Lezione 16
Tecniche di pilotaggio dei dispositivi di I/O

- Le periferiche si differenziano notevolmente fra loro perché
  - Trasferiscono differenti quantità di dati
  - Funzionano a velocità diverse
  - Utilizzano 'formati di dato' differenti

- Tipicamente funzionano a velocità più basse rispetto a CPU e Memoria

- Per interfacciarsi con il resto del sistema una periferica ha bisogno di un "controller" o "interfaccia"

Velocità 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

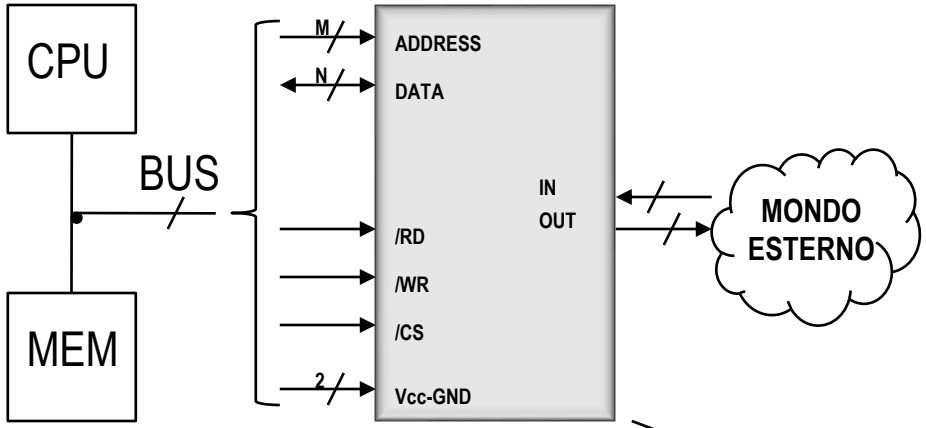
- **USB**
  - v1 (low speed) 1.5Mb/s=187 KB/s
  - v1.1 (full speed) 12Mb/s=1.5MB/s
  - v2.0 (hi-speed,2001) 480Mb/s=60MB/s
  - v3.0 (superspeed,2009) 5Gb/s=625MB/s
  - v3.2gen1 (2013) 10Gb/s=1.25GB/s
  - v3.2gen3 (2019) 40Gb/s=5GB/s
  - v4 gen4 (2023) 80Gb/s=10GB/s
- **SATA**
  - 1.5Gb/s (2001) → 130MB/s(HDD),200MB/s(SSD)
  - 3.0Gb/s (2001) → 200MB/s (reale)
  - 6.0Gb/s (2008) → 400MB/s (reale)
  - classica (1980) 10 Mb/s = 1.25MB/s
- **ETHERNET**
  - 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
  - 400xGiga (2017) 400Gb/s = 50GB/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)
  - v4.0 (2017) 2GB/s(1lane),32GB/s(16-lane)
  - V5.0 (2019) 4GB/s(1lane),63GB/s(16-lane)
  - V6.0 (2022) 8GB/s(1lane),121GB/s(16-lane)
  - V7.0 (2025 – atteso) 16GB/s(1lane),242GB/s(16-lane)

Architettura del Sistema di I/O

- L'architettura del sistema di I/O è…
  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 più 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

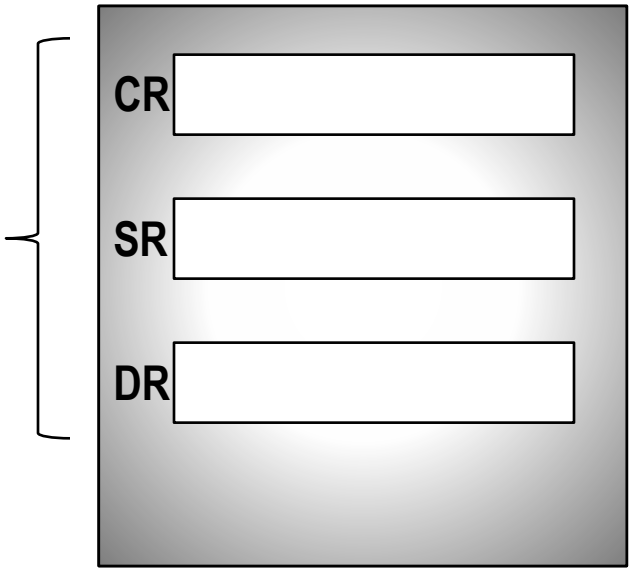
Nota: nel seguito si parlerà di "dispositivo" sottintendendo che ci si riferisce sempre al "controller del dispositivo"

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"



Porte di I/O tipicamente presenti in un dispositivo:

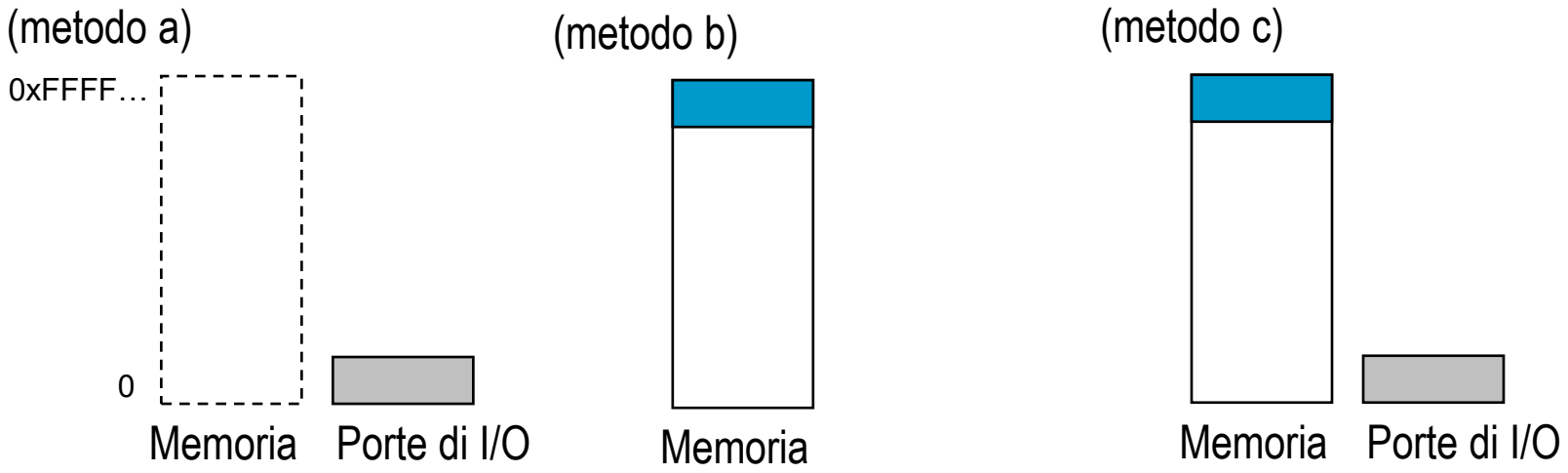
CR=CONTROL REGISTER
SR=STATUS REGISTER
DR=DATA REGISTER

Metodi di comunicazione fra processore e dispositivo

(a) Spazi separati di I/O e memoria

(b) Memory-mapped I/O

(c) Soluzione ibrida



Esempio (architettura x86 antiche):
- IN AL,[0x60] → legge un byte all'indirizzo dello spazio di I/O 0x60

Esempio (architettura RISC-V):
- LB t0,0(0x60) → legge un byte all'indirizzo dello spazio di I/O 0x60 **mappato in memoria**

Esempio (architettura x86 moderne):
- IN AL,0x60 → legge un byte all'indirizzo dello spazio di I/O 0x60
- MOV AL,[0x60] → legge un altro byte all'indirizzo di memoria 0x60 (es. da un altro dispositvo)

Nei casi a e b, i dispositivi si basano solo sulle porte o sulla memoria rispettivamente

Nel caso c, il processore può parlare con i dispositivi usando uno dei due metodi o entrambi

Istruzioni per la comunicazione fra processore e dispositivo

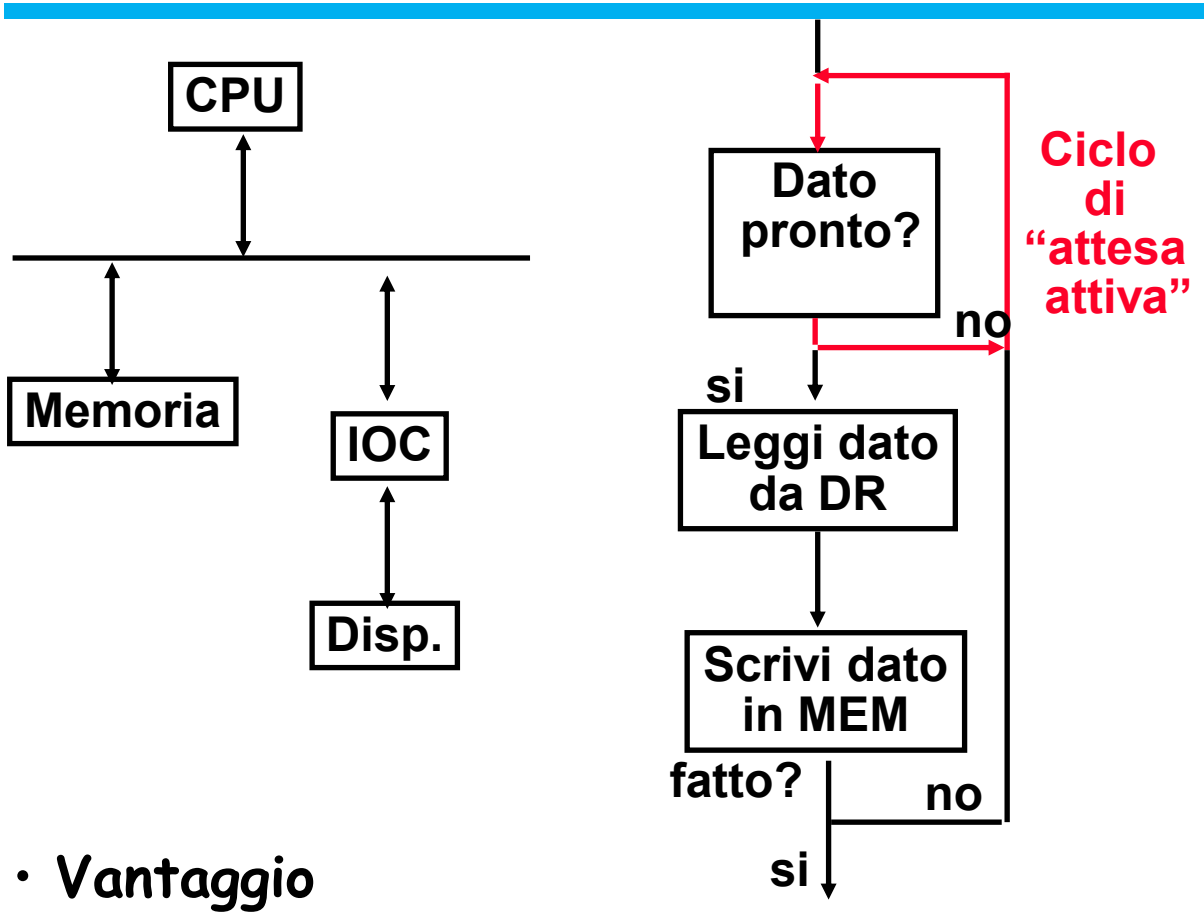
- Se si usano i metodi (a) o (c) per scambiare dati fra processore e memoria si rendono necessarie
  - Istruzioni speciali di I/O (es. Architetture x86)
  - L'istruzione speciale specifica il numero del dispositivo
    - Questo può essere trasmesso "come un indirizzo" sul bus di I/O
  - L'istruzione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

