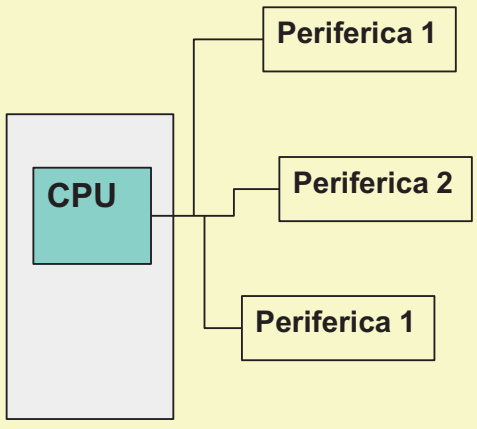
La gestione della I/O e le interruzioni

Un computer è un sistema complesso costituito da:
- una Unità Centrale di Elaborazione (**CPU**)
- e da un insieme più o meno numeroso di dispositivi periferici chiamati, semplicemente, **periferiche**

Tra la **CPU** ed una qualsiasi periferica si deve necessariamente stabilire un ***sistema di comunicazione*** che consiste, in sostanza, in una richiesta di **I/O** da parte della periferica stessa.



Si pone allora il problema fondamentale:

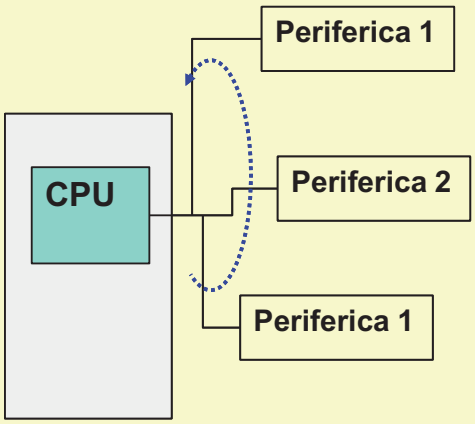
come far dialogare la **CPU**
con le **periferiche**
nel modo più efficiente possibile

per risolvere un tale problema esistono tre metodi principali denominati:

- **polling** (sondaggio)
- **DMA** (accesso diretto alla memoria)
- **Interrupts** (interruzioni).

Polling

Il metodo del polling consiste nel fatto che la **CPU**, ad intervalli di tempo regolari, "**interroga**" **a rotazione** ciascuna delle periferiche (polling) per sapere se c'è una eventuale richiesta di **I/O** per scrivere o leggere dati (interrogazione ciclica delle periferiche).



Questo metodo prende anche il nome **I/O a controllo di programma** perché la CPU

inizia, **coordina** e **termina**

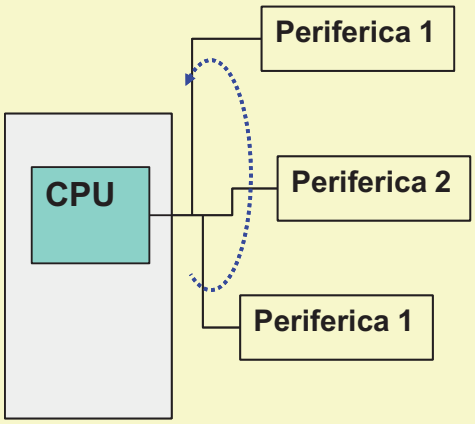
l'operazione di I/O

rimanendo in attesa del completamento.

Questa modalità operativa viene detta di "**attesa attiva**" perché presuppone il controllo periodico da parte della CPU.

Polling

Le operazioni sono le seguenti:



1. Il processore continua a leggere in polling **il bit busy** del dispositivo finché non lo trova sul valore **0** (**disponibile**)
2. Se ha bisogno di tale dispositivo, il processore specifica il comando nel **registro di controllo** e setta il bit **command-ready** a **1** in modo che la periferica se ne accorga
3. Il controller della periferica mette a **1** il bit **busy** (**occupato**) ed esegue il comando
4. Tutti i dati vengono scambiati tramite una porta di lettura/scrittura
5. Terminata l'esecuzione vengono resettati entrambi i bit **command-ready** e **busy**

Polling

Vantaggi :

- la **semplicità circuitale**, la **flessibilità** e il **basso costo**

ed è utilizzabile in sistemi piccoli, poco complessi in cui siano presenti **poche periferiche**, tutte molto veloci nel rispondere al sondaggio effettuato dalla **CPU**.

