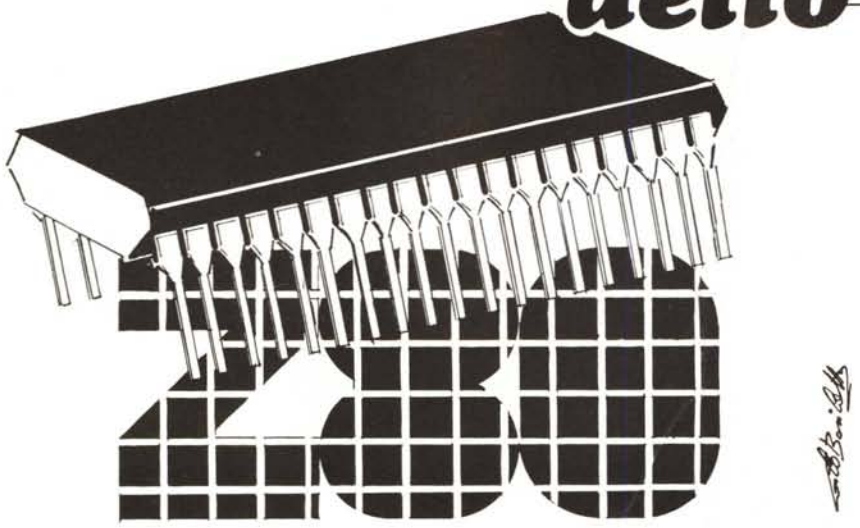
L'ASSEMBLER dello Z80



di Pierluigi Panunzi

In questa puntata parleremo delle istruzioni riguardanti la gestione dei blocchi di memoria: in particolare si tratta di istruzioni molto potenti e che non trovano riscontro in altri microprocessori ad 8 bit.

La gestione dei blocchi di memoria

Si intende innanzitutto per "blocco di memoria" un insieme di celle di memoria consecutive, di lunghezza prefissata e che inizia ad un ben determinato indirizzo.

Tali possono essere tabelle di valori, stringhe, oppure più semplicemente frammenti di programmi: in ogni caso si deve sempre trattare di locazioni di memoria consecutive.

Un problema che può sorgere dovendo lavorare su tali tipi di oggetti è di "spostarli" in un'altra zona della memoria, il cui indirizzo iniziale è a sua volta ben noto.

Per eseguire tale operazione in assembler si deve prevedere l'uso di

— un contatore, che perciò indica "quante" celle di memoria dobbiamo spostare

— un puntatore alle celle del blocco, puntatore che all'inizio si riferisce alla prima cella del blocco

— un puntatore alla zona di memoria dove il blocco deve essere spostato, puntatore che all'inizio si riferirà alla prima cella del blocco di arrivo.

L'operazione si dovrà svolgere nel seguente modo:

1) si carica in accumulatore il contenuto di una cella puntata dal primo puntatore e perciò appartenente al blocco di partenza

2) si memorizza tale valore nella cella puntata dal secondo puntatore e perciò nel blocco di arrivo

3) si incrementano di un'unità i due puntatori, per andare ora a puntare alla cella successiva

4) si decrementa il contatore delle celle da spostare: se il contatore è ancora diverso da 0, allora si torna al punto 1), altrimenti si termina.

Ora questo tipo di problema potrebbe essere risolto, per un microprocessore meno potente dello Z80 (ad esempio il 6502 o l'8080), con un programmino del tipo:

```
        LOAD CONT, lunghezza blocco
        LOAD P0, indir. iniziale
        LOAD P1, indir. d'arrivo
LABEL   LOAD A, cella puntata da P0
        LOAD cella puntata da P1,A
        INCR P0
        INCR P1
        DECR CONT
        JUMP NZ,LABEL
```

È evidente che il linguaggio usato non si riferisce ad alcun microprocessore, ma se ci caliamo ad esempio nel primo di quelli segnalati, il 6502, ci accorgiamo che anche un programmino così semplice comporta alcuni problemi:

a) innanzitutto il contatore CONT dovrebbe essere a 16 bit per poter trattare blocchi di grandezza superiore a 256 celle di memoria: già 300 parole comporterebbero un allungamento del programma, fatto che non è certo auspicabile; ma nel 6502 non si hanno registri a 16 bit (a parte lo Stack Pointer ed il Program Counter!) e perciò si dovrebbero usare due celle di memoria consecutive.

