

## 8259: controllore programmabile delle interruzioni

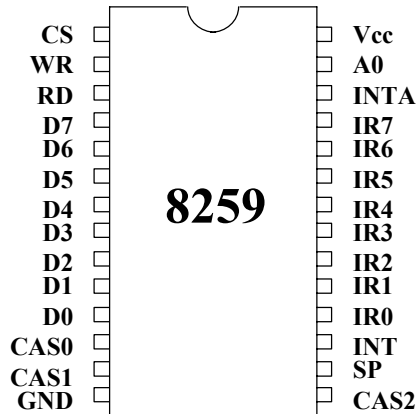
---

### Introduzione

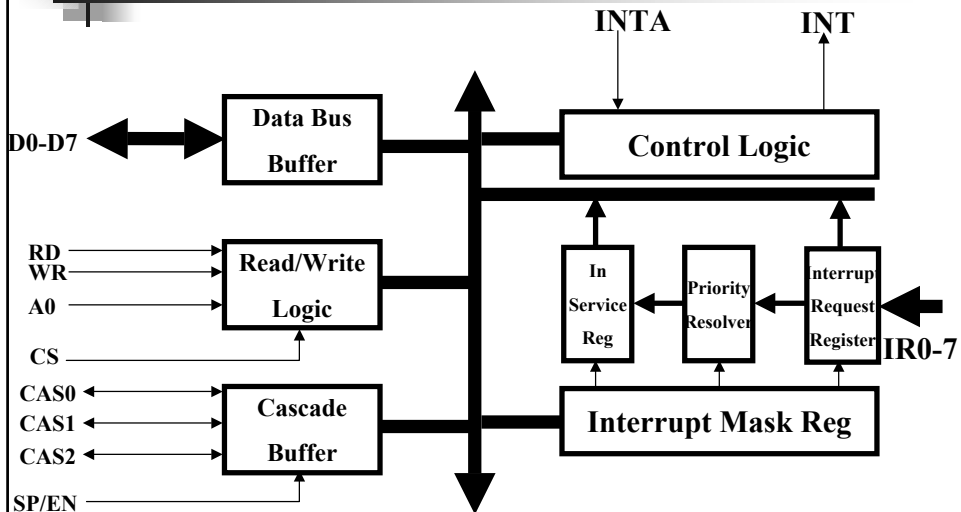
---

- L'8259 è stato progettato per minimizzare il software ed i tempi di risposta per la gestione di livelli multipli di interrupt a diversa priorità.
  - Il dispositivo è un chip LSI contenuto in un DIP da 28 pin.
  - Gestisce fino a 8 livelli di interrupt vettorizzati.
  - Permette la gestione di più controllori di interrupt in cascata fino ad un massimo di 64 livelli di interrupt.
  - Permette la programmazione di diversi modi di gestione delle priorità tra i diversi livelli.
  - È formato da circuiteria statica (assenza dell'input di clock).

# Il chip



# Diagramma a blocchi



## Sequenza di interrupt

- Una linea di richiesta di interrupt sale alta settando il corrispondente bit del registro IRR.
- L'8259 valuta le richieste e manda un segnale di INT alla CPU.
- La CPU conferma la richiesta ed invia un primo segnale di INTA.
- La richiesta a priorità più alta viene selezionata settando l'opportuno bit del registro ISR e resettando il corrispondente bit del registro IRR.

## Sequenza di interrupt

- La CPU invia un secondo impulso di INTA
- L'8259 invia sul data bus il codice (1 byte) del dispositivo che ha fatto richiesta di interruzione
- Il ciclo di interrupt è concluso resettando il bit ISR. In modo AEIOI (automatic end of interrupt) ciò avviene in modo automatico; in alternativa il ciclo di interrupt deve essere terminato con una esplicita istruzione di EOI (end of interrupt).

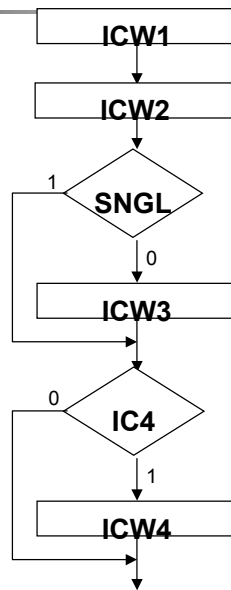
## Programmazione dell'8259

- L'8259 è programmato attraverso due tipi di parole di comando pilotate dalla CPU:
  - Initialization Command Words (ICWs)
  - Operation Command Words (OCWs).
- L'ordine delle ICWs è fisso e normalmente vengono inviate una volta sola in fase di inizializzazione, mentre le OCWs possono essere inviate singolarmente in qualunque fase del programma.