(Ad es. è adatto a testare periferiche come il **mouse** che può essere interrogato 30 volte/sec per non perdere i movimenti)

Svantaggi:

- **scarso utilizzo della CPU** che "spreca tempo" nell'esecuzione del ciclo di polling con un rallentamento generale del sistema
- è difficile tener conto del concetto di "urgenza" e le latenze risultano spesso notevoli

Una periferica che impiega molto tempo per rispondere al sondaggio, tiene inutilmente occupata la **CPU** e finisce anche per creare una lunga coda di richieste di **I/O** da parte di altre periferiche costrette ad attendere il loro turno.

DMA

DMA (Direct Memory Access, "*accesso diretto alla memoria*") è un meccanismo che permette ad alcuni sottosistemi hardware di un computer (periferiche) di **accedere direttamente alla memoria** di sistema per scambiarsi dati, oppure leggere o scrivere, **senza chiamare in causa la CPU.**

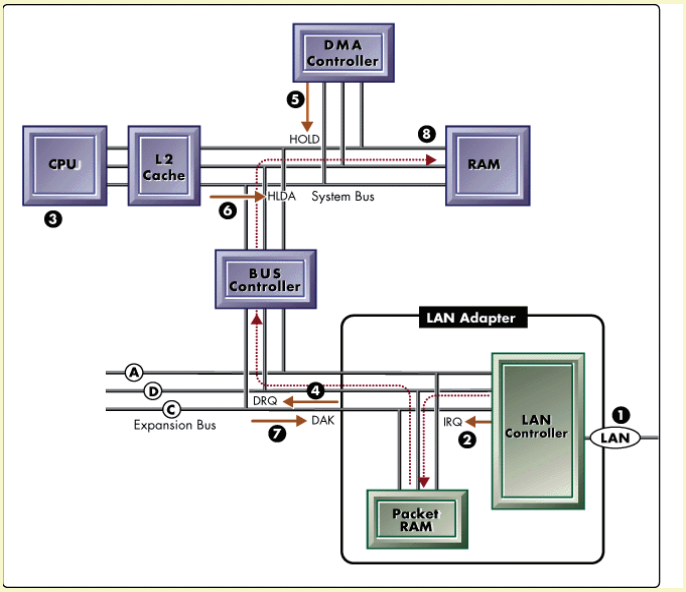
In un trasferimento DMA un blocco di memoria viene copiato da una periferica a un'altra.

La **CPU si limita a dare avvio al trasferimento** rilasciando il bus dati, mentre **il trasferimento vero e proprio è svolto dal *controller DMA***. *La CPU in questo modo* può continuare a svolgere altre operazioni

DMA

Il DMA, tramite il **controllore di accesso diretto** (**DMAC**), ha quindi il compito di gestire i dati passanti nel BUS permettendo a periferiche che lavorano a velocità diverse di comunicare senza costringere la CPU a un enorme carico di interrupt che ne interromperebbero continuamente il rispettivo ciclo di elaborazione.

Il DMA è usato da molti sistemi hardware come controller di unità a disco, schede grafiche e schede audio.



Interrupt

Il metodo delle interrupts comporta:

una **complessità circuitale nettamente superiore** al polling,
ma garantisce una enorme **efficienza generale del sistema**;

proprio per questo motivo, si tratta di un metodo largamente utilizzato sui **PC** e su molte altre piattaforme hardware.

Il metodo delle interruzioni prevede che tutte le richieste di **I/O** *vengano intercettate* da un **apposito dispositivo PIC** (Programmable Interrupt Controller).

Interrupt

Programmable Interrupt Controller (PIC)

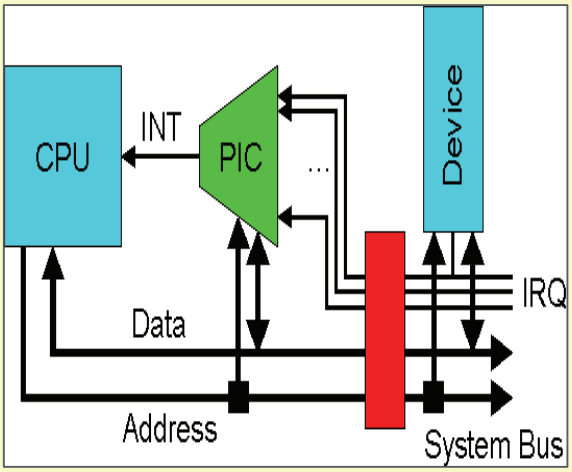
lo scopo di tale dispositivo è quello di:

creare **una coda di attesa**

dove le varie richieste di **I/O** vengono ordinate

in base alla **priorità** assegnata a ciascuna di esse.

Al momento opportuno, il dispositivo **PIC** invia alla **CPU** una richiesta di dialogo da parte della periferica alla quale è stata assegnata la priorità maggiore.