b) Stesso discorso deve valere per i puntatori P0 e P1, i quali, dal momento che puntano celle di memoria, dovrebbero essere anche loro a 16 bit (ricordiamo che un indirizzo di memoria è formato da 16 bit, per l'appunto): anche in questo caso i registri X ed Y del 6502 non servono a molto in quanto innanzitutto ad 8 bit e poi perché per ottenere quello che vogliamo dovrebbero essere "appoggiati" ciascuno ad una coppia di celle di memoria. Anche così facendo lo "spostamento" in avanti rispetto agli indirizzi di partenza potrebbe arrivare solo a 256, il che non ci basta assolutamente per le nostre 300 celle.

c) Gli incrementi dei puntatori ed il decremento del contatore sarebbero molto macchinosi in quanto il 6502 stesso non è molto abituato a gestire e perciò incrementare e decrementare dati a 16 bit: in questo caso bisogna infatti incrementare di una unità la "parte meno significativa" del puntatore ed in caso di riporto bisogna incrementare di uno la parte "più significativa".

Viceversa per decrementare di uno il contatore bisogna decrementare la parte "meno significativa" ed in caso di "prestito", decrementare anche la parte più significativa. Il test per vedere se si è terminato lo spostamento infine deve essere fatto "sulle" due parti (meno e più significativa) con il controllo se entrambe si sono azzerate. Vogliamo vedere il tutto come si risolve con il nostro Z80?

Con lo Z80, il programmino precedente diventa:

```
LD BC, lungh. blocco
LD HL, indir. di partenza
LD DE, indir. di arrivo
LDIR
```

Mentre le tre istruzioni iniziali sono già note, ecco dunque la prima di quattro che realizza lo spostamento di blocchi: la LDIR.

Vediamo insieme cosa fa questa istruzione che, insieme alle altre citate in questa puntata, rappresenta un punto di forza dello Z80 rispetto alla concorrenza (!).

In particolare LDIR significa "LoaD, Increment and Repeat" e cioè:

— carica da memoria a memoria

— incrementa i puntatori

— ripeti se non zero

Ancora più in dettaglio abbiamo il seguente specchietto, che poi commenteremo:

LDIR

```
(DE) ← (HL)
DE ← DE + 1
HL ← HL + 1
BC ← BC - 1
Repeat until
BC = 0
```

Nel nostro caso dunque abbiamo le celle di memoria, di partenza e di arrivo, puntate rispettivamente da HL e da DE, entrambi a 16 bit; il contatore è rappresentato da BC ed è anche questo a 16 bit ed infine il test è effettuato sul contenuto di BC: il tutto automaticamente e con una sola istruzione.

Le altre tre istruzioni analoghe alla LDIR sono la LDI, l'istruzione LDDR e la LDD.

Vediamo perciò le tre tabelline relative alle tre istruzioni, che poi commenteremo.

LDI

(DE) ← (HL)
DE ← DE + 1
HL ← HL + 1
BC ← BC - 1

LDDR

(DE) ← (HL)
DE ← DE - 1
HL ← HL - 1
BC ← BC - 1
Repeat until
BC = 0

LDD

(DE) ← (HL)
DE ← DE - 1
HL ← HL - 1
BC ← BC - 1

L'istruzione LDI praticamente è una LDIR che però si ferma al decremento del contatore, senza effettuare il test sul raggiungimento dello zero.

L'istruzione LDDR e la "ridotta" LDD invece effettuano le stesse operazioni viste precedentemente, con la sola differenza che invece di incrementare di uno i puntatori li decrementano di un'unità: ciò può essere utile quando lo spostamento è fatto "al rovescio" e cioè partendo da indirizzi alti e retrocedendo verso indirizzi bassi.

In ogni caso il pregio delle quattro istruzioni è di racchiudere in se stesse parecchie operazioni elementari.

Per quanto riguarda il discorso dei flag settati o meno dalle istruzioni, si hanno i seguenti fatti:

— la LDIR e la LDDR, e cioè le due istruzioni che prevedono la ripetizione, azzerano i flag P/V, N ed H, mentre non toccano il Carry, lo Zero ed il Sign (e cioè i più importanti).

— la LDI e la LDD trattano i flag come le due istruzioni precedenti con l'unica differenza che gestiscono il flag P/V a seconda del risultato del decremento di BC: è questo un comportamento alquanto insolito, ma che ritroveremo nelle prossime istruzioni.

In particolare se per effetto del decremento, la coppia BC arriva a 0, allora il flag P/V diventerà 1; in caso contrario P/V sarà mantenuto a 0.

In un certo senso il flag P/V fa in questo caso le veci del flag Z (di Zero), che come sappiamo invece non viene alterato dall'istruzione di decremento di una coppia di registri, al raggiungimento dello zero.

