specificare le operazioni
- uno o più *registri di stato* per memorizzare flag che indicano lo stato d'esecuzione delle operazioni della periferica

Spesso uno stesso registro svolge due o più funzioni, ad esempio contemporaneamente di comando e di stato



Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Esempio: interfaccia logica della tastiera

La tastiera di un calcolatore è un tipico dispositivo estremamente lento in confronto al processore: produce eventi con una frequenza diversi miliardi di volte inferiore a quella della CPU

Nei calcolatori con architettura Intel o compatibili, l'interfaccia logica della tastiera AT o PS/2 utilizza due sole porte con indirizzi di I/O isolati:

$(60)_{16}$ registro di dati, utilizzato sia in lettura (ad es., per ottenere il codice del tasto premuto) sia in scrittura (ad es., per impostare la velocità di ripetizione automatica)

$(64)_{16}$ registro di comando e di stato, utilizzato per scrivere sulla periferica un comando e per leggere i flag di stato (ad es., la disponibilità di un codice tasto ancora da leggere)

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Interfaccia software di una periferica

- L'interfaccia software di una periferica è realizzata dal SO
- È anche nota come interfaccia di I/O
- Collega la periferica alle applicazioni utente
  - definisce una modalità tramite la quale l'applicazione può utilizzare la periferica
- Consente anche l'utilizzo della periferica da parte di altri programmi del kernel del SO

Per ciascuna periferica funzionante, il SO deve disporre del **driver** per pilotarla e di una **interfaccia** di I/O che consenta di richiedere i suoi servizi

Il driver della periferica può essere considerato come esterno all'interfaccia di I/O oppure un suo componente

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Tipologie di interfaccie di I/O delle periferiche

Le interfaccie di I/O delle periferiche si distinguono per diverse caratteristiche, ad esempio:

- Tipo di trasferimento:
  - a carattere: singoli byte (es.: mouse, tastiera)
  - a blocchi: gruppo di byte di dimensione prefissata (es.: disco rigido, scheda sonora)
- Tipo di accesso:
  - sequenziale: i dati trasferiti non sono selezionabili (es.: mouse, tastiera, scheda sonora, nastro, scheda di rete)
  - diretto: i dati trasferiti sono selezionabili (es.: disco magnetico, disco ottico)
- Prevedibilità dei trasferimenti:
  - sincrono: è possibile prevedere i tempi dei trasferimenti (es.: unità a nastro)
  - asincrono: non è possibile prevederli (es.: mouse, tastiera)
- Direzione dei trasferimenti
  - sola lettura (es.: disco ottico)
  - sola scrittura (es.: scheda video)
  - lettura e scrittura (es.: scheda di rete)

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

I device file dei sistemi Unix

- I sistemi di tipo Unix fanno uso dei *device file*: sono un particolare tipo di file che rappresenta l'interfaccia di I/O di una periferica del sistema
- L'apertura (`open()`) di un device file permette ad un processo di interagire direttamente con la periferica
- La lettura (`read()`) e la scrittura (`write()`) sul device file attivano trasferimenti di dati da e verso la periferica
- Operazioni diverse che non comportano il trasferimento di dati sono realizzate dalla chiamata di sistema `ioctl()`
- Un device file è caratterizzato da:
  - Il tipo: *a blocchi* ("`b`") o *a carattere* ("`c`")
  - Il *major number*: identifica il driver della periferica
  - Il *minor number*: identifica uno tra i diversi dispositivi pilotabili dal driver

Il nome o percorso del device file non è caratterizzante!

Driver delle periferiche
Marco Cesati

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Esempi di device file in Linux

Device file a blocchi

(b,8,0)	1° disco SATA/SCSI	`/dev/sda`
(b,8,18)	1ª partizione del 2° disco	`/dev/sdb2`
(b,2,0)	1° floppy disk	`/dev/fd0`
(b,11,0)	1° CD-ROM SATA/SCSI	`/dev/sr0`
(b,22,0)	disco master della 2ª catena IDE	`/dev/hdc`

Device file a carattere

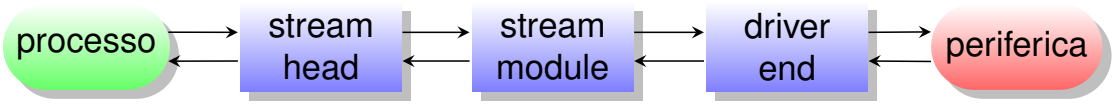
(c,4,64)	1ª porta seriale	`/dev/ttyS0`
(c,14,3)	processore della scheda sonora	`/dev/dsp`
(c,1,5)	generatore di zeri	`/dev/zero`
(c,1,8)	generatore pseudo-casuale	`/dev/random`

