



Politecnico di Torino

III Facoltà di Ingegneria
Corso di Ingegneria Informatica

Wiki Book

Architetture dei Sistemi di Elaborazione

Gestione dei periferici nell'architettura 80x86

A cura di:

Marco Palena:

- Sottosistema di I/O nell'architettura 80x86;
- Gestione degli interrupt nell'architettura 80x86;
- Direct Memory Access nell'architettura 80x86;

Giovanni Pessiva:

- Interfaccia parallela 8255;
 - Interfaccia seriale 8250;
 - Temporizzatore di intervalli 8253;
-

ANNO ACCADEMICO 2010/2011

Indice

Indice.....	1
1. Sottosistema di I/O nell'architettura 80x86.....	5
1.1 Modello del sottosistema di I/O.....	5
1.2 Interfacce dei periferici.....	6
1.3 Indirizzamento dei periferici.....	8
1.3.1 Isolated I/O.....	9
1.3.2 Memory mapped I/O.....	11
1.4 Gestione dei dispositivi di I/O.....	13
1.4.1 Parametri fondamentali di I/O.....	13
1.4.2 Tipi di dispositivi di I/O.....	14
1.4.3 Metodi di gestione dei dispositivi di I/O.....	15
2. Interfaccia parallela 8255.....	17
2.1 Modello 8255.....	17
2.1.1 Chip 8255.....	17
2.1.2 Modello logico 8255.....	18
2.2 Programmazione dell'8255.....	19
2.2.1 Selezione del modo di funzionamento.....	20
2.2.2 Modalità Single Bit Set/Reset.....	21
2.3 Modi di funzionamento.....	21
2.3.1 Modo 0.....	23
2.3.2 Modo 1.....	24

2.3.3 Modo 2.....	29
3. Interfaccia seriale 8250.....	32
3.1 Le comunicazioni seriali.....	32
3.1.1 Modi operativi.....	32
3.1.2 Trasmissione asincrona.....	32
3.1.3 Bit per carattere.....	34
3.1.4 UART.....	34
3.2 Modello 8250.....	35
3.2.1 Piedinatura.....	35
3.2.2 Registri interni.....	37
3.3 Programmazione dell'8250.....	38
3.3.1 Line status register.....	38
3.3.2 Line control register.....	39
3.3.3 Interrupt enable register.....	41
3.3.4 Interrupt identification register.....	41
4. Temporizzatore di intervalli 8253.....	43
4.1 Modello 8253.....	43
4.1.1 Il chip 8253.....	43
4.1.2 Modello logico 8253.....	44
4.1.3 I contatori.....	45
4.2 Programmazione dell'8253.....	45
4.2.1 Registro di controllo.....	46
4.3 Modi di funzionamento.....	46

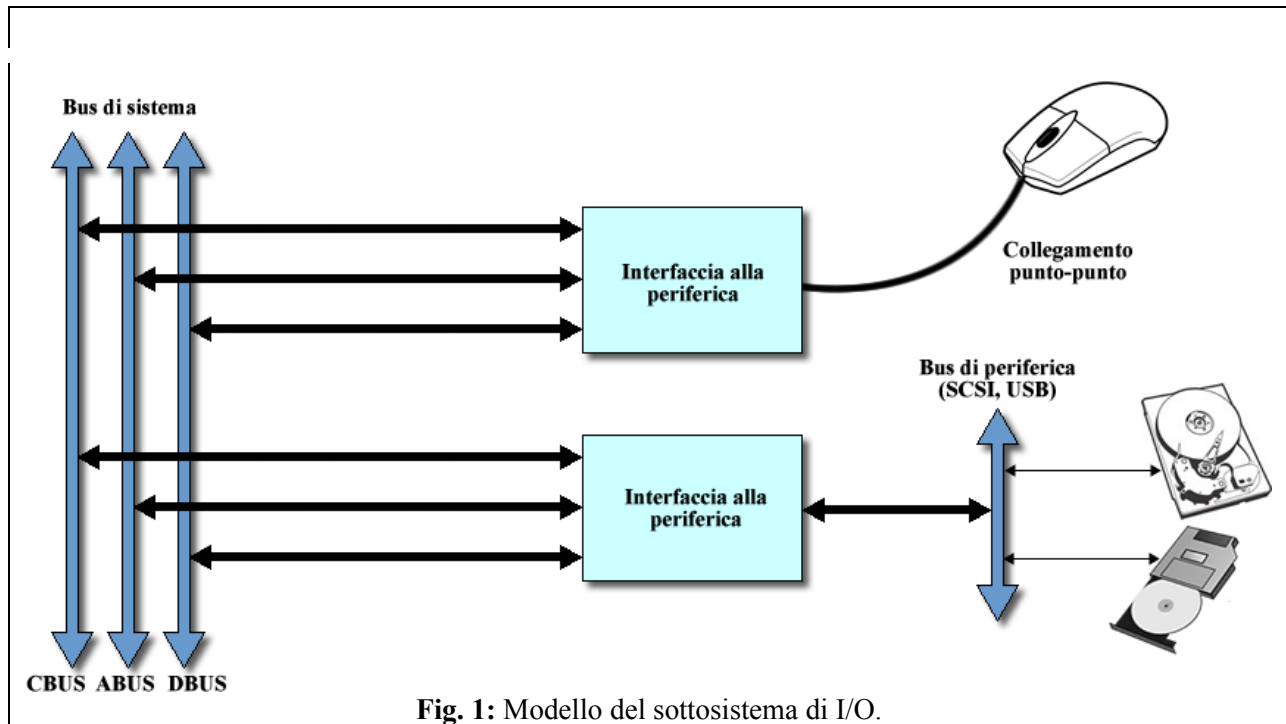
4.3.1 Modo 0 (000).....	47
4.3.2 Modo 1 (001).....	48
4.3.3 Modo 2 (X10).....	48
4.3.4 Modo 3 (X11).....	49
4.3.5 Modo 4 (100).....	49
4.3.6 Modo 5 (101).....	50
5. Interrupt.....	52
5.1 Gestione in interrupt dei dispositivi.....	52
5.1.1 Principi progettuali della gestione I/O in interrupt.....	54
5.2 Interrupt nell'architettura 80x86.....	56
5.2.1 Tipi di interrupt.....	58
5.2.2 Gestione degli interrupt.....	59
5.2.3 Priorità degli interrupt.....	62
5.2.4 Protocollo di interrupt.....	64
5.2.5 Interrupt Vector Table.....	68
5.3 Interrupt controller 8259/APIC.....	70
5.3.1 Struttura del controllore 8259.....	72
5.3.2 Sequenza di interruzione.....	75
5.3.3 Programmazione del controllore 8259.....	76
5.3.4 Modalità operative ed impostazioni.....	91
5.3.5 Collegamento in cascata di controllori 8259.....	101
5.4 Gestione delle interruzioni hardware nei PC.....	104
5.4.1 Gestione delle interruzioni mascherabili.....	104
5.4.2 Gestione delle interruzioni non mascherabili.....	106

6. Direct Memory Access nell'architettura 80x86.....	108
6.1 Schema di collegamento del DMA controller.....	110
6.2 Metodi di trasferimento in DMA.....	112
6.3 Cicli di DMA.....	113
6.4 Sequenza di trasferimento DMA.....	114
6.5 DMA Controller Intel 8237.....	117
6.5.1 Segnali e schema di collegamento.....	117
6.5.2 Modalità di funzionamento.....	121
6.5.3 Cicli di DMA.....	123
6.5.4 Canali di DMA.....	126
6.5.5 Canali di DMA.....	127
6.5.6 Modalità di trasferimento.....	129
6.5.7 DMA controller 8237 in cascata.....	130
6.6 Gestione dei trasferimenti in DMA nei PC.....	131
Riferimenti.....	133

1. Sottosistema di I/O nell'architettura 80x86

1.1 Modello del sottosistema di I/O

Nell'architettura 80x86 l'interazione tra la CPU e i dispositivi periferici avviene secondo il modello presentato in Fig. 1.



Tale modello è costituito da quattro componenti fondamentali, ciascuna delle quali concorre alle attività di input/output:

- **Bus di sistema:** insieme di segnali che realizzano le transazioni di lettura o scrittura orientate all'input/output (*cicli di I/O*), a fronte di un'istruzione di I/O. Tali transazioni non sono altro che cicli di bus di lettura o scrittura, eventualmente estesi con un numero opportuno di periodi di wait, relativi ad indirizzi associati a dispositivi periferici;
- **Interfaccia:** dispositivo alla base del funzionamento dei periferici che permette il dialogo tra uno specifico dispositivi di I/O e la CPU. Per ogni periferico (o classe di periferici) si ha una interfaccia che svolge principalmente i seguenti compiti:

- Tradurre, conoscendo i protocolli di comunicazione impiegati dal bus di sistema e le caratteristiche del periferico, la semantica dei cicli di bus in un insieme di segnali congrui con il funzionamento del periferico;
- Sincronizzare la velocità di trasferimento del periferico con quella del bus di sistema. Il dispositivo periferico infatti è caratterizzato da una velocità di trasferimento (dipendente dal parallelismo e dalla frequenza di clock del periferico stesso) che in generale è minore di quella del bus di sistema;
- Le interfacce più evolute possono essere programmate per scegliere la modalità operativa di dialogo da utilizzare per il periferico, all'interno di una certa gamma di modalità operative disponibili;
- **Collegamento tra l'interfaccia e il periferico:** può essere un bus di periferica (come ad esempio il bus SCSI o il bus USB) o un collegamento punto a punto;
- **Dispositivo periferico:** dispositivo di input/output caratterizzato da determinate caratteristiche elettroniche ed elettromeccaniche;

1.2 Interfacce dei periferici

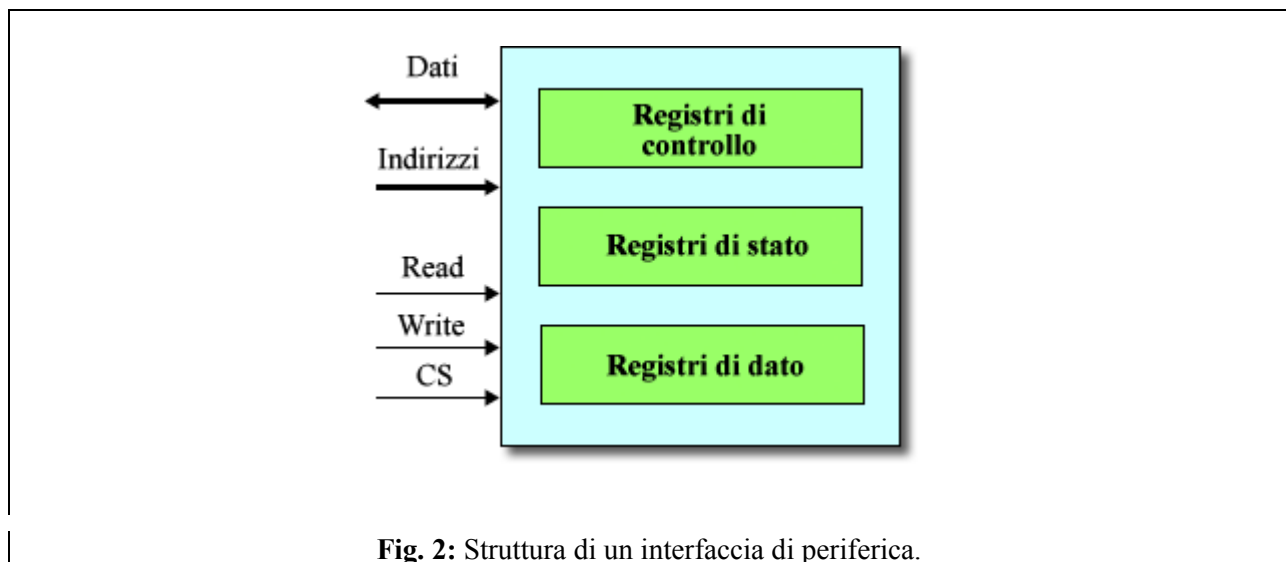
Un'interfaccia è normalmente costituita da un'insieme di registri (o **porte**) che permettono la gestione del periferico e il dialogo con il bus di sistema. Le porte delle interfacce possono essere raggruppate in tre grandi categorie:

- **Registri di comando:** registri che definiscono le modalità con cui l'interfaccia/periferico deve operare, tipicamente programmabili;
- **Registri di stato:** registri che mantengono informazioni riguardanti lo stato dell'interfaccia e/o del periferico. Specificano informazioni quali l'operatività del periferico, le impostazioni per esso configurate, ecc.

- **Registri di dato:** registri in cui vengono memorizzati i dati che dal periferico devono essere inviati al bus o viceversa. Tali registri svolgono il ruolo di buffer, permettendo di sincronizzare le diverse velocità di trasferimento delle parti coinvolte nella trasmissione.

La comunicazione tra il processore e i dispositivi periferici avviene mediante le porte delle interfacce. Queste sono accessibili tramite il bus di sistema allo stesso modo delle celle di memoria, ognuna di esse viene identificata per mezzo di un indirizzo e può essere scritta o letta mediante un ciclo di bus relativo a tale indirizzo. Quando un ciclo di bus si riferisce ad un indirizzo associato alla porta di una interfaccia, tale ciclo viene detto **ciclo di I/O**.

La struttura di un'interfaccia può essere schematizzata come descritto nella Fig. 2.



I segnali mediante i quali l'interfaccia comunica con il bus di sistema sono:

- **Segnali di indirizzo:** specificano la porta coinvolta nella comunicazione con il bus, sono costituiti da una porzione dei segnali del bus indirizzi;
- **Segnali di dato:** vengono usati per trasferire i dati da o verso la porta coinvolta nella comunicazione, costituiscono una porzione del bus dati di parallelismo compatibile con il parallelismo dei registri di interfaccia;
- **Segnale di chip-select:** svolge il ruolo di segnale di abilitazione per la specifica interfaccia, viene generato da una porzione dell'indirizzo posto sull'address bus;

- **Segnali di read/write:** identificano il tipo di transazione richiesta per la periferica, se di lettura o scrittura;

Essendo le interfacce dei periferici direttamente connesse al bus di sistema, le modalità di selezione mediante le quali tali interfacce vengono abilitate sono le stesse usate dal bus per l'abilitazione dei banchi di memoria:

- Un segnale di chip select abilita la specifica interfaccia, tale segnale è ricavato dalla decodifica di una porzione dell'indirizzo presente sull'address bus per mezzo di un *decoder binario-lineare*;
- La rimanente porzione di indirizzo identifica la porta dell'interfaccia coinvolta nella comunicazione;

Il numero di segnali di indirizzi utilizzati per identificare le porte interne di una determinata interfaccia, sarà determinato dal numero di porte indirizzabili per tale interfaccia.

1.3 Indirizzamento dei periferici

Per come sono strutturate le interfacce dei periferici, la comunicazione tra la CPU e tali dispositivi avviene in maniera del tutto analoga a quella tra la CPU e la memoria.

Esistono due strategie principali con cui è possibile assegnare gli indirizzi alle porte dei periferici:

- *Memory mapped I/O:* lo spazio di indirizzamento dei periferici è mappato su una porzione dello spazio di indirizzamento destinato alla memoria;
- *Isolated I/O:* lo spazio di indirizzamento dei periferici è separato da quello della memoria;

Nel definire gli indirizzi associati alle porte dei periferici occorre destinare un certo numero di bit di indirizzo per la selezione dell'interfaccia, dimensionato in base al numero di interfacce che si

vuole indirizzare, ed un certo numero di bit per la selezione del registro interno all'interfaccia, dimensionato in base al massimo numero di porte presente tra le interfacce indirizzate.

1.3.1 Isolated I/O

Nel caso in cui gli indirizzi di input/output vengano assegnati mediante la strategia dell'**isolated I/O**, si distinguono due spazi di indirizzamento disgiunti, uno per le porte delle interfacce e l'altro per le locazioni di memoria, ognuno dotato di 2^n indirizzi (con n parallelismo dell'address bus).

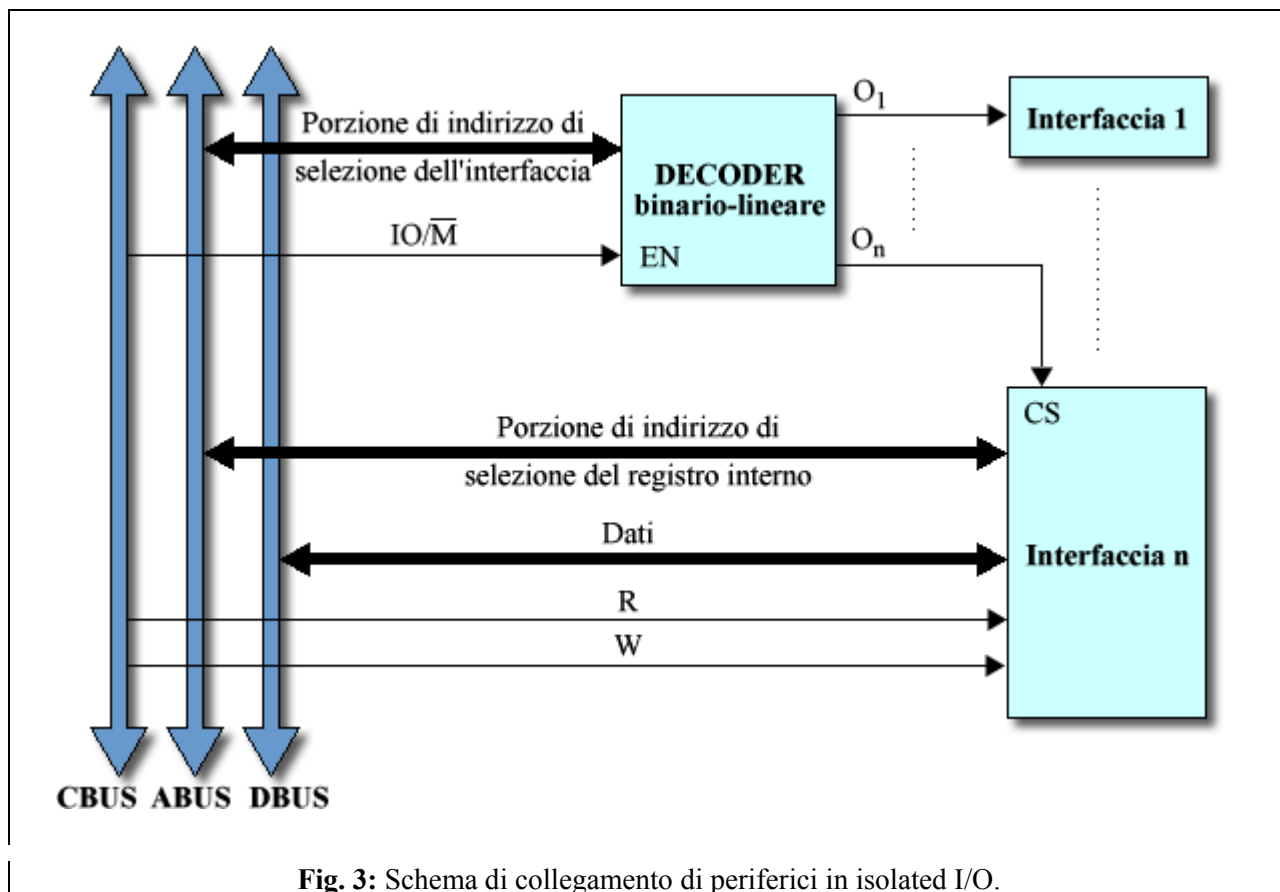
Appositi segnali di controllo (come il segnale IO/\overline{M} per i processori 80x86) vengono utilizzati per multiplexare sulle n linee del bus degli indirizzi i due spazi di indirizzamento da 2^n indirizzi.

Durante ogni transazione sul bus, i segnali di controllo indentificano il tipo di ciclo, che può essere:

- *Ciclo di I/O*: in tal caso l'indirizzo presente sull'address bus deve essere interpretato come l'indirizzo della porta dell'interfaccia interessata dalla transazione;
- *Ciclo di memoria*: in tal caso l'indirizzo presente sull'address bus deve essere interpretato come l'indirizzo della locazione di memoria interessata dalla transazione;

I segnali di controllo che identificano il tipo di ciclo svolgono il ruolo di abilitazione per il controllore della memoria o per la logica di decodifica delle interfacce, a seconda che la transazione sul bus sia relativa ad un indirizzo di memoria o di input/output.

Affinchè il processore possa discriminare il tipo di ciclo, devono essere previste nell'istruzione set delle apposite istruzioni per l'esecuzione delle operazioni di input/output. Solo differenziando le istruzioni di trasferimento dati relative alla memoria da quelle relative alle periferiche, infatti, il processore sarà in grado di distinguere il tipo di transazione da effettuare sul bus e pertanto generare opportunamente i segnali di controllo.



Utilizzando l'isolated I/O lo spazio di indirizzamento dei periferici non riduce lo spazio di indirizzamento dedicato alla memoria, ma non può essere acceduto attraverso le normali istruzioni di accesso in memoria. Per l'assembler x86 le istruzioni che permettono di effettuare l'accesso alle porte dei periferici sono le istruzioni IN e OUT.

La strategia dell'isolated I/O è quella comunemente utilizzata nei sistemi di elaborazione general purpose, e pertanto anche nelle architetture x86.

Esercizio (Esempio di utilizzo delle istruzioni di I/O)

Si supponga che un dispositivo 8255 sia configurato in modo che l'indirizzo di partenza delle porte sia 38H, la porta A sia in output, le porte B e C siano in input e che tutte le tre porte siano programmate in modo 0. Si scriva un programma assembler 8086 che legga i dati dalle porte B e C, ne esegua la differenza e scriva in output il risultato su A.

```
PORT_A      EQU      38H    ;La porta A è indirizzata da 38H
PORT_B      EQU      3AH    ;La porta B è indirizzata da 3AH
PORT_C      EQU      3CH    ;La porta C è indirizzata da 3AH

.MODEL small

.STACK

.DATA

.CODE

.STARTUP


IN  AL, PORT_B

MOV BL, AL

IN  AL, PORT_C

SUB AL, BL

OUT PORT_A, AL


.EXIT

END
```

1.3.2 Memory Mapped I/O

Nel caso in cui gli indirizzi di input/output vengano assegnati mediante la strategia del **memory mapped I/O**, si dispone di un unico spazio di indirizzamento da 2^n indirizzi (con n parallelismo del bus indirizzi) suddiviso tra: indirizzi di memoria ed indirizzi di input/output.

In tal caso la comunicazione con le periferiche sfrutta le normali istruzioni di accesso in memoria rendendo di fatto indistinguibili, dal punto di vista del processore, le due tipologie di cicli di bus.

La natura della transazione sul bus non viene pertanto pre-determinata dal processore, ma risulta legata al particolare indirizzo di volta in volta presente sull'address bus; essa viene determinata

direttamente dalla logica di decodifica delle periferiche o dal controllore di memoria sulla base di tale indirizzo.

Tale strategia permette una maggiore flessibilità di programmazione ma riduce allo stesso tempo il numero di locazioni di memoria indirizzabili. Gli indirizzi non assegnati a locazioni di memoria, e quindi assegnati alle porte delle interfacce, devono essere noti a priori per evitare che operazioni di accesso in memoria si traduca in realtà in operazioni di I/O.

La strategia di assegnazione degli indirizzi di memory mapped I/O non viene tipicamente utilizzata nei sistemi general purpose.

Esercizio (*Esempio di programmazione di un dispositivo di I/O*)

Supponendo di operare in un sistema in cui gli indirizzi delle periferiche sono assegnati in modalità memory-mapped, scrivere la sequenza di istruzioni assembler x86 necessarie per inizializzare il registro di controllo di un dispositivo 8255, con indirizzo 406H, in modo che la porta A sia una porta di output, le porte B e C porte di input e tutte le tre porte siano programmate per funzionare in modo 0. La parola di controllo richiesta è pari a 8BH.

```
;La porta di controllo è indirizzata da 406H
```

```
CONTROL_REG          EQU          406H
```

```
;La parola di controllo richiesta è 8BH
```

```
CONTROL_WORD          EQU          8BH
```

```
        .MODEL small
```

```
        .STACK
```

```
        .DATA
```

```
        .CODE
```

```
        .STARTUP
```

```
        XOR    AX, AX
```

```
PUSH    DS

MOV     DS,  AX

MOV     BX,  CONTROL_REG

MOV     AL,  CONTROL_WORD

MOV     [BX], AL

POP     DS

.EXIT

END
```

1.4 Gestione dei dispositivi di I/O

La funzione del sottosistema di input/output di un calcolatore è permettere il trasferimento di dati tra la memoria principale e i dispositivi periferici, o meglio le interfacce di tali dispositivi.

1.4.1 Parametri fondamentali di I/O

I parametri fondamentali che caratterizzano il trasferimento di dati da o verso dispositivi di input/output sono i seguenti:

- **Parallelismo** del dispositivo (n): la dimensione in bit dei dati trattati in un'unica operazione di trasferimento che coinvolge il periferico.
- **Frequenza di trasferimento** del dispositivo (f_{tr}): indica ogni quanti secondi il dispositivo è in grado di eseguire un trasferimento di dati, si esprime di hertz (Hz);
- **Transfer rate** o **velocità di trasferimento** del dispositivo ($T.R.$): la quantità di dati al secondo trasferiti dalla periferica. Viene espressa in byte al secondo (Bps) o bit al secondo (bps). È ottenuta come il prodotto tra il parallelismo (n) della periferica e la sua frequenza di trasferimento (f_{tr}):

$$T.R. = n \cdot f_{tr}$$

- **Tempo di accesso o tempo di latenza** (t_a) del dispositivo: l'intervallo di tempo che intercorre tra l'inizio di una operazione di input/output che coinvolge il dispositivo e il momento in cui il dato inizia ad essere trasferito;
- **Tempo di trasferimento** (t_{tr}) della periferica: tempo necessario per completare il trasferimento del blocco di dati richiesto per la specifica periferica. Viene ottenuto come il rapporto tra la quantità di dati da trasferire (m) e il transfer rate della periferica ($T.R.$):

$$t_{tr} = \frac{m}{T.R.}$$

- **Tempo di I/O** (t_{IO}): tempo necessario per completare l'operazione di input/output. È dato dalla somma del tempo di accesso (t_a) e del tempo di trasferimento (t_{tr}):

$$t_{IO} = t_a + t_{tr}$$

1.4.2 Tipi di dispositivi di I/O

Ai fini del trasferimento dati, si possono distinguere due tipologie di dispositivi periferici:

- **Dispositivi a carattere**: in cui i trasferimenti coinvolgono tipicamente un byte alla volta ed avvengono in istanti di tempo non prevedibili a priori. Non è possibile prevedere l'intervallo di tempo che intercorre tra il trasferimento di un dato e il trasferimento del dato successivo da parte di tali dispositivi. Esempi di tali dispositivi sono le tastiere e le linee asincrone;
- **Dispositivi a blocco**: in cui i trasferimenti coinvolgono tipicamente blocchi di dati di dimensione prefissata. Essendo la dimensione prefissata, è possibile prevedere a priori il tempo complessivo di trasferimento del blocco e l'intervallo di tempo che intercorre tra il trasferimento di un dato e il trasferimento del dato successivo. Un'esempio di tali dispositivi sono i dischi.

Per i dispositivi a carattere risulta significativo unicamente il tempo di accesso alla periferica, mentre per i dispositivi a blocco anche il tempo di trasferimento assume una certa rilevanza.

1.4.3 Metodi di gestione dei dispositivi di I/O

Dal momento che i dispositivi periferici sono caratterizzati da velocità di trasferimento molto varie, occorre adottare un meccanismo di sincronizzazione tra la CPU e i periferici, al fine di evitare situazioni che possono portare ad una perdita di dati o alla lettura/scrittura di dati non validi. Ciò si verifica ad esempio quando il processore scrive i dati sulle porte dei periferici troppo velocemente, sovrascrivendo il contenuto dei registri d'interfaccia prima che questo possa essere processato dal periferico, oppure quando il processore legge i dati da un registro d'interfaccia prima che questi siano presenti in modo stabile su di esso.

Per conciliare tale differenza di velocità e gestire le richieste di servizio da parte dei dispositivi di I/O, la CPU può utilizzare tre metodi:

- **Polling:** un ciclo software interroga a turno lo stato dei periferici, per verificare la presenza di eventuali richieste di servizio. Si scandiscono periodicamente le interfacce delle periferiche: se la periferica corrente richiede di essere servita, questa viene servita, altrimenti si passa ad interrogare la periferica successiva. La maggior parte del tempo è impiegata nell'esecuzione del ciclo di polling; ciò può portare ad elevati tempi di accesso dei dispositivi. Tale tecnica può essere usata solo se il transfer rate dei dispositivi interrogati è abbastanza costante. La gestione delle periferiche con tale modalità è realizzata completamente via software.
- **Interrupt:** quando una periferica necessita di essere servita, invia un segnale hardware al processore che, se può, interrompe il flusso di esecuzione corrente e passa ad eseguire una routine di servizio per la periferica. La gestione delle periferiche, viene in questo modo realizzata in parte via hardware e in parte via software.

- **Direct Memory Access (DMA)**: utilizzato per i trasferimenti di grandi moli di dato tra memoria centrale e periferici; il trasferimento viene effettuato da un dispositivo dedicato (*DMA Controller*) che diventa master del bus di sistema quando la CPU non ne necessita l'utilizzo. La modalità di trasferimento in DMA può essere realizzata unicamente per i dispositivi a blocco, come i dischi. La gestione delle periferiche con tale modalità è realizzata completamente in hardware.

2. Interfaccia parallela 8255

2.1 Modello 8255

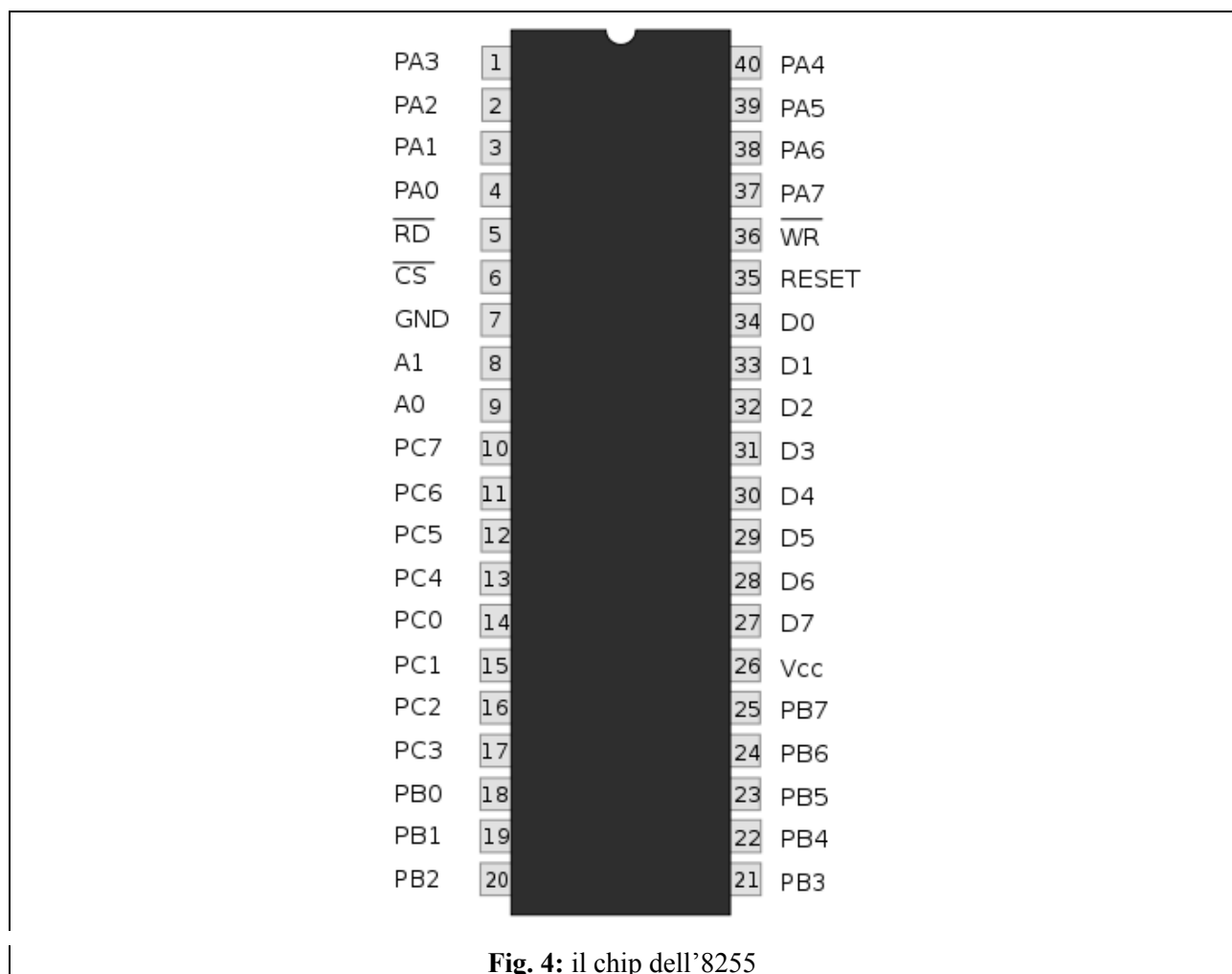
Il circuito integrato Intel 8255 implementa un'interfaccia parallela di I/O per sistemi delle famiglie 8085 e 8086.

Permette di eseguire input/output di bit, nibble (4 bit) e byte in modo completamente programmabile via software, in modo da rendere inutile l'uso di circuiti logici esterni.

Gestisce fino a 3 porte di Input/Output indipendenti da 1 byte ciascuna.

2.1.1 Chip 8255

È realizzato tramite un chip LSI di tipo DIP a 40 pin, oggi integrato nei chip set.



Il numero totale dei pin è 40, di cui 24 vengono usati per l'I/O.

Descriviamo brevemente le loro funzionalità.

- **PA₀₋₇**: bit della porta A
- **PB₀₋₇**: bit della porta B
- **PC₀₋₇**: bit della porta C
- **D₀₋₇**: si tratta di un buffer bidirezionale a 8 bit il cui compito è quello di interfacciare l'8255 con il bus di sistema. I dati vengono trasmessi o ricevuti sul buffer in corrispondenza dell'esecuzione di opportune istruzioni di I/O sulla cpu. Le word di controllo e le informazioni di stato vengono trasmesse tramite questo buffer.
- **CS** (*Chip select*): un input basso su questo pin abilita la comunicazione tra 8255 e CPU.
- **RD** (*Read control*): un valore di input basso su questo pin permette alla CPU di leggere dal bus dati, nel quale l'8255 avrà posto dei dati significativi.
- **WR** (*Write control*): un valore di input basso su questo pin permette alla CPU di scrivere sul bus dati, informazioni che verranno lette dall'8255.
- **A0 e A1** (*Address*): questi due pin permettono, insieme a RD e WR, di selezionare una delle porte interne o il registro di controllo. Solitamente vengono collegate a questi pin i bit meno significativi del bus degli indirizzi del sistema.
- **RESET**: un valore di input basso cancella il contenuto delle porte e del registro di controllo.
- **VCC e GND**: vengono usati rispettivamente per l'alimentazione (+5 volt) e il collegamento a massa.

