Lezione 12
Tecniche di pilotaggio dei dispositivi di I/O

http://www.dii.unisi.it/~giorgi/didattica/arcal1

Gestione dell'Input/Output

- Le periferiche si differenziano notevolmente fra loro perche'
  - Trasferiscono differenti quantita' di dati
  - Funzionano a velocita' diverse
  - Utilizzano 'formati di dato' differenti
- Tipicamente funzionano a velocita' piu' basse rispetto a CPU e Memoria
- Per interfacciarsi con il resto del sistema tipicamente una periferica ha bisogno di un "controller" o "interfaccia"

Velocita' di trasferimento tipiche per dispositivi di I/O

Dispositivo	Vel.Trasferimento (MB/s)
Tastiera	0.00001
Mouse	0.0001
Modem 56k	0.007
Canale telefonico	0.008
Doppia Linea ISDN	0.016
Stampante Laser	0.1
Scanner	0.4
Ethernet classica	1.25
USB (Universal Serial Bus) (2.5W)	1.5
Cinepresa Digitale	4
Disco IDE	5
CD-ROM 40x	6
Fast Ethernet	12.5
Bus ISA	16.7
Firewire (IEEE 1394) (15W)	50
Monitor XGA	60
Rete SONET OC-12	78
Disco SCSI Ultra2	80
Gigabit Ethernet	125
Conventional PCI 32-bit/33MHz	133
Nastro Ultrium	320
Bus PCI-X (1998)	533
10xGigabit Ethernet (2002)	1250
HyperTransport 1.0 (2001)	6400
100xGigabit Ethernet (2010)	12500
Bus PCI-E (2004)	16000
QuickPath (QPI)	25600
HyperTransport 3.1 (2008)	51200

- **USB**
  - v1 (low speed) 1.5Mb/s=187 KB/s
  - v1.1 (full speed) 12Mb/s=1.5MB/s
  - v2.0 (hi-speed,2001) 480Mb/s=60MB/s
  - v3.0 (super speed,2009,4.5W) 5Gb/s=625MB/s
- **SATA**
  - 1.5Gb/s (2001) → 130MB/s(HDD),200MB/s(SSD)
  - 3.0Gb/s (2001) → 200MB/s (reale)
  - 6.0Gb/s (2008) → 400MB/s (reale)
- **ETHERNET**
  - classica (1980) 10 MB/s = 1.25MB/s
  - fast (1995) 100MB/s = 12.5MB/s
  - Giga (1999) 1Gb/s = 125MB/s
  - 10xGiga (2002) 10Gb/s = 1.25GB/s
  - 100xGiga (2010) 100Gb/s = 12.5GB/s
- **PCI-EXPRESS (PCI-E o PCIe)**
  - v1x (2004) 250MB/s(1lane),4GB/s(16-lane)
  - v2.0 (2007) 500MB/s(1lane),8GB/s(16-lane)
  - v3.0 (2010) 1GB/s(1lane),16GB/s(16-lane)

Requisiti di Banda nel Multimedia

- Reduced Quality Audio (Mono)
  - (11,050 audio samples / sec) (8-bit samples) (1 channel) = 0.1 Mbps (11 kB/s)
- High Quality Audio (Stereo)
  - (44,100 audio samples / sec) (16-bit samples) (2 channels) = 1.4 Mbps (176 kB/s)
- Reduced Quality Video
  - QVGA: 320x240@15fps (16-bit color / pixel) = 18 Mbps (2.2 MB/s)
- High Quality Video
  - VGA: 640x480@30fps (24-bit color / pixel) = 221 Mbps (27.6 MB/s)

- La compressione cambia radicalmente queste quantita'…
  - H.264 encode
    - HD720p→1280x720@30fps(24bit)=633Mbps→14-56Mbps+100Gop/sec
    - HD1080p→1920x1080@30fps(24bit)=1424Mbps→20-80Mbps
  - Digital TV
    - 2004: 9000op/pixel →450 Gop/s, 2008: 18000 op/pixel →900 Gop/s

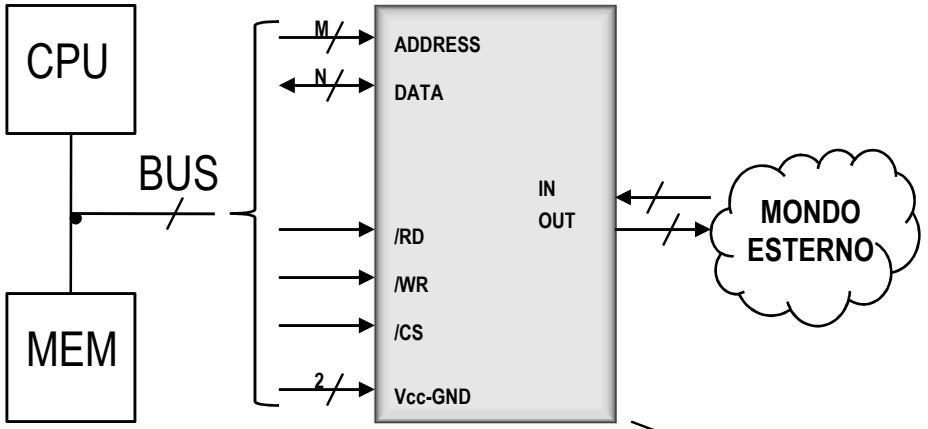
fps=frame/sec

Architettura del Sistema di I/O

- L'architettura del sistema di I/O e'…
  l'interfaccia fra il dispositivo e il processore
- I dispositivi di I/O sono costituiti da
  - Componenti sensori/attuatori (e.g. tastiera, display, disco, …)
  - Componenti elettronici (controller del dispositivo)
  - Componenti software (il "device driver")
- Compiti del controller
  - Eventuale controllo di piu' dispositivi
  - Convertire il flusso seriale di bit in BLOCCHI di byte
  - Effettuare eventuali correzioni di errore
  - Rendere disponibili i dati alla memoria principale
    o prelevare i dati dalla memoria principale
- Compiti del device driver
  - Inizializzare il controller/dispositivo
  - Gestire le operazioni fra controller/dispositivo e processore

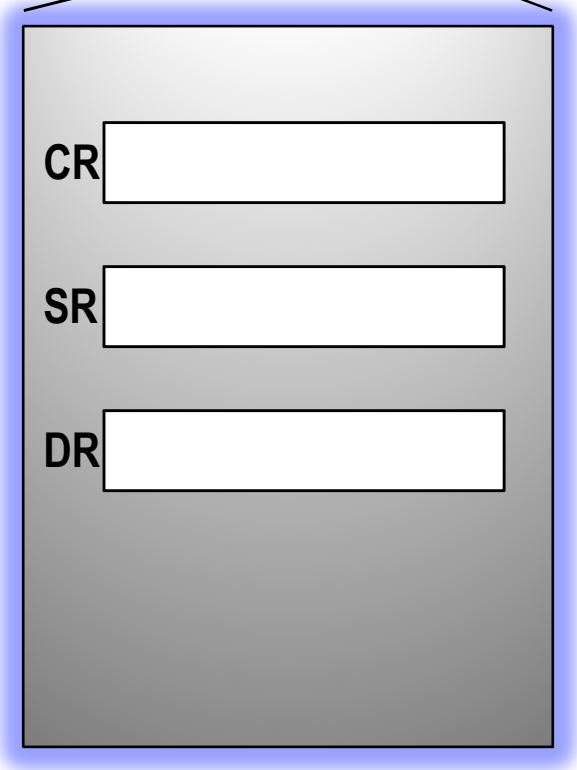
Nota: nel seguito si parlara' di "dispositivo" sottintendendo che ci si riferisce sempre al "controller del dispositivo"

Visione elettrica e logica del dispositivo



RD=READ
WR=WRITE
CS=CHIP SELECT

Il dispositivo viene visto dal processore come un insieme di locazioni dette "PORTE di I/O"



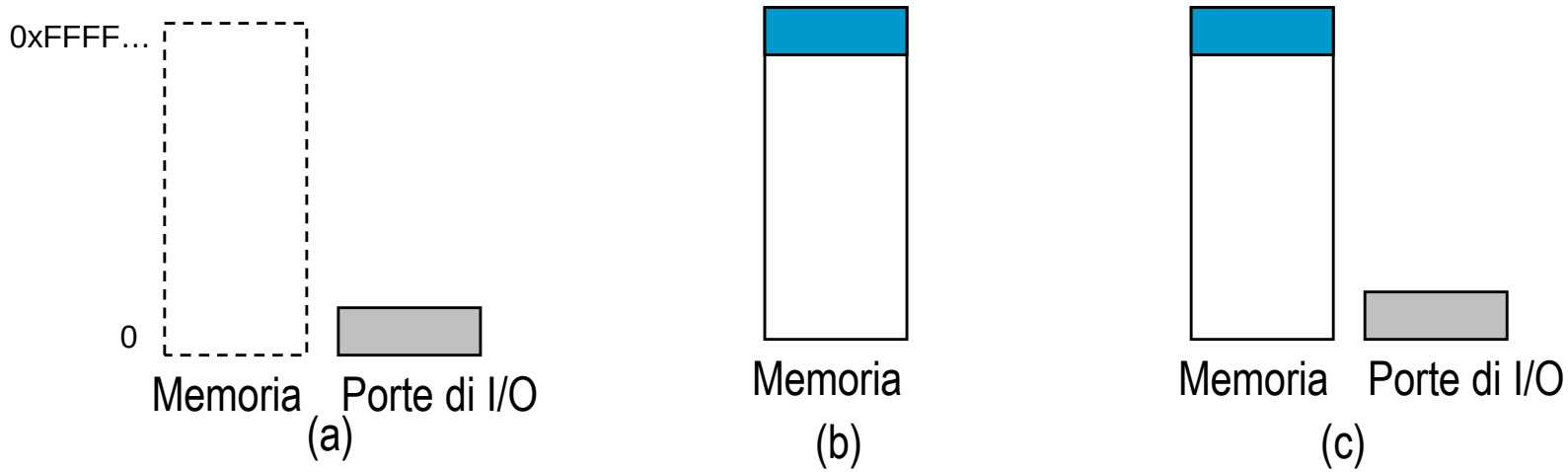
CR=CONTROL REGISTER
SR=STATUS REGISTER
DR=DATA REGISTER

Metodi di comunicazione fra processore e dispositivo

(a) Spazi separati di I/O e memoria

(b) Memory-mapped I/O

(c) Soluzione ibrida



Istruzioni per la comunicazione fra processore e disp.