- Se si usa il metodo (b) non è necessario introdurre nuove istruzioni per comunicare con l'I/O
  - è 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 perché risiederanno in una zona riservata allo spazio di indirizzamento kernel
  - Questa tecnica è 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 è notificato che l'operazione è finita con un interrupt dal DMA/IOP verso il processore

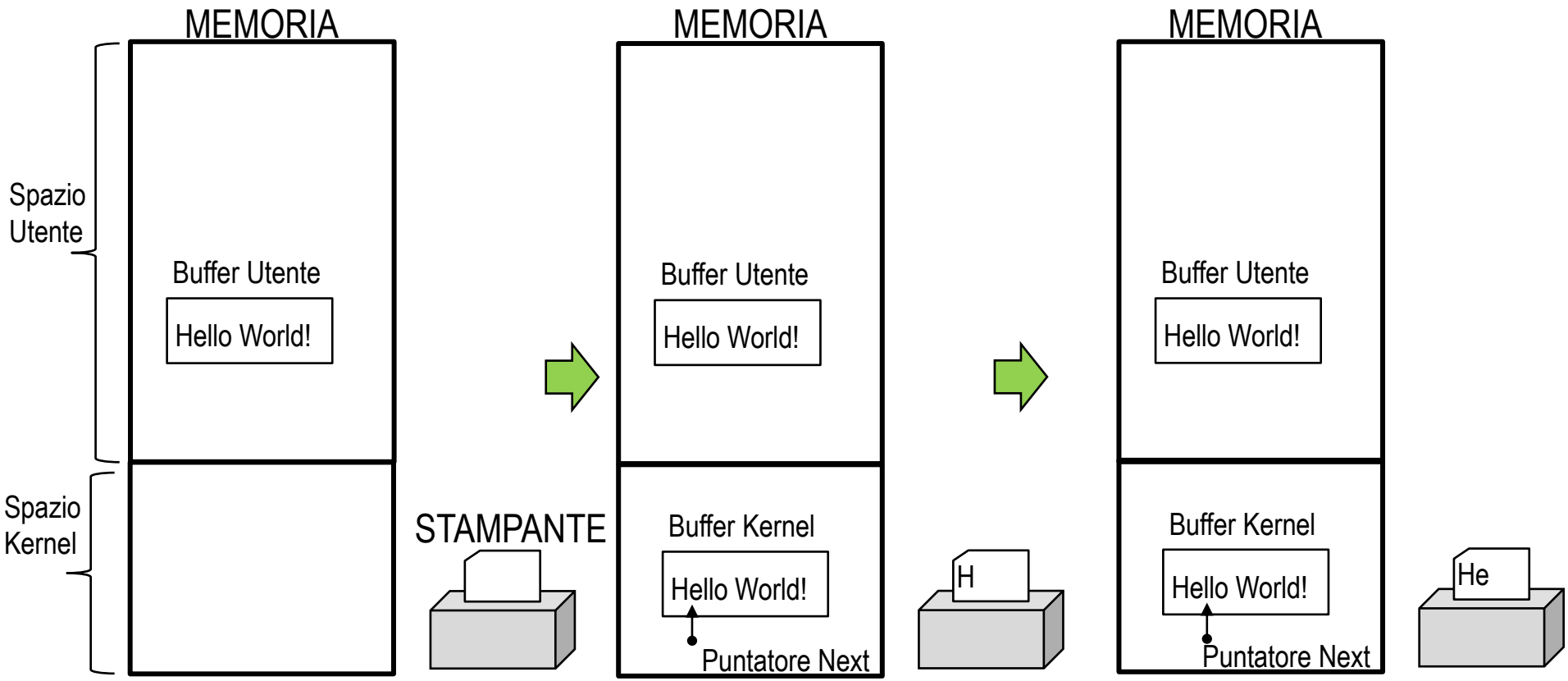
→ Protocollo di POLLING (o di I/O Programmato)



1. La CPU richiede un'operazione di I/O (chiama una system-call)
2. Il controller di I/O effettua l'operazione di I/O
3. Il controller di I/O attiva un bit nello "status register"
   a) Il controller di I/O non informa direttamente la CPU
   b) Il controller di I/O non interrompe la CPU
4. La CPU guarda periodicamente il bit nello "status register"
   a) La CPU può attendere oppure ritornare a vedere più tardi

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

Esempio di Polling: stampa di una stringa



Stampa di una stringa con Polling (Driver)

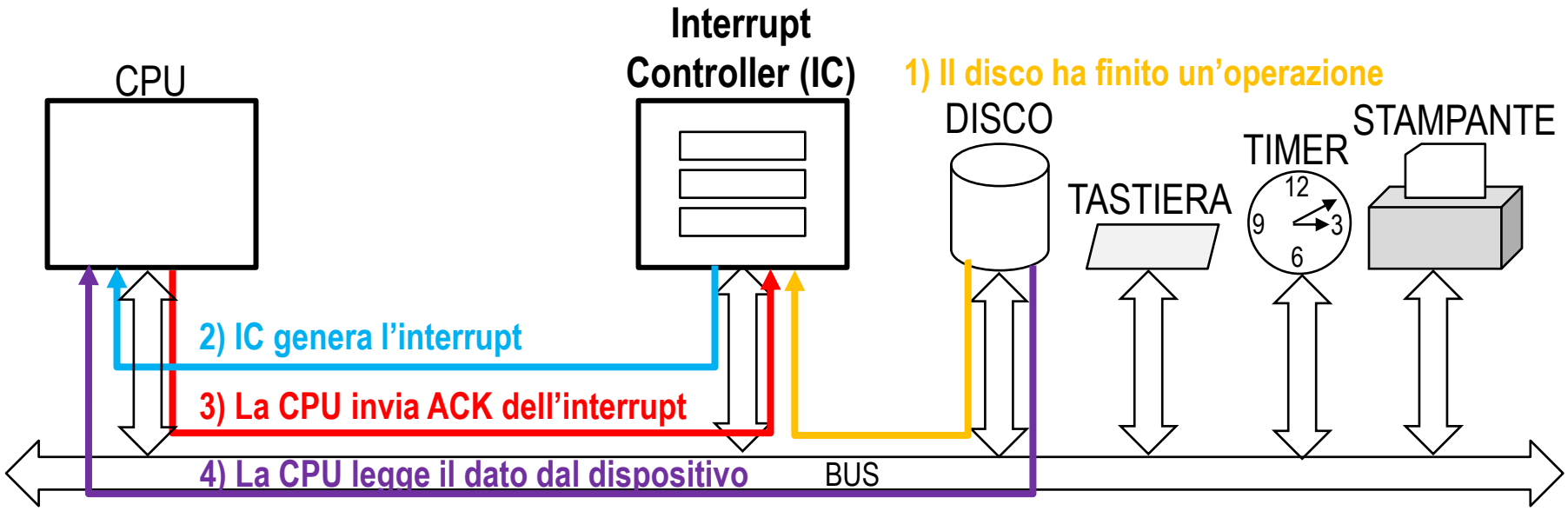
- **Implementazione della SYSTEM-CALL che fa la stampa-stringa:**

```
copy_from_user(buffer, p, count);          /* p è 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();
```

dove la funzione copy_from_user non è altro che:
copy_from_user(buffer, p, count) { for (i=0; i<count; ++i) p[i]=buffer[i] }

→ Protocollo ad INTERRUPT

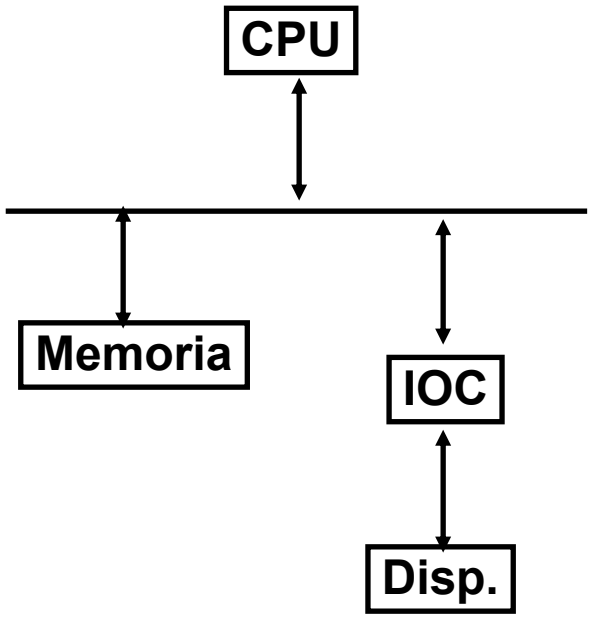
- Evita l'attesa attiva (controlli sullo stato del dispositivo) da parte della CPU
- Il controller del dispositivo interrompe una sola volta quando ha bisogno
- La CPU dà un comando di lettura (es. dato da disco) e questo punto la CPU si occupa di altro… (trascorre tempo)…
  1) Quando il dato è pronto, il dispositivo invia un interrupt al controller di interrupt (IC), che quindi raccoglie gli eventuali interrupt dai vari dispositivi
  2) IC interrompe la CPU
  3) La CPU disabilita gli interrupt e notifica IC (segnale di ACK)
  4) La CPU legge il dato dal disco e riabilita gli interrupt

