di Pierluigi Panunzi

Un'altra serie di istruzioni (5)

In questo numero proseguiremo l'analisi delle istruzioni del 386: questa analisi proseguirà per altre puntate ancora e solo al termine riprenderemo un discorso molto più importante e stimolante, quello della programmazione in Protected Mode e di tutte le nuove sfaccettature introdotte con l'80386.
Per ora accontentiamoci dunque delle istruzioni base del nostro microprocessore, che già a paragone del 286 contengono innovazioni notevoli

Le istruzioni di gestione delle stringhe

La novità apportata dal 386 nelle istruzioni di gestione delle stringhe, così come abbiamo più volte detto per altre istruzioni già analizzate nelle puntate precedenti, riguarda non tanto l'introduzione di nuove funzionalità, ma bensì l'estensione a 32 bit dei concetti noti, con le ovvie conseguenze che comporta tale estensione e che abbiamo già imparato ad individuare subito.

In particolare ricordiamo che il concetto di «stringa» in questo caso è una generalizzazione di quello che conosciamo: mentre una stringa in genere è una serie di carattere ASCII (perciò byte), preceduta da un byte contenente la lunghezza della stringa stessa, nel caso dei microprocessori della serie 80×86 per stringa si intende una serie di elementi che possono essere byte, word e (nel caso del 386) double-word, in tutti i casi senza la necessità di avere in testa alla «stringa» un contatore.

All'elenco formato dalle istruzioni MOVS, LODS, STOS, SCAS, CMPS, INS ed OUTS (che non specificano implicitamente il tipo di elementi della stringa, ma che lo richiedono come operando) e dalle istruzioni MOVSB, MOVSW, LODSB, LODSW, STOSB, STOSW, ecc. (nelle quali il tipo è viceversa specificato nell'istruzione stessa), dobbiamo dunque aggiungere le varie MOVSD, LODSD, STOSD, SCASD, CMPSD, INSD ed OUTSD.

Senza scendere ancora una volta nel dettaglio delle singole nuove istruzioni, per il quale rimandiamo alle puntate della rubrica Assembler 8086 (per la precisione i numeri 63 e 64 di MC...), diamo soltanto alcune informazioni riguardanti le funzioni estese: in tutti i casi i registri che consentono l'indirizzamento degli elementi della stringa sono diventati quelli estesi.

In particolare la stringa «sorgente» è ora puntata dalla coppia DS:ESI, quella di «destinazione» dalla coppia ES:EDI, il contatore (nel caso delle istruzioni di stringa precedute da uno dei prefissi di ripetizione REPx) è ora ECX, mentre l'accumulatore è ora EAX; inoltre, ricordando che queste istruzioni decrementano o incrementano automaticamente i puntatori (a seconda dello stato del «Direction Flag» DF), nel caso del 386 tale incremento o decremento è di 4 unità, appunto il numero di byte che formano una double-word.

Detto questo, non resta da aggiungere che qualche considerazione.

Per quanto riguarda le istruzioni INSD e OUTSD, c'è da aggiungere che il registro che indica l'indirizzo della porta di I/O interessata è ancora una volta DX (senza estensioni) in quanto anche nel 386 lo spazio di I/O è «limitato» a 65536 indirizzi. Viceversa, però, queste istruzioni si utilizzeranno solo laddove la porta in esame è a 32 bit, contrapposta agli 8 bit usuali (porte di comunicazione, ecc.) ed ai 16 bit (porte delle schede video, in particolare la VGA): a quanto ci consta nessun fabbricante ha previsto porte di I/O con tale elevato numero di bit, ma in caso affermativo potremo subito utilizzarle (solo però con il 386).

Altra considerazione riguarda l'uso di prefissi di ripetizione: si tratta ancora di REP, REPE, REPZ, REPNE e REPNZ, che possono essere applicati solo in parte alle istruzioni di stringa. In particolare alle istruzioni INSD, OUTSD, MOVSD, e STOSD (ed alle analoghe «ridotte», relative a byte ed a word, non dimentichiamocelo...) può essere applicato solo il prefisso REP (ripetizione semplice, senza controlli) in quanto sarebbe inutile effettuare una REPZ MOVSD dal momento che la MOVSD non setta flag da testare con la REP.

Ma mentre con l'8086 ed il 286 ciò poteva essere anche fatto (con scarsi risultati, però), nel caso del 386 ciò comporta addirittura la generazione di una «exception» di tipo «codice sconosciuto».