- Se si usano i metodi (a) o (c) per scambiare dati fra processore e memoria si rendono necessarie
  - Istruzioni speciali di I/O (es. Architetture x86)
  - L'istruzione speciale specifica il numero del dispositivo
    - Questo puo' essere trasmesso "come un indirizzo" sul bus di I/O
  - L'istruzione speciale specifica il comando da dare al dispositivo
    - Questo puo' essere trasmesso "come un dato" sul bus di I/O

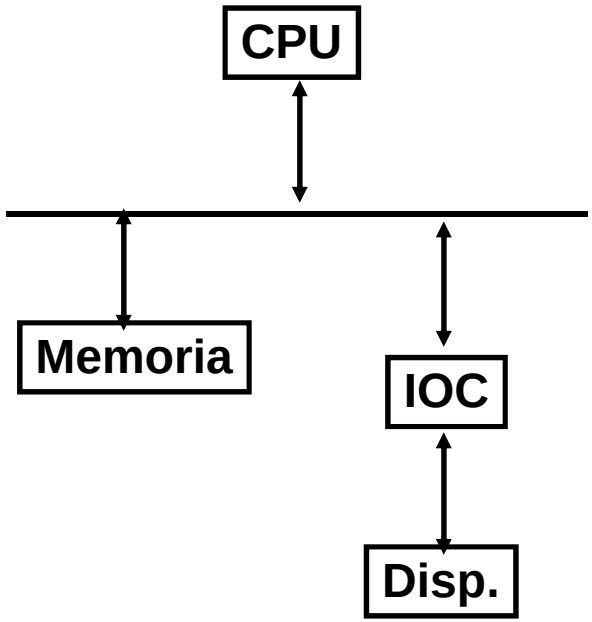
- Se si usa il metodo (b) non e' necessario introdurre nuove istruzioni per comunicare con l'I/O
  - E' sufficiente riservare certi indirizzi di memoria, non per mappare RAM ma per mappare indirizzi relativi al dispositivo
  - Letture e scritture a tali indirizzi sono interpretati dalla periferica come comandi
  - Si impedisce all'utente l'uso di questi indirizzi perche' risiederanno in una zona riservata allo spazio di indirizzamento del kernel
  - Questa tecnica e' utilizzabile anche nel caso (c)

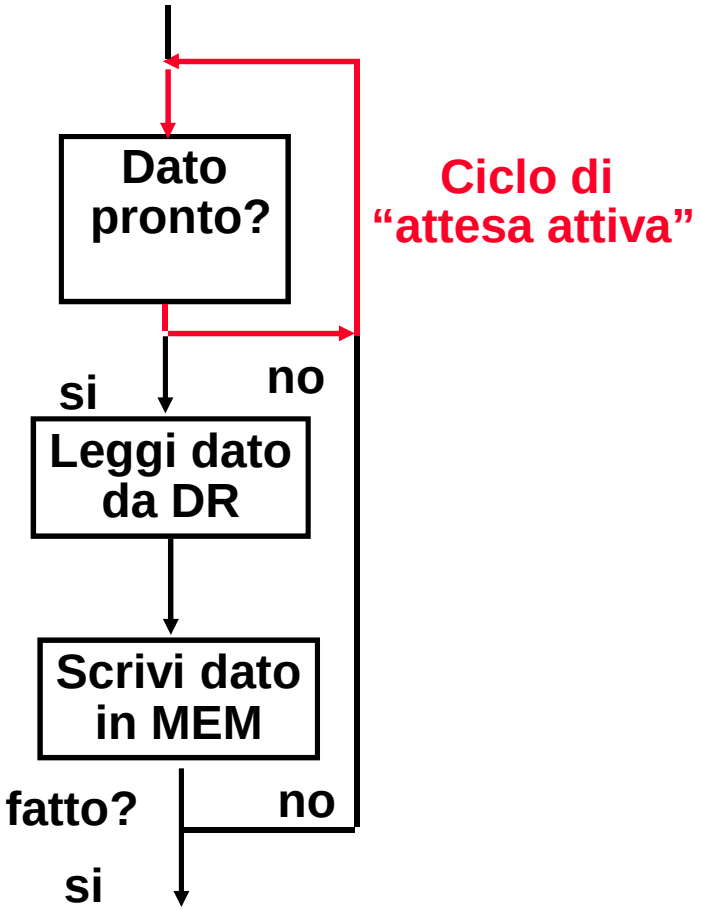
Protocolli di comunicazione fra processore e dispositivo

- Il processore ha bisogno di sapere quando
  - Il dispositivo ha completato un'operazione
  - Il dispositivo ha incontrato un errore
- Esistono tre protocolli fondamentali
  - Polling:
    - Il dispositivo mette il dato in un "data register" e segnala questo fatto tramite uno "status register"
    - Il processore controlla periodicamente lo "status register"
  - Interrupt:
    - Ogniqualvolta il dispositivo ha bisogno di attenzione dal processore, interrompe il processore tramite linea dedicata
  - DMA/IOP:
    - Il processore inizia l'operazione
    - Un processore ausiliario si occupa del trasferimento del dato
    - Al processore e' notificato che l'operazione e' finita con un interrupt dal DMA/IOP verso il processore

Polling

Protocollo di Polling (o di I/O Programmato)



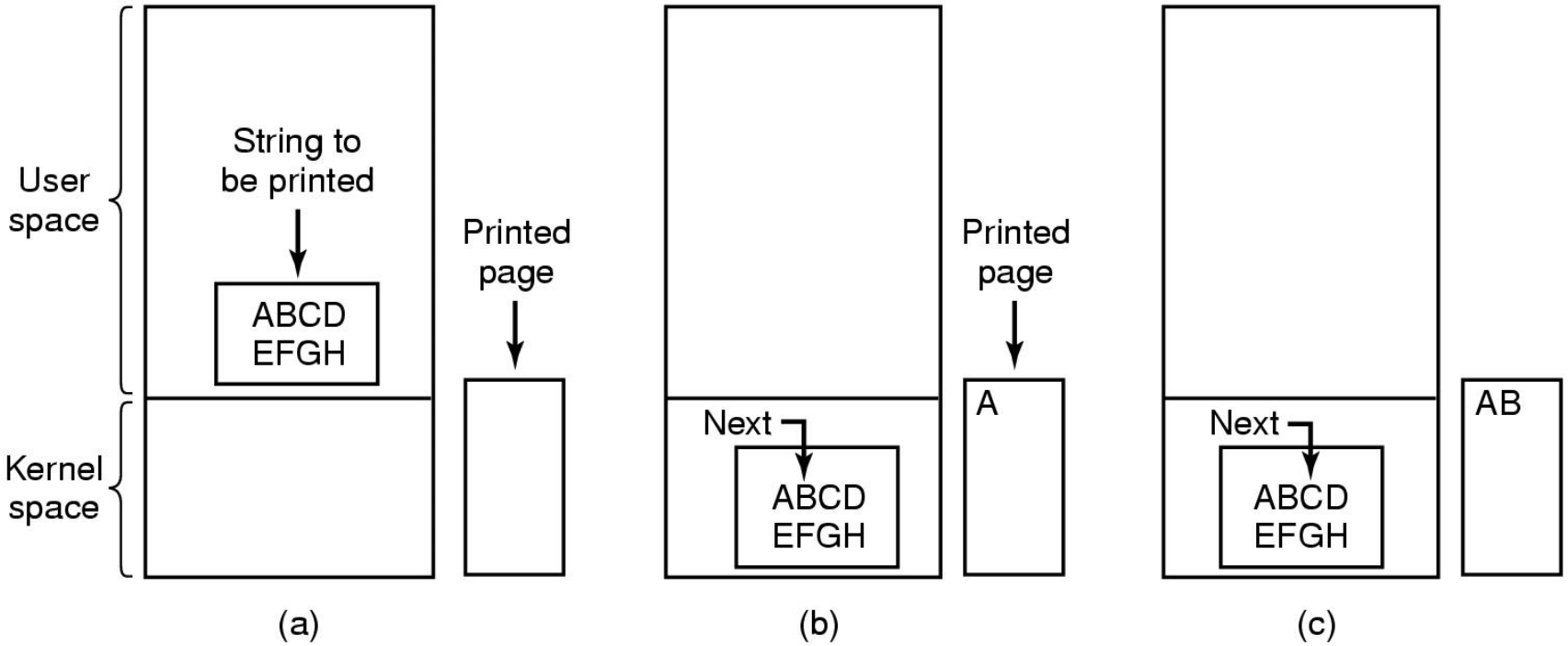


- Vantaggio
  - Semplice: il processore ha il controllo totale e fa tutto il lavoro
- Svantaggio
  - La gestione del polling puo' consumare molto tempo del processore
    - Trucco: distribuire i controlli di I/O fra codice che effettua altre operazioni

