ASSEMBLER 8086 8088

di Pierluigi Panunzi

Il set di istruzioni Istruzioni logiche

prima parte

Siamo arrivati al gruppo di istruzioni che consentono di effettuare le più comuni operazioni logiche in genere su operandi ad 8 o a 16 bit e generalmente alterando alcuni dei flag principali.
Fondamentalmente le istruzioni logiche si possono suddividere in tre sottogruppi, a seconda se operano su uno, su due o su nessun operando, in ogni caso ad 8 o 16 bit.
In questa puntata cominciamo dalle istruzioni che agiscono su di un solo operando. ■

Le istruzioni di Shift e di Rotate

Appartengono a questo gruppo le istruzioni che agiscono a livello di bit sull'operando, effettuando quell'operazione logica chiamata genericamente «shift», consistente, per chi non lo sapesse, nel far «scorrere» i bit che costituiscono l'operando o verso destra («right shift») o verso sinistra («left shift»): nell'effettuare questa operazione è interessante vedere che fine fa il bit che «esce» dall'operando come pure bisogna decidere quale valore deve assumere il bit che viceversa «entra» all'interno dell'operando.

In breve, se il bit che entra era quello appena uscito dall'altra parte allora si ha una «rotate» (sia «left» che «right»), mentre in altri casi, che analizzeremo in dettaglio nel seguito si avrà una «rotate through carry» oppure ancora un «arithmetic shift».

Comunque diciamo subito che tutte le operazioni di «shift» (intendendo con questo termine generale anche le «rotate», che sono sempre delle «shift») possono essere effettuate o un bit alla volta oppure per un numero di bit contenuto nel registro CL.

Andiamo però con ordine osservando la tabellina che riporta i tipi di operandi possibili per tutti i tipi di istruzioni di shift, dove abbiamo indicato con «operando» e «termine» gli elementi di una generica istruzione di shift (SHR, SAR, SHL o SAL, ROL, ROR, RCR, RCL) che interessano sia un solo bit che il numero di bit contenuto in CL:

Istruzioni di shift

Operando	Termine
1) memoria	1
2) memoria	CL
3) registro	1
4) registro	CL

Ancora una volta, come abbiamo già visto nelle scorse puntate, i termini indicati nella tabellina precedente hanno il seguente significato:

— «Memoria»: indica una qualunque locazione di memoria indirizzata direttamente o indirettamente o tramite un registro base o tramite un registro indice, secondo le ben note modalità;

— «Registro»: è uno qualsiasi dei registri, sia i «general purpose» (AX, BX, CX, DX se i dati sono a 16 bit e AL, AH, BL, BH, CL, CH, DL e DH se i dati sono ad 8 bit), che i registri indice (DI e SI), nonché i registri BP e SP: sono pure possibili operazioni di shift sui quattro registri di segmento, compreso anche CS, che di solito non è facile manipolare.

Alcuni esempi di istruzioni relativi ai quattro tipi indicati sono i seguenti:

```
1) SHL ALFA, 1
   ROR [BX][SI], 1
   SAR BETA+5,1

2) ROL ALFA[BP][DI+3], CL
   SHL [BX], CL

3) SHR BX, 1
   RCR CS, 1
   SAL DI, 1
   ROR BL, 1
   ROR BP, 1

4) SHR CL, CL
   ROL CX, CL
   SHL BP, CL
```

dove possiamo vedere che si possono usare tutti i registri interni della CPU come operandi.

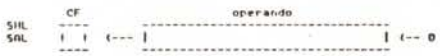
Analizziamo ora in dettaglio i vari tipi di istruzioni di «shift», abbinandole a coppia (la «left» con la rispettiva «right»).

Le istruzioni SHL (o SAL), SHR e SAR

Ecco dunque le istruzioni capostipiti del gruppo, che si leggono rispettivamente «Shift Left» (o «Shift Ari-

thmetic Left»), «Shift Right» e «Shift Arithmetic Right»: le prime due effettuano lo shift a sinistra ed a destra dei bit dell'operando, ponendo il bit «uscente» nel Carry (CF) e viceversa facendo «entrare» uno 0 dall'altra parte, mentre l'ultima lascia inalterato il bit più significativo.