## ICWs

- L'inizializzazione dell'8259 viene fatta attraverso una sequenza di parole di comando.
- Questa sequenza è riconoscibile perché il primo dato (ICW1) è caratterizzato dal segnale di indirizzo  $A0 = 0$  ed il bit di dato  $D4 = 1$ .
- Quando la CPU invia una sequenza di ICW viene resettato il registro di maschera IMR.

# ICWs



# ICW1

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

- LTIM (Level Triggered Mode)
- SNGL: modo Single o Cascade
- IC4: presenza della parola ICW4
- D4 = 1
- X: bit usati solo per CPU della famiglia 8080/85.

## ICW1

LTIM

0	1
triggerato sui fronti	triggerato sui livelli

IC4

0	1
NO ICW4	ICW4 SI

SNGL

0	1
8259 in cascata	single chip 8259

## ICW2

A0 D7 D6 D5 D4 D3 D2 D1 D0

1	T7	T6	T5	T4	T3	X	X	X
---	----	----	----	----	----	---	---	---

- Con la parola ICW2 la CPU determina i tipi di interrupt corrispondenti agli 8 segnali di richiesta di interruzione:
  - T3-T7: 5 bit alti dell'indirizzo del vettore degli interrupt.
  - X: settati in accordo alla richiesta di interrupt su IR0-7.

# ICW3

	A0	D7	D6	D5	D4	D3	D2	D1	D0
<b>Dispositivo Master</b>	1	S7	S6	S5	S4	S3	S2	S1	S0

Ciascun bit della parola ICW3 specifica se il corrispondente segnale IR è un 8259 slave (bit a 1) oppure un normale dispositivo periferico (bit a 0).

	A0	D7	D6	D5	D4	D3	D2	D1	D0
<b>Dispositivo Slave</b>	1	0	0	0	0	0	ID2	ID1	ID0

Specifica allo slave il numero del livello IR master a cui è collegato.

# ICW4

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	SFNM	BUF	M/S	AEOI	μPM

- SFNM: Special fully nested mode.
- BUF & M/S: Buffer Mode.
- AEOI: Automatic end of interrupt.
- μPM: Microprocessor mode.

## ICW4

$\mu$ PM

0	1
8080/85 mode	8086 mode

AEOI

0	1
Normal EOI	Automatic End Of Interrupt

SFNM

0	1
No Special Fully Nested Mode	Special Fully Nested Mode

## Buffered mode

BUF

M/S

0	1	1
X	0	1
No modo buffered	Buffered mode slave	Buffered mode master

- Nel modo buffered il pin SP/EN è un pin di Output che permette di abilitare i buffer tranceivers quando l'8259 fa un'operazione di scrittura sul Data Bus.
- In modo non buffered il bit M/S specifica via s/w se il chip è master o slave.

## OCW1

A0 D7 D6 D5 D4 D3 D2 D1 D0

1	M7	M6	M5	M4	M3	M2	M1	M0
---	----	----	----	----	----	----	----	----

- OCW1 permette di caricare il registro IMR.
  - Ad ogni bit della parola OCW1 corrisponde un bit nel registro IMR.
  - Settando il bit  $M_i$  ad 1 si setta il bit  $IMR_i$  e dunque si maschera il canale di interrupt  $IR_i$ .

## OCW2

A0 D7 D6 D5 D4 D3 D2 D1 D0

0	R	SL	EOI	0	0	L2	L1	L0
---	---	----	-----	---	---	----	----	----

- I bit R, SL e EOI controllano i modi di gestione della rotazione delle priorità e dell'end of interrupt.
- I bit L2-L0 specificano un particolare canale di interrupt.

## OCW2

R	SL	EOI	
0	0	1	Non Specific EOI command
0	1	1	Specific EOI command
1	0	1	Rotate on Non Specific EOI command
1	0	0	Rotate in AEOI mode (set)
0	0	0	Rotate in AEOI mode (clear)
1	1	1	Rotate on Specific EOI command
1	1	0	Set Priority Command
0	1	0	No operation

## OCW3

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	ESMM	SMM	0	1	P	RR	RIS

- I bit ESMM e SMM permettono di settare o resettare lo Special Mask Mode.
- I bit RR e RIS permettono di gestire la lettura dei registri interni IRR ed ISR.
- Il bit P permette di settare o resettare il Poll Command.