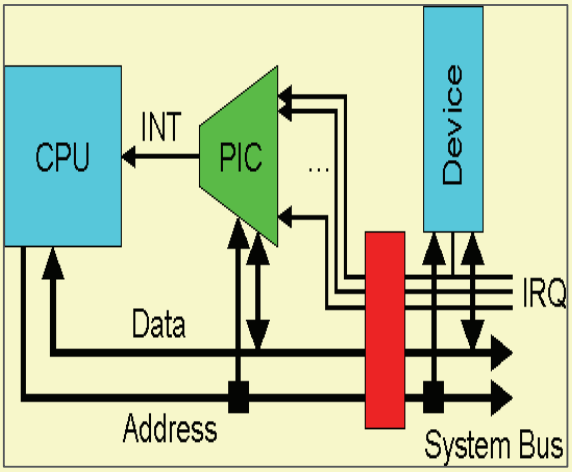
Interrupt

Programmable Interrupt Controller (PIC)

Solo in quel momento, la **CPU** interrompe il programma in esecuzione (da cui la denominazione di "**interrupt**") e soddisfa la richiesta della periferica.

In sostanza, grazie a questo metodo, la **CPU** viene "disturbata" solo quando è strettamente necessario;

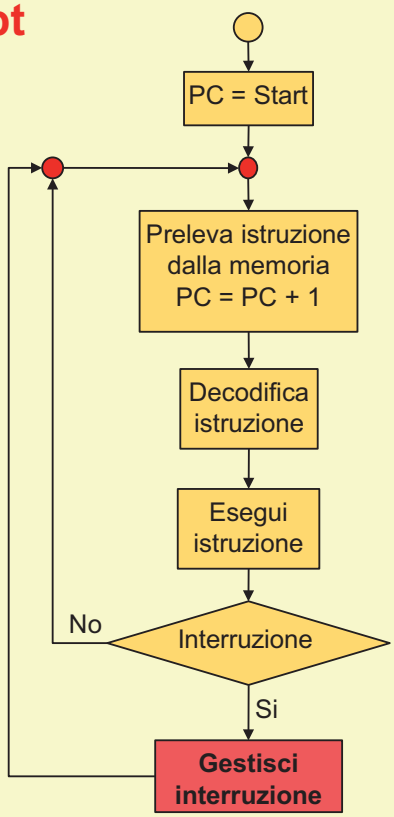
il programma in esecuzione viene quindi interrotto per il minor tempo possibile.



Interrupt

Esecuzione delle istruzioni e

controllo dell'arrivo di interruzioni



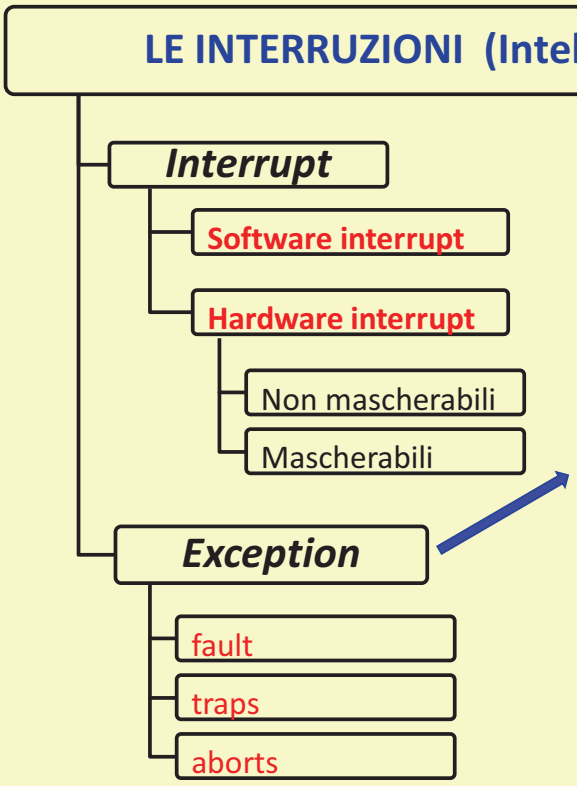
Interrupt

Un'**interruzione** è un **segnale** che comunica alla CPU il **verificarsi di un evento**

Per es.: - la richiesta di scambio dati con una periferica
- la fine di un time slice quando un processo esaurisce il tempo di utilizzo della CPU che gli è stato assegnato

Un'interruzione può essere provocata da:

- **dal software** con istruzioni del tipo **INT n** o **INTO**
- **dispositivi esterni (hardware)** che richiedono un servizio
- **dal processore stesso** in seguito a particolari condizioni interne (**eccezioni**) causate dal programma in esecuzione in quel momento



Le **eccezioni** avvengono per :

