ISTRUZIONI DI SALTO

JP pq

E' un'istruzione di salto assoluto incondizionato. Per salto assoluto si intende il fatto che grazie a quest'istruzione, il contenuto del Program Counter viene sostituito completamente con il nuovo indirizzo pq espresso direttamente all'interno dell'istruzione. In tal modo si può saltare ad una qualunque locazione di memoria. Si dice incondizionato poiché questo salto viene eseguito senza dover sottostare a nessuna condizione. In sostanza il microprocessore è obbligato ad eseguire il salto quando incontra l'istruzione durante l'esecuzione del programma. L'istruzione non modifica il registro dei flag.

Es. JP 3A52

Prima

PC **1B00**

Dopo

PC **3A52**

JP cc, pq

E' un'istruzione di salto assoluto condizionato. Quest'istruzione ha un effetto identico a quello dell'istruzione precedente ma a differenza di essa viene eseguita soltanto se la condizione cc espressa nell'istruzione risulta essere vera. Le condizioni sono espresse nella seguente tabella

JP NC,pq	Esegui il salto se il flag di carry si trova a zero
JP C, pq	Esegui il salto se il flag di carry si trova ad uno
JP NZ, pq	Esegui il salto se il flag di zero si trova a zero
JP Z, pq	Esegui il salto se il flag di zero si trova ad uno
JP PO, pq	Esegui il salto se il flag di parità/overflow si trova a zero
JP PE, pq	Esegui il salto se il flag di parità/overflow si trova ad uno
JP P, pq	Esegui il salto se il flag di segno S si trova zero
JP M, pq	Esegui il salto se il flag di segno S si trova ad uno

JP (HL)

E' un'istruzione di salto assoluto incondizionato. Per salto assoluto si intende il fatto che grazie a quest'istruzione, il contenuto del Program Counter viene sostituito completamente con il nuovo indirizzo che è costituito dal contenuto del registro HL. In tal modo si può saltare ad una qualunque locazione di memoria. Si dice incondizionato poiché questo salto viene eseguito senza dover sottostare a nessuna condizione. In sostanza il microprocessore è obbligato ad eseguire il salto quando incontra l'istruzione durante l'esecuzione del programma. L'istruzione non modifica il registro dei flag.

Es. JP (HL)

Prima

PC	**1B00**	HL	**0411**

Dopo

PC	**0411**

JP (IX)

JP (IY)

Sono del tutto equivalenti all'istruzione JP (HL). In esse i registri IX ed IY svolgono la funzione di HL.

JR e

E' un'istruzione di salto relativo. Si differenzia da quella di salto assoluto perché, in questo caso, il contenuto del Program Counter non viene completamente alterato. Il valore e, detto offset, viene sommato al vecchio contenuto del Program Counter. Tenendo presente che l'offset è un dato ad otto bit con segno ciò significa che, con un salto relativo, non possiamo spostarci a piacere nell'intera area di memoria ma, a

partire dalla posizione attuale del PC, possiamo andare in avanti di +127 locazioni o andare indietro di −128 locazioni.

ESEMPIO

JR FD

Prima

PC **1B00** Dopo

Teniamo presente che $FD|_{16}=11111101|_2=-3$ (rappresentazione complemento a due)

PC **1AFD**

JR cc, e

E' un'istruzione di salto relativo come la precedente, ma è condizionata. Il salto verrà eseguito soltanto se la condizione cc è verificata. Le condizioni sono espresse nella seguente tabella

JR NC,pq	Esegui il salto se il flag di carry si trova a zero
JR C, pq	Esegui il salto se il flag di carry si trova ad uno
JR NZ, pq	Esegui il salto se il flag di zero si trova a zero
JR Z, pq	Esegui il salto se il flag di zero si trova ad uno

DJNZ e

Questa è un'istruzione di salto relativo condizionato un po' speciale. Essa presuppone che voi stiate effettuando un ciclo in cui utilizziate il registro B come contatore. Quest'istruzione allora decrementa B ed effettua un salto se il regiastro non è ancora diventato nullo. Se dopo l'operazione di decremento il contenuto del registro B è diventato nullo, l'istruzione di salto non viene più eseguita. Il salto viene eseguito sommando al contenuto del PC, l'offset e..