2.1.2 Modello logico 8255

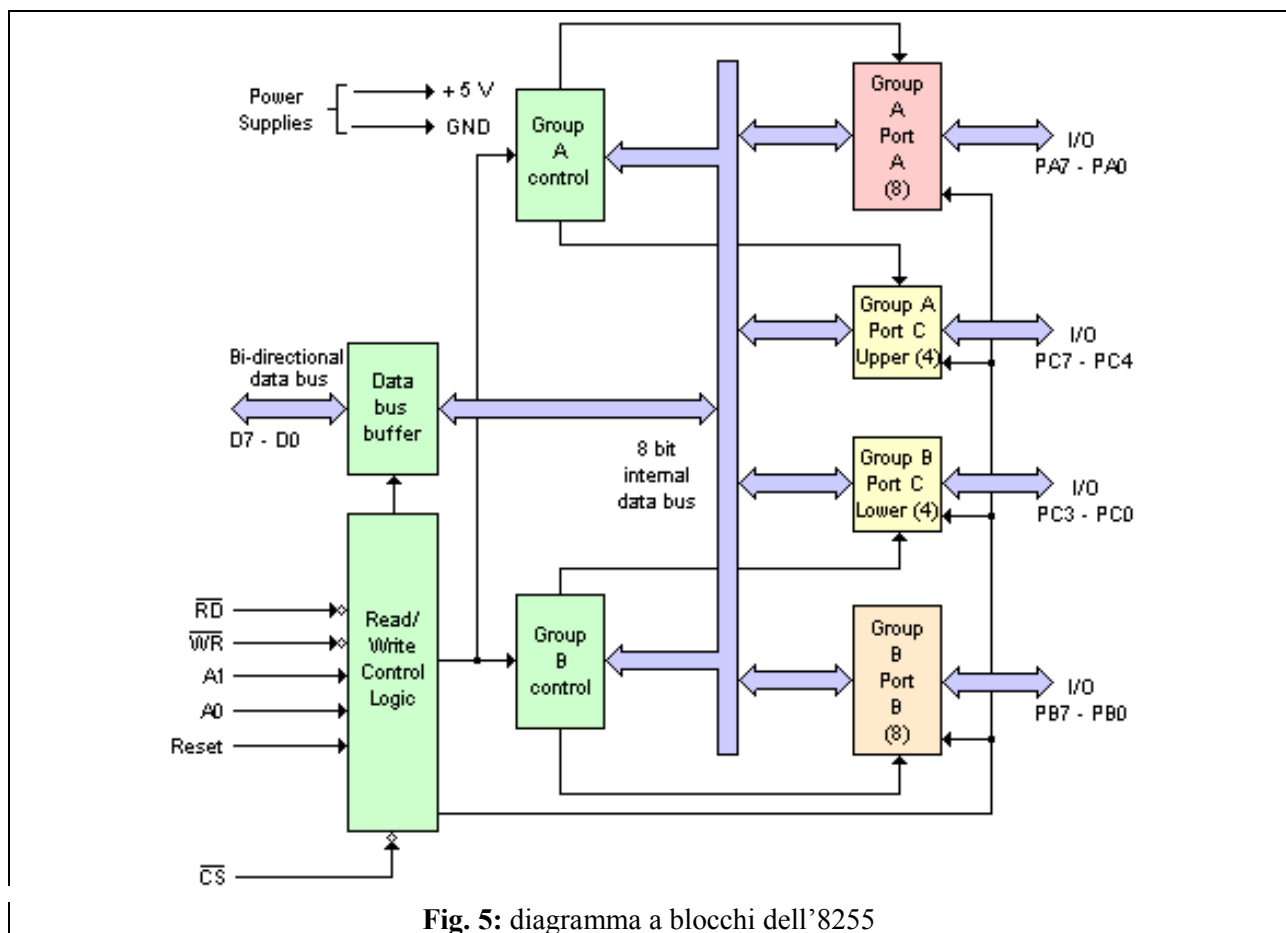
Dal punto di vista del programmatore l'8255 si presenta come un insieme di 4 registri da 8 bit, corrispondenti alle 3 porte (A, B, C) ed al Registro di Controllo.

I 24 pin di I/O sono suddivisi in 2 Gruppi di 12 pin:

- Gruppo A: Porta A e Porta C (parte alta)
- Gruppo B: Porta B e Porta C (parte bassa)

Accedendo ai registri associati alle 3 porte si esegue il trasferimento dati; accedendo al Registro di Controllo (write-only) si programmano le 3 porte tramite parole di controllo.

Questi 4 registri sono accessibili tramite i pin D_{0-7} , selezionando quello desiderato tramite i pin di indirizzo A_0 e A_1 .



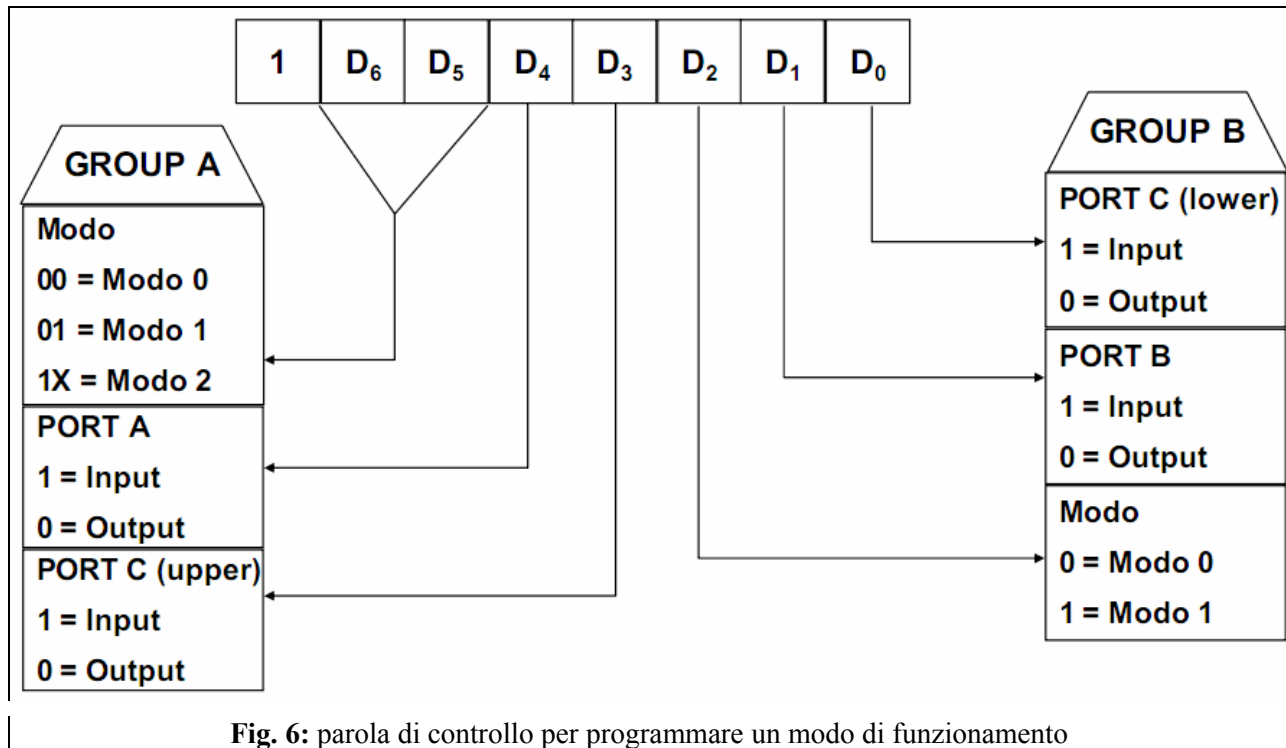
2.2 Programmazione dell'8255

La parola di controllo viene scritta dalla CPU nel registro di controllo dell'8255.

Può avere due funzioni:

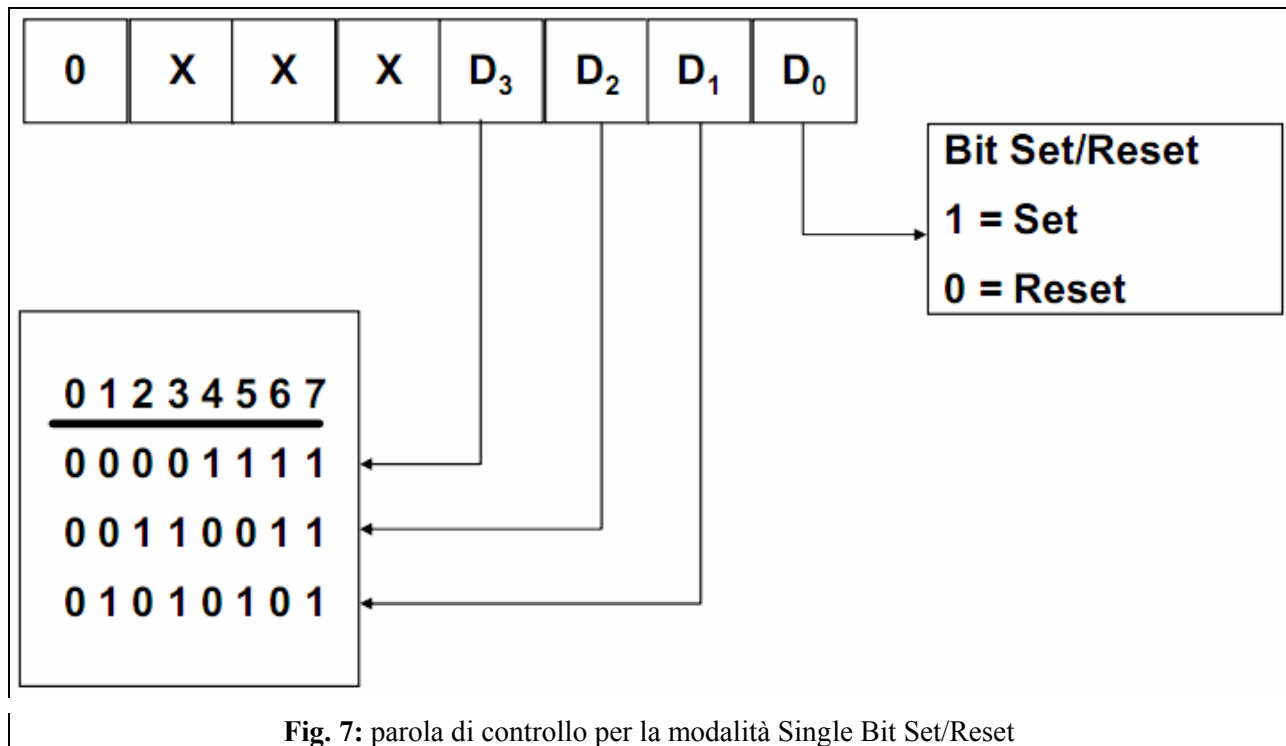
1. programmazione del modo di funzionamento delle porte dell'8255
2. scrittura di un valore logico in un singolo bit della porta C

2.2.1 Selezione del modo di funzionamento



Permette di programmare il dispositivo impostando il modo di funzionamento delle porte dell'8255.

2.2.2 Modalità Single Bit Set/Reset



Permette di effettuare la scrittura di un valore logico in un singolo bit della porta C (single bit set/reset) impostando i bit D0-3 alla posizione del bit da modificare e D0 al valore da attribuire. Il valore dei bit segnati con X in figura non è significativo.

2.3 Modi di funzionamento

Le porte dell'8255 possono essere programmate in 3 modi operativi:

- Modo 0: Basic Input/Output
- Modo 1: Strobed Input/Output
- Modo 2: Bidirectional Bus.

Al reset l'8255 è inizializzato con tutte le porte programmate in modo 0 in Input.

Nell'8255 i modi di funzionamento servono per descrivere come un determinato gruppo di porte funziona, per tale motivo è fondamentale la suddivisione delle porte in due gruppi.

Si ha la possibilità di usare tre diversi modi operativi:

1. **modo 0**: basic input/output,
2. **modo 1**: strobed input/output,
3. **modo 2**: bidirectional bus.

Al reset l'8255 è inizializzato con tutte le porte programmate in modo 0 in input.

Nel modo 0, ciascun gruppo di 12 pin può essere programmato considerando insiemi di 4 bit come input o output: non vi è quindi alcuna forma di handshaking. Nel secondo modo, il modo 1, ciascun gruppo può essere programmato considerando che 8 dei pin del gruppo sono usati per l'input o l'output. Dei rimanenti 4 pin, 3 vengono usati per l'handshaking, mentre 1 viene usato per il segnale di controllo degli interrupt. Infine il modo 2 implementa un bus bidirezionale, in cui 8 delle linee sono usate per i dati, mentre altre 5 linee (una viene prestata dall'altro gruppo) sono usate per l'handshaking.

Nel caso in cui l'8255 venga fatto funzionare nel modo 1 o nel modo 2, il dispositivo può ricevere un segnale di interrupt request tramite il pin opportuno della porta C.

E' possibile effettuare un mascheramento di tali segnali ponendo a 0 il flip flop INTE (interrupt enable), tramite la funzionalità di set/reset del bit della porta C che vedremo nel seguito.

Viene effettuato il reset dei flip flop di mascheramento ogni volta che si seleziona un modo o che il dispositivo viene resettato.

In modo 1 e 2 alcuni segnali di controllo, provenienti dalla porta C, possono essere utilizzati per inviare una richiesta di interrupt alla CPU.

Tali segnali possono essere disabilitati o abilitati settando o resettando il flip-flop interno di interrupt enable (INTE) attraverso l'operazione di bit set/reset della Porta C.

INTE abilita l'interrupt quando l'opportuno bit della Porta C è forzato ad 1.

2.3.1 Modo 0

Questa configurazione operativa permette di instaurare su tutte le 3 porte le normali funzioni di lettura o scrittura di un dato. Non è richiesto alcun segnale di handshaking in quanto i dati vengono semplicemente letti o scritti nelle porte.

Caratteristiche:

1. due Porte ad 8 bit e due Porte a 4 bit;
2. qualsiasi Porta può essere definito come ingresso o come uscita;
3. le uscite sono memorizzate;
4. gli ingressi sono memorizzati;
5. sono possibili 16 diverse combinazioni di I/O

Esempio di funzionamento in Modo 0:

Si consideri un sistema a microprocessore basato su 8086 che deve comunicare tramite un'interfaccia di comunicazione parallela 8255, mappata all'indirizzo 50h, acquisendo un valore dalla porta PA e ripetendolo sulla porta PC.

La Control Word relativa è illustrata nella Fig 8.

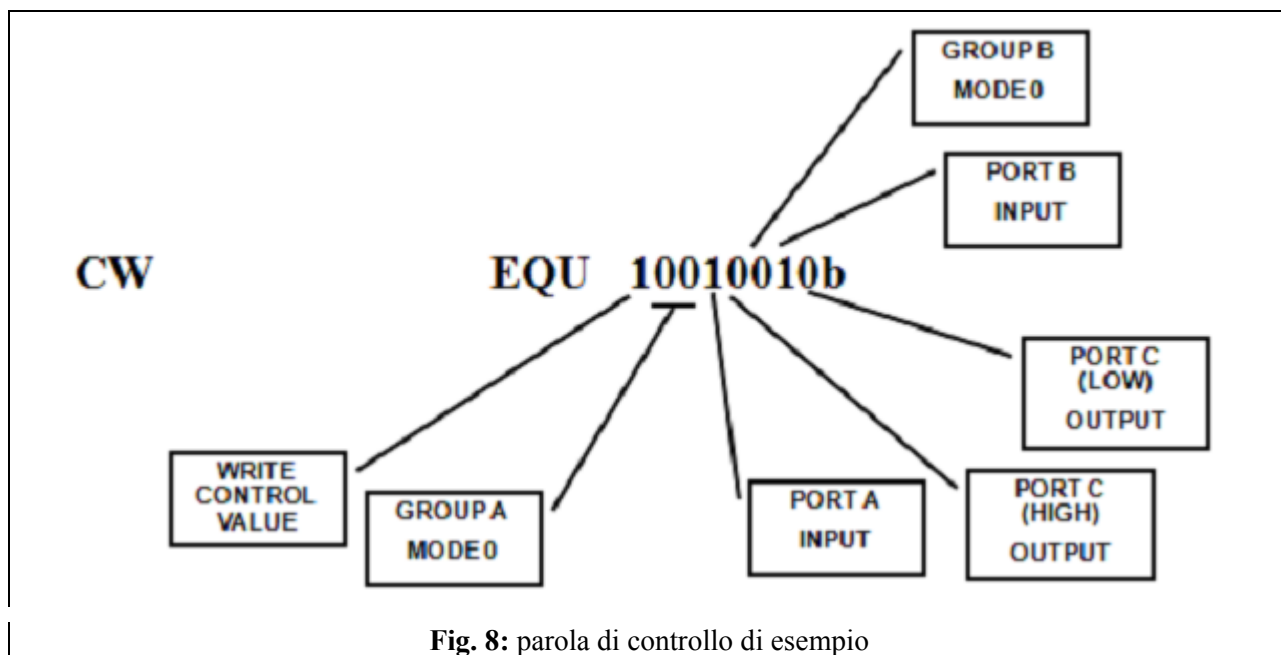
Il codice Assembler è il seguente:

```

; Valori di inizializzazione
PORTA EQU 50H           ; Indirizzi porte
PORTB EQU 51H
PORTC EQU 52H
CONTROL EQU 53H         ; Indirizzo registro di controllo
CW EQU 10010010b        ; Parola di controllo

; Codice per l'inizializzazione
MOV DX, CONTROL ; Scrittura del registro di controllo
MOV AL, CW
OUT DX, AL
MOV DX, PORTA
IN AL, DX        ; Acquisizione dato dalla porta A
MOV CX, DELAY
ciclo: LOOP ciclo ; Ciclo di busy waiting prima di effettuare
```

MOV DX, PORTC ; l'operazione di scrittura
OUT DX, AL ; Scrittura dato sulla porta C



2.3.2 Modo 1

Questo modo permette di trasferire dati dalla porta A o B mediante il controllo di segnali di handshaking, associati alle linee della porta C. Utilizza un protocollo di tipo handshake unidirezionale (R/W), nella quale sono definiti due segnali: uno di DATO PRONTO inviato dal

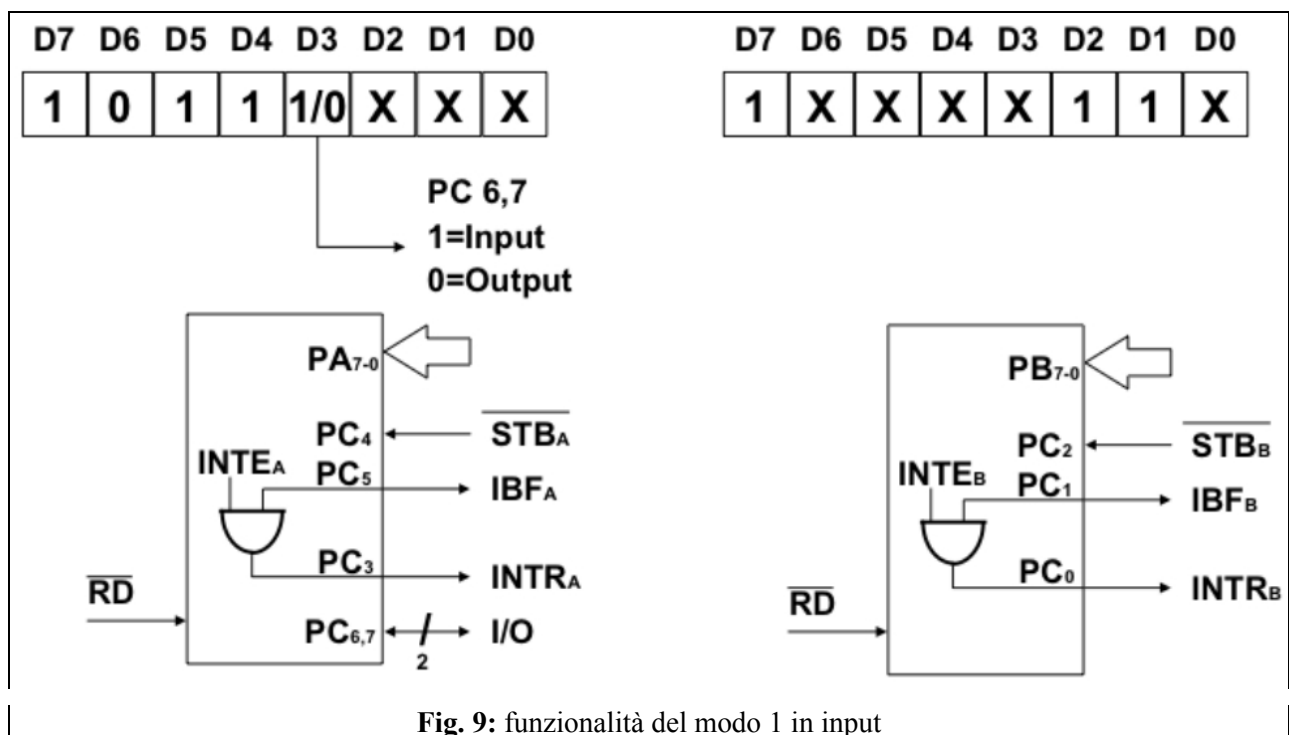
trasmettitore quando il dato da trasmettere è pronto sulla porta ed uno di DATO RICEVUTO inviato dal ricevitore quando il dato trasmesso è stato immagazzinato. Nel modo 1 la porta A e la porta B possono essere individualmente come porte di input o di output, per cui possono essere effettuate applicazioni di tipo diverso.

Caratteristiche:

- due gruppi di trasferimento (Gruppo A e Gruppo B);
- ciascun gruppo contiene una porta di dati ad 8 bit ed una porta di controllo dei dati da 4 bit;

- le due porte di dati ad 8 bit possono essere sia d'ingresso che d'uscita, sia le uscite che gli ingressi sono memorizzati su latch;
- le due porte a 4 bit vengono usati per emettere i segnali di controllo e fornire lo stato del dispositivo alle periferiche collegate alle due porte ad 8 bit;
- le linee della porta C non utilizzate come controlli possono essere programmate in ingresso o in uscita.

Modo 1 - Input

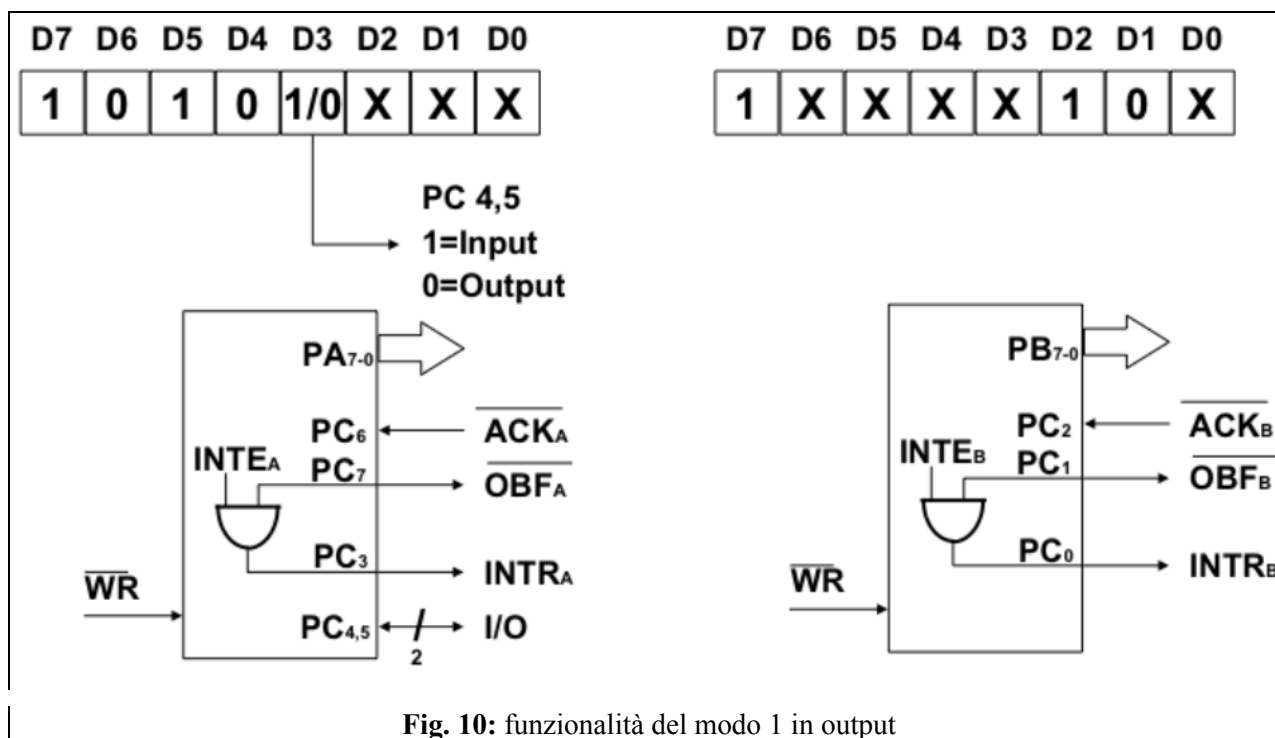


I segnali di controllo sono:

- **STB** (*Strobe Input*): quando assume un valore basso permette il caricamento del dato nell'input latch.
- **IBF** (*Input Buffer Full*): quando assume un valore alto indica che il dato è stato caricato correttamente nel latch di input, viene usato come acknowledgment. IBF è settato quando STB va basso, ed è resettato dal fronte di salita di RD.

- **INTR** (*Interrupt Request*): se assume un valore alto, viene inviata una richiesta di interrupt alla cpu. INTR è settato quando STB va alto, IBF e INTE sono entrambi alti (infatti nella figura precedente sono entrambi collegati ad una porta and); è resettato sul fronte di discesa di RD.
- **INTE A** (*Interrupt Enable per il gruppo A*): viene controllato dal bit set/reset di PC4, se basso disattiva gli interrupt per il gruppo A.
- **INTE B** (*Interrupt Enable per il gruppo B*): viene controllato dal bit set/reset di PC2, se basso disattiva gli interrupt per il gruppo B.

Modo 1 – Output



I segnali di controllo sono:

- **OBF** (*Output Buffer Full*): quando assume un valore basso indica che la CPU ha scritto il dato sulla porta. Viene settato sul fronte di salita di WR e resettato quando ACK diventa basso.

- **ACK** (*Acknowledge Input*): quando assume un valore basso informa l'8255 che il dato è stato ricevuto dalla periferica.
- **INTR** (*Interrupt Request*): un valore alto può essere usato come richiesta di interrupt per la CPU.
- **INTR**: è resettato sul fronte di discesa di WR, ed è settato quando ACK è alto, e sia OBF che INTE sono alti.
- **INTE A, INTE B**: sono stati descritti in precedenza.

Esempio di funzionamento in Modo 1:

Un sistema a microprocessore basato su 8086 deve ricevere ed elaborare valori provenienti da un dispositivo esterno, tramite un'interfaccia parallela 8255 all'indirizzo 50H.

1. Il sistema riceve dal dispositivo esterno un valore in input ed un segnale di STROBE a segnalare che un dato è stato inviato.
2. Segnala al dispositivo esterno la disponibilità a ricevere un nuovo dato tramite un segnale di ACK.
3. Ogni 200 valori ricevuti, il processore 8086 ne esegue la media e la mette a disposizione dell'esterno tramite lo stesso 8255.

Uno schema strutturale del sistema descritto è riportato in Fig.11.

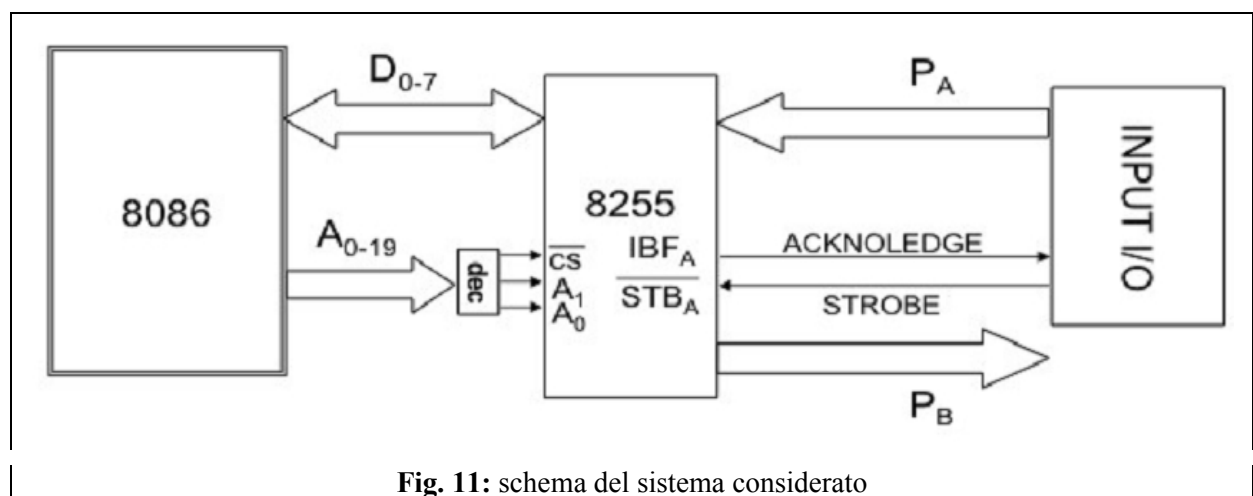


Fig. 11: schema del sistema considerato

Una soluzione al problema in linguaggio Assembly basata sul polling, è riportata di seguito:

```
                ; Definizione delle costanti e della parola di controllo
PORTA    EQU 50H                ; Indirizzi porte
PORTB    EQU 51H
PORTC    EQU 52H
CONTROL  EQU 53H                ; Indirizzo registro di controllo
CW       EQU 10110000b          ; Parola di controllo

                ; Inizializzazione dispositivo
MOV AL, CW
MOV DX, 53H
OUT DX, AL

                ; Programma principale
MOV BX, 0
MOV DI, 0
MOV CX, 200
MOV DX, PORTC
non_pronto: IN AL, DX
            TEST AL, 00010000b    ; Controlla il pin IBF (Polling)
            JZ non_pronto
            INC DI
            MOV DX, PORTA
            IN AL, DX
            CBW ADD BX, AX          ; Effettua la somma dei 200 valori
            CMP DI, CX
            JNE non_pronto
            XCHG BX, AX
            DIV CL                  ; Calcola la media, viene posta in AL
            MOV DX, PORTB
            OUT DX, AL
```

Una soluzione alternativa, facente uso di un meccanismo di interrupt realizzato mediante un dispositivo 8259, è riportata in Fig. 12.

Il codice Assembler è il seguente:

```
                ; Inizializzazione dispositivo
MOV AL, CW
MOV DX, 53H
OUT DX, AL
MOV AL, SPC4
OUT DX, AL
```

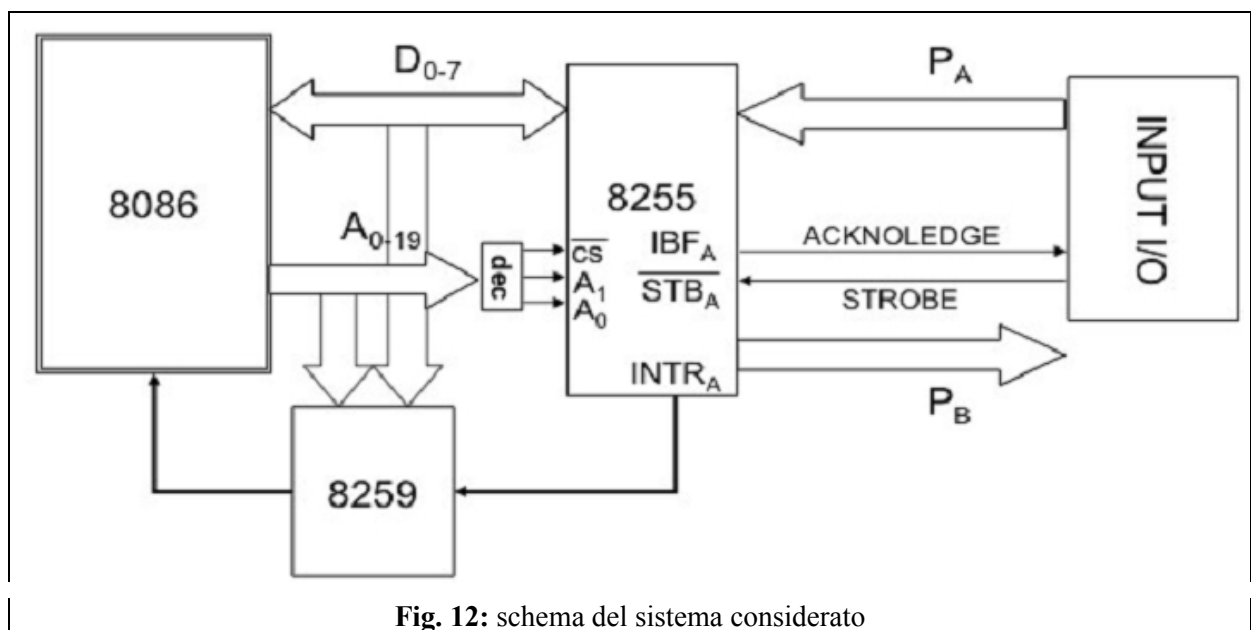
```

; Programma principale
MOV DI, 0
MOV CX, 200
non_pronto: HLT      ; Pone il sistema in uno stato di
                    ; attesa da cui si esce esclusivamente
                    ; con un evento di interrupt

CMP DI, CX
JNE non_pronto
XCHG BX, AX
DIV CL      ; Calcola la media e la pone in AL
MOV DX, PORTB
OUT DX, AL

; Procedura di gestione dell' INTERRUPT
leggi PROC ; Eseguita 200 volte in corrispondenza della ricezione di ogni interrupt
MOV DX, PORTA
IN AL, DX
CBW
ADD BX, AX ; Calcola la somma dei valori
INC DI
IRET
leggi ENDP

```



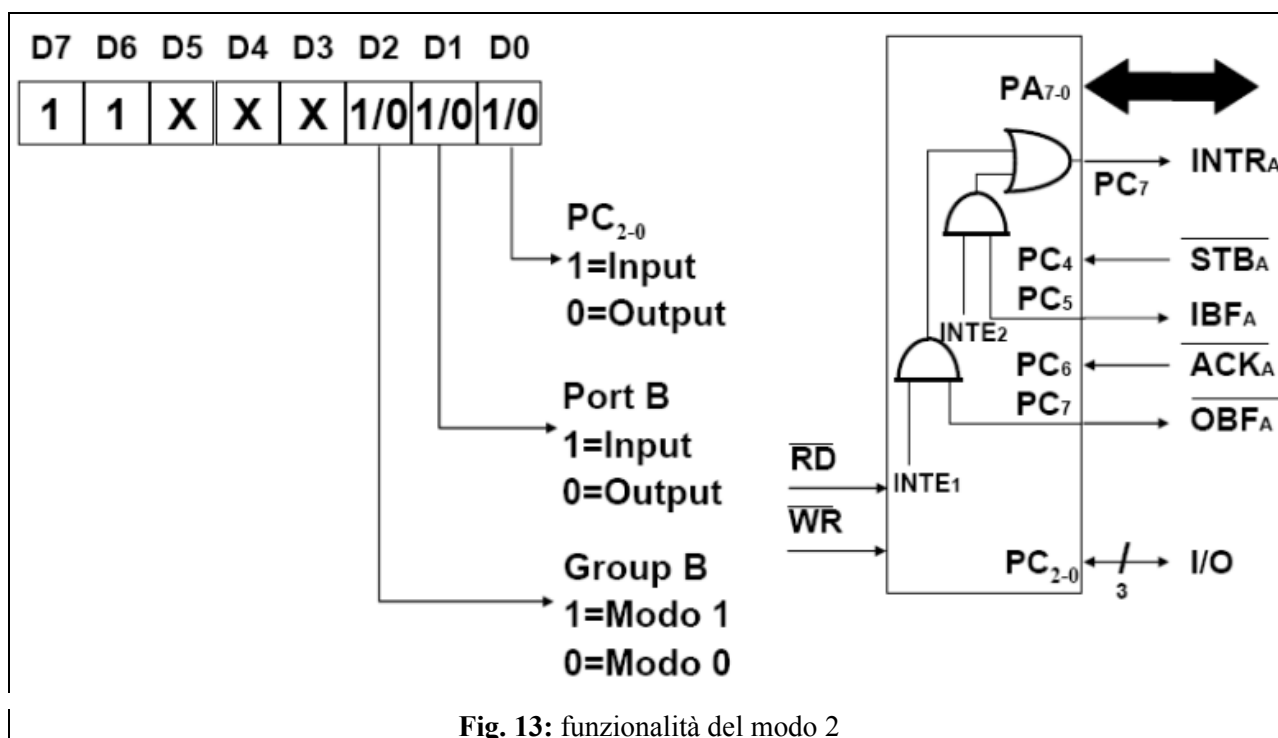
2.3.3 Modo 2

Questa configurazione operativa permette all'8255A di interfacciarsi direttamente ad un dispositivo avente un bus bidirezionale ad 8 bit. I segnali di controllo (con modalità di handshake

bidirezionale) permettono di regolare il flusso dei dati in maniera analoga a quella vista per il Modo 1, è possibile inoltre sia generare un segnale di interrupt che di abilitarne e disabilitarne il funzionamento.