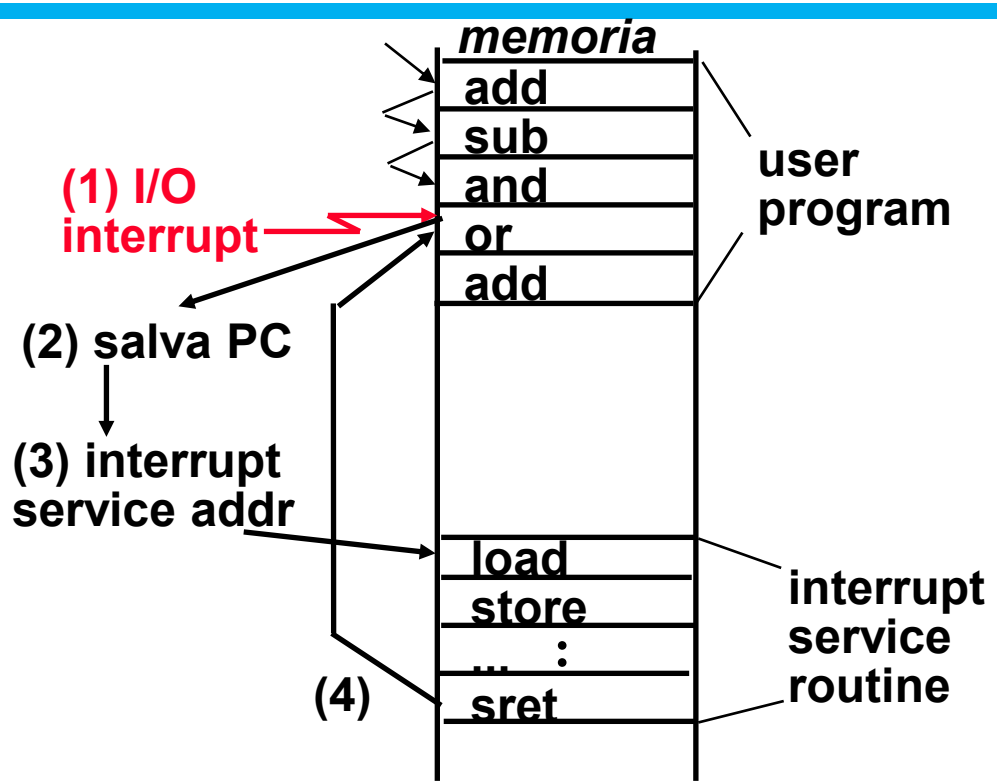


Dal punto di vista della CPU…

- La CPU dà un comando di lettura
- La CPU passa a fare altro lavoro
- Al termine di ogni istruzione guarda se c'è un interrupt pendente
- Se c'è 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 è bloccato solo durante il tempo effettivo per trasferire il dato dal dispositivo alla memoria
- Svantaggio: è 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

La chiamata allo scheduler comporta il blocco (cioè la sua deschedulazione dalla CPU) del processo che ha invocato questa system call

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();
```

Inizialmente avrò k=1

Interrupt

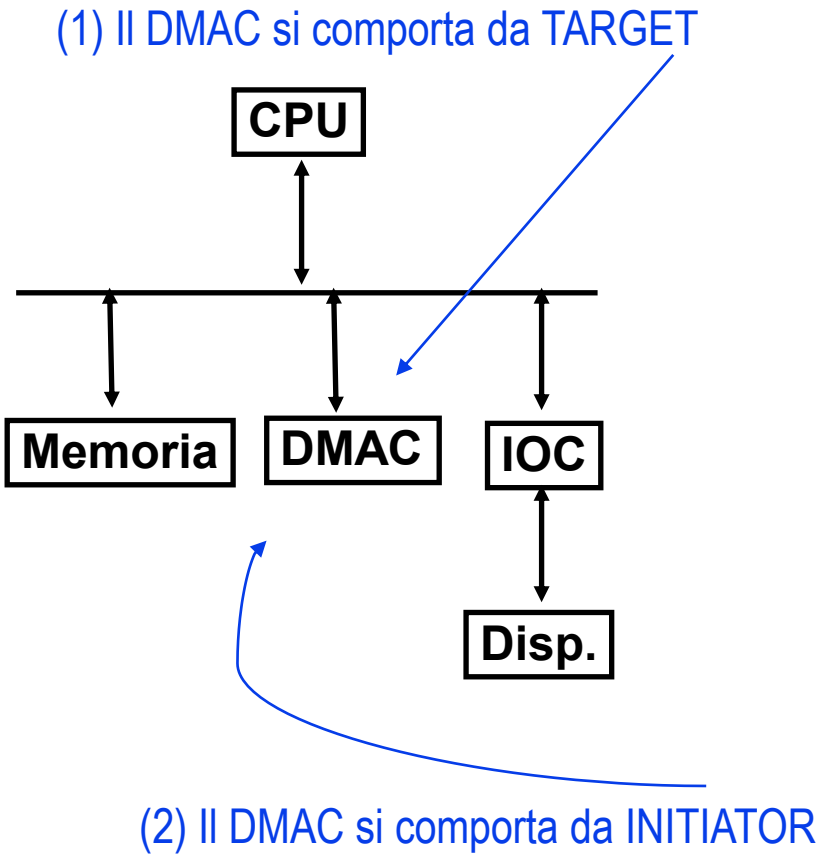
- Per la CPU, l'interrupt è esattamente come un'eccezione salvo che
  - L'interrupt è un evento asincrono
    - Non è provocato da istruzioni
    - Non impedisce il completamento di alcuna istruzione
  - Ad esso fa seguito un trasferimento di informazioni

- L'implementazione dell'interrupt è più complicata di quella di un'eccezione, perché
  - Deve consentire di identificare il dispositivo che ha generato l'interrupt
  - Le richieste di interruzione possono avere priorità 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 è 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

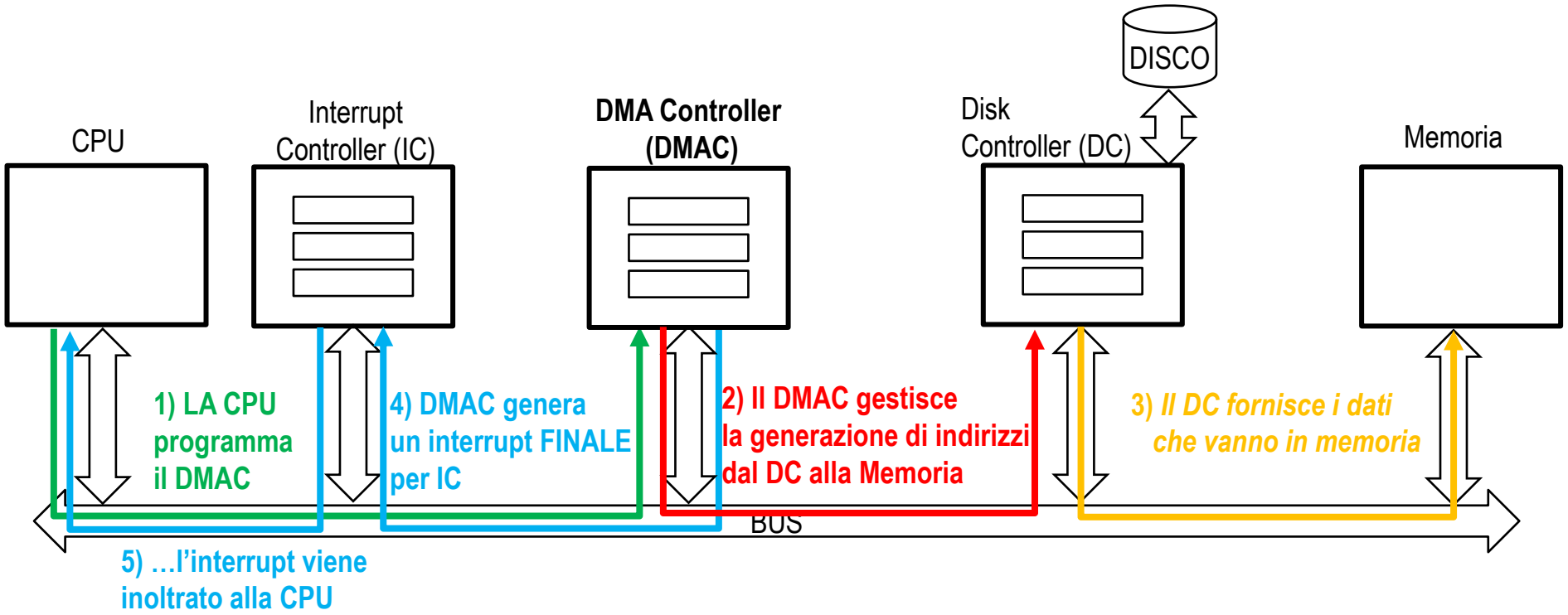
→ Direct Memory Access (DMA)



1) La CPU dice al DMA Controller:
   - Indirizzo iniziale della zona di memoria coinvolta
   - Indirizzo del controller del dispositivo
   - Direzione (Read/Write)
   - Quantità di dati da trasferire
   - Infine la CPU da lo "start"
- La CPU può svolgere altro lavoro

2) Il DMA Controller si occupa del trasferimento
   - genera tutti i necessari segnali per indirizzare la periferica e la memoria e per gestire l'handshake
- Il DMA Controller manda un interrupt quando ha finito

DMA -dettaglio



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();
```

La chiamata allo scheduler comporta il blocco (cioè la sua deschedulazione dalla CPU) del processo che ha invocato questa system call→ il controllo viene quindi passato al sistema operativo che schedula un altro processo

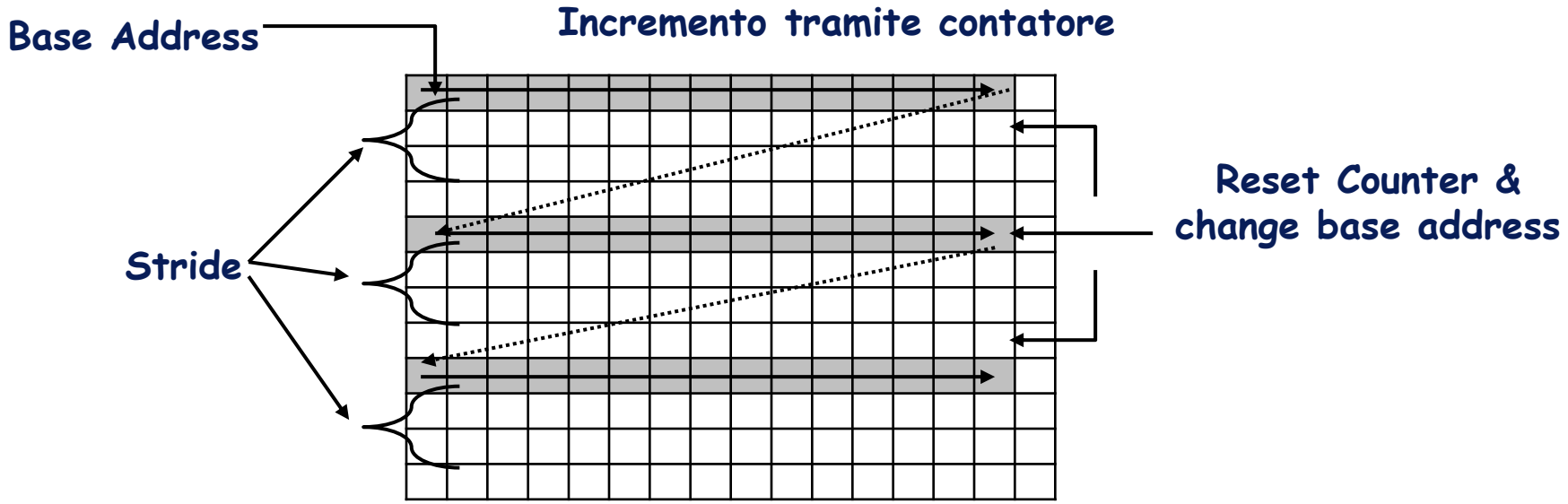
Routine di servizio dell'interrupt

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

DMA Controller (DMAC)

- Il DMA può supportare diverse modalità di indirizzamento
  - 1D, ovvero un singolo registro per gli indirizzi
  - 2D, ovvero due registri per l'indirizzamento
  - 3D, ovvero tre o più registri per l'indirizzamento