- **Tentativo di accedere ad aree di memoria protette** o a dati che non sono in memoria, **divisione per 0** o **overflow** (fault)
- Richiesta di **interruzione al termine di ogni istruzione** single-step (traps)
- Rilevamento di **errori hardware** o **inconsistenze gravi** che terminano prematuramente il programma in esecuzione (aborts)

Le eccezioni

INT	Eccezione
00h	Si è verificata una divisione per zero durante una operazione
01h	Esecuzione single-step di un programma (debug mode)
03h	Breakpoint incontrato in un programma
04h	Si è verificato un overflow durante una operazione
05h	Bound range exceeded (indice fuori limite in un vettore)
06h	Opcode non valido
07h	Dispositivo (o estensione della CPU) non disponibile
0Dh	General protection fault (protected mode)
0Eh	Page fault (protected mode)

Interruzioni hardware

Le **interruzioni provocate dall'hardware esterno**

possono essere di due tipi:

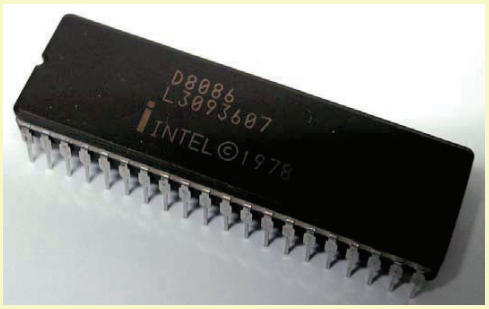
mascherabili

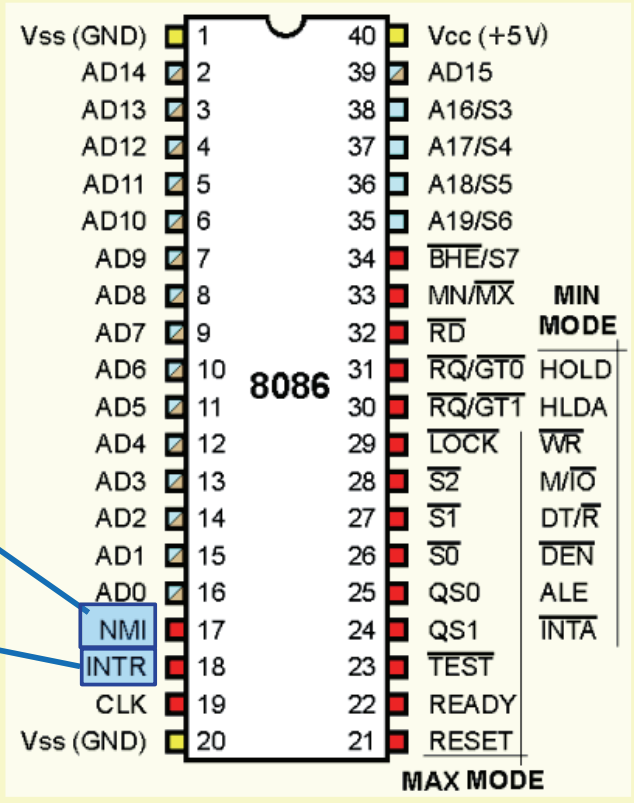
possono essere ignorate dalla CPU
agiscono sul pin **INTR** del processore

