ARCHITETTURA DEI MICROPROCESSORI INTEL 8086/8088

- microprocessori Intel della terza generazione
- progetto del 1978/79
- address bus: 20 bit ➜ 1M byte
- data bus: 8 bit per l'8088, 16 bit per l'8086
- identico formato delle istruzioni

Caratteristiche	8086/8088	80286	80386
address bus	20 bit	24 bit	32 bit
data bus	16 / 8 bit	16 bit	32 bit
indirizzamento	1M byte	16M byte	4G byte
		1G byte virtual	64T byte virtual
registri	16 bit	16 bit	32 bit
piedini	40	68	132
computer IBM	25,30 / PC,XT	AT,50,60	80

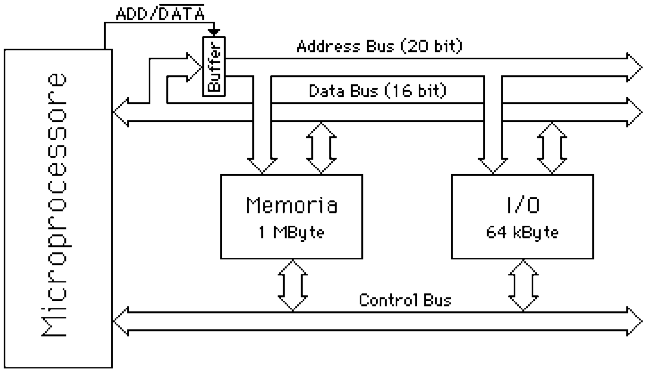
Compatibilità del software (assembler):

Intel

8086/8088 ➜ 80286 ➜ 80386 ➜ 80486 ➜ Pentium…

Motorola

68000 ➜ 68020 ➜ 68030 ➜ 68040 ➜ . . .



MULTIPLEXING

Costo µP proporzionale al numero dei piedini (*pin*)

Unico bus a 20 bit **suddiviso nel tempo** (*time multiplexed*) tra funzioni diverse

Unico bus = bus degli indirizzi o bus dei dati
+ segnale di controllo per selezionare
+ registri esterni (buffer) in grado di memorizzare indirizzi e/o dati

Accesso alla memoria e ai dispositivi di I/O meno veloce:
⇒ ogni trasferimento richiede 2 cicli di bus

MEMORIA PRINCIPALE

- 1M byte di memoria
  2^{20} = 1.048.576 locazioni di memoria di 8 bit
- il primo byte ha indirizzo 0
- l'ultimo byte ha indirizzo 0FFFFFh

Accesso contemporaneo al massimo a 4 segmenti di memoria di 64k byte ciascuno (max 256k byte)

INPUT e OUTPUT

L'8086 gestisce i dispositivi di I/O mediante:
- indirizzi di memoria
- indirizzi di I/O, distinti dagli indirizzi di memoria, in uno spazio di indirizzamento di 64k byte (0 ÷ 0FFFFh)

Gli indirizzi di I/O possono essere utilizzati esclusivamente nelle istruzioni di I/O:

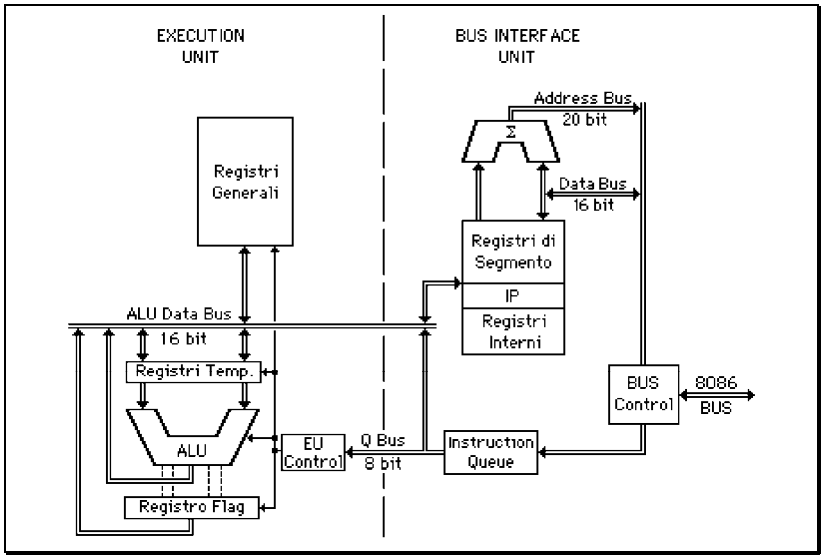
IN destinazione, 2

OUT 3, sorgente

La CPU

La CPU è costituita da due blocchi funzionali:

- l'Execution Unit (EU):
  - esegue le istruzioni (fase di execute)
- il Bus Interface Unit (BIU):
  - rintraccia le istruzioni (fase di fetch)
  - legge gli operandi
  - scrive i risultati



EU e BIU

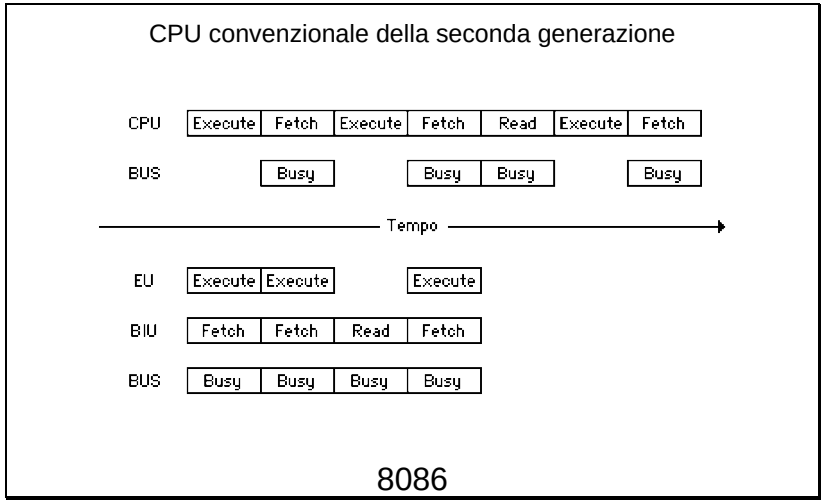
Le due unità possono operare
in modo indipendente l'una dall'altra.

⇓

La CPU è in grado di sovrapporre
le fasi di fetch e execute.

⇓

Tutte le volte che l'EU deve eseguire un'istruzione
che il BIU ha già caricato nella coda delle istruzioni,
il tempo richiesto per il fetch delle istruzioni è **nullo**



Execution Unit

- **esegue le istruzioni**
- **fornisce dati e indirizzi al BIU**
- **modifica registri generali e registro flag**

ALU, registri e bus interno a 16 bit (8086/8088)

L'EU non ha connessioni dirette con il bus di sistema (cioè, con il mondo esterno)

Quando l'EU deve eseguire una nuova istruzione, la ottiene dalla coda gestita dal BIU e se la coda è vuota si pone in attesa
Quando un'istruzione richiede di accedere alla memoria o a un device periferico, l'EU richiede al BIU di ottenere o memorizzare il dato

Gli indirizzi manipolati dall'EU sono di 16 bit
Il BIU effettua le operazioni che permettono di accedere all'intero spazio di memoria disponibile

Bus Interface Unit

Il BIU esegue tutte le richieste dell'EU che coinvolgono il mondo esterno (e quindi il bus di sistema), cioè i trasferimenti di dati tra la CPU e la memoria o i dispositivi di I/O

- **calcola gli indirizzi reali a 20 bit sommando, in un sommatore dedicato, l'indirizzo del segmento e l'offset (entrambi a 16 bit)**
- **esegue il trasferimento dei dati da e verso l'EU**
- **carica le istruzioni nella coda delle istruzioni (prefetch)**

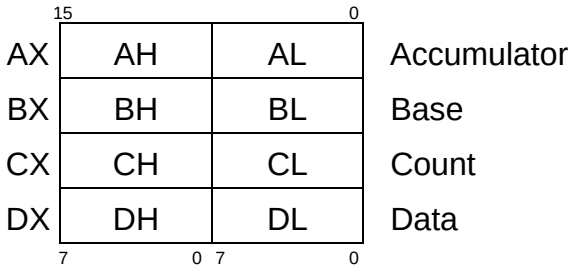
Le istruzioni caricate dal BIU nella coda sono quelle che *seguono* l'istruzione correntemente in esecuzione nell'EU