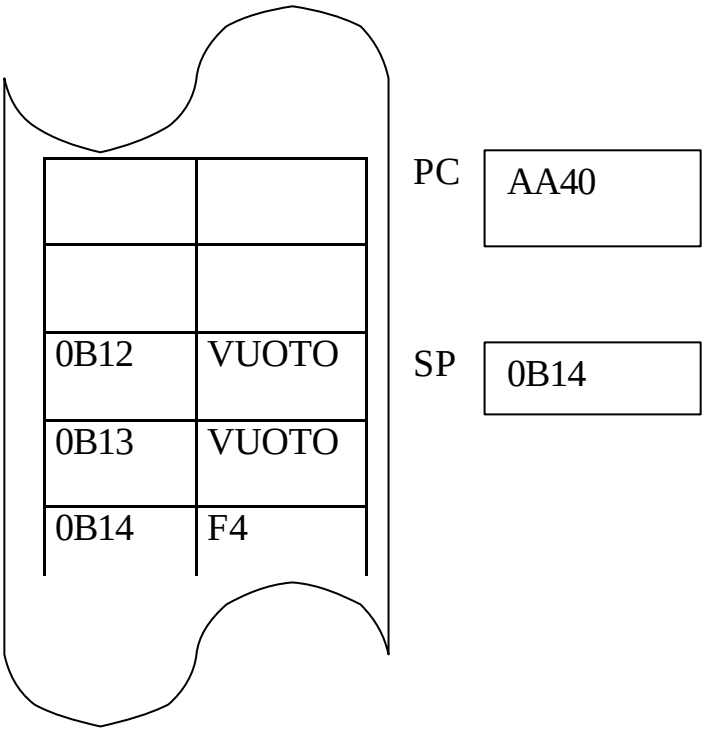
ISTRUZIONI DI CHIAMATA DI SOTTOPROGRAMMI E RITORNO DA SOTTOPROGRAMMA

CALL nn

Quest'istruzione effettua la chiamata al sottoprogramma che si trova all'indirizzo nn. Prima di effettuare la chiamata il contenuto attuale del Program Counter viene salvato nello stack successivamente nel Program Counter viene inserito l'indirizzo nn da cui parte il sottoprogramma.