Caratteristiche:

- si può usare solo il gruppo A;
- un bus bidirezionale ad 8 bit disponibile sul port A e 5 linee di controllo accessibili sul port C;
- sia gli ingressi che le uscite sono memorizzate su latch;
- il port di controllo a 5 bit (port C) viene usato per controllare e leggere lo stato del bus bidirezionale ad 8 bit (port A);
- il port B invece potrà essere, indipendentemente, utilizzato in modo zero o uno.



I segnali di controllo sono:

- **INTR**: assume un valore alto per determinare una richiesta di interrupt per la CPU.

- **OBF**: assume un valore basso per indicare che la CPU ha scritto un dato sulla porta A.
- **ACK**: assume un valore basso per abilitare l'invio di un nuovo dato.
- **STB**: se assume un valore basso si carica il dato nell'input latch.
- **IBF**: se assume un valore alto indica che il dato è stato caricato sull'input latch.
- **INTE 1** (*InterruptEnable*): controllato dal bit set/reset di PC 6.
- **INTE 2** (*InterruptEnable*): controllato dal bit set/reset di PC 4.

3. Interfaccia seriale 8250

3.1 Le comunicazioni seriali

Lo scambio di informazioni tra due dispositivi o periferici deve essere effettuata attraverso comunicazioni digitali. Possono essere identificate due topologie di canale di comunicazione:

- **comunicazione seriale:** le informazioni sono comunicate una di seguito all'altra.
- **comunicazione parallela:** è possibile trasmettere più segnali contemporaneamente (a scapito di architetture e protocolli più complessi).

Le unità di misura usate per la velocità di trasmissione su una linea seriale sono:

- **bps:** numero di bit trasmessi per secondo;
- **baud:** numero di simboli trasmessi per secondo.

Ad esempio, un segnale a 300 baud che codifica 4 bit di informazione per ogni simbolo ha una velocità di trasmissione pari a 1200 bps.

Se il simbolo è composto da un solo bit, allora baud e bps sono equivalenti.

3.1.1 Modi operativi

I dispositivi possono essere messi in comunicazione in modalità diverse a seconda delle caratteristiche intrinseche del canale:

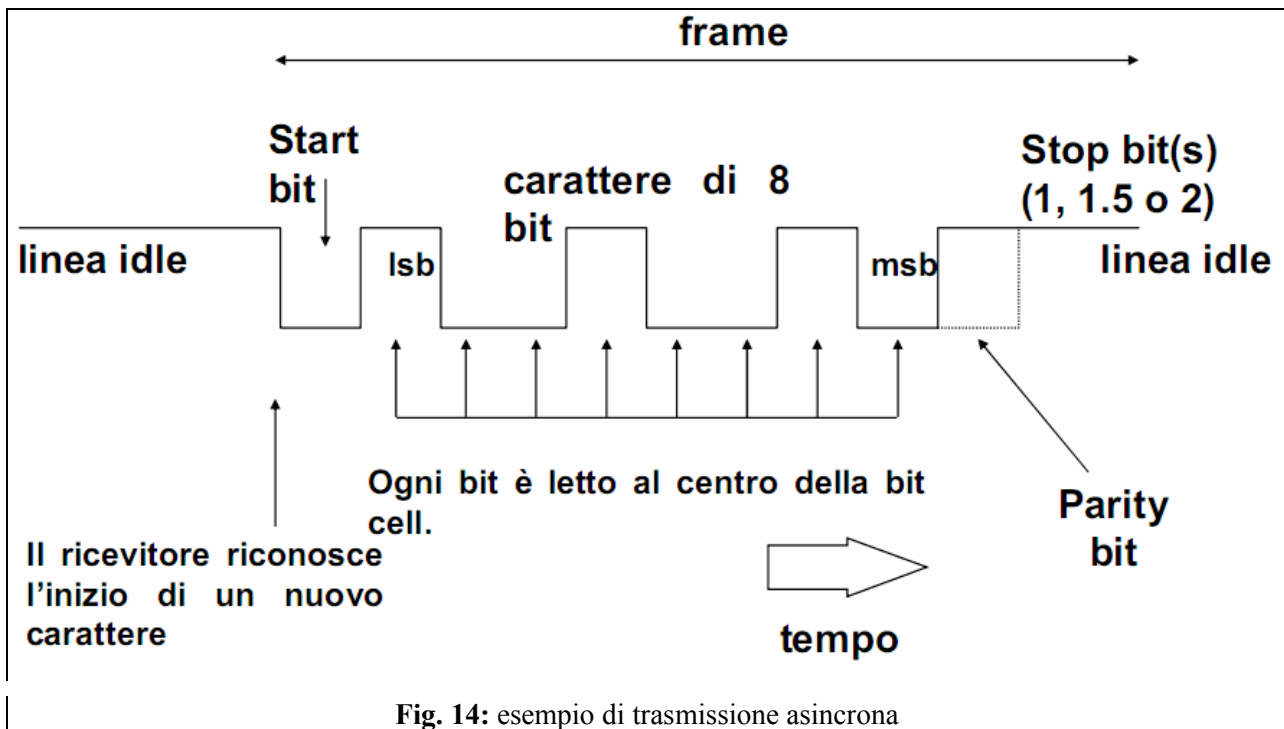
- **Simplex:** la comunicazione è unidirezionale;
- **Half Duplex:** la comunicazione è bidirezionale, ma avviene su un'unica linea; Il canale quindi è mono direzionale ma il verso può essere invertito.
- **Full Duplex:** la comunicazione avviene attraverso due linee, una per ciascuna direzione.

3.1.2 Trasmissione asincrona

I dati sono trasmessi serialmente in caratteri di n bit. All'interno di ciascun dispositivo i dati sono memorizzati in forma parallela.

I dispositivi di controllo della trasmissione devono svolgere le seguenti funzioni:

- conversione parallelo-seriale di ciascun elemento per prepararlo alla trasmissione
- conversione seriale-parallela di ciascun elemento ricevuto
- sincronizzazione in fase ricevente
- controllo di eventuali errori di comunicazione.



La frequenza di trasmissione dati è decisa all'atto della programmazione, comunicando al dispositivo ricevente il **fattore di scalamento K** tra il suo clock ed il clock di ricezione.

Dall'istante della prima transizione 1→0 il ricevitore lascia passare $K/2$ colpi di clock, e poi campiona il segnale di dato (Fig. 14). Se questo è basso, il bit viene interpretato come bit di start, altrimenti la transizione viene considerata fasulla.

A partire dal bit di start, il ricevitore campiona il segnale di dato ogni K colpi di clock; una volta letti i bit di dato, il ricevitore campiona i bit di stop, in modo da verificare la corretta terminazione del carattere.

Il frame di trasmissione è quindi composto da:

- start-bit
- payload di 8 bit
- stop-bit

3.1.3 Bit per carattere

La dimensione di ciascun carattere (in bit) fa parte dei parametri di connessione; solitamente si utilizzano caratteri di 5, 6, 7 o 8 bit. È possibile prevedere un bit aggiuntivo per ciascun carattere, che contenga un'informazione di parità, che può venire utilizzata per rilevare eventuali errori di trasmissione.

Per trasmettere ciascun carattere sono necessari 10 bit (1 bit di start, 8 bit di dato e 1 bit di stop), più eventualmente un bit di parità. Supponendo di trasmettere ad una frequenza di 1200 bit al secondo con controllo di parità, il tasso di trasmissione è quindi pari a 1200/11 byte al secondo.

3.1.4 UART

I circuiti di interfaccia per gestire la comunicazione asincrona sono noti come *Universal Asynchronous Receiver and Transmitter* (UART). L'attributo di "universale" sta a significare che il dispositivo è programmabile e dunque l'utente può specificare le caratteristiche operative richieste inviando una opportuna parola di controllo.

Il dispositivo 16550 UART è un circuito integrato che permette di implementare una interfaccia per la comunicazione seriale; questo dispositivo è considerato lo standard sulle architetture IBM e sui PC, ed è spesso integrato sulla motherboard e collegato a dispositivi quali modem o stampanti.

Alcuni modelli di UART:

- 8250: baud rate massimo 9600
- 16450: baud rate massimo 115200
- 16550: buffer FIFO da 16 byte per memorizzare dati ricevuti o da trasmettere

- 16650: buffer FIFO da 32 byte
- 16750: buffer FIFO da 64 byte.

Il software scritto per diverse versioni è compatibile.

3.2 Modello 8250

L'8250 è l'interfaccia standard di comunicazione seriale di tipo UART della National Semiconductor; permette di generare internamente il bit-rate, nell'intervallo di valori previsti dallo standard RS-232.

Caratteristiche principali:

- comunicazione Asincrona;
- fattore di scalamento K pari a 16;
- riconoscimento di errori di trasmissione.

3.2.1 Piedinatura

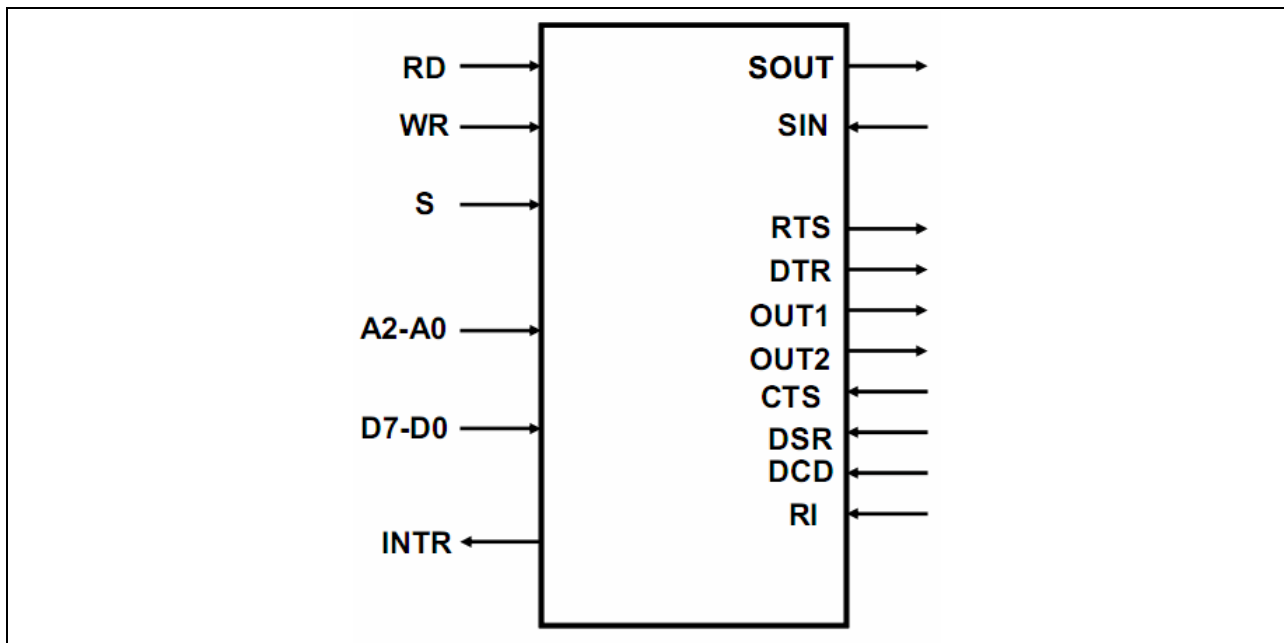


Fig. 15: schema dell'8250

- **RD**: lettura
- **WR**: scrittura
- **S**: selezione
- **A₀₋₂**: bit di indirizzo
- **D₀₋₇**: bus dati
- **INTR**: interrupt request
- **SIN**: serial input
- **SOUT**: serial output
- **OUT1, OUT2**: user's outputs
- **RTS**: request to send
- **CTS**: clear to send
- **DCD**: data carrier detect
- **DTR**: data terminal ready
- **DSR**: data set ready
- **RI**: ring indicator

I pin RTS, DTR, OUT1-out2, CTS, DSR, DCD, RI sono necessari per rispettare lo standard RS-232.

3.2.2 Registri interni

L'8250 ha al suo interno 10 registri interni: alcuni contengono dati ricevuti dall'esterno, altri controllano la trasmissione e gli interrupt.

	A2	A1	A0	DLAB	Accesso
RBR	0	0	0	0	lettura
THR	0	0	0	0	scrittura
DLR-LSB	0	0	0	1	lettura e scrittura
DLR-MSB	0	0	1	1	lettura e scrittura
IER	0	0	1	0	lettura e scrittura
IIR	0	1	0	-	lettura
LCR	0	1	1	-	lettura e scrittura
MCR	1	0	0	-	lettura e scrittura
LSR	1	0	1	-	lettura
MSR	1	1	0	-	lettura

Fig. 16: accesso ai registri

Poiché tre bit non sono sufficienti a indirizzare 10 registri si utilizza il DLAB (*Divisor Latch Access Bot*), un bit di indirizzo aggiuntivo contenuto nel bit 7 del registro LCR.

Il suo valore è normalmente 0, tranne per la programmazione del Divisor Latch Register.

I registri sono:

- **Receiver Buffer Register (RBR):** riceve i bit in forma seriale e li invia in forma parallela. Ottiene i dati dall'interfaccia seriale tramite il segnale SIN. Il primo bit ricevuto è messo nel bit meno significativo del registro.
- **Transmitter Holding Register (THR):** funziona in modo opposto al RBR, ricevendo i bit in forma parallela e inviandoli in forma seriale sul pin SOUT. Il bit che sarà trasmesso per primo è contenuto nel bit meno significativo del registro.
- **Divisor Latch Registers (DLRs):** i due registri DLR permettono di determinare il bit-rate (unico per la trasmissione e per la ricezione).

- **Modem Status Register (MSR):** contiene informazioni sulle 4 linee in input CTS, DSR, DCD, RI.
- **Modem Control Register (MCR):** usato per eseguire operazioni di handshaking con la periferica
- **Interrupt Enable Register (IER):** permette di abilitare le richieste di interruzioni selettivamente rispetto ad alcune condizioni di errore.
- **Interrupt Identification Register (IIR):** è accessibile esclusivamente in lettura e permette di identificare lo stato delle richieste di interruzioni.
- **Line Status Register (LSR):** contiene le informazioni relative allo stato dell'interfaccia; è accessibile esclusivamente in lettura.
- **Line Control Register (LCR):** definisce il formato del frame (sia in trasmissione che in ricezione).

3.3 Programmazione dell'8250

3.3.1 Line Status Register

X	X	EO	B	FE	PE	OE	FI
---	---	----	---	----	----	----	----

FI (Full Input) = 1: un nuovo carattere è stato ricevuto dall'interfaccia ed è disponibile in RBR.

Posto a 0 quando il processore legge RBR.

OE (Overrun Error) = 1: errore di Overrun.

PE (Parity Error) = 1: errore di Parità.

FE (Frame Error) = 1: errore di Frame.

B (Break) = 1: ricezione di un break, stato in cui la linea di dato è tenuta a 0 per un tempo superiore al tempo necessario per trasmettere una parola (incluso il tempo di trasmissione del bit di start, stop e parità).

EO (Empty Output) = 1: il dato contenuto nel registro THR è stato trasmesso. Posto a 0 quando il processore scrive in THR.

I flag di errore nel registro LSR sono:

- *parity error*: errore di trasmissione rilevato attraverso il bit di parità;
- *framing error*: mancata ricezione di un bit di stop;
- *overrun error*: il ricevitore ha ricevuto un nuovo dato senza che la CPU abbia letto il dato contenuto nel buffer ricevente.

I flag di errore sono posti a 0 dopo che il processore legge il registro LSR.

La segnalazione di tali errori non blocca il funzionamento dell'UART, ma è compito della CPU operare le necessarie azione correttive.

3.3.2 Line Control Register

DLAB	BE	SP	PS	P	STOP	L2	L1
-------------	-----------	-----------	-----------	----------	-------------	-----------	-----------

L2, L1 (Number of Data Bits) definiscono il numero di bit di dato (bit per carattere)

L2	L1	Numero di bit
0	0	5
0	1	6
1	0	7
1	1	8

STOP (Number of Stop Bits) Numero di bit di stop

- LE=0: 1 bit di stop
- LE=1: 1.5 bit di stop (se il carattere è di 5 bit)
2 bit di stop (se il carattere è di 6, 7 o 8 bit)

P (Parity)= 1: presenza di un bit di parità

Se PE = 1 i bit SP (*Sticky Parity*) e PS (*Parity Select*) assumono il seguente significato:

SP	PS	significato
0	0	bit di parità a 1 se nel dato c'è un numero dispari di bit a 1
0	1	bit di parità a 1 se nel dato c'è un numero pari di bit a 1
1	0	bit di parità = 1
1	1	bit di parità = 0

BE (Break Enable) = 1: trasmissione di un segnale di break: il piedino SOUT si porta a 0 e rimane in questo stato fino a che BE non viene posto a 0.

DLAB (Divisor Latch Access Bit). DLAB = 1 accesso ai registri DLR.

L'interfaccia ricava il bit-rate dividendo la sua frequenza di pilotaggio per 16 e per il numero contenuto nei due registri DLR (detto costante di tempo).

Il Bit-rate si calcola quindi come: *frequenza di pilotaggio / (16 * costante di tempo)*.

In base al valore del DLR si sceglie il fattore di divisione per la comunicazione.

3.3.3 Interrupt Enable Register

0	0	0	0	SINP	ERBK	TBE	RDR
---	---	---	---	------	------	-----	-----

RDR (Received Data Ready) = 1: abilitazione ad effettuare una richiesta di interruzione quando un byte è pronto in RBR

TBE (Transmitter Buffer Empty) = 1: abilitazione ad effettuare una richiesta di interruzione quando THR è vuoto

ERBK (*Error & Break*) = 1: abilitazione ad effettuare una richiesta di interruzione quando viene rilevato un errore o un segnale di break

SINP (*Serial Input*) = 1: abilitazione ad effettuare una richiesta di interruzione quando uno dei segnali di input del protocollo RS-232 (CTS, DSR, DCD, RI) cambia di stato

3.3.4 Interrupt Identification Register

0	0	0	0	0	ID1	ID0	PND
----------	----------	----------	----------	----------	------------	------------	------------

PND (*Pending Bit*) = 1: nessun interrupt è pendente;

= 0: interrupt pendente

ID1, ID0 (*Identify Bits*): bit di identificazione delle richieste di interruzione

ID1	ID0	significato
0	0	cambio di un segnale RS-232 (priorità 3: min)
0	1	THR vuoto (priorità 2)
1	0	RBR pieno (priorità 1)
1	1	errore in ricezione o break (priorità 0: max)

Le richieste di interruzione a priorità più basse sono bloccate se è pendente una richiesta a priorità più alta.

Esempio di inizializzazione dell'interfaccia:

Si supponga che i parametri della configurazione seriale siano i seguenti:

- bit-rate: 9600 bit/s
- bit per carattere: 8
- parità: disabilitata

- bit di stop: 1
- richieste di interruzione: tutte disabilitate

La procedura assembler è la seguente:

```
                                ; Definizione delle costanti
dlr_lsb EQU 03F8h
dlr_lsb EQU 03F9h

lcr EQU 03FBh
ier EQU 03F9h

ini_com PROC NEAR
    PUSH AX
    PUSH DX
    MOV DX, lcr
    IN AL, DX          ; Lettura del contenuto di LCR
    OR AL, 80h         ; Set DLAB
    OUT DX, AL         ; Scrittura in LCR (bit-rate: 9600)
    MOV AX, 000Ch
    MOV DX, dlr_lsb
    OUT DX, AL
    MOV AL, AH
    INC DX
    OUT DX, AL         ; 1 bit di stop, 8 bit/carattere, parità disabilitata, DLAB = 0
    MOV AL, 03h
    MOV DX, lcr
    OUT DX, AL         ; Disabilitazione delle richieste di interruzione
    MOV AL, 00h
    MOV DX, ier
    OUT DX, AL
    POP DX
    POP AX
    RET
ini_com ENDP
```

4. Temporizzatore di intervalli 8253

4.1 Modello 8253

L'intel 8253 è un chip dedicato a temporizzazione e conteggio, disegnato per essere utilizzato come periferica per microcomputer. Non ha quindi la possibilità di scambiare informazioni da o verso l'esterno, e viene utilizzato dal processore principalmente per:

- generazione di ritardi controllati via software
- generazione di segnali impulsivi o ad onde quadre, e con frequenza programmabile
- misurazione di intervalli
- divisione di frequenza
- conteggio di eventi

La modalità tipica di funzionamento è la seguente:

1. il programmatore configura l'8253
2. il programmatore inizializza uno dei contatori dell'8253 con il valore desiderato
3. l'8253 esegue un conteggio ed interrompe la CPU quando ha completato il suo compito.

L'8253 è stato sostituito dall'8254, che implementa le stesse funzioni con alcune aggiunte.

4.1.1 Il chip 8253

L'8253 è realizzato con tecnologia nMos, e permette di utilizzare 3 contatori indipendenti (ognuno dei quali formato da 16bit) funzionanti ad un clock programmabile fino a 2.6 Mhz.

Si possono separare i pin in diversi gruppi, a seconda del loro utilizzo:

- **D₀₋₇**: contengono la parola di controllo proveniente dal processore
- **A₀₋₁**: indicano a quale contatore si vuole accedere

- **CLOCK – GATE – OUT** (3 x 3 pin): ingresso, output e clock per ogni contatore
- **CS – RD – WR**: selezionano che operazioni effettuare sul dispositivo
- **GND – VCC**: alimentazione e ground del circuito

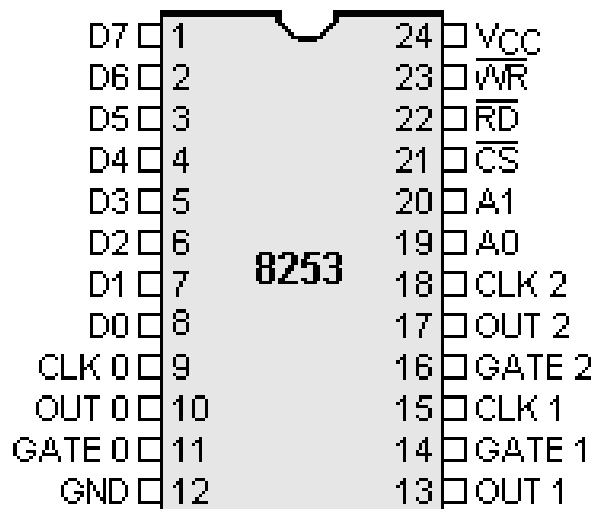


Fig. 17: il chip dell'8253

4.1.2 Modello logico 8253

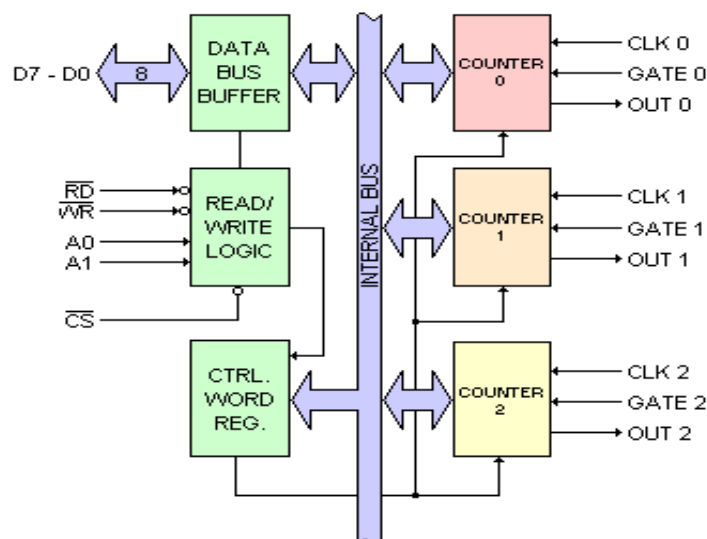


Fig. 18: diagramma a blocchi dell'8253

4.1.3 I contatori

L'8253 include 3 contatori completamente indipendenti da 16 bit ciascuno; ognuno ha un proprio input ($GATE_n$) e output (OUT_n), ed esegue le proprie operazioni ad una velocità dettata dal proprio clock (CLK_n).

Il registro da 16 bit dei contatori può essere letto in ogni momento, e viene programmato indipendentemente dagli altri tramite una parola di controllo impostata dal processore.

Ogni contatore può essere:

- caricato dall'esterno via sw con un valore prefissato
- fatto contare (a ritroso) agendo sul relativo segnale di CLK
- programmato come contatore binario o BCD impaccato
- programmato indipendentemente dagli altri agendo sul Registro di Controllo del dispositivo e scegliendo tra 6 modi diversi di funzionamento
- letto via sw

4.2 Programmazione dell'8250

La modalità di funzionamento dell'8253 è programmabile via software; un insieme di parole di controllo devono essere scritte dalla CPU per inizializzare ciascun contatore (prima dell'inizializzazione, il modo di funzionamento, il contenuto e l'uscita di ogni contatore sono indefiniti).

Per ciascun contatore, il valore del Registro di Controllo determina:

- il modo di funzionamento
- le modalità di caricamento del valore di inizializzazione
- il tipo di conteggio (binario o BCD)

4.2.1 Registro di controllo

SC1	SC0	RL1	RL0	M2	M1	M0	BCD
------------	------------	------------	------------	-----------	-----------	-----------	------------

Il registro di controllo è composto da 8 bit, e viene utilizzato per ricevere la parola di controllo inviata dal processore. È composto da:

SC1 – SC0: utilizzati per selezionare il contatore da programmare

SC1	SC0	Contatore
0	0	#0
0	1	#1
1	0	#2
1	1	Configurazione illegale

RL1 – RL0: predispongono il contatore a operazioni successive

RL1	RL2	Modalità
0	0	Operazione di “Counter Latch”: il valore corrente viene memorizzato staticamente, pronto ad una successiva lettura
0	1	legge/scrive il byte meno significativo
1	0	legge/scrive il byte più significativo
1	1	legge/scrive il byte meno significativo, e poi quello più significativo

M2 – M1 – M0: imposta uno dei 6 metodi di funzionamento

M2	M1	M0	Modo
0	0	0	#0
0	0	1	#1
X	1	0	#2
X	1	1	#3
1	0	0	#4
1	0	1	#5

BCD: definisce il metodo di conteggio. 0 = binario, 1 = BCD

4.3 Modi di funzionamento

Ciascun contatore può essere programmato per funzionare in uno dei seguenti modi:

- modo 0: Interrupt al Termine del Conteggio
- modo 1: One-shot programmabile
- modo 2: Generatore di Frequenza
- modo 3: Generatore di Onde Quadre
- modo 4: S/W Triggered Strobe
- modo 5: H/W Triggered Strobe

Tali modalità di funzionamento possono essere divisi in 2 categorie:

- quelli che generano una forma d'onda dopo N colpi di clock
- quelli che generano forme d'onda periodiche dopo N (divisore di frequenza)

4.3.1 Modo 0 (000)

In questa modalità viene caricato un valore iniziale nel contatore, che verrà decrementato fino al raggiungimento dello 0 ad un rate pari al clock d'ingresso. Dal momento della programmazione l'uscita resterà bassa fino al raggiungimento dello 0, una volta passato allo stato 1, OUT rimarrà tale fino ad una nuova programmazione del contatore o ad una scrittura del contatore. Importante notare che il decremento viene effettuato esclusivamente se l'ingresso gate è impostato a 1.

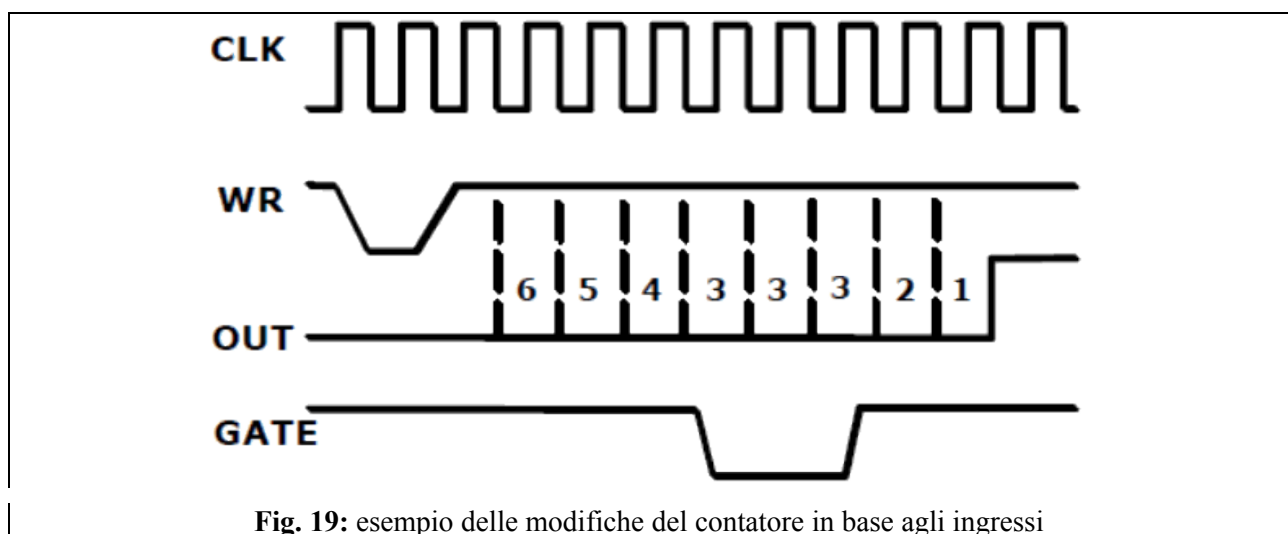


Fig. 19: esempio delle modifiche del contatore in base agli ingressi

4.3.2 Modo 1 (001)

In questa modalità l'8253 viene utilizzato come un multivibratore monostabile.

Dopo aver impostato la parola di controllo il bit di uscita sarà in uno stato logico alto. Nel momento in cui si avrà un ingresso gate alto, out verrà portato a 0 al clock successivo, e rimarrà basso per N cicli di clock, dove N è il valore impostato nel contatore.

È inoltre possibile reiterare il meccanismo senza riscrivere N.

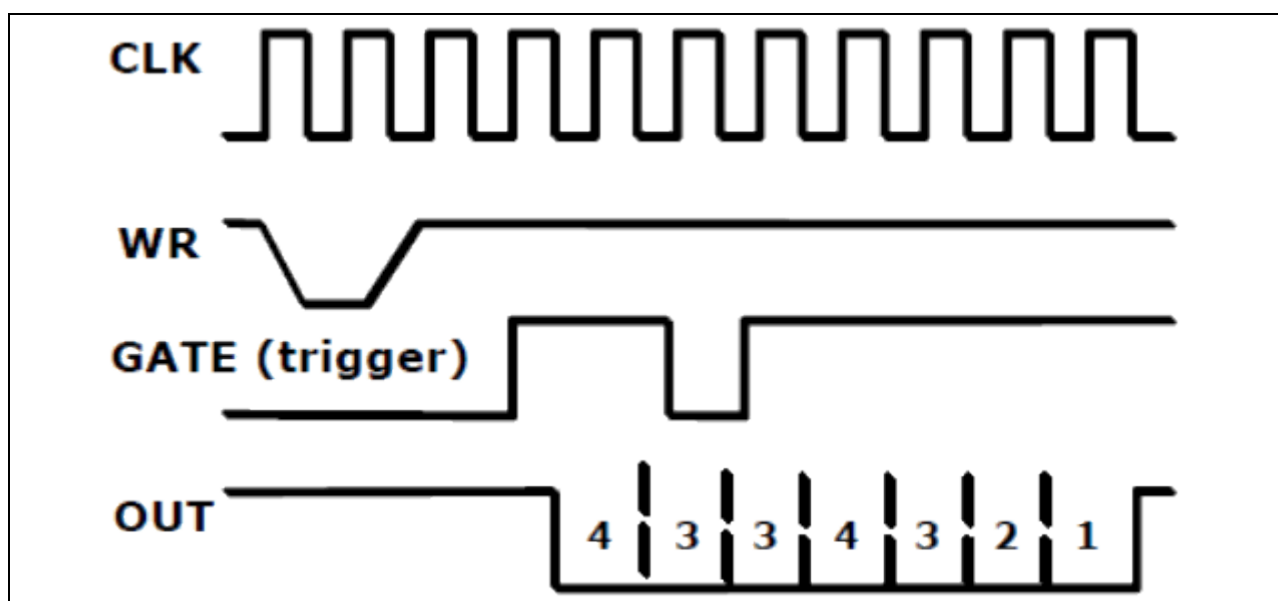


Fig. 20: esempio di funzionamento del modo 1

4.3.3 Modo 2 (X10)

Il dispositivo lavora come un divisore, utile quando si deve generare una serie di interrupt ad un clock specifico. Dal clock successivo all'inserimento di count, esso verrà decrementato fino al raggiungimento del valore 1.

L'uscita passerà quindi dal valore logico 1 al valore logico 0 per un ciclo di clock, e tornerà poi alto per ripetere il processo.

Il valore del contatore che verrà settato deve essere pari al rapporto tra il clock di input e il clock che desideriamo ottenere.

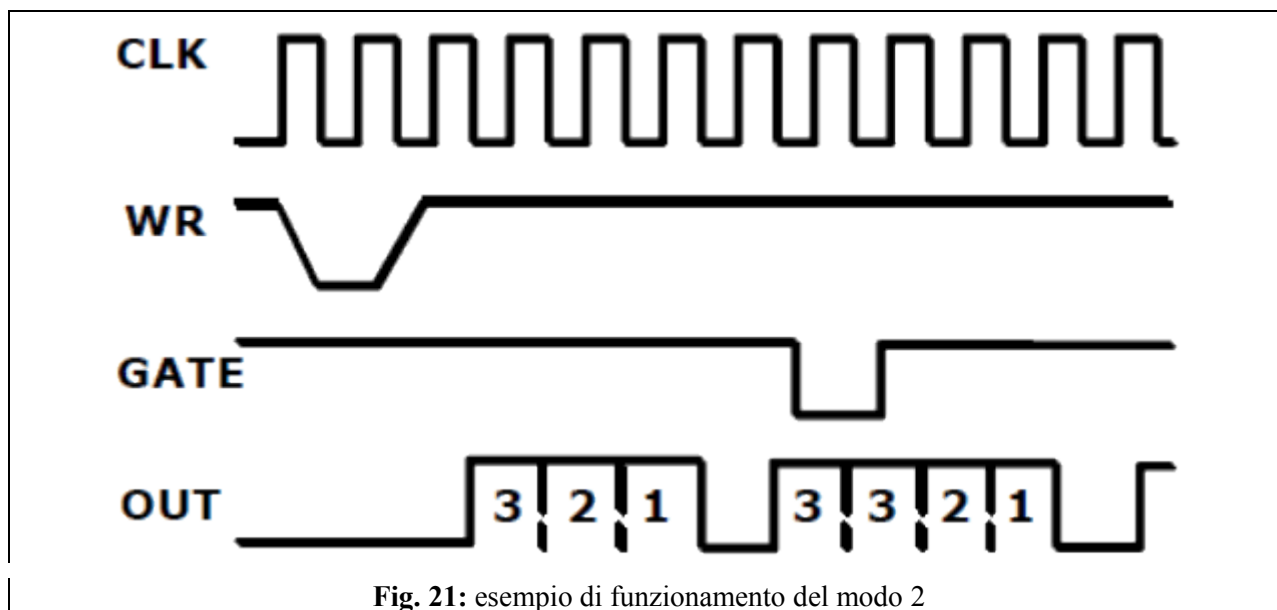


Fig. 21: esempio di funzionamento del modo 2

4.3.4 Modo 3 (X11)

Funzionamento analogo al modo 2, con la differenza che il periodo in cui l'uscita è bassa e quello in cui è alta è identico, ed uguale ad $N / 2$.