non mascherabili

non possono essere ignorate dalla CPU
Agiscono sul pin **NMI** del processore

Pin del microprocessore Intel 8086



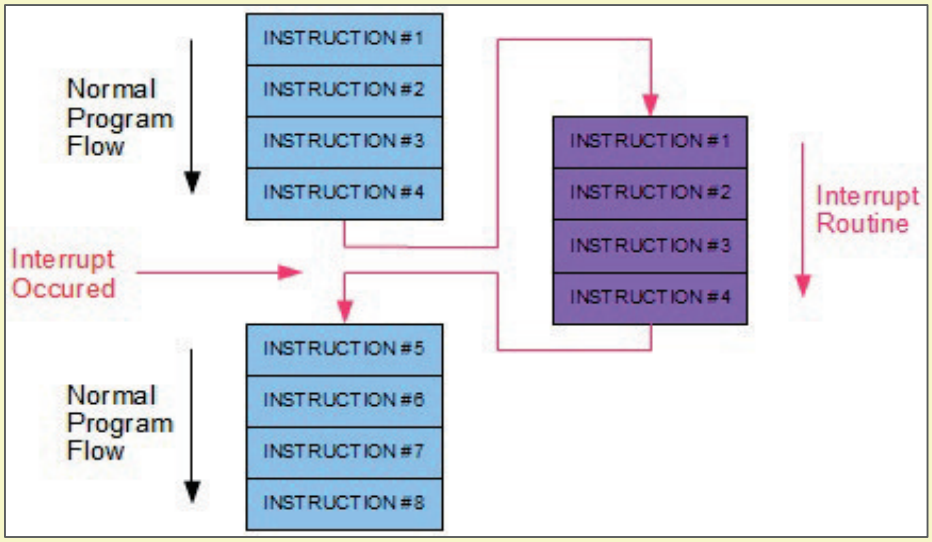


Non - maskable interrupt

Interrupt request

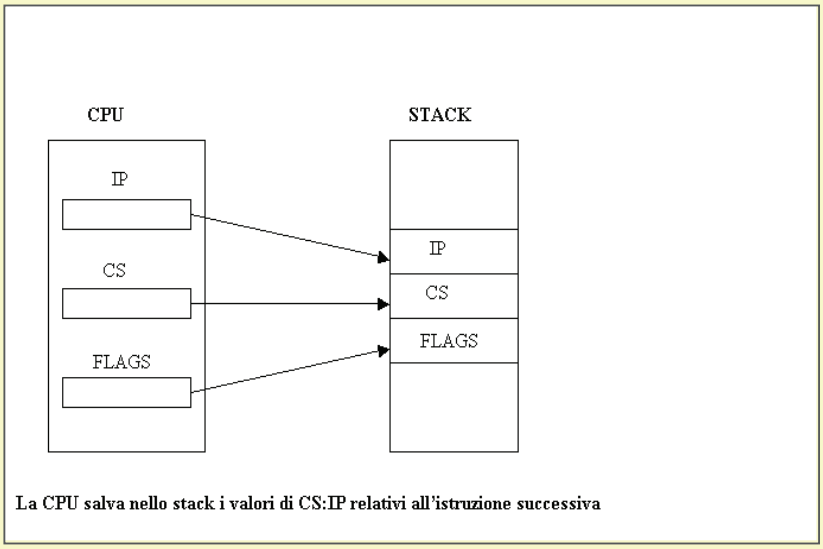
Al verificarsi di un'interruzione la CPU:

- **interrompe il programma** in esecuzione
- **esegue un sottoprogramma**, chiamato **routine di interrupt,** allocato ad uno specifico indirizzo di memoria.
- terminata la routine d'interrupt, **la CPU torna al programma precedentemente interrotto** e ne prosegue l'esecuzione.



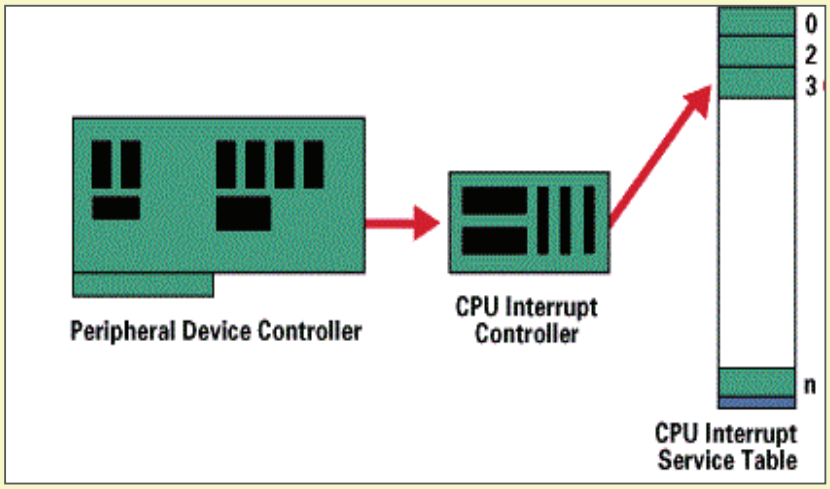
Dettaglio del processo di gestione dell'interruzione

1) Ricevuto il segnale di interrupt la CPU **completa l'esecuzione dell'istruzione corrente**
2) **salva nello stack** (PUSH) il registro dei *flags* e l'indirizzo logico dell'istruzione successiva del programma corrente (*IP* e *CS*)



Dettaglio del processo di gestione dell'interruzione

3) **legge un codice di 8 bit** che è compito della periferica inviare sul bus dati (gli 8 bit meno significativi).
Questo codice si chiama **vettore d'interruzione** e viene utilizzato dalla CPU come un *indice per accedere ad una tabella in memoria*, la **tabella dei vettori delle interruzioni** (Interrupt Vector Table), contenente gli **indirizzi logici** delle routine d'interrupt.