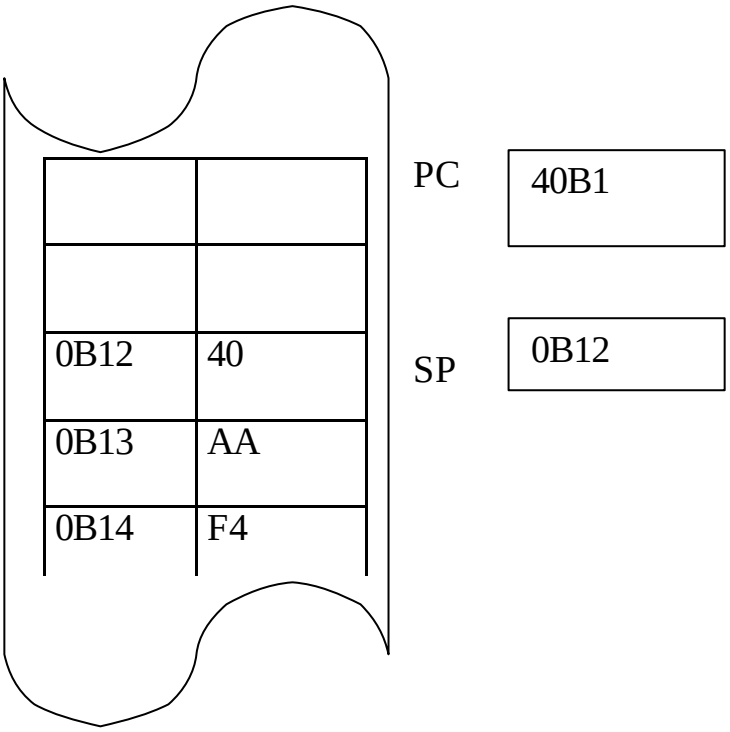
ESEMPIO. CALL 40B1

Prima



STACK

Dopo



STACK

CALL cc, nn

L'istruzione svolge le stesse funzioni di quella precedente. L'unica differenza consiste nel fatto che essa è condizionata. La chiamata del sottoprogramma, cioè, verrà effettuata soltanto se la condizione cc è vera. Le condizioni sono espresse nella seguente tabella

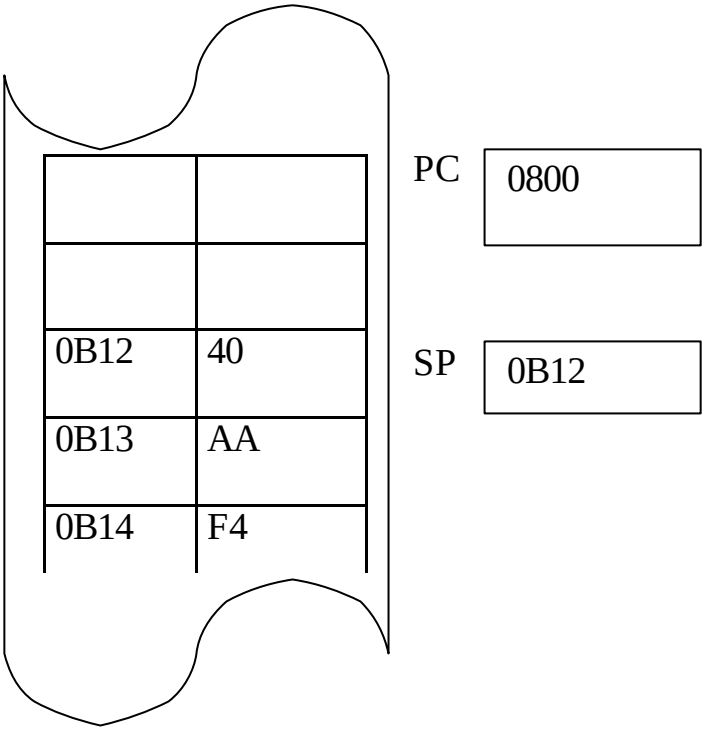
CALL NC,pq	Esegui se il flag di carry si trova a zero
CALL C, pq	Esegui se il flag di carry si trova a uno
CALL NZ, pq	Esegui se il flag di zero si trova a zero
CALL Z, pq	Esegui se il flag di zero si trova ad uno
CALL PO, pq	Esegui se il flag di parità/overflow si trova a zero
CALL PE, pq	Esegui se il flag di parità/overflow si trova a uno
CALL P, pq	Esegui se il flag di segno S si trova zero
CALL M, pq	Esegui se il flag di segno S si trova ad uno

RET

Quest'istruzione termina un sottoprogramma e riporta il controllo al programma che aveva chiamato il sottoprogramma. In sostanza, quando esegue quest'istruzione, il microprocessore accede allo stack. La locazione puntata dallo Stack Pointer contiene la parte bassa del Program Counter, la locazione immediatamente precedente contiene la parte alta. Poiché quest'istruzione corrisponde al prelievo di due byte dallo stack, questo decresce di due locazioni per cui lo Stack Pointer, che deve puntare sempre alla cima dello Stack verrà automaticamente incrementato di due locazioni.