Per quanto concerne le restanti REPx (che a coppia non sono altro che sinonimi), c'è invece da dire che possono essere applicate solo alle istruzioni CMPSD e SCASD (nonché alle «ridotte»), pena la generazione di una «exception» analoga alla precedente: da questo punto di vista il 386 è nettamente e giustamente severo.

Ultima considerazione riguarda i tempi di esecuzione delle istruzioni di stringa: si va dai 5 cicli della LODS ai 7 della MOVS e SCAS, per arrivare ai 15 della INS: in tutti i casi indistintamente i tempi indicati sono indipendenti dal tipo dell'operando e cioè sia che si operi su un byte che su una double-word, il che è ancora una volta notevole.

Infine, per quanto riguarda le istruzioni precedute da un prefisso, abbiamo tempi abbastanza differenti da quelli visti, che vanno dai 5 ai 13 cicli di clock «base» ai quali vanno aggiunti da 5 a 9 cicli supplementari per ogni ripetizione: ad esempio la REP OUTS (anche in questo caso non importa su che tipo di operandi si lavora, byte, word o double-word che siano) richiede 12+5n cicli di clock, dove «n» è il numero di ripetizioni da compiere.

Le nuove istruzioni di manipolazione di bit

Eccoci finalmente ad un insieme di istruzioni create appositamente per l'80386: consentono di lavorare ancora una volta su byte, word o double-word, non visti nella loro interezza, ma semplicemente su di un singolo bit alla volta, bit specificato dal secondo operando.

Già si vede qui che non c'è più necessità di usare maschere di bit per poterne isolare uno: finora infatti, per testare se il bit 5 della cella ALFA (supponiamo che sia un byte) è resettato ed eventualmente saltare alla label THERE, dopo aver settato tale bit, bisognava scrivere un programmino del genere:

```
...
TEST ALFA, 00100000B; valore binario...
PUSHF
AND ALFA, 00100000B
POPF
JZ THERE
...
```

dove ancora bisognava ricordarsi se il salto dopo la TEST fosse sulla condizione di Zero o di Not-Zero, oltre al fatto di dover salvare lo stato del flag.

Evidentemente l'esempio di sopra poteva anche essere scritto così:

```
...
MOV AL,ALFA
AND ALFA,00100000B
TESTA AL,0010000B
JZ THERE
...
```

o in mille altri modi ancora...

Invece tale frammento di programma con il 386 diventa semplicemente

```
...
BTS ALFA,5
JNC THERE
...
```

e cioè prevede l'uso dell'istruzione BTS («Bit Test and Set»), la quale testa il bit 5 di ALFA (ponendo tale bit nel flag di carry) e successivamente lo setta, indipendentemente dal suo stato precedente.

Proprio con questa istruzione è possibile dunque implementare in un batter d'occhio un «semaforo» di accesso a risorse condivise da parte di un processo: in particolare si possono implementare le primitive di «domanda di passaggio attraverso un semaforo» e di «rilascio del semaforo» (comunemente indicate come «passlock» e «freelock» rispettivamente).

La prima fa sì che il processo che la esegue resti in un loop d'attesa per tutto il tempo che il semaforo è «rosso» (settato) per poi proseguire nell'esecuzione solo quando è diventato «verde» (resettato), ma con l'obbligo di settarlo subito per far sì che altri processi lo trovino «rosso».

Tale routine, supponendo che il semaforo interessato è ad esempio il bit 2 della cella di tipo word SEMAFORI (che perciò può contenerne fino a 16, o 32 se double-word...), può essere implementata con:

```
PASSLOCK:  BTS SEMAFORI,2
           JC PASSLOCK
           RET
```

In questo caso dunque viene posto nel carry il bit 2 di SEMAFORI e comunque viene posto ad 1: ciò è proprio quello che vogliamo, in quanto comunque dobbiamo porre ad 1 tale bit, specie se il processo raggiunge la sospirata RET.

Nel caso che si lavori in multiprogrammazione, con più processori contemporaneamente, allora si dovrà anche porre il prefisso LOCK davanti alla BTS, per far sì che durante la sua esecuzione il bus non venga ceduto ad un altro processore, magari per prendere anche lui il controllo della stessa risorsa.