Per comprendere meglio quali sono le operazioni compiute da queste istruzioni (e lo stesso verrà fatto per le istruzioni successive) abbiamo realizzato alcuni diagrammi, che indicano subito, con un colpo d'occhio, ciò che fa ogni funzione:



Nel caso della SHL/SAL si ha il vero e proprio «shift» a sinistra, operazione che consente di moltiplicare per 2 l'operando, che abbiamo supposto indifferentemente ad 8 o a 16 bit, dal momento che non cambia nulla: dato che l'operando può anche esprimere quantità negative (se aveva il bit più significativo ad «1»), può capitare che effettuando lo shift a sinistra il nuovo bit più significativo sia pari a «0» nel qual caso si otterrebbe che il doppio di un numero negativo è un numero positivo.

Potrebbe pure succedere che il bit più significativo sia pari a «0» (indicando così un numero positivo) e che dopo lo shift a sinistra diventi «1», ottenendo così che il doppio di un numero positivo è una quantità negativa.

Non è però un problema, perché comunque il bit uscente viene posto nel carry ed il microprocessore effettua il test se il (nuovo) CF è diverso dal (nuovo) bit più significativo dell'operando ed in tal caso setta il flag di overflow (OF).

Questo discorso, attenzione, vale solo nel caso delle istruzioni che shiftano di un bit, mentre per quelle in cui il numero di shift è contenuto in CL il flag OF rimane indefinito.

Anche nel caso della SHR, dato che il bit più significativo viene sostituito da uno «0», può capitare che un valore originariamente negativo (bit più significativo pari ad «1») diventi positivo dopo uno shift a destra, che equivale ad una divisione per due, il cui resto è proprio il carry: comunque il processore setta il flag OF se il (nuovo) bit significativo è diverso dal bit immediatamente a destra (e cioè il vecchio bit più significativo, prima dello shift).

Sarà poi comunque cura del programmatore prendere i dovuti provvedimenti nel caso in cui OF è settato, mentre anche in questo caso, se lo shift è multiplo (numero di shift contenuto in CL), il bit OF non è utilizzabile.

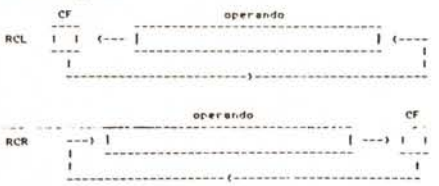
Invece l'istruzione SAR, effettua uno shift aritmetico a destra, intendendo con questo termine che viene mantenuto il segno dell'operando originario: il (vecchio) bit più significativo oltreché shiftare verso destra, viene lasciato anche al suo posto ed in tal modo una quantità negativa rimane negativa ed una positiva rimane positiva.

Notevole è il fatto che questo discorso vale sia se si shifta di un bit sia nel caso di shift multipli: supponendo ad esempio che si voglia shiftare di 5 bit un operando verso destra aritmeticamente, allora il risultato finale avrà almeno cinque bit pari ad «1» o a «0» (da sinistra) a seconda se l'operando era rispettivamente negativo o positivo.

In tutti questi casi, dato il meccanismo dell'istruzione, il flag OF non potrà mai essere settato.

Le istruzioni RCL, RCR, ROL e ROR

Ecco dunque le istruzioni di shift «chiuse su se stesse» e che si leggono rispettivamente «Rotate through Carry Left» «Rotate through Carry Right» ed infine «Rotate Left» e «Rotate Right».



Nei primi due casi il meccanismo prevede che il bit che «esce» dall'operando va a «scalzare» il bit contenuto nel Carry Flag, il quale bit «rientra» dall'altra parte nell'operando: ovviamente nella prima istruzione lo shift avviene verso sinistra, mentre nella seconda verso destra. Vediamo ora le altre due «rotate».

