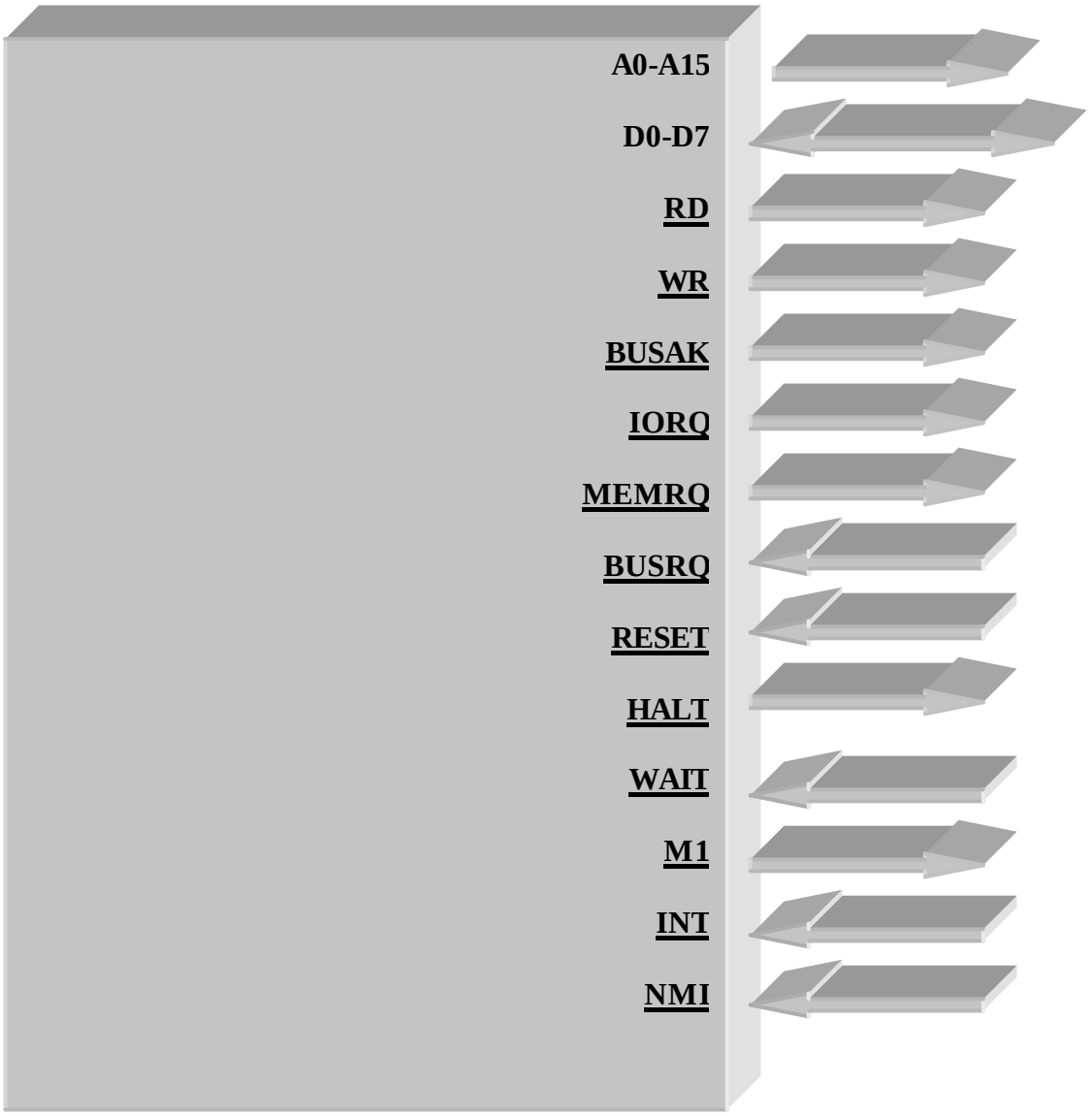
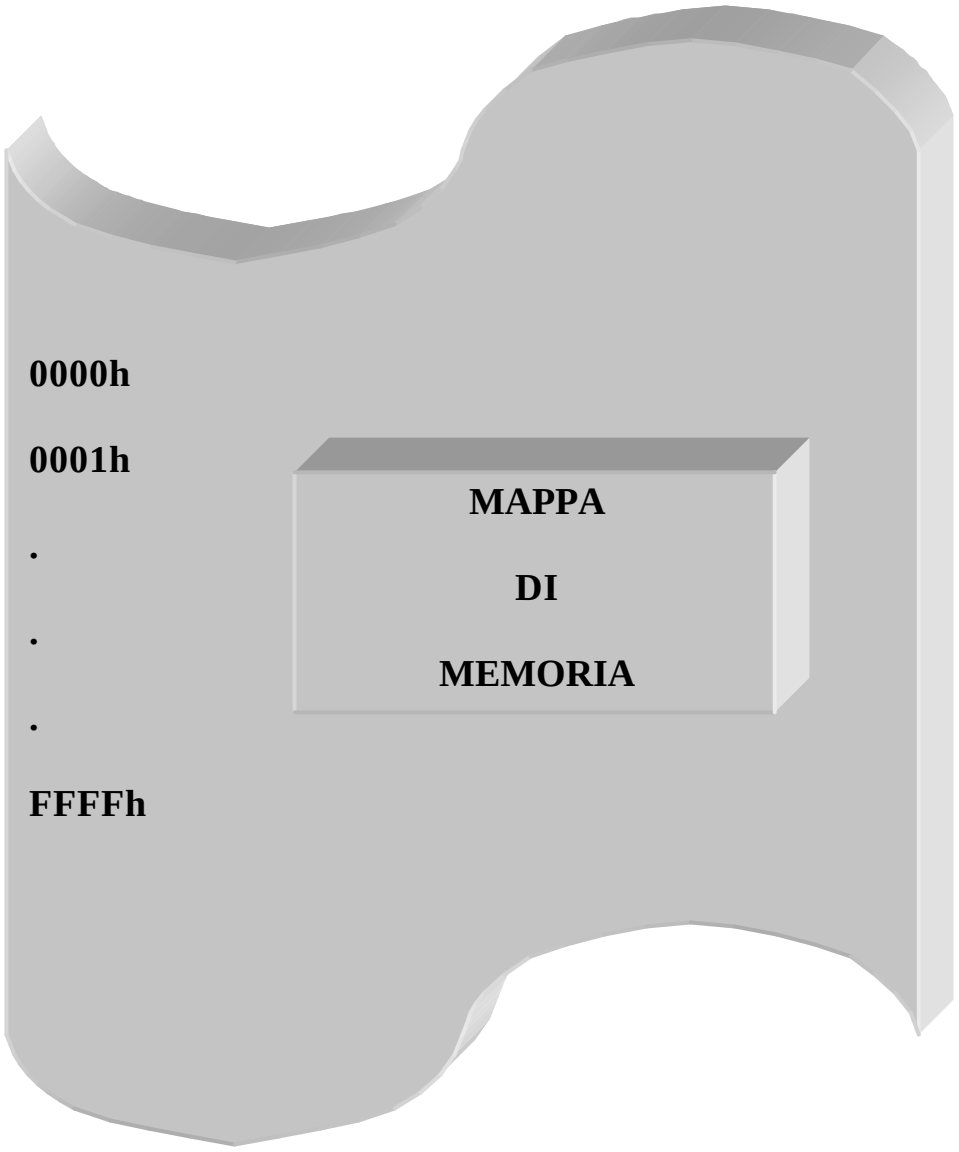
Piedinatura dello Z80



Bus indirizzi

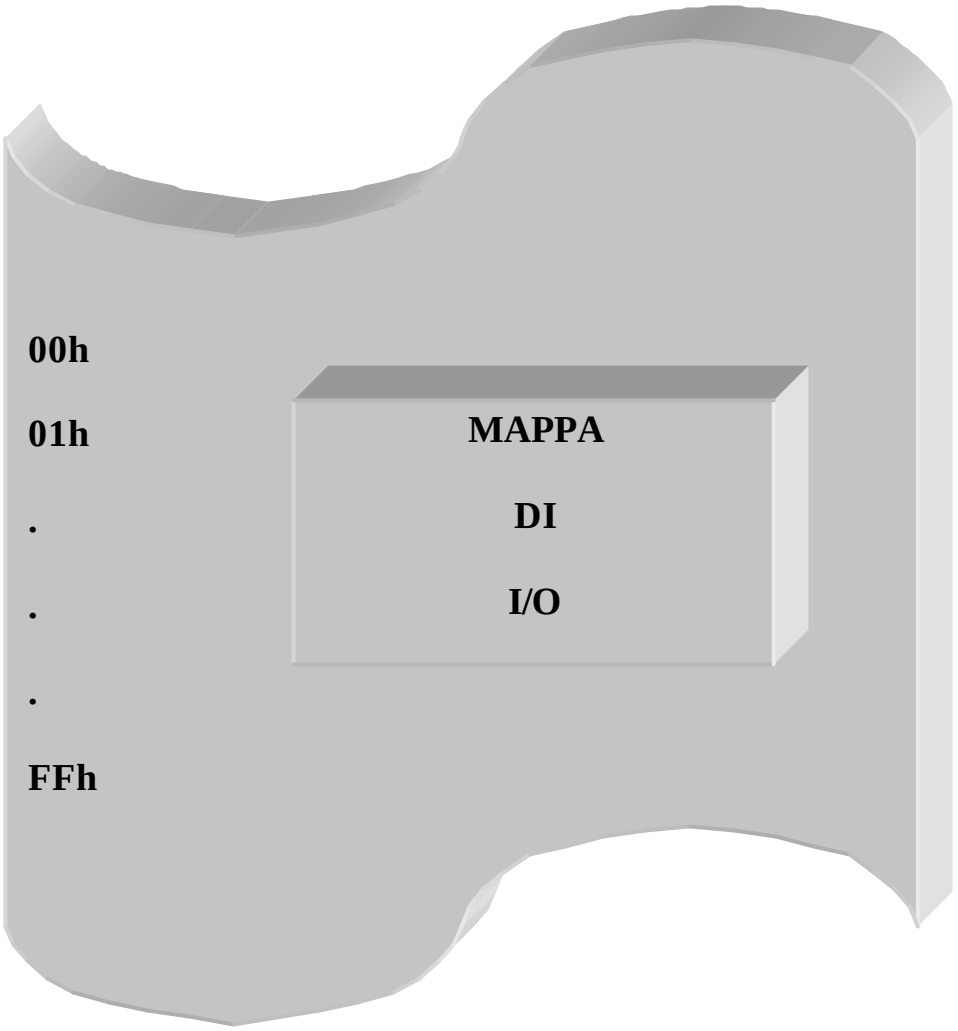
Abbiamo 16 piedini denominati da A0 ad A15 che costituiscono il bus indirizzi. Il bus indirizzi è unidirezionale e viene utilizzato dal microprocessore per immettervi l'indirizzo della locazione di memoria o dell'unità di ingresso/uscita con cui vuole dialogare. L'ampiezza del bus indirizzi costituisce la *capacità d'indirizzamento* del microprocessore. Per lo Z80 essa è pari a 2^{16}=65536 byte. Quindi allo Z80 si possono

collegare, al massimo, 65536 locazioni di memoria. Poiché lo Z80 non è in grado di generare ulteriori indirizzi, ad esso non si possono collegare altre locazioni di memoria poiché non potrebbero essere identificate dal microprocessore stesso. L'insieme degli indirizzi gestibili dallo Z80 (da 0000h a FFFFh) prende il nome di *mappa di memoria.*



La capacità di indirizzamento di memoria è diversa dalla capacità di indirizzamento di I/O. Infatti, nelle operazioni di I/O, il microprocessore non utilizza tutti i 16 bit del bus indirizzi per formare un indirizzo di una porta di I/O, ma soltanto i primi 8, da A0

ad A7. In tal caso lo Z80 può formare soltanto 2^8=256 indirizzi diversi per cui soltanto 256 unità di IO al massimo possono essere collegate al microprocessore: l'insieme degli indirizzi possibili di IO, da 00h a FFh prende il nome di *mappa di ingresso uscita*



Bus dati

IL bus dati è costituito dagli otto bit denominati D0 – D7, ed è bidirezionale. Infatti sul bus dati viaggiano i dati sia dal microprocessore alla memoria e alle unità di IO, sia il viceversa. L'ampiezza del bus dati è importante poiché ad essa è legata la mas-

sima quantità di informazioni che si possono trasmettere in una singola operazione di lettura o scrittura. Essa prende il nome di *parallelismo.*

Read e Write

Questi due segnali sono attivi bassi e sono utilizzati dallo Z80 per indicare ai circuiti esterni se esso vuole svolgere un'operazione di lettura o di scrittura, secondo la seguente tabella

RD	**WR**	
0	**0**	**Non consentito**
0	**1**	**Operazione di lettura**
1	**0**	**Operazione di scrittura**
1	**1**	**Nessuna operazione**

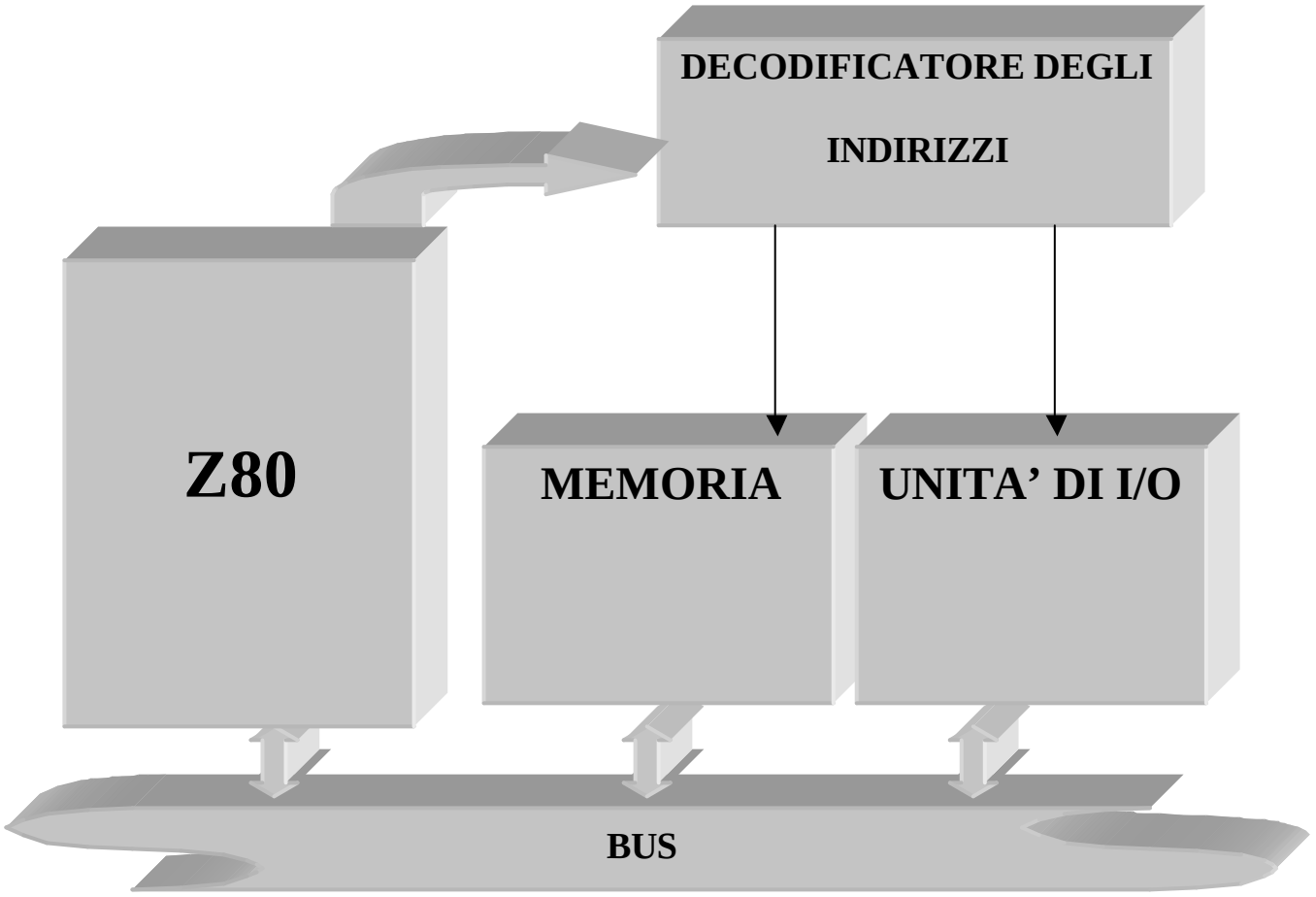
IORQ e MEMRQ

Con questi due segnali il microprocessore indica ai circuiti esterni se intende lavorare con la memoria o con unità d'ingresso uscita.

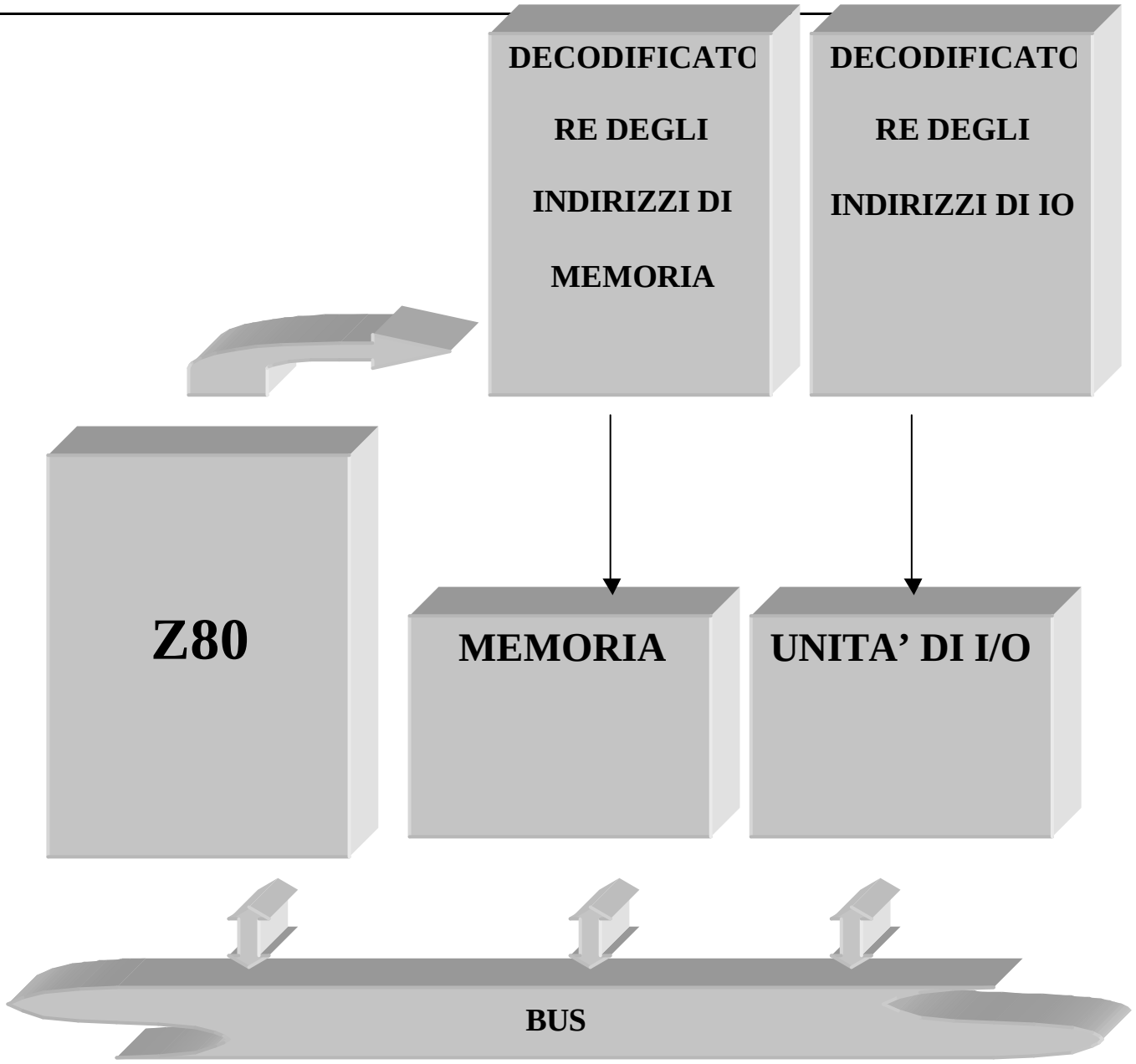
Un microprocessore può gestire memoria ed unità d'ingresso uscita in due modalità diverse:

- *Tecnica memory mapped – in questa tecnica la mappa di memoria è utilizzata per assegnare un indirizzo sia a locazioni di memoria che ad unità d'ingresso/uscita. In sostanza il microprocessore non fa differenza memorie e unità di*

unità di I/O. IN questo caso il circuito esterno è costituito da un decodificatore di indirizzi che comanda sia i chip di memoria che le unità di IO



- *Tecnica IO mapped- In questa tecnica vi sono due mappe di indirizzi separate per le locazioni di memoria e per le unità di IO. Questo vuol dire che una locazione di memoria ed una unità di IO possono avere lo stesso indirizzo e che vi sono due decodificatori di indirizzi separati. Ma a questo punto occorre che i decoder sappiano che lo Z80 sta eseguendo un'operazione in memoria oppure su unità di IO in modo che venga attivato il decoder giusto.*


DECODIFICATO
RE DEGLI
INDIRIZZI DI
MEMORIA
DECODIFICATO
RE DEGLI
INDIRIZZI DI IO
Z80
MEMORIA
UNITA' DI I/O
BUS

I segnali IORQ e MEMRQ avvertono i circuiti esterni di quale tipo di operazione si tratta secondo la seguente tabella

IORQ	MEMRQ	
0	**0**	**Non consentito**
0	**1**	**Operazione di IO**
1	**0**	**Operazione di memoria**
1	**1**	**Nessuna operazione**

Reset

Questo segnale è inviato dall'esterno al microprocessore ed ha lo scopo di resettare o azzerare tutti i suoi registri interni.

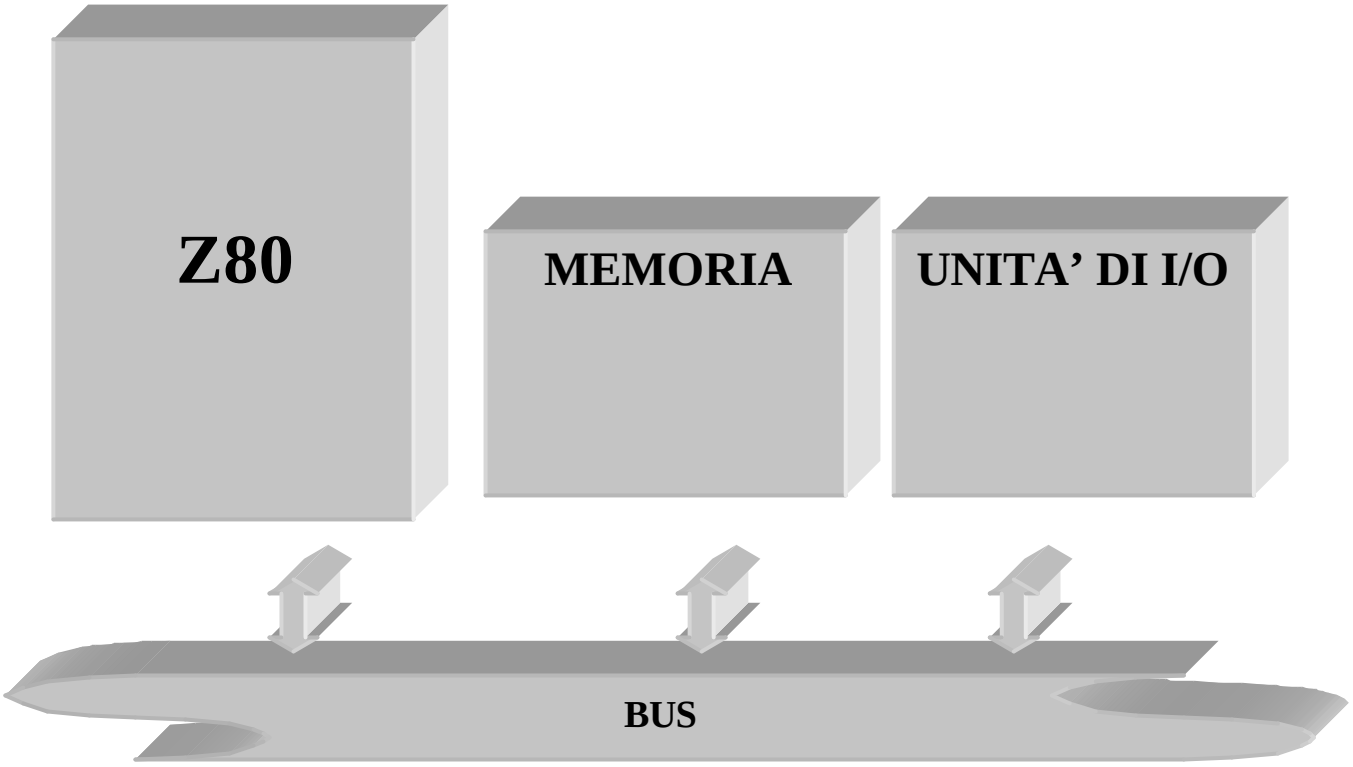
Il microprocessore è un automa sequenziale, il cui stato è rappresentato dal contenuto dei suoi registri interni. E' evidente che, all'accensione, poiché il microprocessore non ha una storia, il suo stato e quindi il contenuto dei suoi registri interni deve essere nullo. Ma quando si dà alimentazione alla scheda a microprocessore, per motivi del tutto casuali come ad esempio disturbi elettromagnetici, il contenuto dei registri potrebbe assumere un valore del tutto casuale. Immaginiamo, ad esempio, che, all'avvio, il Program Counter non abbia un contenuto nullo: ciò significa che lo Z80 inizierebbe ad eseguire un programma da un punto qualsiasi. Per questa ragione bisogna assicurarsi che, all'accensione, arrivi allo Z80 un impulso di reset, in modo da

azzerare tutti i registri del microprocessore. L'impulso di reset è efficace se la sua durata è di almeno tre cicli di clock.

Busrq e Busak

E' il segnale di Bus Request, utilizzato da un dispositivo esterno per chiedere al microprocessore di scollegarsi elettricamente dal bus, ponendo le sue uscite in three state.

Le schede a microprocessore sono schede *bus oriented.* Ciò significa che, in esse, esiste una sola via di comunicazione, il bus, per tutti i dispositivi presenti sulla scheda.



Ciò significa che, istante per istante, possono comunicare soltanto due dispositivi alla volta, utilizzando il bus unico. In caso contrario si avrebbe il cosiddetto *conflitto di*

bus. Ne deriva la necessità che uno dei dispositivi della scheda funzioni da *master,* cioè si trovi ad un livello gerarchicamente superiore agli altri dispositivi e permetta loro di collegarsi al bus soltanto su suo ordine. Per questo motivo, tutti i dispositivi che si vuole collegare ad una scheda a microprocessore devono avere uscite di tipo three state. Una uscita di tipo three state è un'uscita che può assumere il valore logico 0, il valore logico 1 ed il valore *alta impedenza* , stato nel quale il dispositivo risulta essere *elettricamente scollegato* dal bus. Questi dispositivi devono poi avere un ingresso di *chip enable* in modo da poter essere abilitati dal master a collegarsi al bus.

Nella generalità dei casi il master è costituito dal microprocessore stesso. In altri casi nella scheda possono essere presenti altri dispositivi come coprocessori matematici o controllori DMA (*Direct Memory Access* =Accesso Diretto in Memoria) che occupano un livello gerarchicamente superiore al microprocessore per cui possono richiedere a questo di scollegarsi dal bus inviando un segnale attivo basso sul piedino BUSRQ. Oltre a porre le sue uscite in three state, il microprocessore invia un segnale di risposta e accettazione ponendo a zero la sua uscita BUSAK (*Bus Aknowledge)*.