Inutile dire che da qualche altra parte ci dovrà necessariamente essere un altro processo (od un altro processore) che, al termine di certe operazioni legate alla risorsa condivisa, sblocchi il semaforo (ponendo tale bit a 0) per permettere dunque al processo (o processore) in attesa l'accesso alla risorsa.

Evidentemente la seconda routine (la «freelock»), quella che sblocca il semaforo, sarà semplicemente formata da:

```
FREELOCK:  AND SEMAFORI, 11111011B
           RET
```

Passando dunque alla teoria, si hanno a disposizione quattro funzioni nuove: la BT («Bit Test»), la già vista BTS («Bit Test and Set»), la BTR («Bit Test and Reset») nonché la BTC («Bit Test and Complement»), le quali indistintamente pongono nel carry il bit indicato ed infine, rispettivamente, lasciano il bit inalterato, lo settano, lo resettano e lo complementano, avendo così a disposizione tutte le possibilità.

Abbiamo già visto che il bit può essere individuato in modo immediato (ad 8 bit) per mezzo del secondo operando: altra possibilità è indicare quale è il bit interessato per mezzo del valore posto in un registro a 16 o 32 bit.

Per quanto riguarda il primo operando, quello sul quale viene effettuata l'operazione di test, c'è da dire che innanzitutto può essere o un registro o una locazione di memoria a 16 o 32 bit.

Nel caso si tratti di un registro (a 16 o 32 bit), il numero del bit interessato (posto nel secondo operando come valo-

```
66 0F BA E0 03        bt    eax,3
66 0F BA EB 34        bts   ebx,34H
66 0F BA F1 00        btr   ecx,0
66 0F BA FE 01        btc   esi,1
66 0F BA FF DD        btc   edi,0DDH
```

Figura 1 - Alcuni esempi di istruzioni di test e manipolazione di bit relative a registri (estesi) e nelle quali il numero del bit da gestire è espresso da un valore immediato.

```
2E 66 0F BB 06 20 01     btc  cs:[0120H],eax
2E 0F BA 3E 20 01 21     btc  cs:word ptr [0120H],21H
```

Figura 2 - Questi sono altri due esempi dell'istruzione BTC applicata stavolta ad una word posta in memoria e della quale vogliamo testare e complementare il bit 21H (supponiamo dunque che EAX contenga tale valore): nel primo caso verrà testato e complementato il bit 1 della cella di indirizzo 124H, mentre nel secondo caso verrà testato e complementato il bit 1 della cella 120H: da notare in questo caso l'ampia estensione dei codici operativi, laddove il 26H è il prefisso di override di segmento (cs:) e 66H è il prefisso per i registri estesi (laddove si lavori in un segmento a 16 bit, non dimentichiamocelo).

re immediato oppure come contenuto di un registro), viene calcolato appunto dal secondo operando, ma rispettivamente modulo 16 e modulo 32, impedendo dunque la possibilità di testare il bit 43 del registro AX!

In figura 1 vediamo alcuni esempi di istruzioni di questo gruppo, assemblate e disassemblate con l'ormai indispensabile Turbo Debugger.

Se invece il primo operando è una locazione di memoria (word o doubleword che sia), si innesca un meccanismo differente, nel caso in cui il numero d'ordine del bit da testare è dato da un valore (posto nel secondo operando) eccedente rispettivamente i valori 15 e 31.

In particolare succede il fatto seguente:

— se il secondo operando è dato da un valore immediato, allora il numero del bit su cui operare è ottenuto ancora una volta modulo 15 e modulo 31;

— se il secondo operando è invece contenuto in un registro, piuttosto che effettuare l'operazione di modulo, il 386 si sposta automaticamente in avanti nella memoria di un certo numero di byte rispetto alla cella di memoria interessata ed opera su di un certo bit di questa nuova cella: ma vediamo meglio il meccanismo.

Supponiamo di aver posto in EAX il valore 32 e di voler eseguire l'istruzione BTC ALFA,EAX.

In questo caso il 386 opererà sul bit di posto 32 a partire dal bit meno significativo della cella (ad esempio di tipo word) ALFA: rapidi calcoli dicono che il bit 32 in questione è proprio il bit 0 della cella ALFA+4.

Infatti abbiamo che:

— il bit 0 del byte posto all'indirizzo ALFA+1 è proprio il bit 8 di ALFA inteso come word;

— il bit 0 del byte posto ad ALFA+2 può essere considerato proprio il bit 16 di ALFA;

— il bit 0 del byte di indirizzo ALFA+3 sarebbe il bit 24 di ALFA ed infine

— il bit 0 di ALFA+4 è proprio il bit 32 di ALFA.