In questi due casi la rotazione avviene semplicemente all'interno dell'operando ed il carry conterrà una replica del bit che, uscendo da una parte, rientra dall'altra: in questo caso si ha una rotazione «pura» di quantità a 16 oppure ad 8 bit, a seconda della scelta del programmatore, mentre nel caso delle istruzioni RCR e RCL il Carry Flag funge da diciassettesimo o nono bit oppure come vero e proprio «bit di riporto».

Ma analizziamo un esempio per comprendere l'utilità delle rotate attraverso il carry.

Supponiamo di avere una certa quantità espressa con 32 bit (ad esempio il risultato di una moltiplicazione) e posta in due celle di memoria (due word) chiamate HIGH e LOW, contenenti rispettivamente la parte alta e la parte bassa della double-word di partenza: supponiamo di dover effettuare uno shift a sinistra dell'intera quantità a 32 bit.

Come si può verificare graficamente, il problema consiste nell'inserire un bit pari a «0» nella word chiamata LOW, fare uscire il bit più significativo da essa per inserirlo nella word HIGH, dalla quale far uscire il bit più significativo che andrà nel carry.

Ecco che perciò sulla word LOW dovremo effettuare uno shift verso sinistra e cioè una SHL (o SAL, che è la stessa cosa), con il che il vecchio bit più significativo di LOW andrà a finire nel carry.

Ora questo bit deve «entrare» in HIGH, da quale deve uscire il bit 15: ecco che dunque il carry attualmente rappresenta il diciassettesimo bit di HIGH, da porsi direttamente alla destra del bit meno significativo di HIGH.

Ora una «rotate» attraverso il CF (si riveda la sua figura per avere ben chiara la situazione) farà proprio quello che vogliamo: tra l'altro il bit che ora è andato a finire nel Carry Flag potrebbe già essere pronto per una nuova «rotate» nel caso in cui si avesse a che fare non con quantità a 32 bit, ma a 48, 64, ecc.

In definitiva per ottenere lo shift a sinistra della doubleword contenuta in HIGH e LOW bastano le due istruzioni:

```
SHL LOW, 1
RCL HIGH, 1
```

Sapendo poi che le istruzioni di shift agiscono indifferentemente su quantità ad 8 e a 16 bit, si possono risolvere problemi su quantità «miste», come ad esempio quantità a 24 bit, rispettivamente formate nella parte alta da un byte (che supporremo chiamarsi HIBYTE) e nella parte bassa da una word (chiamata in questo caso LOWORD): lo shift a sinistra della quantità a 24 bit formata dunque da HIBYTE e LOWORD si ottiene ancora una volta con due istruzioni:

```
SHL LOWORD, 1
RCL HIBYTE, 1
```

ma si capisce subito che non ci sarebbero stati problemi se la parte alta fosse stata in una word (HIWORD) e la parte bassa in un byte (LOBYTE), nel qual caso le istruzioni sarebbero diventate:

```
SHL LOBYTE, 1
RCL HIWORD, 1
```

Tutto questo nel caso di shift di un bit soltanto: nel caso di una quantità a 32 bit (e perciò posta su due word) che deve essere shiftata di più bit, non si può assolutamente ragionare come per i due esempi riportati in quanto porterebbero a risultati completamente errati.

Supponiamo dunque di voler shiftare di 4 bit verso sinistra la quantità a 32 bit posta in HI (parte alta) e LO (parte bassa), dove HI e LO sono ovviamente due word.

Quello che NON dovremo fare in questo caso è:

```
MOV CL, 4
SHL LO, CL
RCL HI, CL
```

in quanto così passeremmo da LO ad HI solamente un bit, per la precisione il quarto, invece dei quattro che vogliamo.