Polling - dettaglio

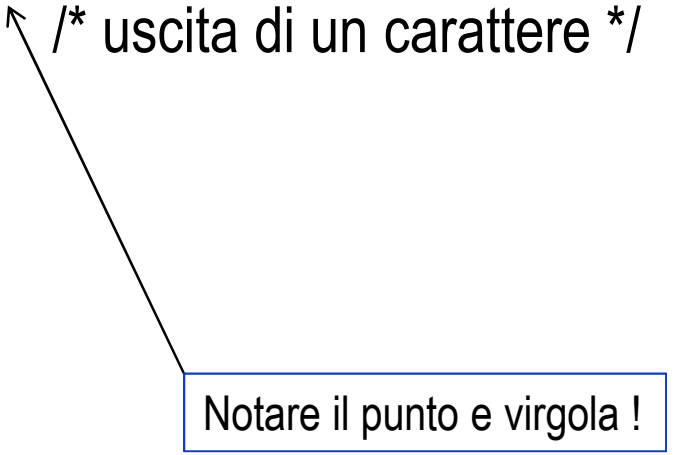
- La CPU richiede un'operazione di I/O
- Il controller di I/O effettua l'operazione di I/O
- Il controller di I/O attiva un bit nello "status register"
  - Il controller di I/O non informa direttamente la CPU
  - Il controller di I/O non interrompe la CPU
- La CPU guarda periodicamente il bit nello "status register"
  - La CPU puo' attendere oppure ritornare a vedere piu' tardi

Esempio di Polling: stampa di una stringa



Stampa di una stringa con Polling

```
copy_from_user(buffer, p, count);          /* p e' un buffer nel kernel */
for (k=0; k<count; ++k) {                  /* ripeti per ogni carattere */
    while (*printer_status_reg != READY); /* attendi il bit di READY */
    *printer_data_register = p[k];         /* uscita di un carattere */
}
return_to_user();
```



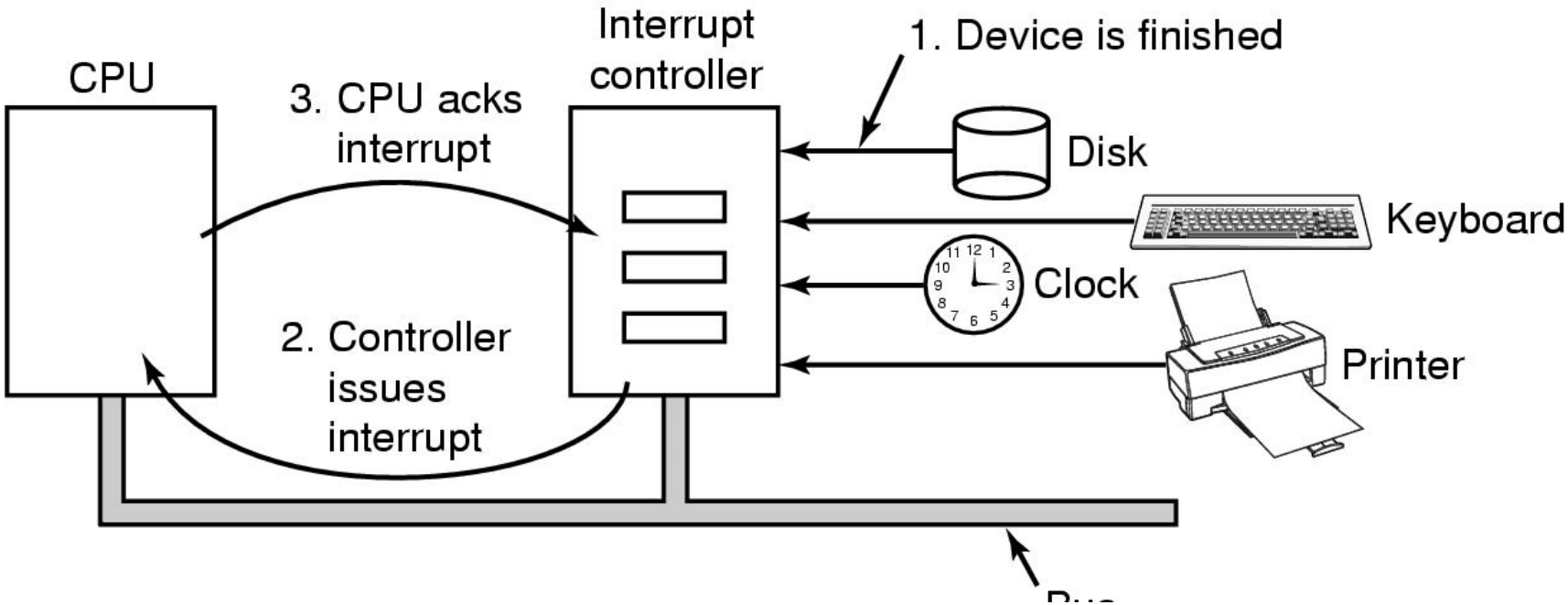
Interrupt

Protocollo ad Interrupt

- Evita l'attesa attiva del processore
- Evita di fare numerosi controlli sullo stato del dispositivo
- Il controller del dispositivo interrompe una sola volta quando ha bisogno

Protocollo ad Interrupt: funzionamento

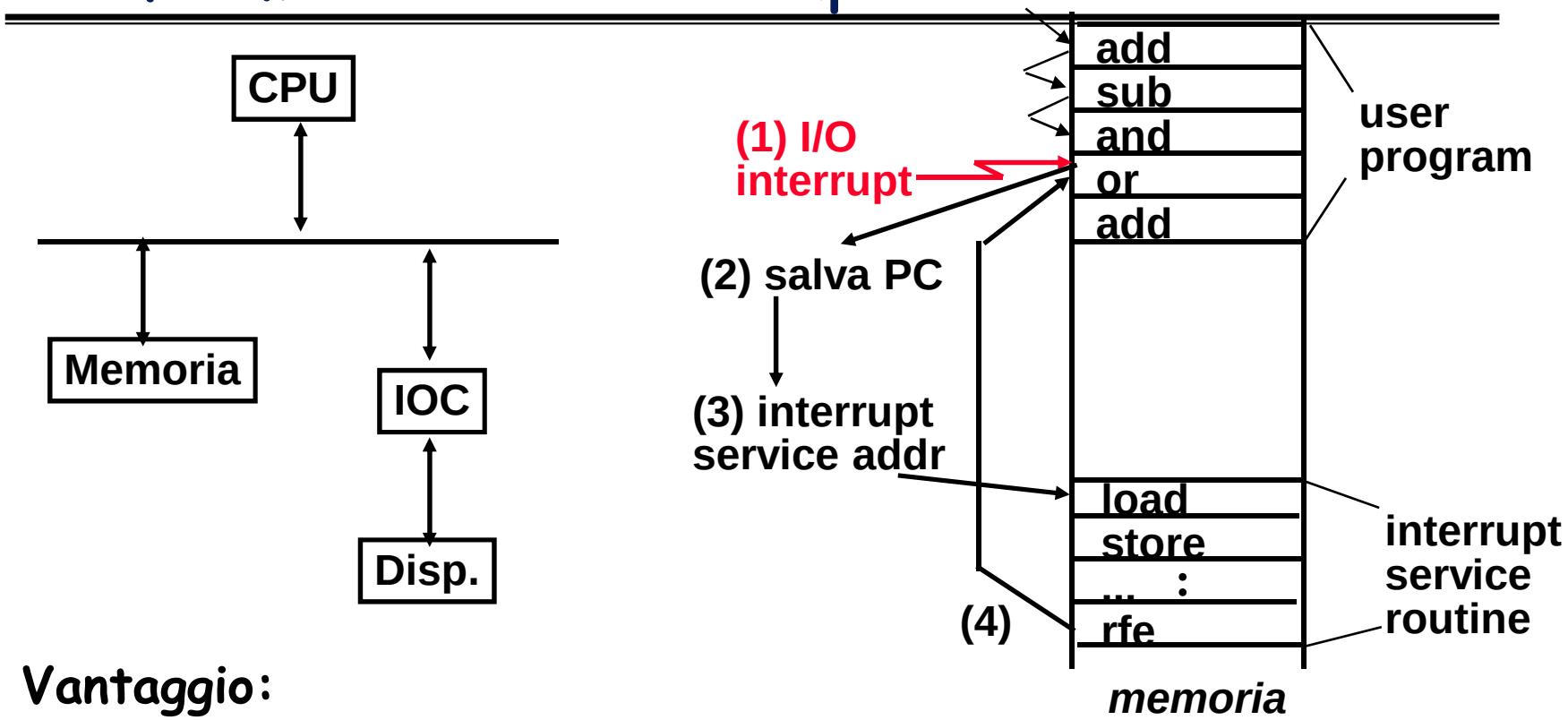
- La CPU da' un comando di lettura
- Il controller di I/O prende un dato dal dispositivo mentre la CPU si occupa di altro
- Il controller di I/O interrompe la CPU
- La CPU richiede i dati
- Il controller di I/O trasferisce i dati al processore



Dal punto di vista della CPU…

- La CPU da' un comando di lettura
- La CPU passa a fare altro lavoro
- Al termine di ogni istruzione guarda se c'e' un interrupt pendente
- Se c'e' un interrupt pendente
  - Salva il contesto
  - Serve l'interrupt: preleva il dato e lo mette in memoria

Trasferimento dati a Interrupt



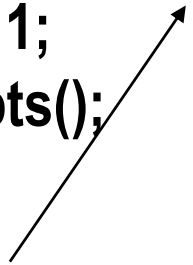
- Vantaggio:
  - Il programma utente e' bloccato solo durante il tempo effettivo per trasferire il dato dal dispositivo alla memoria
- Svantaggio, e' necessario hardware dedicato per
  - Generare l'interrupt (lato controller di I/O)
  - Accorgersi dell'interrupt (lato CPU)
  - Salvare lo stato della CPU e ripristinare lo stato dopo l'interrupt

Stampa di una stringa ad Interrupt

Codice di preparazione alla stampa (es. Implementazione di un system call):

```
copy_from_user(buffer, p, count);

while (*printer_status_reg != READY);
*printer_data_register = p[0];
count = count - 1;
enable_interrupts();
scheduler();
```



Trasferisco il primo carattere

Inizialmente avro' k=1

Routine di servizio dell'interrupt

```
if (count == 0) {
   unblock_user();
} else {
   *printer_data_register = p[k];
   count = count - 1;
   k = k + 1;
}
acknowledge_interrupt();
return_from_interrupt();
```

