di Bruno Rosati

Speciale linguaggi: dal LISP al Prolog

Termina con questo articolo il breve ciclo dedicato alla collana AcornSoft relativa ai package di programmazione. Dopo aver presentato l'Assembler, il Twin, l'ANSI-C ed il Fortran 77, in questa quarta ed ultima parte chiuderemo il cerchio andando ad analizzare le caratteristiche, relative alle implementazioni curate dalla Acorn, di due High Level che più vanno per la maggiore. Lisp e Prolog, ovvero l'Intelligenza Artificiale che dopo gli anni delle ricerche universitarie, attraverso la commercializzazione dei linguaggi appositamente studiati, approda nei disk drive dei nostri personal

Lisp (SKL 14) / Prolog (SKL 68)

Produttore:
Acorn Computer Ltd. Fulbourn Road, Cherry Hinton, Cambridge, CB1 4JN, UK

Distributore per l'Italia:
G. Ricordi & C. S.p.A.
Via Salomone 77 - 20138 Milano
Prezzi *(IVA esclusa):*

Lisp (SKL 14)	*L. 424.000*
Prolog (SKL 68)	*L. 424.000*

LISP

Aperta la confezione, sul volantino di accompagno, il *leaflet* per gli anglofili, oltre alle note sulle modalità di backup e l'uso dell'FPE240, troviamo immediata conferma che a lire 424.000 corrispondono «solo» un ADFS-disk ed un manuale di 150 pagine.

È evidente che il «peso specifico» del Lisp Processing sviluppato dalla AcornSoft è di livello assoluto. Ce ne rendiamo benissimo conto, andando a leggere la locandina posta sul recto della confezione e le note introduttive delle prime pagine: un LISP-editor a pieno schermo, velocissime (ed ottimizzate) routine di stampa, precisione aritmetica di tipo arbitrario; un compilatore «di serie»; un graditissimo package aritmetico per scrivere sistemi algebrici avanzati — con integer di qualsiasi taglia e primitive aritmetiche predisposte e ai numeri razionali e a quelli in virgola mobile. Il tutto, condito dalla iper-velocità del RISC. Sono queste le principali caratteristiche dell'Acorn-LISP; una implementazione pienamente compatibile con quel Cambridge-LISP, sviluppato dall'Università dello Utah, in congiunzione con la Rand corporation, quale proposta di standardizzazione.

Due note per chi ha fin qui usato altri dialetti *lispiani* ed eccoci a dare il run al sistema:

```
LISP - image m image filename
```

dove la parola chiave «image» dovrà esser inserita con la specifica del pathname relativo alla directory che contiene i file-image. Gradite le opzioni di Start-up. L'AcornLISP ne dispone in numero di otto (fig. 1). Dalla **-from** e **-to** per l'input e l'output del LISP, alla **-image** stessa — appena vista nella sintassi del run — con la quale si specifica il nome di una directory nella quale cercare.

Quindi la **-dump,** con la quale, specificato il nome della directory, ci si dispone ad inserirvi sia l'image che i moduli (di fast load) generati. Con l'opzione **-leave** si concede ad Archie l'uso di un certo numero di byte (specificati come unità da 1024 byte l'una) quale spazio di lavoro. L'opzione **-store** a sua volta aspetta un certo numero di byte (sempre in unità da 1024) che il LISP userà per girare. Con **-identify** invece si provvederà ad abilitare alcune linee di informazioni iniziali (quali il numero della versione dell'implementazione, l'ammontare dello spazio usato, il taglio dell'immagine, la data di creazione della store-image, etc.). L'opzione **-help** infine, rende una breve descrizione di tutte le opzioni disponibili.

LISP Editor

L'editor dell'AcornLISP è una potentissima base di lavoro utilizzabile al meglio delle caratteristiche relative all'implementazione. I suoi comandi sono concepiti intorno al concetto della «s-expression», intesa come un **atom** oppure una lista di partenza a parentesi aperte. L'editor può essere invocato attraverso due funzioni predisposte: **sedit** e **fedit**. Dove con **sedit** ci si predispone all'edit della s-expression, mentre con **fedit** si passa a definire la funzione.

