Assembler 68000

di Marco Pesce

Abbiamo trascorso le vacanze con molte cose da studiare (mi riferisco al materiale dello scorso numero...) e spero che il tentativo di catturare la vostra attenzione su questo affascinante argomento sia riuscito, ovvero che non vi sia passata la voglia di imparare il mitico linguaggio macchina dopo appena la prima lezione

In effetti la mole di argomenti trattati nello scorso numero era relativamente notevole e magari ci sarà qualcuno che avrà incontrato delle difficoltà con tutta quella teoria; non è il caso di preoccuparsi perché in questo numero ci dedicheremo alla stesura di alcuni listati chiarificatori. Prima però vorrei completare l'elenco dei modi di indirizzamento dello scorso numero. Eravamo rimasti al modo «assoluto corto». In questa modalità l'indirizzo di memoria è specificato direttamente da un valore a 16 bit che subisce la solita estensione del segno. Ad esempio l'istruzione:

```
MOVE.L $4000,D0
```

Trasferisce il contenuto della longword contenuta a partire dalla locazione $4000 nel registro dati D0.

Il prossimo modo è simile al precedente solo che la locazione viene specificata da un dato a 24 bit, quindi ad esempio:

```
MOVE.W $F45000,D3
```

che sposta in D3 la word contenuta a partire da $F45000. L'indirizzamento "PC con scostamento" si avvale del program counter per individuare la locazione da indirizzare, ovvero prende il contenuto del PC e lo utilizza come indirizzo base, al quale va poi aggiunto uno scostamento di 16 bit. Vediamo l'esempio;

```
MOVE.B 100(PC),A2
```

Tale istruzione prende la locazione posizionata 100 locazioni più avanti a partire dall'istruzione MOVE e la trasferisce in A2.

Il penultimo modo di indirizzamento prevede ancora l'utilizzo del PC, ma aggiunge anche un indice (contenuto in un registro). Per esempio, l'istruzione:

```
MOVE.B 30(PC,A0.W),A1
```

ricava la locazione sommando il PC con il valore 30 e con il valore del registro indirizzi A0; il contenuto della locazione risultante è trasferito in A1.

E siamo giunti all'ultimo modo, ovvero quello "immediato" che non implica né registri né locazioni di memoria; il dato da trasferire infatti viene specificato direttamente nell'istruzione, quindi:

```
Elenco di alcune  delle istruzioni del 68000

-ADD;  addiziona
-SUB;  sottrae
-MOVE; sposta
-AND;  operazione di AND logico
-OR;   operazione di OR logico
-NOT;  operazione di NOT logico
-CMP;  confronta
-BEQ;  salta se uguale
-BNE;  salta se diverso
-BMI;  salta se negativo
-BPL;  salta se positivo
-BCC;  salta se il CARRY e resettato
-BCS;  salta se il CARRY e resettato
-JMP;  salta
-JSR;  salta alla subroutine
-RTS;  ritorna dalla subroutine

-----------------------------------------
```

Figura 1

```
        MOVE.L #1000,D0
LOOP    SUB.L #1,D0
        BNE LOOP
        RTS
```

Figura 2

MOVE.W #$A3F2,D0

trasferisce direttamente il valore $A3F2 (nota il simbolo #, che indica appunto il modo "immediato") in D0.

Esistono altre modalità che non sono state elencate nella tabella della puntata precedente e sono il "riferimento implicito" che è quello utilizzato da istruzioni che non necessitano di operandi (ad esempio la "RTS") e il modo "quick" che permette di ottenere una maggiore velocità a discapito però della lunghezza dei dati. Vediamo in dettaglio quest'ultimo modo. L'istruzione:

MOVEQ #10,D0

trasferisce il dato 10 in D0, ma tale dato è esclusivamente in formato byte, mentre l'istruzione:

ADDQ.W #1,D0

aggiunge il valore 1 al registro D0 con la limitazione che tale valore oscilla tra 1 e 8.

Le prime istruzioni

Terminato, per il momento, il discorso sui modi di indirizzamento cominceremo ad esaminare le prime istruzioni che ci permetteranno di scrivere alcuni semplici listati. In figura 1 potete osservare un ridotto elenco di istruzioni con relativa sintetica descrizione. Tali istruzioni possono essere sufficienti alla stesura di listati anche molto complessi, ma in alcuni casi è preferibile utilizzare anche le altre istruzioni del 68000 che permettono compiti più specifici (come, ad esempio, l'operazione di moltiplicazione).