Nei casi di n dispari OUT sarà uguale a 1 per $(N+1) / 2$ clock, e basso per $(N-1) / 2$.

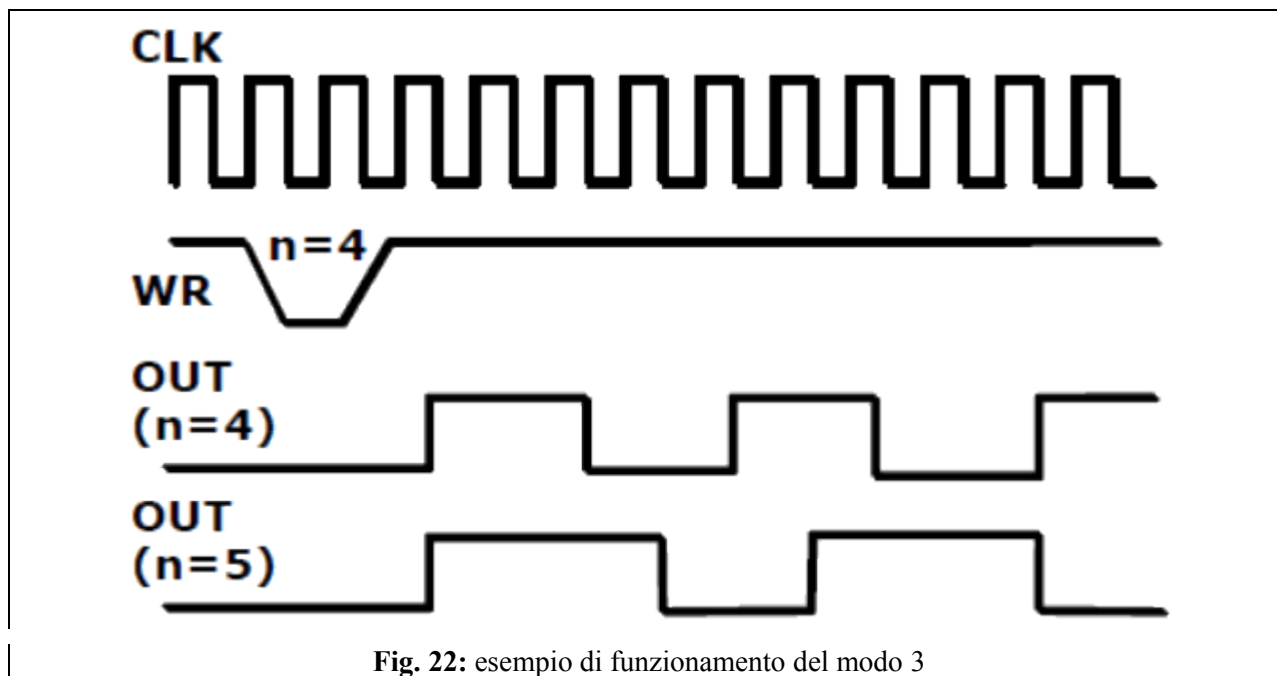
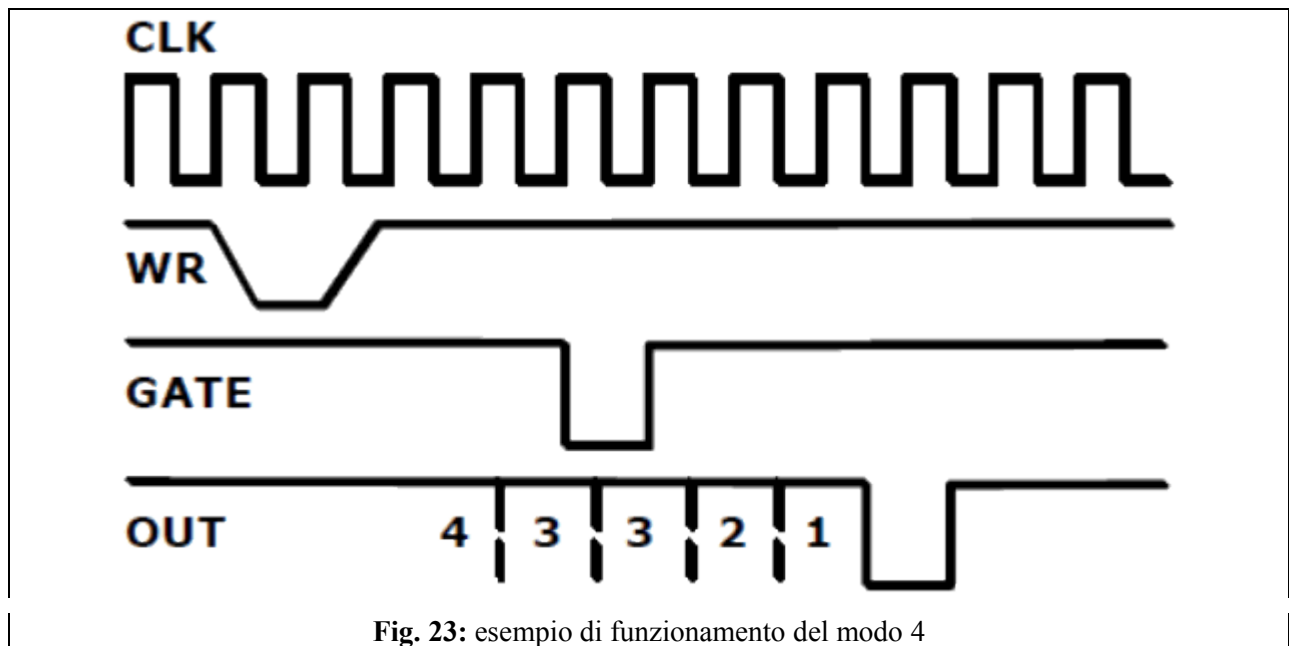


Fig. 22: esempio di funzionamento del modo 3

4.3.5 Modo 4 (100)

Dopo il caricamento del contatore OUT resterà alto finché il contatore non raggiungerà lo 0.

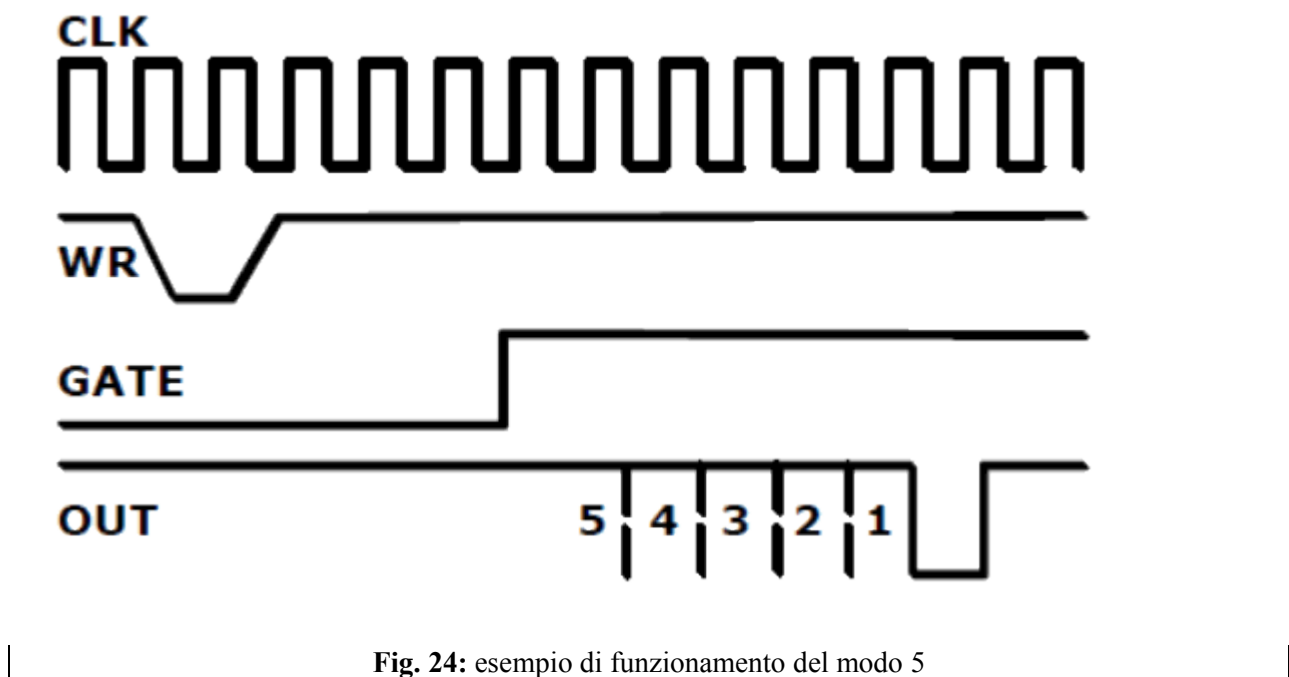
A quel punto si avrà l'uscita bassa per 1 clock, e tornerà alta per ricominciare il ciclo.



4.3.6 Modo 5 (101)

Simile al modo precedente, tuttavia il processo è attivato dal gate.

L'output sarà settato alto, e il dispositivo attenderà un fronte di salita dell'ingresso gate prima di far partire il conteggio.



Una volta raggiunto lo 0 si avrà un ciclo con uscita bassa, e il processo verrà iterato nuovamente.

Esempio di inizializzazione dell'interfaccia:

Si supponga che il dispositivo sia mappato con la parola di controllo all'indirizzo 63h, e con i contatori agli indirizzi 60h, 61h e 62h.

La procedura per l'inizializzazione è la seguente:

```
init_8253 PROC
    MOV AL, 00100000b ;
    OUT 063h, AL      ;inializzo il contatore 0
    MOV AL, 01010100b
    OUT 063h, AL      ;inializzo il contatore 1
    MOV AL, 10010000b
    OUT 063h, AL      ;inializzo il contatore 2
    MOV AL, 00000001b
    OUT 060h, AL      ;carico il valore del contatore 0
    MOV AL, 00000001b
    OUT 061h, AL      ;carico il valore del contatore 1
    MOV AL, 00000101b
    OUT 062h, AL      ;carico il valore del contatore 2
    RET
init_8253 ENDP
```


5. Interrupt

5.1 Gestione in interrupt dei dispositivi

I dispositivi di input/output gestiti in **interrupt** non vengono interrogati direttamente dalla CPU.

Quando uno di essi necessita di essere servito, attiva un segnale di richiesta di interrupt che, rilevato dal processore, porta eventualmente quest'ultimo a interrompere il flusso di esecuzione corrente per passare ad eseguire una opportuna procedura di servizio per la periferica.

La gestione in interrupt dei dispositivi di I/O coinvolge pertanto due diversi aspetti dal punto di vista progettuale:

- *Aspetto software*: costituito dalla procedura di servizio che risponde alle richieste della periferica;
- *Aspetto hardware*: costituito dall'insieme di segnali hardware che veicolano le richieste di interruzione ai piedini del processore su cui tali richieste vengono ricevute e dalla presenza di un eventuale dispositivo controllore dell'interrupt che svolge il ruolo di aggregatore delle richieste quando il numero di piedini ad esse adibito per la CPU non è sufficiente;

La gestione dei dispositivi di I/O in interrupt non è pertanto completamente demandata al software, non si spreca così tempo di CPU per interrogare a turno le periferiche.

Per attivare una richiesta di interrupt, l'interfaccia del dispositivo deve disporre di:

- Un segnale che indichi la presenza di un dato pronto e stabile per la lettura nei registri di dato dell'interfaccia, se questa è relativa ad una periferica di input;
- Un segnale che indichi l'avvenuto completamento di una operazione di scrittura sulla periferica, se questa svolge funzioni di output.

Ci si riferisce a tali segnali di richiesta di interrupt in maniera generica come segnali di *Interrupt Request* (INTR).

In un sistema complesso sono tipicamente presenti un certo numero di dispositivi gestiti in interrupt. Può pertanto capitare il caso in cui più richieste di interrupt vengano scatenate da parte di periferici diversi, prima che la CPU abbia il tempo di servirle.

Per poter rilevare la presenza di più richieste di interrupt, i processori devono disporre di più piedini di interruzione su cui collegare i segnali di INTR provenienti dalle periferiche.

A seconda del numero di piedini di interrupt disponibili per il processore, la gestione degli interrupt può avvenire secondo diversi modelli:

- **Interrupt vettorizzato.** Se la CPU dispone di un unico piedino di interruzione, viene in tal caso utilizzato un apposito dispositivo detto **controllore dell'interrupt** (o *interrupt controller*) che svolge la funzione di aggregatore delle richieste di interruzione. Tale dispositivo raccoglie le richieste di interruzione provenienti dalle periferiche, assegna ad esse una priorità secondo qualche algoritmo ed invia alla CPU un'unica richiesta di interruzione sull'unico piedino disponibile. Quando il processore sarà pronto per servire l'interruzione, il controllore dell'interrupt si farà carico di comunicargli, mediante il bus di sistema, l'identificativo della periferica da servire, cioè di quella a più alta priorità che ha fatto richiesta di interruzione. Tale identificativo verrà utilizzato dalla CPU come indice per accedere in memoria ad una tabella (*Interrupt Vector Table* o **IVT**), memorizzata in posizione fissa, contenente gli indirizzi delle routine di servizio degli interrupt.
- **Linee di interrupt indipendenti.** Quando la CPU fornisce un certo numero di piedini dedicati alla rilevazione delle richieste di interrupt da parte di diversi dispositivi. In tal caso il processore riceve le richieste sui diversi piedini ed è in grado di risolvere subito quale dei

dispositivi servire applicando un certo criterio di priorità. La latenza con cui il processore serve il dispositivo è in questo caso minima.

- **Polling hardware.** Se la CPU dispone di un unico piedino di interruzione, è possibile porre semplicemente in ingresso a tale piedino l'AND di tutti i segnali di richiesta di interruzione. Una volta ricevuto il segnale di interrupt sull'unico piedino disponibile, la CPU inizia a scandire in hardware i registri di stato di tutte le periferiche a turno, secondo un certo ordine di priorità, per identificare la periferica da servire. Nel caso peggiore il processore dovrà scandire lo stato di tutte le periferiche prima di individuare quella che effettivamente necessita di essere servita. Tale tecnica comporta pertanto elevati tempi di latenza.

5.1.1 Principi progettuali della gestione I/O in interrupt

Uno dei requisiti fondamentali nella progettazione di un sottosistema di I/O basato su interrupt è l'essere in grado di garantire che il tempo impiegato dalla CPU per servire un dispositivo che ha fatto richiesta di interruzione non superi un certo tempo di latenza massimo, oltre il quale si perde la correttezza del trasferimento.

Se la periferica in questione è una periferica di input con un certo trasferimento rate e il processore non è abbastanza veloce nel leggere il dato da essa presentato, il dato memorizzato nell'interfaccia di tale periferica verrà sovrascritto con un nuovo dato.

Se la periferica in questione è una periferica di output con un certo trasferimento rate e il processore non è abbastanza veloce nel fornire ad essa un nuovo dato, al posto di tale dato ne viene fornito in output uno non corretto.

Si definisce **tempo di latenza dell'interrupt** (T_{lat}), l'intervallo di tempo che intercorre tra l'istante in cui la richiesta di interrupt viene scatenata da una periferica di I/O e l'istante in cui il

dato oggetto dell' input/output viene trasferito. Tale tempo di latenza può essere definito come la somma di tre tempi:

$$T_{lat} = T_i + T_{hw} + T_r$$

Dove:

- T_i è il tempo necessario, nel caso peggiore, per terminare l'esecuzione dell'istruzione corrente e rilevare la presenza di una richiesta di interrupt.
- T_{hw} è il tempo necessario, nel caso peggiore, per attivare la routine di servizio dell'interrupt. Tale tempo è determinato dal meccanismo hardware di gestione dell'interrupt che viene utilizzato. Nel caso di interrupt vettorizzato può essere scomposto nei seguenti tempi:

$$T_{hw} = t_1 + t_2 + t_3$$

Dove t_1 rappresenta il tempo necessario per implementare il protocollo hardware di comunicazione tra la CPU e il controllore dell'interrupt al fine di ottenere l'identificativo di posizione con cui accedere alla IVT, t_2 è il tempo necessario per salvare lo stato di esecuzione corrente nello stack e t_3 è il tempo necessario per accedere in memoria alla posizione specificata della IVT e saltare all'indirizzo di inizio della routine di servizio dell'interrupt.

- T_r è il tempo che intercorre, nel caso peggiore, tra l'inizio dell'esecuzione della routine di servizio dell'interrupt e l'esecuzione dell'istruzione che effettua il trasferimento dati vero e proprio.

Affinchè il trasferimento di dati tra la periferica e la CPU avvenga correttamente, occorre dimensionare tale tempo di latenza in modo che risulti compatibile con il transfer rate del periferico. Deve pertanto valere la relazione:

$$T_{lat} < \frac{1}{T.R.}$$

Al fine di garantire tale compatibilità, la massima priorità di interruzione è attribuita di norma al periferico con il massimo transfer rate, che deve essere servito con la minima latenza. In tal modo un periferico lento può essere interrotto da uno veloce. Bisogna però in tal caso garantire che il tempo di latenza massimo per il periferico più lento non sia superato dai tempi di esecuzioni di tutte le procedure di servizio dei periferici più veloci annidate.

Esempio:

Una scheda di rete Ethernet è caratterizzata da una velocità di trasferimento pari a 10Mbps e da registri di dato interno di parallelismo pari a un byte. Supponendo che tale scheda non sia bufferizzata e che sia gestita in interrupt, calcolare il valore massimo del tempo di latenza che assicura la correttezza dei trasferimenti dati che coinvolgono tale periferica:

$$T.R. = 10 \text{ Mbps} \rightarrow T.R. = 1.25 \text{ Mbps}$$

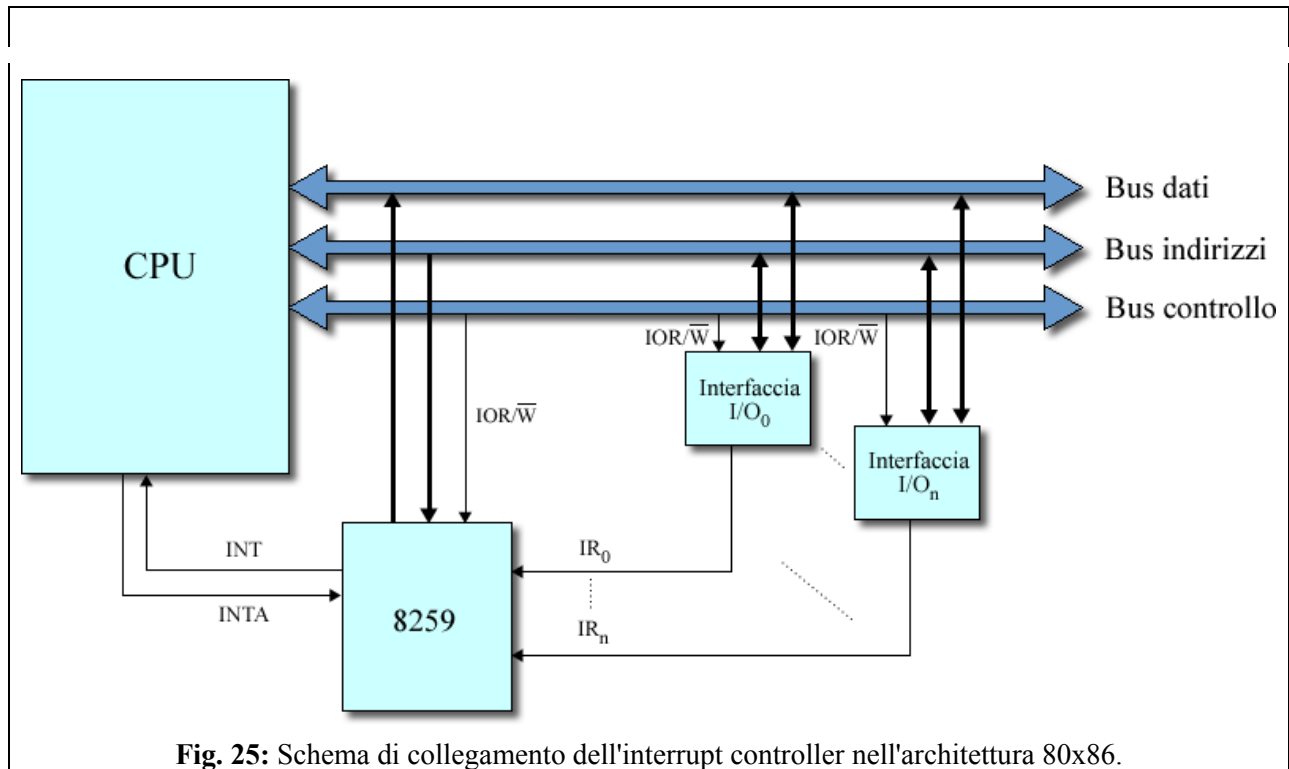
$$T_{lat} < \frac{1}{T.R.} \rightarrow T_{lat} < 0.8 \mu s$$

5.2 Interrupt nell'architettura 80x86

I processori della famiglia 80x86 permettono di gestire i dispositivi di input/output in interrupt, secondo il modello di interrupt vettorizzato. Tali processori mettono a disposizione infatti un unico piedino, chiamato INT, su cui ricevere i segnali di richiesta di interruzione provenienti dalle periferiche.

Un dispositivo **controllore dell'interrupt** (o *interrupt controller*) ha il compito di raccogliere le richieste di interruzione provenienti dalle varie periferiche, assegnare ad esse una priorità ed inviare alla CPU una unica richiesta di interruzione sul piedino INT. Quando il processore sarà poi pronto per servire l'interruzione, il controllore dell'interrupt dovrà infine di comunicare, mediante il bus di sistema, quale dispositivo di I/O servire, sulla base delle priorità assegnate.

Il controllore dell'interrupt di riferimento per le architetture 80x86 è il dispositivo PIC (*Programmable Interrupt Controller*) 8259. Tale dispositivo, generalmente integrato nel chipset, è collegato alla CPU attraverso il bus di sistema.



Oltre al piedino INT, i processori della famiglia 80x86 dispongono di un piedino INTA (*Interrupt Acknowledge*), mediante il quale possono implementare un semplice protocollo di comunicazione con il controllore dell'interrupt, e un piedino NMI (*Non-maskable Interrupt*) su cui ricevere richieste di interruzione non mascherabili provenienti da un opportuno circuito.

Oltre alle componenti hardware descritte, i processori della famiglia 80x86 forniscono un certo numero di istruzioni dedicate alla gestione delle interruzioni. Tra queste le istruzioni:

- **INT:** permette di scatenare via software una richiesta di interruzione. Utilizzata soprattutto per effettuare chiamate a procedure di servizio del sistema operativo (*system calls*);

- **INTO**: permette di scatenare via software una richiesta di interruzione in maniera condizionata al valore del flag di overflow.
- **IRET**: è una speciale istruzione di ritorno, utilizzata al termine dell'esecuzione di una procedura di servizio dell'interrupt per ripristinare lo stato di esecuzione interrotto;
- **CLI** e **STI**: permettono di disabilitare o abilitare la rilevazione di interrupt mascherabili provenienti dall'esterno;

5.2.1 Tipi di Interrupt

Nei processori della famiglia 80x86 le interruzioni possono essere classificate in quattro grandi categorie:

- **Interrupt interni o trap**: sono richieste di interruzione direttamente generate dal processore in seguito alla rilevazione di condizioni anomale durante l'esecuzione delle istruzioni. In funzione della modalità operativa, il processore monitora il verificarsi di determinate condizioni anomale e, al presentarsi di una di queste, scatena la richiesta di interruzione per interrompere il programma corrente e passare ad eseguire la procedura opportuna per gestire tale situazione. Esempi di trap sono la *divisione per zero*, l'*overflow* o l'*invalid memory access* (in protected mode).
- **Interrupt esterni o mascherabili**: sono richieste di interruzione provenienti via hardware dai dispositivi periferici. La rilevazione di tali interrupt da parte del processore può essere abilitata o disabilitata attraverso opportune istruzioni macchina. Il flag **IF** della *Process Status Word*, indica se la rilevazione delle interrupt mascherabili è correntemente abilitata o disabilitata sul processore. Al termine dell'istruzione corrente il processore verificherà la presenza di richieste di interrupt esterne se e solo se il flag IF risulta abilitato.

- **Interrupt non mascherabili:** sono richieste di interruzione provenienti via hardware dall'esterno del processore che non possono essere ignorate dalla CPU. Al termine della istruzione corrente il processore rileverà la presenza di una interrupt non mascherabile indipendentemente dal valore del flag IF. Sono tipicamente scatenate in seguito al verificarsi di eventi critici quali la caduta dell'alimentazione o un errore di lettura in memoria.
- **Interrupt software:** sono richieste di interruzione scatenate via software in seguito all'esecuzione di opportune istruzioni, come l'istruzione **INT**. Permettono di richiamare routine di servizio, precedentemente definite, e i cui indirizzi siano stati preventivamente caricati nel vettore delle interruzioni. Tipicamente tali routine sono procedure di servizio del sistema operativo. Le interrupt software permettono di eseguire chiamate indirette a determinate procedure la cui posizione all'interno della memoria non è fissa. L'allocazione dei moduli di un sistema operativo in memoria ad esempio non è fissa, le procedure di sistema possono essere caricate in memoria ogni volta ad indirizzi diversi. Non si può pertanto fare riferimento ad un determinato indirizzo per richiamare una procedura di servizio del sistema, come ad esempio quella per l'acquisizione di un carattere da tastiera. Si utilizza quindi una interruzione software per richiamare indirettamente la procedura desiderata avendo memorizzato, al caricamento del sistema operativo, l'indirizzo di tale procedura in una delle posizioni libere della IVT.

5.2.2 Gestione degli Interrupt

La CPU, terminata l'esecuzione dell'istruzione corrente, verifica la presenza di richieste di interruzione ed eventualmente passa ad eseguire una routine di servizio opportuna. La **routine di servizio dell'interrupt** o **ISR** (*Interrupt Service Routine*), viene chiamata indirettamente,

accedendo mediante un indice, caratteristico per l'interruzione rilevata, ad un vettore di puntatori memorizzato in una zona fissa della memoria. Tale vettore, detto **Interrupt Vector Table** o **IVT**, contiene gli indirizzi delle procedure di servizio degli interrupt.

La IVT nei processori della famiglia 80x86, in modo reale, contiene fino a 256 puntatori a routine di servizio, ognuno di questi memorizzato su 4 byte, dove:

- I due byte più alti contengono l'indirizzo del segmento di codice in cui è posizionata la procedura di servizio;
- I due byte più bassi contengono l'offset che indica la posizione, all'interno del segmento specificato, della procedura di servizio;

Al termine dell'esecuzione dell'istruzione corrente, il processore verifica nel seguente ordine la presenza di richieste di interruzione:

1. Si controlla se l'esecuzione dell'istruzione appena terminata ha portato al verificarsi di condizioni anomale associate a interrupt interni (trap). In tal caso la CPU scatena una interruzione interna. La posizione all'interno della IVT a cui accedere per richiamare la routine di servizio corrispondente viene generata direttamente dalla CPU sulla base della condizione anomala rilevata.
2. Si verifica la presenza di richieste di interrupt non mascherabili sul piedino NMI. Nel caso in cui queste siano presenti, vengono servite accedendo all'elemento in seconda posizione all'interno della IVT;
3. Se il flag IF è attivato, si verifica la presenza di richieste di interrupt esterni sul piedino INT. Nel caso in cui queste siano presenti, il processore comunicherà mediante un opportuno protocollo con il controllore degli interrupt per identificare la posizione all'interno della IVT in cui è memorizzato il puntatore alla ISR da eseguire;

4. Si verifica il valore del flag **TF** della *Process Status Word*, se tale flag risulta abilitato il processore scatena internamente una interruzione *single-step*, la cui ISR è puntata dall'elemento in prima posizione all'interno del vettore delle interruzioni;

Le interruzioni software vengono invece servite in maniera sincrona nel momento in cui le istruzioni che le specificano vengono eseguite.

Alla prima richiesta di interruzione rilevata, procedendo nell'ordine esposto, il processore esegue le seguenti operazioni:

- Il contenuto del registro flag è salvato nello stack;
- Entrambi i flag IF e TF vengono disabilitati. Ciò impedisce ad ulteriori richieste di interrupt hardware esterne di interrompere l'esecuzione della procedura di servizio e disabilita la modalità di esecuzione *single-step* per tale procedura.
- Il contenuto del registro di segmento di codice (CS) viene salvato nello stack;
- Il contenuto dell'Instruction Pointer (IP) viene salvato nello stack;
- Si accede all'elemento della IVT associato all'interruzione da servire. Per effettuare tale accesso il processore moltiplica per quattro l'indice di posizione della ISR all'interno della IVT precedentemente ottenuto e utilizza il risultato come indirizzo per accedere in memoria. L'elemento acceduto contiene il puntatore alla ISR da eseguire, si caricano pertanto nei registri IP e CS rispettivamente i due byte più bassi e più alti in esso memorizzati.
- Si esegue la routine di servizio dell'interrupt;

Il processore esegue la stessa sequenza di operazioni durante l'esecuzione dell'istruzione di interruzione software INT o INTO, in tal caso la posizione all'interno della IVT a cui accedere è codificata direttamente all'interno dell'istruzione.

Al termine della ISR, l'esecuzione del programma interrotto viene ripresa effettuando le seguenti operazioni:

- Il valore dell'indirizzo di offset di ritorno viene estratto dallo stack e caricato nel registro IP;
- Il valore dell'indirizzo di segmento di ritorno viene estratto dallo stack e caricato nel registro CS;
- Lo stato del processore precedente all'interruzione viene estratto dallo stack e caricato nel registro flag;
- Si riprende l'esecuzione del programma interrotto;

5.2.3 Priorità degli Interrupt

L'ordine di priorità tra i tipi di interruzione permette di definire:

- Quale richiesta di interruzione servire nel caso in cui richieste di interruzione di diverso tipo si presentino contemporaneamente;
- Quali richieste di interruzione possono interrompere una routine di servizio dell'interrupt correntemente in esecuzione;

Al termine di una istruzione ci si può trovare in una situazione in cui più interrupt di tipo diverso sono state scatenate e attendono di essere servite. In particolare:

- Una interrupt mascherabile è stata scatenata se è presente un valore logico alto sul piedino INTR;
- Una interrupt non mascherabile è stata scatenata se è presente un valore logico alto sul piedino NMI;
- Una interrupt interna è stata scatenata se si è verificata una condizione anomala in seguito all'esecuzione dell'istruzione;

- Una interrupt software è stata scatenata se è stata eseguita una istruzione di interruzione come INT o INTO;

Sulla base del procedimento di gestione delle interruzioni precedentemente esposto, è possibile definire l'ordine di priorità con cui il processore decide quale interrupt servire, in presenza di interrupt di diverso tipo in attesa di essere servite al termine di una istruzione.

Tale ordine di priorità, presentato per priorità dalla maggiore alla minore, è il seguente:

- *Interrupt interni.* Gli interrupt interni vengono verificati per primi al termine dell'istruzione corrente. Se una condizione anomala è incorsa la corrispondente procedura di servizio viene immediatamente eseguita. L'incorrere delle varie condizioni anomale monitorate viene verificato in un ordine prestabilito e non modificabile. Tale ordine definisce una relazione di priorità fissa tra gli interrupt di tipo interno.
- *Interrupt non mascherabili.* Gli interrupt non mascherabili vengono verificati subito dopo gli interrupt interni, indipendentemente dal valore del flag IF. Tali interrupt vengono serviti mediante una unica procedura di servizio il cui indirizzo è situato in seconda posizione all'interno della IVT, all'interno di tale ISR le possibili cause dell'interrupt non mascherabile verranno interrogate in polling secondo un ordine fisso di priorità.
- *Interrupt software.* Gli interrupt software godono di una priorità maggiore di quelli mascherabili dal momento che, durante l'esecuzione dell'istruzione di interruzione che gli scatena, il flag IF viene disabilitato. Al termine dell'esecuzione di tale istruzione, le richieste di interruzioni mascherabili, se presenti, vengono pertanto ignorate, mentre eventuali richieste di interruzione interne o non mascherabili vengono servite.
- *Interrupt hardware esterni.* Gli interrupt mascherabili presentano una priorità minore di quelli software perchè il flag IF viene disabilitato durante l'esecuzione delle istruzioni di

interruzione software. Presentano inoltre una priorità minore degli interrupt di tipo interno o non mascherabile perchè verificati per ultimi al termine dell'esecuzione di ogni istruzione.

All'interno dei gruppi di interrupt software e interrupt mascherabili si distingue una relazione di priorità tra le interruzioni, basata sul numero identificativo della posizione all'interno della IVT ad esse associato. Interrupt di tale tipo memorizzate hanno priorità tanto maggiore quanto più è basso l'identificativo di vettore ad esse associato.

L'esecuzione di una ISR può essere interrotta solo da interrupt appartenenti ad un gruppo di priorità più elevata o a interrupt con priorità più elevata appartenenti allo stesso gruppo. La priorità delle singole interruzioni all'interno dei gruppi di interrupt software e mascherabili può essere modificata variando le posizioni ad esse associate all'interno della IVT. Gli interrupt interni e non mascherabili hanno invece priorità fissa.

5.2.4 Protocollo di Interrupt

Quando il processore rileva la presenza di una richiesta di interrupt hardware esterno sul piedino INT, deve comunicare con il controllore delle interruzioni al fine di identificare la routine di servizio dell'interrupt da eseguire.

La comunicazione con il controllore dell'interrupt avviene attraverso un opportuno protocollo che coinvolge il bus dati e i segnali INT, INTA del processore.