## OCW3

Read Register  
Command

RIS	X	0	1
RR	0	1	1
	No action	Read IRR	Read ISR

Special Mask  
Mode

SMM	X	0	1
ESMM	0	1	1
	No action	Reset Special Mask	Set Special Mask

## Fully nested mode

- Modo di funzionamento di default per l'8259.
- Le richieste di interruzione sono ordinate per livelli di priorità da 0 a 7. Il livello 0 è il livello a maggiore priorità.
- Quando la CPU abilita una richiesta di interrupt (mediante un primo impulso di INTA), l'8259 calcola la richiesta a priorità più alta; il corrispondente indice del vettore delle interruzioni è posto sul Data Bus ed il corrispondente bit del registro ISR è settato.
- Il bit in ISR rimane settato finché la CPU invia un comando di EOI immediatamente prima di ritornare dalla routine di servizio dell'interruzione; se invece il bit AEOI è settato il bit in ISR è resettato automaticamente dopo il fronte di salita del secondo segnale di INTA.

## Fully nested mode

---

- Fintanto che il bit in ISR è settato, tutte le successive richieste di interrupt a priorità più bassa sono disabilitate. Solo le richieste a priorità più alta generano una richiesta di interruzione.

## Automatic End of Interrupt Mode: AEOI

---

- Se il bit AEOI della parola ICW4 è stato fissato ad 1, l'8259 opera in modo AEOI.
- In modo AEOI l'8259 resetta automaticamente il registro ISR dopo il fronte di salita del secondo segnale di INTA.

## End of Interrupt: EOI

- Se il bit AEOI del registro ICW4 è fissato a 0, allora per resettare il bit IS nel registro ISR occorre un esplicito comando di EOI.
- È buona norma inviare tale comando come ultima istruzione prima di una istruzione di IRET.
- In una cascata di 8259 occorre inviare due comandi di EOI, uno per il master ed uno per lo slave servito.
- Ci sono due diverse forme di EOI:
  - Specific End of Interrupt
  - Non Specific End of Interrupt.

## Non Specific EOI

- Valido in un modo di funzionamento che conserva il fully nested mode, in cui dunque è mantenuto un ordine di priorità statico.
- Quando è inviato un comando di EOI non specifico il bit a priorità più alta nel registro ISR, corrispondente all'ultimo livello di interrupt abilitato e servito, viene resettato.
- Un Non Specific EOI viene inviato attraverso la parola OCW2 ( $R = 0$ ,  $SL = 0$ ,  $EOI = 1$ ).

## Specific EOI

---

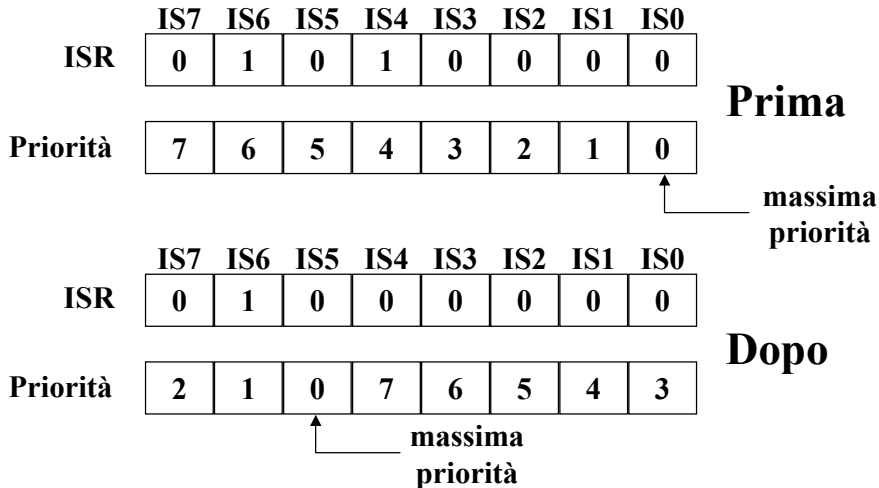
- In un modo di funzionamento diverso dal fully nested mode la priorità non è più statica e dunque l'8259 non ha nessun modo per riconoscere quale è l'ultimo livello di interruzione abilitato.
- Inviando un comando di EOI occorre specificare il livello di interruzione che si vuole resettare nel registro ISR.
- Uno Specific EOI viene inviato attraverso la parola OCW2 ( $R = 0$ ,  $SL = 1$ ,  $EOI = 1$  ed i bit L0-L2 con il valore binario corrispondente al livello che si vuole resettare).