In definitiva il valore posto nel registro (secondo operando) viene diviso per 8: il quoziente dice di quanti byte deve essere incrementato l'indirizzo iniziale, mentre il resto indica quale bit della cella così trovata deve essere analizzato.

In figura 2 vediamo invece due esempi di istruzioni di questo gruppo relative ad un primo operado posto in memoria.

A questo punto perciò bisogna stare attenti a come viene usato questo gruppo di nuove istruzioni, in quanto possono andare ad interessare celle di memoria al di là di quelle previste, solo però nel caso che il numero del bit sia posto in un registro.

```
66 0F BC C1     bsf   eax,ecx
66 0F BC DE     bsf   ebx,esi
```

Figura 3 - Qui vediamo due esempi dell'istruzione BSF: con il primo si analizza ECX ed il risultato (se ECX è diverso da 0, il numero d'ordine del primo bit posto ad 1 a partire da destra) viene posto in EAX. Analogamente si ha per il secondo esempio.

Altre due nuove istruzioni di analisi di bit

Si tratta delle due istruzioni BSF («Bit Scan Forward») e BSR («Bit Scan Reverse»), particolarmente utili in quanto colmano una lacuna nel set di istruzioni dei microprocessori precedenti: si tratta di due istruzioni che consentono in parole povere di calcolare subito qual è il primo bit settato (a partire da destra oppure da sinistra) in un certo registro o cella di memoria.

La sintassi delle due istruzioni è la seguente:

BSF reg16,op16
BSF reg32,op32
BSR reg16,op16
BSR reg32,op32

dove con reg16 e reg32 si indicano rispettivamente un registro a 16 e a 32 bit, mentre con op16 e op32 si intendono o registri o celle di memoria, a 16 e a 32 bit.

L'istruzione BSF fa sì che il 386 scandisca op16 o op32 a partire dal bit meno significativo per fermarsi laddove incontra un eventuale bit settato: a questo punto pone in reg16 o reg32 il numero d'ordine del bit trovato ad 1 e contemporaneamente resetta il flag di Zero (condizione «NZ»).

Se viceversa l'operando a 16 o 32 bit su cui opera fosse direttamente nullo allora non viene fatta alcuna scansione ed il flag di Zero viene direttamente settato (condizione «Z»), lasciando però indefinito il registro che dovrebbe contenere il numero del primo bit settato.

Se ad esempio eseguiamo le seguenti istruzioni

MOV EBX,8000H
BSF AX,EBX

dove cioè abbiamo settato il bit più significativo di EBX, a seguito della BSF verrà posto il valore 31 in AX: infatti il primo bit ad 1 di EBX è proprio quello di ordine 31.

In figura 3 possiamo vedere altri due esempi di tale funzione, che possono essere testati al volo con il solito Turbo Debugger.

Per quanto riguarda l'altra istruzione, la BSR, vale tutto quanto detto per la BSF, salvo che ora la scansione avviene a partire da sinistra e cioè dal bit più significativo, ma comunque il valore che si ottiene nel registro (primo operando) è il numero d'ordine del primo bit posto ad 1 e trovato da sinistra verso destra.

Per vedere dunque la differenza tra le due istruzioni in esame, supponiamo che il registro SI contenga il valore 2001H: a seguito delle due istruzioni

BSF EAX,SI
BSR BP,ESI

si otterrà in EAX il valore 1 (bit 1 è il primo che incontra da destra verso sinistra), mentre in BP si troverà il valore 13, dal momento che è proprio il bit 13 del registro esaminato il primo bit posto ad 1.

Tra l'altro c'è da notare, nel secondo esempio, che noi andiamo a testare tutto ESI invece di SI soltanto, dal momento che ciò è perfettamente lecito e non comporta problemi di sorta (al pari di usare indifferentemente AL, AH ed AX in istruzioni successive...): ricordiamo inoltre che SI è proprio la parte meno significativa del registro ESI.

Un ripensamento...

Prima di terminare la puntata citiamo un fatto strano: sul data book del 386 (laddove in ogni pagina c'è riportata la dicitura «Advanced Information»), all'interno della tabella delle istruzioni di tale microprocessore e nell'ambito delle istruzioni di «Bit Manipulation» si trovano due istruzioni «fantasma» che non trovano infatti riscontro nella letteratura più recente.