Tale protocollo prevede i seguenti passi:

1. Il controllore delle interruzioni, ricevute richieste di interruzione da parte dei dispositivi ad esso collegato, attiva il segnale INT diretto al processore;
2. Il processore, quando è pronto a gestire eventuali richieste di interrupt mascherabili e il flag IF è abilitato, rileva la presenza sul piedino INT di una richiesta di interruzione. Invia

allora al controllore degli interrupt un impulso sul segnale INTA per segnalare che l'interrupt è stato rilevato;

3. Il processore invia poi un secondo impulso sul segnale INTA per richiedere al controllore degli interrupt di inviare sugli 8 bit bassi del data bus, il numero identificativo dell'interruzione da servire, stabilito internamente dall'interrupt controller sulla base delle priorità delle richieste di interrupt attive;
4. Il processore legge dal bus dati il numero l'identificativo della posizione all'interno della IVT della routine di servizio da eseguire;
5. La CPU prosegue eseguendo la sequenza di operazioni, precedentemente descritta, per l'esecuzione della procedura di servizio dell'interrupt;

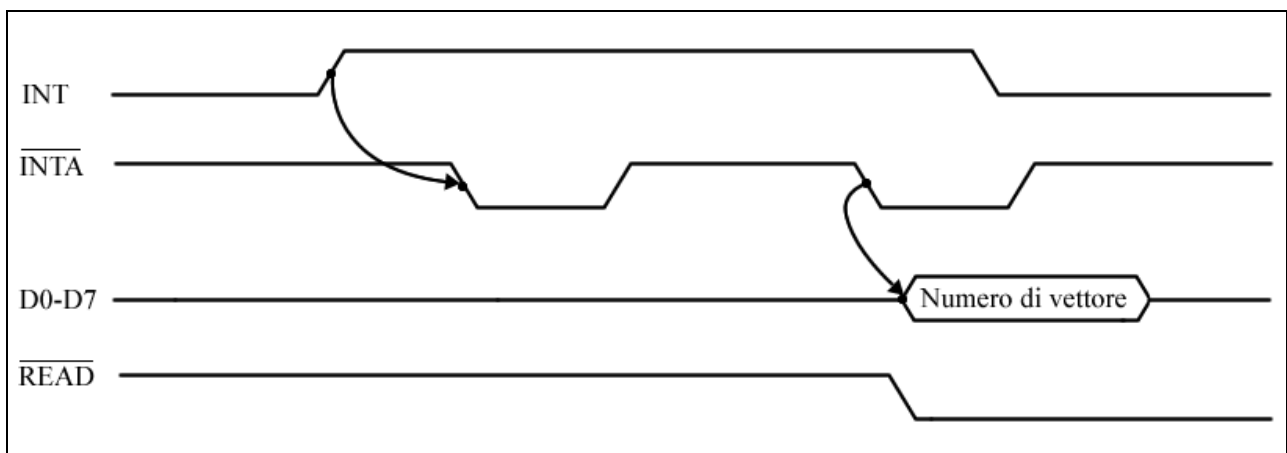


Fig. 26: Tempistica del protocollo di interruzione.

La definizione della posizione all'interno della IVT a cui accedere per eseguire la ISR, è demandata al dispositivo controllore dell'interrupt. Tale dispositivo è programmato dal BIOS in fase di avviamento del sistema, con un valore iniziale k . Tale valore rappresenta la posizione di partenza all'interno della IVT in cui verranno memorizzati i puntatori alle ISR relative alle interruzioni hardware gestite dal controllore. Il controllore degli interrupt associa ad ogni sua linea INTR di ingresso un indice i di scostamento. L'identificativo di posizione all'interno della IVT,

verrà generato dal controllore dell'interrupt sommando il valore iniziale k al valore di scostamento i relativo alla richiesta di interruzione attiva con priorità maggiore. Il criterio di priorità adottato normalmente dai controllori di interrupt è quello che associa priorità maggiori alle richieste ricevute sulle linee con associati indici i minori.

Esempio

Si consideri una interfaccia asincrona UART (di tipo 16550) programmata, che presenta le seguenti caratteristiche:

- Trasmissione costituita da 8 bit di dati, un bit di inizio, un bit di parità e un bit di stop (11 bit totali);
- Baud rate 56Kbaud;

Determinare il massimo tempo di latenza dell'interrupt superato il quale si presenta un errore di overrun. Supponendo che:

- L'interfaccia UART sia attestata su un bus ISA, caratterizzato da parallelismo del bus dati pari a 16 bit, frequenza di funzionamento pari a 8MHz e durata del ciclo di bus pari a due periodi di clock;
- Il flag IF sia abilitato e non sopraggiungano richieste di interruzione da altri dispositivi;
- Il tempo di accesso in memoria sia $t_{\text{mem}}=100$ ns e la cache sia disabilitata;
- Il massimo tempo di esecuzione delle istruzioni sia dell'ordine del μs ;
- All'interno della procedura di servizio dell'interrupt relativa alla interfaccia UART, prima di accedere effettivamente a tale interfaccia vengano eseguite 10 istruzioni;

stimare il tempo di latenza dell'interrupt.

Il tempo di bit dell'UART vale:

$$t_{\text{bit}} = \frac{1}{56 \cdot 10^3} \approx 18 \mu\text{s}$$

il tempo necessario per caricare un byte nell'UART è pertanto:

$$t_{\text{byte}} = 11 \cdot t_{\text{bit}} \approx 198 \mu\text{s}$$

Al fine di evitare errori di overrun il byte ricevuto dall'UART deve essere letto entro 198 μs dalla richiesta di interrupt.

$$t_{\text{lat}} < 198 \mu\text{s}$$

Il tempo di accesso alle interfacce dei periferici attestati sul bus ISA è:

$$t_{\text{aIO}} = \frac{2}{8 \cdot 10^6} = 250 \text{ ns}$$

Il tempo T_i necessario nel caso peggiore per terminare l'esecuzione dell'istruzione corrente e rilevare l'interrupt è pari a circa 1 μs (durata massima supposta per una istruzione).

Il tempo T_{hw} necessario per passare dal rilevamento dell'interruzione all'esecuzione della ISR è dato da:

- t_1 : tempo necessario per leggere il numero di vettore n da servite, posto dal controllore dell'interrupt sul bus ISA, è pari alla durata di un ciclo di bus;
- t_2 : tempo necessario per salvare lo stato corrente nello stack, è costituito da tre accessi in memoria (salvataggio della parola di stato, del registro di segmento e dell'offset correnti);
- t_3 : tempo necessario per accedere al vettore delle interruzioni e passare ad eseguire la ISR, corrisponde ad un accesso in memoria alla locazione $4 \cdot n$;

Pertanto:

$$T_{\text{hw}} = t_{\text{aIO}} + 3 \cdot t_{\text{mem}} + t_{\text{mem}} = 250 \text{ ns} + 300 \text{ ns} + 100 \text{ ns} = 650 \text{ ns}$$

Il tempo T_r , che intercorre dall'inizio dell'esecuzione della routine di servizio e l'esecuzione dell'istruzione di lettura dei dati della UART è pari al tempo di esecuzione di 10 istruzioni, nel caso peggiore pertanto: $T_r = 10 \mu\text{s}$.

Il tempo di latenza stimato risulta essere:

$$T_{\text{lat}} = T_i + T_{\text{hw}} + T_r = 1 + 0.65 + 10 \approx 12 \text{ } \mu\text{s}$$

Essendo tale tempo di latenza molto minore di 200 μs , tale configurazione non determina errori di overrun.

5.2.5 Interrupt Vector Table

Al fine di identificare la posizione in memoria della routine di servizio da eseguire in seguito ad una richiesta di interruzione, ad ognuna di tali routine viene assegnato in numero identificativo.

Una volta rilevata la richiesta di interrupt, il corrispondente numero identificativo viene calcolato direttamente dal processore o chiesto al dispositivo controllore dell'interrupt, sulla base del tipo di interrupt rilevato.

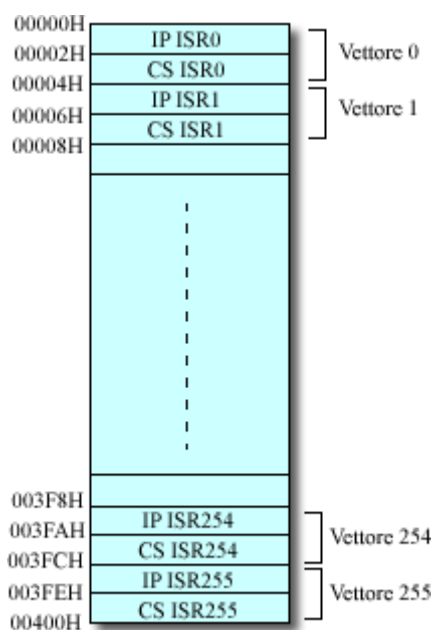


Fig. 27: Struttura della Interrupt Vector Table nei processori 80x86 in modo reale.

Tale numero viene poi utilizzato come indice per accedere ad un vettore di memoria contenente, per ciascuna routine di servizio, un puntatore alla prima istruzione di tale routine.

Questo vettore è detto **vettore delle interruzioni** o **IVT** (*Interrupt Vector Table*) e risiede in una zona fissa della memoria.

Nei processori 80x86, in modo reale, la IVT è memorizzata nel KB basso della memoria, in corrispondenza degli indirizzi che vanno da 00000h a 003FFh, ma può essere rilocata durante il funzionamento in modo protetto mediante un registro interno della CPU.

La IVT contiene fino a 256 puntatori a routine di servizio dell'interrupt, ognuno di essi memorizzato su 4 byte:

- I due byte più alti contengono l'indirizzo del segmento di codice in cui è posizionata la procedura di servizio;
- I due byte più bassi contengono l'effective address che indica la posizione, all'interno del segmento specificato, della procedura di servizio;

Gli indici con cui il processore accede alla tabella identificano uno dei 256 puntatori a ISR da quattro byte. Pertanto, l'accesso alla IVT in posizione n , si traduce in un accesso in memoria ai 4 byte memorizzati a partire dalla locazione $4 \cdot n$.

All'interno della IVT si distinguono:

- *Posizioni fisse*: riservate per la memorizzazione di puntatori a procedure di servizio dell'interrupt determinate e non modificabili. Sono le prime cinque posizioni del vettore delle interruzioni e sono identiche per tutti i processori della famiglia 80x86. Tra queste sono presenti le posizioni relative alle ISR di gestione di alcuni interrupt interni e delle interrupt non mascherabili:
 - *Posizione 0 (Divide Error)*: contiene l'indirizzo della ISR da eseguire in seguito al verificarsi della condizione di divisione per zero o di overflow durante l'operazione di divisione;

- *Posizione 1 (Single Step)*: contiene l'indirizzo della ISR da eseguire in seguito all'interruzione interna di esecuzione single step;
- *Posizione 2 (Non-maskable Interrupt)*: contiene l'indirizzo della ISR da eseguire in seguito alla rilevazione di un'interruzione di tipo non mascherabile;
- *Posizione 3 (Breakpoint)*: contiene l'indirizzo della ISR da eseguire in seguito all'esecuzione dell'interruzione software speciale INT 3, codificata su un solo byte, ed utilizzata nel debug dei programmi per inserire breakpoint;
- *Posizione 4 (Overflow)*: contiene l'indirizzo della ISR da eseguire in seguito all'esecuzione dell'interruzione software INTO, utilizzata per reagire in modo apposito ad una condizione di overflow;
- *Posizioni riservate*: i processori 80x86 riservano le posizioni che dalla 5 vanno alla 31 per routine di servizio destinate alla gestione di particolari interruzioni interne relative al modo protetto o di interruzioni hardware esterne. Tra le più importanti di queste:
 - *Posizione 6 (Invalid Opcode)*: contiene l'indirizzo della ISR da eseguire in seguito al verificarsi della condizione di rilevamento di un opcode indefinito all'interno di un programma;
 - *Posizione 11 (Segment not Present)*: contiene l'indirizzo della ISR da eseguire in seguito al verificarsi della condizione di rilevamento di un tentativo di accesso ad un segmento di memoria non presente o non valido;
 - *Posizione 13 (General Protection)*: contiene l'indirizzo della ISR da eseguire in seguito al verificarsi della condizione di rilevamento di un tentativo di violazione della memoria protetta;
 - *Posizione 14 (Page Fault)*: contiene l'indirizzo della ISR da eseguire in seguito al verificarsi della condizione di page fault;

- *Posizioni libere*: in cui è possibile memorizzare indirizzi a routine di servizio del BIOS, del sistema operativo, o dei programmi utente, vanno dalla posizione 32 alla 255;

5.3 Interrupt Controller 8259/APIC

Il dispositivo 8259 è un **controllore degli interrupt programmabile** (*PIC: Programmable Interrupt Controller*) che permette di gestire fino a otto linee di interrupt vettorzate a diversi livelli di priorità. Tale dispositivo è progettato per minimizzare il codice e i tempi di risposta necessari per la gestione delle interruzioni a diversi livelli di priorità.

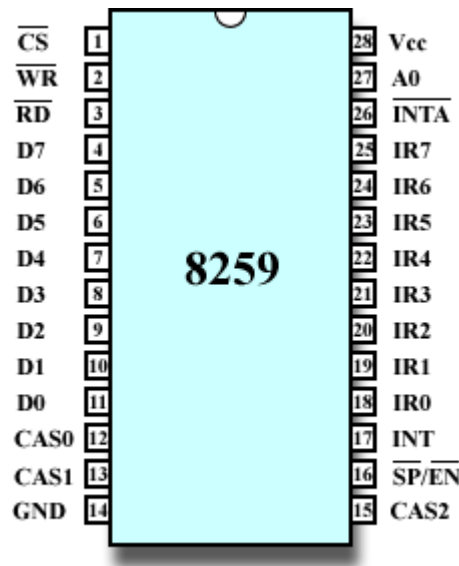


Fig. 28: Piedinatura del controllore delle interruzioni 8259.

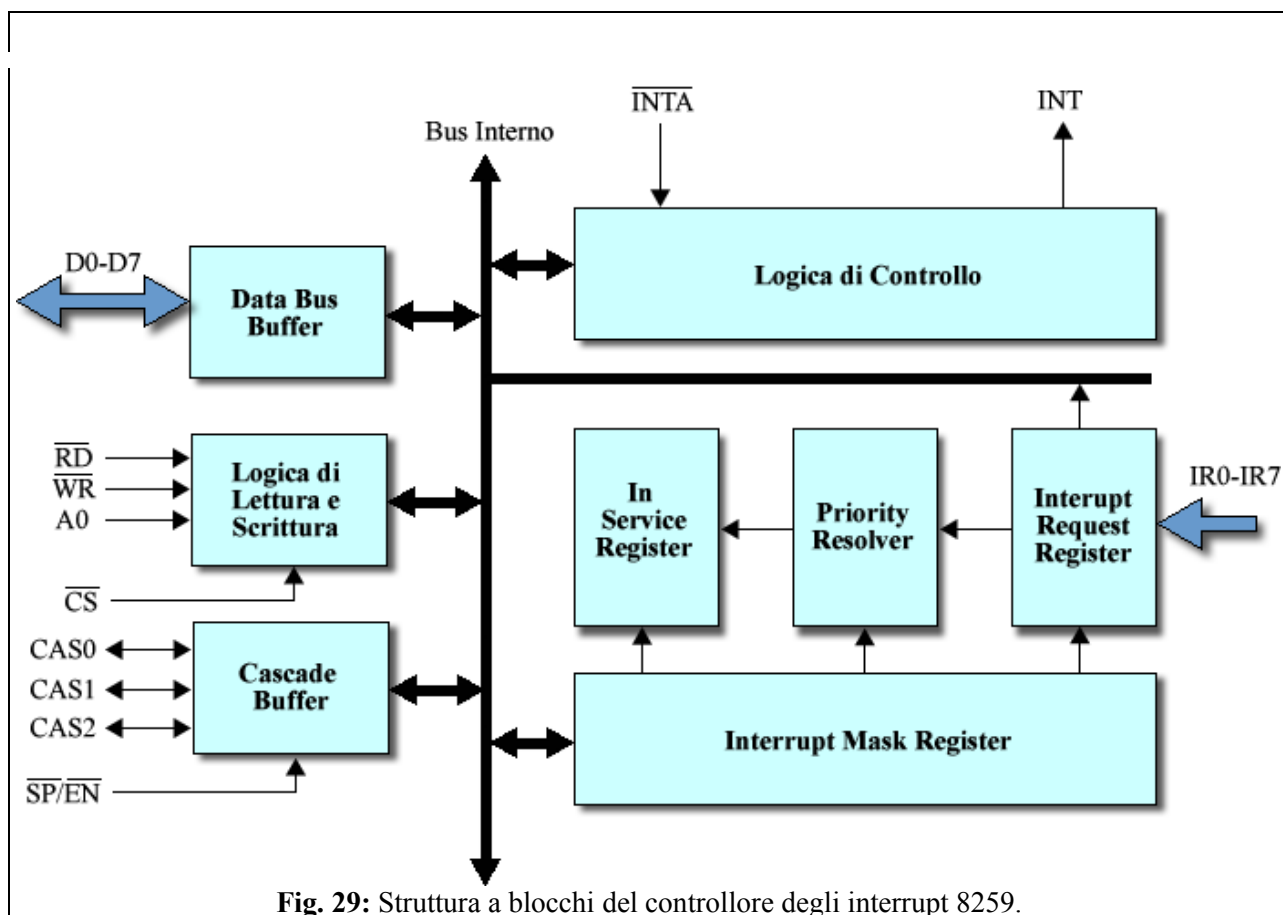
Tale dispositivo può essere programmato per utilizzare diversi metodi di assegnazione delle priorità alle linee di interrupt. La programmazione dell'8259 avviene inviando opportune parole di controllo per mezzo degli otto segnali bassi del data bus.

In origine il controllore 8259 era costituito da un chip a sé stante, mentre nei moderni PC due di tali dispositivi sono incorporati all'interno del chipset, tipicamente posizionati nel south bridge.

Il controllore degli interrupt 8259 può essere facilmente espanso per gestire fino a 64 linee di richiesta di interruzione, collegando fino a otto *controllori slave*, in cascata alle linee di interrupt di un *controllore master*.

5.3.1 Struttura del controllore 8259

La struttura interna del controllore delle interruzioni 8259 può essere descritta mediante il diagramma a blocchi riportato in Fig.29.



Il **data bus buffer** è un buffer tri-state bidirezionale a 8 bit in cui vengono memorizzati i dati letti da o che devono essere scritti sul data bus. Le parole di controllo, le parole di stato e gli identificativi dei vettori, vengono tutti fatti transitare in tale buffer.

La **logica di controllo** è l'elemento decisionale del dispositivo, controlla e coordina gli altri blocchi funzionali e gestisce i piedini di *INT* e *INTA* su cui rispettivamente invia al processore la richiesta di interruzione e riceve da quest'ultimo i segnali di acknowledge.

L'**in service register** (ISR), è un registro da 8 bit utilizzato per tenere traccia della richiesta di interruzione correntemente in servizio. Ciascuno degli 8 bit corrisponde ad una delle 8 linee di interruzione. Tra questi solo il bit corrispondente alla richiesta di interruzione correntemente in servizio risulterà abilitato.

Il registro **interrupt request register** (IRR) è utilizzato per memorizzare su quali delle 8 linee di interrupt è stata ricevuta una richiesta di interruzione. Consiste di 8 bit, ad ognuno dei quali corrisponde una delle linee di interrupt, e che indicano, se pari a uno, la presenza di una richiesta di interruzione su tale linea.

Il blocco **priority resolver** si occupa di determinare la priorità delle richieste di interruzione attualmente attive (corrispondenti a bit abilitati nel registro IRR), sulla base della particolare modalità di assegnazione delle priorità impostata per il controllore in fase di programmazione. Tale componente identifica la richiesta di interruzione a priorità maggiore e abilita, una volta ricevuto il segnale di *INTA* dalla CPU, il corrispondente bit all'interno del registro ISR.

L'**interrupt mask register** (IMR) è un registro a 8 bit, che può essere programmato per memorizzare una particolare *maschera*, mediante la quale è possibile disabilitare a priori specifiche linee di interrupt. A ciascuno degli 8 bit di tale registro corrisponde una delle 8 linee di interrupt, ogni linea risulterà non mascherata nel caso in cui il corrispondente bit di IMR sia a 0, mascherata altrimenti. Una linea di interrupt mascherata via software, programmando il registro IMR in modo opportuno, non potrà essere riconosciuta e servita dal processore.

La **logica di lettura/scrittura** gestisce le operazioni di lettura e scrittura che coinvolgono il controllore delle interruzioni:

- Scrittura di *parole di controllo*;
- Lettura di *parole di stato* o *identificativi di vettore*;

Comprende al suo interno una serie di registri in cui il dispositivo memorizza le parole di controllo ricevute in fase di programmazione. Il segnale di chip select (\overline{CS}) permette alla CPU di selezionare il dispositivo, i segnali di \overline{RD} , \overline{WR} e $A0$ permettono invece alla CPU di definire il tipo di operazione di lettura o scrittura ad esso richiesta.

Il **cascade buffer** è la componente che genera i segnali di controllo che regolano il funzionamento in cascata di più dispositivi 8259. In caso di controllori dell'interrupt in cascata, i segnali *CAS* del master sono collegati a quelli degli slave. Tali segnali sono di output per il dispositivo master e di input per i dispositivi slave. Sono utilizzati dal master, in seguito alla ricezione del segnale di interrupt acknowledge dal processore, per abilitare lo slave con una richiesta attiva correntemente a più alta priorità. Tale dispositivo dovrà rispondere al segnale di *INTA* della CPU ponendo sul data bus l'identificativo del vettore a priorità massima tra i suoi correntemente attivi.

Il segnale $\overline{SP} / \overline{EN}$ presenta due diverse funzioni a seconda che l'8259 operi in *buffered mode* o in *non-buffered mode*. Nel primo caso è un segnale di output attivo basso utilizzato per disabilitare i data buffer degli altri dispositivi connessi al bus durante i trasferimenti dati tra il controllore 8259 e la CPU. Nel secondo caso è un segnale di input attivo basso che indica il ruolo del dispositivo:

- Se $\overline{SP} = 1$ il dispositivo opera da master;
- Se $\overline{SP} = 0$ il dispositivo opera da slave;

Operando in *buffered mode*, il ruolo del dispositivo viene invece comunicato allo stesso via software.

5.3.2 Sequenza di interruzione

Per poter operare il controllore delle interruzioni 8259 deve essere opportunamente inizializzato tramite parole di controllo inviate dalla CPU sugli 8 bit bassi del data bus. Una volta che il controllore è inizializzato, si pone in attesa di richieste di interruzione.

Un segnale di richiesta di interrupt ricevuto dall'8259 su una delle 8 linee di interruzione determina il settaggio del corrispondente bit del registro IRR. Nel caso in cui tale richiesta non venga mascherata dal registro IMR, viene rilevata dalla logica di controllo che si occupa di avvisare il processore attivando il segnale di *INT*.

La CPU, rileva la richiesta di interruzione hardware esterna se il flag IF risulta attivato. In tal caso il processore, quando disponibile a servire la richiesta di interruzione, invia all'8259 un primo segnale di *INTA*. Tale segnale viene rilevato dalla logica di controllo che comanda il priority resolver di selezionare, secondo il particolare metodo di assegnazione delle priorità determinato in fase di programmazione, la richiesta di interruzione attiva a priorità più alta. Per fare ciò il priority resolver interpella tre registri: il registro IRR per conoscere le richieste di interruzione attive, il registro ISR per conoscere la richiesta di interruzione correntemente in servizio e il registro IMR per conoscere quali richieste di interruzione devono essere mascherate. La richiesta di interruzione selezionata è quella che dovrà essere comunicata al processore per essere servita.

Il priority resolver si occupa di settare il bit corrispondente alla richiesta selezionata nel registro ISR, per tenere traccia di quale richiesta si è passati a servire, e di resettare quello nel registro IRR per indicare che tale richiesta è stata accettata.

La CPU invia un secondo impulso sul segnale di *INTA*, che viene rilevato dalla logica di controllo. Questa, intergendo con la logica di lettura/scrittura, effettua il trasferimento sugli 8 bit bassi del data bus dell'identificativo di vettore relativo alla richiesta selezionata.

Il ciclo di interrupt è concluso con un eventuale resettaggio del bit abilitato del registro ISR. Ciò avviene se l'8259 è stato opportunamente inizializzato per operare in modo **AEOI** (*Automatic End of Interrupt*). In tale modo di funzionamento, il bit di ISR viene resettato subito dopo avere inviato alla CPU il numero identificativo della richiesta di interruzione da servire. In tal caso risulta pertanto possibile che una interruzione a priorità minore interrompa la routine di servizio di una a priorità maggiore.

Alternativamente il controllore può essere inizializzato per operare in modo **EOI** (*End of Interrupt*) in cui il ciclo di interrupt viene terminato, e pertanto il bit di ISR viene resettato, inviando al termine della routine di servizio (appena prima della istruzione IRET) un comando esplicito di fine interruzione. Tale comando viene inviato per mezzo di una opportuna parola di controllo. In tale modalità di funzionamento, il controllore degli interrupt permette di evitare che richieste di interruzione a priorità minore interrompano la routine di servizio di una a priorità maggiore.

Il flusso di esecuzione è trasferito alla routine di servizio della interruzione selezionata.

Dalla sequenza di funzionamento illustrata consegue che una particolare richiesta di interruzione potrà essere servita solo se le seguenti condizioni sono verificate:

- Il flag IF risulta disabilitato;
- Tale richiesta di interruzione non risulta mascherata dal registro IMR;
- Tale richiesta di interruzione non viene prevaricata da una richiesta di interruzione a priorità maggiore;

5.3.3 Programmazione del controllore 8259

Il controllore 8259 viene programmato dal processore attraverso due tipi di parole di controllo su 8 bit:

- *Initialization Command Words (ICWs)*: permettono di inizializzare il dispositivo, devono pertanto essere inviate all'8259 in un determinato ordine prima che questo possa iniziare a funzionare;
- *Operation Command Words (OCWs)*: definiscono particolari comandi che il processore può inviare all'8259 singolarmente e in un qualunque momento durante il suo funzionamento;

Initialization Command Words

Le ICWs vengono inviate all'8259 dal processore in fase di inizializzazione per definire il funzionamento basilare del dispositivo. Sono definiti in tutto quattro tipi di ICW, due obbligatorie e due opzionali. Durante la fase di inizializzazione il processore invia pertanto al controllore da due a quattro delle ICW disponibili, secondo un ordine definito.

Una descrizione sommaria delle quattro parole di inizializzazione, nell'ordine in cui queste possono essere inviate, è la seguente:

- **ICW₁**: programma le funzioni di base del controllore 8259;
- **ICW₂**: programma nel dispositivo il numero di vettore iniziale a partire dal quale sono memorizzate, in ordine all'interno della IVT, i puntatori alle ISR relative alle richieste di interrupt da esso gestite;
- **ICW₃**: utilizzata solo quando più controllori 8259 sono collegati in cascata; indica, al dispositivo master, su quale linee di interruzione sono collegati i dispositivi slave e, ai dispositivi slave, su quale linea del dispositivo master sono collegati;
- **ICW₄**: può essere opzionalmente inviata per specificare alcune funzionalità aggiuntive come il meccanismo di priorità utilizzato, la modalità di definizione del termine del ciclo di interruzione, ecc;

La ICW_1 apre la sequenza di inizializzazione dell'8259, tale parola di controllo è immediatamente riconoscibile perchè caratterizzata dai seguenti valori dei segnali $A0$ e $D4$:

$$A0=0, D4=1$$

La ICW_1 determina l'inizio della sequenza di inizializzazione e causa il reset da parte del dispositivo del registro IMR. A seguire il processore invia da una a tre altre parole di controllo, riconosciute dal dispositivo perchè caratterizzate dal segnale $A0=1$.

La sequenza di inizializzazione procede secondo il flow-chart riportato nella Fig. 30.

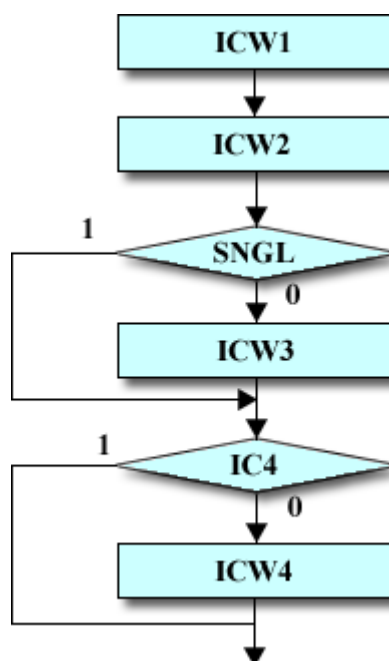


Fig. 30: Procedimento di inizializzazione del controllore degli interrupt 8259.

In particolare, due bit della prima parola di inizializzazione (i bit $SNGL$ e $IC4$) definiscono se rispettivamente le parole ICW_3 e ICW_4 verranno inviate. Il dispositivo conosce pertanto, in seguito alla ricezione della prima parola di inizializzazione, quante altre ICW riceverà successivamente.

Struttura della ICW_1

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	X	X	X	1	LTIM	X	SGNL	IC4

Fig. 31: Struttura della prima parola di inizializzazione.

La ICW_1 permette di definire il funzionamento di base del dispositivo controllore degli interrupt. I significati associati ai bit della ICW_1 sono i seguenti:

- **LTIM** (*Level Triggered Mode*): definisce la modalità di rilevamento delle richieste di interrupt inviate all'8259 da parte dei dispositivi ad esso collegati, per mezzo dei segnali IR0-IR7. Si distinguono due modalità di rilevazione di tali richieste:
 - *Level Triggered*: la richiesta di interruzione viene rilevata sul livello logico alto del segnale IR, attiva se LTIM=1;
 - *Edge Triggered*: la richiesta di interruzione viene rilevata sul fronte di salita del segnale IR, attiva se LTIM=0;
- **SNGL** (*Single Mode*): definisce se la modalità di funzionamento prevede un controllore singolo, attiva se SNGL=1, o più controllori in cascata, attiva se SNGL=0. In base al valore di tale bit il controllore si aspetterà o meno, l'invio da parte della CPU della ICW_3 ;
- **IC4** (*ICW4 Needed*): indica se nella fase di inizializzazione corrente il programma ha intenzione di inviare la quarta parola di inizializzazione, IC4=1, o meno, IC4=0;
- **X**: nessun significato associato;

Alla ricezione della ICW_1 il controllore rileva l'inizio di una nuova fase di controllo, inizializza allora ai rispettivi valori di default i propri registri di controllo interni e resetta il registro IMR, prima di processare la prima parola di controllo. Tra tali impostazioni di default si evidenziano:

- Priorità delle richieste di interruzione statica, con IR7 richiesta a priorità più bassa;
- Modalità non-buffered;

- Modalità end of interrupt;
- Processor mode = 8085
- Fully nested mode;
- Edge trigger mode;
- Single mode;
- No special mask mode;
- Status read IRR;
- No Polling mode;

Struttura della ICW₂

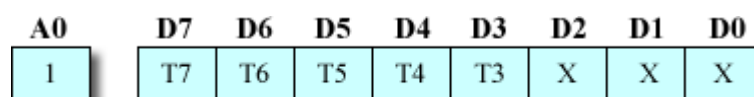


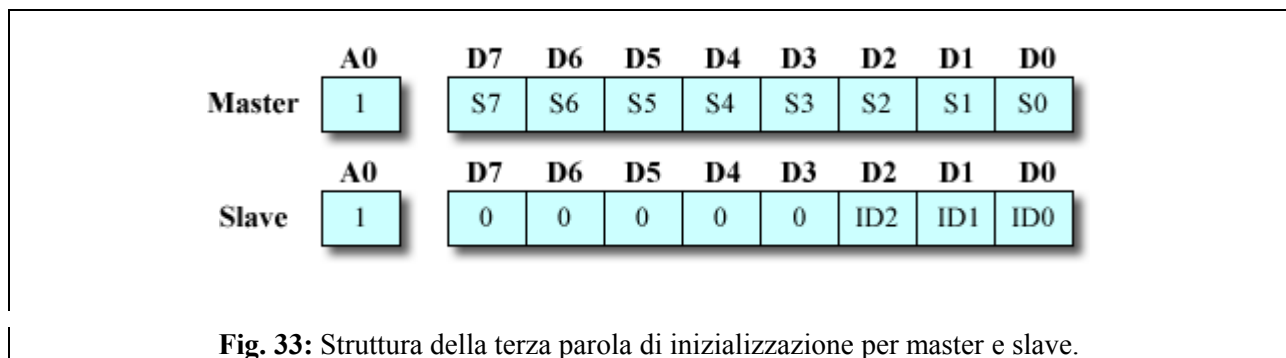
Fig. 32: Struttura della seconda parola di inizializzazione.

La ICW₂ permette all'8259 di definire il valore iniziale n del vettore, all'interno della IVT, a partire dal quale il sistema ha caricato i puntatori alle routine di servizio delle linee di interruzione gestite da controllore. Tale valore è definito dal BIOS o dal loader del sistema operativo in seguito al caricamento in memoria delle routine di servizio dell'interrupt associate ai periferici e della memorizzazione nella IVT dei puntatori ad esse relativi.

Nella seconda parola di inizializzazione, i 3 bit bassi non sono assegnati ad alcun significato particolare, mentre i 5 bit alti sono utilizzati per definire l'identificativo del vettore di partenza n , all'interno della IVT. Tale valore viene ottenuto affiancando ai cinque bit T7-T3 tre bit a zero.

Il vettore di partenza è pertanto identificato sempre da un numero multiplo di 8, a partire dal quale sono definiti gli 8 vettori di interruzione per i dispositivi gestiti dal controllore. Il valore n rappresenta pertanto l'identificativo di vettore associato alla richiesta di interruzione IR0.

Struttura della ICW₃



La ICW₃ viene inviata dalla CPU nel caso di funzionamento in cascata di più controllori 8259. In tal caso ogni dispositivo verrà programmato, oltre che con le prime due parole di inizializzazione obbligatorie, con una terza parola di controllo che assume una forma diversa a seconda del ruolo svolto dal particolare 8259 all'interno della cascata.

Per il dispositivo master la ICW₃ indica su quale delle 8 linee di richiesta di interruzione sono collegati gli slave. Essa consiste in 8 bit di flag, a ciascuno dei quali corrisponde una delle 8 linee IR del master. Ognuno di tali bit specifica se, sulla linea ad esso associato, è collegato un controllore 8259 slave (bit a 1) oppure un generico dispositivo periferico (bit a 0).

Per ogni dispositivo slave la ICW₃ indica su quale linea IR del master questo è collegato. In tal caso la terza parola di inizializzazione presenta i 5 bit alti posti a 0, mentre sui 3 bit bassi viene codificato il numero di linea IR del master a cui lo slave è collegato.

Struttura della ICW₄

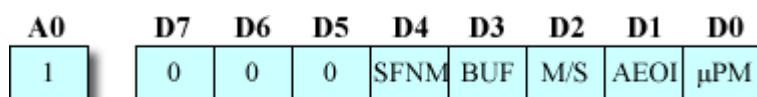


Fig. 34: Struttura della quarta parola di inizializzazione.

La ICW₄ permette alla CPU di configurare ulteriori impostazioni di funzionamento per il controllore 8259. Degli 8 bit che costituiscono la quarta parola di inizializzazione, i tre bit alti sono fissi a 0, mentre i cinque bit bassi vengono utilizzati per definire particolari aspetti di funzionamento. I significati associati ai cinque bit della ICW₄ sono i seguenti:

- **SFNM** (*Special Fully Nested Mode*): permette di attivare, se posto a 1, il modo di funzionamento **special fully nested**. Tale modalità permette di conservare la priorità di interruzione tra gli slave, nel funzionamento di più controllori in casata;
- **BUF** e **M/S** (*Buffered Mode* e *Master/Slave*): il bit BUF permette di definire, per il controllore 8259, il metodo di funzionamento *buffered* (BUF=1) o *non-buffered* (BUF=0). Nel caso in cui il modo sia *buffered*, il bit M/S permette di determinare se il dispositivo in questione operi da dispositivo master, M/S posto a 1, o da dispositivo slave, M/S posto a 0. Nel caso di funzionamento *non-buffered*, il ruolo dei dispositivi è determinato dal valore logico fissato sul piedino $\overline{SP} / \overline{EN}$. Quando invece il controllore opera in modalità *buffered*, il segnale $\overline{SP} / \overline{EN}$ viene utilizzato per l'abilitazione dei buffer dati dei dispositivi connessi al bus.