## Rotazione automatica delle priorità

---

- In molte applicazioni può succedere che i diversi dispositivi di I/O abbiano uguale priorità.
- In questi casi è preferibile non avere livelli di priorità statici, ma avere livelli di priorità rotanti.
- Un dispositivo appena viene servito riceve il livello di priorità più basso.

## Priorità rotante



## Rotazione e EOI

- A seconda che l'8259 sia in modo AEOI oppure in modo EOI occorre programmare l'8259 con parole OCW2 differenti.
- La combinazione  $R = 1$ ,  $SL = 0$ ,  $EOI = 1$  forza la rotazione di priorità in modo EOI.
- La combinazione  $R = 1$ ,  $SL = 0$ ,  $EOI = 0$  forza la rotazione di priorità in modo AEOI.
- La combinazione  $R = 0$ ,  $SL = 0$ ,  $EOI = 0$  resetta la rotazione di priorità ed impone la normale priorità fissa in modo AEOI.

## Rotazione di priorità specifica

- Il programmatore può cambiare i livelli di priorità specificando nell'OCW2 il livello a priorità più basso mediante i bit L2-L0 avendo la seguente combinazione di OCW2:  $R = 1$ ,  $SL = 1$ .
- In modo EOI è possibile inviare un comando di EOI specifico e contemporaneamente ruotare la priorità mediante la combinazione di OCW2  $R = 1$ ,  $SL = 1$ ,  $EOI = 1$  e  $L0-L2 =$  livello IR di cui si vuole resettare il bit IS forzandolo alla priorità più bassa.

## Maschera delle interruzioni

- Ogni livello di richiesta di interruzione IR può essere mascherato attraverso la programmazione della parola OCW1.
- Ogni bit di OCW1 disabilita il corrispondente canale IR di interruzione se settato ad 1.

## Special mask mode

- Il registro di mascheramento IMR può essere utilizzato in un modo diverso da quello tradizionale.
- I bit settati in IMR *disabilitano* il livello IR corrispondente da ulteriori richieste di interrupt ed *abilitano* gli interrupt *di tutti gli altri livelli*.
- Questo può essere utile quando in modo EOI una richiesta è stata abilitata ed un comando di EOI non ha ancora disabilitato il bit IS. In questo caso le richieste a priorità più basse sono disabilitate. Utilizzando il modo di maschera speciale ho l'abilitazione di tutti i livelli aventi valore 0 in IMR.
- Il modo di maschera speciale è settato e resettato attraverso la parola OCW3.

## Poll command

- È possibile gestire l'8259 in *polling*.
- Il modo polling è fissato settando il bit P della parola OCW3.
- L'8259 interpreta la successiva istruzione di lettura come acknowledge di interrupt.
- L'8259 invia sul data bus la *poll word*:
  - il bit I è posto ad 1 se vi è una richiesta di interrupt;
  - i bit W2-W0 rappresentano il livello IR a priorità più alto richiedente servizio di interruzione.
- La CPU non esegue la solita sequenza di segnali di INTA.

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

## Lettura registri

- È possibile leggere lo stato dell'8259 mediante la lettura dei registri interni.
- Possono essere letti i registri IRR, ISR ed IMR.
- IRR può essere letto nel primo ciclo di lettura (all'indirizzo avente  $A0 = 0$ ) successivo ad una OCW3 con  $RR = 1$  e  $RIS = 0$ .
- ISR può essere letto nel primo ciclo di lettura (all'indirizzo avente  $A0 = 0$ ) successivo ad una OCW3 con  $RR = 1$  e  $RIS = 1$ .
- Non è necessario inviare una OCW3 prima di ogni lettura di registro. L'8259 memorizza l'ultima OCW3 e dunque se si vuole leggere sempre lo stesso registro non è necessario cambiare l'OCW3.
- Per default il registro leggibile è IRR.

## Lettura IMR

- Per leggere il registro *IMR* non è necessaria nessuna parola OCW3: è sufficiente eseguire un ciclo di lettura all'indirizzo avente  $A0 = 1$ .

## Interrupt sensibile ai fronti o ai livelli

- È possibile rendere i livelli di interrupt sensibili al fronte od ai livelli mediante il bit 3 (LTIM) in ICW1.
- Se  $LTIM = 0$ , le richieste di interruzione saranno riconosciute da una transizione da 0 ad 1 su un ingresso di IR. Il segnale IR può rimanere alto senza generare nessun'altra richiesta di interrupt.
- Se  $LTIM = 1$ , le richieste di interruzione saranno riconosciute da un livello alto su un ingresso di IR. La richiesta di interruzione va rimossa prima del comando di EOI per prevenire un'altra richiesta di interruzione.