Halt

Lo Z80 può porsi in una particolare condizione, detta di Halt, derivante dall'esecuzione dell'istruzione HLT. In questa condizione, il microprocessore continua ad accedere in memoria eseguendo cicli di fetch, ma, in realtà, il bus dati non viene letto, quindi non vengono prelevati codici operativi, e in luogo di essi viene seguita l'istruzione NOP (*Not Operation)*. Il piedino di Halt segnala ai circuiti esterni che lo Z80 è in tale condizione.

M1

E' un segnale che interviene nel ciclo di fetch e nel ciclo di riconoscimento delle interruzioni che sarà definito più avanti.

INT

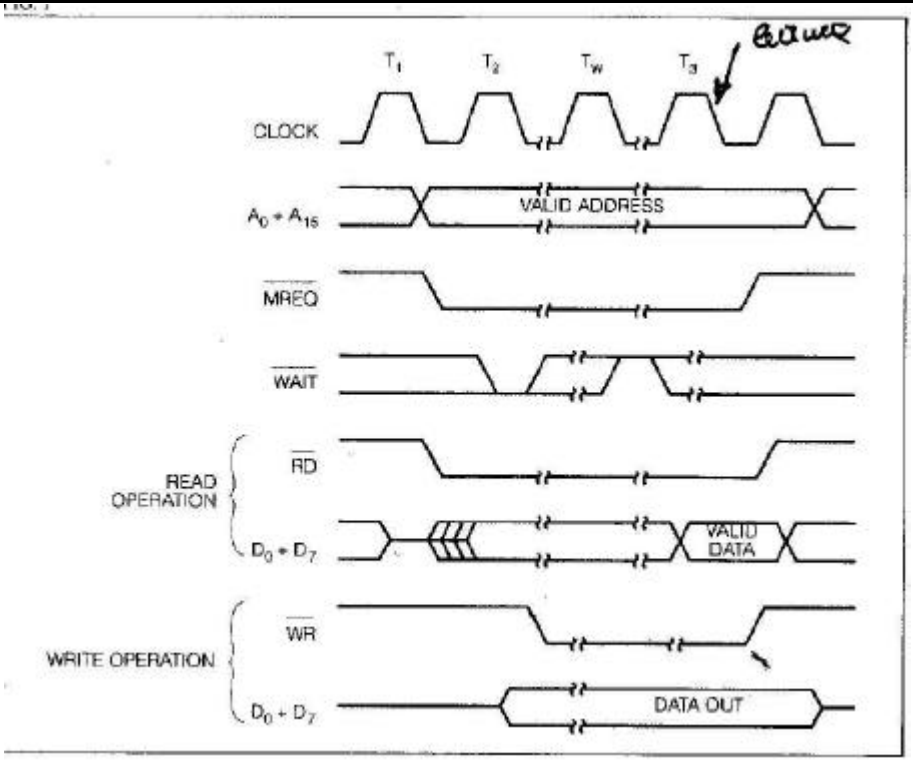
Ingresso delle interruzioni mascherabili (vedi più avanti).

NMI

Ingresso delle interruzioni non mascherabili (vedi più avanti).

Wait

Un ciclo di lettura in memoria o su unità di IO ha sempre una durata predeterminata come si può notare dalla figura a pagina seguente. In particolare, un'operazione di lettura in memoria dura sempre tre cicli di clock, iniziando al clock T_1 e terminando al clock T_3. Sul fronte di discesa di T_3 lo Z80 acquisisce il dato messo a disposizione dalla locazione di memoria selezionata. Ciò presuppone che la memoria sia in grado di scaricare sul bus dati l'informazione in tempo utile altrimenti l'operazione di lettura fallisce. Una memoria lenta può, però, inviare un segnale di Wait al microprocessore. Esso costringe lo Z80 a rallentare il ciclo di lettura. Infatti lo Z80 introduce dei cicli di wait finché vede il segnale di Wait basso.

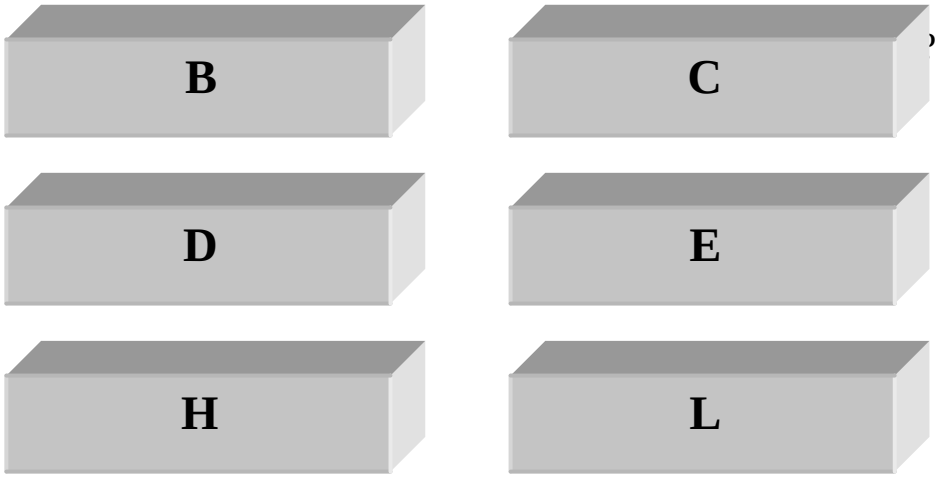


Registri dello Z80

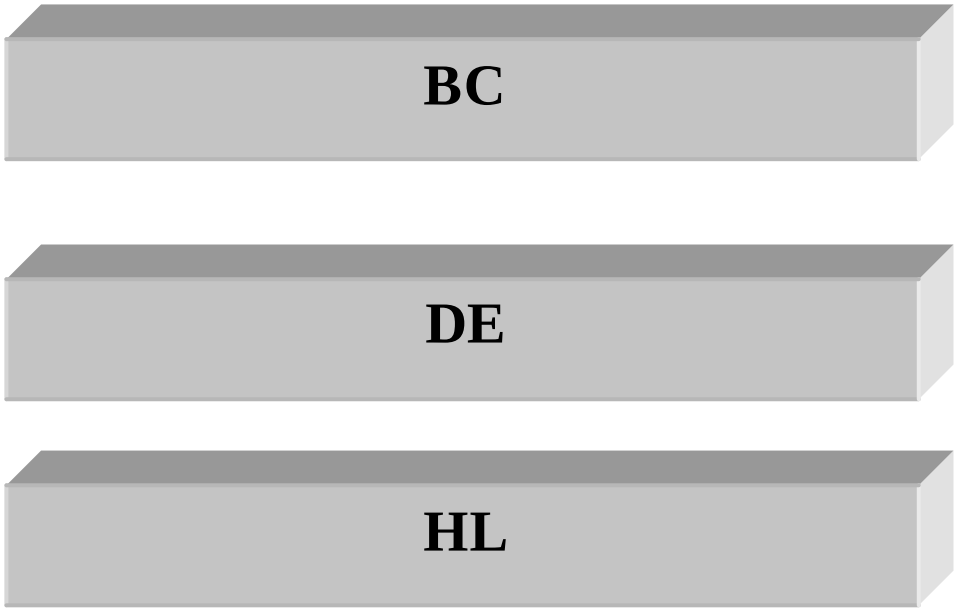
1. Registri di uso generale (General Purpose Registers)

Vengono così definiti i registri dello Z80 che possono essere utilizzati liberamente dal programmatore non essendo vincolati ad un compito specifico

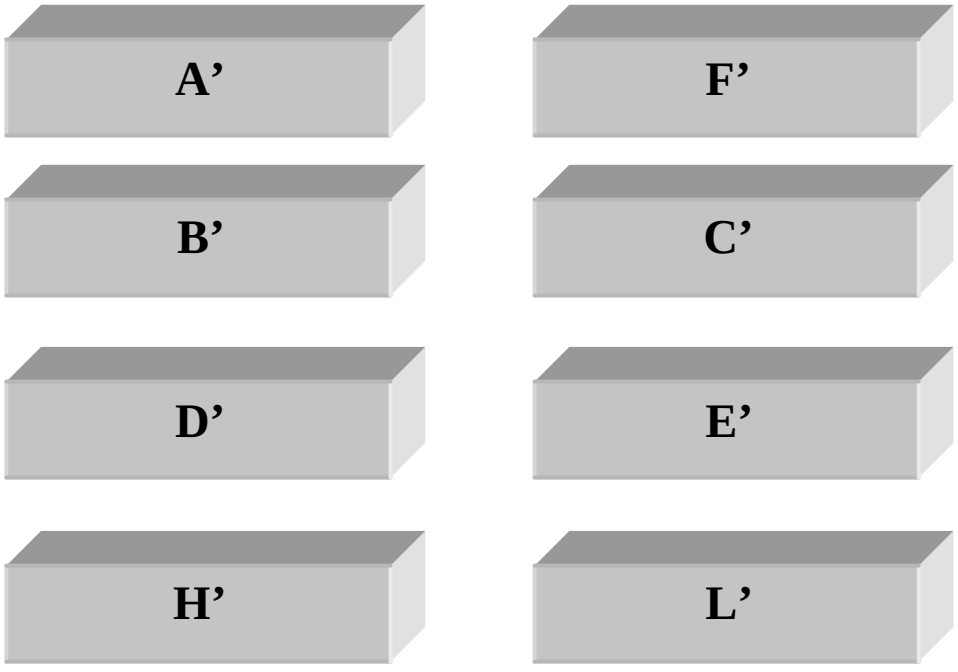
Abbiamo sei registri di uso generale così denominati



Questi registri, tuttavia, possono essere utilizzati in coppie costituendo 3 registri a 16 bit



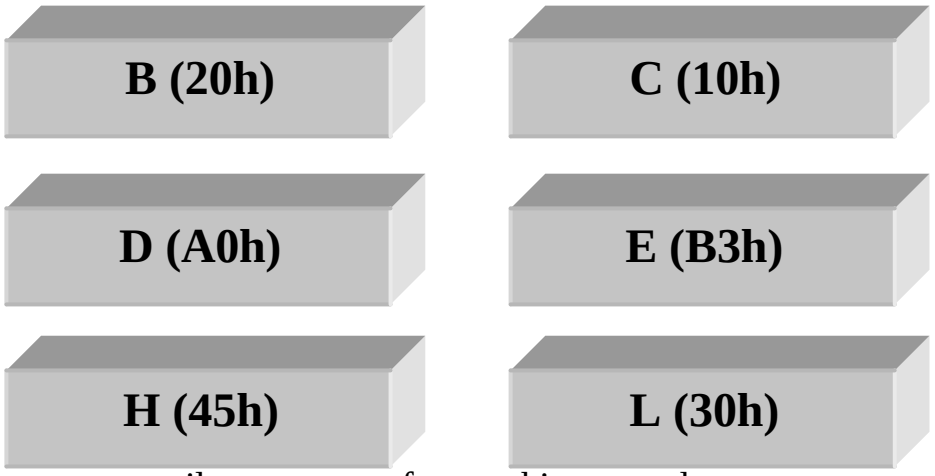
Nello Z80 esiste anche un banco di registri gemelli dell'accumulatore, del registro dei flag e dei 6 registri di uso generale



Questi registri, però, non possono essere utilizzati contemporaneamente ai registri del banco principale. Non si può effettuare ad esempio, un'operazione in cui si somma il registro accumulatore del banco principale e il registro B' del banco alternativo.

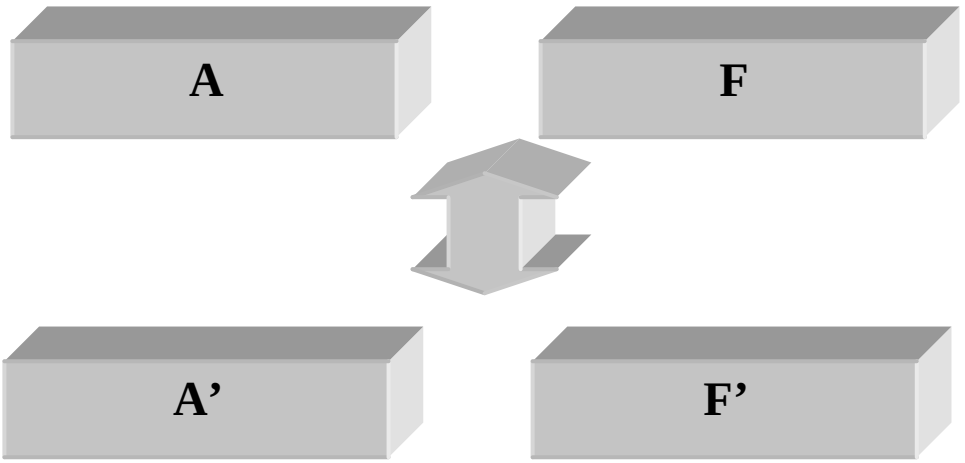
ADD A, B' Vietata

Questi registri possono essere utilizzati soltanto come aree temporanee di memorizzazione dei dati. Supponiamo, ad esempio, che il microprocessore stia eseguendo un programma che abbia accumulato i seguenti dati in alcuni registri



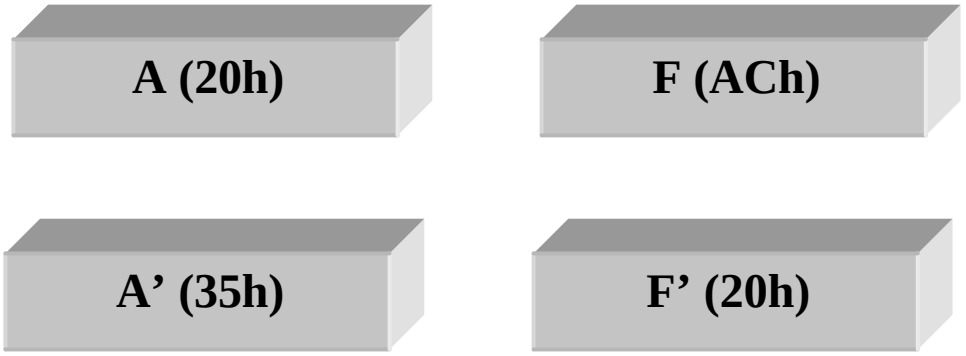
A questo punto il programma fa una chiamata ad un sottoprogramma il quale ha bisogno , a sua volta, di utilizzare i registri B, C, ed E. Agendo su questi registri, il sottoprogramma ne distruggerebbe il contenuto, per cui i rusltati delle elaborazioni condotte fino a quel momento dal programma principale verrebbero persi. Il sottoprogramma può ovviare a questa difficoltà salvando preliminarmente il contenuto dei registri del banco principale nel banco dei registri alternativi. Per far questo abbiamo a

disposizione due operazioni Ex e EXX. La prima scambia il contenuto dei registri A ed F con A' e F'

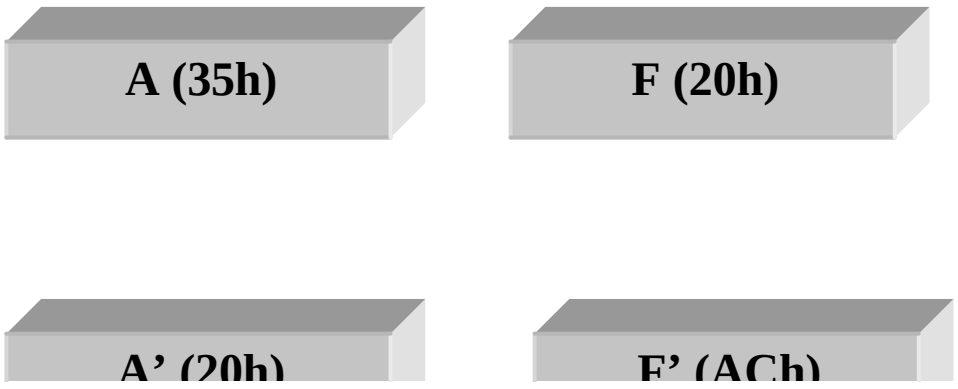


Esempio:

Contenuto dei registri prima dell'esecuzione dell'istruzione EX

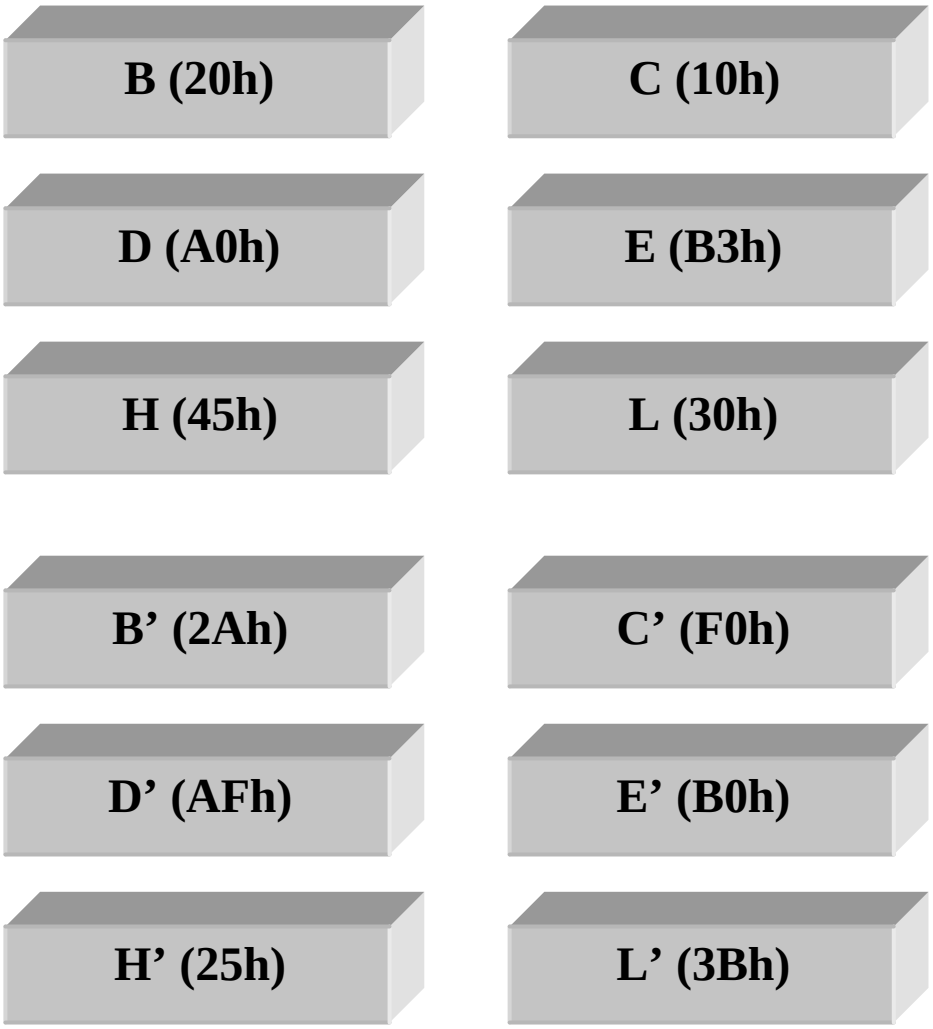


Contenuto dei registri dopo l'esecuzione dell'istruzione

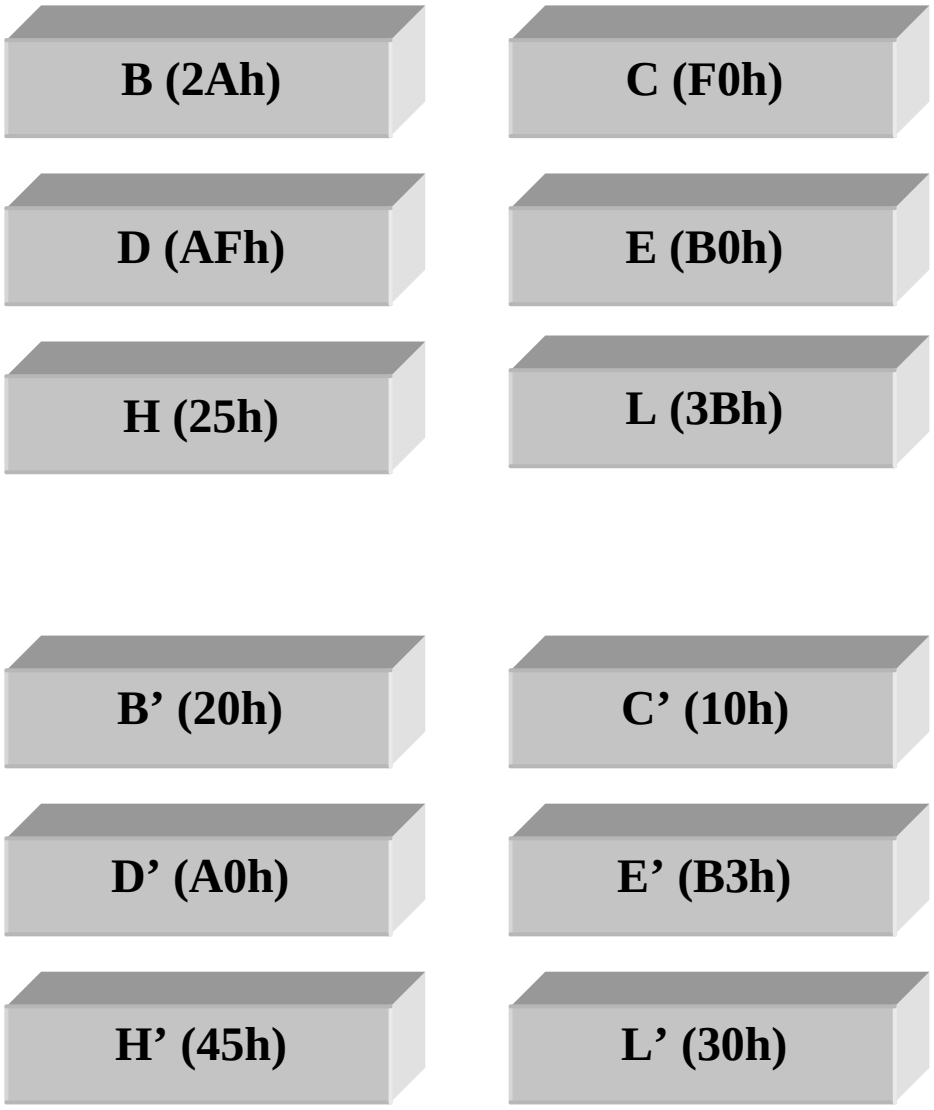


L’istruzione EXX consente, invece, di scambiare il contenuto dei registri BC, DE; HL e dei registri BC’, DE’, HL’

Esempio. Prima:

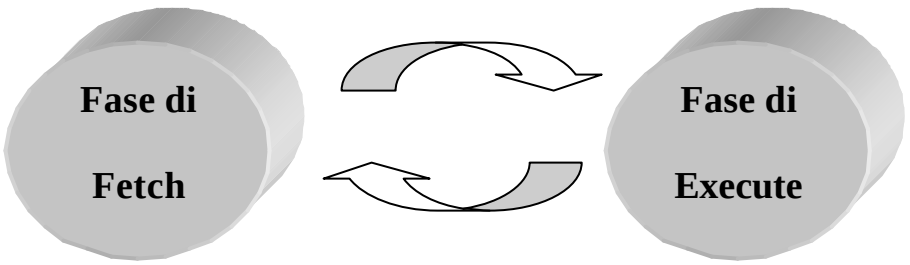


Dopo



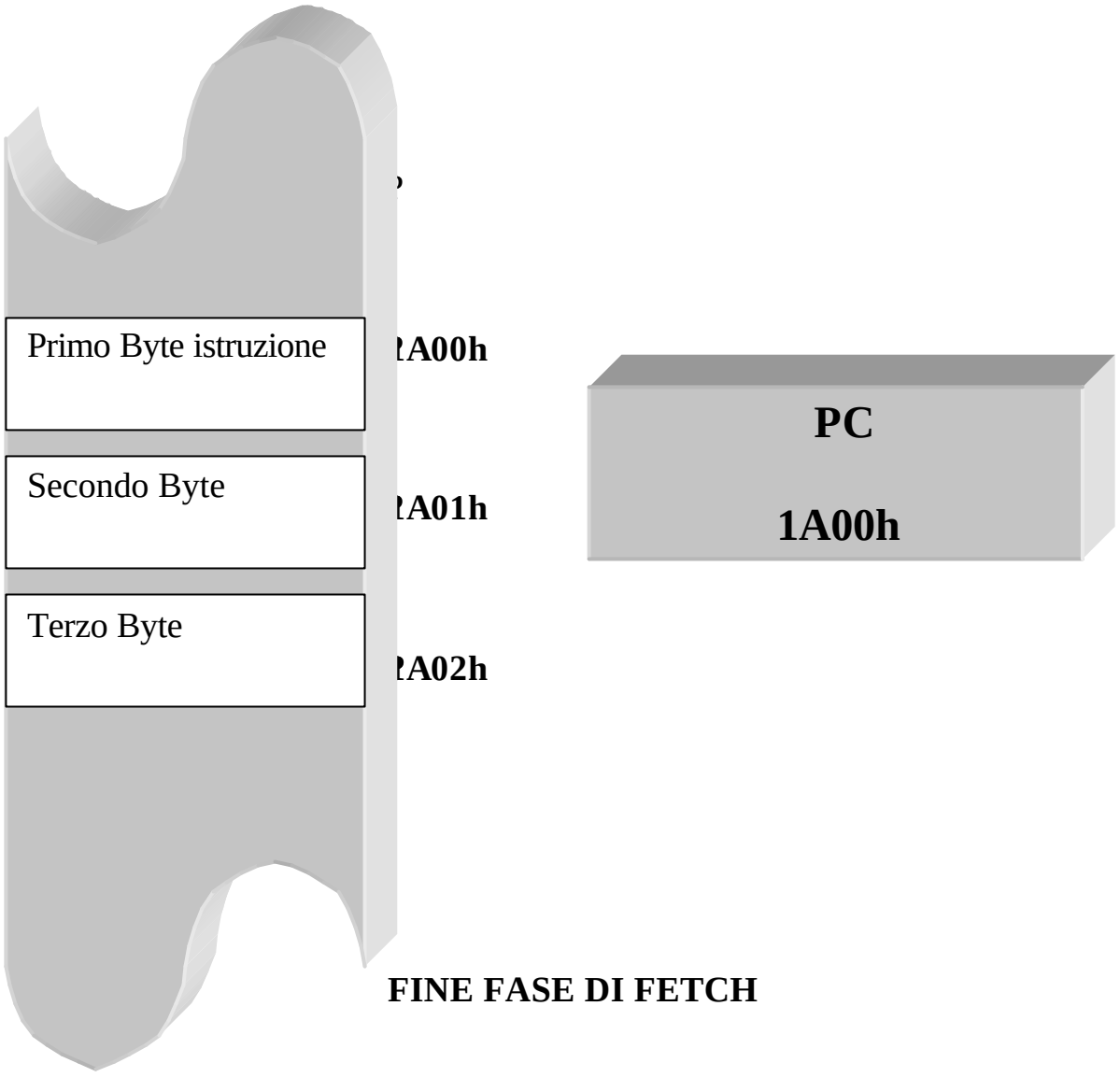
2. Program Counter

Il Program Counter è un registro a 16 bit che contiene l'indirizzo della prossima istruzione da eseguire. Ricordiamo che il microprocessore è un automa sequenziale che oscilla far due macrostati

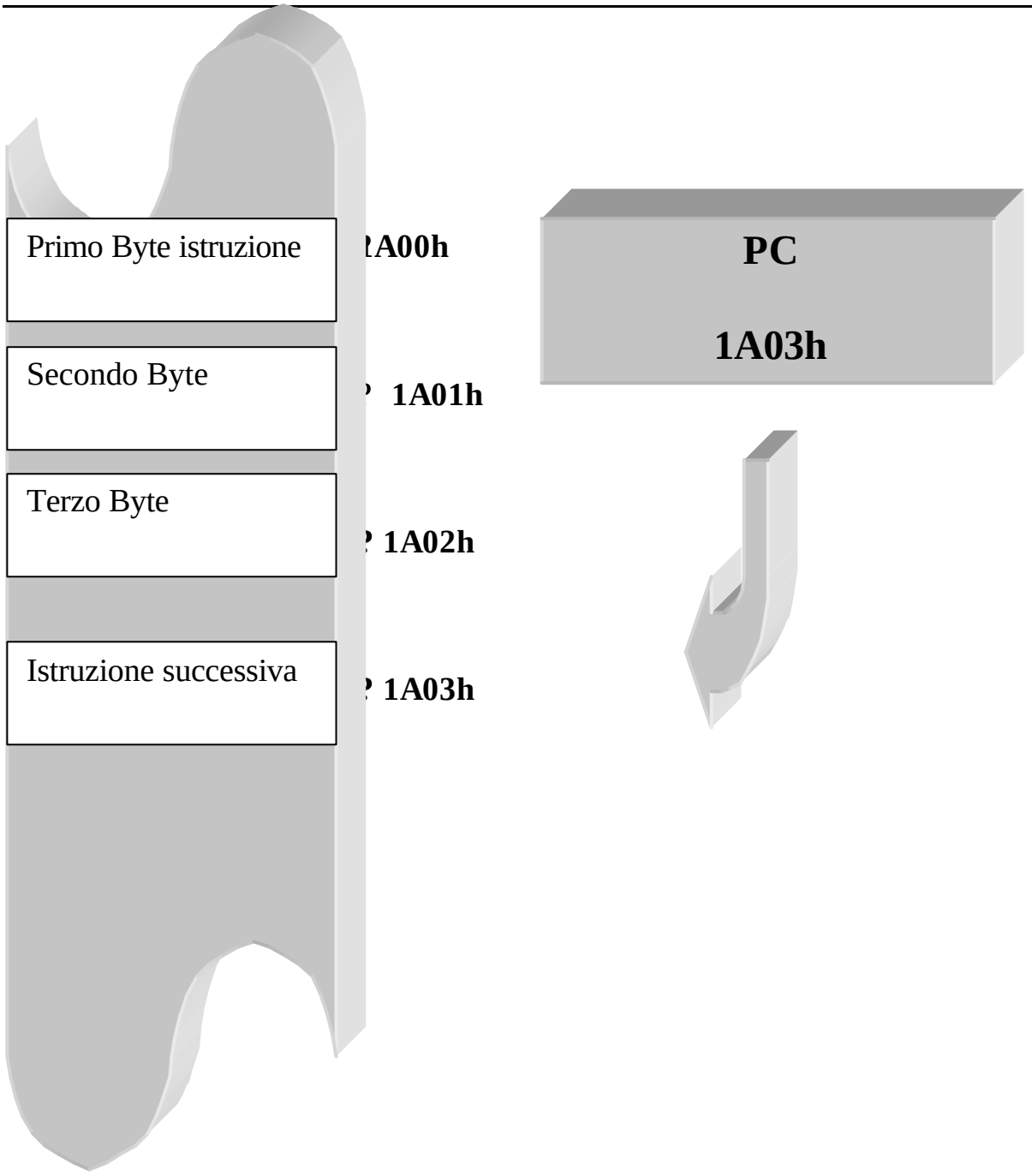


Nella fase di Fetch il microprocessore accede alla memoria prelevando il contenuto della locazione di memoria il cui indirizzo è contenuto nel Program Counter. Il Program Counter viene, in questa fase, automaticamente incrementato di uno in modo tale che, se l'istruzione si compone di un solo byte, nella successiva fase di fetch, il PC punterà automaticamente alla locazione successiva contenente la prossima istruzione da eseguire. Se, invece, l'istruzione si compone di più byte, il microprocessore li preleva dalla memoria e il PC viene incrementato ulteriormente.