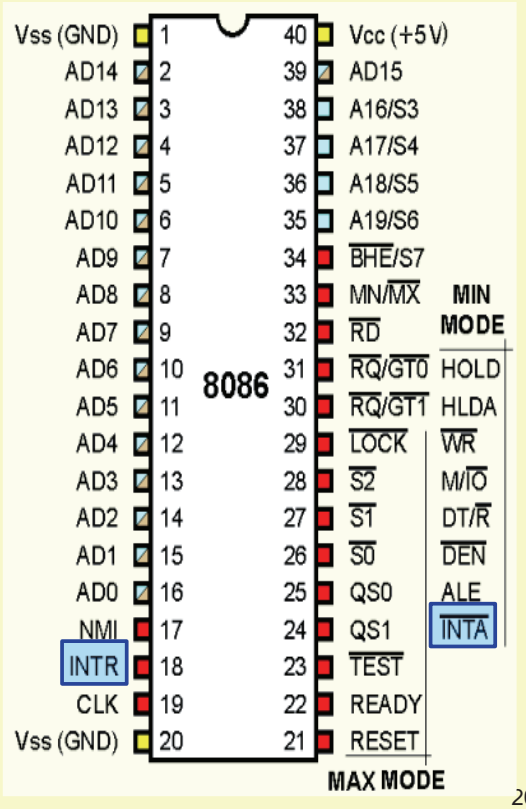


Dettaglio del processo di gestione dell'interruzione

4) **Se l'interruzione è mascherabile** (arriva sul pin INTR), la CPU, prima di iniziare le operazioni di gestione dell'interruzione descritte, invia alla periferica il segnale **INTA** con cui comunica che **l'interruzione è stata accettata**.

A questo punto la periferica deve inviare il vettore d'interruzione.

In assenza di tale segnale la richiesta viene considerata non accolta.



Dettaglio del processo di gestione dell'interruzione

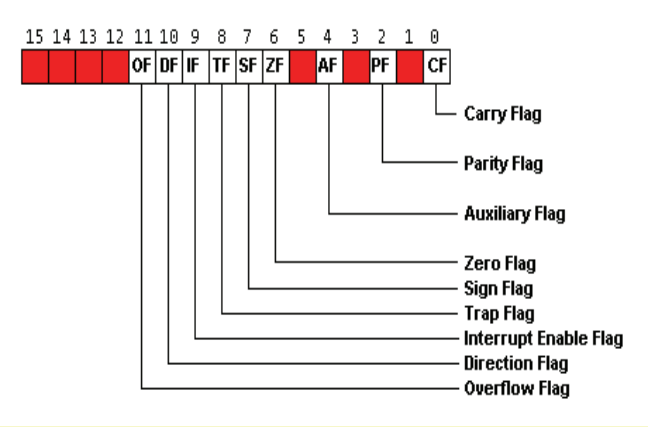
L'abilitazione o meno delle interruzioni dipende dal flag **IF** del **registro di flag** del processore.

Se **IF=1** le interruzioni sono **abilitate**
se **IF=0** sono **disabilitate**.

L'Assembly fornisce due istruzioni a questo scopo:
STI che pone **IF**=1 e **CLI** che pone **IF**=0.

Occorre comunque tenere presente che la CPU resetta IF automaticamente dopo l'accettazione di un interruzione.

Il compito di riabilitare le interruzioni quindi spetta al programmatore che dovrà inserire in un punto opportuno della sua routine d'interrupt l'istruzione **STI**.



Dettaglio del processo di gestione dell'interruzione

La gestione delle interruzioni mascherabili comporta due problemi:

- si devono **collegare più periferiche** al singolo ingresso **INTR** della CPU
- la CPU deve riconoscere la periferica che ha richiesto l'interruzione

Entrambi i problemi vengono risolti interponendo fra le periferiche e la CPU un componente programmabile che controlla le interruzioni, il **PIC** 8259 (Programmable Interrupt Controller)