## Special fully nested mode

- Questo modo è usato nel caso di una cascata di 8259, quando si vuole conservare la priorità all'interno di ciascuno slave.
- Quando una richiesta di interruzione da parte di uno slave è in servizio, questo slave non è bloccato dal master, ma possibili richieste da livelli a priorità più alta provenienti dallo stesso slave saranno riconosciute dal master.
- Uscendo dalla routine di servizio prima di inviare i comandi di EOI al master occorre leggere il registro ISR dello slave (dopo aver inviato un EOI non specifico). Se il registro ISR è zero allora si può inviare un EOI non specifico al master.



## 8259 in cascata: funzionamento

- Quando uno slave ha una richiesta di interruzione su un suo livello, invia una richiesta al pin IR del master mediante il segnale di INT.
- Tale richiesta è inoltrata alla CPU (nel caso in cui tale livello non sia mascherato e sia il livello richiedente a massima priorità).
- Quando la CPU invia il primo segnale, il master setta l'opportuno bit del registro ISR, pulisce il corrispondente bit in IRR e verifica, leggendo il registro ICW3, se tale richiesta proviene da uno slave oppure no.
- Se la richiesta non proviene da uno slave, il master invia sul data bus il contenuto di ICW2 corrispondente all'indice della routine di servizio dell'interruzione.

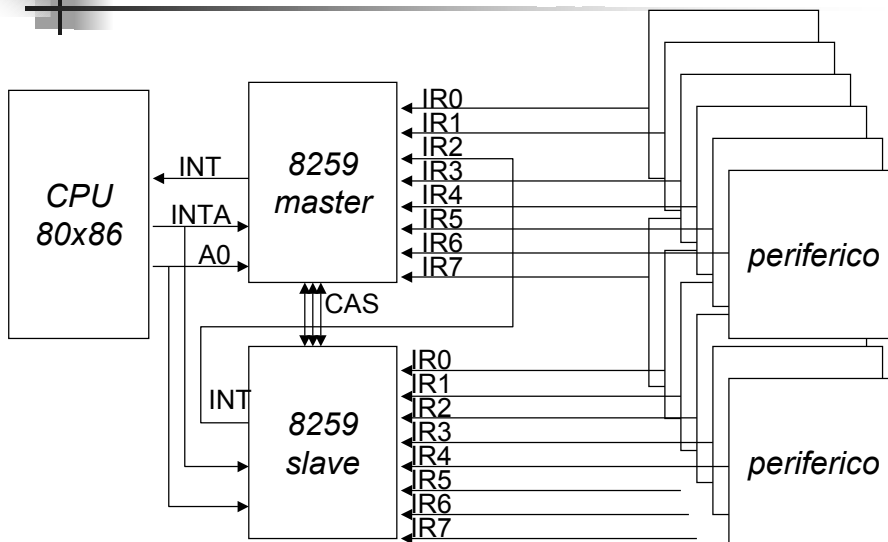
## 8259 in cascata: funzionamento

- Se la richiesta proviene da uno slave, il master piazza il numero del livello IR sulle linee CAS.
- Il segnale di INTA è ricevuto da tutti gli slave. Ciascuno slave confronta il proprio ID con il numero letto sulle linee CAS; se c'è corrispondenza, riconosce che il segnale di INTA è diretto a lui.
- Lo slave selezionato setta al suo interno l'opportuno bit ISR, pulisce il corrispondente bit IRR e pone sul data bus l'indirizzo della vector table contenuto nella propria ICW2.
- In un sistema master/slave, occorre mandare un duplice comando di EOI sia al master che allo slave interessato.

# Gestione degli interrupt nei PC

- Nei personal computer le interruzioni mascherabili sono gestite da due dispositivi 8259A (o compatibili), collegati in modalità master-slave.
- Il P.C. è quindi in grado di ricevere 15 tipi di interruzioni diverse
  - 7 sul master (l'ottavo serve per il collegamento in cascata con lo slave)
  - 8 sullo slave.
- Inoltre vi è una fonte di interruzioni non mascherabili, collegata alla logica di rivelamento degli errori di parità nella memoria dinamica.