INIZIO FASE DI FETCH



FINE FASE DI FETCH



3. Stack Pointer

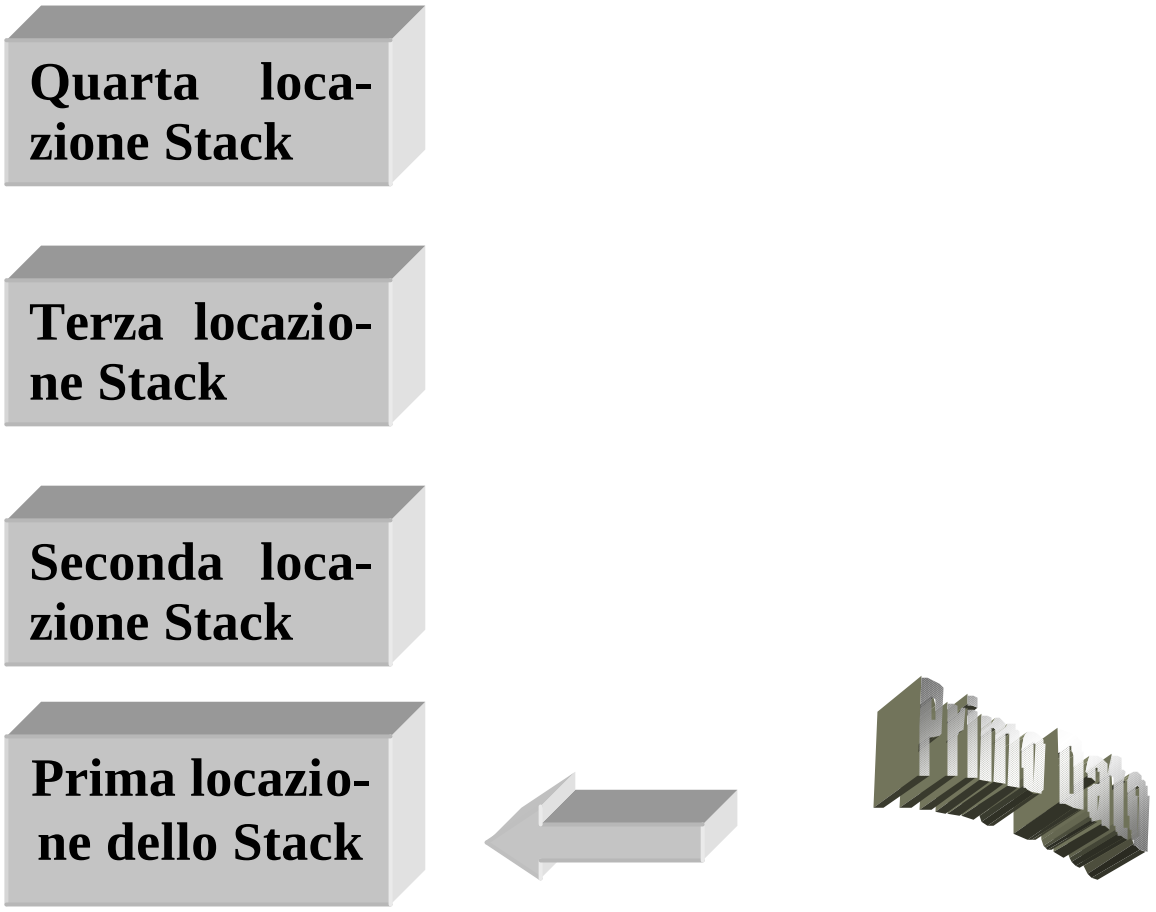
Lo Stack Pointer è un registro a 16 bit che punta alla cima di un'area particolare di memoria denominata stack.

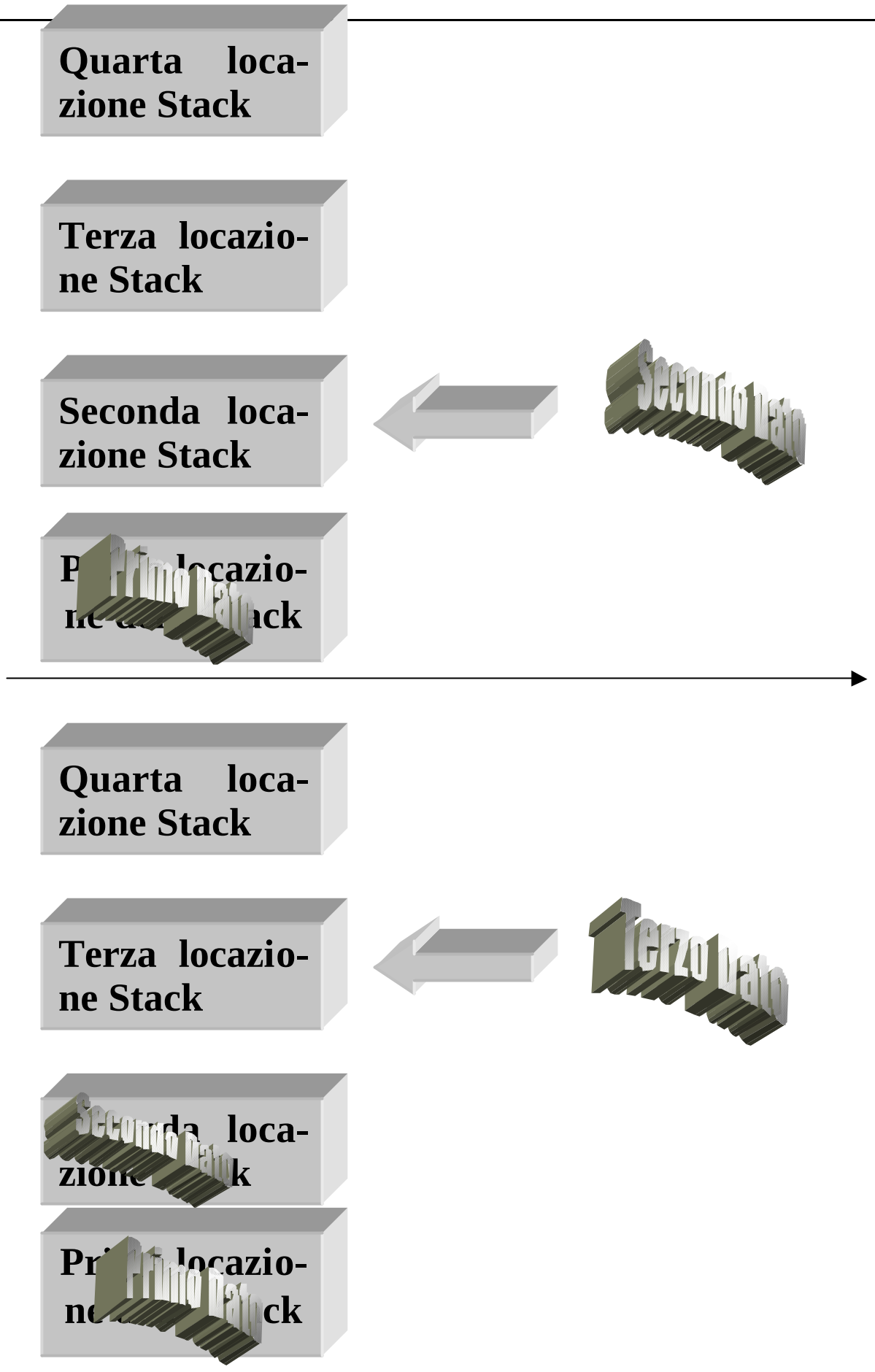
Lo Stack

Normalmente i banchi di memoria collegati ad un microprocessore sono organizzati in modo che si possa accedere ad essi in modo casuale, laddove il termine casuale va

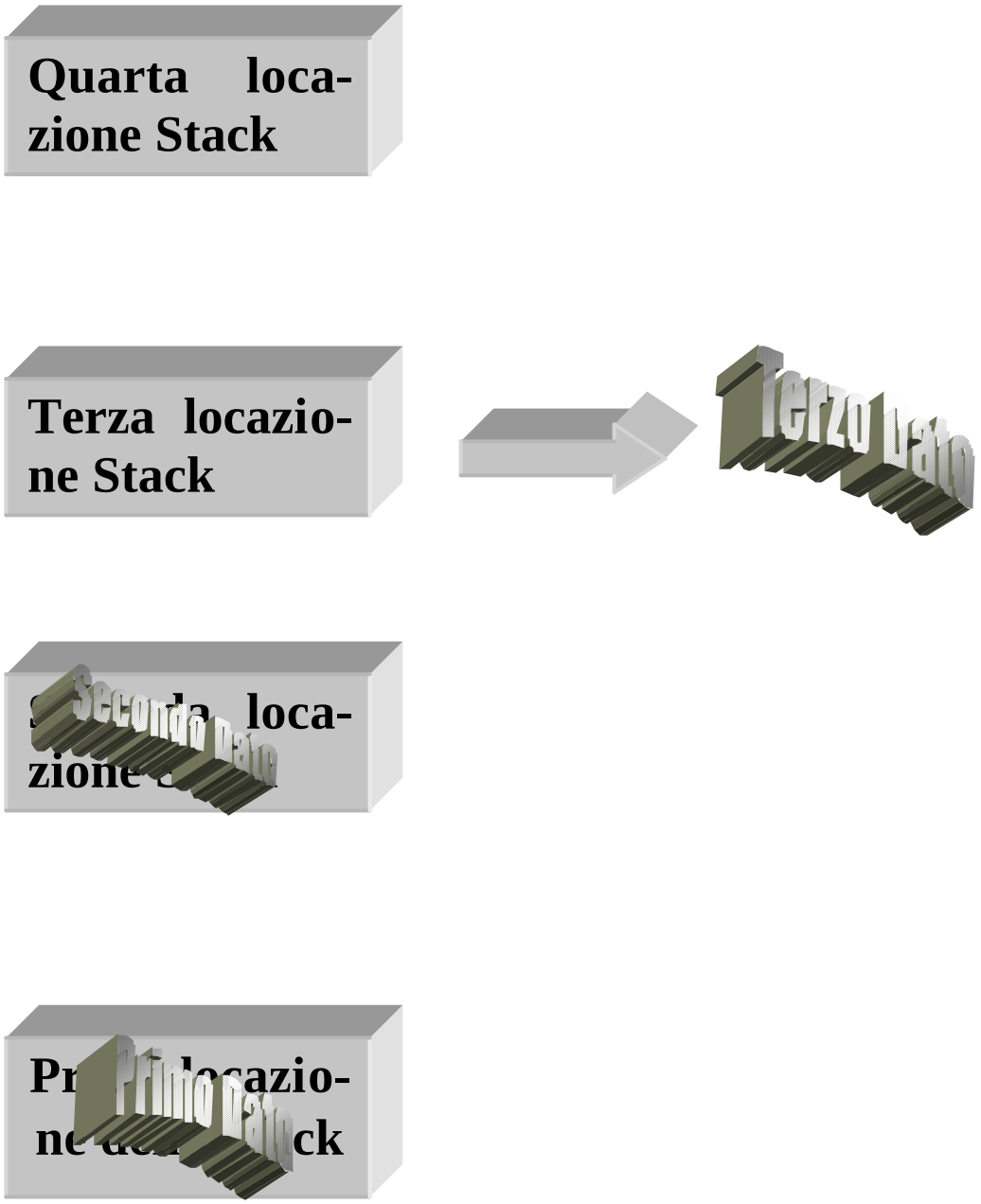
inteso nel senso che il microprocessore può decidere di accedere ad una qualsiasi delle locazioni, senza alcun vincolo che non sia il programma che sta eseguendo.

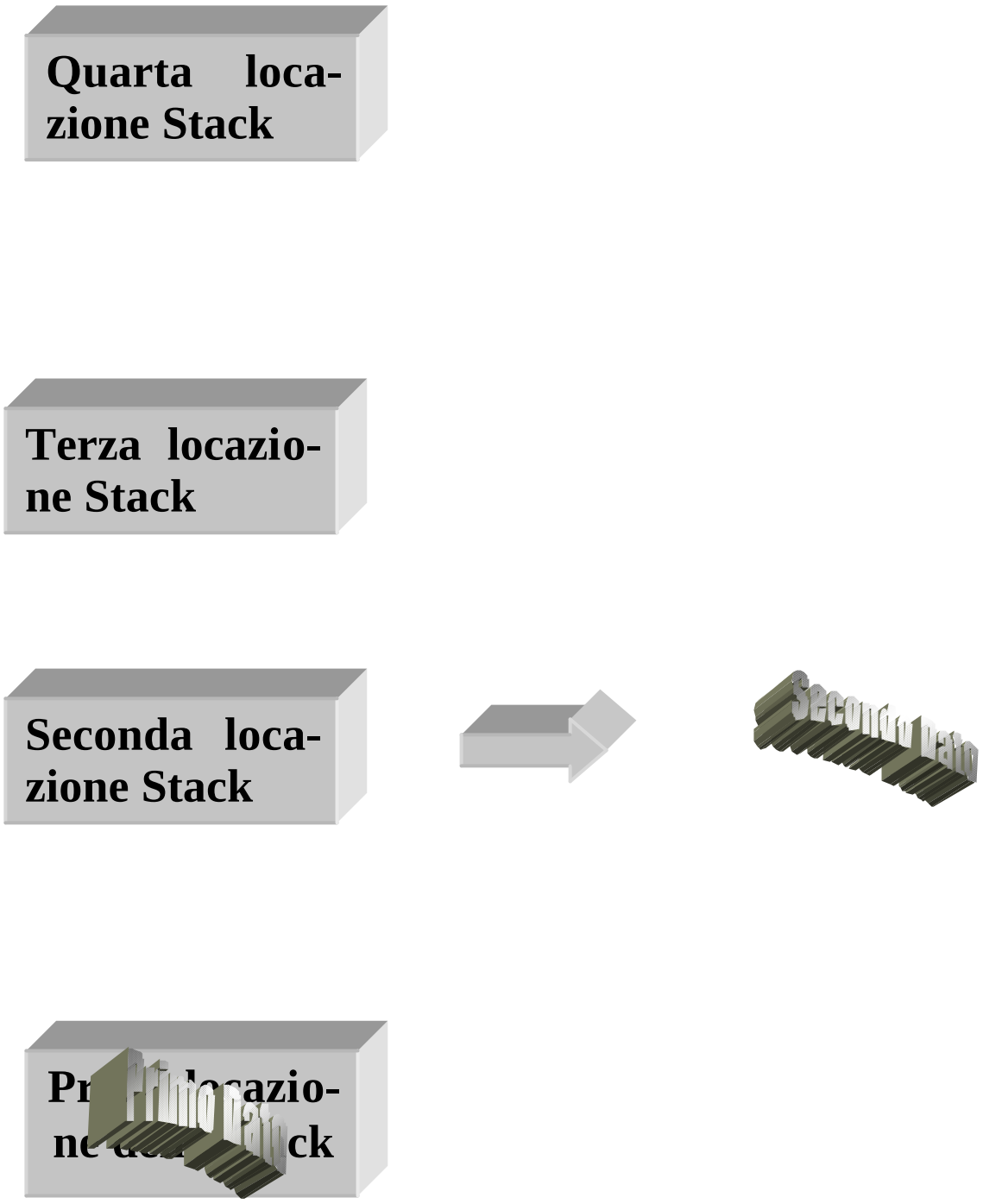
Lo Stack è, invece, un'area di memoria organizzata secondo la struttura LIFO (Last In First Out = L'ultimo ad entrare è il primo ad uscire). In sostanza in un'area di memoria che abbia questa organizzazione, un nuovo dato può essere inserito soltanto nella prima locazione libera in cima allo stack.




Quarta locazione Stack
Terza locazione Stack
Seconda locazione Stack
Secondo dato
Primo locazione Stack
Primo dato
Quarta locazione Stack
Terza locazione Stack
Terzo dato
Seconda locazione Stack
Primo locazione Stack
Primo dato

Inversamente, per prelevare un dato dallo stack , occorre prima prelevare tutti i dati che sono stati inseriti dopo di esso, nell'ordine inverso a quello in cui sono stati inseriti.

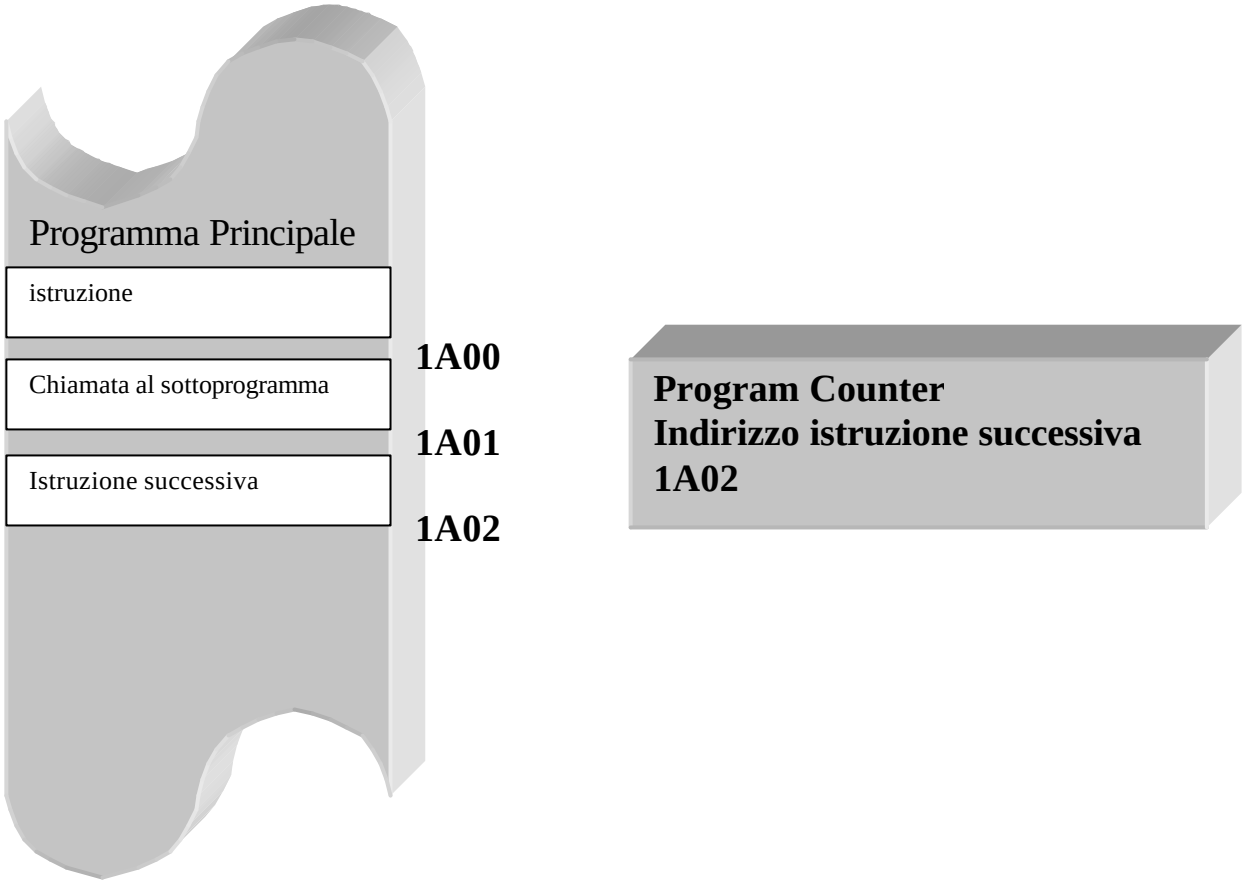


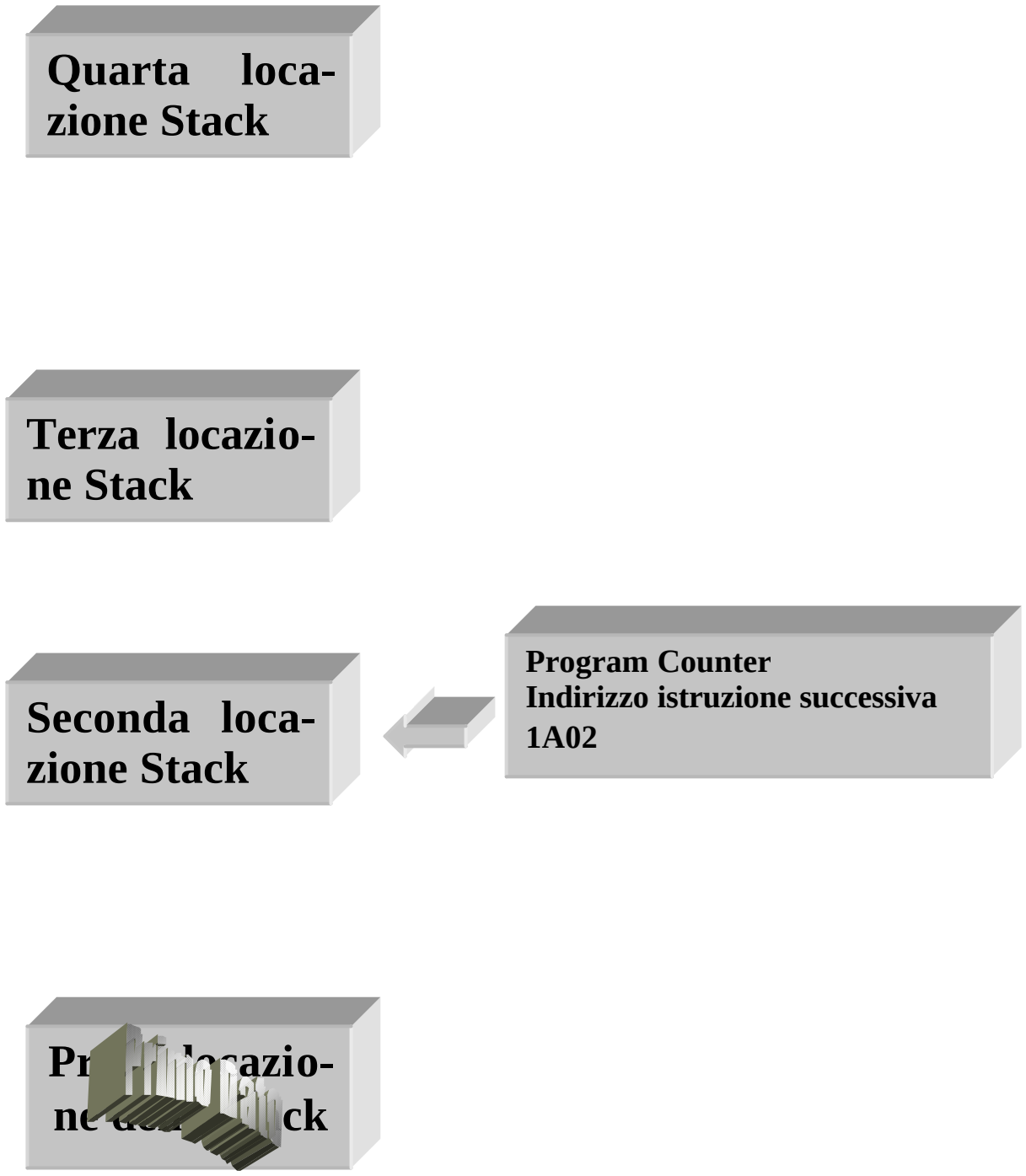


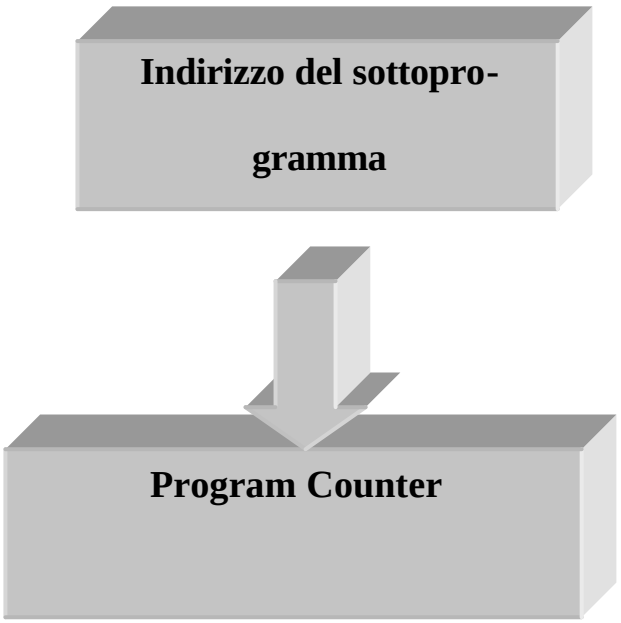
A cosa serve lo Stack? La nidificazione dei sottoprogrammi.