- Esempio di indirizzamento 2D
  - Utile ad es. per prelevare i payload dai pacchetti Ethernet



EXTRA SLIDES

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 quantità…
  - 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

→ Direct Memory Access (DMA)

- Interrupt frequenti e polling richiedono l'assistenza della CPU
  - La velocità di trasferimento è limitata
  - La CPU è impegnata in troppe attività

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

- Il DMA controller occupa il bus per un ciclo (di bus)
- Non è un interrupt (opera in "cycle stealing")
  - La CPU non deve fare salvataggio/ripristino stato !
- La CPU eventualmente attende di più quando dovrà 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

Struttura di un Controller DMA Generico

- Sono tipicamente presenti i seguenti segnali
  - 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
  - Bus Requester

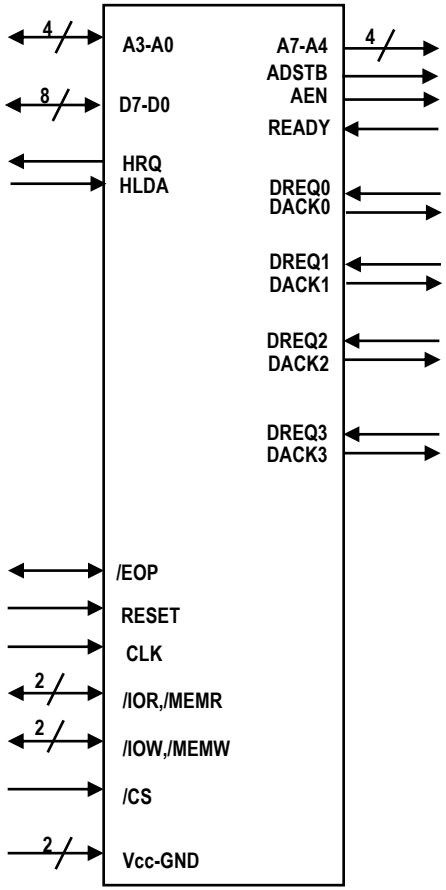
    Utilizzato per richiedere il bus alla CPU
  - Local Peripheral Control

    Permette al controller DMA di selezionare la periferica e gestirla
  - Interrupt signals

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

DMA Controller Intel-8237 (1/3)

- Visione Elettrica del chip (40pin, il pin5 è a Vcc):

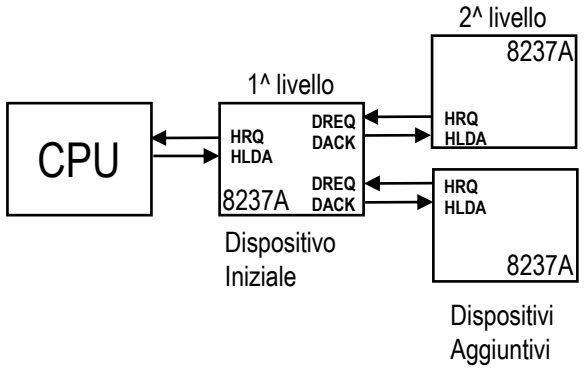


- A3-A0: indirizzamento del chip (e generazione indirizzi) (notare che questi segnali sono *bidirezionali*), insieme a /CS
- A7-A0: solo generazione di indirizzi
- ADSTB,AEN:(il sistema ha 16 bit di indirizzamento, gli A15-8 sono emessi sul bus dati quando AEN è attivo; con ADSTB si permette di memorizzare A15-A8 su un latch esterno
- D7-D0: scambio dati su 8 bit (o uscita di A15-A8)
- /IOR, /MEMR: lettura da/verso I/O o MEMoria
- /IOW, /MEMW: scrittura da/verso I/O o MEMoria
- /CS, Vcc-GND : Chip-Select e Alimentazione
- CLK, RESET: clock, reset
- READY: opposto del "WAIT" che abbiamo già visto per prolungare i trasferimenti in caso di read e write (attivato dal target)
- HRQ,HLDA: HRQ è la richiesta *al processore* di accesso al bus e HLDA è il permesso (grant) *da parte del processore*
- DREQx-DACKx: sono usate dalla periferica x (x=1..4) per richiedere l'uso del canale di DMA 'x' e iniziare un trasferimento di un blocco di dati. L'handshake per tale richiesta rispetta la modalità asincrona (REQ/ACK)
- /EOP: in ingresso è la richiesta di terminare le operazioni e re-inizializzare alla condizione iniziale i 4 canali (End-Of-Process); in uscita, indica l'eventuale presenza di una richiesta interna di terminare le operazioni

DMA Controller Intel-8237 (2/3)

- 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
    - Viene trasferito un solo byte alla volta
  - Trasferimento a blocchi
    - Il trasferimento è attivato da DREQ e bloccato* da DACK
  - Modalità a domanda
    - Il trasferimento è attivato da DREQ e bloccato* da DACK oppure dalla disattivazione di DREQ
  - Modalità a cascata – speciale modalità per gestire più controllers 8237 in cascata e ottenere così più canali DMA
    - In questo caso i segnali HRQ/HLDA di un 8237 di secondo livello anziché essere connessi al processore sono connessi ai segnali DREQ/DACK di uno dai canali di un 8237 di primo livello

* Inoltre il trasferimento può essere bloccato dall'attivazione di DACK, dal segnale /EOP o dal termine del conteggio dei dati da trasferire



DMA Controller Intel-8237 (3/3)

- Trasferimento dati
  - Da periferica a memoria (WRITE) o viceversa (READ)
  - Da memoria a memoria (unendo due canali, ma non è usato nei PC)

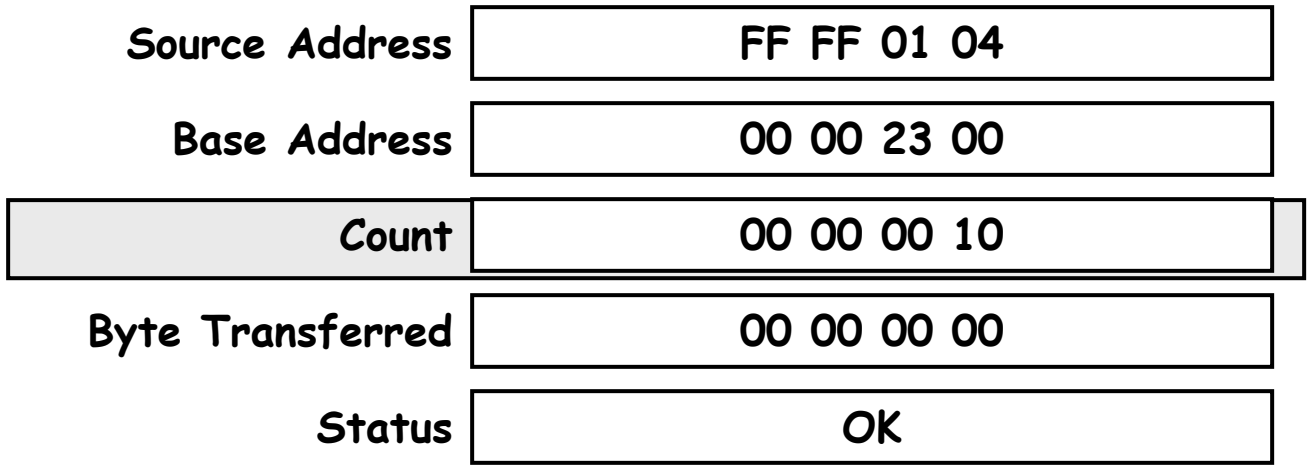
- Verify Transfer Mode :
  - Speciale modalità di trasferimento dati, che è usata nei PC per generare indirizzi dummy per il refresh della memoria DRAM
  - Guidata da un interrupt (ogni 15ms) proveniente del timer 8254

- Gestione della priorità:
  - Schema a priorità fissa: la priorità dipende dalla posizione
  - Schema a priorità rotante: dinamicamente l'ultimo dispositivo servito diventa quello a priorità più bassa

Controller DMA Generico: funzionamento

Programmazione del controller:

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



Es. Registri di controllo del DMA

Controller DMA Generico: funzionamento (2)

2) Inizio del trasferimento

- Il traferimento ha inizio su una esplicita richiesta o dalla periferica 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 più 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 (es. nel 8237, A4-7)
  - Multiplexata (es. nel 8237, D7-0 ➔ A15-A8)
- inoltre il DMAC fornisce altri segnali di controllo, come R/W, strobe per lavorare sul bus
- I più recenti DMAC sono abbastanza generici per poter funzionare con diverse famiglie di processori

Controller DMA Generico: funzionamento (3)