La tabella seguente indica le modalità di funzionamento in funzione dei valori di BUF e M/S:

Modalità	BUF	M/S
Non-buffered Mode	0	X
Buffered Mode Slave	1	0
Buffered Mode Master	1	1

- **AEOI** (*Automatic End of Interrupt*): permette di attivare, per il controllore 8259, la modalità di fine automatica del ciclo di interruzione (AEOI=1). In tale modalità, il bit del registro ISR corrispondente alla richiesta di interruzione correntemente in servizio, viene resettato subito dopo aver comunicato alla CPU l'identificativo di vettore da servire. Ciò permette alle richieste di qualunque livello di priorità di interrompere la routine di servizio in corso di esecuzione, dal momento che il registro ISR non mantiene più traccia di quale interruzione risulta correntemente in servizio.
- **μPM** (*Microprocessor Mode*): permette di indicare al controllore 8259 il tipo di processore a cui risulta collegato. Se μPM=1, il controllore opererà nella maniera richiesta dai processori della famiglia 8086. Se μPM=0, il controllore opererà nella maniera richiesta da i processori basati su 8085. Dal momento che le impostazioni di default per gli aspetti operativi definiti con la quarta parola di inizializzazione corrispondono alle impostazioni ottenibili programmando il controllore con una ICW₄ tutta a zero, se non specificato il controllore opera come se fosse collegato ad un processore 8085. Per utilizzare un controllore degli interrupt 8259 su un sistema basato sull'8086, la programmazione della ICW₄ è obbligatoria e deve prevedere un valore logico alto per il bit meno significativo.

Esempio (*Inizializzane di due dispositivi interrupt controller 8237*)

Due dispositivi 8237 sono posti in cascata in modo che il dispositivo slave sia collegato alla linea di richiesta dell'interrupt IR_2 del dispositivo master. Supponendo che gli indirizzi di I/O di tali dispositivi siano 0A0H per il master e 020H per lo slave, programmare tali dispositivi in modo che:

- Per il dispositivo master:
 - Il numero di vettore iniziale all'interno della IVT sia 08H;
 - Lo *Special Fully Nested Mode* sia disattivato;
 - Il *Buffered Mode* sia disattivato;
 - L'*Automatic End of Interrupt* sia disattivato;
- Per il dispositivo slave:
 - Il numero di vettore iniziale all'interno della IVT sia 70H;
 - Lo *Special Fully Nested Mode* sia disattivato;
 - Il *Buffered Mode* sia disattivato;
 - L'*Automatic End of Interrupt* sia disattivato;

```
ICW1_MASTER      EQU      11H      ;Cascade mode, ICW4 needed
ICW2_MASTER      EQU      08H      ;Cascade mode, ICW4 needed
ICW3_MASTER      EQU      04H      ;Cascade mode, ICW4 needed
ICW4_MASTER      EQU      01H      ;Cascade mode, ICW4 needed
ICW1_SLAVE       EQU      11H      ;Cascade mode, ICW4 needed
ICW2_SLAVE       EQU      70H      ;Cascade mode, ICW4 needed
ICW3_SLAVE       EQU      02H      ;Cascade mode, ICW4 needed
ICW4_SLAVE       EQU      01H      ;Cascade mode, ICW4 needed
MASTER           EQU      0A0H     ;Indirizzo di IO dell'8237 master
SLAVE            EQU      020H     ;Indirizzo di IO dell'8237 slave

.MODEL small

.STACK

.DATA
```

```
.CODE

.STARTUP

;Programmazione della prima parola di controllo

MOV    AL, ICW1_MASTER

OUT    MASTER, AL

MOV    AL, ICW1_SLAVE

OUT    SLAVE, AL

;Programmazione della seconda parola di controllo

MOV    AL, ICW2_MASTER

OUT    MASTER+1, AL

MOV    AL, ICW2_SLAVE

OUT    SLAVE+1, AL

;Programmazione della terza parola di controllo

MOV    AL, ICW3_MASTER

OUT    MASTER+1, AL

MOV    AL, ICW3_SLAVE

OUT    SLAVE+1, AL

;Programmazione della quarta parola di controllo

MOV    AL, ICW4_MASTER

OUT    MASTER+1, AL

MOV    AL, ICW4_SLAVE

OUT    SLAVE+1, AL

.EXIT

END
```

Operation Command Words

Le OCWs possono essere inviate all'8259 dal processore in qualunque momento, e in qualunque ordine, durante l'esecuzione di un programma e sono utilizzate per dirigere le operazioni effettuate dal controllore una volta che questo è stato posto in funzionamento dopo essere stato inizializzato.

È possibile specificare per il controllore 8259 tre tipi di OCW, definite sommariamente come segue:

- **OCW₁**: permette al processore di caricare una maschera di bit nel registro IMR;
- **OCW₂**: permette di gestire i modi di rotazione delle priorità e di end of interrupt;
- **OCW₃**: permette al processore di abilitare lo special mask mode, di leggere lo stato dei registri interni dell'8259 o di abilitare il funzionamento in polling del controllore;

La prima parola di operazione, dovendo definire una maschera qualunque su 8 bit, per non essere confusa con la ICW₁ viene riconosciuta dal controllore quando, in un qualunque momento durante il suo funzionamento al di fuori della fase di inizializzazione, riceve una richiesta di scrittura accompagnata da un valore logico alto sul piedino A0.

Tutte le altre OCW vengono invece distinte dalle ICW diverse dalla ICW₁ perchè indirizzate mediante un valore logico basso del piedino A0, mentre vengono distinte dalla ICW₁ perchè presentano tutti valori logici bassi sul segnale di dato D₄.

Struttura della OCW₁

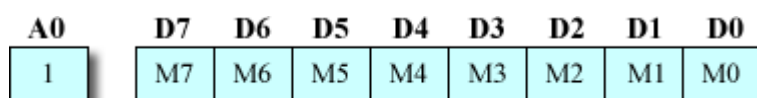


Fig. 35: Struttura della prima parola di comando.

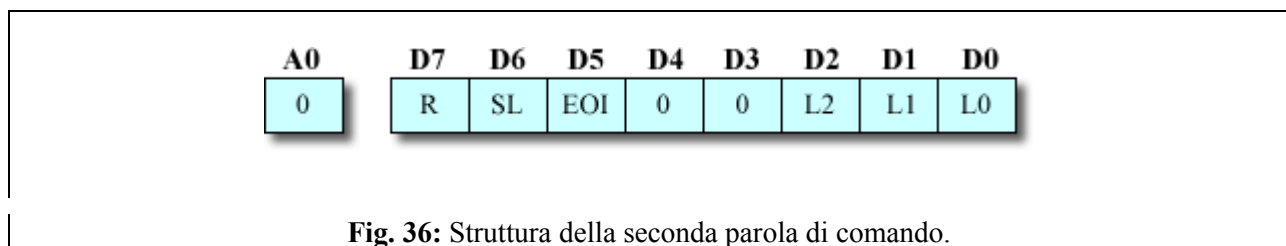
La OCW₁ permette al processore di definire una maschera di bit con la quale disabilitare selettivamente determinate linee di richiesta di interrupt. La maschera definita dagli otto bit M₇-M₀ della OCW₁ viene memorizzata nel registro IMR ed utilizzata dal priority resolver per sapere su quali delle linee IR devono essere rilevate le richieste di interruzione.

Ponendo a 1 il bit M_i di tale parola di operazione, viene posto a uno l' i -esimo bit del registro IMR e di conseguenza viene mascherata la linea di interruzione IR_i .

Dal momento che il contenuto del registro IMR, in seguito all'inizializzazione del dispositivo, non è prevedibile, il contenuto di tale registro deve essere opportunamente programmato mediante la parola di controllo OCW_1 al termine della fase di inizializzazione.

Il processore può effettuare una operazione di lettura sul controllore all'indirizzo $A0=1$ per leggere, in qualunque momento successivo all'inizializzazione del dispositivo, il contenuto del registro IMR.

Struttura della OCW_2



La OCW_2 permette al processore di svolgere i due seguenti compiti:

- Comunicare al controllore degli interrupt la terminazione del ciclo di interruzione, nel caso di modalità EOI;
- Comunicare al controllore degli interrupt se e come variare le priorità associate alle linee IR;

In tale parola di controllo i 3 bit alti sono utilizzati per determinare la particolare operazione richiesta al controllore delle interruzioni, i 3 bit bassi L2, L1, L0 permettono di identificare la particolare linea di interruzione eventualmente oggetto dell'operazione, i rimanenti bit sono posti a 0.

I bit utilizzati per definire l'operazione richiesta all'8259 sono i seguenti:

- **R (Rotate)**: se posto a uno indica che l'operazione richiesta prevede la rotazione di priorità per la linea oggetto della operazione;
- **SL (Selected)**: se posto a uno indica che la linea oggetto dell'operazione è specificata dai tre bit bassi della parola di controllo;
- **EOI (End of Interrupt)**: se posto a uno indica la terminazione del ciclo di interrupt per la linea di interruzione oggetto dell'operazione e causa pertanto il reset del corrispondente bit del registro ISR;

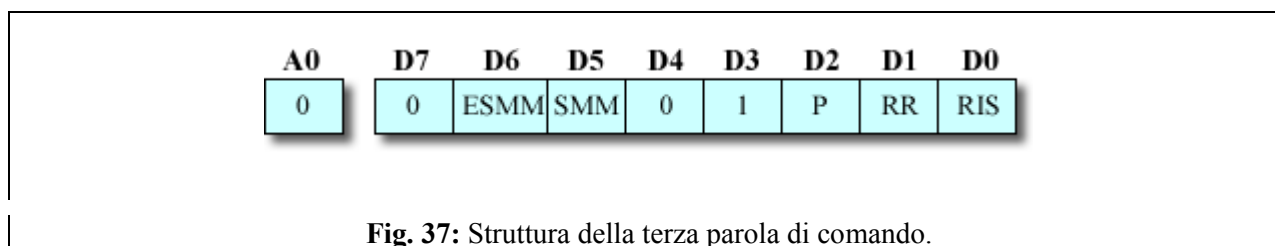
Le operazioni che possono essere richieste al controllore dell'interrupt per mezzo della OCW₂ sono riportate nella tabella seguente:

R	SL	EOI	Nome dell'operazione	Descrizione
0	0	1	Non Specific end of interrupt	Permette al processore di comunicare all'8259 il termine del ciclo di interrupt in servizio. Viene inviata al termine della routine di servizio corrente. Il priority resolver determina quale richiesta di interruzione è attualmente in servizio e resetta il bit corrispondente del registro ISR.
0	1	1	Specific end of interrupt	Permette al processore di resettare lo stato di servizio di una specifica linea di interrupt, definita dai bit L2-L0.
1	0	1	Rotate on non specific end of interrupt	Permette al processore di comunicare all'8259 il termine del ciclo di interrupt in servizio e di richiedere la rotazione del livello di priorità per la corrispondente richiesta di

				<p>interruzione. Viene inviata al termine della routine di servizio corrente. Il priority resolver determina quale richiesta di interruzione è attualmente in servizio e resetta il bit corrispondente del registro ISR. Infine esegue la rotazione di priorità sulla richiesta di interruzione appena terminata ponendola in fondo alla scala di priorità.</p>
1	0	0	Rotate on automatic end of interrupt (ON)	<p>Permette di attivare la modalità di terminazione automatica (AEIO) con rotazione delle priorità. Il bit di ISR relativo alla richiesta correntemente in servizio viene resettato immediatamente dopo aver inviato il numero identificativo del vettore sul data bus e la priorità relativa a tale richiesta di interrupt viene posta in fondo alla scala di priorità.</p>
0	0	0	Rotate on automatic end of interrupt (OFF)	<p>Permette di disattivare la modalità di terminazione automatica (AEIO) con rotazione delle priorità.</p>
1	1	1	Rotate on specific end of interrupt	<p>Permette al processore di resettare lo stato di servizio di una specifica linea di interrupt, definita dai bit L2-L0 e di ruotarne poi la priorità. La richiesta di interruzione specificata viene posta in fondo alla scala di priorità.</p>
1	1	0	Set priority	<p>Permette di ruotare la priorità associata ad</p>

				una specifica linea di interruzione, definita dal valore dei bit L2-L0. La richiesta di interruzione specificata viene posta in fondo alla scala di priorità.
0	1	0	No operation	Nessuna operazione.

Struttura della OCW₃



La OCW₃ permette al processore di effettuare una delle seguenti operazioni:

- Richiedere la lettura di uno dei registri interni IRR o ISR del controllore;
- Abilitare o disabilitare lo special mask mode;
- Abilitare o disabilitare il metodo di funzionamento in polling del controllore;

Oltre ai bit posti a valori fissi riportati in figura, la OCW₃ presenta cinque bit di flag a cui sono associati i seguenti significati:

- **RR e RIS** (*Read Register e Read In Service*): permettono di richiedere la lettura (se RR=1 e RIS=0) del registro IRR o del registro ISR (se RIS=1 e RR=1). La tabella seguente indica le azioni richieste al controllore in funzione dei valori di RR e RIS:

Azione	RR	RIS
Nessuna azione	0	X
Lettura del registro IRR	1	0
Lettura del registro ISR	1	1

La parola di stato contenuta nel registro selezionato verrà resa disponibile al processore, dalla logica di lettura/scrittura, sul data bus per il successivo ciclo di lettura all'indirizzo avente A0=0.

- **ESMM e SMM** (*Enable Special Mask Mode* e *Special Mask Mode*): permettono di abilitare o disabilitare una modalità di funzionamento speciale per il registro IMR chiamata special mask mode. Tali bit operano secondo le configurazioni:

Operazione	ESMM	SMM
Nessuna azione	0	X
Disabilita lo special mask mode	1	0
Abilita lo special mask mode	1	1

- **P** (*Polling Mode*): permette di abilitare, se P=1, o disabilitare, se P=0, il metodo di gestione in polling del controllore 8259;

5.3.4 Modalità operative ed impostazioni

Diverse modalità operative sono configurabili per il controllore 8259 per mezzo delle parole di inizializzazione e di comando, tali modalità definiscono come il controllore opera riguardo ai seguenti aspetti:

- *Gestione dei livelli di priorità*: il controllore può assegnare alle linee di interruzione livelli di priorità fissi o permettere la modifica di tali livelli in corso di funzionamento.
- *Terminazione del ciclo di interrupt*: il controllore può considerare il ciclo di interruzione terminato subito dopo aver comunicato l'identificativo di vettore da servire alla CPU oppure al termine dell'esecuzione della routine di servizio. Nel primo caso si permette a interruzioni a priorità minore di interrompere le routine di servizio di quelle a priorità maggiore, nel secondo ciò non è invece possibile.

- *Utilizzo del registro maschera*: il controllore può utilizzare il registro di maschera per disabilitare selettivamente alcune linee di interrupt o per definire quali richieste di interruzione possono interrompere, indipendentemente dalle priorità, una particolare routine di servizio.
- *Metodo di interpellazione*: il controllore può essere impostato per comunicare con il processore in modo interrupt o in modo polling.

Diverse impostazioni possono inoltre essere definite attraverso parole di comando o inizializzazione per specificare aspetti quali il metodo di riconoscimento delle richieste di interruzione, le modalità di lettura dei registri di stato, il funzionamento di più controllori in cascata, il funzionamento buffered o non buffered ecc.

Terminazione del ciclo di interrupt

La terminazione del ciclo di interruzione deve essere comunicata al controllore perchè questo possa resettare il bit di ISR relativo alla interruzione terminata e pertanto riabilitare la corrispondente linea di richiesta di interrupt.

La notifica di terminazione di un'interruzione può avvenire secondo due modalità, a seconda del valore utilizzato per il flag AEOI della ICW4 in fase di inizializzazione:

- *Automatic End of Interrupt* (AEOI=1): in tal caso, l'interruzione termina subito dopo aver comunicato al processore il numero di vettore da servire. Il bit del registro ISR relativo all'istruzione appena servita viene resettato automaticamente dopo il fronte di salita del secondo segnale di INTA.
- *End of Interrupt* (AEOI=0): in tal caso, il termine dell'interruzione viene comunicato esplicitamente dal processore appena prima di ritornare dalla routine di servizio. Il bit del

registro ISR relativo all'interruzione rimane attivo fino a quando il controllore non riceve una opportuna parola di controllo che specifica il termine dell'interruzione ad esso associata.

Quando si opera in AEIOI il bit da resettare nel registro ISR è quello relativo alla richiesta di interruzione appena servita, in modalità EOI invece, per l'identificare l'interruzione terminata si distinguono due casi, ad ognuno dei quali corrispondono diverse parole di controllo di EOI:

- *Non specific EOI*: utilizzata nel caso in cui la modalità di funzionamento adottata mantenga la struttura di ordinamento di priorità del fully nested mode. In tali condizioni il controllore è in grado di identificare, a seguito della ricezione del comando di EOI, quale tra le richieste in servizio è stata l'ultima ad essere servita, identificando la richiesta in servizio con priorità maggiore.
- *Specific EOI*: utilizzata per tutte quelle modalità di funzionamento che non preservano la struttura di ordinamento di priorità del fully nested mode. In tali casi infatti, il controllore non ha modo di riconoscere quale è l'ultimo livello di interruzione abilitato. L'interruzione terminata deve pertanto essere esplicitamente comunicata dal processore. La struttura di ordinamento di priorità del fully nested mode non viene preservata ad esempio quando viene utilizzato lo special mask mode;

Gestione dei livelli di priorità

Il controllore 8259 permette di gestire i livelli di priorità associati alle linee di richiesta dell'interrupt principalmente in due modi:

- *Statico o fully nested*: alle linee di interruzione sono assegnati, in ordine, dei livelli di priorità che non possono mutare in corso di funzionamento del controllore;
- *Rotante*: alle linee di interruzione sono assegnati, in ordine, dei livelli di priorità che possono essere ruotati in corso di funzionamento del controllore;

L'ordinamento di priorità viene utilizzato per decidere:

- Quale richiesta di interruzione servire nel caso di più richieste ricevute simultaneamente;
- Quali richieste di interruzione possono interrompere richieste correntemente in servizio;

Fully nested mode

La modalità **fully nested mode** è la modalità operativa configurata di default per il controllore e prevede un sistema fisso di assegnazione delle priorità ai dispositivi collegati alle linee IR.

Tali linee di richiesta di interruzione sono ordinate per livelli di priorità crescenti dalla IR7 alla IR0. Ne consegue pertanto che il dispositivo collegato alla linea IR7 riceverà la priorità di interruzione più bassa, mentre quello collegato alla linea IR0 godrà della più alta priorità di interrupt.

In tale modalità, in seguito all'invio da parte della CPU del primo segnale di INTA, il priority resolver seleziona la richiesta di interruzione a priorità maggiore semplicemente identificando la linea di interruzione attiva e non mascherata caratterizzata da un identificativo più basso. Il corrispondente bit nel registro ISR viene settato e, in seguito alla ricezione del secondo segnale di acknowledge da parte della CPU, verrà inviato sul data bus l'indice del vettore delle interruzioni corrispondente.

Fintanto che il bit di ISR rimane settato tutte le richieste di interrupt a priorità più bassa vengono ignorate.

La riabilitazione di tali richieste può avvenire in due modi a seconda della modalità di terminazione del ciclo di interrupt selezionata:

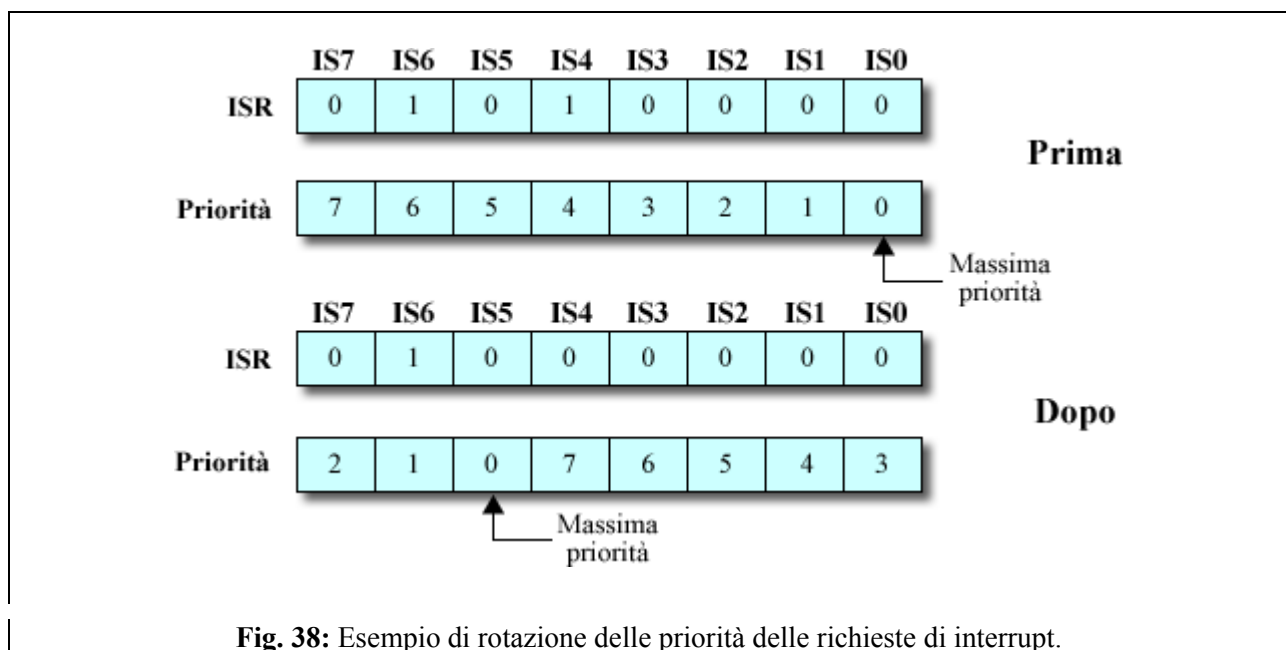
- *Automatic End of Interrupt* (AEIOI=1): in tal caso, il bit precedentemente abilitato del registro ISR viene resettato automaticamente dopo il fronte di salita del secondo segnale di INTA, ad indicare il termine del ciclo di interruzione.

- *End of Interrupt* (AEOI=0): in tal caso, il bit precedentemente abilitato del registro ISR rimane attivo fino alla ricezione del comando di non specific EOI.

Rotazione delle priorità

In molte applicazioni può essere richiesto che ai dispositivi periferici venga assegnata la stessa priorità. In questi casi non è possibile operare con livelli di priorità statici, occorre invece utilizzare livelli di priorità rotanti, cioè livelli di priorità la cui assegnazione alle linee IR viene cambiata al termine di ogni ciclo di interruzione.

Con *rotazione di una linea di interruzione* si intende una operazione di modifica della scala delle priorità che pone la linea oggetto della rotazione in fondo alla scala. Ciò è ottenuto ruotando i valori di priorità associati alle linee di interruzione fino a far coincidere il livello di priorità più basso con la linea oggetto della rotazione. L'ordinamento dei livelli di priorità viene preservato.



Il controllore 8259 permette di operare con livelli di priorità rotanti in due modi:

- **Rotazione automatica delle priorità:** in tale modalità il controllore si occupa, al termine del ciclo di interrupt, di eseguire la rotazione di priorità per la linea di interruzione corrispondente al dispositivo appena servito. Si distinguono due sottocasi:
 - Nel caso in cui il controllore operi in modalità AEOI, la modalità di rotazione automatica delle priorità deve essere attivata esplicitamente programmando il controllore con una opportuna parola OCW_2 ($R=1, SL=0, EOI=0$). In tal caso la priorità del dispositivo appena servito viene ruotata immediatamente in seguito alla ricezione del secondo segnale di INTA, dopo aver resettato il corrispondente bit di ISR. La modalità di rotazione automatica delle priorità in AEOI, può essere disattivata attraverso la OCW_2 ($R=0, SL=0, EOI=0$) che impone la normale priorità fissa in modo AEOI;
 - Nel caso in cui il controllore operi in modalità EOI, la rotazione di priorità è comunicata direttamente dalla CPU insieme alla terminazione del ciclo di interruzione mediante la OCW_2 ($R=1, SL=0, EOI=1$). In tal caso la priorità del dispositivo appena servito viene ruotata immediatamente in seguito alla ricezione di tale parola di controllo, dopo aver resettato il corrispondente bit di ISR;
- **Rotazione specifica delle priorità:** il programmatore può fare ruotare in maniera specifica i livelli di priorità attraverso una opportuna parola OCW_2 . In tal caso viene eseguita la rotazione del livello di interruzione specificato nei tre bit bassi L2-L0 di tale parola. Si distinguono due casi:
 - *Rotazione specifica con terminazione dell'interrupt* ($R=1, SL=1, EOI=1$): utilizzata in modalità EOI per ottenere una rotazione specifica delle priorità e contemporaneamente notificare la fine del ciclo di un interrupt. Il bit di ISR corrispondente al livello di interruzione specificato dai bit L2-L0 viene resettato e la priorità associata a tale livello viene poi fatta ruotare;

- *Rotazione specifica senza terminazione dell'interrupt* (R=1, SL=1, EOI=0): permette di eseguire la rotazione di priorità su una specifica linea di interruzione, definita dai bit L2-L0;

Special mask mode

Lo **special mask mode** definisce una modalità di utilizzo alternativa a quella tradizionale per il registro IMR. Tale modalità è attivata per mezzo della parola di controllo OCW₃, ponendo a uno i bit ESMM e SMM.

Operando in special mask mode, i bit a uno del registro IMR svolgono due funzioni:

- Disabilitano la linea di richiesta dell'interrupt ad essi corrispondente;
- Abilitano tutte le altre linee di richiesta dell'interrupt;

Viene utilizzata durante la routine di servizio di una interruzione per permettere ad altre richieste di interruzione, con livelli di priorità sia maggiori che minori, di interrompere la richiesta di interruzione correntemente in servizio. Caricando opportunamente il registro IMR con una OCW₁ e poi abilitando lo special mask mode con una OCW₃ si permette a tutte le richieste di interruzione diverse da quella corrente e poste a 0 in IMR di interrompere la richiesta correntemente in servizio.

L'attivazione di tale modalità di funzionamento sfalsa la struttura di ordinamento di priorità del fully nested mode. Ne consegue che, utilizzando tale modalità in congiunzione con la modalità EOI, i comandi utilizzati per notificare la terminazione delle interruzioni dovranno essere specifici.

Prima di uscire dalla routine di servizio che ha attivato lo special mask mode, questo deve essere disabilitato, sempre per mezzo della parola di controllo OCW₃.

Poll command

Il controllore 8259 può anche essere gestito in polling dal processore. Per utilizzare il metodo di gestione in polling, questo deve essere abilitato inviando una OCW₃ con l'apposito bit P posto a 1.

In modo polling, l'8259 e la CPU non si scambiano la tradizionale sequenza di segnali INT e INTA. Al contrario il controllore rimane in attesa di una operazione di lettura da parte della CPU, ed interpreta tale operazione come acknowledge delle interruzioni. Non appena una operazione di lettura viene richiesta al controllore degli interrupt, questo pone sul data bus una parola, detta *poll word*, che permette di comunicare al processore se è presente o meno una richiesta di interruzione e, nel caso, qual'è la linea di interruzione attiva a priorità più alta, che deve essere quindi servita.

La struttura della poll word è descritta nella Fig. 39.

D7	D6	D5	D4	D3	D2	D1	D0
I	X	X	X	X	W2	W1	W0

Fig. 39: Struttura della poll word.

I bit della poll word sono associati ai seguenti significati:

- **I** (*Interrupt*): indica la presenza o l'assenza di una richiesta di interruzione, rispettivamente assunto valore uno o zero;
- **W2-W0** (*Waiting Interrupt*): codificano eventualmente l'identificativo di linea della IR a priorità più elevata;
- **X**: sono inutilizzati;

Lettura dei registri interni

È possibile leggere lo stato dei registri interni IRR, ISR e IMR del controllore 8259. Si definisce **parola di stato**, il contenuto su 8 bit dei registri interni posto sul databus per poter essere letto dal processore.

Per effettuare la lettura dei registri IRR e ISR deve essere precedentemente inviata al controllore degli interrupt la parola di controllo OCW₃ con opportuni valori dei flag RR e RIS. La parola di stato richiesta verrà resa disponibile al processore, dalla logica di lettura/scrittura, sul data bus per il successivo ciclo di lettura all'indirizzo avente A0=0.

Non è necessario inviare una OCW₃ prima di ogni lettura di registro. Dal momento che il controllore memorizza l'ultima OCW₃ ricevuta, non è necessario inviarne un'altra se si vuole leggere sempre lo stesso registro. Per default la lettura è impostata sul registro IRR.

La lettura del registro IMR, invece, non deve essere preceduta da alcuna programmazione, ma può essere effettuata in qualunque momento dopo l'inizializzazione eseguendo un ciclo di lettura all'indirizzo avente A0=1.

Interrupt sensibile ai fronti o ai livelli

A seconda del valore impostato in fase di inizializzazione per il flag LTIM, il controllore degli interrupt può rilevare le richieste di interruzione in due modi:

- *Level Triggered Mode* (LTIM=1): le richieste di interruzione su una linea IR sono riconosciute dal livello logico alto su tale linea. La richiesta di interruzione va pertanto rimossa dalla linea prima del termine del ciclo di interruzione (che riabilita la linea) in modo da non causare erroneamente una seconda richiesta di interrupt.
- *Edge Triggered Mode* (LTIM=0): le richieste di interruzione su una linea IR sono riconosciute da una transizione da un livello logico basso ad un livello logico alto su tale linea. Il segnale IR può rimanere attivo sulla linea senza generare erroneamente altre richieste di interruzione.

Special fully nested mode

La modalità **special fully nested mode** è utilizzata in un sistema in cui sono presenti più controllori 8259 in cascata, qualora sia necessario preservare la priorità degli interrupt all'interno di ciascun dispositivo slave e il funzionamento sia di tipo EOI.

Tale modalità di funzionamento deve essere attivata per il controllore master, in modo che questo possa rilevare, quando una richiesta di interruzione da parte di uno slave è in servizio, una richiesta di interruzione a priorità maggiore da parte dello stesso slave.

Se il master non operasse in special fully nested mode infatti, un'eventuale richiesta di interruzione a priorità maggiore ricevuta dallo slave che gestisce la linea IR attualmente in servizio non verrebbe rilevata dal master, questo perchè la linea corrispondente a tale slave risulterebbe disabilitata dal flag attivo ad essa corrispondente del registro ISR.

Lo special fully nested mode opera in maniera analoga al fully nested mode, con le seguenti differenze:

- Quando una richiesta di interrupt da parte di uno slave è in servizio, la linea di interruzione corrispondente a tale slave non viene disabilitata dal master. In questo modo il master è in grado di riconoscere eventuali richieste di interruzione a priorità maggiore provenienti da tale slave;
- Uscendo da una routine di servizio dell'interrupt, il programmatore deve prima inviare un comando di *non specific EOI* allo slave, poi leggere il contenuto del registro ISR di tale slave e, solo nel caso in cui il contenuto di tale registro sia nullo (cioè nessuna altra richiesta di interruzione gestita dallo slave sia in servizio), inviare al master un comando di *non specific EOI*. In caso contrario, delle interrupt provenienti dallo slave sono ancora in servizio e il comando di EOI al master non deve essere inviato;

Buffered mode

Utilizzando il controllore 8259 in un sistema complesso, potrebbe essere necessario utilizzare dei buffer di bus per garantire l'integrità dei dati e vigilare sulle contese del bus. In tali sistemi è necessario che il controllore generi opportuni segnali di abilitazione per i buffer di bus nel momento in cui deve eseguire l'output di dati. Per generare tali segnali di abilitazione, che vengono veicolati attraverso il segnale $\overline{SP} / \overline{EN}$, deve essere abilitata per il controllore la modalità buffered mode, mediante l'invio in fase di inizializzazione della ICW₄ con valore del bit BUF posto a 1.

5.3.5 Collegamento in cascata di controllori 8259

Più di un controllore 8259 può essere utilizzato per espandere il numero di linee di richiesta di interruzione fino ad un massimo di 64. Per fare ciò è possibile collegare tra loro più controllori 8259, secondo uno schema di collegamento master/slave che comprenda un dispositivo master e fino a 8 dispositivi slave.

Ogni controllore 8259 deve essere necessariamente inizializzato, oltre che con le parole di inizializzazione obbligatorie, anche con la terza parola di inizializzazione.

L'identificazione del ruolo del dispositivo può avvenire in due modi, a seconda che il controllore sia utilizzato in modo buffered o non buffered:

- Se il controllore è utilizzato in modo buffered, il ruolo dei dispositivi è esplicitamente comunicato in fase di inizializzazione per mezzo del bit M/S della ICW₄;

- Se il controllore è utilizzato in modo non buffered, il ruolo dei dispositivi è determinato in maniera fissa collegando il piedino $\overline{SP} / \overline{EN}$ degli stessi allo zero logico per gli slave e all'uno logico per il master;

Il dispositivo master controlla gli slave per mezzo delle tre linee CAS, tali segnali sono di output per il dispositivo master e di input per i dispositivi slave. Sono utilizzati dal master, in seguito alla ricezione del segnale di interrupt acknowledge dal processore, per abilitare lo slave con una richiesta attiva correntemente a più alta priorità. Tale dispositivo dovrà rispondere al segnale di *INTA* della CPU ponendo sul data bus l'identificativo del vettore a priorità massima tra i suoi correntemente attivi.

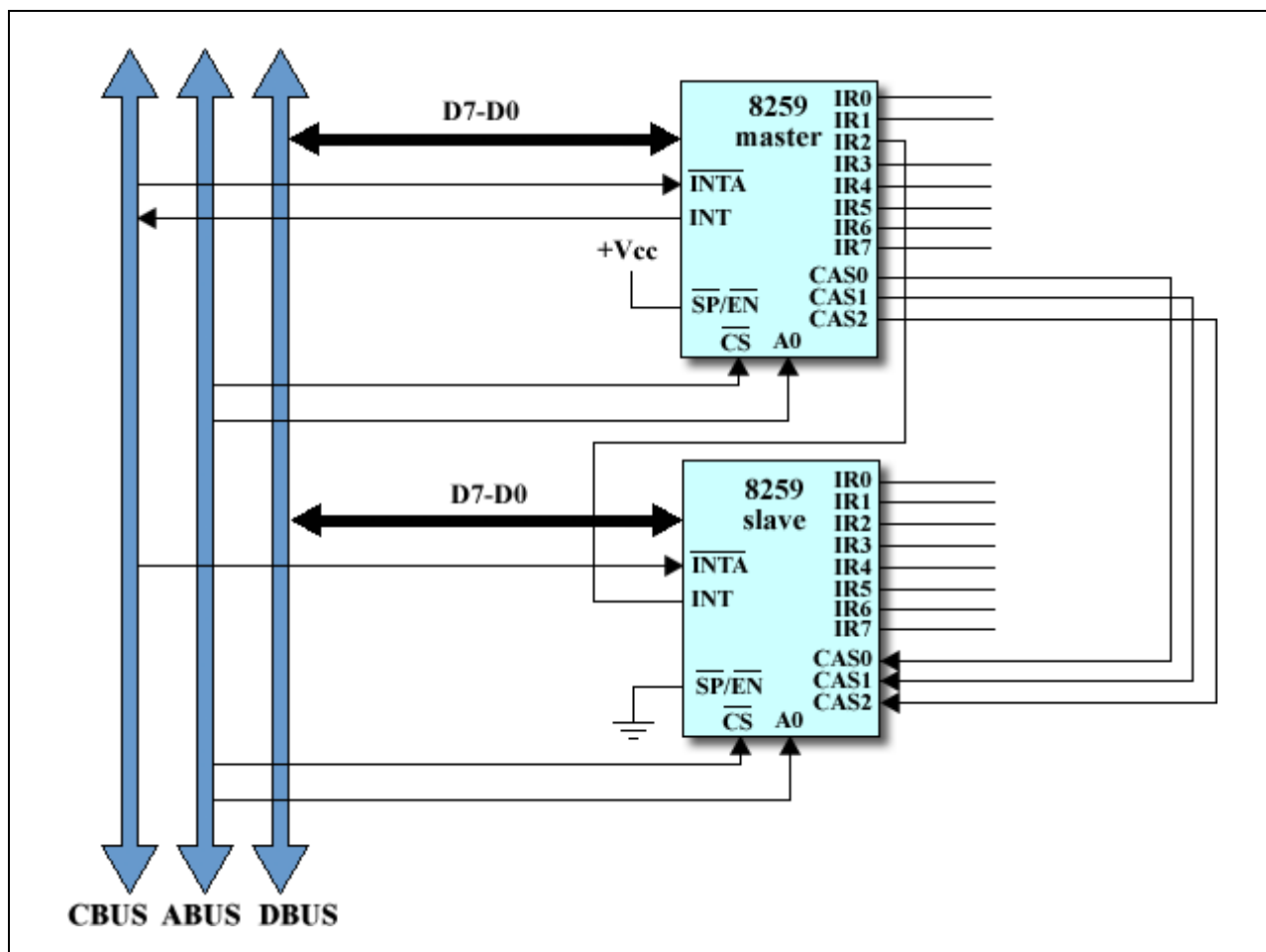


Fig. 40: Collegamento di due controllori dell'interrupt 8259 in cascata. Tale schema di collegamento corrisponde con quello tipico dei controllori 8259 all'interno dei personal computer.