Interrupt

- Per la CPU, l'interrupt e' esattamente come un'eccezione salvo che
  - L'interrupt e' un evento asincrono
    - Non e' provocato da istruzioni
    - Non impedisce il completamento di alcuna istruzione
  - Ad esso fa seguito un trasferimento di informazioni

- L'implementazione dell'interrupt e' piu' complicata di quella di un'eccezione, perche'
  - Deve consentire di identificare il dispositivo che ha generato l'interrupt
  - Le richieste di interruzione possono avere priorita' diversa
  - Le richieste di interruzione provenienti da interrupt diversi possono sovrapporsi (interrupt multipli)

Identificare il dispositivo che ha generato l'interrupt

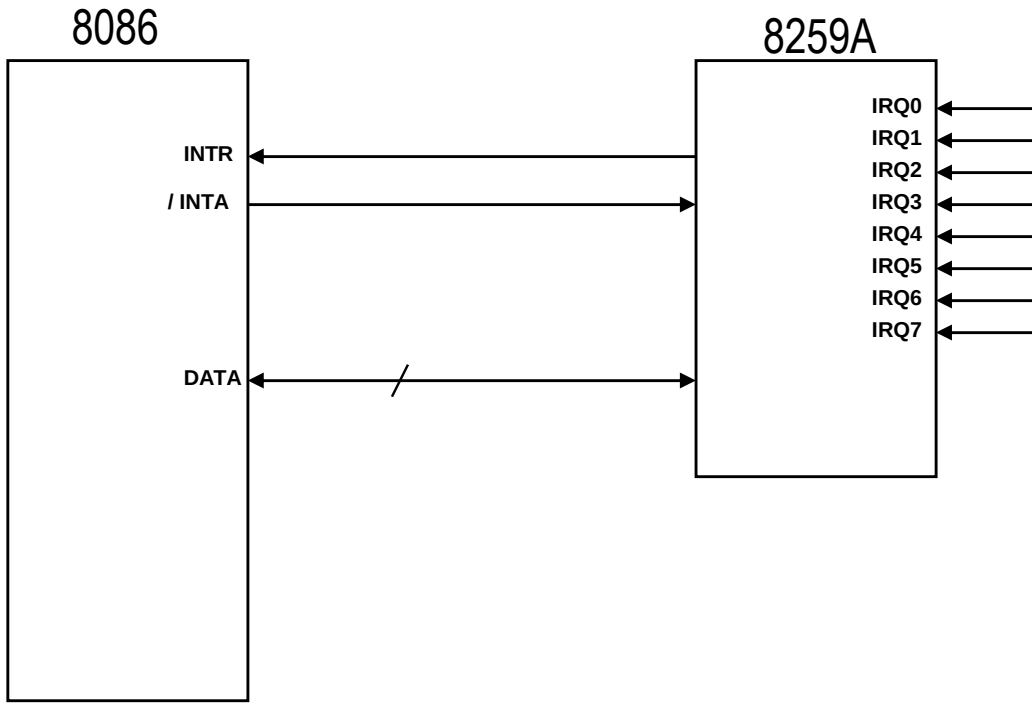
- Differenti linee per ogni controller
  - Usato nei PC
  - Svantaggio: limita il numero dei dispositivi
- Software poll
  - La CPU chiede ad ognuno dei controller se ha generato un interrupt
  - Lento
- Daisy Chain (Hardware poll)
  - Il segnale di Interrupt Acknowledge viene inviato in daisy-chain
  - Il controller e' responsabile di mettere sul bus un "vettore"
  - La CPU usa il vettore per identificare la routine di servizio
- Bus Mastering
  - Il controller deve richiedere il bus prima di fare interrupt
  - e.g. PCI, SCSI

Interrupt Multipli

- Ogni interrupt ha una priorita' associata
- Dispositivi a piu' alta priorita' possono interrompere anche quando si servono dispositivi a piu' bassa priorita'
- Nel caso di bus mastering l'ulteriore interrupt puo' venire solo dall'attuale master

Esempio - PC

- 80x86 ha una solo linea di interrupt
- I sistemi basati su 8086 usano un "controller di interrupt" chiamato 8259A
- Il chip 8259A ha 8 linee di interrupt

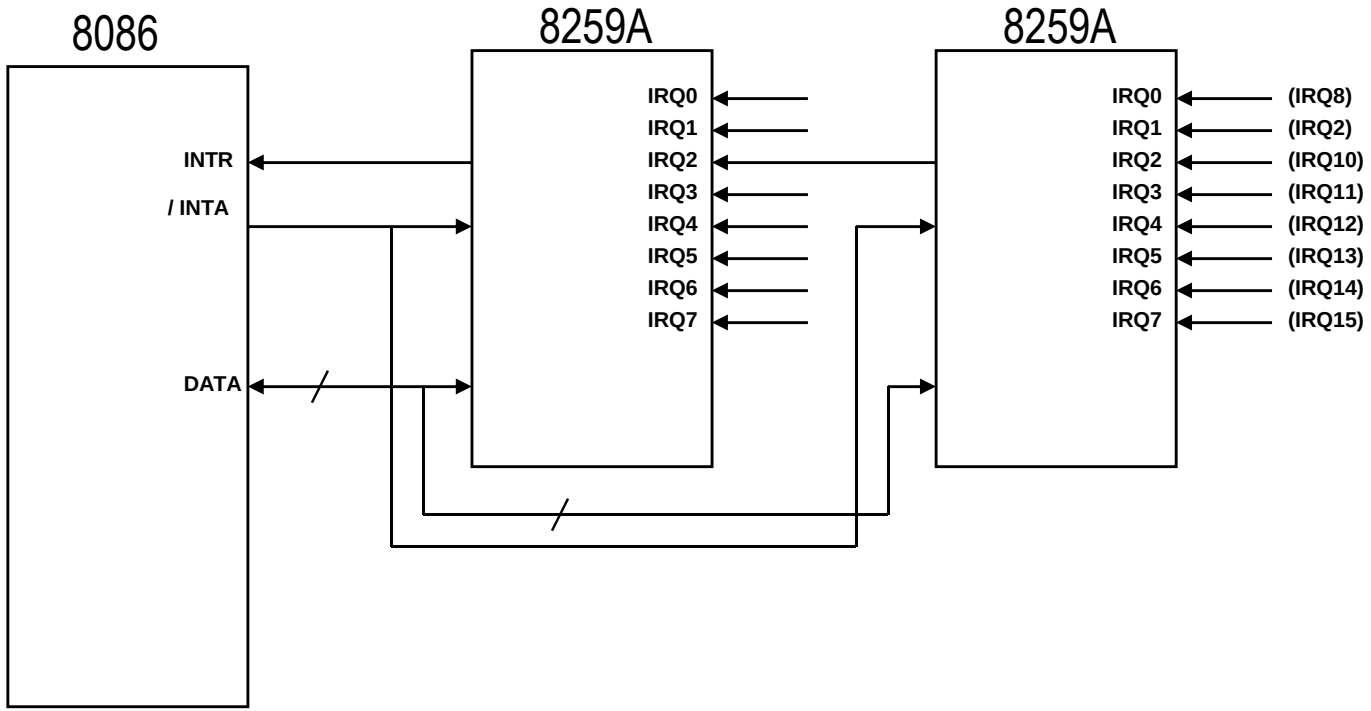


Sequenza di Eventi

- L' 8259A accetta interrupt
- L' 8259A determina la priorita'
- L' 8259A avvisa l' 8086 (attiva la linea INTR)
- La CPU invia una conferma (linea INTA)
- L' 8259A mette sul bus dati della CPU il "vettore" cioe' un numero che identifica la sorgente di interupt
- La CPU serve l'interrupt mandando in esecuzione la relativa routine di servizio

Cascading

- Nei PC si usarono due 8259A in cascata
- Fornisce 15 linee di interrupt
- Per mantenere la compatibilita' il vecchio interrupt 2 viene rimappato sull'interrupt 9
- Dai 486 in poi sono entrambi incorporati nella CPU



DMA
Direct Memory Access

Direct Memory Access

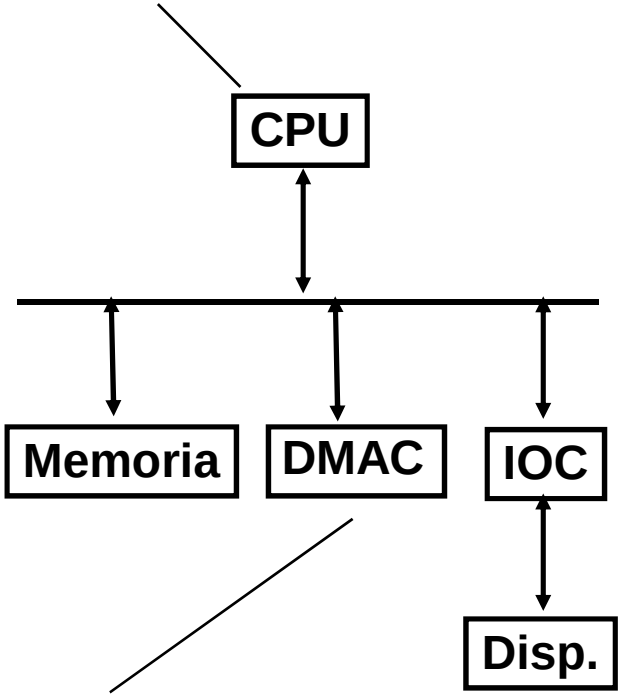
- Interrupt frequenti e polling richiedono l'assistenza del processore
  - La velocita' di trasferimento e' limitata
  - La CPU e' impegnata in troppe attivita'

…il DMA e' la risposta

- Esterno alla CPU
- Agisce come master sul bus
- Trasferisce blocchi di dati da/verso la memoria senza l'intervento della CPU