Si diceva dei comandi. Se guardate la figura 2, potete prendere completa visione del set messo a nostra disposizione. Si tratta di 22 comandi tipicamente ripartiti in 5 raggruppamenti. Il primo,

Opzioni di Star-up

```
-from       -leave
-to         -store
-image      -identify
-dump       -help
```

Figura 1

Lisp Editor: i comandi

```
Elementary Moving:

h (muove all'head della s-expression)
t (muove alla relativa coda)
u (muove ad un livello diverso)
b (muove all'inizio di una lista inclusa)

Find & Move:

lf (muove alla successiva "occurence" in ordine di stampa)
lk (come lf, ma in ordine di stampa INVERSO)
l  (ripete l'ultimo comando)
ma (inserisce un marcatore alla posizione di edit)
mo (muove al precedente marcatore inserito)

Structure Modification:

d  (Cancella la s-expression)
r  (Rimpiazza la s-expression con un'altra versione)
c  (Ripete l'ultima modifica)
i  (Inserisce una s-expression nella posizione del pointer)
s  (Splice in una lista specificata al pointer)
n  (funzione di "UNDO" sull'ultimo comando eseguito)
x  ( "Esplode" la s-expression per effettuare modifiche)

Reformatting Screen:

?  (Inizializza lo schermo)
zi (Zoom IN nella s-expression per vederne i dettagli)
zo (Zoom OUT)

Eval loop:
e  (introduce un loop "read-eval-print")
w  (Carica e ridata la struttura editata)
q  (Quit senza ridatazione)
```

Figura 2

Elementary Move (h, t, u, b), è il quartetto dedicato al movimento condizionato del cursore. Il gruppo che segue, detto *Find and Move*, si compone di 5 comandi predisposti, oltre che allo spostamento, anche alla ricerca di strutture particolari. Gli *Structure Modification* (d, r, c, i, s, n, x) sono il gruppo più folto. Sette comandi predisposti alle operazioni di modifica alle strutture. Sotto il nome *Reformatting Screen* invece, vengono concentrati tre soli comandi utili per la inizializzazione dello schermo e la funzione «zoom» di controllo. Gli ultimi tre comandi (e, w, q) sono infine predisposti all'introduzione di loop «read-eval-print» e quit-are dall'editor.

Input & Output

Basata sul concetto degli stream selezionabili, l'intera struttura dell'Input/Output dell'AcornLISP, possiamo (grossolanamente) esemplificarla attraverso la funzione di due stream; uno, il **-from** che si apre al run per gestire tutti gli input; l'altro, il **-to** che viene selezionato al momento dell'output.

Certamente le cose sono un pochino più complesse (fortunatamente anche più complete...) e nella funzionalità dell'Input/Output sono da mettere in conto le routine di gestione. Ovvero: le **open/close** che aprono o chiudono uno stream, e le **wrs/rds** che possono leggere e scrivere, influenzando così sia l'input che l'output.

Ovviamente la routine di principale importanza sarà la open che, oltre ad aprire il flusso, una volta ottenuto un valore, lo metterà a disposizione, sottoforma di file-handler delle altre tre routine per la lettura, la scrittura e la relativa chiusura dello stream modificato.

Altro raggruppamento interessante delle I/O è quello delle routine di stampa. Nell'AcornLISP trovano posto quattro, differenti stili di scrittura. Il primo raggruppamento (comandi preposti: **prin**, **print** e **printm**) produce la stampa di un atom o di una lista: il secondo (tramite i comandi: **pric**, **printc** e **printcm**) gestisce la stampa quando il programma sta ancora provando a produrre un output non LISP-like. Ovvero: stampando atom e stringhe senza nessun marcatore. Il terzo gruppo (**prinl**, **printl** e **printcl**) è da utilizzare per produrre strutture loop-ate. L'ultima routine, la **superprint**, ci fornirà, dei programmi sviluppati sotto LISP, una stampa provvista dello stile di indentazione.