In questo caso invece dovremo creare un loop da ripetere quattro volte e tale che ogni volta venga passato un bit dalla word LO alla word HI: rimandando i dettagli dell'istruzione LOOP alla puntata in cui parleremo delle istruzioni di controllo del flusso del programma, il nostro problema si risolve con appena quattro istruzioni:

```
      MOV CX, 4
LAB:  SHL LO, 1
      RCL HI, 1
      LOOP LAB
```

dove appunto la LOOP salta all'etichetta LAB fintanto che CX si mantiene maggiore di 0, decrementandolo ogni volta: quando CX si azzera allora si uscirà dal loop e si proseguirà nel programma.

Supponiamo ora per esempio di voler effettuare la rotazione verso destra di una quantità a 32 bit, posta nei due registri DX ed AX (rispettivamente la parte alta e quella bassa). Graficamente vogliamo ottenere ciò:

In questo caso dobbiamo stare attenti al fatto che vogliamo una rotazione totale e perciò non dobbiamo perdere alcun bit: possiamo cominciare tanto da DX che da AX e noi sceglieremo il primo.

Allora dovremo:

— shiftare a destra DX salvando il suo (vecchio) bit 0 nel carry: non ci dobbiamo dimenticare che ora il (nuovo) bit 15 è stato riempito con uno «0» dall'esterno, mentre non è detto che il bit che arriverà da AX (il suo vecchio bit 0) sia proprio uno «0», anzi abbiamo solo il 50% delle probabilità!

— effettuare una rotazione attraverso il carry di AX in quanto ora il bit di ingresso in AX è proprio il carry: viceversa sappiamo che il bit uscente va a finire ancora una volta nel CF.

— ripristinare il bit più significativo di DX per tener conto del bit uscito da AX.

In termini di istruzioni il terzo punto si può risolvere in svariati modi, che dipendono senz'altro dai gusti del programmatore: una prima soluzione è la seguente:

```
      SHR DX, 1
      RCR AX, 1
      JNC OLTRE
      OR DX, 8000H
OLTRE: ...
```

In questa soluzione, dopo la rotate di AX si testa il CF: se non è settato (condizione di NC, «Not Carry») allora si salta all'etichetta «OLTRE», mentre se il carry è settato allora si forza il bit più significativo di DX con un OR opportuno.

Altra soluzione, che non richiede istruzioni di salto, è quella che prevede le seguenti operazioni:

— il salvataggio iniziale di DX in un altro registro (ad esempio SI, tanto per ribadire il fatto che a livello logico-matematico i registri sono praticamente equivalenti)

— lo shift a destra di DX

— la rotate di AX a destra, con il che il (vecchio) bit 0 di AX è arrivato nel CF

— la rotate a destra di SI, con il che il bit 15 è stato correttamente sostituito dal vecchio bit 0 di AX

— il salvataggio di SI in DX.

Come si vede questa soluzione è alquanto più elegante anche perché prevede l'uso di registri della CPU e soprattutto una dose maggiore di ragionamento alla base...

In definitiva il nostro problema si risolve con:

```
MOV SI, DX
SHR DX, 1
RCR AX, 1
RCR SI, 1
MOV DX, SI
```

dove l'ultima MOV potrebbe essere sostituita da una XCHG.

Vediamo dunque un ultimo esempio: supponiamo ora di voler ruotare di 3 bit verso sinistra la quantità a 48 bit contenuta nelle tre celle di memoria (tre word), che dalla più significativa alla meno supponiamo chiamarsi HW, MW e LW.

Il problema è ancora quello di dover impegnare tre bit invece di uno, senza perderne ovviamente alcuno: possiamo dunque implementare un ciclo da percorrere tre volte, all'interno del quale sfruttiamo l'idea di salvare in un registro la prima word da shiftare, per poi ripristinarla dopo aver ruotato la word più significativa.

Il programmino è dunque il seguente:

```
      MOV CX, 3
CICLO: MOV AX, LW
      SHL AX, 1
      RCL MW, 1
      RCL HW, 1
      RCL LW, 1
      LOOP CICLO
```

Anche in questo caso si può sfruttare un altro trucchetto, che evita di usare un registro d'appoggio, favoriti questa volta dal fatto che stiamo effettuando la rotazione verso sinistra: fissiamo l'attenzione su LW e cioè la word meno significativa.