Quando uno dei dispositivi slave riceve una richiesta di interruzione su uno dei suoi livelli, e questa non è mascherata o disabilitata perchè richieste di priorità maggiore sono in servizio, attiva il proprio segnale di INT, collegato ad una delle linee di interruzione del master.

Nel caso in cui il master non mascheri le richieste su tale livello o non stia correntemente servendo una richiesta proveniente da uno slave a priorità maggiore (o dallo stesso slave, se il master non opera in special fully nested mode), la richiesta di interruzione pervenuta al master viene inviata alla CPU per mezzo del segnale INT del master.

Il processore, appena è disponibile a servire tale richiesta, invia al master il primo segnale di acknowledge. Ricevuto tale segnale il master setta il bit relativo allo slave richiedente a massima priorità nel proprio registro ISR, resetta il corrispondente bit di IRR e verifica, leggendo il contenuto di un registro interno in cui è stata memorizzata la ICW₃, se tale richiesta proviene o meno da un controllore slave.

Se la richiesta non proviene da uno slave il funzionamento del master prosegue come un normale controllore 8259 che rileva la richiesta di interruzione di un periferico.

Se la richiesta proviene da un controllore slave, il master abilita tale controllore alla ricezione del segnale di acknowledge ponendo sui segnali CAS il codice binario corrispondente al numero di linea IR a cui tale slave è collegato.

Il segnale di INTA inviato dal processore è ricevuto da tutti i dispositivi. Ogni slave confronta il proprio identificativo, ricavato da un opportuno registro interno in cui è stata memorizzata la ICW₃, con il segnale presente sulle linee di CAS. Se c'è corrispondenza, lo slave si riconosce come destinatario del segnale di INTA. Lo slave selezionato setta il bit relativo al proprio livello attivo a priorità massima nel proprio registro ISR e resetta il corrispondente bit di IRR.

All'invio da parte del processore del secondo segnale di INTA, lo slave selezionato pone sul data bus l'identificativo di vettore corrispondente alla richiesta di interruzione da esso selezionata.

Nel caso di funzionamento in AEOI, sia il master che lo slave resettano il valore del bit di ISR relativo alla linea IR appena servita subito dopo la ricezione del secondo impulso di INTA.

Nel caso di funzionamento in EOI, la routine di servizio dovrà inviare un comando di EOI sia al master che allo slave, ponendo attenzione eventualmente alla modalità di funzionamento in special fully nested mode del master.

5.4 Gestione delle interruzioni hardware nei PC

5.4.1 Gestione delle interruzioni mascherabili

Nei personal computer le interruzioni hardware mascherabili sono gestite per mezzo di due controllori 8259 in cascata, tipicamente integrati nel southbridge del chipset.

I PC sono pertanto in grado di gestire fino a 15 linee di interruzione mascherabili di cui rispettivamente 7 collegate direttamente al controllore master e 8 collegate al controllore slave.

I due controllori 8259 sono collegati in modalità master/slave, con il controllore slave collegato alla linea di richiesta dell'interrupt IR2 del master (come mostrato in Fig. 40). L'ordine di priorità delle interruzioni mascherabili vede pertanto, le richieste associate alle due linee basse del master alla priorità maggiore, immediatamente seguite dalle richieste, in ordine, associate alle linee dello slave per finire con le richieste associate alle rimanenti cinque linee del controllore master.

La configurazione dei due controllori 8259 per i personal computer è descritta nella seguente tabella:

Impostazione	Master	Slave
Modalità di gestione delle priorità	Fully Nested Mode (non special)	Fully Nested Mode
Modalità di terminazione del ciclo di interruzione	End of Interrupt	End of Interrupt
Valore iniziale n dei vettori	08H	70H
Indirizzo di I/O	20H	A1H

Tali controllori vengono inizializzati all'avvio del sistema dal BIOS, o dal sistema operativo al caricamento di quest'ultimo, per mezzo delle opportune parole di inizializzazione.

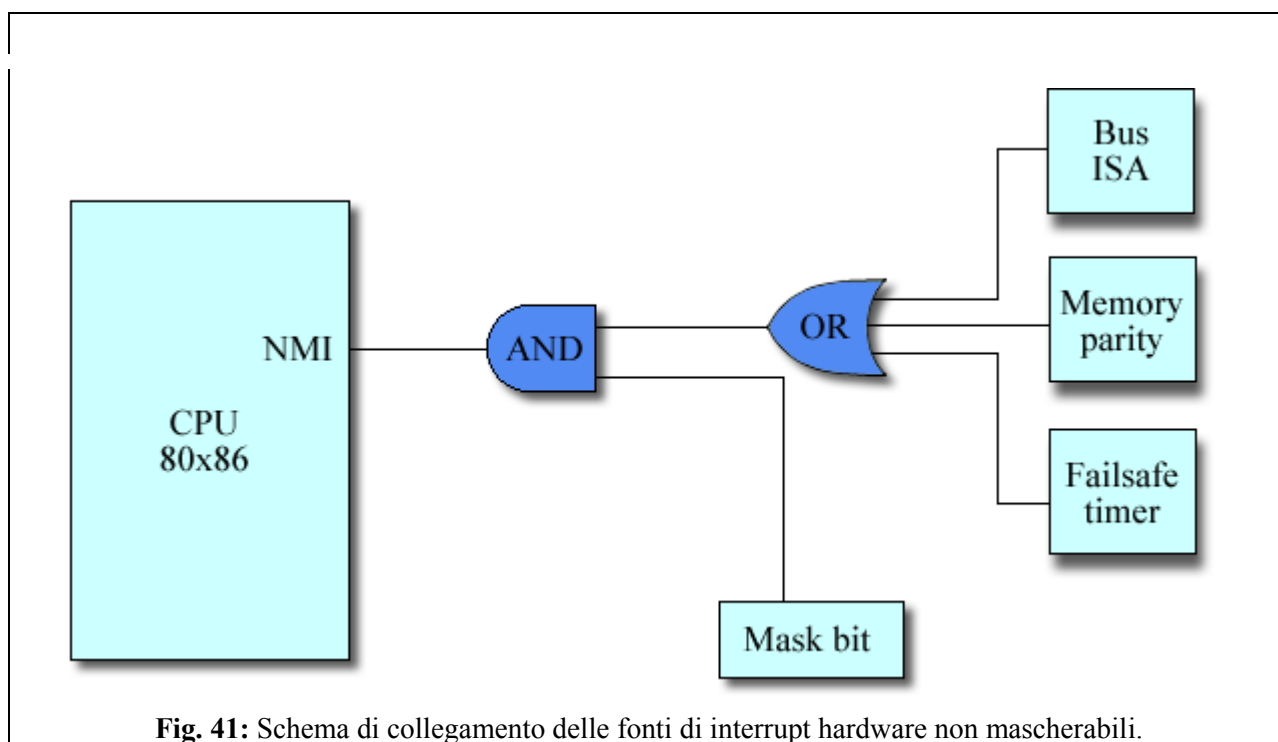
L'assegnazione standard delle linee di richiesta delle interruzioni alle periferiche è la seguente, in ordine di priorità:

- IR0_{master}: 8253 timer 0 (determina l'intervallo di temporizzazione del sistema);
- IR1_{master}: Tastiera;
- IR2_{master}: 8259 slave;
 - IR0_{slave}: Riservato/Real Time Clock;
 - IR1_{slave}: Riservato/LAN;
 - IR2_{slave}: Riservato;
 - IR3_{slave}: Riservato;
 - IR4_{slave}: Coprocessore/Mouse (non seriale);
 - IR5_{slave}: Coprocessore 80x87;
 - IR6_{slave}: Hard Disk;
 - IR7_{slave}: Riservato;
- IR3_{master}: COM2;
- IR4_{master}: COM1;
- IR5_{master}: LPT2;
- IR6_{master}: Floppy;
- IR7_{master}: LPT1;

5.4.2 Gestione delle interruzioni non mascherabili

Oltre alle interruzioni hardware mascherabili, i personal computer permettono la rilevazione di diverse fonti di interruzione hardware non mascherabili. Essendo disponibile, sui processori x86 un unico piedino di rilevazione di interrupt non mascherabili, le richieste provenienti dalle diverse fonti di tali interruzioni devono essere convogliate su tale piedino per mezzo di una apposita circuiteria.

In particolare, tutti i segnali di richiesta di interruzione non mascherabile vengono raccolti e posti in OR tra loro per mezzo di una apposita logica di OR. Il segnale in output a tale logica identifica la presenza di una richiesta di interruzione non mascherabile da parte di una delle diverse fonti.



Tale segnale viene poi posto in AND con un apposito bit di maschera prima di essere inviato al piedino NMI del processore. Il bit di maschera è uno speciale bit attivabile e disattivabile

via software che permette di disabilitare la rilevazione delle interrupt non mascherabili. Ciò può essere utile ad esempio in fase di test.

Tra le possibili fonti di interruzioni non mascherabili si evidenziano:

- Segnali del bus ISA utilizzati per rappresentare l'incorrere di eventi critici;
- Segnali provenienti dalla memoria ad indicare l'incorrere di errori di parità sui banchi;
- Segnali provenienti da un particolare dispositivo timer, detto failsafe timer, utilizzabile per temporizzare eventi real-time;

Nel momento in cui il processore rileva la presenza di una interruzione non mascherabile, il flusso di esecuzione corrente viene sospeso e si passa ad eseguire la routine di servizio dell'interrupt il cui puntatore è memorizzato in posizione 2 all'interno del vettore delle interruzioni. All'interno di tale routine le diverse fonti di interrupt non mascherabili vengono interrogate in polling, secondo un determinato ordine di priorità, per determinare quale tra esse è la fonte a priorità più alta ad aver scatenato l'interruzione.

6. Direct Memory Access nell'architettura 80x86

Il **direct memory access** o **DMA** è un metodo di gestione delle periferiche di I/O che permette di effettuare trasferimenti dati diretti, dalla memoria ai periferici e viceversa, senza coinvolgere la CPU.

È la modalità di trasferimento tipicamente utilizzata per gestire dispositivi a blocco, cioè quei dispositivi che trasferiscono sempre blocchi di dati di dimensione prefissata, e ad alta velocità. I trasferimenti in DMA sono utilizzati tipicamente per operazioni di lettura e scrittura da dischi o interfacce di rete, operazioni di scrittura su memoria video per il refresh dello schermo, trasferimenti da memoria a memoria ad alta velocità, ecc.

Quando un periferico, gestito in DMA, richiede una operazione di trasferimento, invia una richiesta di trasferimento DMA (*DREQ*) ad un apposito dispositivo, chiamato **DMA controller**. Tale dispositivo è in grado di ottenere il controllo del bus di sistema ed eseguire il trasferimento diretto di dati tra il periferico richiedente e la memoria. Per fare ciò, il DMA controller deve negoziare l'acquisizione del bus con la CPU e deve essere dotato di una *Bus Interface Unit* che gli permetta di amministrare il bus, generando i segnali di indirizzo e di controllo con le opportune tempistiche (cioè generando i cicli di bus).

Un DMA controller è in grado tipicamente di gestire i trasferimenti diretti tra la memoria e un certo numero di dispositivi periferici. Per fare ciò, un DMA controller è organizzato logicamente in **canali di DMA**, a ciascuno dei quali è associato un diverso dispositivo periferico. Un canale di DMA si compone di un certo numero di registri, utilizzati per mantenere le informazioni necessarie ai trasferimenti che coinvolgono il particolare dispositivo, e di alcuni segnali che permettono l'handshaking tra il DMA controller e il dispositivo periferico.

Per poter operare il trasferimento, ogni canale del DMA controller deve essere precedentemente programmato (in genere da un opportuno programma **driver**, eseguito all'avvio del sistema operativo), con le seguenti informazioni:

- *Indirizzo di memoria di partenza da cui iniziare a leggere/scrivere i dati;*
- *Dimensione in byte del blocco di dati da trasferire;*
- *Modalità di trasferimento;*

Una volta ottenuto il controllo del bus dalla CPU, il DMA controller genererà i successivi cicli di bus necessari per eseguire il trasferimento, sulla base dei propri parametri di inizializzazione. Il controllo del bus verrà infine rilasciato alla CPU.

Si definisce **lettura DMA** l'operazione di trasferimento dati dalla memoria ad un dispositivo periferico. Si indica invece con **scrittura DMA** l'operazione di trasferimento dati da un dispositivo periferico alla memoria.

I vantaggi nell'effettuare trasferimenti in DMA sono principalmente due:

- *Possibilità di eseguire istruzioni in parallelo al trasferimento:* sollevando la CPU dall'esecuzione del trasferimento dati, questa potrebbe essere in grado di avanzare nel flusso di esecuzione corrente, mentre il trasferimento viene eseguito dal DMA controller. Ciò risulta possibile unicamente se le successive istruzioni da eseguire non coinvolgono il bus di sistema (se ad esempio gli operandi sono contenuti in registri interni, in cache, ecc). Tale situazione però non risulta sempre verificata, rendendo tale vantaggio probabilistico;
- *Riduzione del numero di cicli di bus necessari al trasferimento:* il trasferimento di una parola sul bus avviene sempre, in DMA, in un numero minore di cicli di bus. Trasferendo una parola non in DMA, questa deve essere in primo luogo trasferita dalla periferica (o dalla memoria) ad un registro interno della CPU e in secondo luogo trasferita da tale registro all'interfaccia di periferica (o alla memoria). In DMA, invece, i dati non transitano dalla

CPU ma fluiscono direttamente tra la memoria e i periferici. La tabella successiva indica il numero di cicli di bus richiesti per il trasferimento di dati da un dispositivo periferico alla memoria in diverse condizioni (in cui F indica un ciclo di fetch e E un ciclo di execute):

Situazione	Cicli di bus richiesti	Numero totale di cicli
Macchina sequenziale senza DMA	$F_{IN} + E_{IN} + F_{MOV} + E_{MOV}$	4 cicli
Macchina con pipeline senza DMA	$E_{IN} + E_{MOV}$	2 cicli
Macchina con pipeline e DMA	E_{MOV}	1 ciclo

I trasferimenti di dati in DMA sono sempre più veloci perchè coinvolgono un numero inferiore di cicli di bus.

Per permettere la gestione dei periferici in DMA, l'architettura deve prevedere:

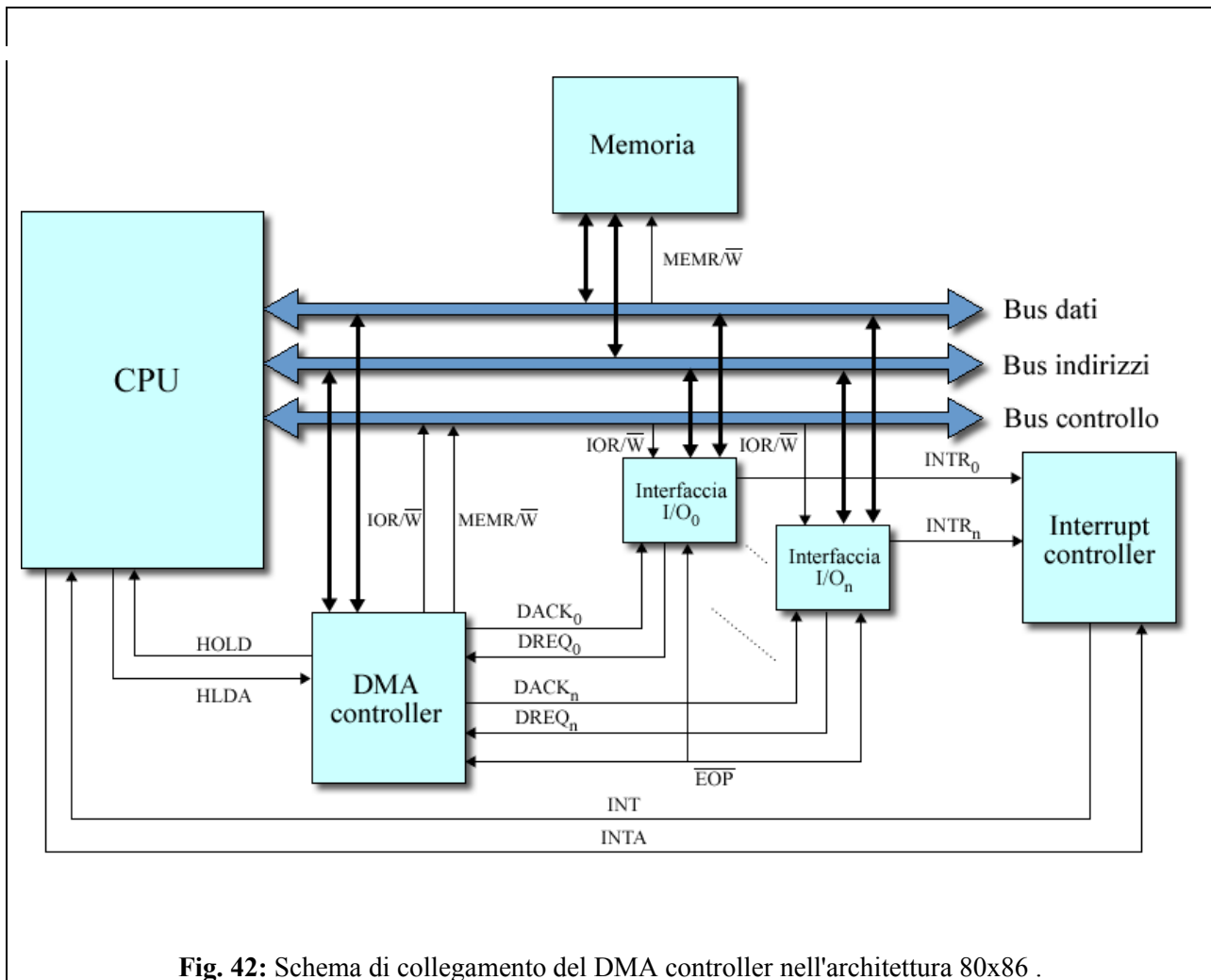
- *Cicli di bus specifici per i trasferimenti in DMA*: cicli che permettano il trasferimento diretto tra memoria e periferici;
- *Dispositivo DMA controller*: dispositivo dotato di una *Bus Interface Unit* in grado di generare gli appositi cicli di bus dedicati ai trasferimenti in DMA;
- *Meccanismo di arbitraggio del bus*: meccanismi che permettano di dirimere le contese di utilizzo del bus di sistema da parte del DMA controller, della CPU e di eventuali altri dispositivi master;

6.1 Schema di collegamento del DMA controller

In un sistema a microprocessore, il controllore del DMA comunica direttamente con ciascun periferico da esso gestito per mezzo di due segnali:

- **DREQ** (*DMA Request*): utilizzato dal dispositivo periferico per richiedere l'esecuzione di una operazione di trasferimento in DMA;

- **DACK** (*DMA Acknowledge*): utilizzato dal DMA controller per indicare ad un periferico che la sua richiesta è stata accolta e che il bus è pronto per iniziare il trasferimento;



Due segnali di controllo vengono utilizzati dalla CPU e dal DMA controller per negoziare il controllo del bus di sistema. Tali segnali sono:

- **HOLD** (*Hold*): utilizzato dal DMA controller per richiedere il controllo del bus di sistema alla CPU. La presenza di un segnale di HOLD a livello logico alto, viene testata dal processore ad ogni colpo di clock. Il processore può pertanto rilevare tale richiesta, ad un qualunque colpo di clock, anche nel mezzo dell'esecuzione di una istruzione. Quando la CPU rileva il segnale di HOLD, termina il ciclo di bus corrente e pone i propri segnali in

uno stato di alta impedenza, in cui il processore risulta fisicamente scollegato dal bus di sistema.

- **HLDA** (*Hold Acknowledge*): utilizzato dal processore per indicare al DMA controller di aver accolto la sua richiesta di controllo del bus. Viene inviato dalla CPU subito dopo aver posto i propri segnali in stato di alta impedenza.

Il segnale di HOLD, essendo verificato ad ogni colpo di clock, viene riconosciuto e servito con una priorità maggiore rispetto ai segnali di INT o NMI.

Il segnale di controllo \overline{EOP} viene utilizzato dal DMA controller per indicare all'esterno il completamento del trasferimento in DMA dell'intero blocco di dati. Tale segnale viene tipicamente collegato alle interfacce dei periferici. Quando il periferico, abilitato dal segnale DACK, vede attivarsi il segnale \overline{EOP} , può attivare una richiesta di interruzione verso il processore al fine di richiamare in esecuzione il programma driver ad esso associato e riprogrammare eventualmente il canale del DMA controller per un nuovo trasferimento.

6.2 Metodi di trasferimento in DMA

A seconda della velocità del periferico coinvolto nel trasferimento, il DMA controller può essere programmato per operare in due modalità:

- **Cycle stealing**: il DMA controller esegue un ciclo di DMA, trasferendo una singola parola di dati, dopodichè rilascia il controllo del bus di sistema alla CPU. Il trasferimento del blocco di dati per il quale il DMA controller viene programmato, avviene mediante una sequenza di singoli cicli di DMA, intervallati da periodi in cui il controllo del bus di sistema ritorna alla CPU. Tale modalità di trasferimento risulta conveniente per la gestione in DMA di periferici lenti.

- **Burst transfer:** il DMA controller mantiene il controllo del bus fino al termine del trasferimento del blocco di dati per il quale è stato programmato. Il trasferimento del blocco avviene per mezzo di una serie di cicli di DMA contigui, al termine dei quali il bus viene rilasciato alla CPU. Tale modalità di trasferimento risulta conveniente per la gestione in DMA di periferici veloci.

6.3 Cicli di DMA

Si definisce **ciclo di DMA**, un particolare ciclo di bus generato dal DMA controller che permette il trasferimento diretto di dati tra un dispositivo periferico e la memoria.

Tale trasferimento è operato abilitando contemporaneamente il segnale di lettura della memoria e di scrittura dell'I/O, per una lettura in DMA, oppure il segnale di scrittura della memoria e di lettura dell'I/O, per una scrittura in DMA.

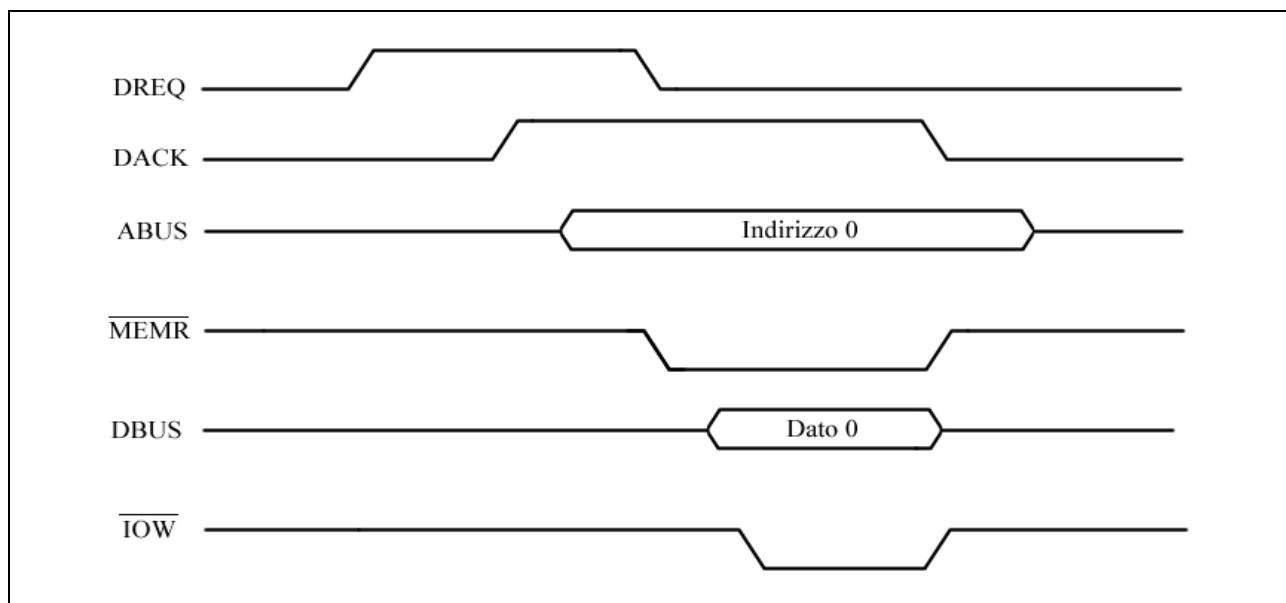


Fig. 43: Tempistica di un ciclo di lettura DMA.

Il bus indirizzi viene utilizzato, nel ciclo di DMA, per specificare l'indirizzo della cella di memoria di partenza coinvolta nel trasferimento. Il dispositivo periferico viene invece selezionato direttamente dal DMA controller per mezzo del segnale DACK dedicato (che si può collegare, ad esempio al piedino di abilitazione \overline{CS}).

Tipo di operazione DMA	Segnali di controllo abilitati durante il ciclo di DMA
Lettura DMA (Memoria → Periferica)	MEMR, I/OW
Scrittura DMA (Periferica → Memoria)	MEMW, I/OR

6.4 Sequenza di trasferimento DMA

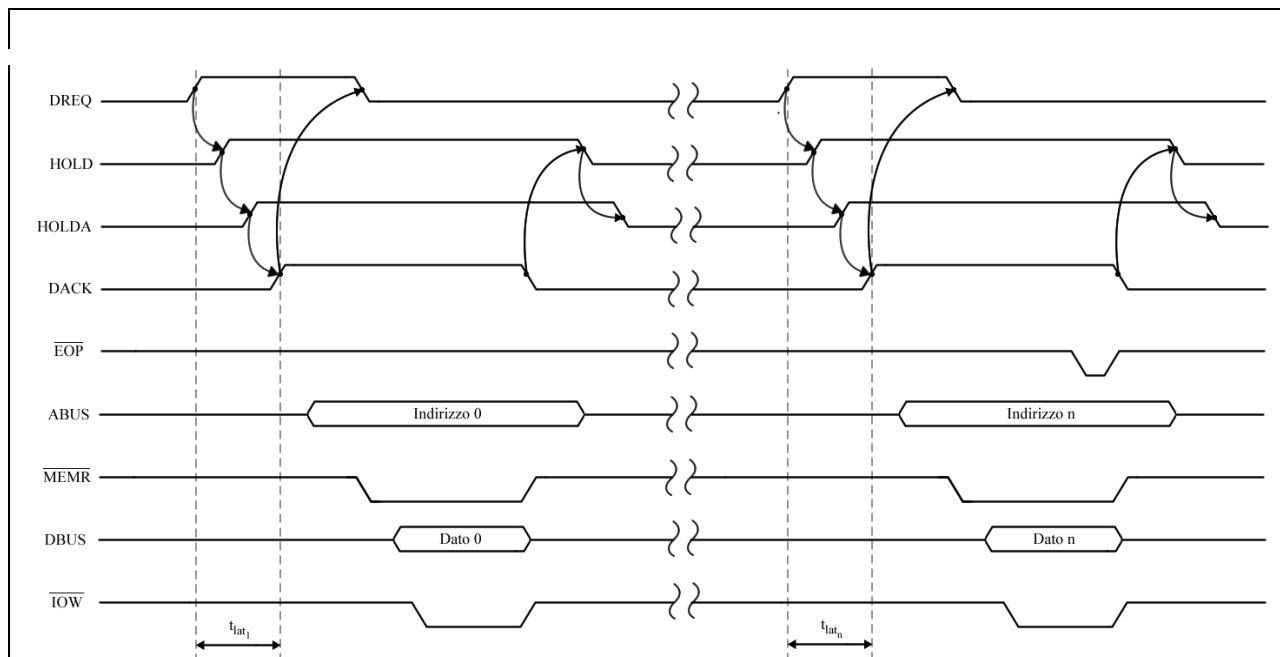


Fig. 44: Tempistica di un trasferimento in DMA di un blocco di dati da un dispositivo periferico a memoria.

La sequenza di richiesta di trasferimento in DMA è descritta nel diagramma temporale e opera nel seguente modo:

1. Ogni canale del DMA controller deve essere opportunamente inizializzato con l'indirizzo di partenza dell'area di memoria da cui leggere o su cui scrivere i dati (*buffer di I/O*), la dimensione in byte del blocco da trasferire e la modalità di trasferimento desiderata. Tale inizializzazione viene tipicamente eseguita da un programma driver della periferica associata al canale.
2. Il dispositivo periferico che necessita di una operazione di trasferimento in DMA attiva il segnale DREQ.

3. Il segnale di richiesta viene rilevato dal DMA controller che richiede a sua volta alla CPU il controllo del bus di sistema, attivando il segnale HOLD.
4. Dopo un certo tempo di latenza t_{lat} , la CPU accoglie la richiesta di controllo del bus da parte del DMA controller. Il processore si scollega dal bus di sistema ponendo i propri segnali in stato di alta impedenza ed indica al DMA controller il rilascio del bus mediante il segnale di acknowledge HLDA.
5. Il DMA controller riceve il segnale HLDA e diventa il master del bus. Da tale momento in poi i segnali di indirizzo e di controllo verranno generati dalla BIU interna al DMA controller. Il DMA controller seleziona il dispositivo che ha richiesto il trasferimento, per mezzo del segnale DACK. Predispone poi il bus per effettuare un ciclo di DMA, attivando gli opportuni segnali di controllo (MEMR e I/OW per i cicli di lettura DMA oppure MEMW e I/OR per i cicli di scrittura) e ponendo sul bus degli indirizzi, l'indirizzo relativo alla cella di partenza per il trasferimento.
6. Il segnale di acknowledge inviato al dispositivo funge da segnale di abilitazione, tale dispositivo, una volta abilitato, diventa sensibile ai segnali di controllo I/OW e I/OR ed immette o legge di conseguenza, sul bus dati, la parola da trasferire.
7. Il contatore dei byte da trasferire del canale viene decrementato della dimensione della parola trasferita, mentre l'indirizzo di memoria da o a cui trasferire viene incrementato dello stesso valore.
8. Si distinguono due casi a seconda della modalità di trasferimento impostata per il canale:
 - **Burst transfer**: il DMA controller prosegue effettuando ulteriori cicli di DMA fino a trasferire l'intero blocco, mantenendo attivo il segnale HOLD. In tal caso, i successivi cicli di DMA non devono essere preceduti da una nuova fase di negoziazione del bus e i trasferimenti avvengono come dal passo 6 in poi.

- **Cycle stealing:** il DMA controller disabilita il dispositivo, rimuovendo il segnale DACK, e rilascia il controllo del bus alla CPU. Quando il dispositivo avrà poi a disposizione una nuova parola, la sequenza di trasferimento riprenderà dal passo 2 con una nuova fase di negoziazione ed un nuovo ciclo di DMA.
9. Al termine del trasferimento dell'intero blocco, cioè quando il contatore dei byte da trasferire giunge a zero (cioè si raggiunge il *terminal count*: **TC**), il DMA controller invia un segnale di interruzione EOP per notificare la conclusione del trasferimento in DMA. Disabilita poi il dispositivo, rimuovendo il segnale DACK, e rilascia il controllo del bus alla CPU.

Il rilascio alla CPU del controllo del bus di sistema avviene mediante il seguente procedimento:

1. Il DMA controller notifica alla CPU di non necessitare più il controllo del bus di sistema, disattivando il segnale di HOLD.
2. Quando il processore rileva un valore logico basso sul segnale di HOLD, revoca il controllo del bus al DMA controller disattivando il segnale di HLDA e facendo uscire i propri segnali dallo stato di alta impedenza.

Il *tempo di latenza* t_{lat} è definito come il tempo che intercorre tra la richiesta di trasferimento in DMA da parte di un periferico e l'istante in cui il bus è pronto per effettuare il ciclo di DMA. Dal momento che il segnale di richiesta del bus viene rilevato dal processore ad ogni colpo di clock e che il processore deve terminare il ciclo di bus corrente prima di lasciare il controllo al DMA controller, il massimo tempo di latenza, tralasciando i tempi di propagazione dei segnali, è pari al tempo necessario per effettuare un ciclo di bus. In un trasferimento operato in modalità burst, i tempi di latenza di ciascun ciclo di DMA successivo al primo risultano nulli.

6.5 DMA controller Intel 8237

Il dispositivo Intel 8237 è un DMA controller a quattro canali, ognuno programmabile separatamente, che permette di eseguire tre tipi di trasferimenti diretti:

- Da dispositivo periferico a memoria;
- Da memoria a dispositivo periferico;
- Da memoria a memoria;

Più dispositivi 8237 possono essere collegati in cascata, secondo una strategia master/slave, per aumentare il numero di canali disponibili, e quindi il numero di periferiche che possono essere gestite in DMA.

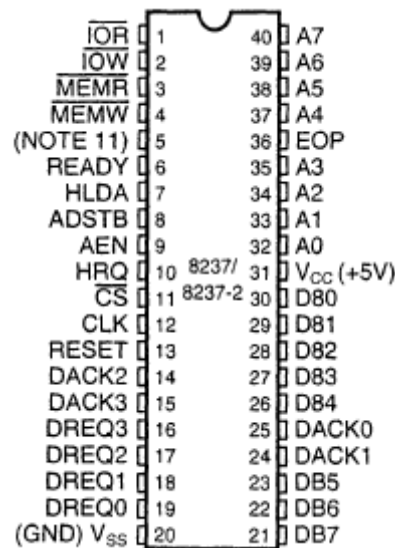


Fig. 45: Piedinatura del controllore di DMA 8237.

Tale dispositivo, inizialmente costituito da un chip a sé stante, nei moderni sistemi è integrato all'interno del southbridge. Permette di effettuare trasferimenti ad una velocità massima di 1.6 Mbyte/sec, operando sulla base di un proprio clock interno con frequenza massima 5 Mhz.