5) Trasferimento dei dati

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

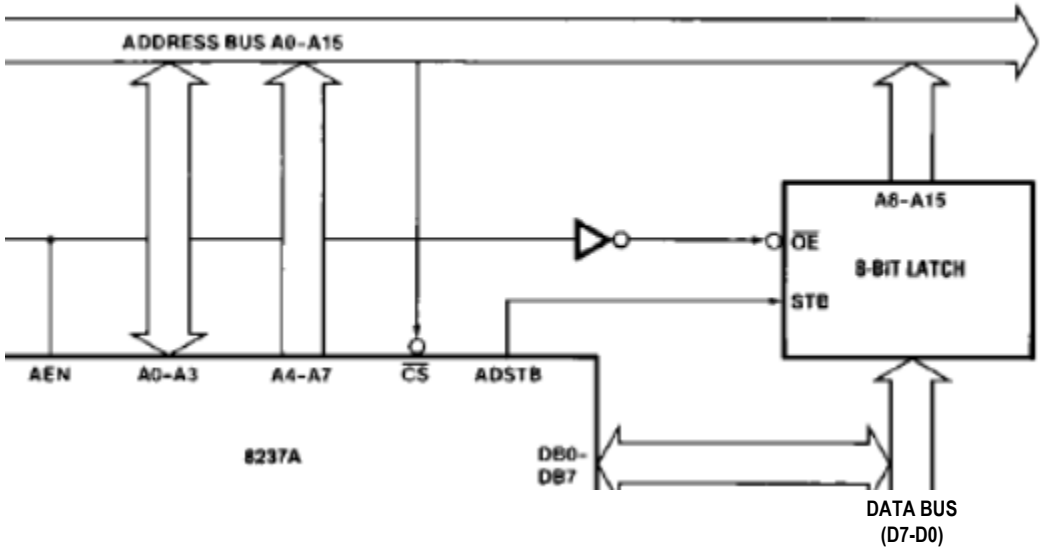
6) Aggiornamento dell'indirizzo

- 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

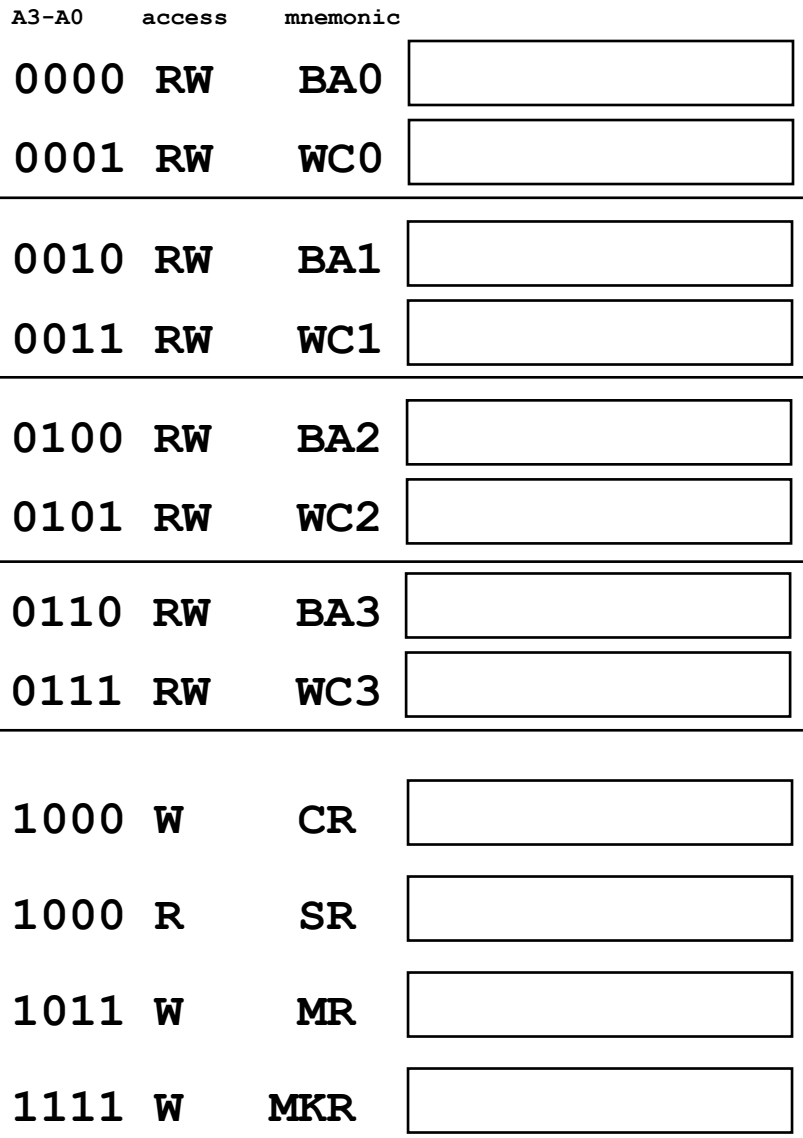
8237A - Dettaglio su uso di AEN e ADSTB



- Nota: il bus indirizzi è a 16-bit
  - Per poter inviare 16 bit il 8237 usa D7-D0 per inviare A15-A8

DMA Intel-8237A

- **Visione logica del chip (tutti registri a 8 bit):**



Ch.0 - Base and Current Address
Scrivo l'indirizzo base e leggo l'indirizzo attuale

Ch.0 - Base and Current Word Count
Scrivo i byte da trasf.* e leggo i byte che resta da trasf.*

Ch.1 - Base and Current Address
Scrivo l'indirizzo base e leggo l'indirizzo attuale

Ch.1 - Base and Current Word Count
Scrivo i byte da trasf.* e leggo i byte che resta da trasf.*

Ch.2 - Base and Current Address
Scrivo l'indirizzo base e leggo l'indirizzo attuale

Ch.2 - Base and Current Word Count
Scrivo i byte da trasf.* e leggo i byte che resta da trasf.*

Ch.3 - Base and Current Address
Scrivo l'indirizzo base e leggo l'indirizzo attuale

Ch.3 - Base and Current Word Count
Scrivo i byte da trasf.* e leggo i byte che resta da trasf.*

Command Register

Status Register

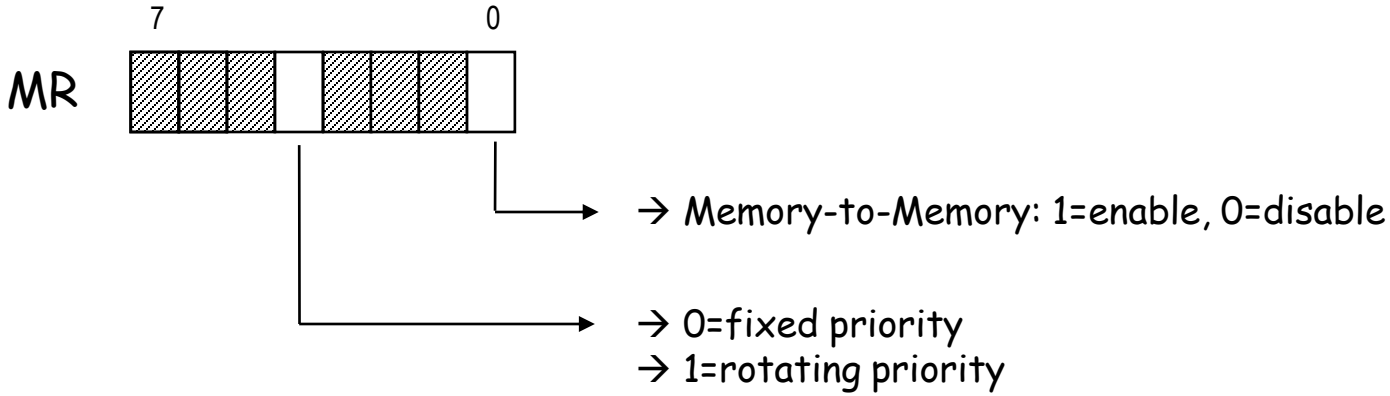
Mode Register

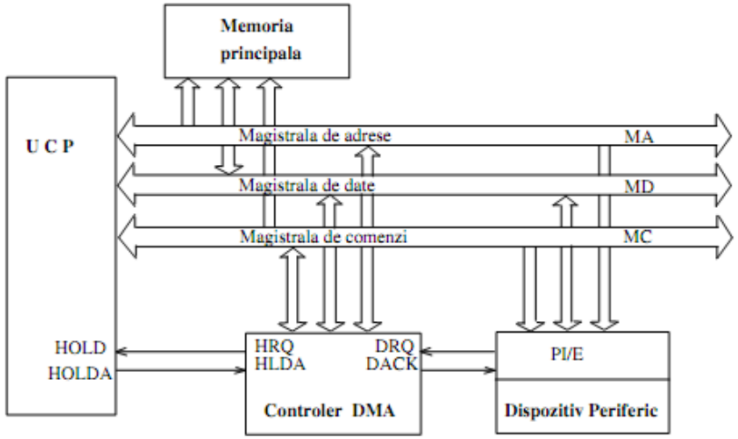
MasK Register

Nota: nei registri BAx,WCx
viene scritto prima LSB e poi MSB
(inoltre all'inizio va dato un commando di Clear-Byte-Pointer-FlipFlop, ovvero scrittura all'indirizzo 1100)