È evidente che conoscere un linguaggio e non conoscere la macchina sul quale lo si utilizza ci impedirebbe di sfruttarlo per scopi che vanno al di là del semplice esempio didattico; ovviamente noi non vogliamo limitarci a ciò, quindi osserveremo con attenzione le risorse del sistema operativo di Amiga e quelle hardware, in modo da accedere ai dispositivi di input/output e sfruttare al meglio le possibilità del computer. Per il momento cominciamo a prendere confidenza con il nostro pane quotidiano, appunto, le istruzioni, preoccupandoci più che altro di capire la loro funzione e di entrare quindi nella logica dell'LM (ulteriori approfondimenti sui possibili modi di indirizzamento previsti per una data istruzione e affini verranno fatti in seguito).

Il listato di figura 2 utilizza la ben nota MOVE unitamente ad altre tre nuove istruzioni. Vediamo il suo funzionamento. La MOVE deposita nel registro dati D0 il valore 1000 (indirizzamento "immediato"). La scritta "LOOP" non è una istruzione, ma una "label" ovvero una sorta di segnale che contrassegnerà quel punto del programma nel quale essa si trova; ovviamente possono essere "piazzate" label un po' ovunque, dove ci fa più comodo. Dopo la label troviamo l'istruzione SUB. La sua funzione è quella di sottrarre il valore dell'operando sorgente (quello alla sinistra dell'istruzione) all'operando destinazione (il registro D0); in questo caso viene sottratto il valore 1 (sempre indirizzamento "immediato") al registro D0, che prima era stato riempito con il valore 1000 e che quindi viene decrementato a 999. Procediamo con la lettura e incontriamo una BNE. È una delle tante istruzioni che permettono di prendere "decisioni" ovvero che condizionano dei salti o il proseguimento del programma a seconda se la condizione imposta sia verificata o meno, in questo caso la condizione è che l'ultima operazione eseguita abbia dato come risultato un numero diverso da zero. In questo caso l'istruzione effettua un salto alla locazione indicata nella label. Nell'esempio il salto verrà effettuato in quanto 999 è un valore diverso da zero. Il salto ci riporterà al punto indicato dalla label "LOOP", quindi all'istruzione SUB. Il registro dati D0 verrà quindi nuovamente decrementato e il ciclo si ripeterà fin quando in D0 ci sarà il valore 0. A questo punto l'istruzione BNE non vedrà verificata la sua condizione e il programma continuerà con l'istruzione successiva, ovvero la RTS. Questa è un'istruzione che di solito si trova alla fine di una subroutine (come potrebbe essere considerata quella in esame) e impone un ritorno al punto che l'aveva chiamata (vedremo un esempio con il listato di figura 4). Nel nostro caso essa sta ad indicare la fine del "programma" e quindi se un tale listato fosse realmente assemblato (non è vietato farlo) e lanciato, al termine dei cicli di "sottrazione" il programma terminerebbe e verrebbe restituito il controllo al sistema (nella fattispecie il CLI dell'AmigaDOS).

```
LOOP    MOVE.W #$FFFF,D0
        NOT D0
        BNE LOOP
        RTS
```

Figura 3

```
LOOP    MOVE.L #10,A0
        JSR TEST1
        BEQ LOOP
        RTS
TEST1   AND #1,A0
        RTS
```

Figura 4

Passiamo al prossimo programmino, quello di figura 3. Esso introduce soltanto una nuova istruzione, la NOT, che rientra nella categoria degli operatori logici (vedi anche AND e OR). In questo caso particolare l'istruzione comporta "l'invertimento" dei bit dell'operando specificato nell'istruzione (il registro D0). Dal momento che con l'istruzione precedente tale registro era stato impostato a $FFFF (ovvero tutti i bit della word settati ad 1), con quest'ultima istruzione i bit verranno invertiti e quindi saranno tutti azzerati. Ovviamente la successiva istruzione BNE non effettuerà il salto e quindi verrà eseguita la RTS.