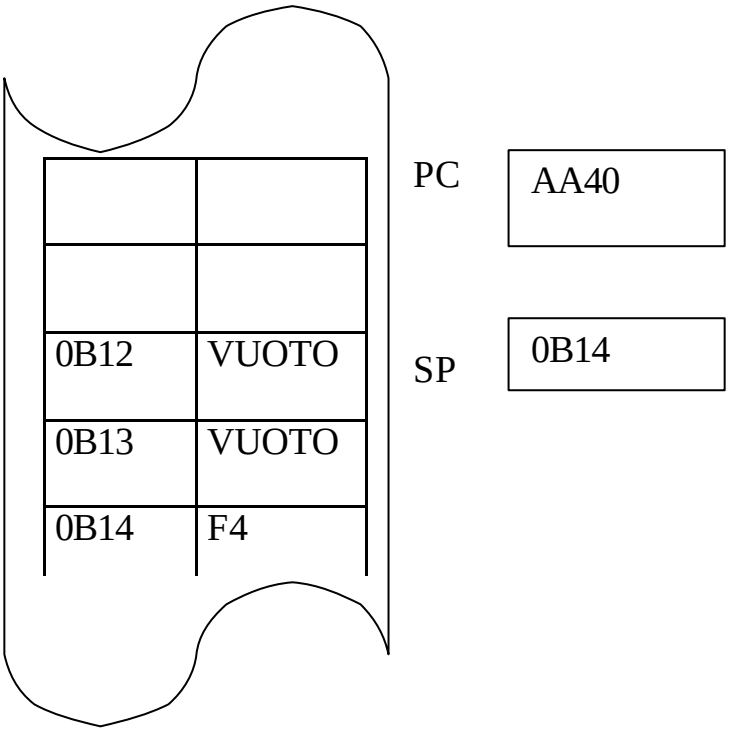
ESEMPIO. RET

Prima



STACK

Dopo



STACK

RET cc

L'istruzione è del tutto identica alla precedente. L'unica differenza è che l'esecuzione di quest'istruzione è condizionata. Il ritorno al programma principale avverrà soltanto se la condizione espressa nell'istruzione è vera. Le condizioni sono espresse nella seguente tabella

RET NC,pq	Esegui se il flag di carry si trova a zero
RET C, pq	Esegui se il flag di carry si trova a uno
RET NZ, pq	Esegui se il flag di zero si trova a zero
RET Z, pq	Esegui se il flag di zero si trova ad uno
RET PO, pq	Esegui se il flag di parità/overflow si trova a zero
RET PE, pq	Esegui se il flag di parità/overflow si trova a uno
RET P, pq	Esegui se il flag di segno S si trova zero
RET M, pq	Esegui se il flag di segno S si trova ad uno

RETI

Quest'istruzione ha lo stesso effetto di RET. Essa viene usata per porre termine ai sottoprogrammi che gestiscono interruzioni di periferiche. Spieghiamo perché le routine di gestione delle interruzioni necessitano di un'istruzione speciale di ritorno. La questione si ricollega alla gestione delle priorità delle interruzioni attraverso il daisy chain. Ricordiamo che una periferica che genera un'interruzione, inibisce tutte le periferiche che si trovano alla sua destra tramite la sua linea IEO. Essa tiene bloccate le periferiche che hanno priorità inferiore fino a che la sua routine di servizio non termini. La periferica deve allora sapere in qualche modo quando la routine termina. Ciò avviene quando essa vede transitare sul bus dati, dalla memoria al microprocessore, il codice operativo corrispondente a RETI. Ecco il motivo per cui le routine di gestione delle interruzioni devono terminare con un'istruzione speciale.

RETN.

Quest'istruzione termina la routine che gestisce le interruzioni non mascherabili. In cosa si differenzia dalle altre istruzioni di ritorno? Occorre ricordare che un'interruzione non mascherabile, onde evitare che la routine che la gestisce venga interrotta da un'interruzione mascherabile, pone a zero il flip flop IFF1. Il vecchio contenuto di IFF1 è riposto in IFF2. Al termine della gestione dell'interruzione non mascherabile, il vecchio contenuto di IFF1, memorizzato temporaneamente in IFF2, deve essere ripristinato. Quest'operazione è compiuta dall'istruzione RETN. Per il resto, RETN svolge operazioni del tutto identiche a RET e RETI.

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.