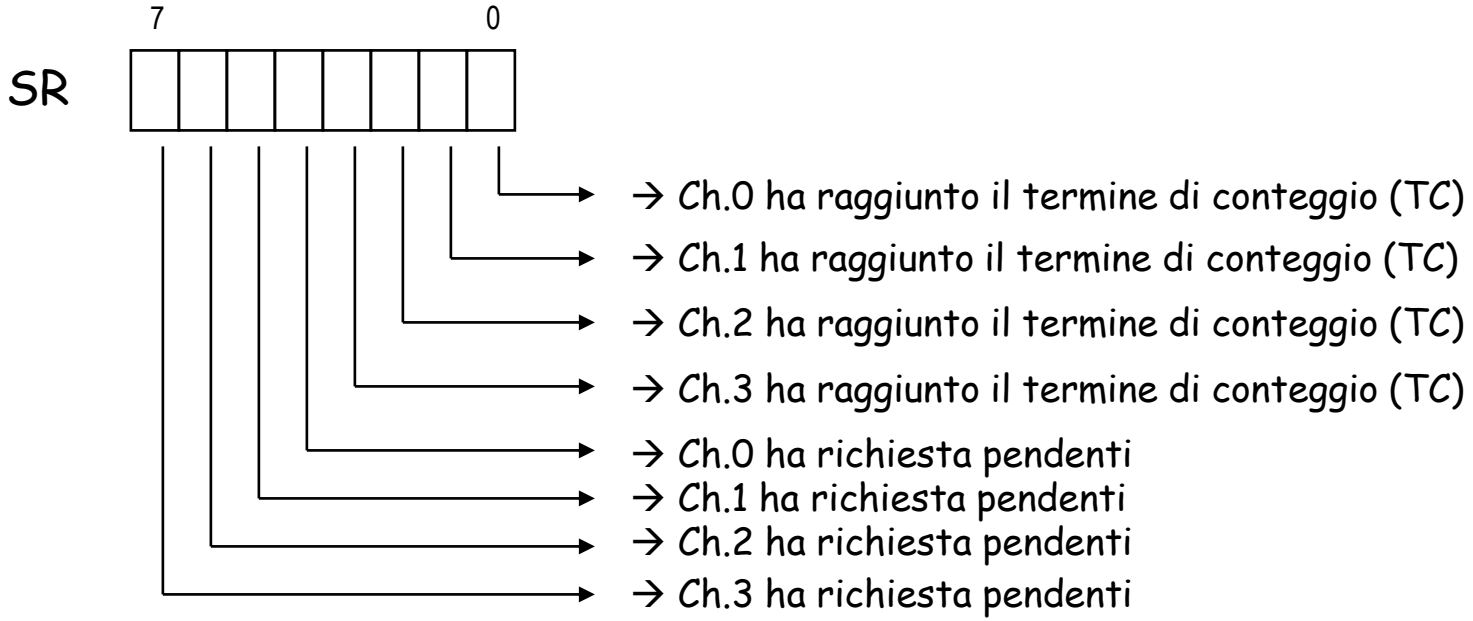
*diminuiti di 1

8237A CR – Command Register

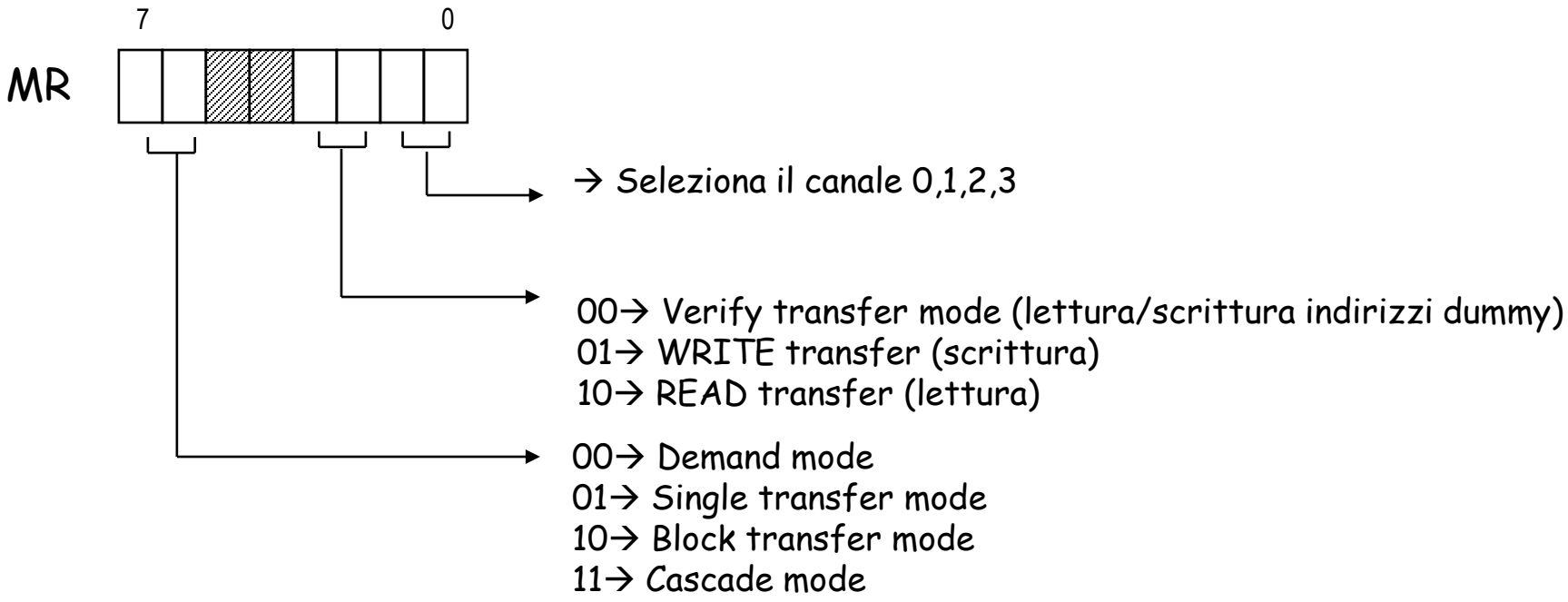




8237A SR – Status Register

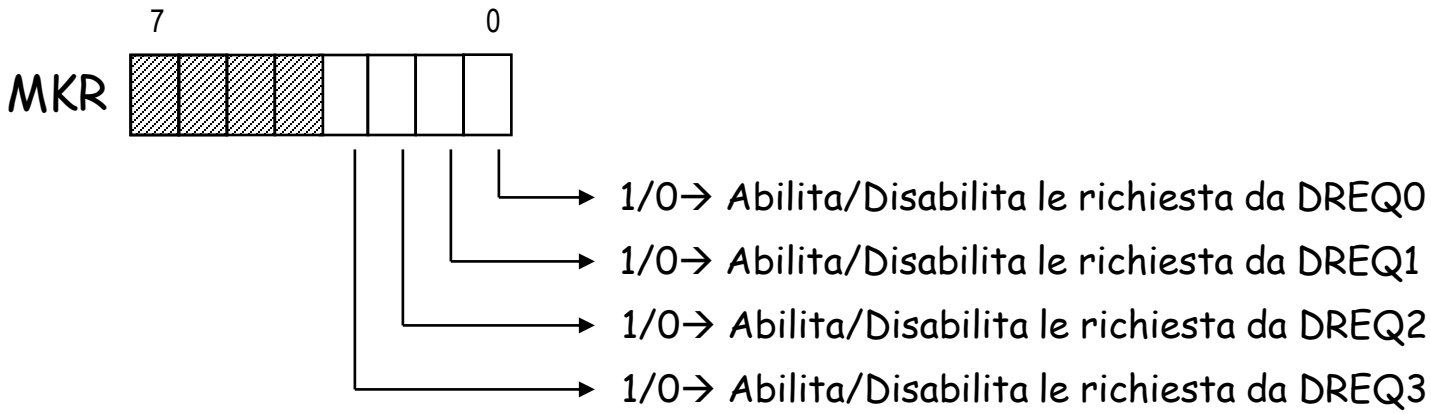


8237A MR – Mode Register



In Demand mode la periferica richiede attenzione al DMAC e fornisce il dato e il DMAC provvede invece a inviare il dato verso la memoria; ovvero il DMAC preleva il dato dalla memoria e lo invia alla periferica (non c'è bisogno di indirizzamento della periferica perché il DMA è il master del bus di Sistema e la periferica gestisce il colloquio con il DMAC con i segnali DREQ/DACK).

8237A MRK – MasK Register



PC - I/O MAP

Table 5.4 I/O map for a Pentium PC

I/O device / port	I/O address, Hex	Size, bytes	I/O device/port	I/O address, Hex	Size, bytes
DMA controller-1	0000–000F	16	Numeric coprocessor	00F0–00FF	16
DMA controller-2	00C0–00DF	32	PCI configuration register	0CFC–0CFF	4
DMA page registers	0080–008F	16	CMOS RAM/RTC	0070–0073	4
Interrupt controller-1	0020–0021	2	Floppy disk controller–1	03F0–03F5	6
Interrupt controller-2	00A0–00A1	2	USB controller	Relocatable	64
Counter/Timer-1	0040–0043	4	PS/2 Keyboard controller	0060,0062,0064,0066	4
Counter/Timer-2	0048–004B	4	NMI, speaker control	0061,0063,0065,0067	4
Primary IDE channel	01F0–01F7	8	Audio	0220–022F /	16
Primary IDE command port	03F6	1		0240–024F	
Secondary IDE channel	0170–0177	8	APM Control Port	00B2	1
Command port	0376	1	APM Status Port	00B2	1
COM1	03F8–03FF	8	LPT1 (SPP/EPP)	0378–037A/037F	3/8
COM2	02F8–02FF	8	LPT2 (SPP/EPP)	0278–027A/027F	3/8
COM3	03E8–03EF	8	LPT3 (SPP/EPP)	0228–022A/022F	3/8
COM4	02E8–02EF	8	ECP (for LPTn)	LPTn, LPTn+400	3/8
Display adapter	03B0–03BB	12			
	03C0–03DF	32			

PC - IRQ, DMA, I/O Addresses

Table 5.5 The IRQs, DMA channels and I/O addresses for the devices in a Pentium PC

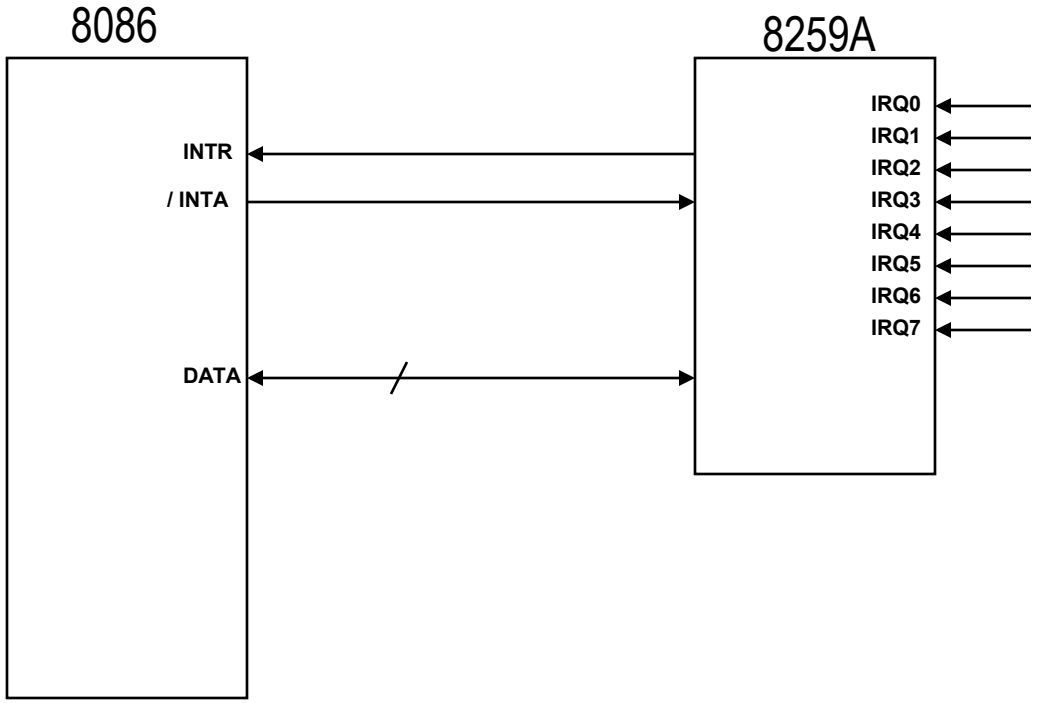
Device	*IRQ*	*DMA channel*	*I/O addresses*
Floppy disk controller-FDD1	6	2	03F0H–03F5H
Floppy disk controller-FDD2	6	2	0370H–0375H
Primary IDE/hard disk controller, HDD1	14	–	01F0H–01F7H
Secondary IDE/hard disk controller, HDD2	15	–	0170H–0177H
Game port/joystick adapter	–	–	0201H–0207H
Serial port-COM1	4	–	03F8H–03FFH
Serial port-COM2	3	–	02F8H–02FFH
Serial port-COM3	4	–	03E8H–03EFH
Serial port-COM4	3	–	02E8H–02EFH
Parallel port-LPT1	7	–	0378H–037FH
Parallel port-LPT2	5	–	0278H–027FH
Parallel port-LPT3	5	–	03BCH–03BFH
PS/2 mouse	12	–	0640H
Sound blaster	5	1	0220H–022FH
	7	3	0240H–024FH
Network interface cards/Ethernet adapter	2	1	0280H–0283H
	3	3	0280H–029FH
	4	5	02A0H–02A3H
	5	7	02A0H–02BFH
SCSI adapter	10	3	0130H–014FH
	11	5	0140H–015FH

Interrupt Multipli

- Ogni interrupt ha una priorità associata
- Dispositivi a più alta priorità possono interrompere anche quando si servono dispositivi a più bassa priorità
- Nel caso di bus mastering l'ulteriore interrupt può 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