Un'utilizzazione della particolare organizzazione dello stack si ha nella nidificazione o nesting dei sottoprogrammi.

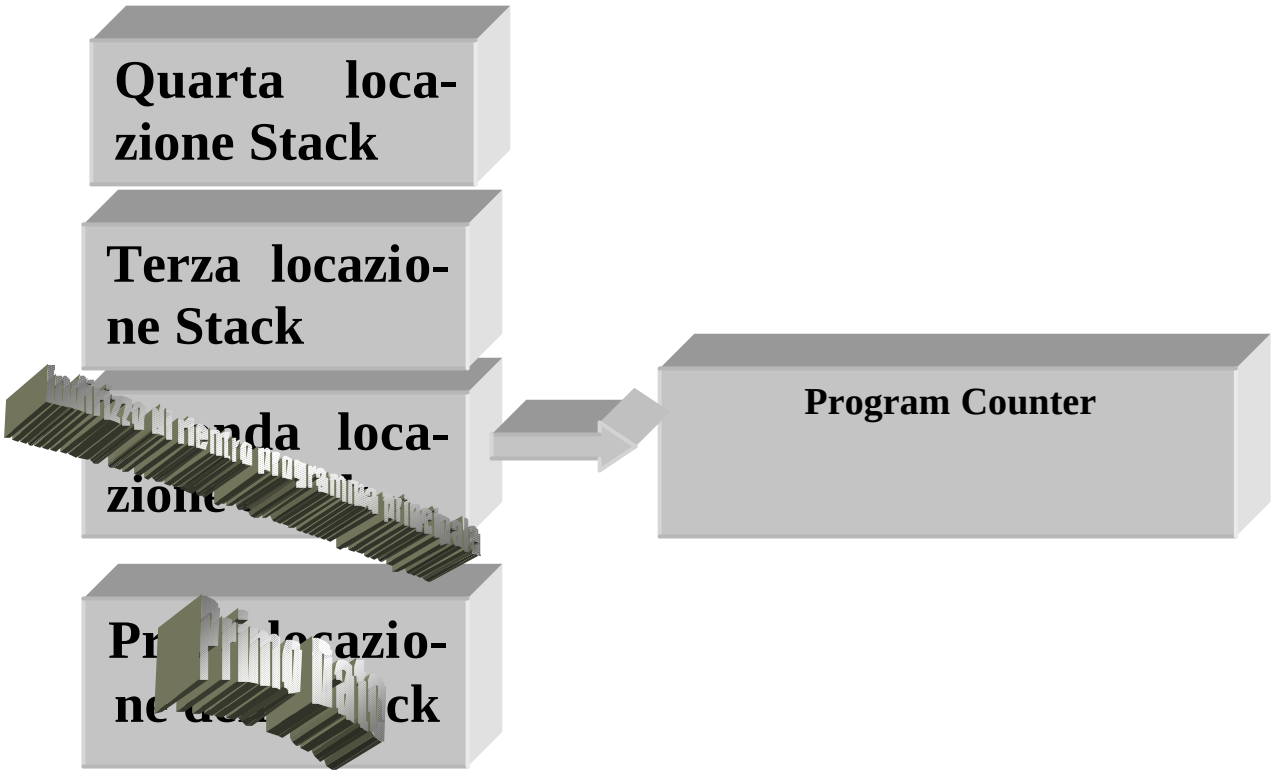
Quando un programma chiama un sottoprogramma ciò che avviene è che il contenuto del PC (che contiene l'indirizzo della prossima istruzione del programma principale da eseguire dopo la chiamata del sottoprogramma) va sostituito con l'indirizzo della prima istruzione del sottoprogramma. Al termine del sottoprogramma si deve ritornare al programma principale, ma ciò significa che il vecchio contenuto del PC va ripristinato. Esso prende il nome di *indirizzo di rientro.* Perché possa essere ripristinato deve essere chiaramente stato precedentemente salvato nello stack.



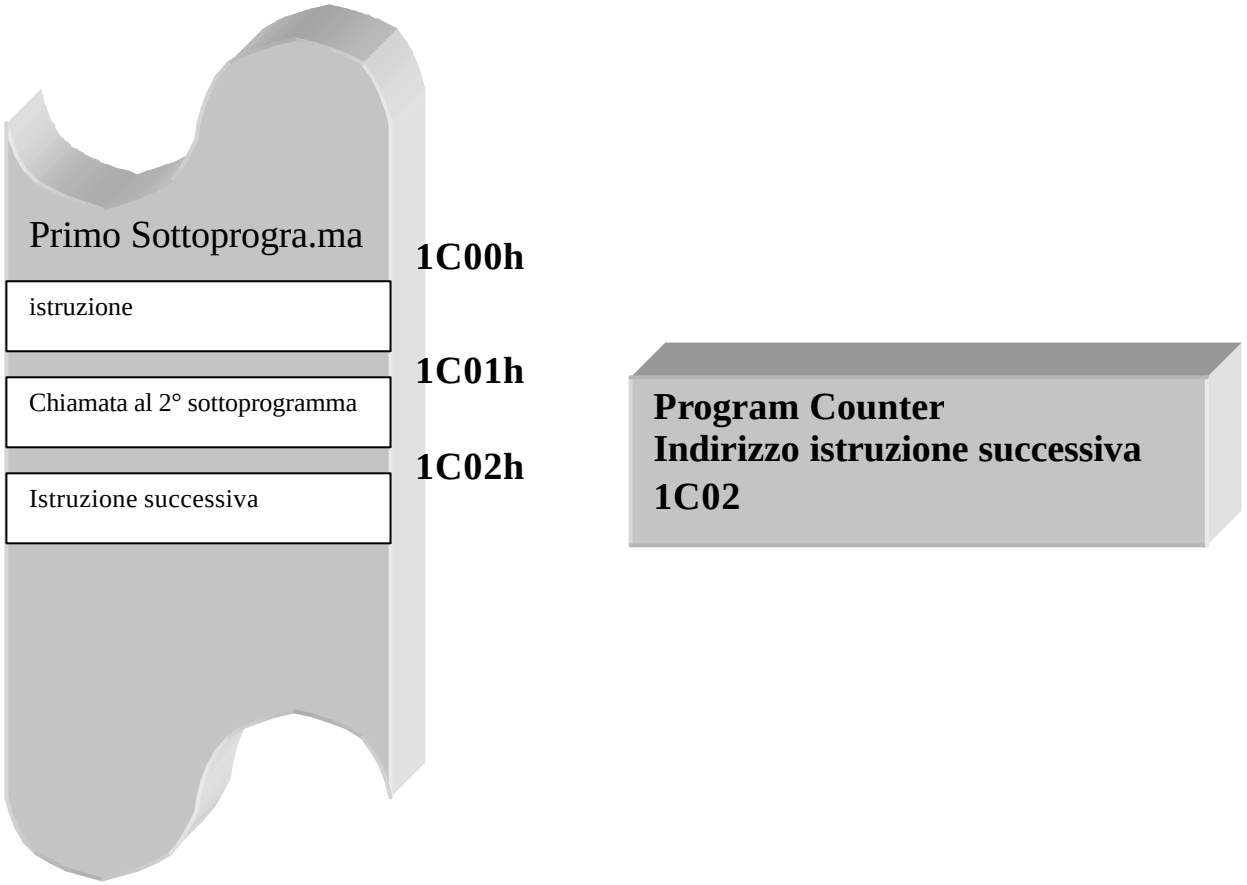

Quarta locazione Stack
Terza locazione Stack
Seconda locazione Stack
Program Counter
Indirizzo istruzione successiva
1A02
Prima locazione Stack

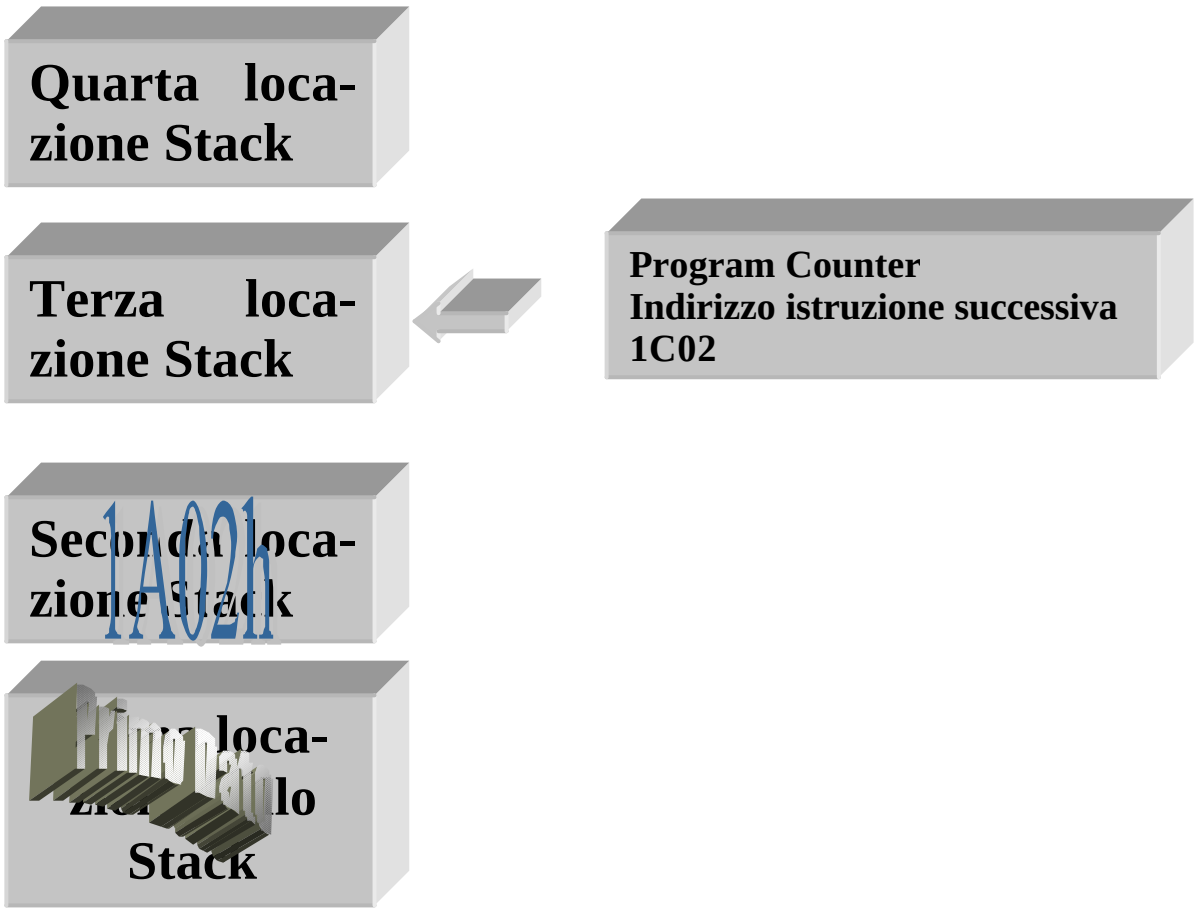


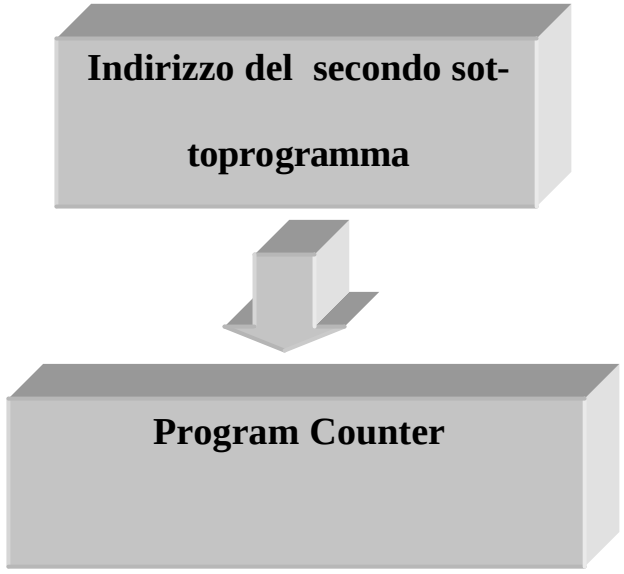
Al termine del sottoprogramma



Può ora accadere che il sottoprogramma chiami a suo volta un altro sottoprogramma, il che significa che bisogna salvare anche l'indirizzo di rientro del primo sottoprogramma.




Quarta locazione Stack
Terza locazione Stack
Program Counter
Indirizzo istruzione successiva
1C02
Seconda locazione Stack
1A02h
Stack


Indirizzo del secondo sottoprogramma
Program Counter

Quando il secondo sottoprogramma termina bisogna tornare al primo sottoprogramma e non al programma principale. Soltanto quando anche il primo sottoprogramma è terminato si può tornare al programma principale. Ciò significa che gli indirizzi di rientro vanno ripristinati nell'ordine inverso a quello in cui sono stati memorizzati nello stack. La struttura LIFO di questo rende la cosa automatica.

Funzione dello Stack Pointer

A causa della sua organizzazione, lo stack è una struttura dinamica nel senso che cresce quando vi vengono inseriti dei dati mentre rimpicciolisce quando vengono prelevati dei dati. Ciò significa che occorre qualcosa che indichi in ogni istante in quale posizione si trova la cima dello stack. Lo Stack Pointer ha la funzione di indicare la cima dello stack poiché contiene l'indirizzo dell'ultima locazione riempita.

4. Registri per la gestione delle interruzioni

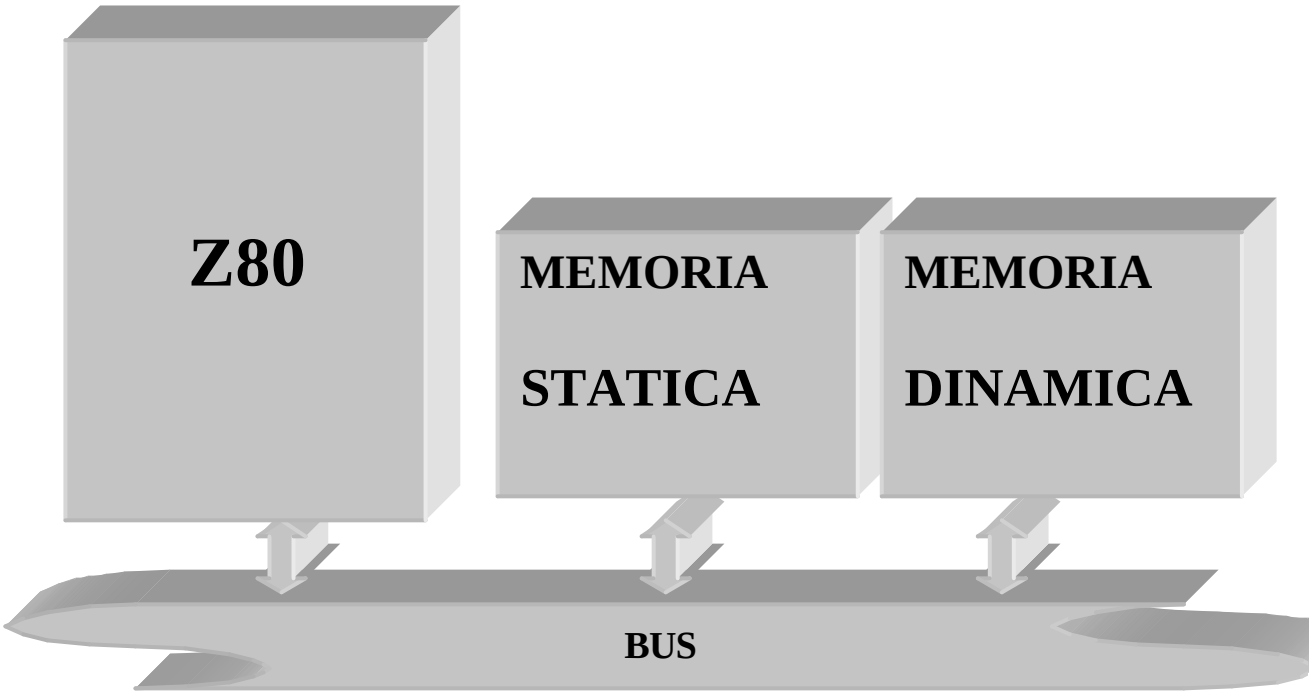
Nello Z80 troviamo il registro ad 8 bit I, il registro a due bit IMF, i due registri ad un bit IFF1 e IFF2. Questi registri verranno descritti quando parleremo della gestione delle interruzioni.

5. Registri indice

I registri indice IX ed IY sono due registri a 16 bit. Essi vengono utilizzati nell'indirizzamento indicizzato che descriveremo più avanti.

6. Registro di Refresh

Il registro R di refresh è ad 8 bit. Esso viene utilizzato nel rinfresco delle memorie dinamiche. Ricordiamo che ad un microprocessore si possono collegare memorie statiche e dinamiche



Le memorie statiche sono costituite da flip flop integrati mentre le memorie dinamiche sono memorie in cui ogni singola cella è realizzata sfruttando l'effetto capacitivo della giunzione di gate di un transistor MOS. Le celle delle memorie dinamiche hanno l'inconveniente di tendere a scaricarsi con l'andare del tempo per cui l'informazione in esse contenuta, tende a degradare e a non essere più intelligibile. Lo Z80 ovvia al problema effettuando periodicamente dei cicli di rinfresco della memoria dinamica, cioè legge il contenuto delle locazioni e lo riscrive, ricaricando così le celle di memoria. Il registro R contiene l'indirizzo della locazione da rinfrescare.

7. Registro dei flag

Il registro dei flag è un registro ad 8 bit che contiene informazioni sullo stato del microprocessore, informazioni che possono essere utilizzate nello svolgimento di un programma.

S	Z	X	H	X	P/V	N	C

Il registro dei flag si differenzia dagli altri registri poiché, nel suo caso, l'informazione non è racchiusa nella stringa di otto bit, considerata nel suo complesso ma in ogni singolo bit, preso separatamente dagli altri.

Flag S o di segno

Questo flag ci dice se l'ultima operazione eseguita dal microprocessore ha dato un risultato positivo o nullo. Se S=0, il risultato è positivo, se S=1 il risultato è negativo.

Flag Z o di segno

Questo flag ci dice se l'ultima operazione ha dato un risultato nullo o diverso da zero. Se Z=0, il risultato è positivo, se Z=1 il risultato è negativo.

Flag H o mezzo riporto.

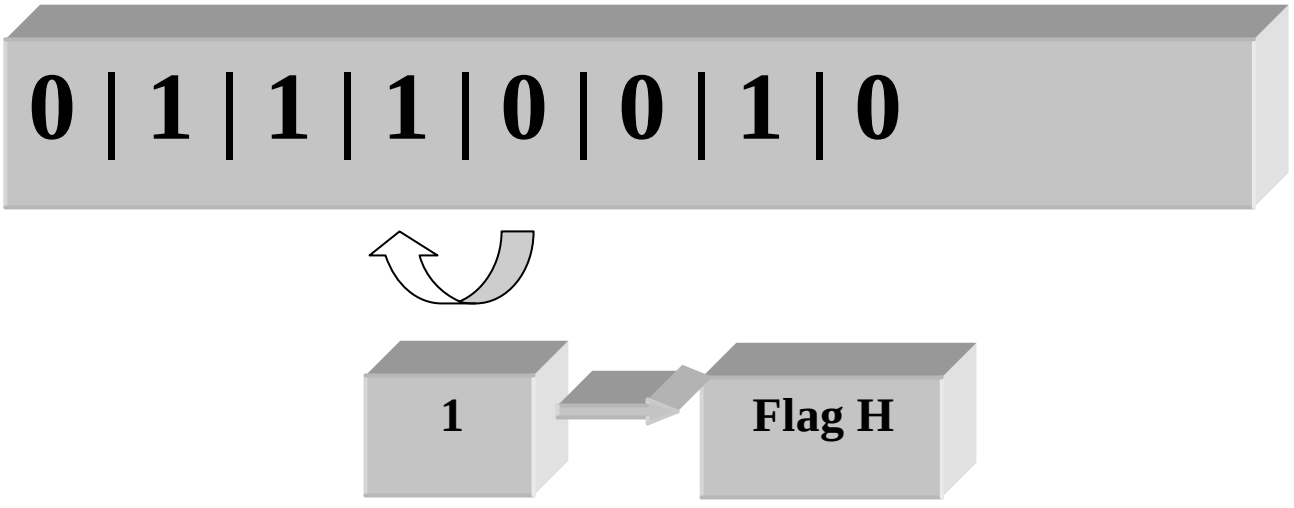
Questo flag indica, nelle somme ad 8 bit, se vi è stato un riporto fra il bit 3 (o quarto bit) ed il bit 4 (o quinto bit).

Esempio

0 | 1 | 1| 0 | 1 | 0 | 1 | 0

+

=

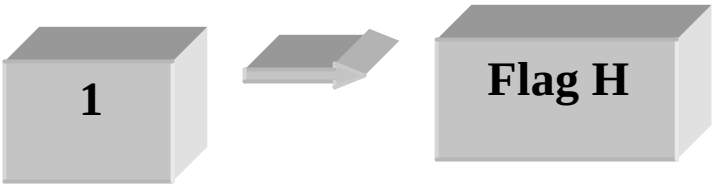


Per le sottrazioni ad 8 bit questo flag indica un prestito dal bit 4 (quinto bit) al bit 3 (quarto bit)

Esempio

-

=



Se si tratta di un'addizione fra due dati a 16 bit, l'half carry viene settato ad 1 se vi è un riporto fra terzo e quarto nibble, cioè tra il dodicesimo bit (bit 11) ed il tredicesimo bit (bit 12). Nel caso di una sottrazione a 16 bit, l'half carry viene settato ad 1 se vi è

un prestito fra quarto e terzo nibble, cioè tra il tredicesimo bit (bit 12) ed il dodicesimo bit (bit 11) .

Flag P/V di parità/overflow

Questo flag ha doppia denominazione poiché ha una doppia funzione. Nel caso d'operazioni aritmetiche, il flag ha funzione di flag d'overflow. Si ha un errore d'overflow, quando un'operazione dà un risultato che eccede il range di valori che si possono memorizzare in un registro.

Un registro ad 8 bit può contenere un dato compreso tra 0 e 255 se memorizza soltanto dati positivi. Se memorizza dati con segno, allora può contenere un dato compreso tra –128 e +127. analogamente un registro a 16 bit può contenere un dato variabile fra 0 e 65535, oppure fra –32768 e +32767.

Supponiamo, allora, di eseguire una somma fra due dati ad 8 bit con segno, ad esempio +110 e +60.

$$\mathbf{110|_{10} = 01101110|_2}$$

$$\mathbf{60|_{10} = 00111100|_2}$$

$$\mathbf{110|_{10} + 60|_{10} = 170|_{10}}$$

fuori dal range di valori memorizzabili in un registro a 8 bit. In binario accade che

$$\mathbf{01101110|_2}$$

$$\mathbf{+}$$

$$\mathbf{00111100|_2}$$

=

$$\mathbf{10101010|_2}$$

Già possiamo notare la prima anomalia poiché, nella rappresentazione binaria con segno, un numero positivo ha il bit più significativo, il bit 7, pari a zero. Il nostro risultato ha il bit più significativo pari ad 1. questo significa che la somma fra due numeri positivi avrebbe dato un numero negativo! Per essere più precisi

$$\mathbf{10101010|_2}$$

↓

$$\mathbf{01010101|_{2\ complemento\ ad\ uno}}$$

↓

$$\mathbf{01010111|_{2\ complemento\ a\ due}}$$

pari a $\mathbf{-87|_{10}}$. Se non vi fosse alcun meccanismo per rilevare l'occorrenza di questi tipi di errori, il programma proseguirebbe operando su risultati del tutto errati con esiti disastrosi. Il flag di overflow viene settato ad 1 quando incorre questo tipo di errore e può essere utilizzato per segnalare all'utente ciò che è avvenuto.

Uso di P/V come bit di parità

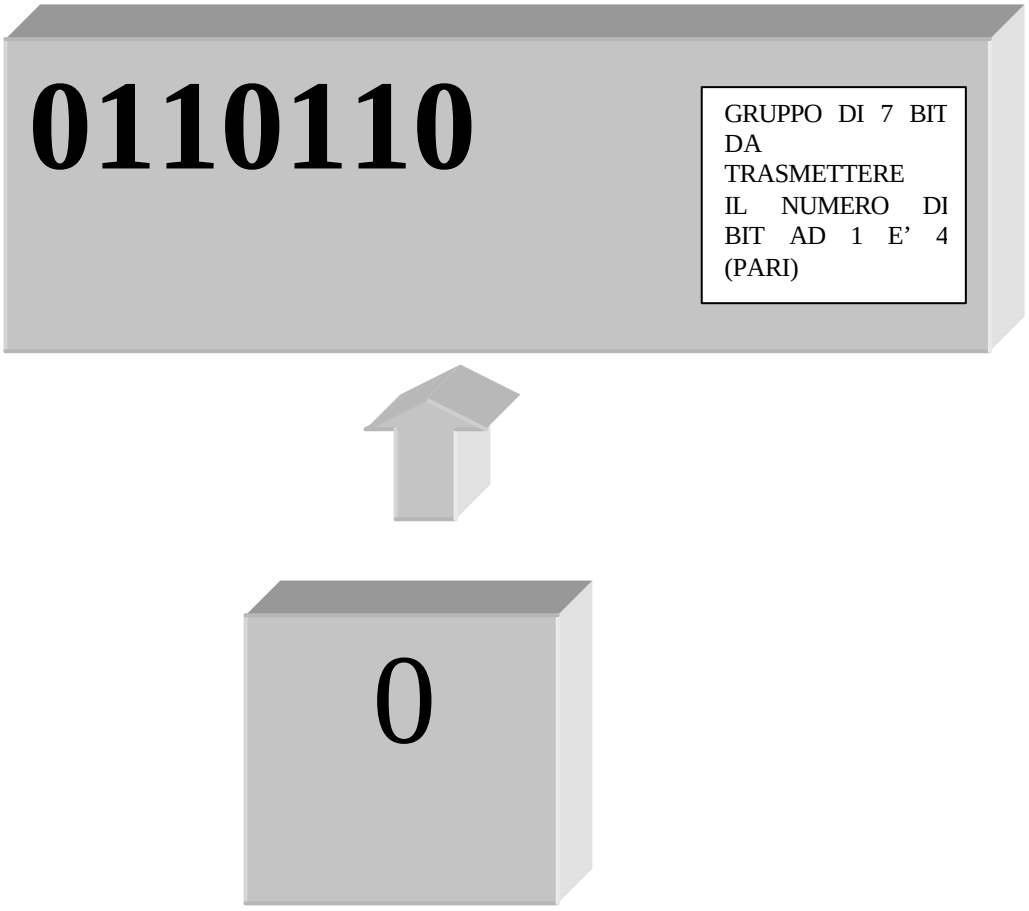
Nelle trasmissioni dei dati occorre verificare che non vi siano stati errori che abbiano trasformato il dato trasmesso. Per un dispositivo di memorizzazione di massa occorre invece verificare l'integrità dei dati conservati su di esso. Il metodo più semplice per verificare la presenza di errori nei dati ricevuti è quello del controllo di parità.