Se l'EU esegue un'istruzione di salto, il BIU svuota la coda e comincia a riempirla di nuovo a partire dal nuovo indirizzo
in questo caso, l'EU deve aspettare la nuova istruzione da eseguire

Registri Generali

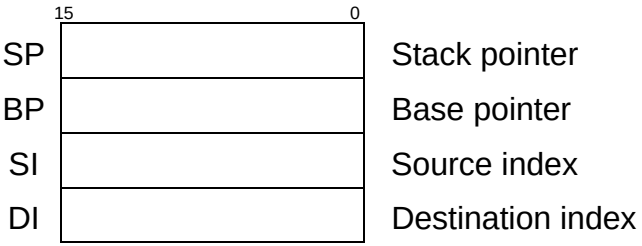
Data register (AX, BX, CX, DX)
Pointer register (SP, BP)
Index register (DI, SI)

Data Register



Utilizzabili come:
registri a 16 bit (AX)
registri a 8 bit (AH e AL - AHigh e ALow)

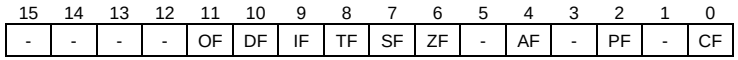
Pointer e Index Register



Registro Flag

Registro a 16 bit contenente:

- 6 flag di stato - modificati dall'EU in base al risultato di operazioni logiche e aritmetiche
- 3 flag di controllo – modificabili da programma al fine di controllare il comportamento della CPU



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	OF	DF	IF	TF	SF	ZF	-	AF	-	PF	-	CF

Flag di controllo:

1. DF (Direction Flag) - indica la **direzione** secondo la quale operare sulle stringhe (da destra a sinistra o viceversa)
2. IF (Interrupt-enable Flag) - **abilita** o **disabilita** gli **interrupt esterni** mascherabili (non ha effetto sugli altri tipi di interrupt)
3. TF (Trap Flag) - pone il processore nella modalità **single-step** per il debugger
   in questa modalità, la CPU genera automaticamente un interrupt interno dopo ogni istruzione, in modo che un programma possa essere controllato istruzione per istruzione

Flag di stato:

1. AF (Auxiliary Flag) - condizione di **riporto** durante un'operazione **BCD** (Binary Coded Decimal)
2. CF (Carry Flag) - condizione di **riporto** durante un'istruzione aritmetica
3. OF (Overflow Flag) - **overflow** aritmetico
4. SF (Sign Flag) - **segno** del risultato
5. PF (Parity Flag) - se il **numero di bit** a 1 del risultato è pari, vale 1, altrimenti, vale 0
6. ZF (Zero Flag) - indica che il **risultato** è **0**

ADD (somma intera) modifica tutti i flag di stato

AND (and logico bit a bit) modifica SF, PF e ZF in base al risultato, pone OF e CF a zero e rende indefinito AF

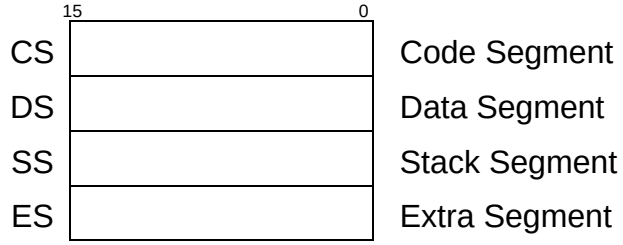
Esistono istruzioni che permettono al programma di controllare il contenuto di tali flag a fini decisionali

Registri di Segmento

Lo spazio di memoria indirizzabile dalla CPU è diviso in segmenti logici (massimo 64k byte)

La CPU può accedere direttamente a 4 segmenti per volta (massimo 256k byte)

Quattro registri di segmento puntano ai quattro segmenti correntemente *attivi*:



CS (Code Segment) punta al segmento codice corrente: il segmento da cui vengono ottenute le istruzioni da eseguire

SS (Stack Segment) punta al segmento contenente lo stack corrente

DS (Data Segment) punta al segmento dati corrente: in genere tale segmento contiene variabili di programma

ES (Extra Segment) punta al segmento extra corrente: in genere anche questo segmento viene utilizzato per memorizzare dati

Segmentazione

Lo spazio di memoria viene visto come un gruppo di segmenti

Ogni segmento:

- è un'unità logica di memoria indipendente, indirizzabile separatamente dalle altre unità
- inizia a un indirizzo di memoria multiplo di 16
- è costituito da locazioni contigue di memoria
- è al massimo di 64k byte

I segmenti si dicono allineati ai 16 byte (o allineati al paragrafo)

Ogni segmento è identificabile univocamente dai primi 16 bit del suo indirizzo di partenza in memoria:

indirizzo fisico 220h

indirizzo segmento 22h

I registri segmento puntano ai quattro segmenti correntemente utilizzabili

Ogni programma in esecuzione può accedere direttamente a:

- 64k byte di codice - CS
- 64k byte di stack - SS
- 128k byte di dati - DS e ES

Per accedere al codice o ai dati contenuti in altri segmenti, è necessario modificare i registri segmento in modo opportuno

Generazione dell'indirizzo fisico

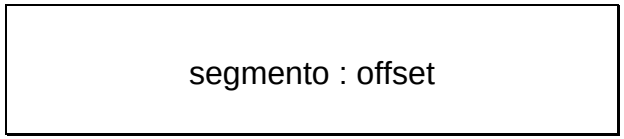
Un indirizzo fisico è un valore di 20 bit che identifica in modo univoco ogni byte dello spazio di memoria di 1M byte

Per trasferire dati tra la CPU e la memoria è necessario utilizzare gli indirizzi fisici

I programmi utilizzano indirizzi formati da:

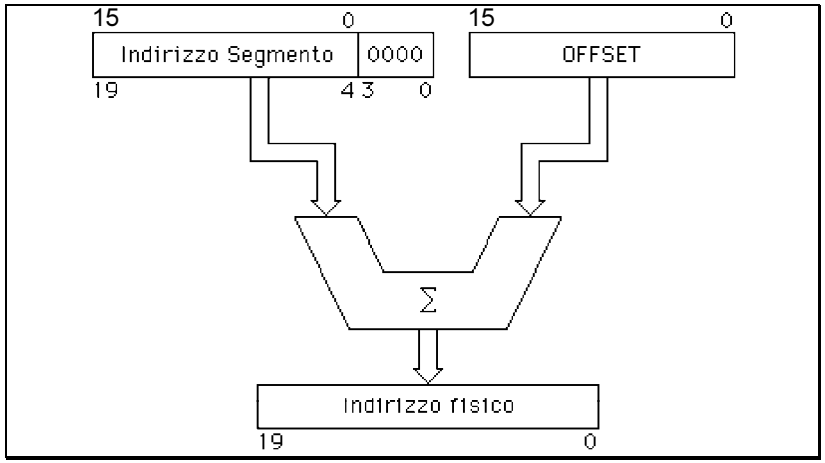
- indirizzo del segmento
- offset nel segmento

entrambi quantità di 16 bit senza segno



Il BIU converte la coppia segmento:offset in indirizzo fisico

Ciò avviene moltiplicando l'indirizzo del segmento per 16 e sommando al risultato l'offset nel segmento



Lo stesso indirizzo fisico può essere ottenuto con diverse coppie segmento:offset

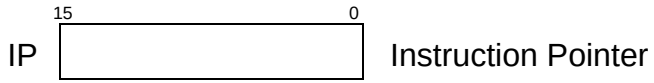
L'indirizzo fisico 1000h può essere ottenuto:
con 100h:0h
con 0F0h:100h
con 0E0h:200h
etc...

Il BIU ottiene la coppia segmento:offset da traslare, in modi diversi

Instruction Pointer

- registro a 16 bit IP
- viene gestito dal BIU
- contiene, in ogni istante, l'offset (cioè la distanza in byte) dell'istruzione successiva dall'inizio del segmento codice corrente (CS)

I programmi non hanno accesso diretto all'IP, ma le istruzioni lo modificano implicitamente



Le istruzioni da eseguire sono nel segmento codice corrente, pertanto l'indirizzo fisico della successiva istruzione è dato da: CS:IP, cioè CS*16+IP

Il program counter classico coincide con CS:IP

Stack

- area di memoria gestita LIFO
- realizzato in memoria centrale
- definito dai registri SS e SP