In particolare, per la cronaca, si tratta della IBTS («Insert Bit String») e della XBTS («eXtract BiT String»), delle quali vengono dati addirittura i codici operativi (che iniziano rispettivamente con 0FH A7H e 0FH A6H, seguiti da altri byte legati all'indirizzamento degli operandi), nonché i tempi di esecuzione (rispettivamente 12 cicli per operandi di tipo registro e 19 per memoria e 6 cicli per registri e 13 per memoria): di tali istruzioni non si usa più nulla, né tantomeno i codici operativi vengono riconosciuti dal Turbo Debugger.

Probabilmente sono state eliminate dal set di istruzioni del 386 magari per malfunzionamenti: tra l'altro il microprocessore, tentando di eseguire gli opcode di cui sopra, esprime il suo disappunto generando un'exception di tipo «unknown opcode».

Detto dunque di queste due istruzioni che non esistono più, diamo l'appuntamento alla prossima puntata, laddove parleremo delle istruzioni di trasferimento del controllo. MC

ELETTRONICA CENTOSTELLE s.r.l.

ZENITH	Lap top
TANDON	Desk top
ASEM	Desk top
NEC	Stampanti

Via Centostelle, 5/a - Firenze - Telefono (055) 61.02.51 - 60.81.07 - Fax 61.13.02

SOFTWARE

WORD PROCESSOR

Microsoft Word 5	it. L.	712.000
Microsoft Word 5 euro	it. L.	570.000
MicroPro Wordstar Prof. 4.0	it L.	590.000
MicroPro Wordstar prof. 5.0	in L.	590.000
MicroPro Wordstar 2000 3.0	it L.	880.000
Lotus Manuscript 2.0	in L.	730.000
AshtonTate Multimate adv. II	it L.	785.000
Borland Sprint	in L.	330.000
Word Perfect	it L.	960.000

SPREADSHEET INTEGRATI

Microsoft Excel	it L.	712.000
Microsoft Excel Euro	in L.	640.000
Microsoft Works	it L.	280.000
Lotus 1 2 3	it L.	695.000
Lotus Symphony	it L.	890.000
Ashton Tate Framework III	it. L.	952.000
Borland Quattro	it L.	330.000
Computer Ass. Supercalc 5	it L.	800.000

DATA BASE MANAGEMENT

AshtonTate dBase III plus	it L.	880.000
AshtonTate dBase IV	it L.	1.080.000
AshtonTate Rapid file	it L.	560.000
Borland Paradox	it L.	1.080.000
Borland Paradox (os/2)	it L.	1.400.000
Borland Paradox 386	it L.	1.400.000
Borland Reflex 2.0	it L.	340.000

GRAPHICS

Microsoft Chart 2	it L.	390.000
Microsoft Chart 3 euro	in L.	540.000
Lotus Freelance V.2.01	it L.	650.000
Auto cad 10.0 (scuole università)	it L.	950.000
Paintbrush plus (per Wind.)	in L.	240.000
Gem Artline	in L.	1.350.000
Gem desktop publishers	in L.	650.000
Lotus GraphWriter II	it L.	723.000
Adobe Illustrator	in L.	1.390.000

DESKTOP PUBLISHING

Ventura Publisher	it L.	1.480.000
Fonts Bitstream	it L.	550.000
AshtonTate Byline	it L.	472.000

AMBIENTI OPERATIVI

Microsoft Project 3.0	it L.	760.000
Microsoft Project 4 Euro	in L.	680.000
Microsoft Windows 286	it L.	180.000
Microsoft Windows 386	it L.	280.000
Microsoft Windows 286 toolkit	in L.	680.000
Lotus Agenda	in L.	650.000

LINGUAGGI

Microsoft Quick basic 4.5	in L.	145.000
Microsoft Quick C compiler	in L.	145.000
Microsoft Basic Compiler 6.0	in L.	380.000
Microsoft C Compiler 5.1	in L.	590.000
Microsoft Fortran Compiler	in L.	630.000
Microsoft Cobol Compiler V3	in L.	1.100.000
Microsoft Macro Assembler	in L.	240.000
Microsoft Pascal Compiler	in L.	550.000
Microsoft OS/2 toolkit	in L.	480.000
Borland turbo Pascal 5.5	it. L.	240.000
Borland turbo basic	it L.	170.000
Borland turbo C 2.0	it L.	240.000
Borland turbo Prolog. 2.0	it L.	190.000
Borland turbo Assembler/debug	it L.	190.000
Borland turbo C professional	it L.	399.000
Borland turbo Pascal Profess.	it L.	399.000
RM Cobol 85	in L.	2.765.000
RM Cobol Compiler	in L.	1.660.000
RM Fortran	in L.	1.405.000