I dati sono organizzati in blocchi di 7 bit. Ad ogni blocco viene aggiunto un bit di controllo.

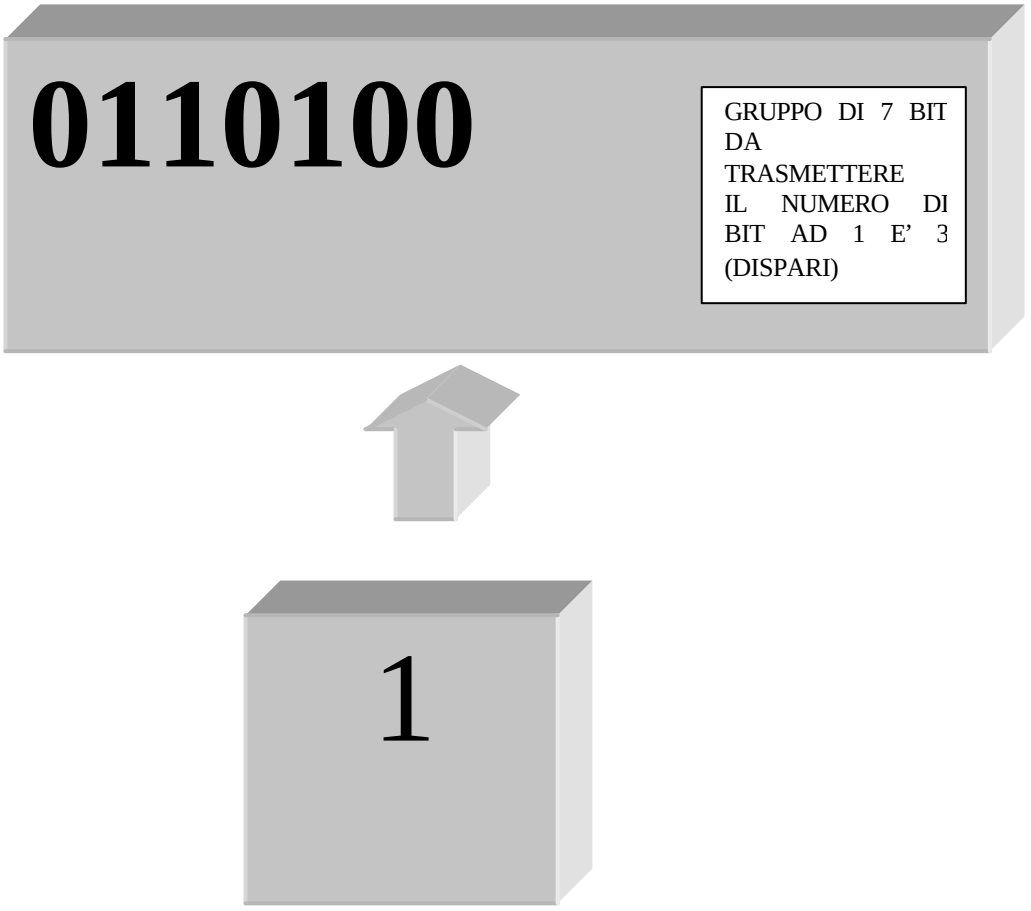
Nel metodo della parità pari si fa in modo che il numero complessivo di bit ad 1 sia pari. Allora se nel gruppo di sette bit che costituiscono il dato il numero di bit ad 1 è già pari, il bit di controllo viene posto a zero, in modo che il numero complessivo di bit a 1 rimanga pari. Se, invece, il numero di bit ad 1 nel dato è dispari, il bit di controllo viene posto a 1, in modo che il numero complessivo di bit ad 1 diventi pari.

Nel metodo della parità dispari si fa in modo che il numero complessivo di bit ad 1 sia dispari. Allora se nel gruppo di sette bit che costituiscono il dato il numero di bit ad 1 è già pari, il bit di controllo viene posto a 1, in modo che il numero complessivo di bit a 1 diventi dispari. Se, invece, il numero di bit ad 1 nel dato è dispari, il bit di controllo viene posto a 0, in modo che il numero complessivo di bit ad 1 rimanga dispari.

PARITA' PARI

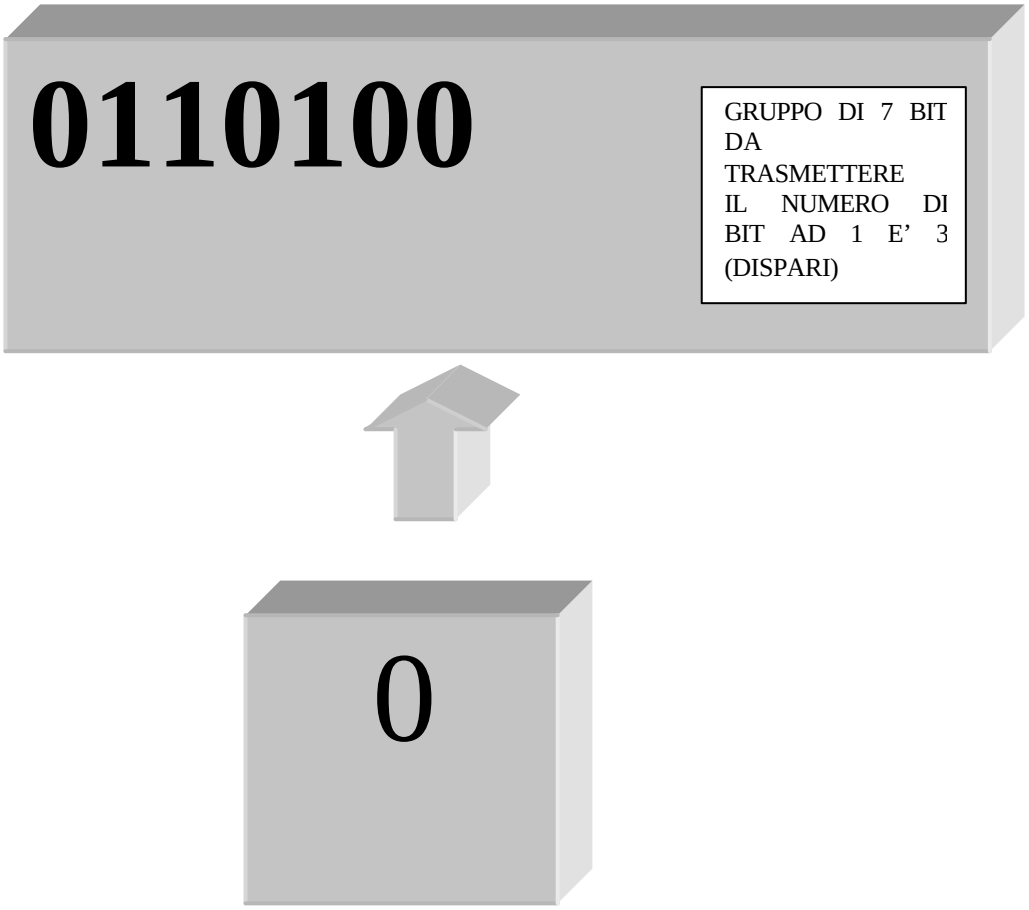


VIENE AGGIUNTO UN BIT DI CONTROLLO PARI A ZERO IN MODO CHE IL NUMERO TOTTALE DI BIT AD 1 RIMANGA 4 (NUMERO PARI)

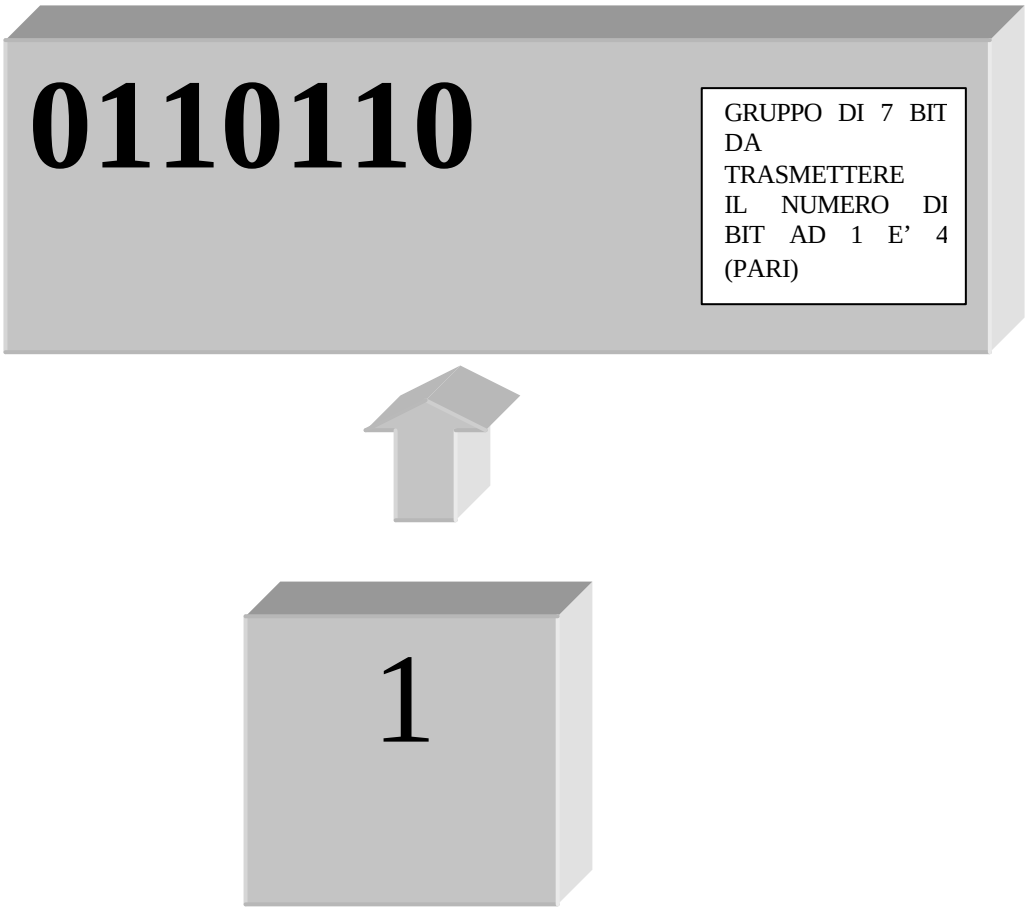


VIENE AGGIUNTO UN BIT DI CONTROLLO PARI A UNO IN MODO CHE IL NUMERO TOTALE DI BIT AD 1 DIVENGA 4 (NUMERO PARI)

PARITA' DISPARI



VIENE AGGIUNTO UN BIT DI CONTROLLO PARI A ZERO IN MODO CHE IL NUMERO TOTALE DI BIT AD 1 RIMANGA 3 (NUMERO DISPARI)



VIENE AGGIUNTO UN BIT DI CONTROLLO PARI A UNO IN MODO CHE IL NUMERO TOTALE DI BIT AD 1 DIVENGA 5 (NUMERO DISPARI)

Supponiamo ora di utilizzare la parità dispari e che la sorgente abbia trasmesso il seguente dato

E che durante la trasmissione incorra un errore che trasformi il bit 7 trasformandolo da 0 ad 1.

In ricezione riscontriamo un dato che ha un numero di bit ad uno pari a 5, cioè dispari anziché pari. Ciò significa che vi è stato un errore.

Nello Z80, se viene ricevuto da una unità di IO un dato con un numero di bit ad 1 dispari, il flag P/V sarà posto a zero. Se la trasmissione è regolata con la parità dispari, rilevare P/V=1 significa che è stato ricevuto un dato con un numero di bit pari, cioè vi è stato un errore nella trasmissione.

Flag N

E' il flag di sottrazione, se è ad 1 indica che la precedente operazione è stata una sottrazione, se è a 0 indica che la precedente operazione è stata una addizione.

Questo flag viene utilizzato per le operazioni in BCD. Ricordiamo che il codice BCD è un codice binario in cui in un gruppo di 4 bit o nibble, non sono consentite tutte le combinazioni possibili ma soltanto quelle che corrispondono alle cifre decimali da 0

a 9. le operazioni su dati binari e su dati BCD seguono regole diverse. In realtà lo Z80 considera sempre i dati come se fossero codificati in binario ed esegue le operazioni su di essi come se fossero dati binari. Successivamente all'esecuzione dell'operazione aritmetica si può utilizzare una particolare istruzione denominata DAA (Decimal Adjust Accumulator) che converte il risultato in codice BCD. L'istruzione DAA opera la conversione, seguendo un algoritmo diverso a seconda che l'operazione che ha prodotto il risultato sia stata un'addizione o una sottrazione. Quest'informazione è ricavata dal flag N.

Flag C

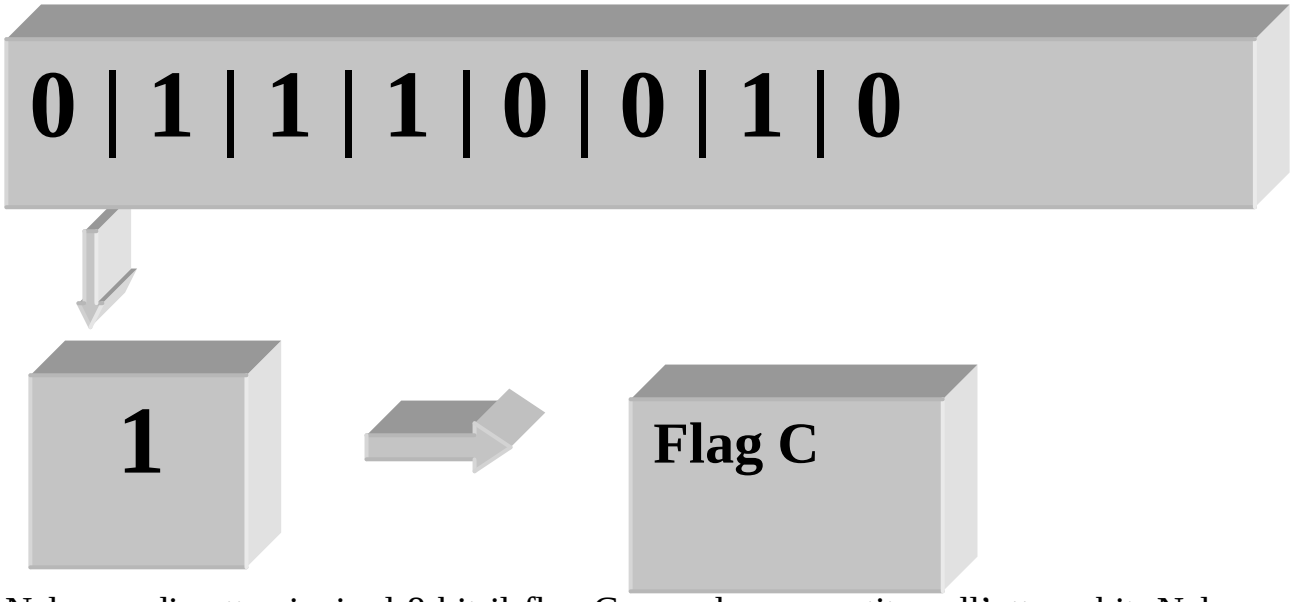
IL flag di riporto viene settato ad 1 se nelle addizioni ad 8 bit vi è un riporto sull'ottavo bit

1 | 1 | 1| 0 | 1 | 0 | 1 | 0

+

1 | 0 | 0 | 0 | 1 | 0 | 0 | 0

=



Nel caso di sottrazioni ad 8 bit il flag C segnala un prestito sull'ottavo bit. Nel caso d'operazioni a 16 bit, il flag C segnala un riporto sul 16esimo bit per le addizioni o un prestito sul 16esimo bit per le sottrazioni.

La gestione delle interruzioni nello Z80

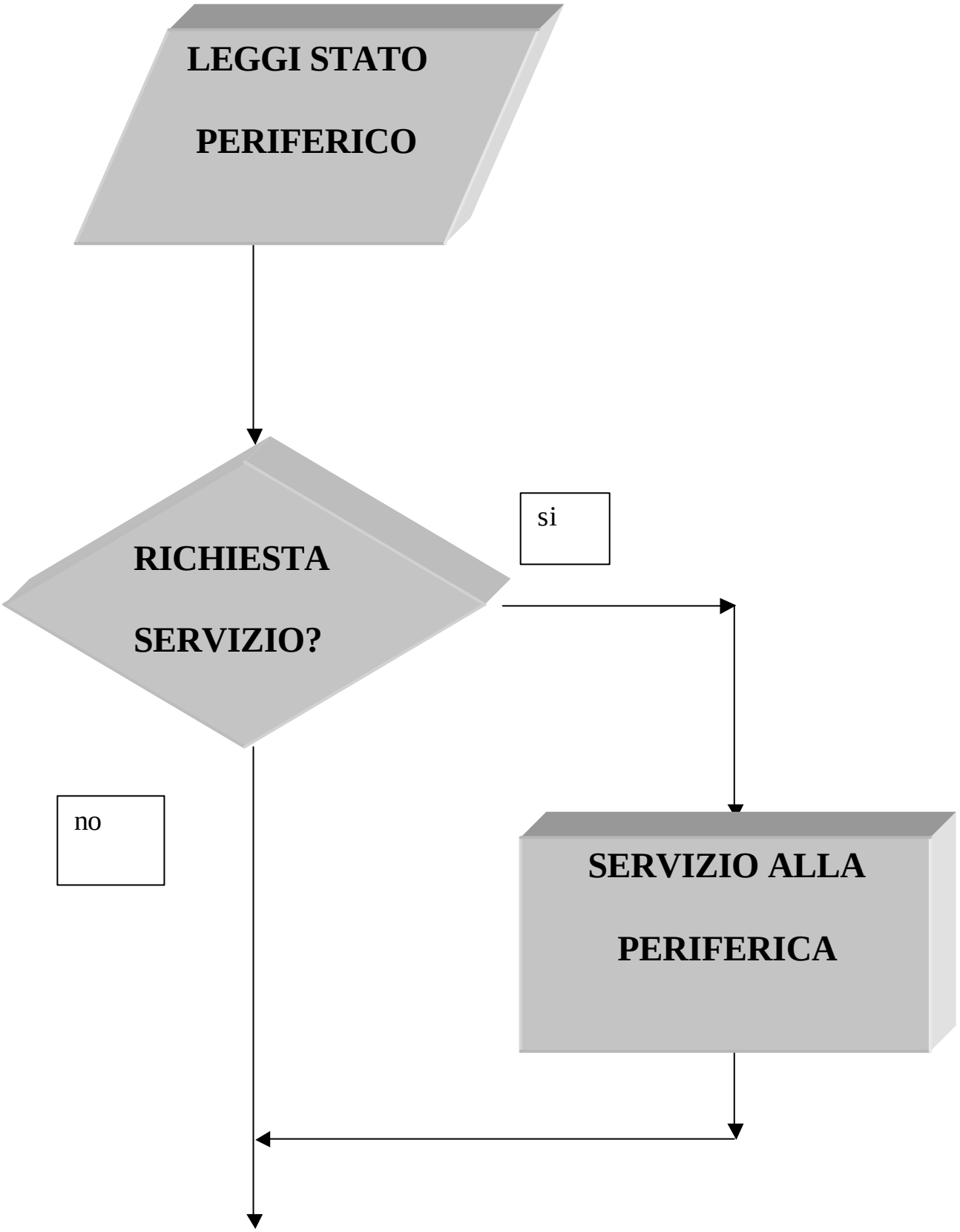
Cos'è un'interruzione?

Un microprocessore è un automa sequenziale il cui ingresso è costituito dal programma scritto in memoria. In questo modo il microprocessore sarebbe un "mondo" isolato dall'esterno. E' evidente che ciò non è possibile. Basta pensare al microprocessore di un Personal Computer: l'utente non avrebbe alcuna possibilità di intervenire da tastiera o con il mouse. Altro esempio è un sistema di controllo di un processo: il microprocessore non avrebbe alcuna possibilità di intervenire su eventi imprevisti.

La tecnica delle interruzioni è una tecnica per far gestire al microprocessore eventi asincroni al programma, dove per asincrono si intende un evento non previsto dal programma.

Il polling

Una prima soluzione è quella del polling. Secondo questa tecnica il microprocessore , ad intervalli regolari, interrompe il programma che sta eseguendo e passa ad eseguire un'interrogazione delle periferiche che potrebbero richiedere un servizio. Se una periferica richiede un servizio il microprocessore passa a servirla altrimenti passa ad interrogare la successiva, al termine torna ad eseguire il programma.


LEGGI STATO PERIFERICO
RICHIESTA SERVIZIO?
si
no
SERVIZIO ALLA PERIFERICA

Questa tecnica presenta diversi svantaggi:

- *La stesura dei programmi diventa più complessa dovendo comprendere anche la gestione dell'interrogazione e del servizio alle periferiche*
- *Vi è un rilevante spreco di tempo poiché il microprocessore deve impiegare parte del suo tempo ad effettuare l'interrogazione delle periferiche anche se queste non hanno richiesto alcun servizio.*
- *Il microprocessore può giungere troppo tardi ad accorgersi della richiesta di servizio da parte di una periferica se questa è una delle ultime ad essere interrogata.*

Le interruzioni

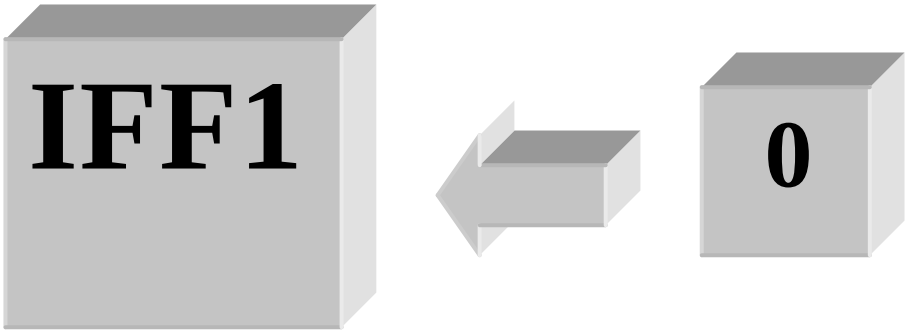
Con la tecnica delle interruzioni è la periferica che necessita di un servizio da parte del microprocessore, a prendere l'iniziativa e ad inviare allo Z80 un segnale di interruzione. Questo segnale costringe il microprocessore ad interrompere l'esecuzione del programma e a saltare ad una locazione di memoria da cui inizia un sottoprogramma che gestisce il servizio richiesto dalla periferica.

Interruzioni mascherabili e non mascherabili

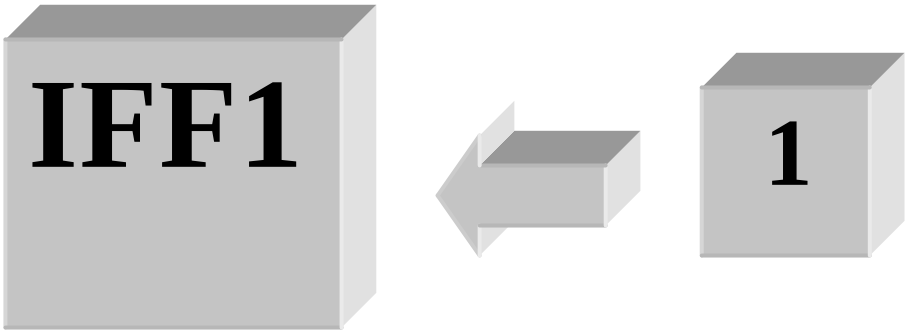
Le interruzioni si dividono in

- Mascherabili: sono interruzioni che il microprocessore può ignorare in particolari condizioni. Per lo Z80 è il programmatore che decide di porre il mi-

croprocessore in condizione da mascherare queste interruzioni. Per impedire che l'esecuzione di un segmento di programma venga interrotta da un'interruzione mascherabile, si fa eseguire al microprocessore l'istruzione DI (Disabile Interrupt). Questa istruzione setta il registro IFF1 a zero



Finche questo registro è settato a zero lo Z80 non prenderà in considerazione le interruzioni mascherabili. Per riabilitare le interruzioni s deve eseguire l'istruzione EI (Enable Interrupt). Quest'istruzione porta IFF1 a uno. Quando IFF1 è settato le interruzioni mascherabili possono interrompere il microprocessore



- Le interruzioni non mascherabili sono, invece, interruzioni che il microprocessore deve necessariamente prendere in considerazione. Nello Z80 i due

tipi di interruzione sono fisicamente separati poiché le interruzioni mascherabili vanno collegate al piedino INT\ e le non mascherabili sono invece collegate al piedino NMI\.

Gestione delle interruzioni

Per gestire un'interruzione occorre effettuare i seguenti passaggi:

- Interrompere il programma che si sta eseguendo, salvando lo stato del microprocessore in modo da poterne riprendere l'esecuzione al termine della gestione del servizio alla periferica che ha richiesto l'interruzione.
- Individuazione della periferica che ha generato l'interruzione ed esecuzione della routine che gestisce il servizio per quella particolare periferica

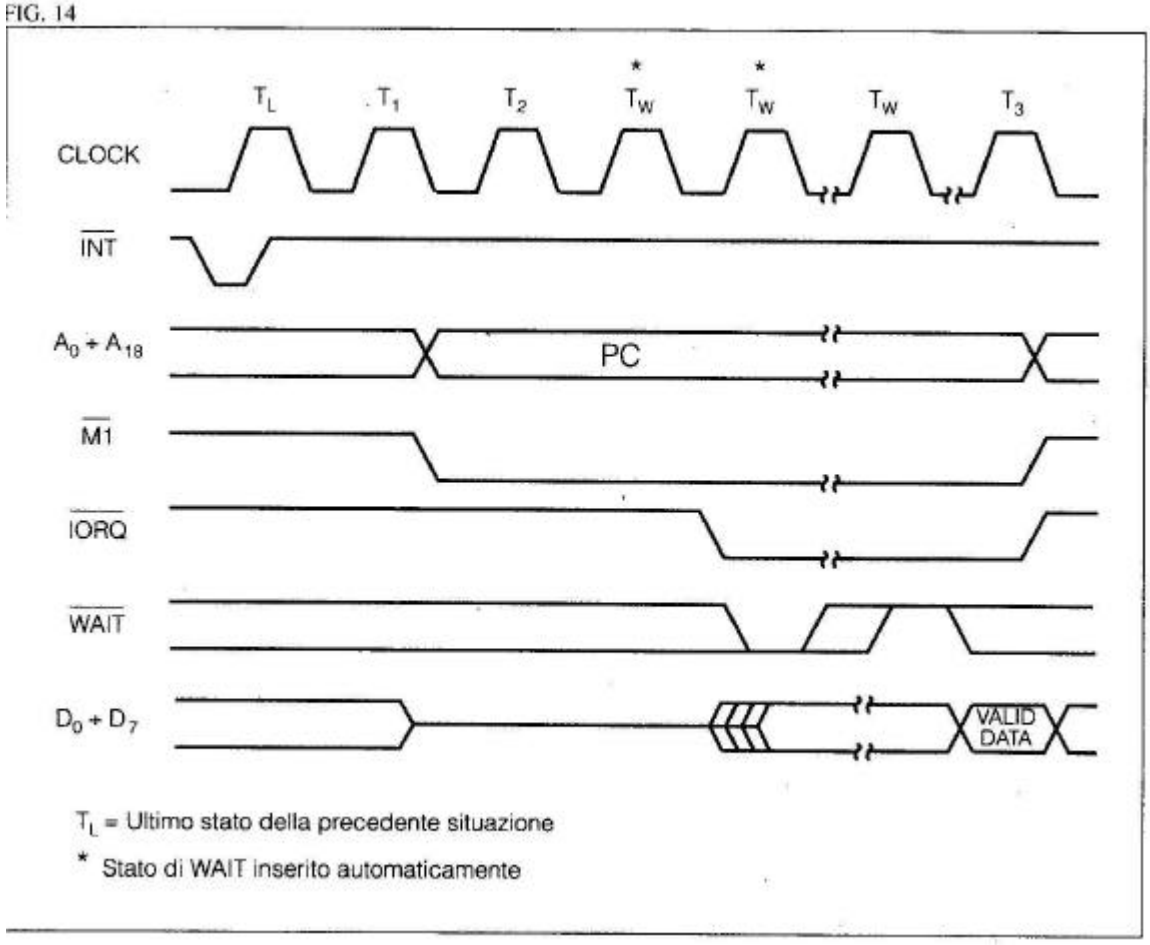
Interruzioni mascherabili

Il microprocessore rileva un'interruzione poiché va ciclicamente a testare la linea INT\ o NMI\. Questo test viene effettuato sull'ultimo clock di ogni istruzione.