Funzioni e variabili

La lunga lista (e la relativa illustrazione delle caratteristiche) delle funzioni e le variabili dell'AcornLISP, a cui la guida di riferimento giustamente dedica lo spazio maggiore, apre la seconda parte del manuale ed in pratica chiude questa nostra veloce carrellata. Si tratta, nel caso specifico, di una sfilza di oltre 300 funzioni, di cui viene dapprima spiegata la particolare struttura (parametri, tipi di argomento e caratteri di notazione)

Lista delle funzioni fissate dal compilatore

```
apply          car          cdr
cdifference    cminus       cons
cplus          ctimes       equal
flagp          gensym       get
getd           getv         iadd1
idifference    ileftshift   ilogand2
ilogor2        ilogxor2     iminus
iplus2         irghtshift   isub1
itimes2        ncons        putv
rplaca         rclacd       rplacw
xcons
```

Figura 3

quindi illustrata la loro specifica ripartizione nei vari raggruppamenti funzionali. Ad esempio le **Dotted-Pair** — prodotto di funzioni di classe cons — sono da intendersi come il tipo primitivo di data riconoscibili dal LISP. Seguono poi il raggruppamento delle funzioni per il **List-Processing** e quelle per il **List Equality And Searching**; quindi le funzioni per il **Pointer Replacement** e quelle per il **List Manipulating,** la creazione di simboli, flag, etc. Un raggruppamento interessante sono poi le **Arithmetic.** Operazioni che l'AcornLISP supporta per quattro differenti classi di atom numerici: piccoli integer (compresi nel range fra −2**24 e +2**24); integer di qualsiasi taglia, numeri in virgola mobile e numeri razionali. Da rilevare è che l'AcornLISP oltre a supportare tutte le usuali operazioni possibili su queste funzioni, permette anche la realizzazione di moduli aritmetici sulla serie dei piccoli integer.

In un ultimo capitoletto a fine guida, troviamo anche una miscellanea di funzioni, fra cui il raggruppamento delle **Graphics**. Routine per il cui uso si deve disporre di un graphics screen mode già invocato con una chiamata sintatticamente del tipo: **mode n**, dove n può essere uguale a 0, 1 e 2. Mode 0 allora provvederà a fornirci il classico 640x256 monocromatico; il Mode 1 il 320x256 a 4 colori ed il Mode 2 un 160x256 a 16 colori. Le funzioni grafiche supportano due differenti stili grafici: il classico cartesiano — (**moveto x y**), (**draw x y**) — ed il turtle graphics — (**turn n**), (**turnto n**) e (**move l**), (**draw l**) —. Lasciandovi snocciolare per vostro conto il resto delle particolarità delle funzioni spostiamoci ora sul raggruppamento delle variabili.

Task eseguibili dall'Acorn-Prolog

```
1) input/output

2) reading in programs.............[compile(F); consult(F)]

3) opening and closing files.......[see(F); seeing(F); seen;
                                    tell(F); telling(F); told(F)]

4) reading and writing Prolog terms.....[read(X); read(X,Y);
                                    write(X); display(X); writeq(X)]

5) getting and putting char.......[get0(N); get(N); skip(N);
                                                  put(N); tab(N)]

6) arithmetic...........[(X=:=Y); (X=\=Y); (X < Y); (X > Y);
                                    (X =<Y); (X>=Y); succ(M,N)]

7) affecting the flow of the execution.........[fair; repeat;
                                                  forall(G,T)

8) classifying and operating on Prolog terms.........[var(X);
                    nonvar(X); atom(X); integer(X); atomic(X);
                functor(T,F,N); arg(I,F,X); X=..Y; name(X,L);
                                    call(X); numbervars(T,M,N)]

9) processing sets..............[setof(X,P,S); bagof(X,P,B);
                                                findall(X,P,L)]

10) comparing terms...............[compare(C,X,Y); sort(L,S);
                                                keysort(L,S)]

11) manipulating the Prolog program db.............[assert(C);
          asserta(C); assertz(C); clause(P,Q); clause(P,Q,R);
                    retract(T); abolish(F,N); listing(F,A)]

12) manipulating the internal indexed db....[recorded(K,T,R);
                    recorda(K,T,R); recordz(K,T,R); erase(R)]

13) interacting with the prog.enviroment....[writedepth(X,Y);
                    writewidth(X,Y); unknown(X,Y); op(P,T,N);
                    break_handler; error_handler(N,X); abort;
                        save(F); statistics(X,Y); system(X)]

14) defining modules................[module(X); endmodule(X);
                                    visa(A,F); import(F,M)]
```

Figura 4

L'argomento «variabili» non viene trattato in una sezione unica ed esaustiva; invero, evitata tale schematizzazione, i vari tipi a disposizione vengono illustrati nel contesto degli argomenti principali. Vedasi nel caso del LISP-editor, laddove, per quanto riguarda il settaggio durante una operazione di «cut and paste», s'introduce il concetto delle **Hash**, oppure delle quattro di tipo ****edit** (ovvero: **-functions**, **-last-function**, **-level** e **-menu** che sono variabili Globali). A conti fatti è il solo gruppo delle **Specialised** che ritroviamo nel capitolo dedicato e ciò non fa altro che confermare l'estrema importanza di una biblioteca LISP-dedicated ben fornita.