Funzionamento del DMA

La CPU invia un indirizzo iniziale, direzione e lunghezza del trasferimento al DMA Controller. Poi da' lo "start"…

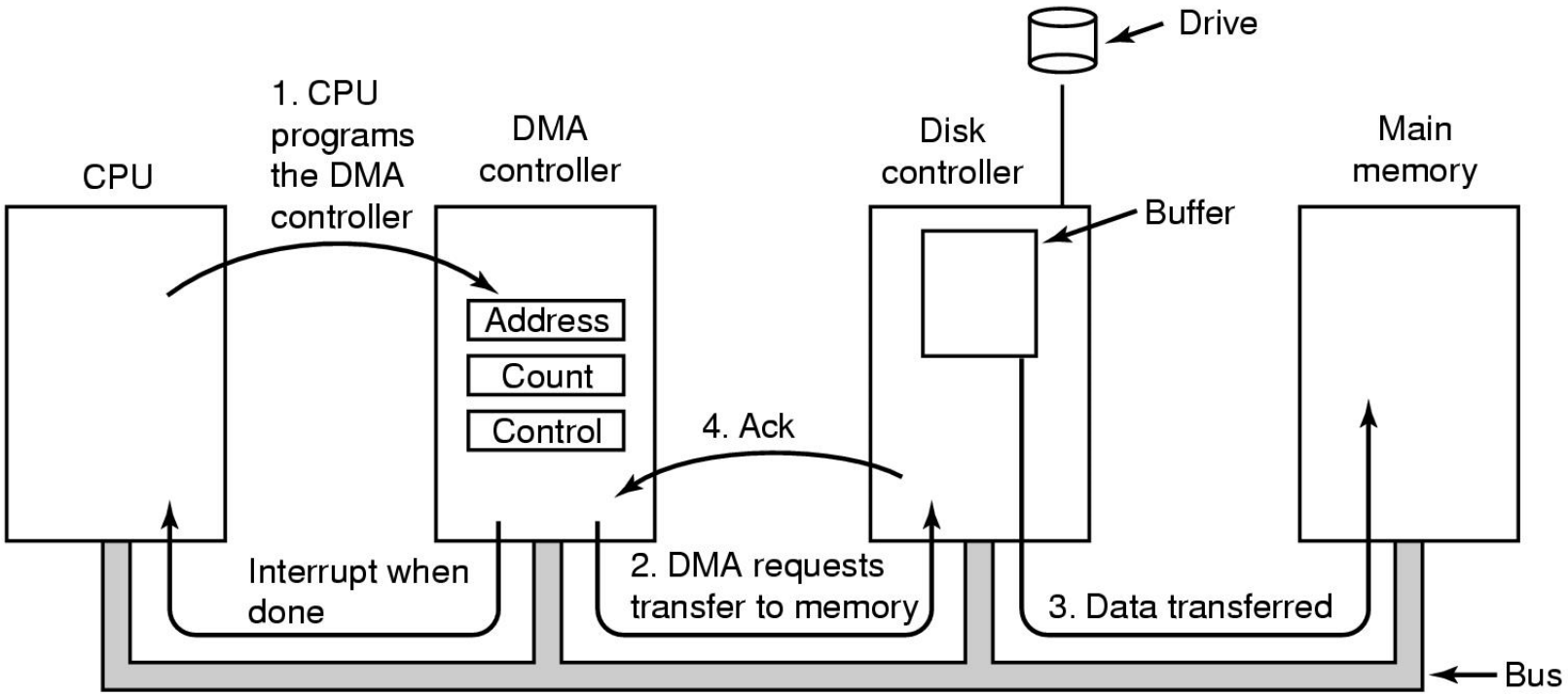


Il DMA Controller genera tutti i necessari segnali per indirizzare la periferica e la memoria e per gestire l'handshake

DMA - dettaglio

- La CPU dice al DMA Controller:
  - Indirizzo iniziale della zona di memoria coinvolta
  - Indirizzo del controller del dispositivo
  - Direzione (Read/Write)
  - Quantita' di dati da trasferire
- LA CPU puo' svolgere altro lavoro
- Il DMA Controller si occupa del trasferimento
- Il DMA Controller manda un interrupt quando ha finito

DMA -dettaglio (2)



Stampa di una stringa con DMA

Codice di preparazione alla stampa (es. Implementazione di un system call):

```
copy_from_user(buffer, p, count);
setup_DMA_controller();
scheduler();
```

Routine di servizio dell'interrupt

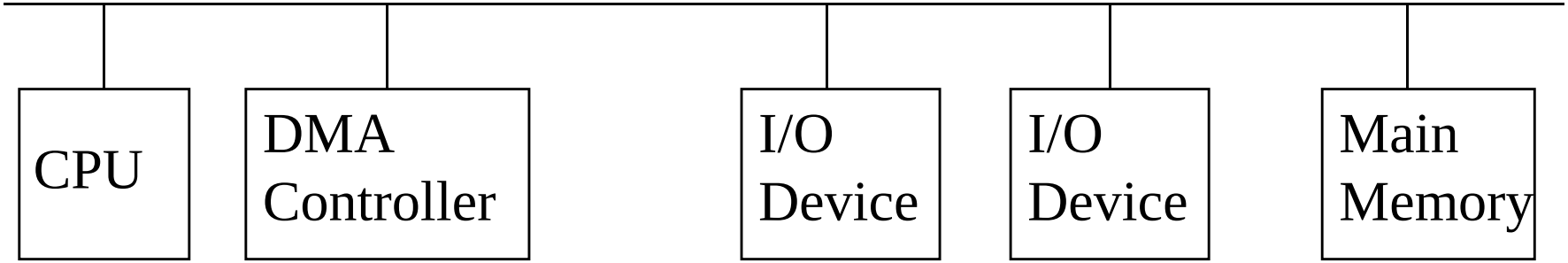
```
acknowledge_interrupt();
unblock_user();
return_from_interrupt();
```

DMA: Cycle Stealing

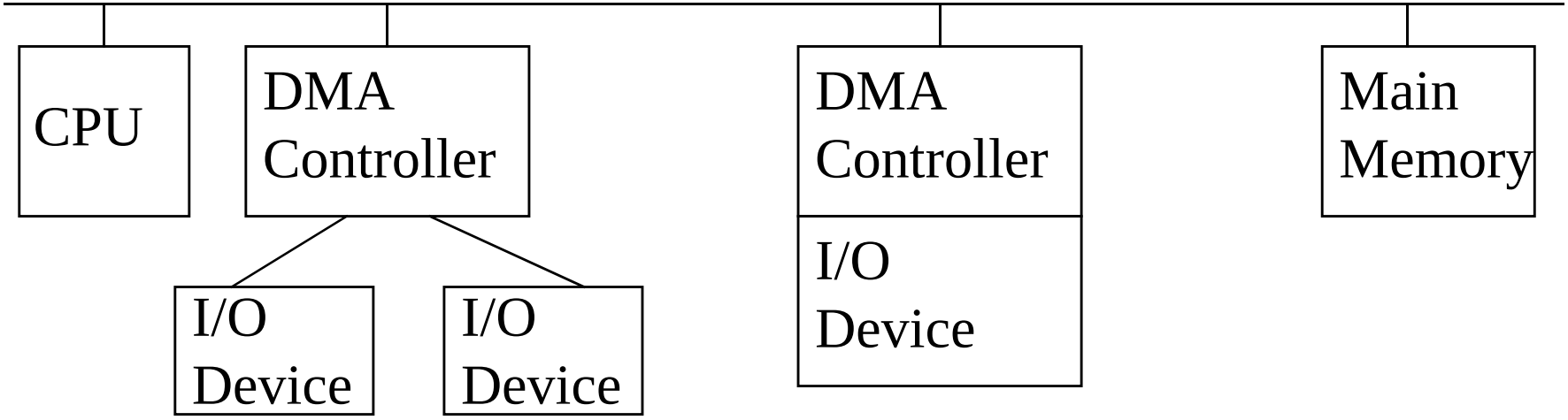
- Il DMA controller occupa il bus per un ciclo (di bus)
- Trasferisce una word di dati
- Non e' un interrupt
  - La CPU non deve fare salvataggio/ripristino stato !
- La CPU eventualmente attende di piu' quando dovra' fare accesso al bus
  - Es. Per fare fetch/read/write
  - La CPU attende un poco ma sicuramente meno di quanto dovrebbe attendere per trasferire il dato da sola

DMA: Configurazioni (1)

- Singolo Bus, DMA controller separato
- Ogni trasferimento usa il bus due volte
  - Da dispositivo a DMA e da DMA a memoria
- LA CPU trova il bus occupato per piu' tempo

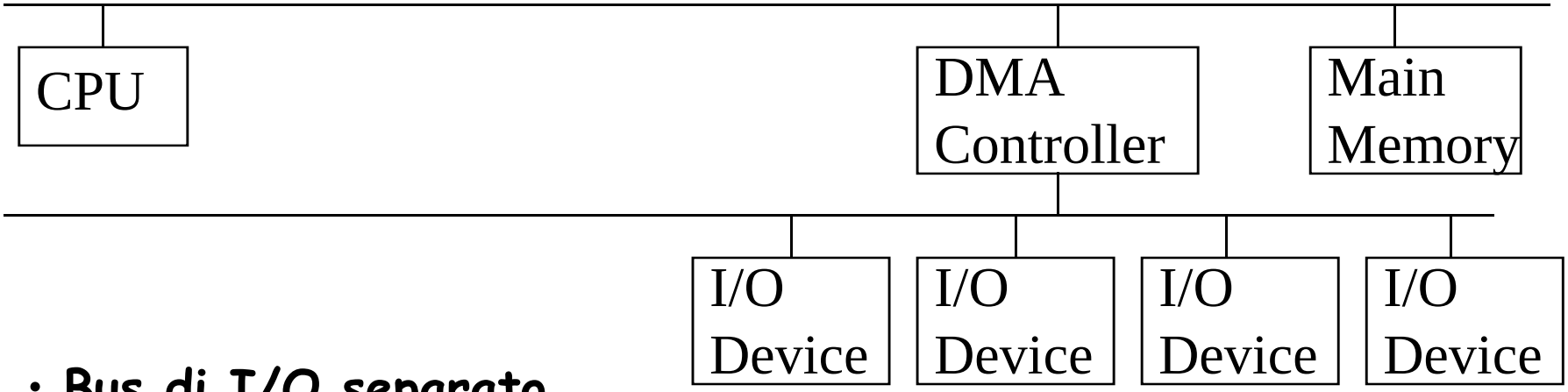


DMA: Configurazioni (2)