UTILITIES

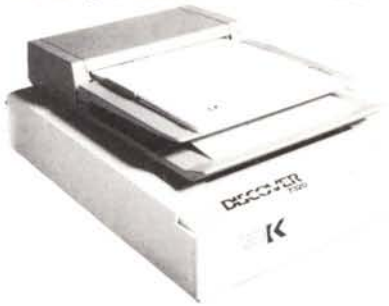
Norton Utilities	in L.	170.000
Norton Commander	in L.	170.000
PC Tools 5.5	in L.	150.000

HARDWARE

Hard Disk Nec 5" 1/4 20 Mb + control	L.	566.000
Hard Disk Seagate 5" 1/4 40Mb + control.	L.	857.000
Copr. Metem. 8087/5 4.77 MHz	L.	233.000
Copr. Matem. 80287/8 8MHz	L.	541.000
Copr. Matem. 80287/10 10MHz	L.	621.000
Copr. Matem. 80387/16 16MHz	L.	932.000
Copr. Matem. 80387/20 20MHz	L.	1.060.000
Copr. Matem. 80387/25 25MHz	L.	1.338.000
Modem int. Hyundai V 21:22	L.	119.000
Modem est	L.	189.000
Monitor colori 14" Ega	L.	748.000
Monitor colori Multisink NEC	L.	1.193.000
Monitor Multisink fosfori bianchi	L.	312.000
Scheda grafica ris. VGA	L.	490.000
Espansioni RAM		Telefonare
Compact Disk Hitachi		Telefonare
Stampante portatile Toshiba	L.	699.000
Stampante 24 aghi 80 col	L.	796.000
Fax Toshiba T 211		Telefonare
Scanner IOR		Telefonare

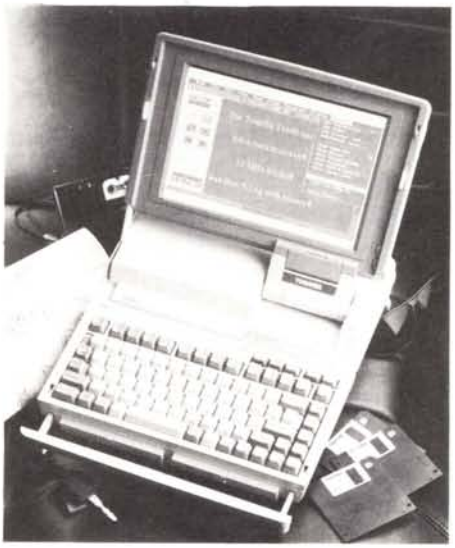
NOVITÀ

Microsoft Quick basic 4.5	it L.	195.000
Microsoft Quick pascal 1.0	in L.	155.000
Microsoft Quick MASM/	in L.	225.000



Concessionario TOSHIBA

Per lo studente
Toshiba T 1000
Stampante NEC 24 aghi P 2200
Software WORKS (Microsoft)
L. 2.195.000



Per le università, scuole e istituti
Toshiba T 1600
Coprocessore matematico
Autocad 10.0
L. 6.580.000

Per il professionista
Toshiba T 1200 con HD 20Mb
Stampante STAR 24 aghi 80 col.
Programma di videoscrittura
WORD (Microsoft) in italiano
L. 4.944.000

Per il manager
T 5200 HD 100 Mb
Corredato di un programma Microsoft ideale per chi deve gestire quotidianamente grosse quantità di numeri, creare tabelle e grafici, sviluppare budget, acquisire e consolidare informazioni ecc.
L. 11.950.000

Tutti i prezzi sono IVA esclusa
Pagamento in contrassegno, vaglia o VISA
Per ordini inferiori a L. 500.000
aggiungere spese postali L. 10.000

ORDINI
a mezzo telefono
Fax
Posta

Consulenza telefonica
gratuita su tutta la
nostra gamma di prodotti
Inserimentio automatico dei nostri clienti
nel servizio Direct Marketing

SI RACCOMANDA AL MOMENTO DELL'ORDINE DI CHIEDERE CONFERMA DEI PREZZI