L'ultimo esempio (figura 4) implica il funzionamento di una subroutine (a scopo puramente dimostrativo, ovviamente). Seguiamo il flusso del programma: viene "caricato" in A0 il valore 10; si esegue il salto alla subroutine indicata dalla label "TEST1", grazie all'istruzione JSR; non appena si incontrerà una istruzione RTS il programma tornerà a questo punto (per "ricordarselo" il 68000 deposita il valore corrente del PC nello stack, vedi puntata precedente); la "subroutine" esegue un gruppo di due istruzioni; la prima è una AND che appunto esegue un AND logico tra il valore 1 e il valore contenuto nel registro D0, depositando il tutto nel registro D0; nel registro D0 ci sarà ora il valore 0 (visto che "1 and 10" dà proprio 0); siamo giunti alla fine della subroutine (istruzione RTS), quindi si ritorna alla locazione che aveva effettuato la chiamata e si prosegue con l'istruzione seguente (alla JSR) che è una istruzione simile alla BNE, ma questa volta la condizione è che il risultato dell'ultima operazione sia 0. Condizione che viene verificata e quindi il programma "uscirà". Non abbiamo con questi tre listati esaurito l'elenco di figura 1, tuttavia l'importante è essere entrati nell'ottica giusta (e questo lo sapete solo

voi se ci siete riusciti). Proseguiamo con una spiegazione delle altre istruzioni. L'istruzione ADD è l'inversa della SUB e ovviamente serve per addizionare valori. L'istruzione AND è stata già esaminata, ma forse qualcuno è a digiuno di nozioni di logica binaria, quindi apriamo una brevissima parentesi sull'argomento. In istruzioni di questo tipo vengono interessati direttamente i bit degli operandi, in coppie formate prendendo di volta in volta un bit dall'operando sorgente e il suo corrispondente (ad esempio il bit 0 con il rispettivo bit 0 e così via) nell'operando destinazione ed effettuando singolarmente per tutte le coppie l'operazione logica indicata. Anche le istruzioni NOT e OR, come detto, sono operatori di logica binaria. L'AND confronta i due bit di una coppia e se sono entrambi a 1 il risultato sarà che il bit dell'operando destinazione sarà impostato ad 1. Il confronto riguarda tutti i bit che abbiamo specificato (a seconda del formato byte, word o longword). Per l'istruzione OR è sufficiente che uno solo dei due bit sia ad 1 per impostare il relativo bit dell'operando destinazione ad 1. L'operazione NOT implica un solo operando ed è eseguita su se stesso; i bit posti ad 1 vengono azzerati e viceversa. Chiudiamo la parentesi e torniamo alle altre istruzioni. La CMP è un'istruzione di confronto che controlla tutti i bit dei due operandi (sorg. e dest.) e se sono tutti coincidenti viene attivato il flag di zero, quindi un'eventuale istruzione di diramazione come la BEQ effettuerebbe il suo salto.

Le successive sei istruzioni sono tutte della categoria "diramazioni"; le BEQ e BNE le conosciamo; la BMI salta se il risultato è negativo, mentre la BPL salta se il risultato è positivo (entrambe secondo la logica del complemento a due); la BCC salta se il flag di CARRY è azzerato e la BCS ovviamente salta nel caso opposto.

L'istruzione JMP è simile alla JSR, ma non memorizza la locazione dalla quale è partita nello stack, in quanto non effettua un salto ad una subroutine, ma ad un altro punto del programma. Abbiamo così concluso con il nostro elenco di figura 1. Il mio consiglio è quello di provare a scrivere listatini per conto vostro e di toccare quindi con mano la materia, in modo da dissipare anche i più piccoli dubbi. Vi faccio notare che non abbiamo specificato nulla nei riguardi dei modi di indirizzamento possibili con una determinata istruzione e dei flag che tale istruzione potrebbe attivare (tranne qualche piccolo accenno), mentre per una completa acquisizione occorrono anche specificazioni in tal senso. Andiamo per un attimo a riguardarci la tabella di figura 1 dello scorso numero; possiamo scorgervi 4 colonne che contengono delle X. Esse ci saranno di aiuto in seguito, ovvero quando presenteremo un elenco completo delle istruzioni e delle loro specifiche, per individuare immediatamente se il modo di indirizzamento «tal dei tali» è incluso tra quelli utilizzabili in una determinata istruzione. Tanto per fare un esempio, l'istruzione NOT permette di indirizzare solo i modi DATI ALTERABILE (nelle colonne citate D e A), quindi potranno essere utilizzati con essa tutti i modi che hanno una X sia nella casella del modo DATI che in quella ALTERABILE. E con questo vi saluto. MC