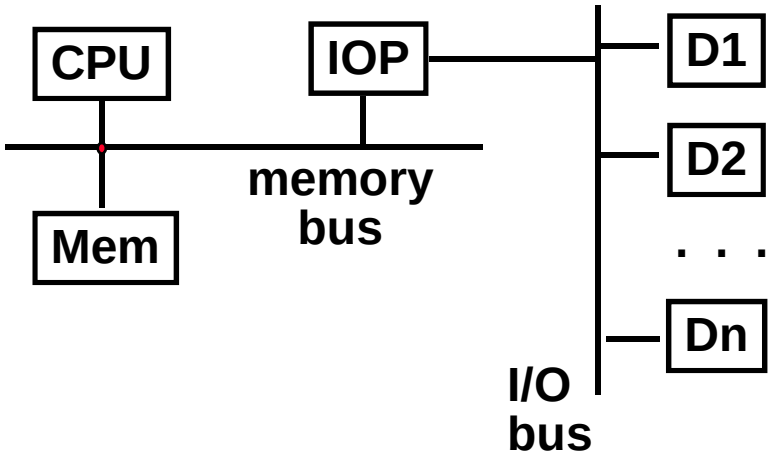
- Singolo Bus, DMA controller integrato
- Il controller puo' gestire piu' di un dispositivo
- Ogni trasferimento usa il bus una sola volta
  - dsa DMA a memory
- La CPU dimezza le volte in cui trova occupato il bus

DMA: Configurazioni (3)



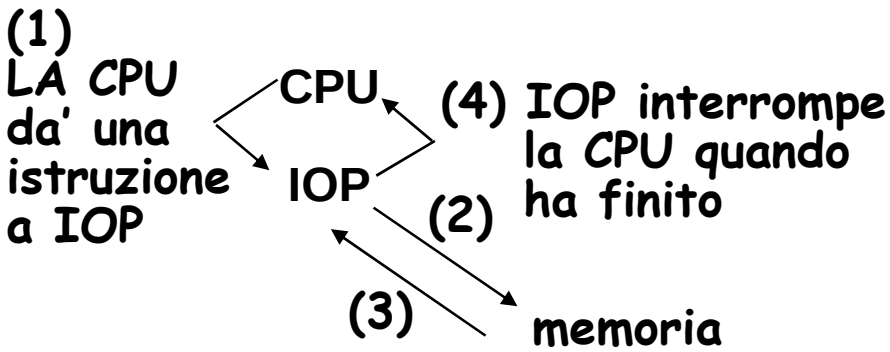
- Bus di I/O separato
- Il bus supporta tutti i dispositivi abilitati al DMA
- Ogni trasferimento usa il bus una sola volta
  - DMA to memory
- La CPU dimezza le volte in cui trova occupato il bus rispetto al primo caso

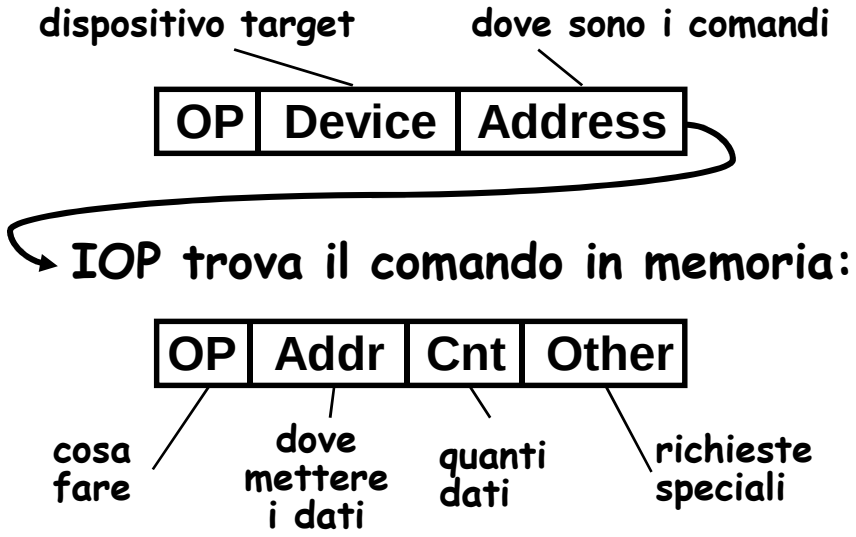
Delegare le operazioni di I/O: IOP (I/O Processor)



I trasferimenti da/verso la memoria sono controllati direttamente dall'IOP

IOP effettua cycle stealing





CONTROLLER DMA

DMA Controller (DMAC)

- Periferica dedicata che può controllare il bus:
  - Inizializza e controlla gli accesi al bus
    - tra un dispositivo di I/O e la memoria
    - tra due aree di memoria
  - Agisce come una implementazione hardware della routine di gestione dell'interrupt di buffer-pieno o buffer-vuoto
- Tre tipi di controllers DMA:
  - 1D, un singolo registro per gli indirizzi
  - 2D, due registri per l'indirizzamento
  - 3D, tre o più registri per l'indirizzamento

Struttura di un Controller DMA Generico (1)

- Costituito da:
  - Address Generator

    Genera gli indirizzi di memoria o di periferica coinvolti nei trasferimenti. Costituito da un registro base e un contatore auto-incrementante

  - Address Bus

    Dove finiscono gli indirizzi generati per accedere ad una specifica locazione di memoria o della periferica

  - Data Bus

    Utilizzato per trasferire dati dal DMA alla destinazione. Molte volte il trasferimento avviene direttamente dalla sorgente alla destinazione, con il controller DMA che solamente seleziona i due dispositivi

Controller DMA Generico (2)

- **Bus Requester**

  Utilizzato per richiedere il bus alla CPU (necessario in una corretta configurazione multimaster)

- **Local Peripheral Control**

  Permette al controller DMA di selezionare la periferica e gestirla (necessario per indirizzamento singolo o implicito)

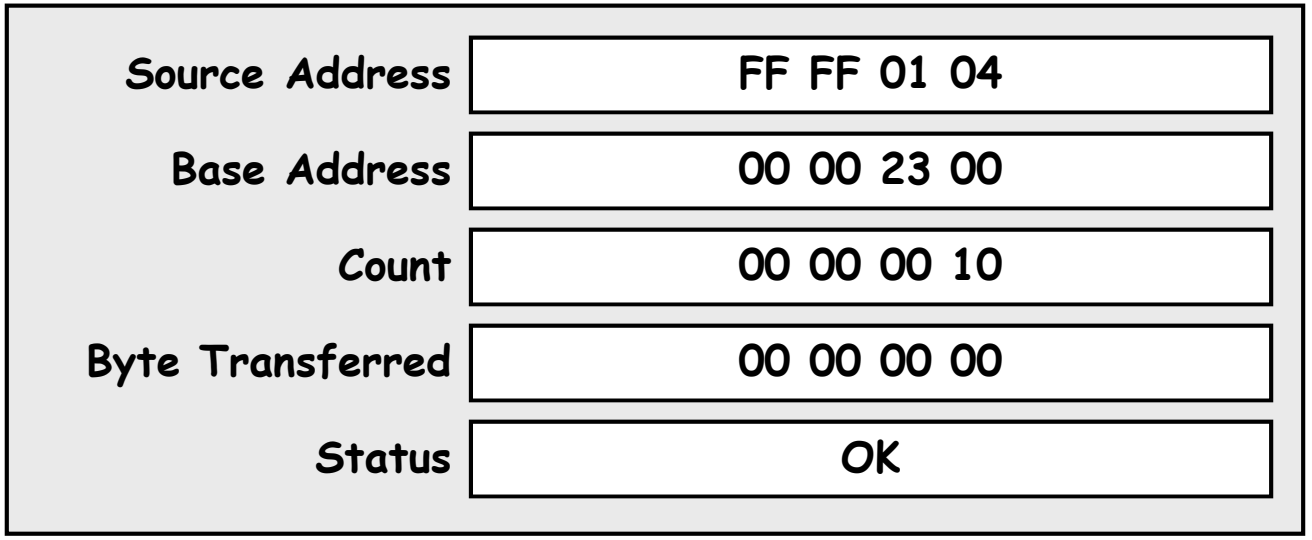
- **Interrupt signals**

  I controllers DMA possono interrompere la CPU appena il trasferimento è terminato o se si è verificato un errore

Controller DMA Generico: funzionamento (1)

1) Programmazione del controller:

- Definizione indirizzo base e dimensione del trasferimento
- Definizione comunicazione con il processore
  - Es. definizione delle condizioni su cui si generano interrupt
- Definizione comunicazione con le periferiche
  - Es. Mappare le periferiche sulle linee di richiesta di DMA e politica di arbitraggio per richieste simultaneee al DMA
- Definizione del tipo di trasferimento dei dati
  - all-at-once , individuale od altro



Es. Registri di controllo del DMA

Controller DMA Generico: operazioni (2)

2) Inizio del trasferimento

- Assume che il controller sia stato configurato correttamente
- Il traferimento ha inizio su una esplicita richiesta
  o verso il DMAC direttamente
  o attraverso il processore (attivato su interrupt)

3) Richiesta del bus

- Il DMAC invia una richiesta alla CPU che rilascia il bus (supportato nei processori piu' recenti)
- In sistemi senza gestione delle richieste su bus, il DMAC deve funzionare in "bus-stealing": il processore resta bloccato su un accesso al bus mentre il DMAC opera