## Connessioni



## Configurazione

- I due 8259A sono collegati in modalità master-slave. Il canale usato per collegare lo slave è l'IR2 del master.
- Nei P.C., la configurazione dei dispositivi è la seguente:
  - No Special Fully Nested Mode, Normal Nested Mode
  - Manual EOI (no Auto EOI)
  - Offset (Master) = 08h
  - Offset (Slave) = 70h
  - Indirizz1 I/O master = 20h e 21h
  - Indirizz1 I/O slave = a0h e a1h

## Esempio

- Per programmare i due dispositivi si usano le seguenti control word:
  - Master:
    - ICW1: 00010001 (cascaded, icw4 needed)
    - ICW2: 00001000 (offset 08h)
    - ICW3: 00000100 (slave at IR2)
    - ICW4: 00000001 (no SFNM, no BUF, no AEOI)
  - Slave:
    - ICW1: 00010001 (cascaded, icw4 needed)
    - ICW2: 01110000 (offset 70h)
    - ICW3: 00000010 (slave ID: 2)
    - ICW4: 00000001 (no SFNM, no BUF, no AEOI)

## Inizializzazione

- La programmazione delle control word precedenti può avvenire come segue:  
mov al, 11h ; ICW1 master & slave  
out 20h, al  
out 0a0h, al  
mov al, 08h ; ICW2 master  
out 21h, al  
mov al, 70h ; ICW2 slave  
out 0a1h, al

## Inizializzazione (segue)

- mov al, 04h ; ICW3 master  
out 21h, al  
mov al, 02h ; ICW3 slave  
out 0a1h, al  
mov al, 01h ; ICW4 master & slave  
out 21h, al  
out 0a1h, al
- N.B. non è necessario inizializzare gli 8259A, in quanto ciò viene già fatto dal sistema operativo.

## Uso degli 8259A

- Durante il normale funzionamento, il sistema operativo, o i drive delle periferiche, solitamente accedono agli 8259A per i seguenti motivi:
  - abilitare un canale di interruzione
  - disabilitare un canale di interruzione
  - inviare un comando Non-Specific End-Of-Interrupt (EOI).

## Abilitare/Disabilitare Interrupt

- Occorre inviare un comando OCW1: i bit che sono posti ad 1 mascherano (disabilitano) il canale IR corrispondente.
- Esempio: disabilitare IR3 sul Master  
in al, 21h ; leggi IMR  
or al, 00001000 ; maschera IR3  
out 21h, al ; OCW1

## End-Of-Interrupt

- Il comando EOI deve essere inviato dalla CPU all'8259A al termine del servizio di ciascuna richiesta di interruzione, al fine di resettare il bit corrispondente del registro ISR. Ciò si ottiene inviando una parola di controllo OCW2:  
out 20h, 0010000b ; EOI master
- Oppure:  
out 0a0h, 0010000b ; EOI slave
- N.B. quando un interrupt è gestito dallo slave, è necessario inviare EOI sia al master che allo slave.

## Tipi di interruzioni

- Visti i valori di offset con cui vengono programmati gli 8259A nei P.C., la corrispondenza tra livello di interruzione e tipo di interrupt che la CPU esegue è la seguente:

8259A	80x86	bus ISA
IR0 master	INT 08h	IRQ0
... ..		
IR7 master	INT 0Fh	IRQ7
IR0 slave	INT 70h	IRQ8
... ..		
IR7 slave	INT 77h	IRQ15

## Periferiche

L'assegnazione "standard" delle periferiche è:

	Master	Slave
IR0	8254 timer 0	Real Time Clock
IR1	tastiera	ex-IR2 master, LAN
IR2	8259 slave, VGA	riservato
IR3	COM2	riservato
IR4	COM1	mouse (non seriale)
IR5	LPT2	coprocessore 80x87
IR6	floppy	hard disk
IR7	LPT1	riservato

## Interrupt Service Routine

- Quando la CPU riceve una richiesta di interruzione (ed il flag IF vale 1), essa inizia ad eseguire la procedura di servizio dell'interruzione (ISR) corrispondente al tipo di interruzione ricevuta.
- Tale procedura è una normale procedura assembler, che però deve soddisfare alcuni requisiti particolari, per garantire la convivenza con il programma in esecuzione e con il sistema operativo.

## Ambiente della ISR

- A differenza di una normale procedura, una ISR viene attivata all'insaputa del programma attualmente in esecuzione.
- Quindi la ISR viene attivata:
  - con DS, ES, SS del programma interrotto
  - con lo stack del programma interrotto
  - con i registri del programma interrotto.
- N.B. Il programma interrotto non è necessariamente il "main", in quanto l'interruzione potrebbe esser giunta mentre il programma richiama le librerie C, il DOS oppure il BIOS.