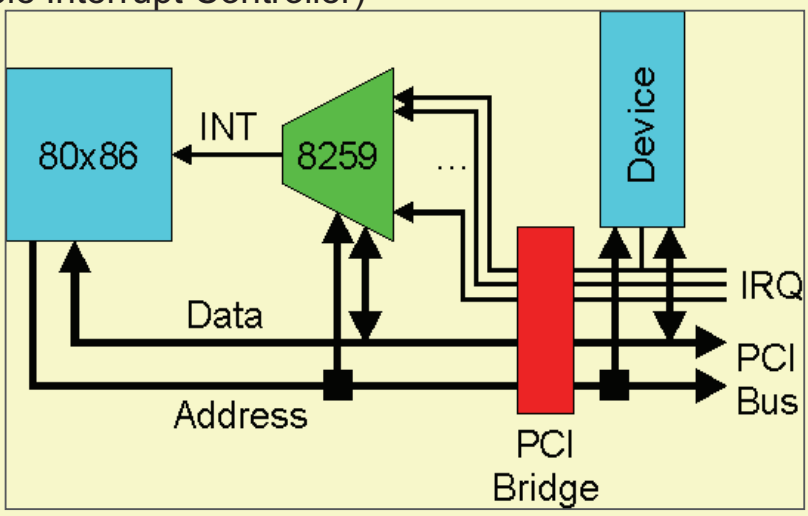


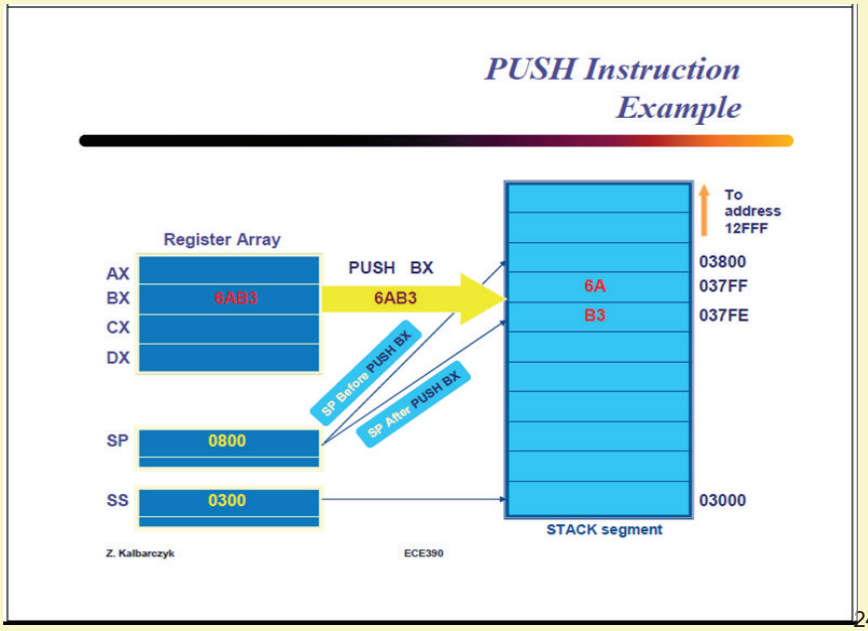
Tabella dei vettori di interruzione

(parte iniziale)

Addr Tab	N°	Vettore	Proprietà	Funzione Assicurata
0000:0000	00H	00A7:1068	utilizzabile	Errore di divisione per zero
0000:0004	01H	0070:018B	IO.Sys	Single-Step, usato nel debugging
0000:0008	02H	03B1:0016	MsDOS.Sys	NMI, Interruzione NON mascherabile
0000:000C	03H	0070:018B	IO.Sys	Punto di Break (BreakPoint), usato nel debugging
0000:0010	04H	0070:018B	IO.Sys	Errore di Overflow aritmetico
0000:0014	05H	0212:06B9	Driver dispositivo	HardCopy, servizio per il tasto Print Screen/Stampa
0000:0018	06H	0212:0740	Driver dispositivo	Codice Operativo (OpCode) non valido / non usato
0000:001C	07H	0212:03FF	Driver dispositivo	Processore matematico non disponibile
0000:0020	08H	0212:0746	Driver dispositivo	IRQ0, Interruz. mascherabile dal Timer di Sistema
0000:0024	09H	0E92:06EC	MsDOS.Sys	IRQ1, Interruz. mascherabile dalla Tastiera
0000:0028	0AH	03B1:003A	MsDOS.Sys	IRQ2, Interruz. mascherabile dal secondo gestore
0000:002C	0BH	03B1:0054	MsDOS.Sys	IRQ3, Interruz. mascherabile dalla Seriale COM2
0000:0030	0CH	03B1:006E	MsDOS.Sys	IRQ4, Interruz. mascherabile dalla Seriale COM1
0000:0034	0DH	03B1:0088	MsDOS.Sys	IRQ5, Interruz. mascherabile dalla Stampante LPT2
0000:0038	0EH	03B1:00A2	MsDOS.Sys	IRQ6, Interruz. mascherabile dal Floppy Disk
0000:003C	0FH	0212:03FF	Driver dispositivo	IRQ7, Interruz. mascherabile dalla Stampante LPT1
Addr Tab	N°	Vettore	Proprietà	Funzione Assicurata
0000:0040	10H	0212:08A9	Driver dispositivo	Funzioni per la Gestione del Video
0000:0044	11H	0212:09A4	Driver dispositivo	Determinazione della dotazione del computer
0000:0048	12H	0212:09AA	Driver dispositivo	Determinazione della dimensione della Memoria
0000:004C	13H	0212:045D	Driver dispositivo	Funzioni per la Gestione dei Dischi
0000:0050	14H	0212:09B0	Driver dispositivo	Funzioni per la Gestione delle Porte Seriali
0000:0054	15H	02ED:020D	Gestore XMS	Funzioni per la Gestione Estesa del Sistema
0000:0058	16H	0212:09C4	Driver dispositivo	Funzioni per la Gestione della Tastiera
0000:005C	17H	0212:058B	Driver dispositivo	Funzioni per la Gestione della Stampante
0000:0060	18H	0212:0C0E	Driver dispositivo	Caricatore del Basic IBM residente (obsoleta)
0000:0064	19H	0212:0C14	Driver dispositivo	Esecuzione del Boot-strap da disco
0000:0068	1AH	0212:0C1F	Driver dispositivo	Funzioni per Gestione dell'Orologio in Tempo Reale
0000:006C	1BH	0212:06AD	Driver dispositivo	Procedura Utente per la Gestione della Tastiera
0000:0070	1CH	0212:06AD	Driver dispositivo	Procedura Utente per la Gestione Timer di Sistema
0000:0074	1DH	F000:F0A4	ROM Bios	Tabella di Inizializzazione dei Parametri Video
0000:0078	1EH	0212:0537	Driver dispositivo	Tabella Base dei Parametri dei Dischetti
0000:007C	1FH	C000:3590	ROM Video	Tabella dei Caratteri Grafici Video (Set2)
Addr Tab	N°	Vettore	Proprietà	Funzione Assicurata
0000:0080	20H	00A7:1072	MsDOS.Sys	Terminatore di un programma
0000:0084	21H	00A7:107C	MsDOS.Sys	Funzioni generali del DOS
0000:0088	22H	1010:01DE	Command.com	Procedura di Terminazione