4) Generazione dell'indirizzo

- Una volta che il controller ha il bus, deve attivare la locazione di memoria. Sono utilizzate diverse interfacce:
  - Non multiplexata
  - Multiplexata
- inoltre il DMAC fornisce altri segnali di controllo, come R/W, strobe per lavorare sul bus
- I piu' recenti DMAC sono abbastanza generici per poter funzionare con diverse famiglie di processori

Controller DMA Generico: operazioni (3)

5) Trasferimento dei dati

- Il trasferimento può avvenire o attraverso un buffer interno al controller o direttamente dalla periferica

6) Aggiornamento dell'indirizzo

- Una volta terminato il trasferimento viene calcolato il nuovo indirizzo per il trasferimento successivo e viene aggiornato il contatore dei trasferimenti

7) Aggiornamento del processore

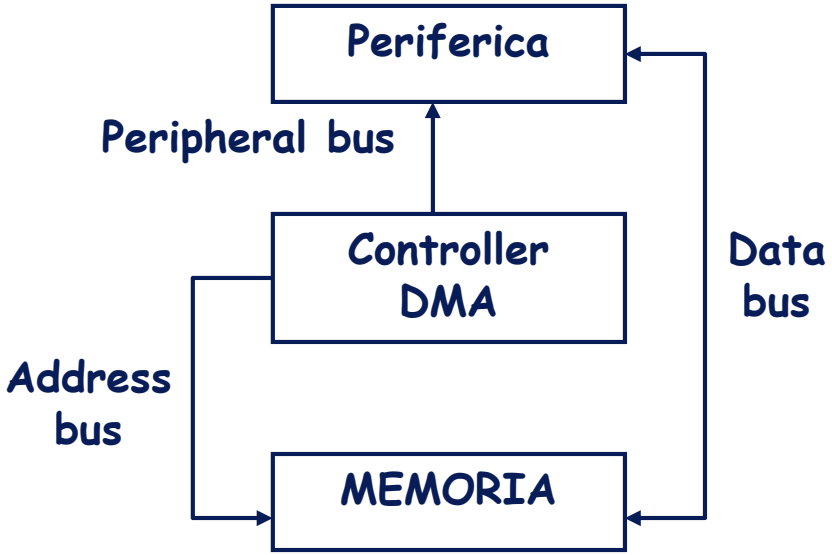
- Il controller notifica al processore l'avvenuto trasferimento di un blocco o un eventuale errore o altri eventi, inviando ad esso un interrupt

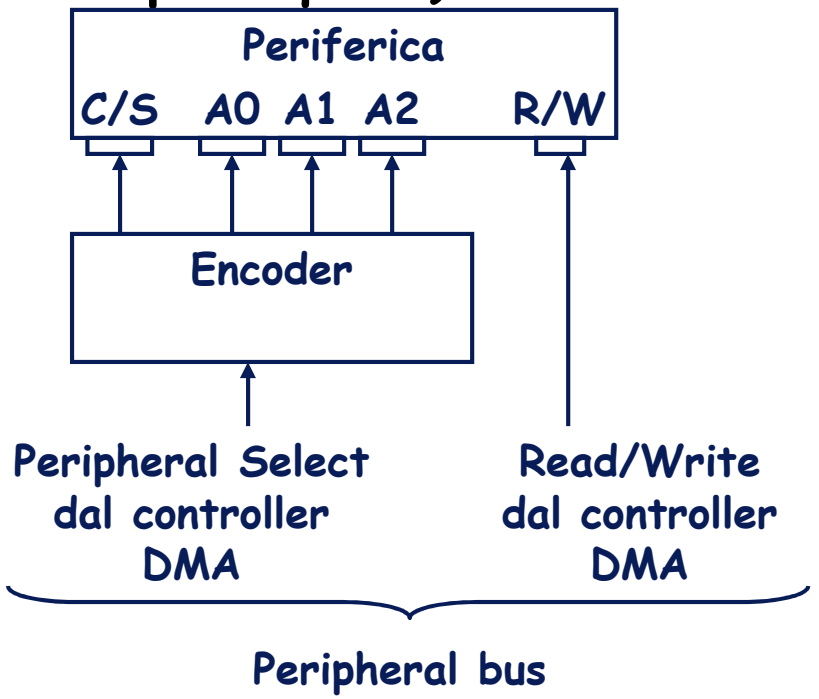
Modelli di controllers DMA

- Vari modi di funzionamento in base:
  - Trasferimento dei dati
    - Singolo
    - Doppio
  - Indirizzamento
    - singolo
    - complesso

Trasferimento dati singolo (indirizzamento implicito)

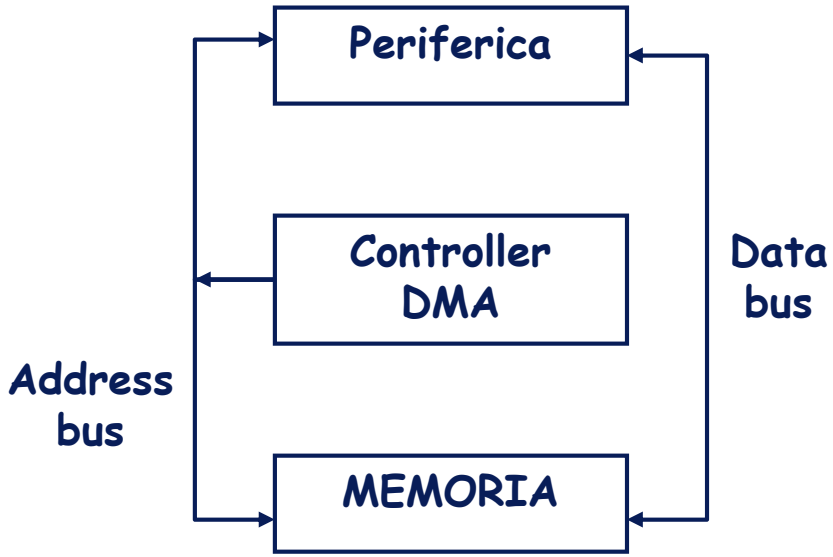
- Il DMA utilizza l'address bus per indirizzare la locazione coinvolta nel ciclo di bus verso la memoria
- Utilizza il bus di periferica (chip select) per selezionare la periferica e rendere disponibile il suo bus dati
  - In conseguenza dell'attivazione del Chip Select puo' essere necessario un encoder per selezionare un dato indirizzo (implicito) della periferica
- Se e' la periferica ad iniziare, abbassa la linea di request (comune l'uso della linea di interrupt request)





Trasferimento dati doppio

- Si usano due indirizzi e due accessi per trasferire dati dalla periferica (o memoria) a un'altra locazione di memoria
  - Consuma 2 CICLI di bus
  - Utilizza un buffer temporaneo interno al controller



1D Model

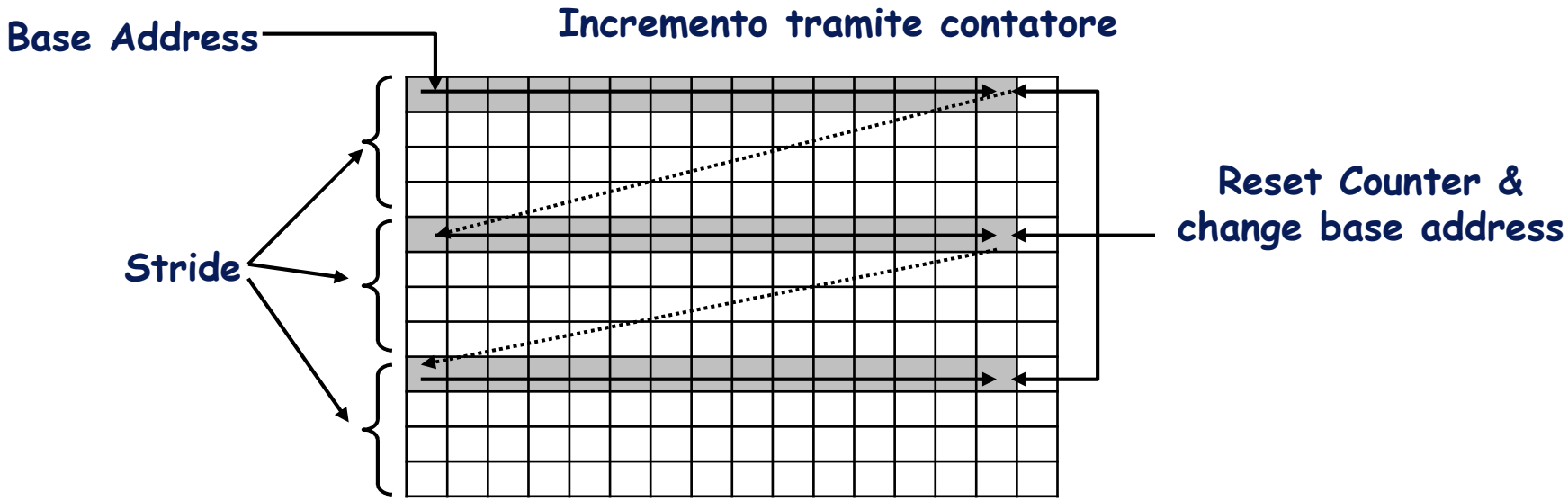
- Usa un indirizzo base e un contatore per definire la sequenza di indirizzi da usare nei cicli DMA
- Svantaggi:
  - Una volta che il blocco di dati è stato trasmesso, l'indirizzo ed il contatore sono resettati potendo sovrascrivere accidentalmente dati precedenti
  - Diventa necessario un interrupt dal DMA al processore ogni volta che il contatore termina il conteggio per cambiare l'indirizzo base
- Utilita'
  - Si puo' implementare un buffer circolare con wrap around automatico