Abbiamo visto che shiftandola verso sinistra di un bit, il bit meno significativo viene forzato a «0», dal momento che così funziona la SHL: dopo aver ruotato a sinistra anche HW dovremo prendere il suo (vecchio) bit più significativo e farlo diventare il bit 0 della word LW.

Ora se questo bit (che ricordiamo essere ancora una volta nel carry) è «0» allora non dobbiamo fare niente, mentre se è «1» allora dobbiamo riportarlo in LW: ciò si ottiene ad esempio con:

```
      MOV CX, 3
CICLO: SHL LW, 1
      RCL MW, 1
      RCM HW, 1
      JNC OLTRE
      OR LW, 1
OLTRE: LOOP CICLO
```

Ma ancora non siamo soddisfatti per la presenza del salto condizionato: pensandoci bene settare il bit 0 di una word se il carry è «1» si può ottenere semplicemente «sommando» alla word il carry!

E questo si ottiene «sommando con il carry» una quantità nulla alla word in esame: in parole povere basta una (ve la ricordate?) ADC...

Ecco che dunque il nostro programmino diventa:

```
      MOV CX, 3
CICLO: SHL LW, 1
      RCL MW, 1
      RCL HW, 1
      ADC LW, 0
      LOOP CICLO
```

In questo modo andiamo a sommare ad LW intanto il valore «0» e poi il carry (se per caso fosse pari a «1») il che è molto facile convincersi che equivale a trasferire il CF nel bit 0 di LW.

Con questo abbiamo terminato l'analisi delle istruzioni di shift: nella prossima puntata parleremo delle altre istruzioni logiche. MC

Hard & soft NIWA

LA PUÒ ESSERE LA TUA MIGLIORE AMIGA®

Distributore autorizzato COMMODORE

In regalo a tutti gli acquirenti di un PC AMIGA la tessera del NIWA AMIGA CLUB.

AMIGA costa £ 2.500.000 IVA comp. consegna GRATIS IN TUTTA ITALIA.

Tutto il software disponibile e l'hardware novità.

Inoltre la NIWA vi propone per il vostro C/64-C/128:

Floppy disk "Memorette" 5¼ ssdd 100% error free	cd	**L. 1.300**
Floppy disk bulk 3½ dsdd 100% error free	da	**L. 3.500**
Allinea testine Cartridge		**L. 32.000**
Allinea testine con turbotape e turbo 202		**L. 39.000**
MPS 802 New Graphic CON MONTAGGIO GRATUITO rende 100% compatibile la tua MPS 802 con i programmi di grafica		**L. 80.000**
O.M.A. Non permettere che i tuoi programmi originali si ROVININO. Con O.M.A., puoi fare una copia di sicurezza in un unico file (!) ricassettabile del tuo software su disco o su nastro		**L. 99.000**
HACKER Cartridge: trasferisce il 99% del tuo software protetto da nastro e da disco a disco in soli 4 minuti senza bisogno di conoscenza Linguaggio.		**L. 80.000**
HACKER-TAPE: permette di ricassettare qualsiasi tipo di programma predentemente trattato con HACKER, senza nessun problema di blocchi, leggendo in turbo da disco e scrivendo in turbo su nastro		**L. 45.000**
OFFERTA: HACKER + HACHER TAPE		**L. 99.000**
Speeddos per C64 **L. 65.000** per C128 **L. 85.000,** per 1541 C **L. 79.000,** Fast load reset **L. 35.000**, Isepic **L. 50.000**, Capture **L. 99.000,** Super Cartridge **L. 99.000,** Super Freere 3 **L. 99.000**		
Double side kit per scrivere sulla seconda faccia del dischetto senza più forarlo - disinserибile.		**L. 10.000**

Via Buozzi, 94 - P.O. BOX 83 - 20099 Sesto - S. Giovanni (MI) MM. Marelli
Tel. 02/2440776-2476523

Hard & soft NIWA