La ricerca sui blocchi

In questo caso le quattro istruzioni che vedremo, analoghe alle precedenti, consentono di cercare, nell'ambito di un blocco di caratteristiche prefissate, una cella di memoria il cui contenuto è pari a quello dell'accumulatore.

Ma vediamo subito l'istruzione CPIR, secondo la tabellina del funzionamento:

CPIR

A — (HL)
HL ← HL + 1
BC ← BC — 1
Repeat until
BC = 0

or
A = (HL)

In questo caso il contenuto della cella puntata da HL viene confrontato con l'accumulatore, sottraendolo da quest'ultimo, così come fa l'istruzione CP di comparazione.

Successivamente viene incrementato il puntatore e decrementato viceversa il contatore: queste operazioni vengono ripetute fino a che si verifica una delle due condizioni:

— è stata trovata la cella ed in questo caso l'accumulatore è uguale al contenuto della cella puntata da HL

— il contatore BC è arrivato a 0.

Nel primo caso si ha una ricerca con esito positivo e viceversa nel secondo caso.

Le tre istruzioni di ricerca tabellare ed analoghe, come detto alle LD, sono la CPI, la CPDR e la CPD: vediamone le tabelline di funzionamento che commenteremo.

CPI

A — (HL)
HL ← HL + 1
BC ← BC — 1

CPDR

A — (HL)
HL ← HL — 1
BC ← BC — 1
Repeat until
BC = 0
or
A = (HL)

CPD

A — (HL)
HL ← HL + 1
BC ← BC — 1

L'istruzione CPI in particolare è uguale alla CPIR, ma non effettua la ripetizione automatica del procedimento.

Le istruzioni CPDI e CPD invece dopo aver effettuato il confronto decrementano il puntatore: la prima ripete tutto il procedimento, mentre la seconda no.

Per quanto riguarda i flag, c'è da dire che in questo caso abbiamo un comportamento identico nei quattro casi.

In particolare settano il flag N, non toccano il Carry, il Sign e l'H: viceversa alterano lo stato dei due flag rimanenti e cioè Z e P/V. Per quest'ultimo valgono le considerazioni viste per il caso delle LD e cioè che il flag P/V viene settato se la coppia di registri BC arriva a 0 per effetto dei decrementi successivi. Il flag Z invece viene settato allorché si ha la coincidenza tra l'accumulatore ed una cella di memoria.

A seconda dello stato dei due flag all'"uscita" di tali istruzioni, si può capire che cosa è successo:

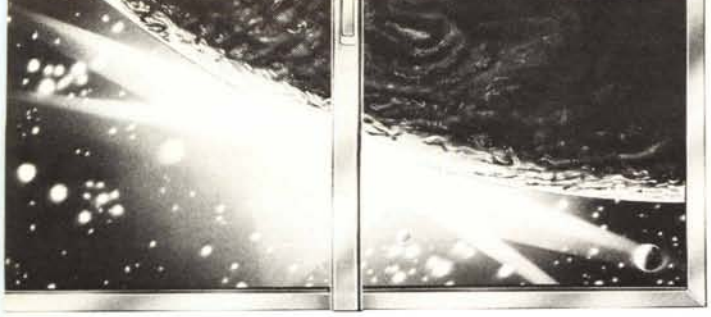
1) Z = 0 e P/V = 1: la ricerca ha avuto un esito negativo, in quanto BC è arrivato a 0 senza che si avesse un confronto favorevole.

2) Z = 1 e P/V = 0: è stata trovata la cella interessata ed in tal caso HL contiene il suo indirizzo già incrementato di 1, mentre BC indica il suo posto nella tabella a partire dal fondo.

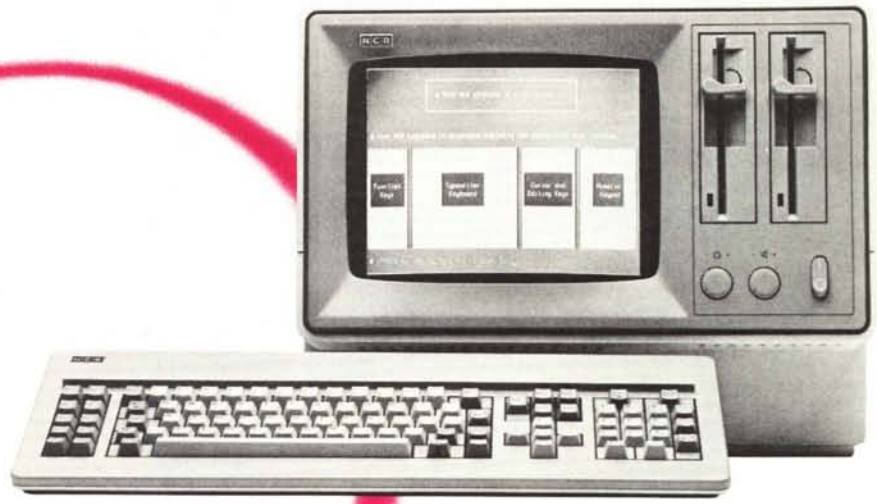
3) Z = 0 e P/V = 0: questa condizione si verifica solo per le due istruzioni non ripetitive, in quanto le altre arrivano in ogni caso ad annullare BC e perciò a settare P/V. Invece in un qualsiasi momento si sia eseguita una CPI o una CPD può capitare appunto di non aversi né l'uguaglianza né l'annullamento di BC.