Lo stack

- Lo stack è usato come memoria temporanea per memorizzare dati e indirizzi.
- Quando viene eseguita una chiamata ad un sottoprogramma (**CALL**), l'8086 automaticamente salva (**PUSH**) il valore corrente di CS e IP nello stack.
- Anche altri registri possono essere salvati nello stack con operazioni PUSH



Lo stack

- Prima di tornare al programma chiamante il sottoprogrammma esegue un'istruzione di **POP** per prelevare i valori dallo stack e riportarli nei registri.

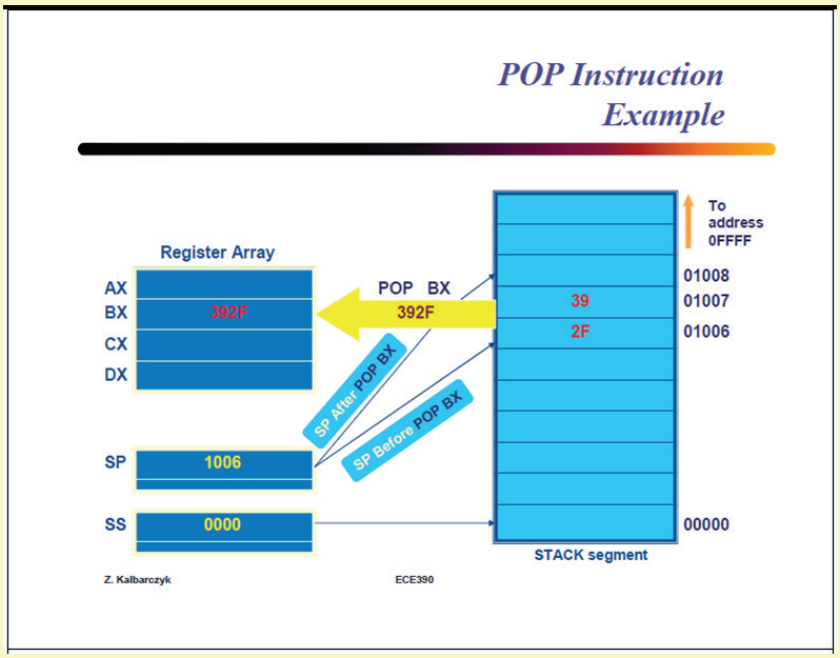


Tabella dei vettori delle interruzioni

La **tabella dei vettori delle interruzioni** è allocata nella parte bassa della RAM e ha una dimensione fissa di **1KB**.
Essa contiene **256** indirizzi logici (ogni indirizzo logico è 4 bytes) di altrettante routine di gestione di tutti i tipi di interruzione (interne, software, hardware).

Ogni vettore d'interruzione punta ad un sottoprogramma allocato in determinato segmento di memoria.