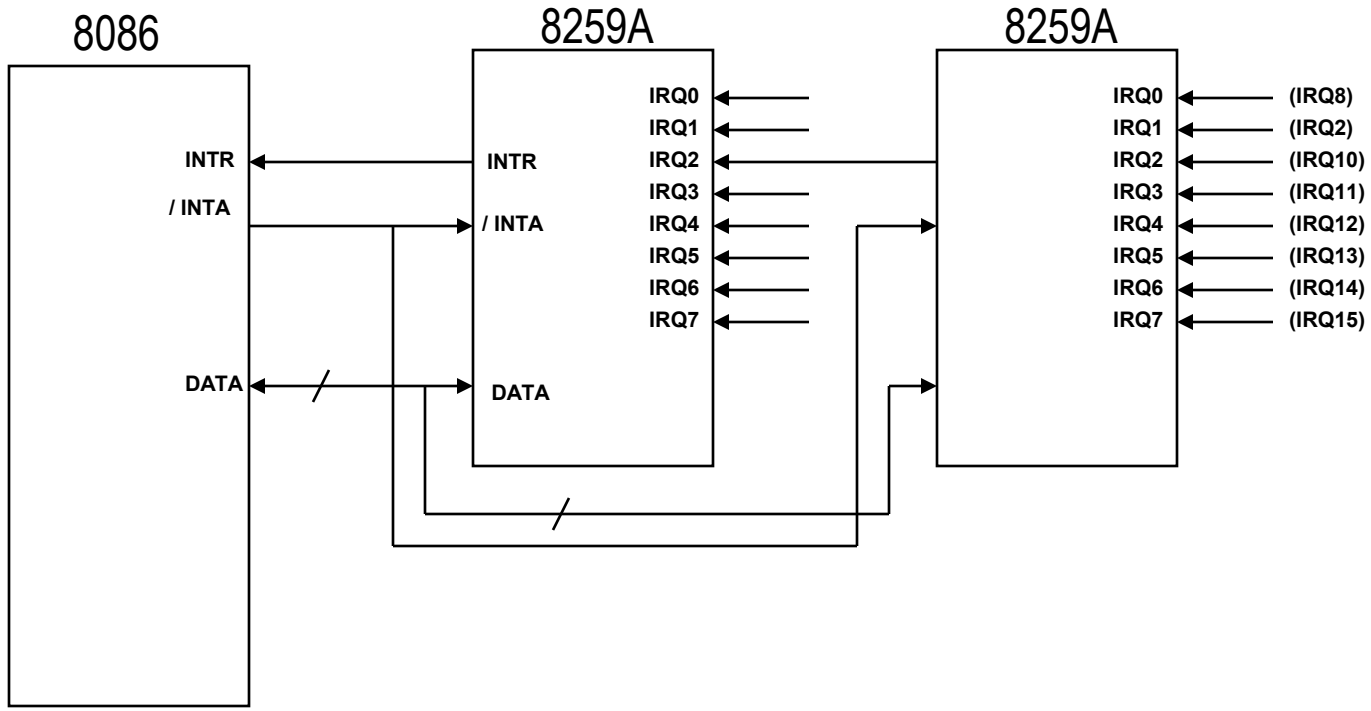


Sequenza di Eventi

- L' 8259A accetta interrupt
- L' 8259A determina la priorità
- 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" cioè un numero che identifica la sorgente di interrupt
- La CPU serve l'interrupt mandando in esecuzione la relativa routine di servizio

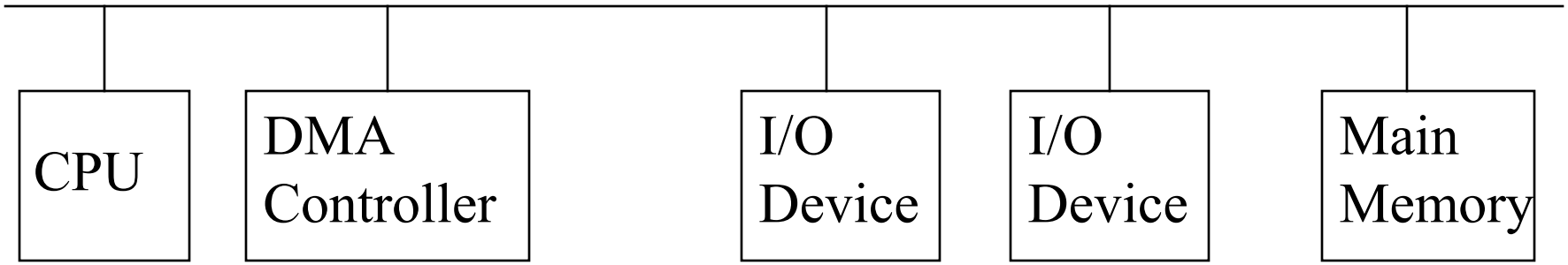
Cascading

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

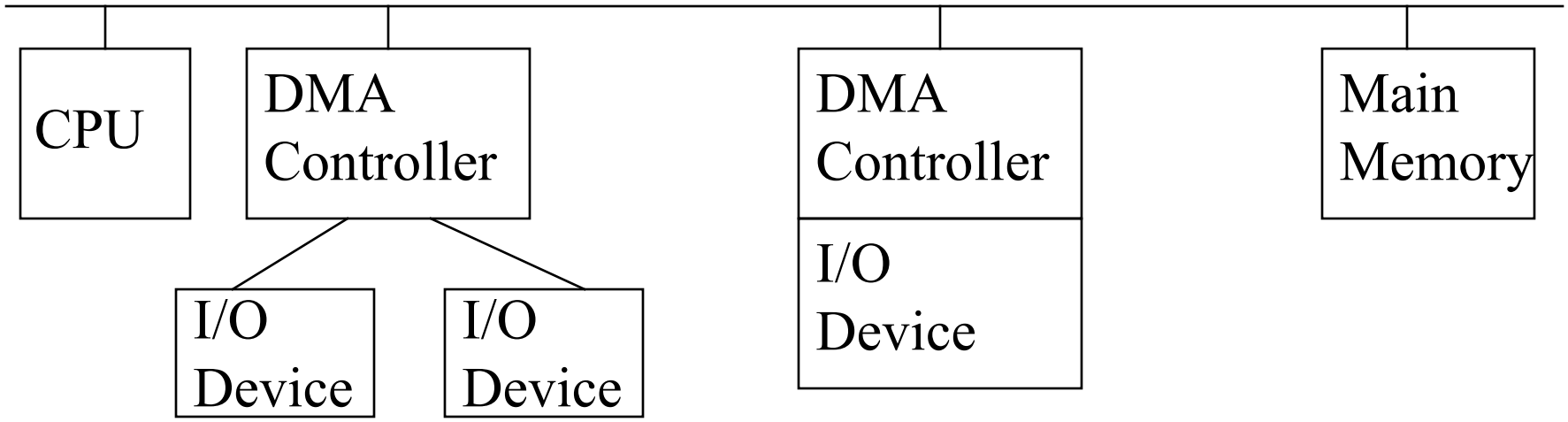


DMA: Configurazioni (1)

- 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 più tempo

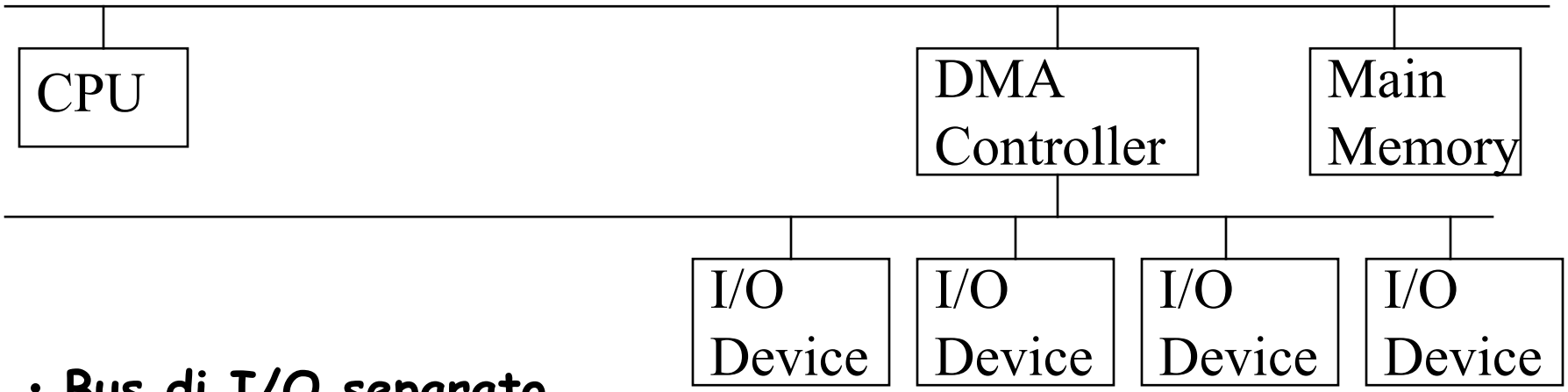


DMA: Configurazioni (2)



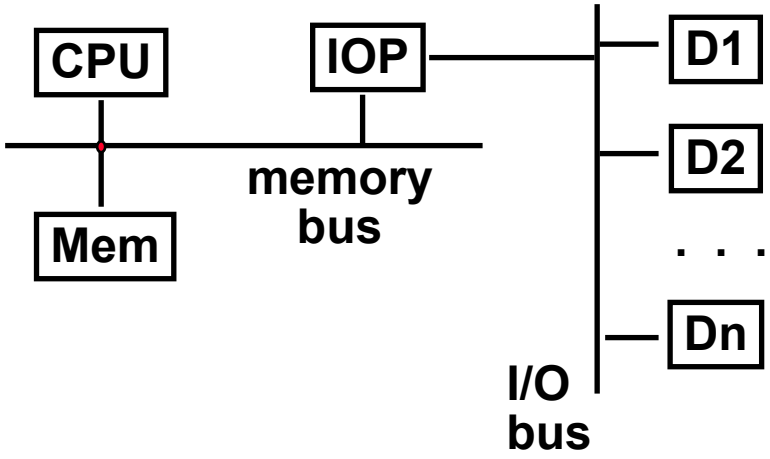
- Singolo Bus, DMA controller integrato
- Il controller può gestire più di un dispositivo
- Ogni trasferimento usa il bus una sola volta
  - da DMA a memoria
- La CPU dimezza le volte in cui trova occupato il bus

DMA: Configurazioni (3)



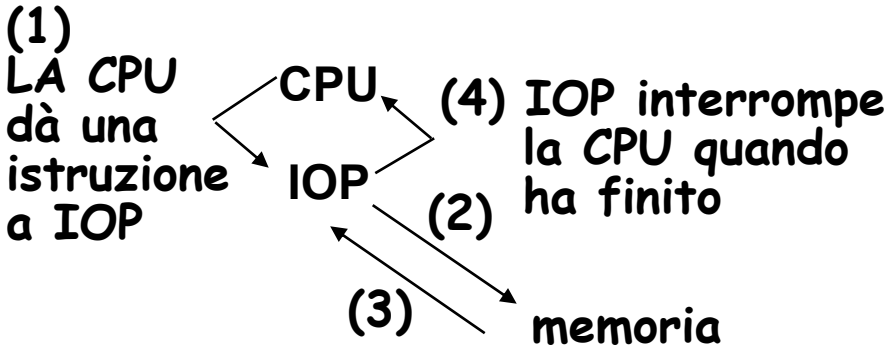
- Bus di I/O separato
- Il bus supporta tutti i dispositivi abilitati al DMA
- Ogni trasferimento usa il bus una sola volta
  - DMA to memoria
- La CPU dimezza le volte in cui trova occupato il bus rispetto al primo caso