2D Model

- Es. Implementazione di protocolli di comunicazione basati sullo scambio di pacchetti
  - Devo spezzare grosse quantita' di dati in pacchetti e aggiungere gli header
- Invece di riportare l'indirizzo di base al valore originale, viene sommato un valore di "stride" alla fine di un trasferimento di un blocco
  - Permette al DMA di utilizzare blocchi di memoria non contigui
- Il registro contatore e' diviso in due:
  - Un registro per l'offset all'interno del blocco
  - Un registro per contare il numero totale di blocchi o i byte totali trasferiti

2D Model

- Posso facilmente spezzare un blocco in pacchetti pronti per l'inserimento degli header
- Successivamente si puo' usare la modalita' 1D per inviare i dati alla scheda di rete

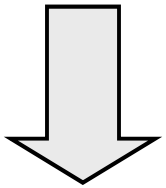


3D model

- Utilizza un meccanismo a "stride" come nel modello 2D
- Inoltre posso cambiare automaticamente lo stride per creare blocchi a indirizzi variabili
- E' possibile simulare tale modalità con un DMA 2D e una procedura software che riprogramma ogni volta il DMAC per cambiare lo stride

Problema non risolto

- Ogni DMA deve essere pre-programmato con un blocco di parametri per poter funzionare correttamente
- L'interfaccia hardware è comune per quasi tutti i differenti set di parametri
- Ogni periferica che necessita di un trasferimento deve chiedere alla CPU di programmare correttamente il DMA (interrupt)
- Spesso una certa periferica programma il DMAC con lo stesso set di parametri

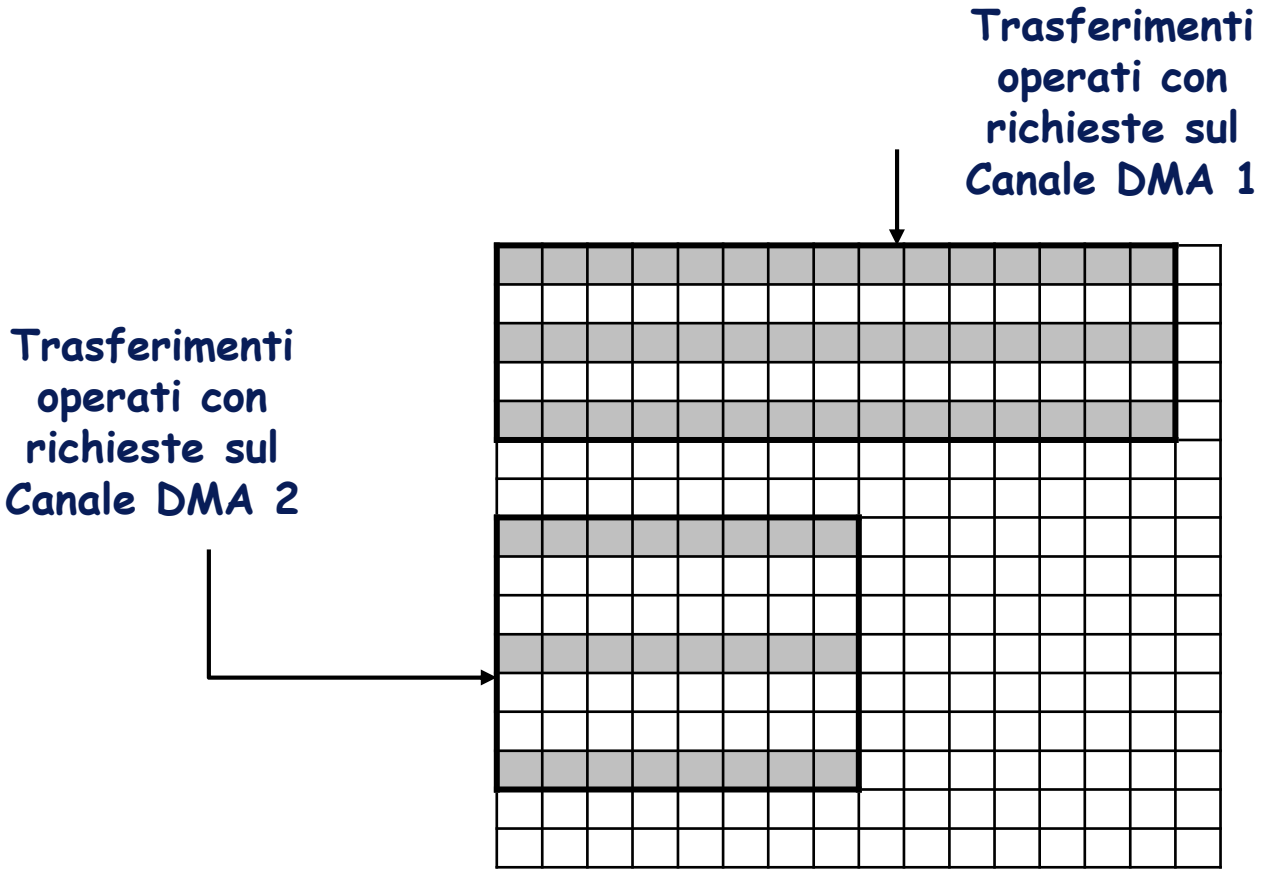


- Ancora un notevole carico di lavoro sulla CPU dato dalla gestione degli interrupt delle periferiche e dalla programmazione del DMA

Canali DMA (Blocchi di controllo)

- Ideati per ridurre l'overhead del processore
- Ad ogni periferica viene assegnata una linea esterna di request (canale DMA)
- Per ogni linea sono programmati una sola volta i parametri per quel tipo di trasferimento (blocchi di controllo)
- Quando una periferica richiede un trasferimento, asserisce il suo canale DMA e inizia il trasferimento in accordo con i parametri assegnati a quel canale
- Possibilità di condividere un singolo controller con più periferiche semza eccessivo overhead per il processore
- Nei PC tipicamente ci sono 4 canali DMA

Uso dei Canali DMA

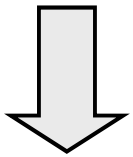


Concatenazione

- Estensione dell'idea dei canali DMA: chaining (concatenazione)
  - Canali collegati a catena in modo da generare pattern piu' complessi
  - Una volta terminato il lavoro di un canale, il controllo passa a quello successivo concatenato
  - Possibilità di schemi complessi di indirizzamento

Concorrenza di accesso ai canali DMA

- Cosa accade se arrivano richieste multiple di trasferimento al controller DMA?



- Necessità di un arbitraggio
  - Canali con priorità maggiore
  - Servizio dei canali con politica Round-Robin

Condivisione della banda del bus

- Il controller DMA compete con il processore per l'utilizzo del bus

- CPU senza cache (es. 80286, MC68000) utilizzano dall' 80% al 95% della larghezza di banda del bus
  - Ritardi nell'accesso al bus degradano enormemente le prestazioni e aumentano il tempo di latenza degli interrupt

- Per dispositivi con cache, l'interferenza del DMA sulle prestazioni è molto minore

Condivisione della banda del bus - 2

- Per consentire il giusto compromesso al progettista, molti DMA utilizzano differenti tipi di accesso al bus:
  - Trasferimento singolo
    - il bus ritorna alla CPU alla fine di ogni singolo trasferimento
    - Perdo tempo per le negoziazioni del bus e per iniziare e chiudere le transazioni di DMA
  - Trasferimento a blocchi
    - il bus è restituito alla CPU dopo il trasferimento di un blocco
    - Puo' essere molto lungo, ma poi termina
    - Il DMAC usa efficacemente il bus
  - Trasferimento su domanda
    - il bus viene trattenuto dal DMA per tutto il tempo che la periferica richiede
    - In teoria anche per un tempo lunghissimo!
    - Il DMAC puo' usare il bus in maniera non ottimale (lasciando "buchi"), ma "se e' onesto" funziona neglio del trasferimento a blocchi

INTEL 8237 (1)

- Il più comune controller DMA, utilizzato su tutti i PC IBM compatibili
- *Oggi* integrato sul chipset della scheda madre
- Supporta 4 modalità di trasferimento:
  - Trasferimento singolo
  - Trasferimento a blocchi
  - Modalità a domanda
  - Modalità a cascata – speciale modalità che gestisce più controllers 8237 a cascata per consentire più canali DMA

INTEL 8237 (2)

- Trasferimento dati
  - Da periferica a memoria
  - Da memoria a memoria (unendo due canali, ma non e' usato nei PC)

- Verify Transfer Mode :
  - Speciale modalità di trasferimento dati
  - Usato nei PC per generare indirizzi dummy per il refresh della memoria DRAM
  - Guidata da un interrupt (ogni 15µs) derivato da un canale del timer 8253

- Gestione interna della concorrenza:
  - Schema a priorità fissa
  - Schema a priorità variabile

Motorola MC68300

- Processori MC68000/MC68020 con controller DMA integrato sul chip
- Architettura:
  - Due canali DMA completamente programmabili
  - Velocità di trasferimento dati
    - 12.5 Mbytes/s in dual address mode a 25 MHz
    - 50.0 Mbytes/s in single address mode a 25 MHz
- Data la sua integrazione sulla CPU, può gestire trasferimenti tra memoria (o periferiche) interna ed esterna
  - Per i trasferimenti interni è possibile specificare la quantità di banda da occupare (25, 50, 75 o 100%)
  - Per i cicli esterni due modalità di trasferimento
    - burst
    - Single
- I registri sorgenti e di destinazione possono essere programmati indipendentemente per rimanere costanti o incrementare

Uso di una CPU separata con firmware

- Utilizzata quando non è disponibile un controller DMA
- LA "CPU separata" richiede una propria memoria e un programma che la faccia funzionare senza creare occupazione del bus della memoria (per la programmazione)
- La CPU DMA istruisce su come effetture trasferimenti DMA
- Vantaggio di questa tecnica:
  - possibilità di essere programmata con software di alto livello per processare e trasferire i dati
- Molti dei processori usati nei sistemi embedded ricadono in questa categoria