## ISR - Regola 1: la procedura è far

- Una chiamata ad una ISR avviene sempre con una call di tipo FAR, ossia ponendo sullo stack CS ed IP (oltre ai flag).
- Il ritorno dalla procedura dovrà avvenire con l'istruzione speciale IRET, anziché la solita RET.
- Se è necessario chiamare altre procedure, assicurarsi che la chiamata sia di tipo FAR, altrimenti il CS potrebbe non essere correttamente inizializzato e ripristinato.

## Esempio

```
isr_mouse    PROC FAR
              . . .
              IRET
isr_mouse    ENDP
```

## ISR - Regola 2: preservare tutto

- Una ISR deve garantire al programma interrotto che tutti i registri utilizzati vengano preservati. Occorre salvare, con PUSH e POP, tutti i registri utilizzati in qualche modo.
- Uniche eccezioni: il contatore di programma CS:IP, ed il registro dei flag, che vengono automaticamente ripristinati dall'istruzione IRET.

## Esempio

---

```
isr_mouse proc far
    PUSH AX
    . . .
    POP AX
    iret
isr_mouse endp
```

## ISR - Regola 3: non abusare dello stack

---

- Poiché la ISR usa lo stack del programma chiamante, non può allocare troppi dati sullo stack.
- Ad esempio, il BIOS garantisce che il proprio stack abbia spazio per 4-6 operazioni PUSH, ma non di più.
- Nel caso in cui la ISR debba usare maggiori risorse di stack, deve caricare SS:SP con valori corrispondenti ad un proprio stack privato.
- La situazione risulta peggiorata se la ISR viene a sua volta interrotta.

## ISR - Regola 4: non chiamare DOS e BIOS

---

- Poiché la ISR potrebbe essere eseguita mentre il DOS o il BIOS sono in esecuzione, e poiché DOS e BIOS non sono stati scritti per sistemi operativi multiutente, non è possibile attivare una seconda chiamata ad una funzione DOS o BIOS mentre un'altra è ancora in corso.
- Sono quindi vietate le chiamate a INT 21h ad altri servizi DOS o BIOS. Sono altresì permesse le chiamate a procedure assembler o C, purché a loro volta non richi amino DOS o BIOS.
- N.B. le funzioni della libreria C fanno ampio uso dei servizi del DOS!

## ISR - Regola 5: lasciarsi interrompere?

---

- Se una ISR di un interrupt a bassa priorità ha un tempo di esecuzione elevato, è opportuno che essa sia interrompibile da interrupt a priorità maggiore.
- Per far ciò, è sufficiente eseguire l'istruzione STI.
- Altrimenti il flag IF verrà ripristinato dall'istruzione IRET.

## Esempio

```
isr_mouse proc far
    push ax
    . . . ; non interrompibile
    STI
    . . . ; interrompibile
    pop ax
    iret
isr_mouse endp
```

*opzionale*

## ISR - Regola 6: ricordarsi dell'EOI

- Prima di ritornare dall'ISR, accertarsi di abilitare future interruzioni inviando il comando EOI all'8259.

## Esempio

```
isr_mouse proc far
    push ax
    . . .
    OUT 0a0h, 20h ; EOI slave
    OUT 20h, 20h ; EOI master
    pop ax
    iret
isr_mouse endp
```

## ISR - Regola 7: comunicare con il programma

- La routine ISR dovrà necessariamente comunicare con il programma principale.
- Il modo più semplice è di utilizzare alcune variabili globali, che verranno controllate periodicamente dal programma principale.
- Nell'accedere a variabili globali, ricordare di caricare il valore corretto nel registro DS.

## Esempio

```
isr_mouse proc far
    push ax
    PUSH DS
    MOV AX, SEG pippo
    MOV DS, AX
    . . .
    out 20h, 20h ; EOI master
    POP DS
    pop ax
    iret
isr_mouse endp
```

## ISR - Regola 8: comunicare con il periferico

- Infine la Interrupt Service Routine dovrà prestare servizio al periferico che l'ha invocata. Queste operazioni dipendono dal periferico, ma in generale le istruzioni centrali (... negli esempi) conterranno:
  - lettura dello/degli registro/i di stato
  - controllo di eventuali errori
  - lettura del/dei dato/i ricevuto/i, oppure
  - scrittura del/dei dato/i da trasmettere
  - eventuale scrittura del/dei registro/i di controllo.
- Sono spesso istruzioni molti simili a quelle usate nella gestione in polling.