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

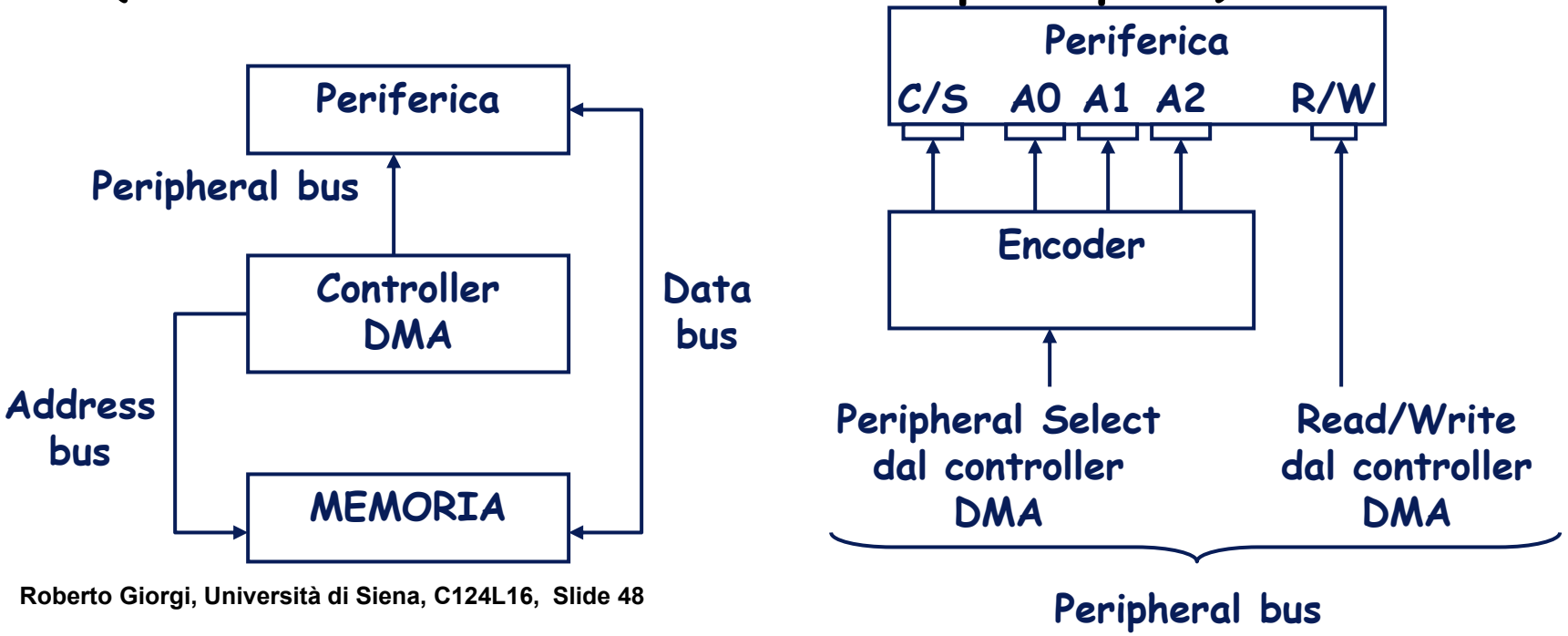


Modelli di controllers DMA

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

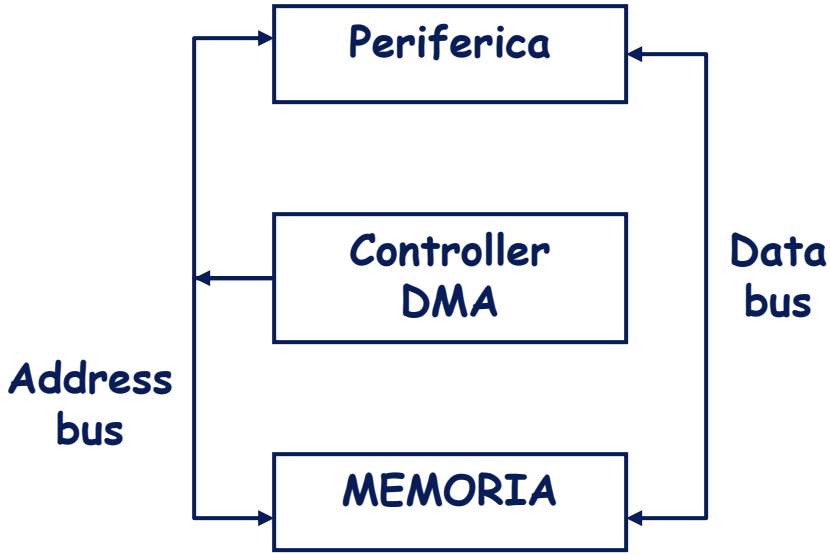
Trasferimento dati singolo (indirizzamento implicito)

- 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 può essere necessario un encoder per selezionare un dato indirizzo (implicito) della periferica
- Se è 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

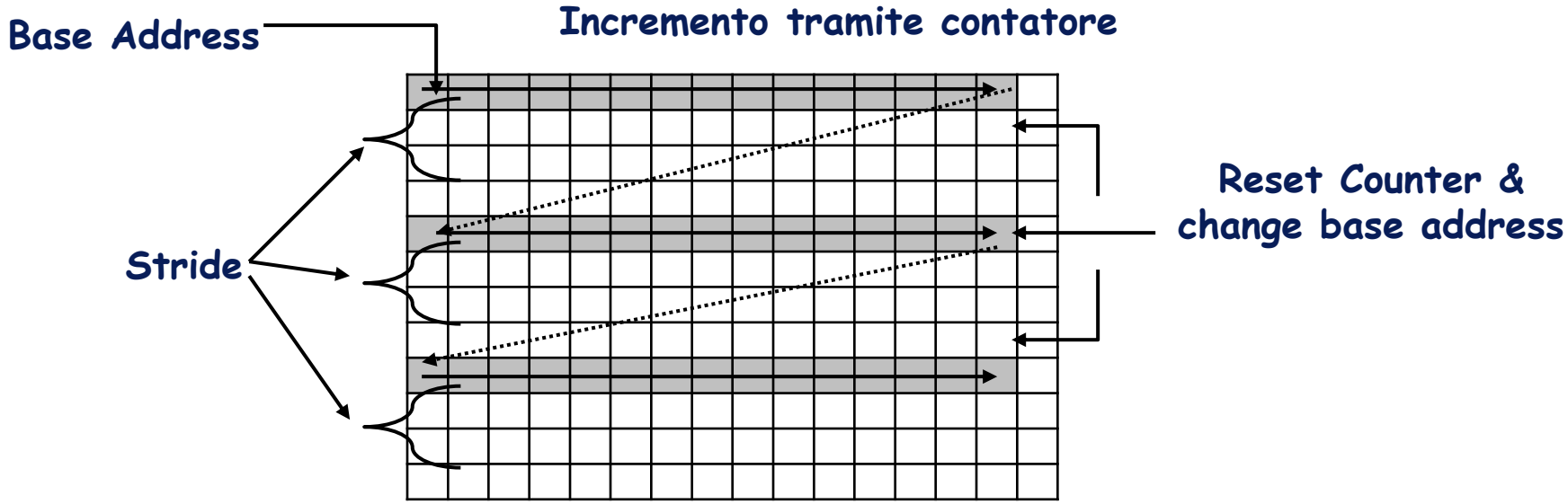
- Usa un indirizzo base e un contatore per definire la sequenza di indirizzi da usare nei cicli DMA
- Svantaggi:
  - Una volta che il blocco di dati è stato trasmesso, l'indirizzo ed il contatore sono resettati potendo sovrascrivere accidentalmente dati precedenti
  - Diventa necessario un interrupt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

2D Model

- Es. Implementazione di protocolli di comunicazione basati sullo scambio di pacchetti
  - Devo spezzare grosse quantità 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 è 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 può usare la modalità 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
- è 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

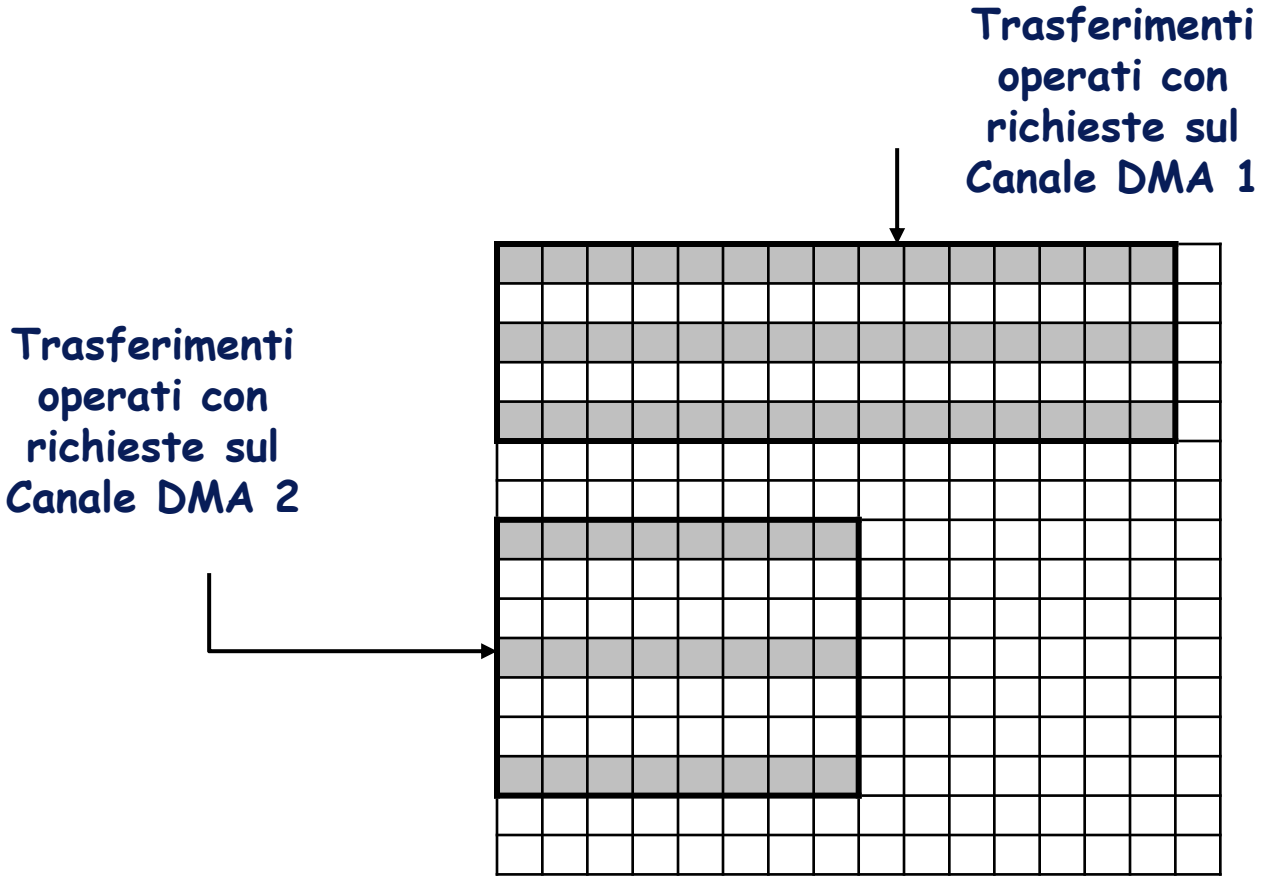
- Ogni DMA deve essere pre-programmato con un blocco di parametri per poter funzionare correttamente
- L'interfaccia hardware è comune per quasi tutti i differenti set di parametri
- Ogni periferica che necessita di un trasferimento deve chiedere alla CPU di programmare correttamente il DMA (interrupt)
- 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 senza 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 più 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
    - Può 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 può usare il bus in maniera non ottimale (lasciando "buchi"), ma "se è onesto" funziona neglio del trasferimento a blocchi

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