In questo modo il microprocessore può essere interrotto soltanto tra un'istruzione e l'altra e non è costretto a lasciare a metà la singola istruzione.

Rilevata un'interruzione sulla linea INT\, lo Z80 testa il registro IFF1 e se questo si trova settato ad 1 (interruzioni abilitate), inizia la sua gestione. Occorre ora individuare qual è stata la causa di interruzione e saltare alla routine che la gestisce. In questo caso il microprocessore intraprende un ciclo detto *ciclo di riconoscimento delle interruzioni.*

FIG. 14



Questo ciclo è caratterizzato dal fatto che vengono attivate contemporaneamente le linee IORQ\ ed M1\. Terminato questo ciclo, lo Z80 procede a disabilitare ulteriori interruzioni. Ciò si rende necessario perché la periferica che ha generato l'interruzione potrebbe tardare nel riportare la linea INT a livello alto ed il microprocessore, testando di nuovo la linea INT, potrebbe credere erroneamente di trovarsi di fronte ad una nuova causa di interruzione. In tal caso interromperebbe la gestione dell'interruzione per passare a gestire l'interruzione stessa. E' quindi necessario che la routine di gestione dell'interruzione contenga un'istruzione EI per riabilitare le interruzioni che rimarrebbero altrimenti disabilitate.

Ora viene salvato lo stato del microprocessore.

Da rilevare che, in questa fase, viene salvato soltanto il contenuto del Program Counter (indirizzo di rientro del programma principale). Il contenuto dei registri interni dello Z80 deve essere salvato, invece, dalla routine che gestisce l'interruzione.

Da questo momento in poi, lo Z80 gestisce le interruzioni in tre modi diversi MODO 0, MODO 1, MODO 2. La modalità di gestione delle interruzioni è stabilità dal contenuto del registro IMF

IMF		
0	**0**	**Modo 0**
0	**1**	**Modo 1**
1	**0**	**Modo 2**
1	**1**	**Non consentito**

Il contenuto del registro viene settato da programma: mediante l'istruzione IM0 viene settato il modo 0, mediante l'istruzione IM1 viene settato il modo 1, mediante l'istruzione IM2 viene settato il modo 2.

Modo 1

E' la modalità di gestione più semplice. In questo caso il microprocessore ignora semplicemente il problema di individuare quale è stata la causa di interruzione e salta sempre ad eseguire il sottoprogramma che inizia alla locazione 0038h. E' evidente che, in questo caso, il microprocessore può gestire una sola causa d'interruzione nel

senso che, potendo saltare ad un solo sottoprogramma, può svolgere soltanto lo stesso tipo di servizio qualunque sia la causa d'interruzione.

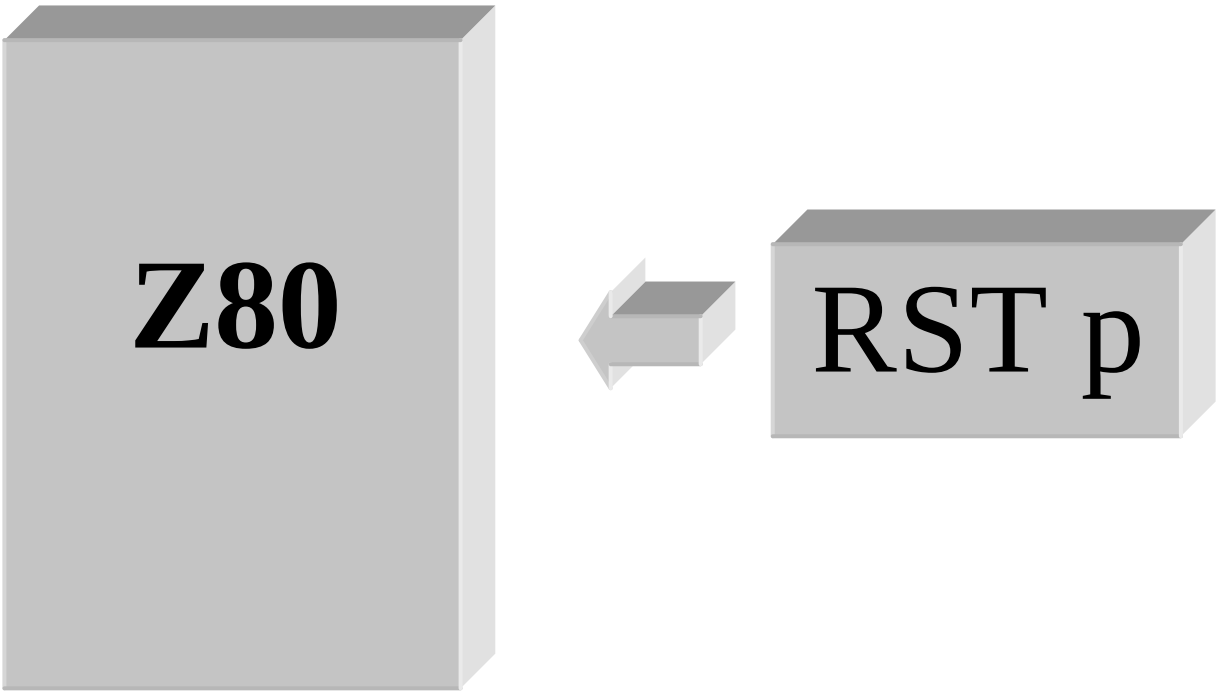
Modo 0

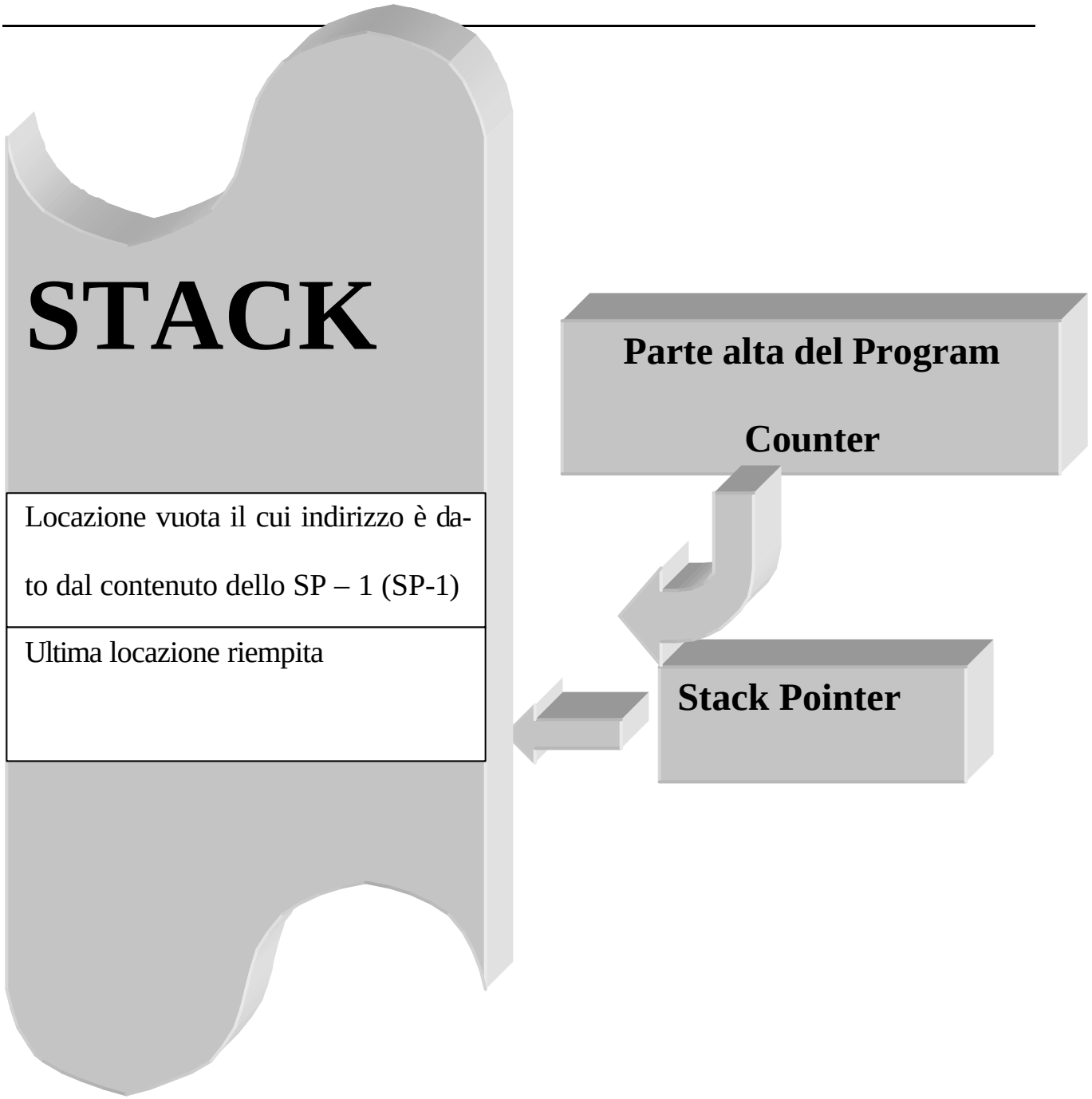
Le periferiche progettate per lavorare con uno Z80, sono in grado di leggere lo stato delle due linee M1 e IORQ. Quando la periferica vede queste due linee a zero nel corso del ciclo di riconoscimento delle interruzioni, essa scarica un byte sul bus dati. Questo byte è il codice operativo dell'istruzione RST. L'istruzione RST è un'istruzione codificata in un solo byte. In questo byte viene codificato anche un numero p a due bit che corrisponde ad un indirizzo a 16 bit secondo la seguente tabella

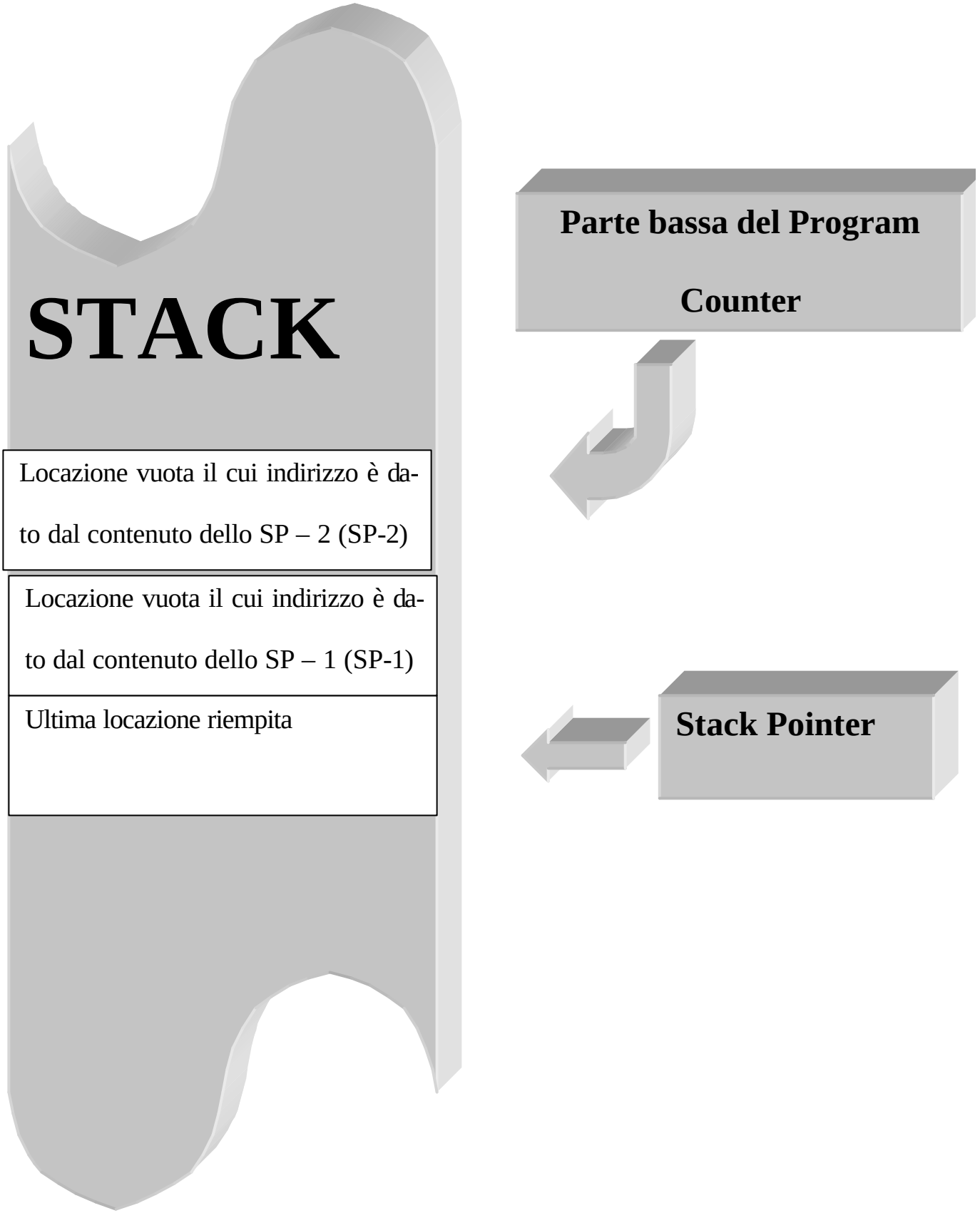
P	indirizzo
00	0000h
08	0008h
10	0010h
18	0018h
20	0020h
28	0028h
30	0030h
38	0038h

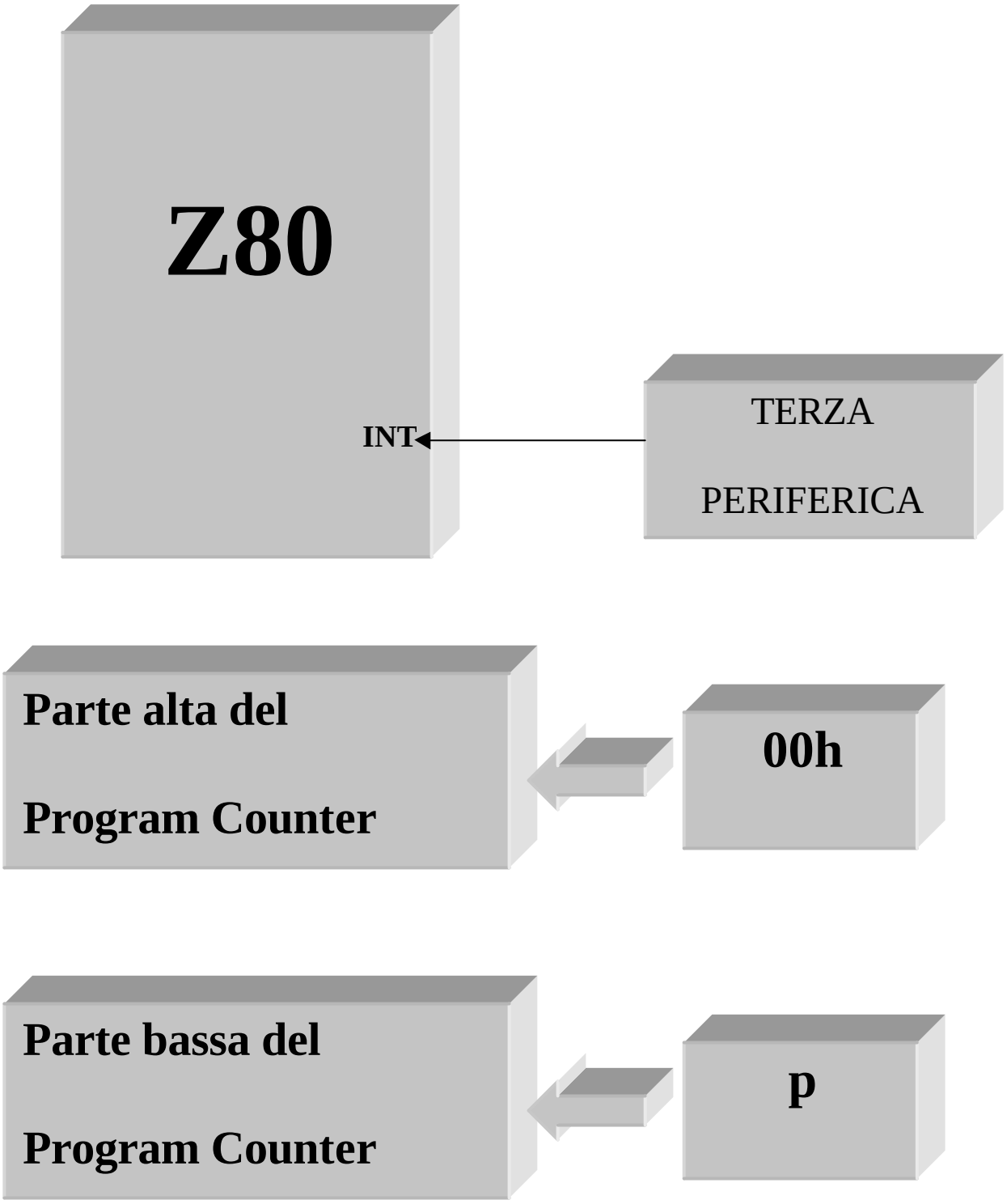
Questo tipo di indirizzamento viene detto *indirizzamento a pagina zero.*

Quando il microprocessore esegue questa istruzione, il Program Counter viene salvato nello stack e in esso viene inserito l'indirizzo codificato nell'istruzione.




STACK
Locazione vuota il cui indirizzo è dato dal contenuto dello SP – 1 (SP-1)
Ultima locazione riempita
Parte alta del Program Counter
Stack Pointer

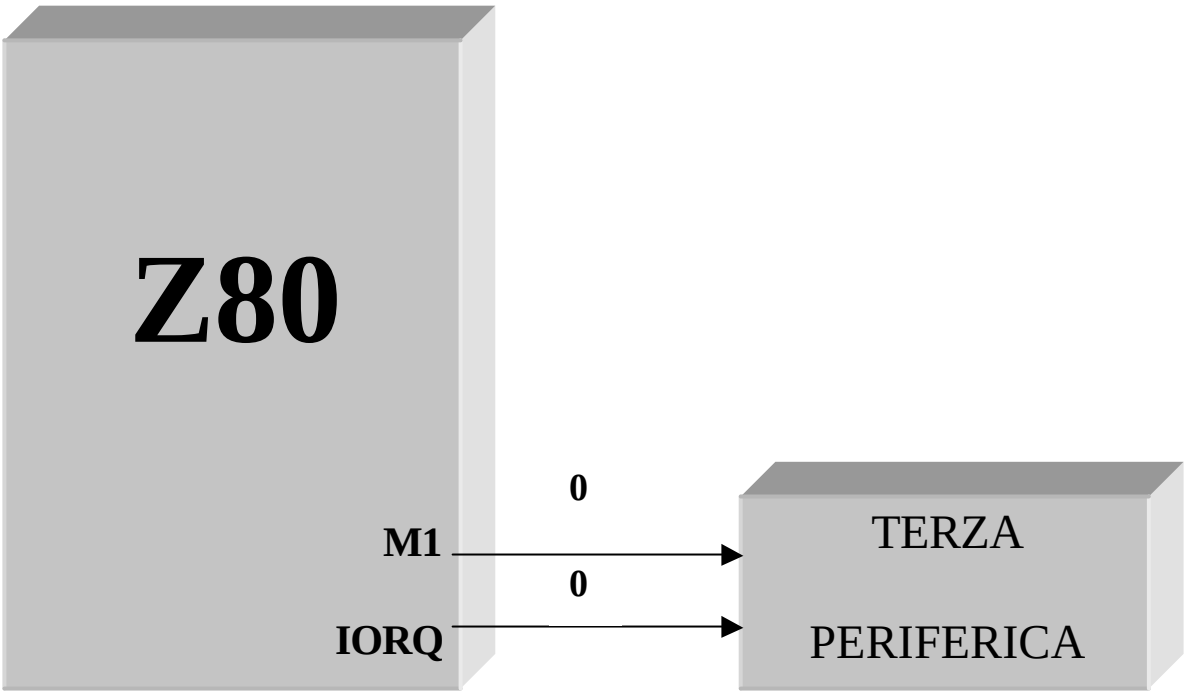

STACK
Locazione vuota il cui indirizzo è dato dal contenuto dello SP – 2 (SP-2)
Locazione vuota il cui indirizzo è dato dal contenuto dello SP – 1 (SP-1)
Ultima locazione riempita
Parte bassa del Program Counter
Stack Pointer

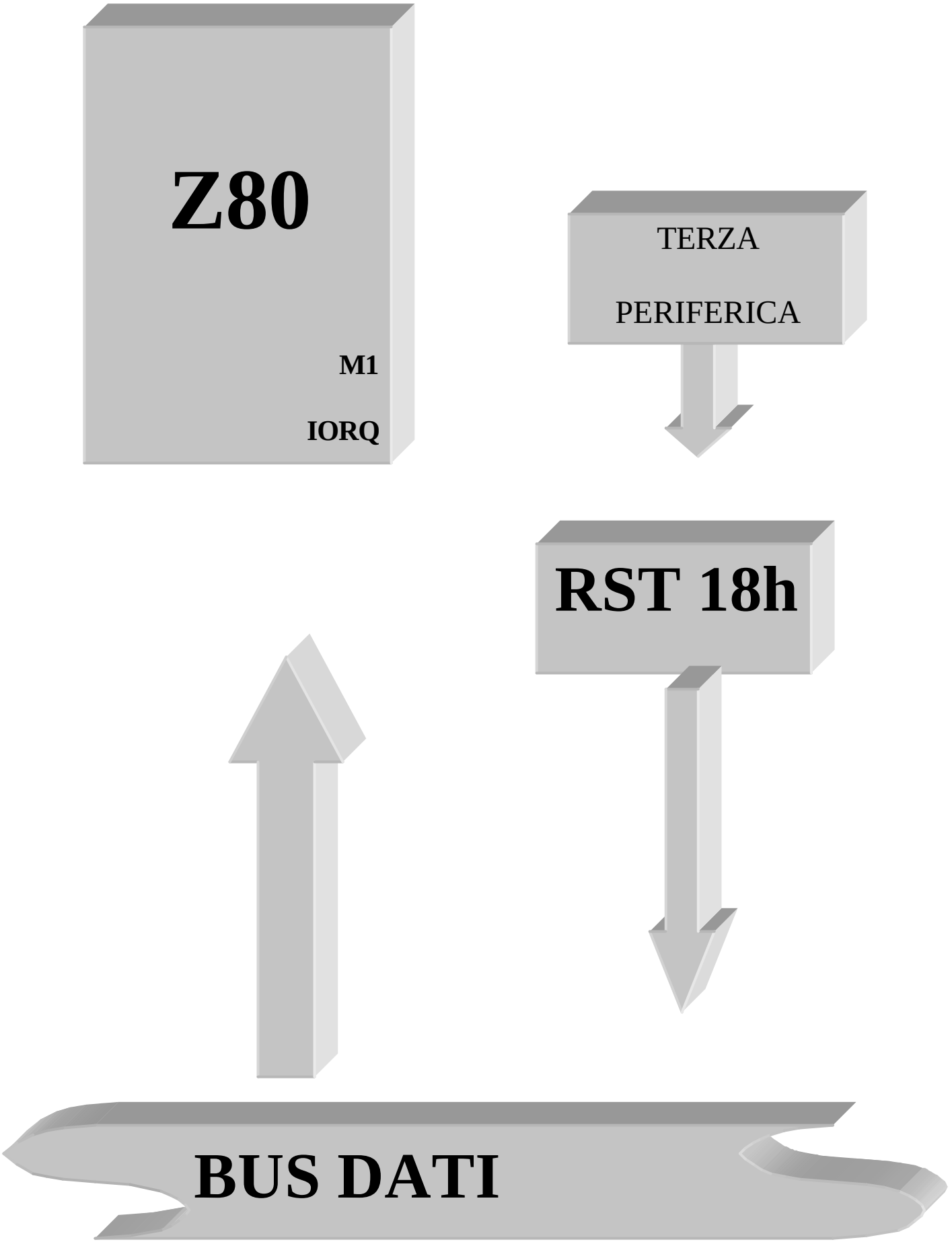


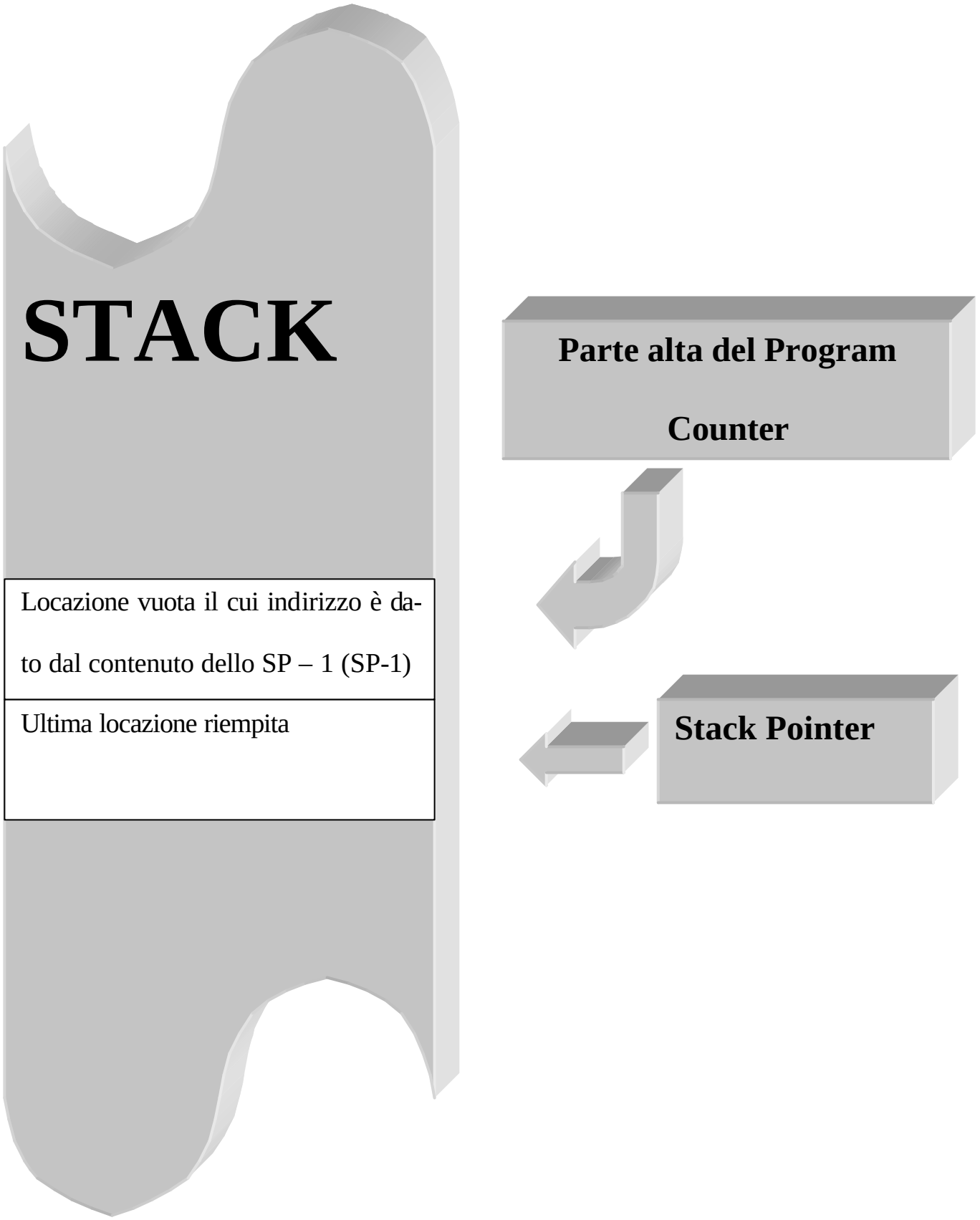
Con questo sistema lo Z80 può gestire otto cause di interruzione diversa. Ognuna delle otto periferiche avrà una routine di gestione che partirà da uno degli otto indirizzi