I device file in Linux sono realizzati assegnando ai metodi degli oggetti del VFS funzioni specifiche del driver della periferica

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

STREAMS

Il SO Unix System V definisce una interfaccia modulare tra processi e dispositivi chiamata *STREAMS*



Uno *stream* è una connessione full-duplex tra un processo ed una periferica costituita da

- *stream head*: elemento iniziale d'interfaccia con il processo
- *driver end*: elemento finale d'interfaccia con la periferica
- *stream modules*: un numero variabile di moduli di elaborazione intermedi

Tra ciascun elemento ed il successivo sono definite due code di messaggi (in scrittura ed in lettura)

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

STREAMS (2)

- È possibile inserire uno stream module tramite la chiamata di sistema `ioctl()`
- Per leggere dati dal dispositivo: `read()` o `getmsg()`
- Per scrivere dati sul dispositivo: `write()` o `putmsg()`
- `putmsg()` e `getmsg()` consentono la trasmissione di singoli messaggi, conservandone i confini
- Le code di messaggi tra i vari moduli possono far uso di controllo di flusso (con eventuale bufferizzazione)

Qual è il vantaggio offerto dagli STREAMS?

È possibile definire driver complessi in modo stratificato e riutilizzando i moduli

Ad esempio, un modulo di un driver per una scheda di rete Ethernet potrebbe essere riutilizzato per un driver per una scheda di rete Token Ring

Molti SO della famiglia Unix fanno uso di STREAMS (Linux invece non lo implementa)

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Gestione delle periferiche

Supponiamo di dover leggere il codice del prossimo tasto battuto sulla tastiera: possiamo semplicemente eseguire una istruzione macchina che legge un valore dalla porta del registro dati $(60)_{16}$ *?* ***No!***

Se la lettura fosse "bloccante", il processore sarebbe fermo in attesa dell'azione dell'utente; se invece fosse "non bloccante", come potremmo distinguere tra vecchi e nuovi eventi?

In generale tutte le periferiche hanno tempi di risposta e di servizio di molti ordini di grandezza peggiori del processore, quindi i trasferimenti dei dati tra processore e periferiche non possono essere effettuati con la stessa modalità che in quelli verso la memoria centrale

Esistono due tipi fondamentali di driver di periferica:

- con controllo di programma
- basato su interruzioni

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Gestione a controllo di programma

La *gestione a controllo di programma* è basata sul controllo periodico da parte del driver di un flag in un registro di stato della periferica che segnala la disponibilità di un dato o la possibilità di eseguire una nuova operazione

Questa modalità è anche conosciuta come *polling* oppure *programmed I/O* (PIO)

Quale è il vantaggio di questa modalità rispetto ad una procedura in cui il processore viene bloccato in una istruzione macchina fino a che la periferica risulta pronta?

Il vantaggio è che nell'intervallo di tempo tra una interrogazione e l'altra del flag di stato il processore è disponibile per eseguire altre istruzioni (eventualmente di altri processi in esecuzione)

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Esempio: driver della tastiera con polling

Frammento di codice per richiedere il codice del prossimo tasto premuto sulla tastiera tramite polling:

```
wait:
        ...                 # altre operazioni
        inb   $0x64         # %al ← [registro di stato]
        andb  $0b10,%al     # controlla il flag IBF
        jz    wait          # salta se IBF è zero
        inb   $0x60         # %al ← [registro dati]
```

Il flag IBF (Input Buffer Full) in seconda posizione del registro di stato vale 1 se un nuovo codice tasto è disponibile nel registro dati; la periferica pone a 0 il flag automaticamente dopo una lettura del registro dati

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Gestione con interruzione

La modalità di gestione delle periferiche tramite polling è:

- semplice da realizzare in hardware
- costosa in termini di organizzazione del sistema operativo e di efficienza complessiva

Prestazioni migliori si potrebbero ottenere se una periferica potesse “informare” il processore che una certa condizione si è verificata *in modo indipendente dalle istruzioni macchina effettivamente eseguite dal processore*

Si definisce *interruzione* un segnale inviato da una periferica al processore per indicare il verificarsi di un evento (ad esempio, il completamento di una operazione o la disponibilità di nuove informazioni)

Quasi tutti i componenti di un calcolatore moderno sono in grado di generare interruzioni, perciò questa modalità di gestione delle periferiche è utilizzata frequentemente

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Circuiti del meccanismo di interruzione

I componenti fondamentali del meccanismo di interruzione:

- Una linea di controllo del bus dedicata alla trasmissione di segnali di interruzione originati dalle periferiche e destinati al processore: è chiamata *linea di richiesta di interruzione*, od anche *linea di IRQ* (Interrupt ReQuest)
- Una linea di controllo del bus dedicata alla trasmissione di segnali di *conferma dell’interruzione* (o *linea di ACK*, Acknowledge) originati dal processore e diretti verso le periferiche
- Circuiti all’interno dell’interfaccia della periferica per abilitare la generazione di segnali di interruzione in predeterminati casi
- Circuiti all’interno del processore per interrompere l’esecuzione del programma in esecuzione e passare all’esecuzione di una procedura predefinita ogni volta che la linea di richiesta di interruzione viene asserita

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Schema di driver basato su interruzioni

La sequenza logica di operazioni eseguite per gestire una periferica con il meccanismo con interruzioni:

1. Il driver scrive opportuni comandi sull'interfaccia della periferica per programmare una determinata operazione e abilitare la generazione di una interruzione alla sua conclusione
2. Il SO passa ad eseguire altri programmi
3. La periferica svolge l'operazione richiesta
4. Quando la periferica termina l'operazione, asserisce la linea di IRQ
   - in questa fase si dice che la richiesta di interruzione è *pending*
5. I circuiti del processore dedicati al controllo della linea di IRQ rilevano il cambio di stato e interrompono l'esecuzione del programma in esecuzione, salvando le informazioni necessarie per riprendere la sua esecuzione in futuro

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Schema di driver basato su interruzioni (2)

6. Il processore passa ad eseguire una procedura del SO chiamata *gestore dell'interruzione* (*interrupt handler*)
   - il suo primo compito è asserire la linea di ACK dell'interruzione
   - L'interfaccia della periferica osserva la linea di ACK: quando essa viene asserita, l'interfaccia de-asserisce la linea di IRQ
7. Il processore passa ad eseguire una procedura del driver della periferica chiamata *Interrupt Service Routine* (o ISR)
   - essa interroga l'interfaccia della periferica e svolge le azioni opportune (ad esempio, programma l'operazione successiva)
8. Al termine dell'ISR il controllo torna alla procedura gestore dell'interruzione, che esegue una istruzione speciale "ritorno da interruzione" (ad esempio, `iret` in IA-32)
9. Il processore ritorna ad eseguire il programma interrotto

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Disabilitazione delle interruzioni

Il processore opera in modo da evitare interruzioni all'interno delle singole istruzioni macchina (il controllo per l'esistenza di una interruzione pendente è eseguito logicamente solo tra la fine dell'esecuzione di una istruzione e l'inizio di un'altra)

Questo meccanismo è sufficiente ad evitare problemi? ***No!***

Esempi di frammenti di programma che non dovrebbero essere interrotti:

- una sequenza di istruzioni che accede alle porte di un dispositivo hardware
- una sequenza di istruzioni che accede ad una struttura di dati utilizzata anche da gestori di interruzione

Tutti i processori dispongono dunque di un meccanismo per *disabilitare* le interruzioni

Nei processori con architettura IA-32 l'esecuzione delle interruzioni è controllata da un flag IF nel registro di stato impostato dalle istruzioni `cli` e `sti`

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Disabilitazione delle interruzioni (2)

Le interruzioni possono essere selettivamente disabilitate anche in altri modi:

- programmando il controllore di interruzione per disabilitare una o più linee di ingresso
- definendo un *livello corrente di priorità* per il processo in esecuzione, e programmando il controllore di interruzione in modo da impedire la generazione di interruzioni aventi priorità troppo bassa
- programmando l'interfaccia di una periferica per impedire la generazione dei segnali di interruzione

La gestione degli eventi asincroni dovuti alle interruzioni è un compito molto difficile: gran parte della complessità nel progetto di un sistema operativo è dovuta proprio alla esistenza delle interruzioni

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Trasferimento memoria-periferica

Il calcolatore elettronico utilizza sempre la memoria primaria come area di memorizzazione per dati e programmi

Consideriamo ad esempio una applicazione “media player” che accede ad un file in formato MPEG (su disco rigido o DVD) e riproduce il relativo flusso audio e video

Generalmente l’applicazione suddivide il proprio lavoro in due fasi distinte (che possono anche svolgersi sovrapponendosi temporalmente):

- l’applicazione copia alcuni blocchi di dati dal file in memoria primaria
- l’applicazione accede ai blocchi di dati in memoria primaria per decodificare i flussi audio e video e riprodurli

Quante volte viene trasferito ciascun dato del file MPEG?