4) Z = 1 e P/V = 1: si è avuta l'uguaglianza proprio all'ultima cella della tabellina ed allora anche in questo caso HL contiene l'indirizzo della cella aumentato di 1.

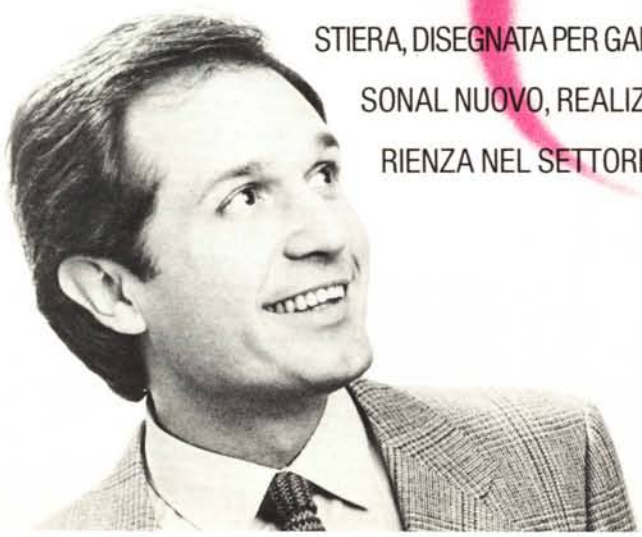
Con questo abbiamo terminato con le istruzioni di spostamento e di ricerca relative a blocchi di celle di memoria: la prossima puntata parleremo delle istruzioni di I/O, che si dividono in "semplici" e "per blocchi". MC



DAL MONDO NCR ARRIVA UN PERSONAL NUOVO.
DIVERSO DA QUELLI CHE CONOSCI, UGUALE A QUELLO CHE VORRESTI.



TU VUOI UN PERSONAL NUOVO: DAI CONTENUTI TECNOLOGICI AVANZATISSIMI, MA SEMPLICE DA USARE. UN PERSONAL COMPATTO, BELLO DA VEDERE, REALIZZATO SECONDO I PIÙ MODERNI CRITERI DI ERGONOMIA E FUNZIONALITÀ: CON IL VIDEO, L'ELETTRONICA E LE UNITÀ DI MEMORIA DI MASSA RACCOLTE IN UN INSIEME INTEGRATO, IN MODO DA OCCUPARE POCO SPAZIO SULLA TUA SCRIVANIA. TU VUOI UN PERSONAL CHE TI CONSENTA UN'ASSOLUTA COMPATIBILITÀ HARDWARE E SOFTWARE CON GLI STANDARD PIÙ DIFFUSI, E CHE TI OFFRA UN'ALTA DEFINIZIONE DELLO SCHERMO, SIA NELLA VERSIONE MONOCROMATICA SIA IN QUELLA A COLORI. TU VUOI UN PERSONAL CON UN'AMPIA GAMMA DI PRODOTTI APPLICATIVI, E CON UNA NUOVA TASTIERA, DISEGNATA PER GARANTIRTI IL MASSIMO COMFORT OPERATIVO. TU VUOI UN PERSONAL NUOVO, REALIZZATO DA UN'AZIENDA CON UNA LUNGA E QUALIFICATA ESPERIENZA NEL SETTORE. IL PERSONAL CHE VUOI SI CHIAMA PC4i. TE LO OFFRE NCR.



NCR
PROTAGONISTA DELL'INFORMATICA.

SEDE E DIREZIONE GENERALE: 20143 MILANO - VIALE CASSALA, 22 - TEL. 02/838741
(20 LINEE) - TELEX 320395 - NCR È SULLE PAGINE GIALLE DI TUTTA ITALIA.