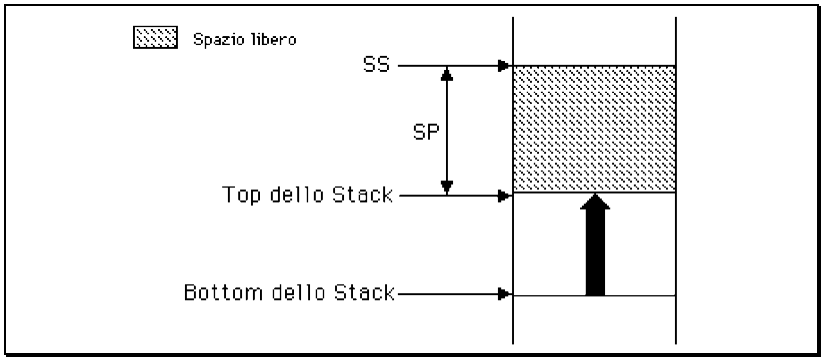
In memoria possono coesistere più stack, ognuno al massimo di 64k byte
se un programma oltrepassa per errore tale limite, ...

Le istruzioni **push** e **pop** agiscono sul segmento stack corrente:

- SS contiene l'indirizzo del segmento stack
- SP contiene l'offset del top dello stack

Lo stack cresce andando dagli indirizzi alti a quelli bassi:

l'indirizzo di partenza dello stack (contenuto in SS) non è il bottom dello stack



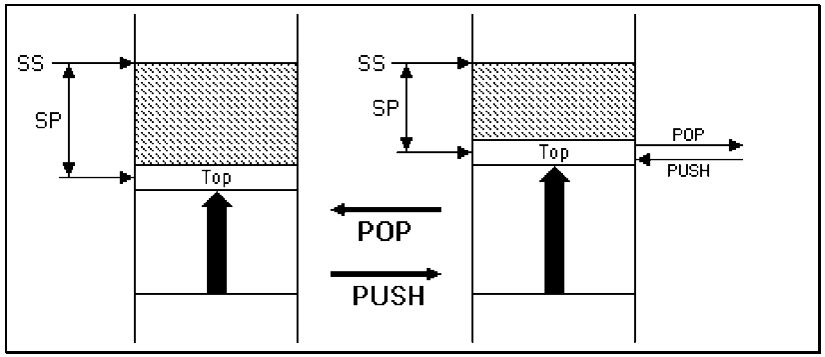
Le istruzioni che operano sullo stack trasferiscono due byte per volta (una word)

Operazione di **push**:

- SP $\leftarrow$ SP - 2
- scrittura di una word al nuovo top

Operazione di **pop**:

- lettura di una word dal top
- SP $\leftarrow$ SP + 2



Riferimento ai dati

Operandi che fanno riferimento alla memoria (variabili di programma) di norma sul data segment corrente (DS)

Il programma può dire al BIU di utilizzare uno qualunque dei quattro segmenti correntemente disponibili

Offset della variabile calcolato dall'EU - dipende dalla modalità di indirizzamento specificata nell'istruzione

Quando in un'istruzione viene utilizzato BP come registro base, se non è indicato diversamente, il BIU suppone che l'operando sia sul segmento stack

Riepilogo delle modalità con cui il BIU ottiene la coppia segmento:offset, in funzione del tipo di riferimento alla memoria:

Tipo di riferimento alla memoria	Segmento di default	Altri segmenti utilizzabili	OFFSET
Fetch dell'istruzione	CS	Nessuno	IP
Operazione sullo stack	SS	Nessuno	SP
Variabile (tranne il caso seguente)	DS	CS, ES, SS	Indirizzo effettivo
BP come Registro Base	SS	CS, DS, ES	Indirizzo effettivo