Se il processore deve sovraintendere a tutti i trasferimenti, almeno tre volte: da disco a CPU, da CPU a memoria primaria, da memoria primaria a CPU

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Accesso diretto alla memoria primaria

Si potrebbe pensare che la memorizzazione in memoria primaria è inutile e potrebbe essere evitata: ad esempio, una applicazione “media player” potrebbe trasferire i dati da memoria secondaria ai registri del processore ed elaborare direttamente i dati nei registri

È una soluzione praticabile? ***No!***

Non è possibile evitare l’utilizzo della memoria primaria a causa della lentezza dei dispositivi di memoria secondaria e della limitata capacità di memorizzazione dei registri della CPU

La soluzione adottata per ridurre il numero di trasferimenti è l’accesso diretto alla memoria da parte delle periferiche:

Definizione di DMA

Si definisce *accesso diretto alla memoria* (o DMA, Direct Memory Access) un meccanismo che consente ad una periferica di trasferire dati da o verso la memoria primaria senza intervento diretto del processore

Driver delle periferiche
Marco Cesati

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Controllore di DMA

Il *controllore di DMA* è un dispositivo che effettua i trasferimenti di dati con accesso diretto alla memoria

È ovvio che ogni trasferimento di dati coinvolge sempre l'utilizzo del bus (o dei bus) a cui sono collegati periferica e memoria centrale

- In bus di concezione più vecchia o più semplice (ad esempio, ISA), il controllore di DMA è un processore ausiliario che svolge solo le operazioni di trasferimento
- Nei bus più recenti (ad esempio, PCI), le periferiche possono accedere direttamente alla memoria grazie a circuiti che implementano un controllore di DMA dedicato

In altri termini, le periferiche di bus recenti come PCI possono assumere il ruolo di unità master per il controllo del bus

Per questo motivo l'accesso diretto alla memoria è anche conosciuto come *bus mastering*

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Struttura di un controllore di DMA

In generale, un controllore di DMA è in grado di programmare più trasferimenti alla volta

Ad esempio, l'interfaccia di un controllore per dischi rigidi in grado di pilotare due dischi è anche in grado di programmare due trasferimenti DMA alla volta (uno per ciascun disco)

Si definisce *canale DMA* l'insieme di registri e circuiti di controllo dedicati ad un singolo trasferimento; il controllore di dischi ha quindi 2 canali DMA

L'interfaccia di ciascun canale DMA possiede:

- Un registro per la posizione dei dati da trasferire sulla periferica (può essere assente)
- Un registro per l'indirizzo dei dati in memoria centrale
- Un registro per la quantità di dati da trasferire
- Un registro di comandi (ad esempio per indicare la direzione del trasferimento e per avviarla)
- Un registro di stato, spesso unificato con il registro di comandi (ad esempio, include un flag che indica se l'operazione di trasferimento è terminata)

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Schema di funzionamento di un driver con DMA

Supponiamo che il sistema operativo debba copiare un blocco di dati da un disco rigido su bus PCI ad un buffer in memoria:

1. Il driver interagisce con l'interfaccia del canale DMA relativo al disco in cui è memorizzato il blocco di dati:
   a) Invia la posizione del blocco di dati sul disco rigido
   b) Invia l'indirizzo del buffer in memoria centrale
   c) Invia la dimensione del blocco di dati, cioè la dimensione del trasferimento
   d) Imposta un flag di direzione per indicare che si vuol leggere i dati dal disco
   e) Imposta un flag per indicare che, al termine dell'operazione di trasferimento, si vuol ricevere una interruzione
   f) Avvia l'operazione di trasferimento

   Fatto questo, il SO passa ad eseguire altri compiti

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA

Schema di funzionamento di un driver con DMA (2)

2. Il controllore del disco fa richiesta per diventare unità master
3. Quando il controllore del disco assume il controllo del bus PCI, imposta i segnali di controllo per richiedere una operazione di scrittura verso la memoria principale
4. Il trasferimento avviene seguendo il protocollo di trasferimento del bus; appositi circuiti nei bridge che collegano il bus PCI al Front Side Bus propagano i dati fino ai circuiti integrati della memoria
5. Quando il trasferimento è terminato, il controllore del disco genera una interruzione per il processore e diventa disponibile a rilasciare il controllo del bus
6. Il processore riceve l'interruzione ed esegue una ISR definita dal driver che
   a) verifica l'esito dell'operazione
   b) notifica l'avvenuto trasferimento al componente del SO che aveva richiesto i dati (tipicamente una cache del disco)

Schema della lezione
Periferiche
Interfaccie di I/O
Polling
Interruzioni
DMA