Le Specialised comunque sono delle variabili che vedono preceduto il nome dal simbolo «*» come dal «&» e che risultando ad «uso-interno» non possono essere modificate. Si tratta di sei variabili a più frequente utilizzo predisposte ad una condizione — *true* — di verifica che, se rispettata, permette l'esecuzione del determinato tipo di operazione alla quale le variabili sono preposte. Le Specialised a più frequente uso sono: ***comp**, ***pgen**, ***lower**, ***raise**, ***savedef**, ***plap**. A *true* verificato, la ***comp** vedrà compilate le definizioni della funzione; la ***pgen** un assembly listing (anche se la forma non sarà compatibile con l'Assembler standard); la ***lower,** la traduzione equivalente dei caratteri dai più alti allocamenti a quelli più bassi (con la variabile ***raise** a rappresentare una sua variante con effetto forzante verso i *case* più alti). Per quanto poi riguarda la ***savedef,** sempre a *true* verificata e con il LISP-compiler a rimpiazzare una funzione con il codice macchina, essa provvederà a salvare la forma originale della s-expression nella lista di appartenenza. Con l'ultima variabile, la ***plap**, il compilatore genererà una forma di macro intermedia che, se ci sarà il *true*, verrà anche stampata.

Ultime note sull'AcornLISP

Quante cose ci sarebbero ancora da dire! Parlare delle caratteristiche acorniane delle funzioni di tipo «load-on-call» — meglio conosciute come FASL — quali facility di caricamento; quindi delle routine di lettura (**read**, **readch**, **read-token** e **tyi**); del sistema per il file handling, con il LISP che seleziona un file per l'Input ed uno per l'Output con la relativa sequela di funzioni preposte alla selezione di nuovi stream; quindi il controllo degli errori ed il tipo di debug possibile nell'AcornLISP (con le relative funzioni di tracing, timing e d'interfaccia all'OS) e le solite due chiacchiere sui

messaggi di errore. Invece è tempo di chiudere, tirare una riga e fare due conti, lasciando poi a voi, *lispiani* della prima ora di correre da Archie e provare l'ebbrezza della implementazione più veloce del mondo. Dalle vostre propensioni verso il campo della ricerca sulle possibilità dell'Intelligenza Artificiale a quelle dei sistemi algebrici... la sfida che l'implementazione acorniana porta, è sinceramente vincente. Il LISP c'è tutto (ed anche qualcosa di più...) e quella Risc-mania del poterci correre da matti, credo proprio che sia *l'ultima tentazione!*

Acorn-Prolog: caratteristiche generali

Le caratteristiche principali dell'Acorn-Prolog sono rilevabili nella compatibilità con la sintassi Edinburgh; la velocità di compilazione e nella riduzione della memoria richiesta; la presenza di un Debugger interpretativo; disponibilità delle regole grammaticali nella programmazione e nel sovraccaricamento delle dichiarazioni e delle parentesi. Dulcis in fundo: il rilevamento automatico degli errori.

L'Acorn-Prolog è un sistema interattivo dotato di un compilatore ad incremento, un sistema di run-time ed una notevole quantità di procedure; il tutto rigidamente compatibile con il DEC system-10 Prolog e suoi discedenti. Di tutto ciò Archimedes se ne fa più di un vanto essendo difatti uno dei pochissimi personal computer system per il quale è stata realizzata un'implementazione Prolog!

Le modalità di lavoro dell'Acorn-Prolog si basano sull'utilizzo dei file-using. Ovvero, nel redigere il testo di un programma in Prolog — preferibilmente nel Twin —, si verranno a creare un certo numero di file di utilizzo che, attraverso un opportuno modo di istruzione, saranno letti dal sistema. Tale meccanismo lavorativo potrà essere effettuato in due modalità: per consultazione e per compilazione. Quando si usa «consultare» un file, l'informazione inerente la clausola necessaria al programma debug-ato, è salvata unitamente alla forma compilata della clausola stessa.