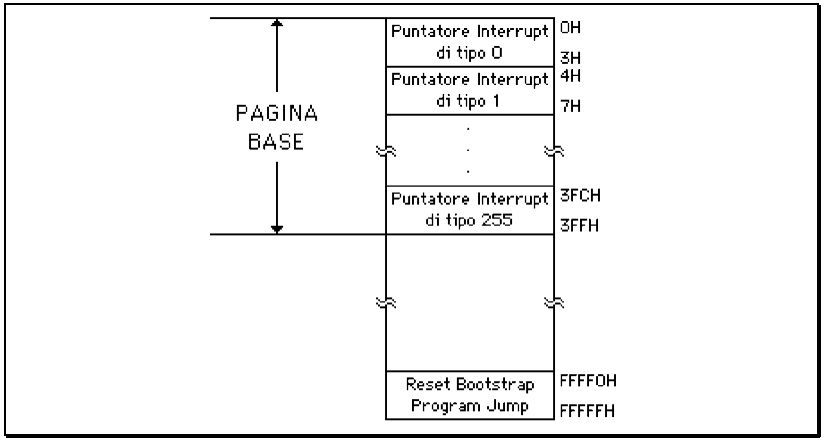
Locazioni di memoria riservate

Le locazioni da 0h a 3FFh (1024 byte) sono dedicate al servizio di 256 tipi di interrupt:
256 routine di servizio indirizzata da un puntatore di quattro byte (16 bit per l'indirizzo del segmento e 16 bit per l'offset)

Le locazioni da 0FFFF0h a 0FFFFFh (16 byte) sono dedicate alla gestione del reset:
contengono un salto alla routine da eseguire in caso di reset della CPU; dopo un reset:

- i registri CS e IP vengono inizializzati rispettivamente a 0FFFFh e a 0h
- la CPU esegue l'istruzione contenuta nell'indirizzo assoluto 0FFFF0h

Le applicazioni non devono utilizzare tali aree per motivi diversi



Interrupt

Gli interrupt possono essere:

- hardware
- software

L'effetto di un interrupt è quello di trasferire il controllo a una nuova locazione di memoria (routine di servizio dell'interrupt)

Gli **interrupt hardware**:

- hanno origine dalla logica esterna
- permettono di gestire eventi asincroni
- si dividono in:
  - mascherabili
  - non mascherabili

Gli **interrupt software**:

- hanno origine dall'esecuzione del programma:
  - **direttamente**
    (per es., l'esecuzione di un'istruzione INT)
  - **indirettamente** - condizioni eccezionali
    (per es., una divisione per zero)

Interrupt VETTORIZZATI

Le locazioni da 0h a 3FFh contengono
una tabella (Interrupt Vector Table) con 256 ingressi

Ogni ingresso contiene due valori di 16 bit che forniscono l'indirizzo della routine di servizio dell'interrupt e che vengono caricati nei registri CS e IP quando l'interrupt viene accettato

I primi cinque elementi della tabella sono dedicati a particolari tipi di interrupt predefiniti nell'8086

I successivi 27 elementi sono riservati e non devono essere utilizzati

I rimanenti elementi (da 32 a 255) sono disponibili per le routine di servizio dell'utente

Un programma può generare un interrupt di tipo n, mediante l'istruzione INT n

- Netta separazione di ambienti
- Trasferimento del controllo a routine di cui non si conosce la posizione in memoria

Interrupt 0 (Divide Error) - segnala un errore durante un operazione di divisione (ad es., divisione per zero)

Interrupt 1 (Single Step) - un'istruzione dopo il settaggio di TF (permette di eseguire una singola istruzione all'interno di un programma - utilizzato dal debugger)

Interrupt 2 (Non-Maskable Interrupt) - è l'interrupt hardware di priorità più alta e non è mascherabile - di norma, è riservato ad eventi importanti e urgenti (ad es., una caduta di tensione, un errore nella memoria, un errore sul bus di sistema)

Interrupt 3 (One Byte Interrupt) - utilizzato dal debugger per i breakpoint

Interrupt 4 (Interrupt on Overflow) - condizione di overflow (OF = 1) e viene eseguita l'istruzione INTO; permette di gestire l'eventuale condizione di overflow

Servizio di un interrupt

A livello hardware:

- viene eseguita una push dei registri flags, CS e IP per salvare la situazione corrente e poterla ripristinare al termine del servizio
- vengono caricati i nuovi valori di CS e IP dalla tabella degli interrupt
- vengono azzerati i flag TF (trap per single step) e IF (interrupt)

L'azzeramento di IF disabilita il riconoscimento di ulteriori interrupt hardware nella routine di servizio a meno che tale riconoscimento non venga riabilitato esplicitamente all'interno della routine di servizio stessa

La routine di servizio deve terminare con un'istruzione IRET (Interrupt RETurn), al fine di ripristinare correttamente la situazione presente al momento in cui si è verificata l'interruzione