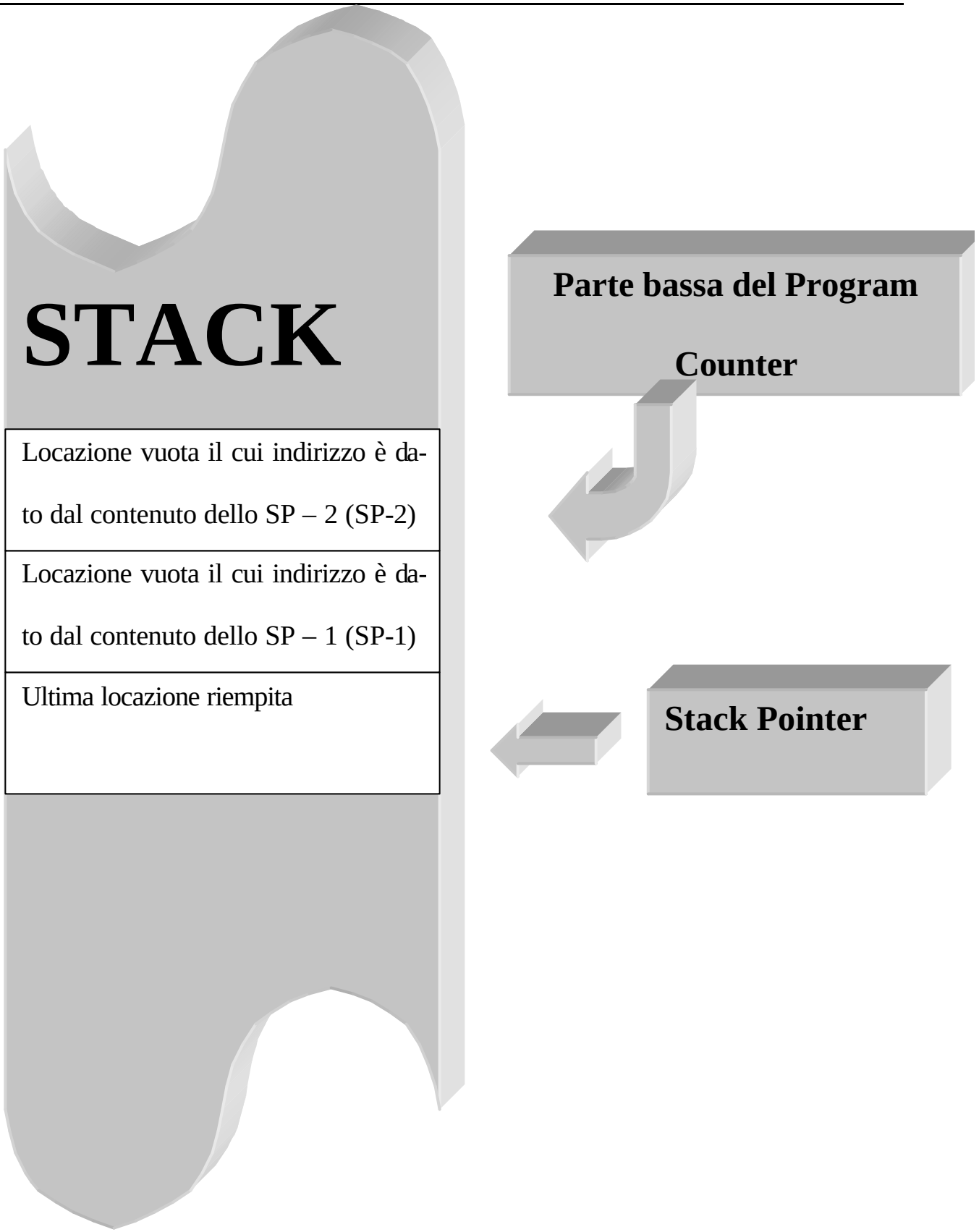
possibili elencati in tabella, oppure a questi indirizzi vi sarà un'istruzione di salto alla zona di memoria dove si trova la routine. La periferica deve inviare al microprocessore l'istruzione RST opportuna.

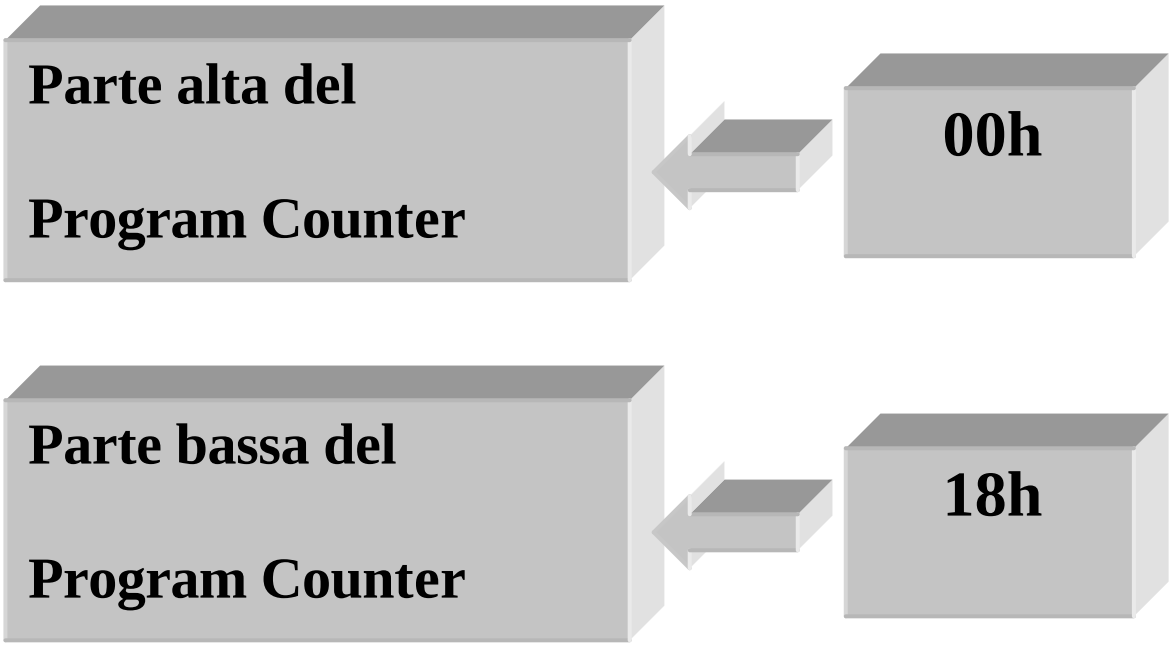
Supponiamo, ad esempio, che la terza periferica abbia la routine di gestione memorizzata a partire dall'indirizzo 1A00h. La periferica invia un segnale di interruzione al microprocessore, questo innesca il ciclo di riconoscimento delle interruzioni, la periferica dovrà reagire scaricando sul bus dati l'istruzione **RST 18.** A questo punto lo Z80 farà un salto alla locazione 0018h in cui sarà presente l'istruzione di salto all'indirizzo 1A00h

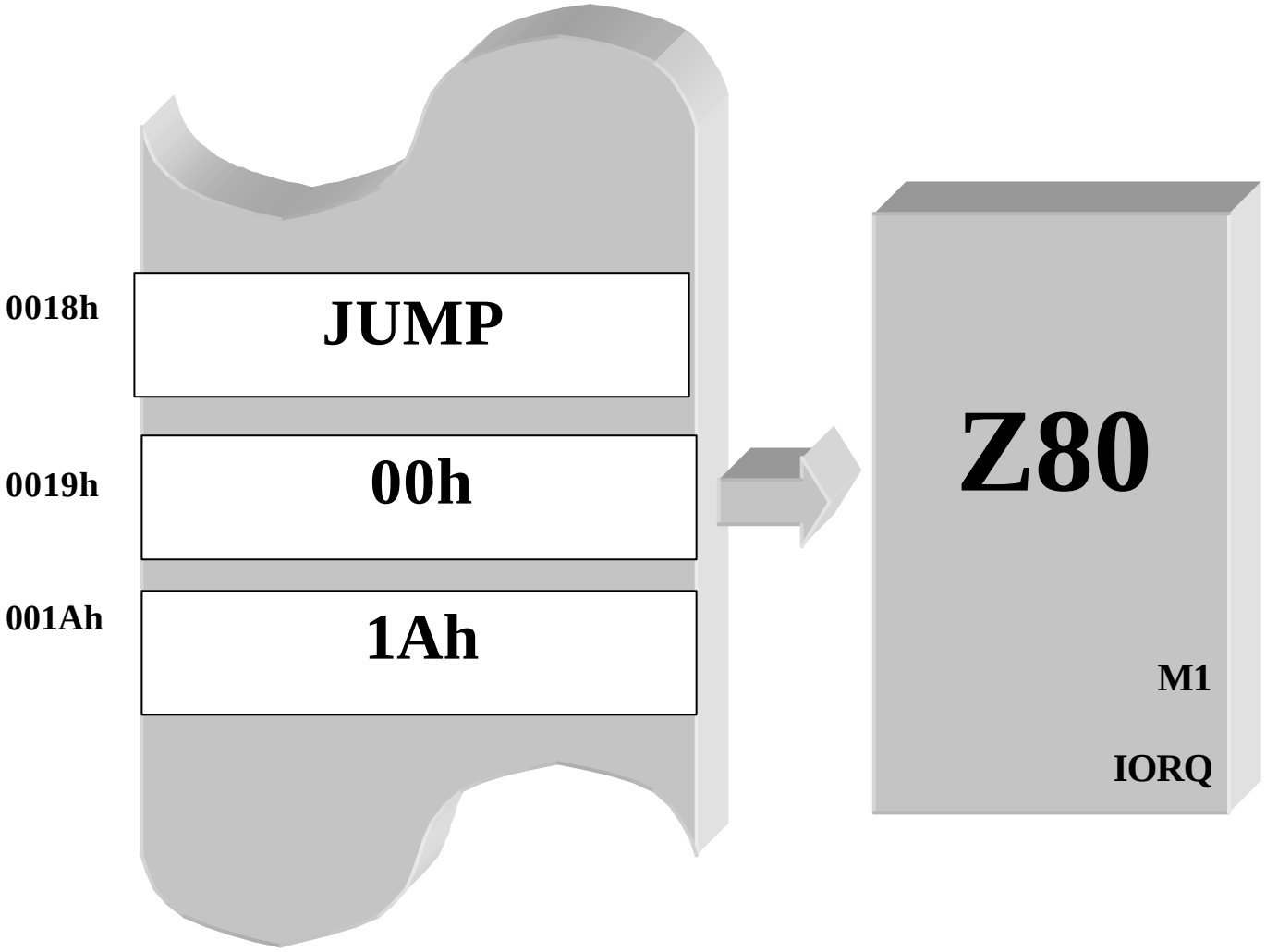


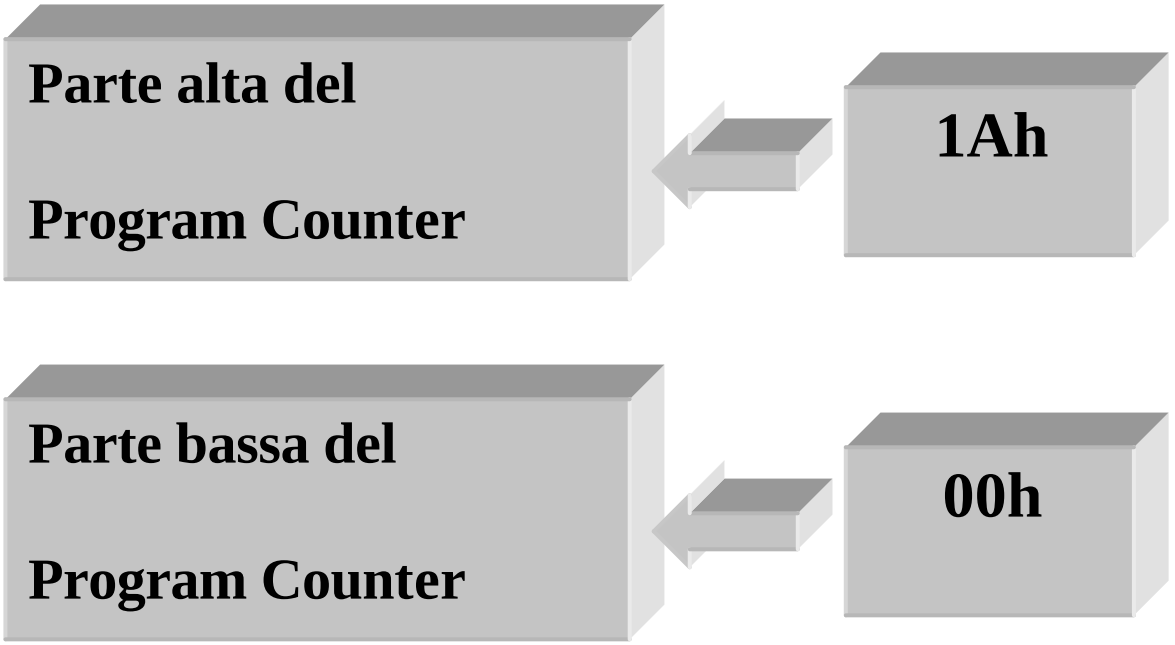

Z80
M1
IORQ
TERZA
PERIFERICA
RST 18h
BUS DATI


STACK
Locazione vuota il cui indirizzo è dato dal contenuto dello SP – 1 (SP-1)
Ultima locazione riempita
Parte alta del Program Counter
Stack Pointer


STACK
Locazione vuota il cui indirizzo è dato dal contenuto dello SP – 2 (SP-2)
Locazione vuota il cui indirizzo è dato dal contenuto dello SP – 1 (SP-1)
Ultima locazione riempita
Parte bassa del Program Counter
Stack Pointer


Parte alta del
Program Counter
00h
Parte bassa del
Program Counter
18h


0018h
JUMP
0019h
00h
001Ah
1Ah
Z80
M1
IORQ

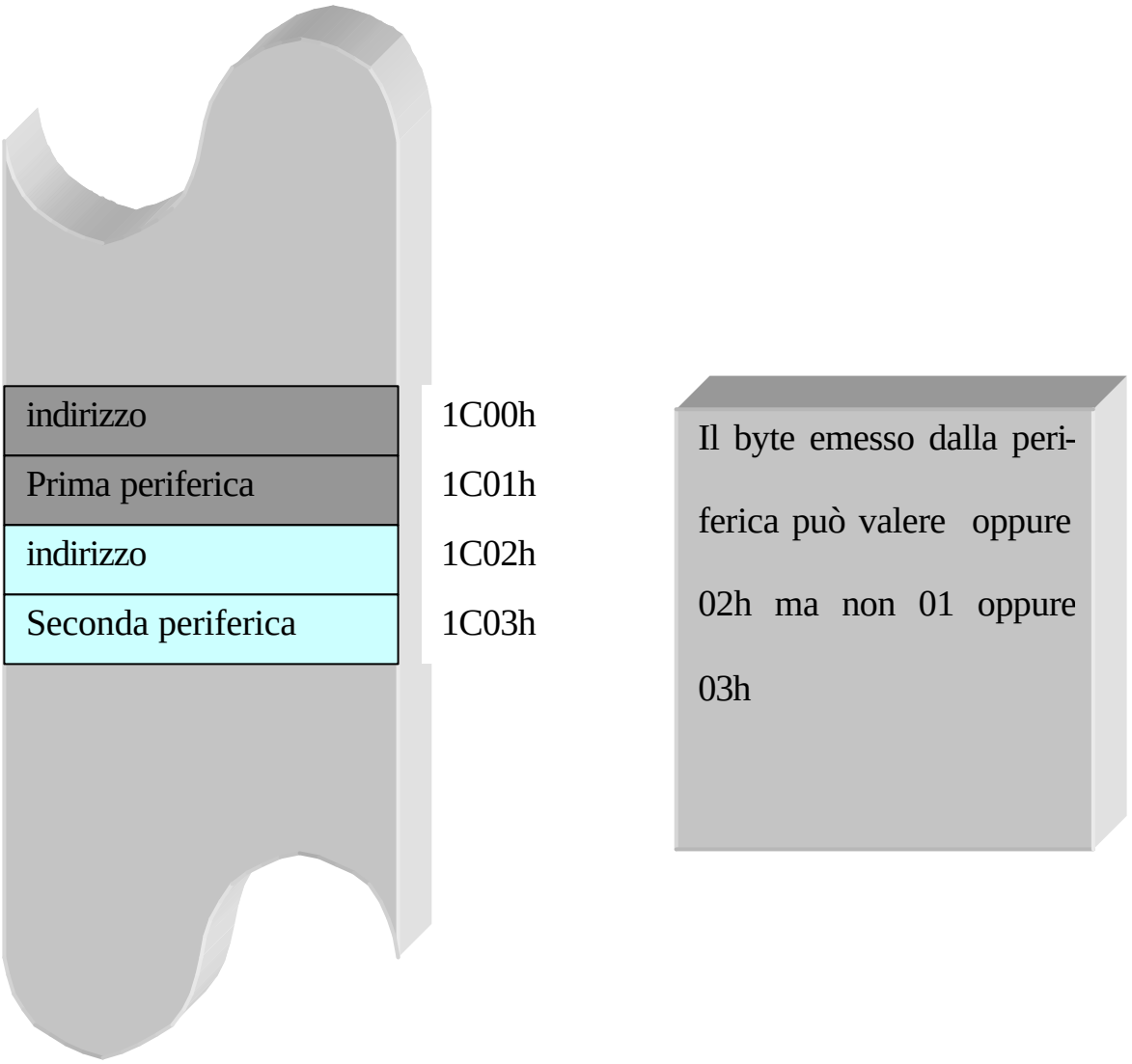


Modo 2

In questa modalità, detta anche delle *interruzioni vettorializzate,* quando la periferica vede le linee M1 e IORQ a zero, pone sul bus dati un byte. Stavolta non si tratta di un codice operativo ma della parte bassa di un indirizzo. Il microprocessore prende questo byte come parte bassa di un dato a 16 bit, la cui parte alta sarà il contenuto del registro I. Il dato a 16 bit così composto, costituisce un indirizzo di una zona di memoria particolare detta *Tabella dei Vettori delle Interruzioni.* Il microprocessore accede alla locazione individuata da questo indirizzo ed in essa e nella locazione successiva preleverà un altro dato a 16 bit : questo è l'indirizzo della routine che gestisce la periferica.

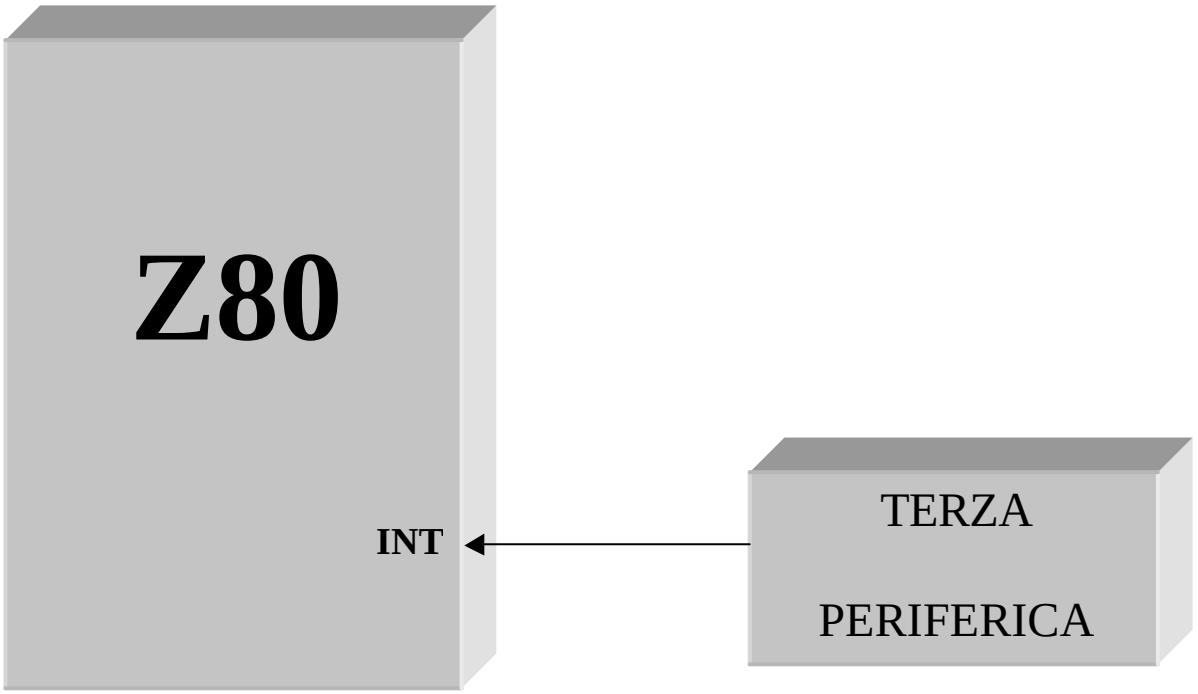
Poiché , nella tabella dei vettori delle interruzioni, occorrono due locazioni per contenere l'indirizzo di una routine di gestione delle interruzioni, accade che il byte emesso dalla periferica non possa assumere tutti i valori possibili ma soltanto valori pari.