A questo punto è nostro uso simulare il run di ogni linguaggio. Un po' per filo conduttore, un po' per ripassarne insieme la sintassi. Stavolta, così come fa il manuale, debbo far precedere la cosa da una curiosa raccomandazione: siccome il Prolog lavora preferibilmente con le minuscole e siccome Archie al momento del boot è generalmente in modo maiuscolo, vi ricordo di disinserire la funzione CapsLock. Fatto ciò, appena digitato:

```
prologx zap-out <arguments>
```

potrete vedere il Prolog produrre un prompt simile al seguente:

```
Memori areas: 2B068, 48528,
      52168, 79268
Heap allocated at 669b0
Prolog-x, release 4.011
?-
```

facendolo precedere dalla sua «bandiera» e da una pausa di attesa durante la quale caricherà se stesso. Da questo momento avremo il Prolog in mano. Le «chiavi del potere» saranno il tasto del «return» che termina l'Input di una linea, la sequenza **end_of_file** per marcare la fine di un Input ed il tasto **ESC** con il quale si potrà interrompere l'esecuzione del programma.

Predicati specifici dell'Acorn-Prolog

```
break_handler
endmodule(A)
error_handler(N,T)
read(T,L)
statistics(X,Y)
system(L)
visa(L)
writewidth(O,N)
```

Figura 5

Predicati relazionali al compilatore

```
arg(N,T,A)
atom(T)
atomic(T)
call(P)
fail
functor(T,F,N)
integer(T)
Y is X
nonvar(T)
var(T)
!
P -> Q ; R
P -> Q
X < Y
X =< Y
X > Y
X >= Y
X =:= Y
X =\= Y
X = Y
```

Figura 6

Caratteristiche specifiche

Come già accennato nelle note introduttive, l'Acorn-Prolog è compatibile con la sintassi Edinburgh, verosimilmente riconducibile, per dettami e caratteristiche implementative, al «sacro» testo *Programming in Prolog.*

Detto ciò comunque, non ci garantiamo certo l'univocità dei sistemi. Invero, l'Acorn-Prolog, rispetto agli altri dialetti, ha la sua brava dote di eccezioni. Alcune sono sufficientemente rilevanti da essere anticipate per tutti coloro che, già *prologando* su altri sistemi, si preparano a salire in sella ad Archie.

Anzitutto l'Acorn-Prolog ci consente Input sia in upper-case che in lowercase. Da ciò ne consegue l'assoluta inutilizzazione di device sintattici quali: **,..., LC, NOLC** e la forma ','. Quindi la importantissima catalogazione del set dei caratteri, sempre di 256 unità rappresentate dai codici degli integer compresi nel range da 0 a 255. Al solito i codici da 0 a 127 sono gli ASCII, mentre le forme comprese fra 128 e 255 non sono definiti, con l'Acorn-Prolog a trattarli come simboli (sign).

Nell'Acorn-Prolog sono inseriti a sistema alcuni predicati fra quelli di più comune utilizzo, prelevati da varie Prolog-library; altri sono prettamente «acorniani». Per maggiori ragguagli a riguardo osservate le figure 5 e 6 dove vengono schematizzati due raggruppamenti di *Predicates.*

All'Acorn-Prolog è tra l'altro garantito un top-level. Come le clausole vengono compilate, la clausola-sorgente verrà inserita nel database insieme ai nomi delle variabili usate dentro la clausola stessa. Ciò può essere usato per listare clausole (listing) esattamente come esse vengono digitate. Tuttavia l'utente può compilare le clausole senza inserirne il sorgente nel database. Il vantaggio che ne deriva è evidente: maggiore velocità di compilazione e minor occupazione di memoria. Unico «contro»: le clausole non saranno più accessibili per nessuna chiamata da sistema del tipo: **clause**, **listing**, **retract**.

Brutte (o buone) notizie a riguardo delle facility a livello di debugging che l'Acorn-Prolog non contiene. Se ciò a livello di compilazione ci rende ulteriori incrementi di performance, il rovescio della medaglia è sempre quello legato al riconoscimento delle clausole, eseguite in tali condizioni, in modo criptico rispetto alle aspettative del programmatore. In tal senso giunge in aiuto il debugger interpretativo che l'Acorn-Prolog possiede come library. Attraverso l'*interpretativo* verrà imposta una strategia di esecuzione di tipo convenzionale. Stop!