6.5.1 Segnali e schema di collegamento

Il dispositivo 8237 presenta i seguenti segnali:

- **CLK** (*Clock*): segnale di sincronismo interno all'8237, sulla base del quale vengono temporizzati i cicli di DMA;
- **\overline{CS}** (*Chip Select*): segnale utilizzato dalla CPU per selezionare il controllore 8237 come un dispositivo di I/O durante il ciclo di *idle*. Utilizzato durante la programmazione dell'8237;
- **$DREQ_0 \div DREQ_3$** (*DMA Requests*): segnali asincroni utilizzati dalle periferiche per richiedere un trasferimento DMA all'8237. Quando una periferica necessita un trasferimento in DMA, attiva tale segnale e lo mantiene attivo fino all'attivazione del segnale DACK corrispondente da parte del controllore 8237;
- **$DACK_0 \div DACK_3$** (*DMA Acknowledges*): segnali utilizzati dal controllore 8237 per abilitare le periferiche per le quali le richieste di trasferimento in DMA sono state accolte;
- **HRQ** (*Hold Request*): segnale mediante il quale il controllore 8237 richiede alla CPU il controllo del bus di sistema;
- **HLDA** (*Hold Acknowledge*): segnale utilizzato dalla CPU per indicare all'8237 di aver rilasciato il controllo del bus di sistema;
- **RESET** (*Reset*): segnale utilizzato per azzerare i registri command, status, request e temporary;
- **READY** (*Ready*): segnale utilizzato per estendere i cicli di DMA in modo da operare in accordo alle periferiche più lente;
- **$D_0 \div D_7$** (*Data*): segnali connessi al bus dati, utilizzati per leggere o scrivere i registri interni del controllore 8237 in fase di programmazione o per specificare il byte più significativo dei 16 bit bassi dell'indirizzo di memoria durante i cicli di DMA;

- **$A_0 \div A_7$ (Address)**: segnali connessi al bus indirizzi, utilizzati per specificare il registro interno da leggere o scrivere in fase di programmazione, per specificare il byte meno significativo dei 16 bit bassi dell'indirizzo di memoria durante i cicli di DMA;
- **\overline{IOR} (I/O Read)**: segnale utilizzato in input, in fase di programmazione, per specificare una operazione di lettura dei registri interni dell'8237, o in output, durante i cicli di DMA, per specificare l'operazione di lettura per l'interfaccia del periferico selezionato;
- **\overline{IOW} (I/O Write)**: segnale utilizzato in input, in fase di programmazione, per specificare una operazione di scrittura dei registri interni dell'8237, o in output, durante i cicli di DMA, per specificare l'operazione di scrittura per l'interfaccia del periferico selezionato;
- **\overline{MEMR} (Memory Read)**: segnale utilizzato in output per specificare l'operazione di lettura della memoria durante i cicli di DMA;
- **\overline{MEMW} (Memory Write)**: segnale utilizzato in output per specificare l'operazione di scrittura della memoria durante i cicli di DMA;
- **\overline{EOP} (End of Process)**: segnale utilizzato in input per interrompere il trasferimento in DMA, o in output per indicare il termine del trasferimento in DMA.
- **AEN (Address Enable)**: segnale utilizzato in output dal controllore 8237 per abilitare i circuiti di latch esterni, necessari per la scrittura della parte alta dell'indirizzo di memoria. Può essere utilizzato anche per disabilitare i buffer di input/output del sistema a microprocessore;
- **$ADSTB$ (Address Strobe)**: segnale utilizzato per indicare alla logica di latch che il byte più significativo dei 16 bit bassi di indirizzo è pronto sul data bus per essere riscritto sulla porzione dell'address bus corrispondente: $A_8 - A_{15}$.

In base alla particolare architettura utilizzata, il DMA controller 8237 deve essere in grado di generare segnali di indirizzo su 20, 32 o 64 bit da porre sull'address bus. Il controllore 8237 presenta però, per sua costruzione, alcune limitazioni che rendono più complicata la definizione di segnali di indirizzo compatibili con il parallelismo dell'address bus:

- Il numero di piedini dedicati ai segnali di indirizzo è pari solamente a 8;
- Il parallelismo interno dei registri di indirizzo è pari solamente a 16 bit;

Per sua costruzione, il controllore 8237 sarebbe pertanto in grado di gestire indirizzi al massimo definiti su 16 bit e di pilotare al massimo 8 delle linee di indirizzo dell'address bus. Occorre pertanto affiancare il dispositivo 8237 con dell'apposita circuiteria che permetta di superare tali limitazioni.

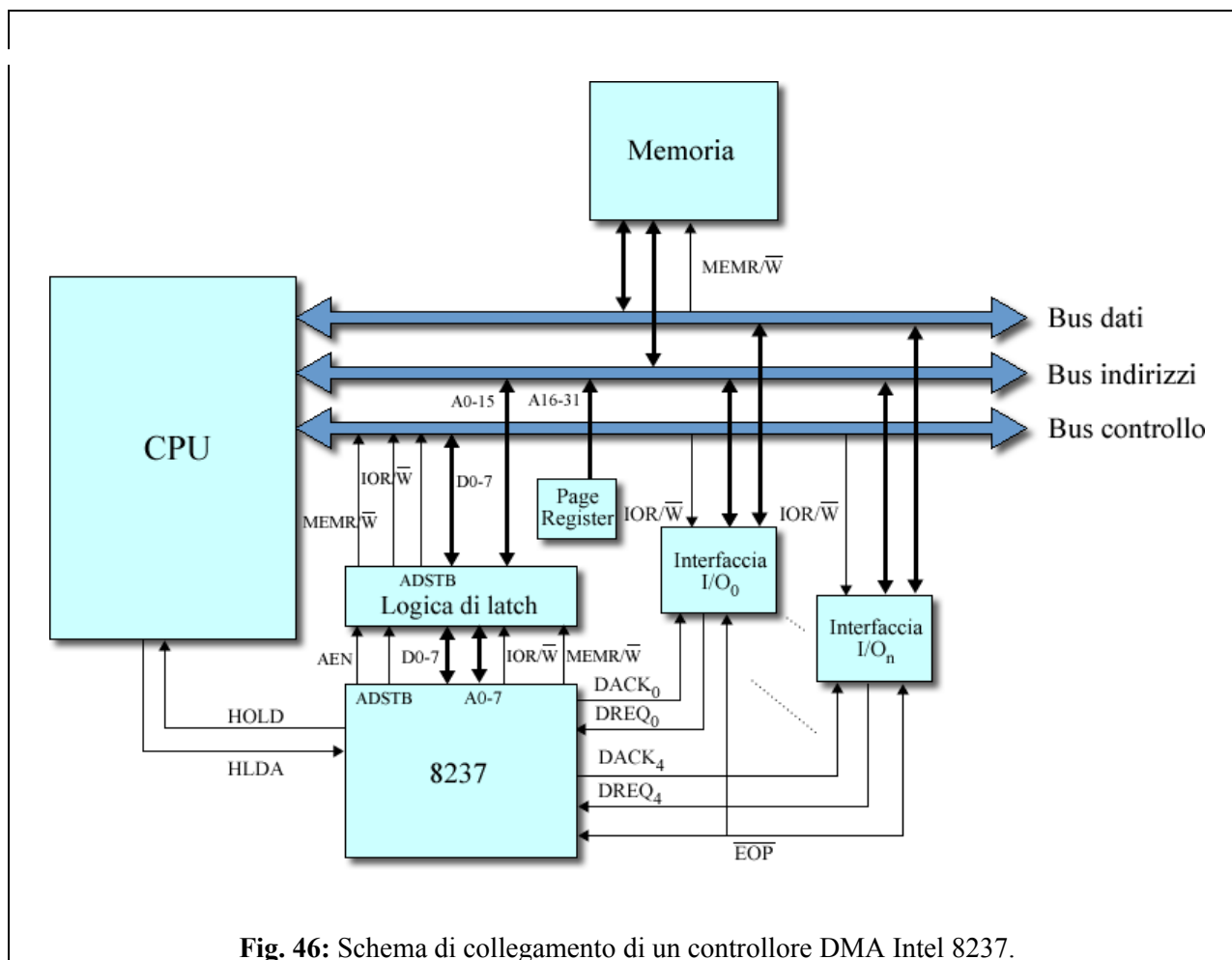


Fig. 46: Schema di collegamento di un controllore DMA Intel 8237.

Per superare la limitazione data dal numero di piedini dedicati ai segnali di indirizzo, si deve affiancare a tale dispositivo una **logica di latch**. Il byte alto dei 16 bit di indirizzo specificabili dall'8237, viene fornito in uscita, durante i cicli di DMA, sui piedini di dato D_0-D_7 . Questi vengono letti dalla logica di latch e riscritti sui corrispondenti piedini dell'address bus: $A_8 - A_{15}$.

Per superare la limitazione data dal ridotto parallelismo dei registri di indirizzo interni all'8237, si introduce invece, per ciascun canale, un registro esterno chiamato **page register**. Tali registri contengono la porzione superiore dell'indirizzo di memoria che deve essere scritto o letto dal canale durante il ciclo di DMA. Devono pertanto essere appositamente programmati prima di ogni trasferimento in DMA. Durante ogni ciclo di DMA, l'indirizzo fisico effettivo che viene posto sull'address bus è dato dalla giustapposizione dei 16 bit bassi generati dall'8237 e dalla porzione alta di indirizzo memorizzata nel page register dell'opportuno canale.

6.5.2 Modalità di funzionamento

L'8237 ha due modalità di funzionamento:

- **Programmazione:** durante la quale la CPU può leggere o scrivere i registri interni del DMA controller;
- **Generazione di cicli di DMA:** durante la quale il DMA controller riceve le richieste di trasferimento dai periferici ed effettua tali trasferimenti generando gli opportuni segnali di controllo ed indirizzo del bus di sistema;

La modalità di programmazione può essere attivata solo quando il dispositivo 8237 si trova nello stato di *idle*, selezionando il DMA controller per mezzo del segnale \overline{CS} .

In modalità di programmazione, l'8237 opera come un normale dispositivo periferico i cui registri interni possono essere acceduti, in lettura o scrittura, mediante i corrispondenti indirizzi. In particolare ciascun registro viene selezionato dai quattro segnali bassi A_0-A_3 dell'address bus.

In modalità di generazione di cicli DMA, l'8237 permane in uno stato di *idle* fino a quando non riceve richieste di trasferimento da parte dei dispositivi periferici collegati ai propri canali. In caso di richieste contemporanee seleziona quale servire per prima sulla base di un certo ordinamento di priorità dei canali. Una volta determinata la richiesta di trasferimento DMA da servire, richiede il controllo del bus alla CPU e si pone in uno stato di *bus wait*. Non appena la CPU cede il controllo del bus all'8237, questo dà avvio ad un ciclo di DMA che in genere dura 4 colpi di clock (ma può essere prolungato con dei cicli di wait in base al valore del segnale READY).

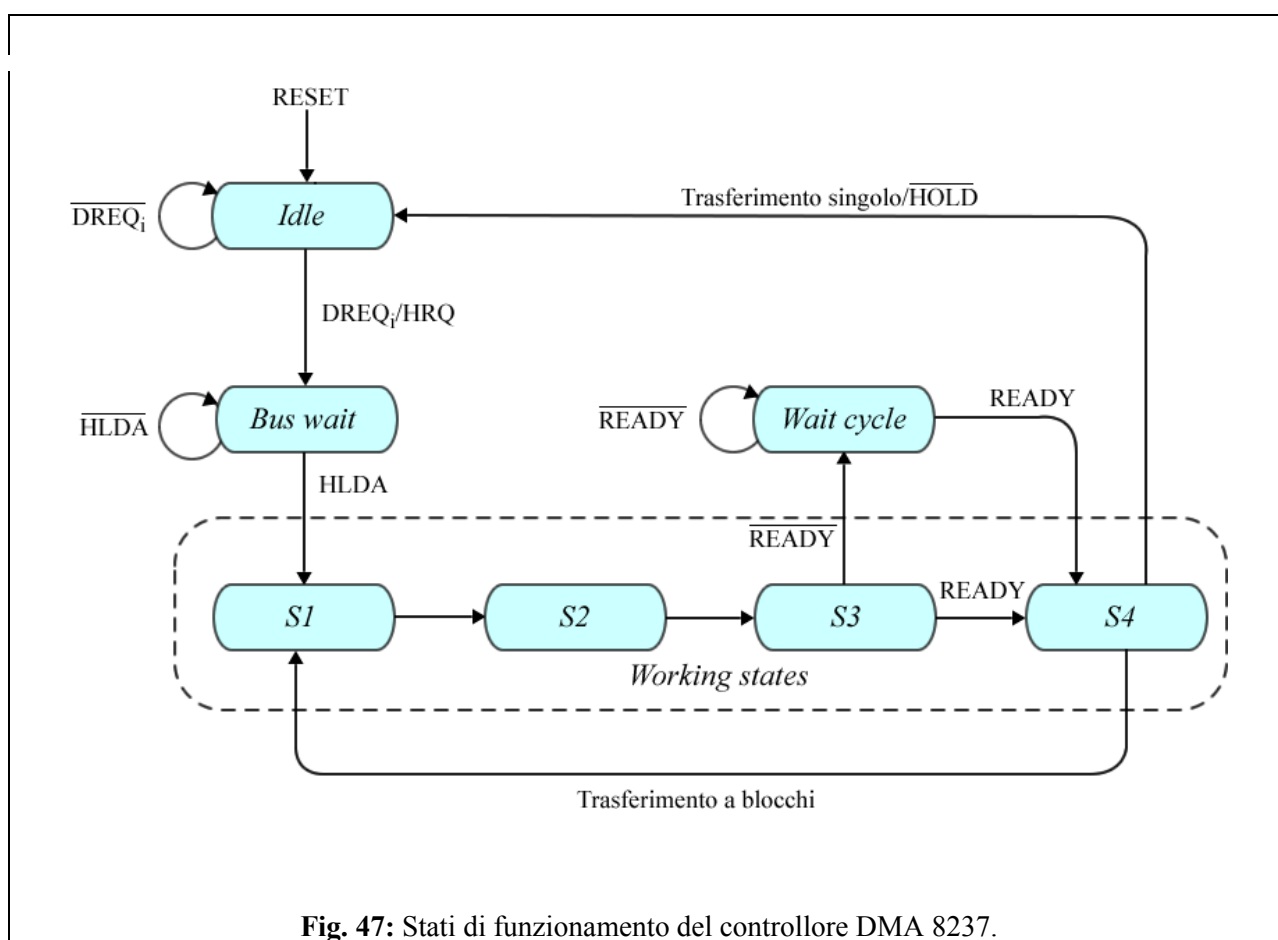


Fig. 47: Stati di funzionamento del controllore DMA 8237.

A seconda della modalità di trasferimento selezionata per il canale, al termine di tale ciclo l'8237 può continuare a generare cicli di DMA fino a terminare il trasferimento del blocco di dati, oppure rilasciare il controllo del bus alla CPU e ritornare in uno stato di *idle*.

Al termine del trasferimento in DMA dell'intero blocco di dati, il segnale di interruzione \overline{EOP} viene utilizzato in output per indicare, al programma driver del dispositivo, la fine del trasferimento in DMA complessivo sul canale. Il segnale \overline{EOP} può essere utilizzato in input per interrompere il trasferimento in DMA corrente.

L'8237 permette inoltre di richiedere l'esecuzione di trasferimenti in DMA direttamente via software.

6.5.3 Cicli di DMA

Il dispositivo 8237 è in grado di generare, i segnali di controllo ed indirizzo necessari per effettuare, sul bus di sistema, tre tipi di cicli di trasferimento in DMA:

1. Ciclo di lettura DMA (Memoria \rightarrow Periferico);
2. Ciclo di scrittura DMA (Periferico \rightarrow Memoria);
3. Ciclo di trasferimento memoria-memoria DMA (Memoria \rightarrow Memoria);

Ciascun ciclo DMA viene portato a termine in un certo numero di colpi di clock. Il clock di riferimento su cui i cicli di DMA vengono sincronizzati è il segnale di temporizzazione CLK che viene fornito all'8237 e che può presentare una frequenza massima di 5 Mhz.

I cicli di lettura e scrittura DMA occupano tipicamente 4 colpi di clock, mentre quelli di trasferimento memoria-memoria vengono portati a termine tipicamente in 8 colpi di clock. Ciascuno di tali cicli può essere opportunamente allungato, in accordo con le velocità dei dispositivi periferici, aggiungendo uno o più *cicli di wait*, sulla base del valore assunto dal segnale READY.

Il numero di byte trasferiti durante ciascun ciclo di DMA dipende dalle caratteristiche delle periferiche collegate ai canali. Il dispositivo 8237 deve essere a conoscenza di tale numero, dal momento che da esso dipendono:

- Il valore di cui deve essere decrementato il registro contatore dei byte da trasferire ad ogni ciclo di DMA per il canale;
- Il valore di cui deve essere incrementato o decrementato il registro contenente l'indirizzo di memoria a cui o da cui trasferire i byte per il ciclo di DMA corrente, per il canale;

La dimensione delle parole di trasferimento DMA deve essere definita in maniera cablata (*hardwired*) per ciascun dispositivo 8237.

Cicli di lettura e scrittura DMA

I cicli di lettura e scrittura DMA per il controllore 8237, occupano in genere 4 cicli di clock e si articolano nel seguente modo:

1. Il DMA controller genera i 16 bit bassi dell'indirizzo di memoria. Lo schema adottato prevede che:
 - Il byte meno significativo sia scritto sugli 8 segnali bassi dell'address bus A_0-A_7 ;
 - Il byte più significativo sia scritto sugli 8 segnali bassi del data bus D_0-D_7 ;
 - Venga attivato il segnale AEN di abilitazione della logica di latch;
 - Venga attivato il segnale di strobe ADSTB;

Il segnale di ADSTB indica l'inizio del ciclo di DMA e specifica alla logica di latch che il byte più significativo dei 16 bit bassi di indirizzo è pronto sul data bus per essere riscritto sulla porzione dell'address bus corrispondente: A_8-A_{15} .

2. Il DMA controller seleziona il dispositivo coinvolto nel trasferimento per mezzo dell'opportuno segnale DACK;

3. Il DMA controller attiva:

- Il segnale IOW se il ciclo è di lettura o il segnale di IOR se il ciclo è di scrittura;
- Il segnale MEMR se il ciclo è di lettura o il segnale di MEMW se il ciclo è di scrittura;

Cicli di trasferimento memoria-memoria DMA

Il ciclo di trasferimento memoria-memoria DMA per il controllore 8237, occupa in genere 8 cicli di clock e si articola in due fasi:

1. Trasferimento dalla memoria sorgente al *temporary register* del controllore 8237. Tale fase avviene in 4 colpi di clock, come un normale ciclo di lettura, DMA ma con le seguenti differenze:
 - Viene attivato il segnale MEMR ma non il segnale IOW;
 - Nessun periferico viene abilitato per prelevare i dati dal bus, questi infatti vengono prelevati direttamente dal controllore 8237.
2. Trasferimento dal *temporary register* interno alla memoria destinazione. Tale fase avviene in 4 colpi di clock, come un normale ciclo di scrittura DMA, ma con le seguenti differenze:
 - Viene attivato il segnale MEMW ma non il segnale IOR;
 - Nessun periferico viene abilitato per porre i dati sul bus dati, questi infatti vengono posti direttamente dal controllore 8237.

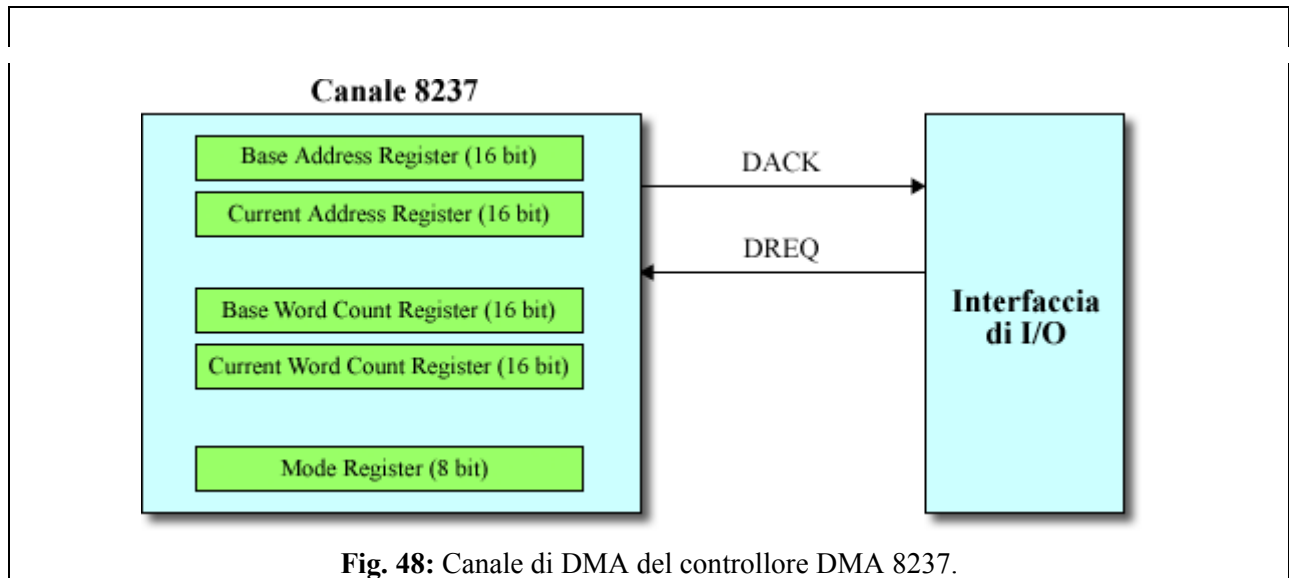
6.5.4 Registri generici dell'8237

L'8237 dispone di diversi registri programmabili, alcuni specifici per i singoli canali e altri generici per l'intero DMA controller. Tra i registri generici:

- **Command Register (CR):** è un registro su 8 bit che controlla il funzionamento complessivo del dispositivo 8237. Il processore può programmare tale registro per definire ad esempio:
 - L'utilizzo di priorità fisse o rotanti per i canali di DMA;
 - Se i segnali DREQ e DACK siano da considerarsi attivi alti o attivi bassi;
 - La possibilità di effettuare trasferimenti da memoria a memoria;
 - La possibilità di utilizzare un timing compresso (che utilizza 2 cicli di bus per ogni ciclo di DMA anziché 4);
 - La possibilità di mantenere costante l'indirizzo di memoria sorgente, durante i trasferimenti da memoria a memoria, per riempire la zona di memoria destinazione con il valore di una stessa cella.
- **Request Register (RR):** è registro a 8 bit che permette di tenere traccia delle richieste di trasferimento in DMA effettuate via software. Un programma è in grado di richiedere un trasferimento in DMA relativo ad un canale (come se il dispositivo collegato a tale canale attivasse il segnale DREQ) settando il corrispondente bit di tale registro.
- **Mask Register (MR):** è un registro a 8 bit che permette di disabilitare selettivamente le richieste di trasferimento DMA relative ai canali.
- **Status Register (SR):** è un registro a 8 bit che mostra lo stato di ciascun canale. In particolare i 4 bit alti (bit DREQ) indicano, per ogni canale, se il segnale di richiesta di trasferimento DMA risulta attivo, mentre i 4 bit bassi (bit TC) indicano, per ogni canale, se il terminal count è stato raggiunto.
- **Temporary Register (TR):** è un registro su 8 bit utilizzato per memorizzare temporaneamente i byte letti durante i trasferimenti tra memoria e memoria in DMA;

6.5.5 Canali di DMA

L'8237 è dotato, per ogni canale, di un certo numero di registri interni che devono essere programmati, in fase di inizializzazione, con l'indirizzo di memoria a o da cui iniziare a trasferire i dati, la dimensione in byte del blocco di dati da trasferire e la modalità di trasferimento desiderata.



Ciascun canale dell'8237, la cui struttura è schematizzata nella Fig. 48, presenta i seguenti registri:

- **Base address register** (16 bit): registro contenente l'indirizzo di partenza in cui scrivere o da cui leggere il blocco di dati. Viene inizializzato dal driver del dispositivo associato al canale.
- **Current address register** (16 bit): registro contenente l'indirizzo di memoria in cui scrivere o da cui leggere la parola di dati per il ciclo di DMA corrente. Viene scritto all'inizializzazione del base address register, ed aggiornato (incrementando o decrementando) ad ogni ciclo di DMA eseguito per il canale.
- **Base word count register** (16 bit): registro contenente la dimensione in byte del blocco di dati da trasferire. Il numero di byte effettivamente trasferiti in DMA è pari a tale valore aumentato di uno. Viene inizializzato dal driver del dispositivo associato al canale.

- **Current word count register** (16 bit): registro contenente il numero residuo di byte da trasferire per completare il trasferimento dell'intero blocco. Viene scritto all'inizializzazione del base word count register, ed aggiornato (incrementando) ad ogni ciclo di DMA eseguito per il canale.
- **Mode register** (8 bit): registro di controllo che permette di definire per il canale: il tipo di trasferimento DMA da effettuare (lettura DMA o scrittura DMA), la modalità di trasferimento (come Single Mode o Block Mode), la direzione di scansione del blocco di memoria (in avanti o a ritroso dall'indirizzo di partenza) e l'abilitazione o meno della modalità *auto-initialize*.

La modalità di **auto-initialize** prevede la reinizializzazione automatica dei valori dei registri current address e current word count con i valori dei registri base. In tal modo è possibile utilizzare la stessa area di memoria per più trasferimenti in DMA che coinvolgono il dispositivo, senza reinizializzare manualmente il canale.

Quando il current word count register di un canale passa dal valore 0000H al valore FFFFH, si verifica la condizione di *terminal count* per il canale. Il bit TC dello status register relativo al canale viene settato e viene attivato il segnale di \overline{EOP} . Se per il canale non è stata impostata la modalità auto-initialize, questo viene automaticamente disabilitato settando il corrispondente bit del mask register. Sarà compito poi del programma driver, riabilitare tale canale dopo averlo opportunamente riprogrammato per un nuovo trasferimento.

Essendo il dispositivo 8237 collegato solamente agli 8 segnali bassi del data bus, la programmazione di ciascun registro da 16 bit deve avvenire in due cicli di bus: il primo per il byte meno significativo, il secondo per quello più significativo.

Essendo i registri count dei registri a 16 bit, con una singola programmazione ciascun canale è in grado di trasferire in DMA fino a 64 Kbyte di dati tra memoria centrale e il dispositivo periferico ad esso collegato.

Ciascun canale dell'8237 è dotato dei due segnali:

- **DREQ** (*DMA Request*): permette all'8237 di ricevere richieste di trasferimento in DMA da parte del periferico collegato al canale;
- **DACK** (*DMA Acknowledge*): permette all'8237 di selezionare il dispositivo collegato al canale che deve prendere parte al ciclo di DMA, quando la sua richiesta di trasferimento in DMA viene accolta;

6.5.6 Modalità di trasferimento

Il controllore del DMA 8237 supporta quattro modalità di trasferimento:

- **Single mode**: il DMA controller esegue un singolo ciclo di DMA dopodichè rilascia il controllo del bus alla CPU. Corrispondente alla modalità di funzionamento *cycle stealing*.
- **Block mode**: il DMA controller mantiene il controllo del bus fintanto che non termina il trasferimento dell'intero blocco di dati. Corrispondente alla modalità di funzionamento *burst transfer*.
- **Demand mode**: come il precedente tranne che il trasferimento può essere bloccato direttamente dal periferico, disattivando il segnale DREQ.
- **Cascade mode**: utilizzato quando al canale è collegato un ulteriore dispositivo 8237 slave. In tale modalità non vengono generati i segnali di controllo e indirizzi relativi al canale.

Operando in *single mode* o in *block mode*, il segnale $DREQ_i$ in servizio deve essere mantenuto attivo, da parte del periferico, fino a quando non viene attivato il segnale $DACK_i$. Per i

trasferimenti in *demand mode*, il segnale $DREQ_i$ deve essere mantenuto attivo, da parte del periferico, fino a quando questo ha a disposizione dati da trasferire in DMA.

Un esempio di trasferimento in block mode è riportato nella Fig. 49.

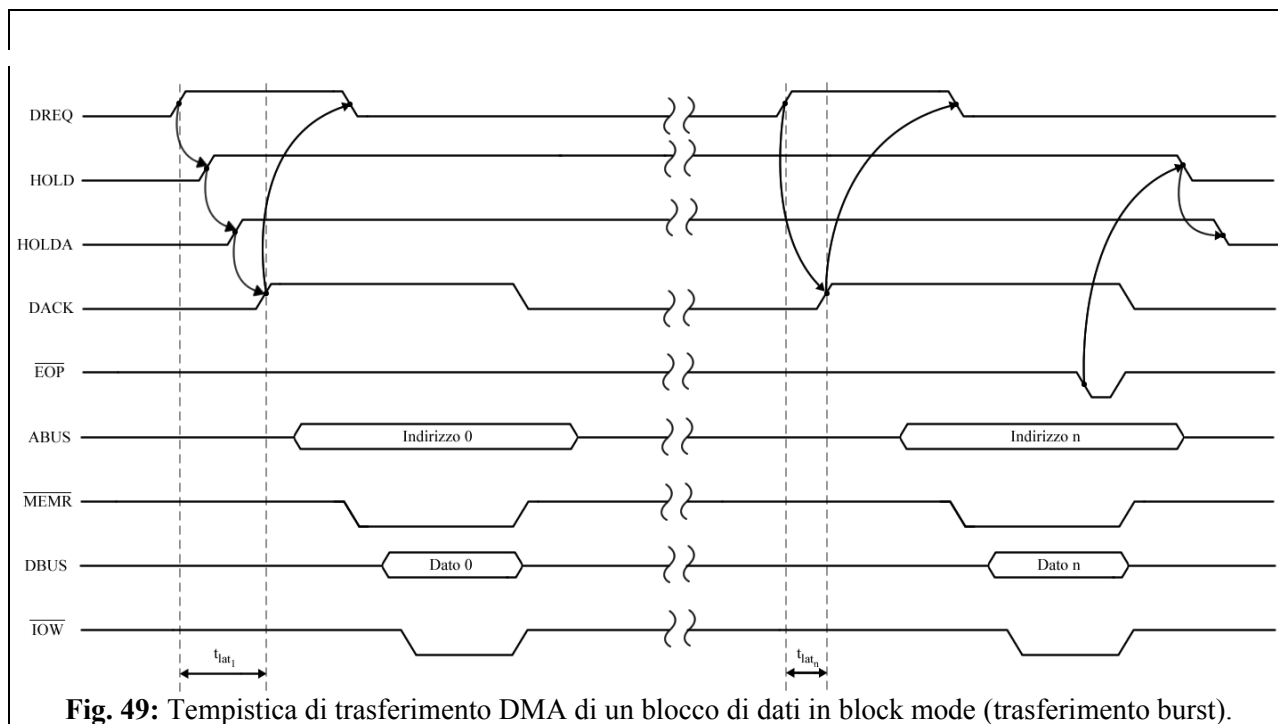


Fig. 49: Tempistica di trasferimento DMA di un blocco di dati in block mode (trasferimento burst).

6.5.7 DMA controller 8237 in cascata

Più controllori 8237 possono essere collegati in cascata, secondo uno schema master slave, per espandere il numero di canali DMA che possono essere gestiti. I segnali di HRQ e $HLDA$ del dispositivo 8237 slave vengono connessi con i segnali $DREQ_i$ e $DACK_i$ dell' i -esimo canale del controllore master. Ciò permette di propagare le richieste di trasferimento DMA ricevute dallo slave all'interno del dispositivo master, in modo tale da mantenere una relazione di priorità tra i canali dei due dispositivi.

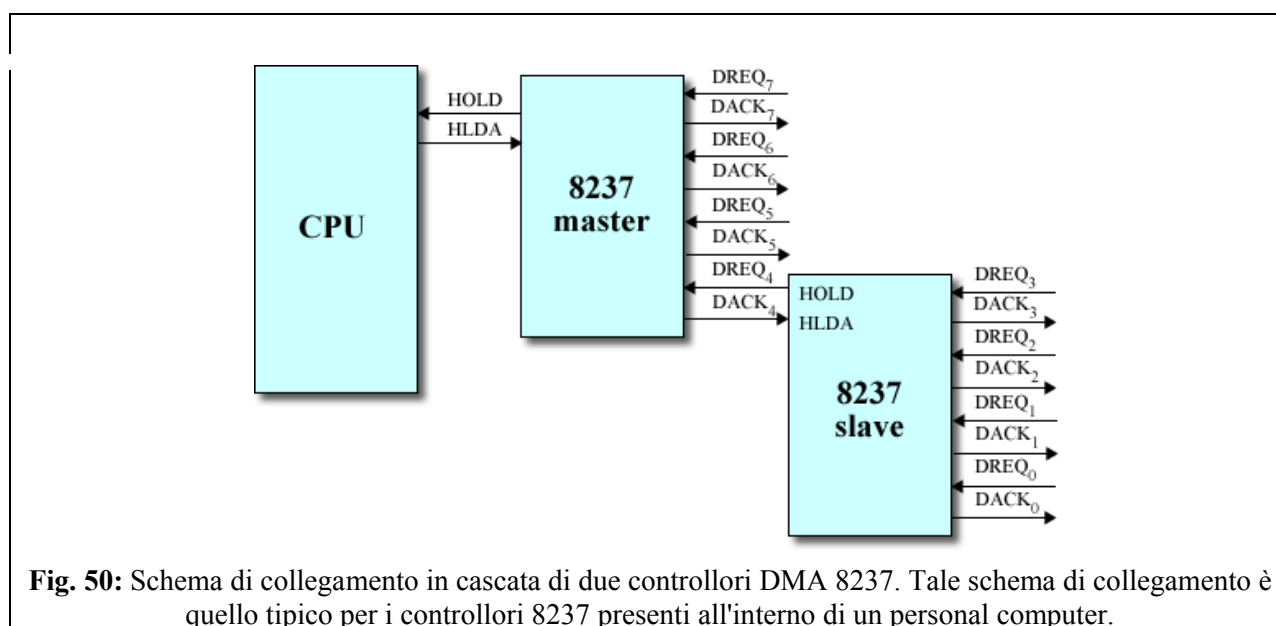
Dal momento che il canale a cui si collega il dispositivo 8237 slave viene utilizzato solo al fine di mantenere l'ordinamento di priorità tra le richieste, il master non deve generare i segnali di

controllo ed indirizzo in seguito alla ricezione di una richiesta su tale canale. Ciò è ottenuto programmando tale canale per operare in *cascade mode*.

Il dispositivo master, in seguito alla ricezione di una richiesta sul canale a cui è collegato lo slave, negozia solamente il controllo del bus con la CPU e, una volta ottenuto, comunica, attraverso il segnale $DACK_i$, la disponibilità del bus al dispositivo slave. Questo, ricevuto il controllo del bus dal master, genererà i segnali di controllo ed indirizzo per il particolare ciclo di DMA ed attiverà il dispositivo periferico richiedente attraverso il segnale DACK del canale ad esso relativo.

6.6 Gestione dei trasferimenti in DMA nei PC

Nei personal computer, i trasferimenti in DMA sono gestiti tramite due controllori 8237 in cascata integrati nel chipset. Tali controllori sono collegati come in Fig. 50 e permettono di gestire fino a 7 canali di DMA.



Il canale 4 del master è collegato ad un dispositivo 8237 slave, è pertanto programmato per operare in *cascade mode*. Le priorità associate sono fisse e decrescenti dal canale 0 al canale 7, ma possono essere riprogrammate per diventare priorità rotanti.

I canali del dispositivo slave, dallo 0 al 3, sono cablati per operare con dispositivi a 8 bit, mentre quelli del dispositivo master, dal 5 al 7, sono cablati per operare con dispositivi a 16 bit. La dimensione massima del blocco che può essere trasferito in DMA è pertanto di 64 Kbyte per i canali del dispositivo slave e 128 Kbyte per i canali del dispositivo master.

In ogni sistema basato sull'IBM-PC, l'hardware di gestione dei trasferimenti in DMA è localizzato alle stesse porte di I/O. In particolare, i dispositivi DMA controller 8237 master e slave e i page register associati ai canali sono raggiungibili attraverso i seguenti indirizzi di I/O:

Dispositivo	Indirizzo di I/O di partenza
DMA controller 8237 slave	0x00H
DMA controller 8237 master	0xC0H
Page registers	0x80H

Riferimenti

- [1] Slide del corso di Architetture dei sistemi di elaborazione. Marco Mezzalama. A.A. 2010-2011.
- [2] Videolezioni del corso di Architetture dei sistemi di elaborazione Marco Mezzalama. A.A. 2010-2011.
- [3] The Intel Microprocessors, Barry Brey, Prentice Hall, 1997
- [4] Fundamentals of computer architecture and design, Sivarama Dandamudi, Springer, 2002
- [5] The Indispensable PC Hardware Handbook, Hans-Peter Messmer, Addison-Wesley, Third Edition
- [6] www.wikipedia.it
- [7] <http://www.lammertbies.nl/comm/info/serial-uart.html>
- [8] http://en.wikibooks.org/wiki/Serial_Programming/8250_UART_Programming
- [9] <http://www.sharpmz.org/mz-700/8253ovview.htm>