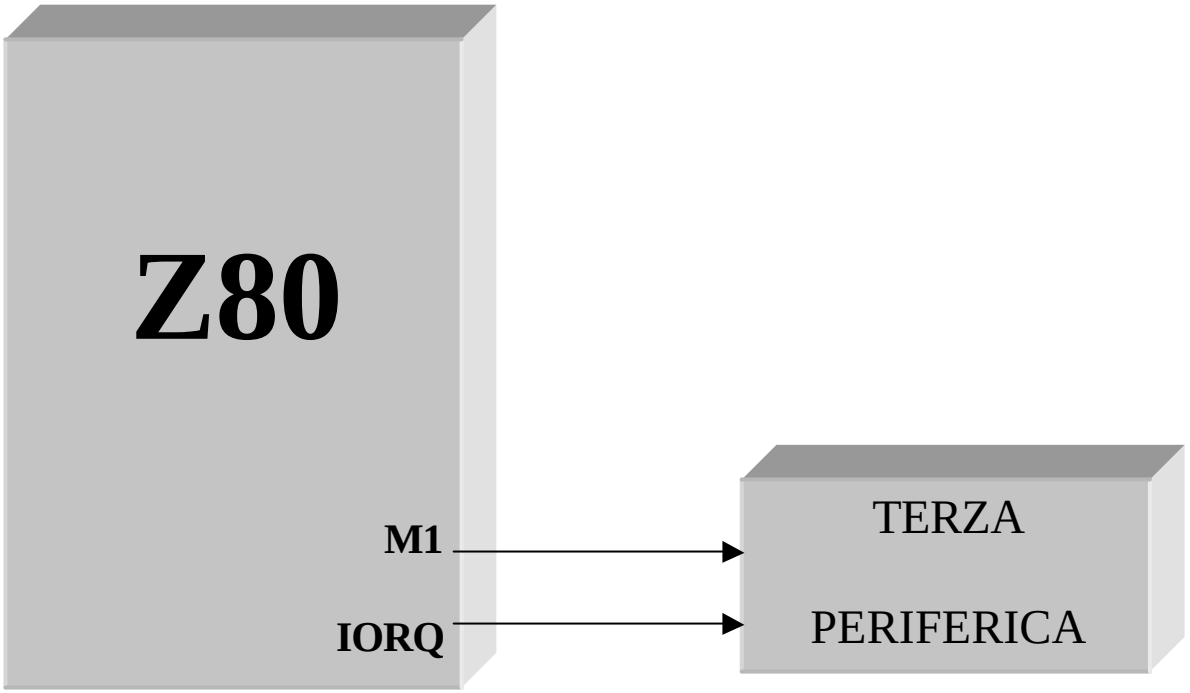
Esempio

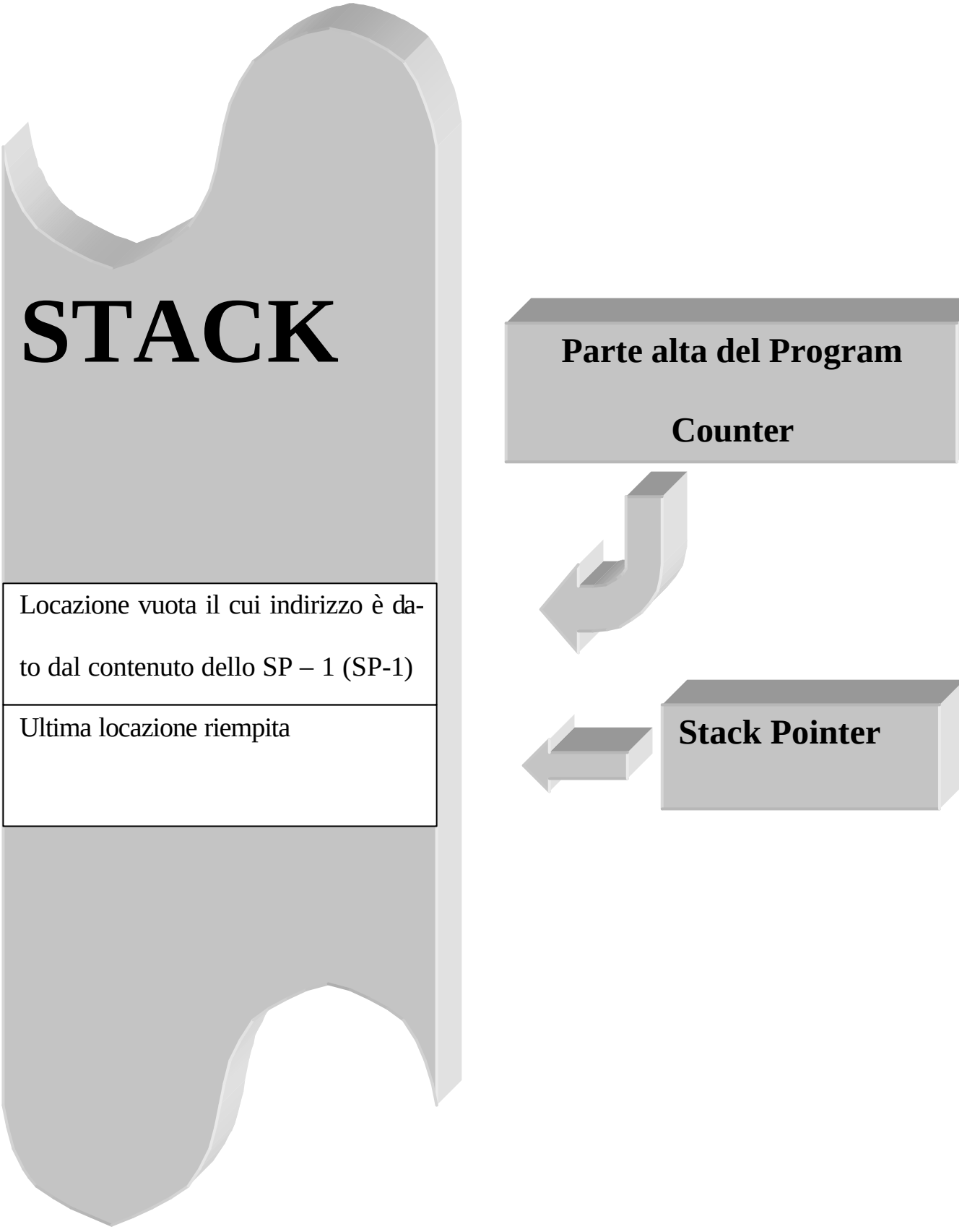


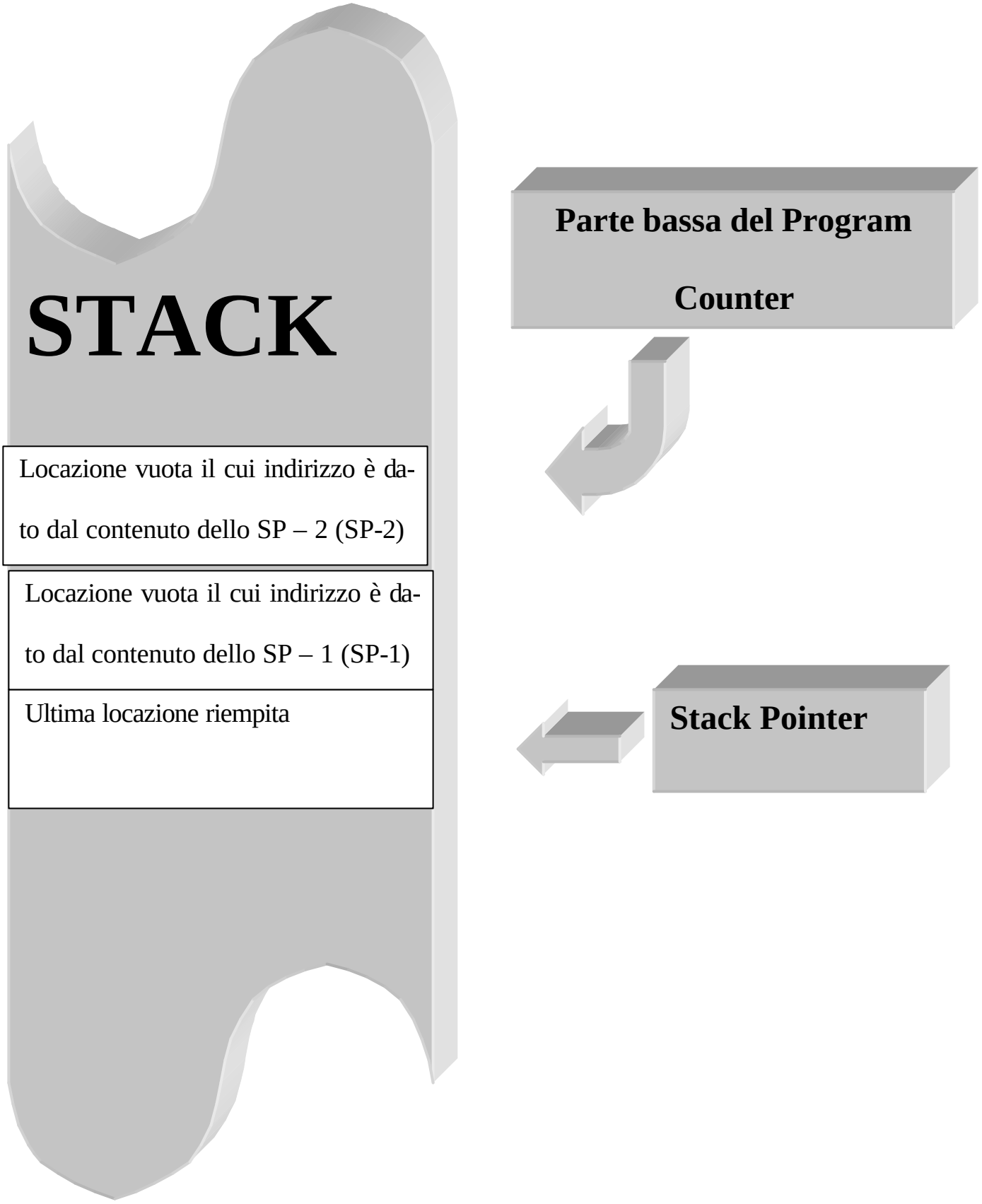
Questo significa, tenendo conto che il contenuto del registro I è fisso, che si possono formare, in questo modo, 256/2=128 indirizzi di versi di routine. Ciò significa che, nel modo 2, lo Z80 può gestire 128 cause di interruzione diverse

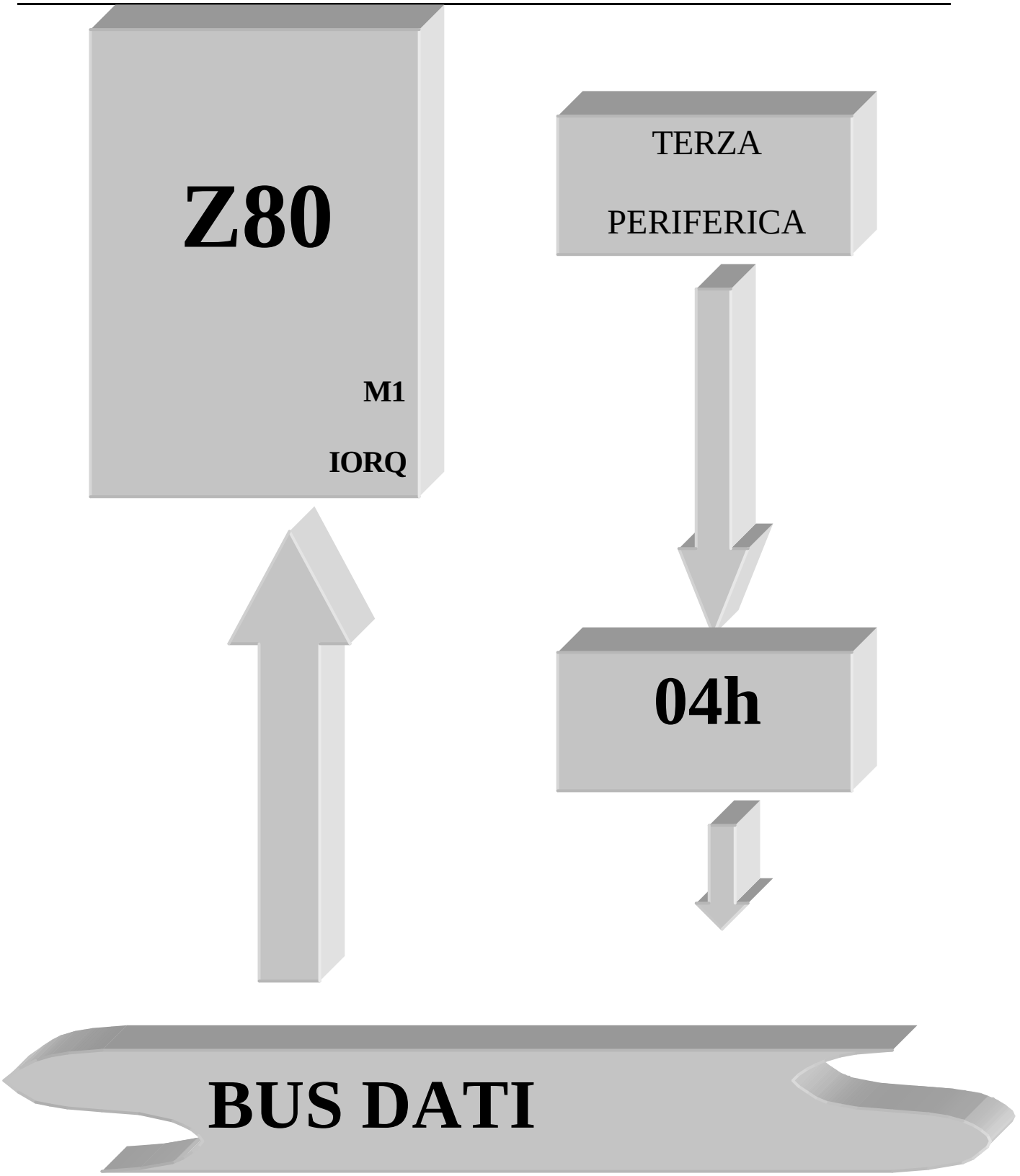
Esempio. Supponiamo che la terza periferica abbia la routine di servizio all'indirizzo 3C00h e che la tabella dei vettori parta dall'indirizzo 1A00h

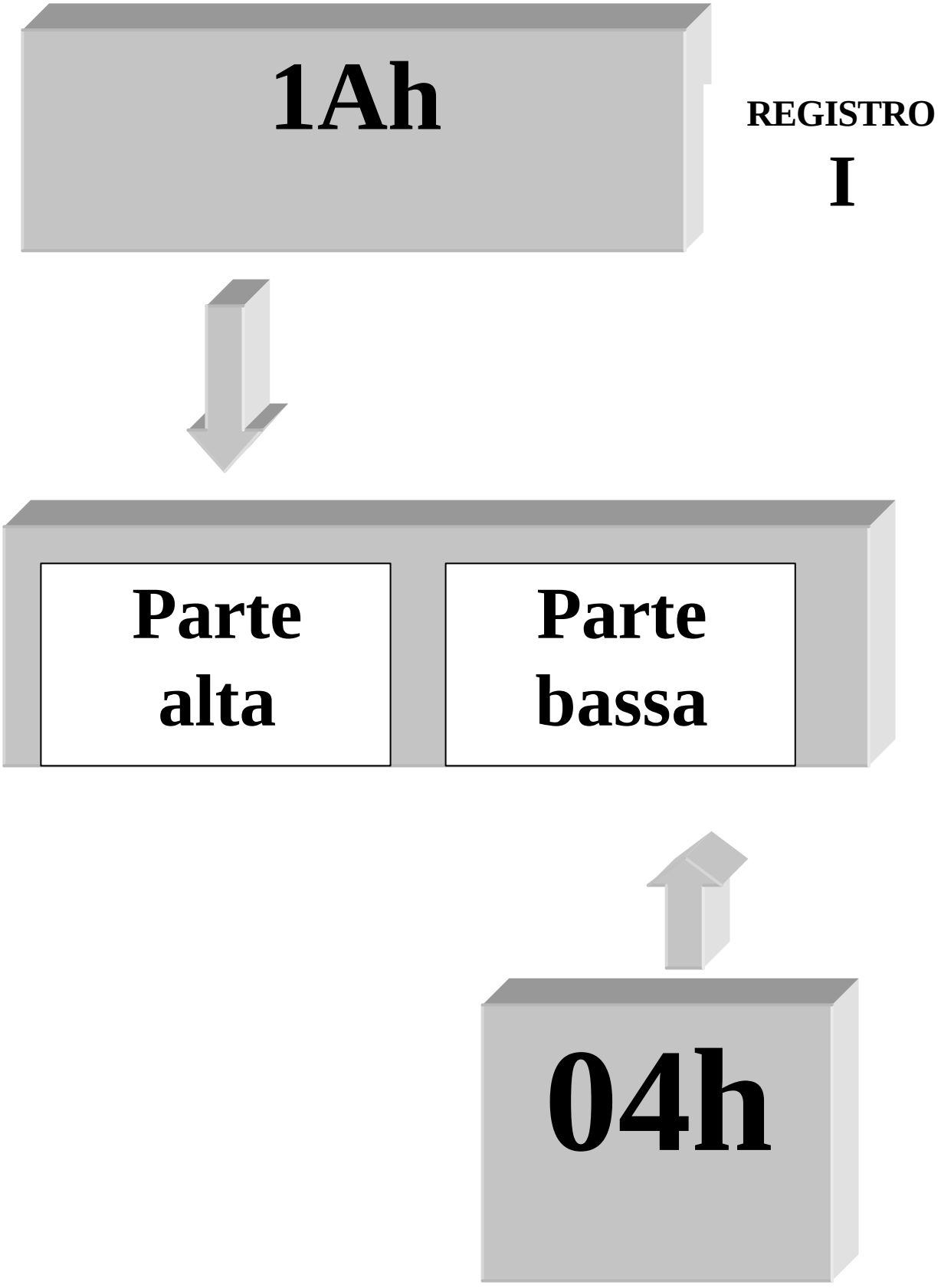

Z80
INT
TERZA
PERIFERICA


Z80
M1
IORQ
TERZA
PERIFERICA


STACK
Locazione vuota il cui indirizzo è dato dal contenuto dello SP – 1 (SP-1)
Ultima locazione riempita
Parte alta del Program Counter
Stack Pointer


STACK
Locazione vuota il cui indirizzo è dato dal contenuto dello SP – 2 (SP-2)
Locazione vuota il cui indirizzo è dato dal contenuto dello SP – 1 (SP-1)
Ultima locazione riempita
Parte bassa del Program Counter
Stack Pointer


Z80
M1
IORQ
TERZA
PERIFERICA
04h
BUS DATI


1Ah
REGISTRO
I
Parte alta
Parte bassa
04h

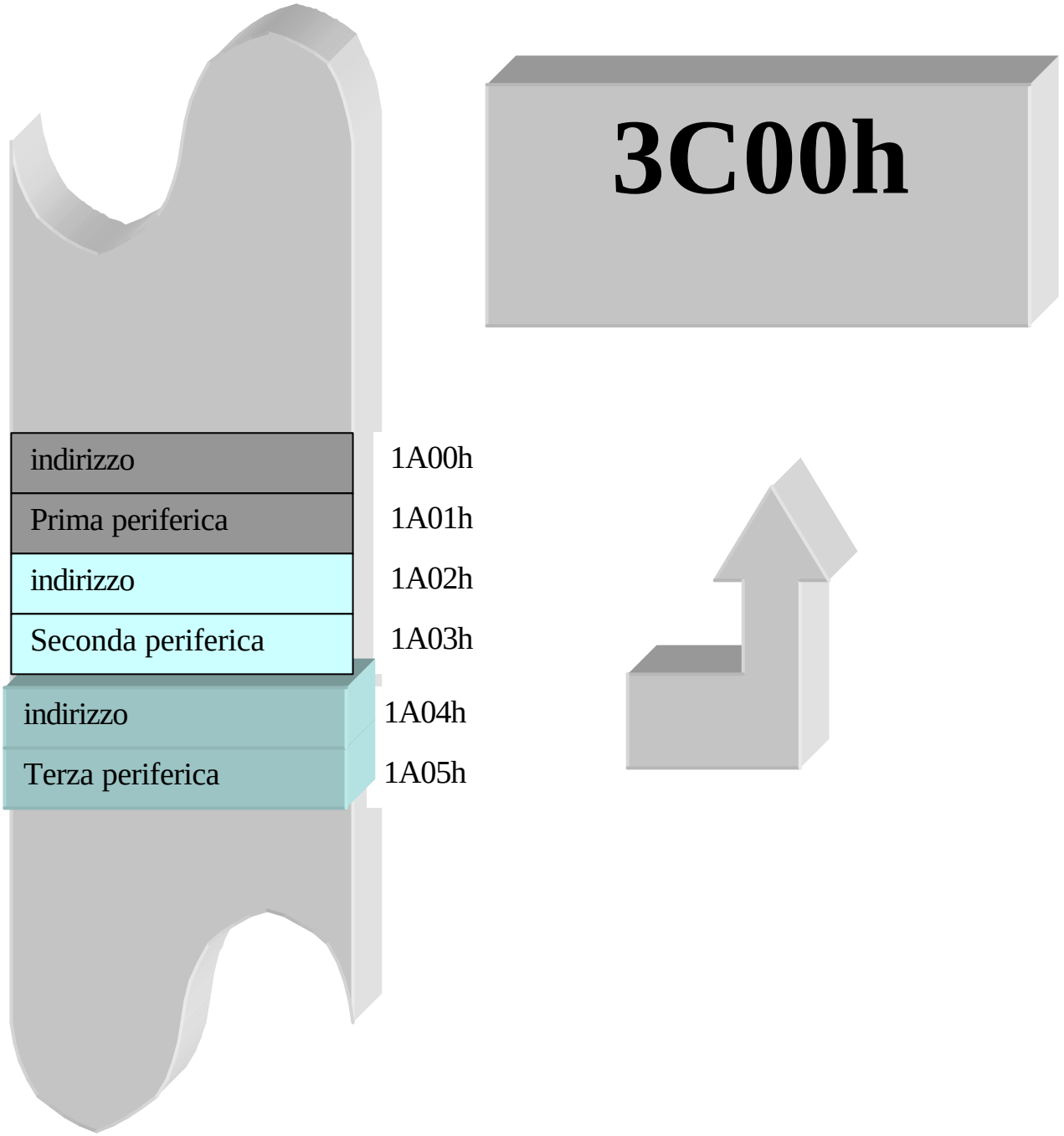
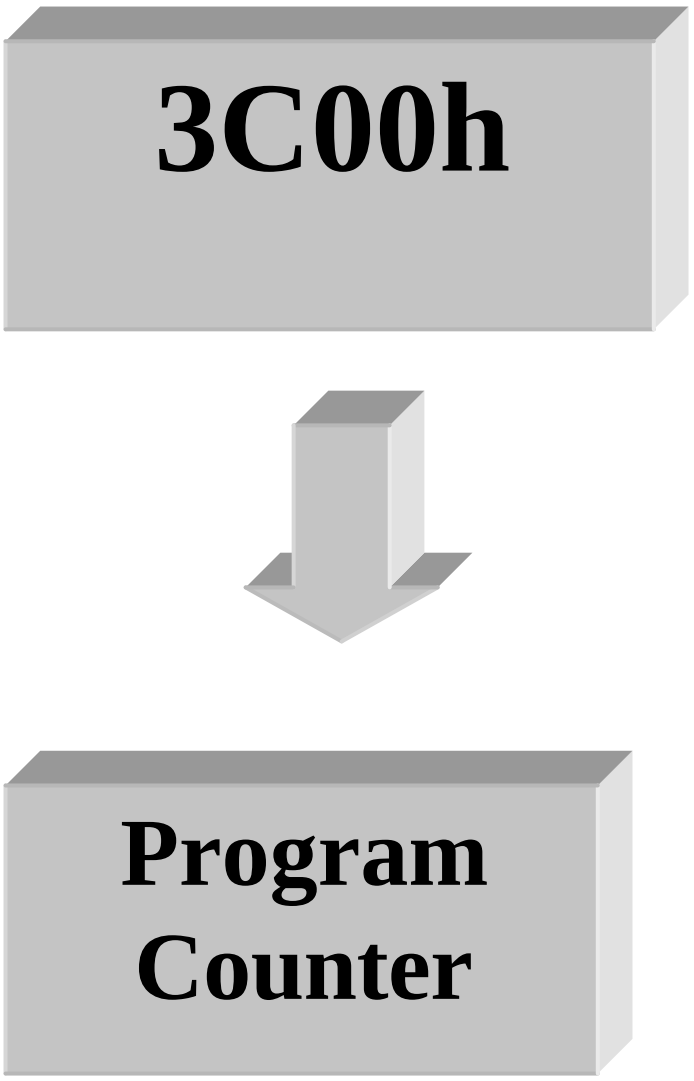

3C00h
indirizzo
Prima periferica
indirizzo
Seconda periferica
indirizzo
Terza periferica
1A00h
1A01h
1A02h
1A03h
1A04h
1A05h

TABELLA VETTORI DELLE INTERRUZIONI

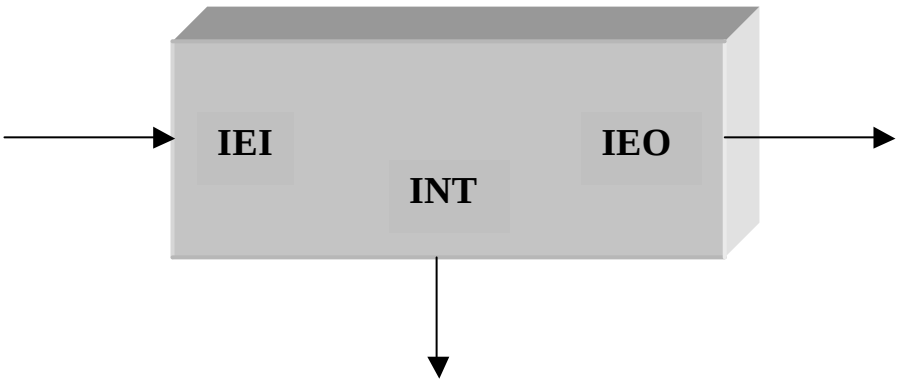


Priorità delle interruzioni

Le periferiche in grado di generare interruzioni possono avere diversi livelli di importanza. Ne discende che è preferibile che l'esecuzione della routine che gestisce la ri-

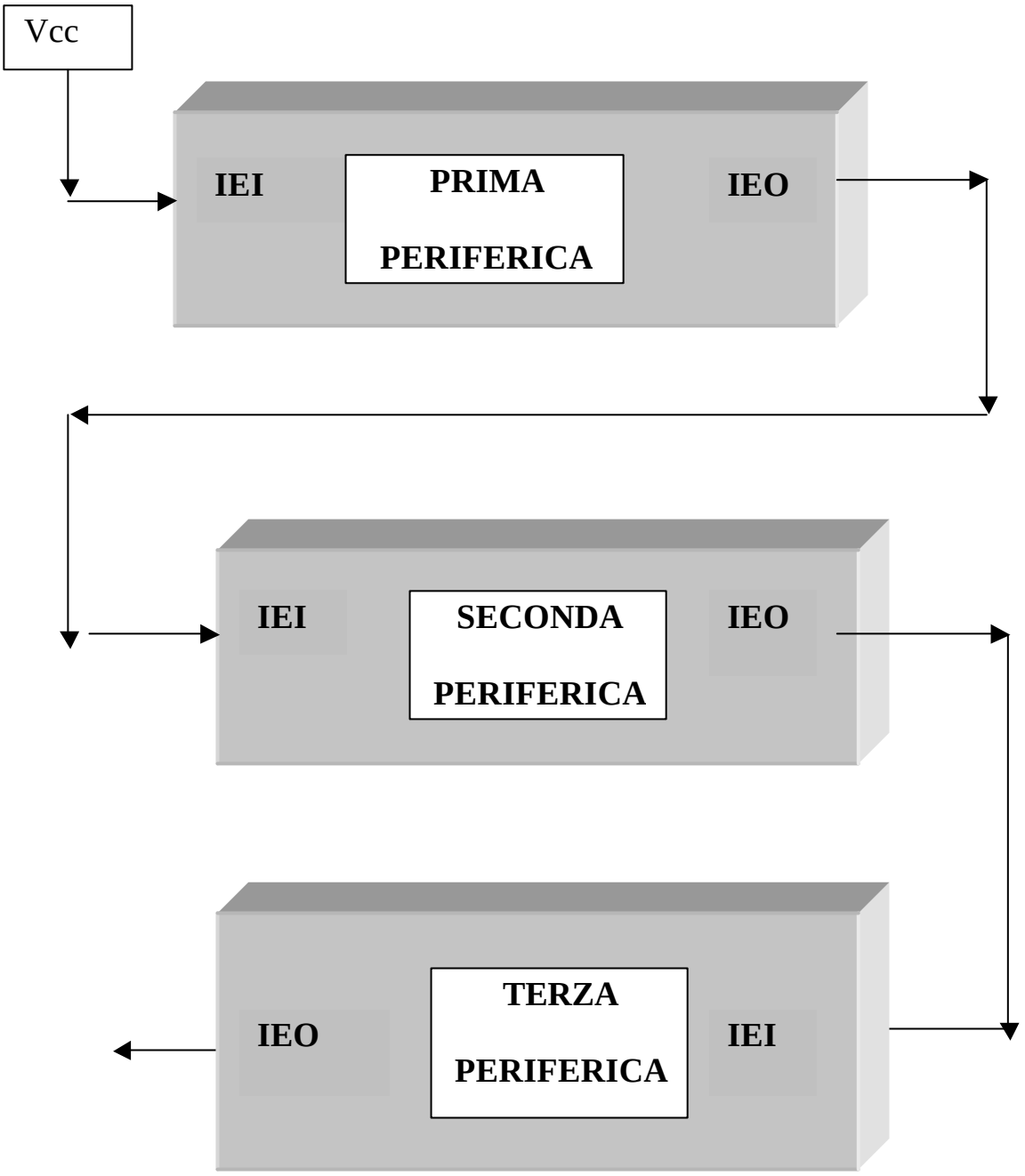
chiesta d'interruzione di una periferica non venga interrotta, a sua volta, dalla richiesta di interruzione proveniente da una periferica meno importante e possa, viceversa, essere interrotta da una causa di interruzione proveniente da una periferica più importante. Questo è il *problema delle priorità*. Per lo Z80 esso è risolto con una soluzione hardware chiamata *Daisy Chain o Ghirlanda di Margherite.* Ogni dispositivo progettato per lavorare con lo Z80 è dotato di due linee

- Un ingresso IEI (Interrupt Enable Input)
- Un'uscita IEO (Interrupt Enable Output)



Quando la periferica riceve un segnale attivo basso sull'ingresso IEI, essa non può più generare interruzioni. Quando essa viene inibita, invia a sua volta un segnale attivo basso sulla sua uscita IEO. Per risolvere il problema delle priorità, le periferiche vengono poste allora in catena, in modo che l'uscita IEO di una periferica sia l'ingresso IEI della successiva. Ogni periferica ha, dunque, una priorità inferiore a quella che la precede ma superiore a quella che la segue nella catena. La periferica di priorità più alta presenta il suo ingresso IEI fissato ad 1, poiché non vi è nessun'altra periferica che la possa inibire. Se la seconda periferica genera un'interruzione , contemporaneamente invia un segnale attivo basso all'ingresso della terza periferica .

Questa invia a sua volta un segnale basso all'ingresso IEI della quarta periferica e così via. In tal modo, tutte le periferiche che si trovano alla destra della seconda periferica



vengono inibite e non potranno generare interruzioni che blocchino la routine di gestione della seconda periferica. La prima periferica, invece, non avendo ricevuto alcun segnale d'inibizione potrà interrompere, a sua volta, la seconda periferica.

La catena del daisy chain non può avere una lunghezza infinita.

Se la catena fosse troppo lunga, tenendo presente che il segnale di inibizione si propaga in un tempo non nullo, potrebbe accadere che una delle ultime periferiche (proprio quelle a priorità più bassa), non avendo ancora ricevuto il segnale di inibizione, generi un'interruzione, riuscendo così ad interrompere una periferica a priorità elevata.

Interruzioni non mascherabili.

Le interruzioni non mascherabili non possono essere ignorate in alcun modo dal microprocessore. Quando arriva un segnale sulla linea NMI, lo Z80 salva il Program Counter ed il registro dei flag nello Stack, ed effettua un salto all locazione prefissata 0066h. Per evitare che la routine di gestione dell'interruzione non mascherabile venga interrotta da un'interruzione mascherabile, lo Z80 pone il registro IFF1 a zero.

Terminata la gestione dell'interruzione non mascherabile, andrebbe ripristinato lo stato delle interruzioni mascherabili (cioè IFF1 andrebbe lasciato a zero se le interruzioni erano già disabilitate oppure riportato ad 1 se le interruzioni mascherabili erano abilitate). Per questo motivo quando giunge un'interruzione non mascherabile, prima che il microprocessore ponga IFF1 a zero, il vecchio contenuto di IFF1 viene salvato nel registro IFF2. Al termine della routine in IFF1 verrà trasferito il contenuto di IFF2.