## Attivazione della ISR

- Per far sì che la ISR venga richiamata, è necessario:
  - scrivere il suo indirizzo nella Interrupt Vector Table
  - abilitare l'8259A (master o slave) ad inviare tale interruzione
  - eventualmente, abilitare gli eventuali latch che nei P.C. bloccano alcuni tipi di interruzioni
  - abilitare il periferico a generare le interruzioni.
- Da questo momento in avanti, la ISR verrà attivata ad ogni richiesta del periferico, purché la CPU abbia il flag  $IF=1$ .

## Interrupt Vector Table

- La Interrupt Vector Table (IVT) risiede a partire dall'indirizzo 0000:0000 e contiene una doubleword (4 byte) per ciascun tipo di interrupt. L'indirizzo della IVT associato a ciascun tipo di interrupt è:  
$$\text{indirizzo} = 4 * \text{tipo di interrupt}$$
- Essa contiene l'indirizzo (SEGMENTO:OFFSET) iniziale della ISR del tipo di interrupt, usato per effettuare l'attivazione della ISR come chiamata di tipo FAR.
- È opportuno scrivere nella IVT solamente con interruzioni disabilitate!

## Esercizio

- Si programmi la IVT in modo da richiamare la ISR `isr_mouse` non appena giunge un interrupt dalla porta seriale COM1.
- La COM1 è collegata alla linea IR4 dell'8259A master, per cui corrisponde ad un tipo di interrupt 0Ch.
- L'indirizzo corrispondente nella IVT è:  $0Ch * 4 = 30h = 48$ .
- È opportuno salvare in due locazioni di memoria l'indirizzo del precedente gestore dell'interruzione INT 0Ch.

## Soluzione

```
mov ax, 0
mov es, ax ; segmento ES=0000
mov bx, 0Ch ; tipo di interrupt
shl bx, 2   ; moltiplica per 4: offset IVT
            ; salva l'indirizzo della vecchia ISR
mov ax, es:[bx]
mov old_int0C_offset, ax
mov ax, es:[bx+2]
mov old_int0C_segment, ax
```

## Soluzione

---

```
cli ; disabilita interrupt prima di
    ; scrivere nella IVT
    ; carica l'indirizzo della nuova ISR
mov ax, OFFSET isr_mouse
mov es:[bx], ax
mov ax, SEG isr_mouse
mov es:[bx+2], ax
sti ; riabilita interrupt
```

## Terminazione del programma

---

- Ogni volta che si programma una ISR, è fondamentale ricordare di disabilitarla prima di tornare al sistema operativo.
- Occorre in particolare:
  - disabilitare il dispositivo a generare interruzioni
  - mascherare l'interruzione nell'8259A
  - ripristinare il contenuto della IVT.

## Sviluppo e Debugging

- Il debug delle procedure ISR è decisamente complesso, perché:
  - è impossibile eseguirle passo-passo con il debugger
  - vengono attivate ogni volta con SS e DS potenzialmente diversi
  - non possono usare INT 21h o chiamate C (tipo printf) per emettere informazione di debugging.

## Suggerimento: sviluppo con CALL

- È possibile sviluppare parte del codice di una ISR richiamandola esplicitamente con un'istruzione CALL, anziché "agganciarla" ad un'interruzione prima che la procedura sia completamente verificata.
- Occorre però ricordare che la chiamata deve essere di tipo FAR e deve porre sullo stack anche i flag (e quindi non può avvenire da C). In assembler si può chiamare la ISR con una sequenza del tipo:  
pushf  
call far ptr isr\_mouse

## Suggerimento: nessun parametro

- Poiché la ISR "vera" verrà attivata in modo autonomo dall'interrupt, essa potrà comunicare con il programma principale solamente attraverso variabili globali condivise.
- È opportuno quindi che, nella fase di sviluppo, il programma principale non passi alcuna informazione sullo stack o nei registri.

## Suggerimento: debug in memoria video

- Per verificare che la ISR venga chiamata, ed eventualmente per capire fin dove procede e quali istruzioni esegue, può essere utile far comparire a video dei caratteri che evidenzino la situazione.
- Ciò è possibile scrivendo direttamente nella memoria video (a partire dal B800:0000).
- Esempio:  
mov ax, 0b800h  
mov es, ax  
inc word ptr es:[0] ; modifica il  
; carattere in alto a sinistra