A questo punto, in mezzo a tanto ciarlare, uno potrebbe giustamente chiedersi: ma allora, quali sono le *grandezze* effettive dell'Acorn-Prolog?

Per prima cosa richiamerei l'attenzione sulla larghezza dello spazio d'indirizzamento: una cosa come 2^{28} byte! Quindi rilevare che diversi predicati sono di tipo a codice-aperto e che l'indice delle clausole è automatizzato sul primo argomento dei goal. Poi, saltando qualche altra cosetta, introdurre il discorso sulle restrizioni delle strutture dei dati. Con gli Integer compresi in un range da −134217728 a +134217727; le stringhe equiparate a liste di caratteri e gli atom, i cui nomi possono essere rappresentati dai byte «impacchettati» di stringhe dalla lunghezza limite di 1677215 caratteri.

Di variabili, in una clausola, ne possono essere utilizzate al massimo 255; mentre, per quanto riguarda i termini composti, è confermato che solo quelli aventi l'*arity* inferiore a 256 possono essere compilati.

Per quanto poi riguarda la gestione degli errori, l'Acorn-Prolog consente l'uso di una procedura dedicata, detta **error_handler,** la cui azione è quella di produrre un messaggio ed un *culprit*; l'atto di accusa.

Ed ora con un balzo all'indietro rispetto al tipo di trattazione che fa la guida di riferimento, vorrei soffermarmi un attimo sull'insieme dei predicati Prologhiani della nostra implementazione. Più sopra abbiamo accennato a due diverse categorie che, unitamente al resto della lista dei predicati di sistema, sono a nostra disposizione. Sto parlando del raggruppamento (fig. 5) dei predicati specifici della implementazione acorniana e di quelli (fig. 6) a codice-aperto che, su effetto del compilatore, emettono una specifica istruzione a livello di codice-macchina. Ebbene, nella figura 7 avrete poi senz'altro letto un forse troppo generico *Prolog-library*; si tratta, invero di solo cinque predicati facenti parti di una libreria di Prolog che è stata inserita nell'implementazione archimediana. Tutte e tre le categorie, così ripartite solo per una visione più chiara, sono infine concentrabili nella più grande tabella dei *Predicates Exported* che il manuale elenca da pagina 58. Sfogliando le quattro pagine dedicate alla lista, sarà possibile notare che, insieme alle liste estrapolate nelle figure sopracitate, figurano difatti tutti gli altri predicati disponibili da sistema.

Al riguardo la figura 4 espone tutta una serie di task che saranno eseguibili utilizzando degli specifici *Predicates* che, a differenza degli altri non possono essere ridefiniti dall'utente. Questi *Predicates* ho provveduto ad inserirli, fra parentesi, accanto alla particolare task a cui sono dedicati.

Prolog - library

```
findall(T,G,L)
forall(P,Q)
length(L,M)
simpleterm(T)
succ(M,N)
```

Figura 7

Espressioni aritmetiche

```
+X ........(addizione unitaria)
-X ........(sottrazione unitaria)
X + Y .....(addizione)
X - Y .....(sottrazione)
X * Y .....(moltiplicazione)
X / Y .....(divisione)
X mod Y....(residuo)
X / \ Y....(congiunzione di bit)
X \ / Y....(disgiunzione di bit)
X << Y.....(bit shiftato a sinistra)
X >> Y.....(bit shiftato a destra)
\ X .......(negazione)
cputime....(calcola il tempo in millisecondi dallo start)
calls......(numero delle procedure di chiamata dallo start)
an integer.(valore di ogni integer)
a  list....(il primo  elemento di  una  lista è  considerato
            come una espressione)
```

Figura 8

Conclusioni

Non ci trovo alcun male né la pur minima reticenza, nel concludere questa seconda parte dell'articolo, con una confessione.

Il sottoscritto è un vecchio *basic-ano* con naturale tendenza *Ci-ista* (tutta colpa di Amiga...) e ai primi approcci amorosi con il *LISP* (tutta colpa di Archie...). Delle tre linguacce credo di tenere in testa un gran minestrone informatico dal quale, bene o male, ho attinto vecchi sapori che miscelati a nuovi aromi ho portato in tavola per servirvi queste pietanze mensili. Credetemi: la cosa mi ha sconvolto a tal punto da farmi tentare anche con il Prolog, ma fortunatamente... non ce l'ho fatta! limitandomi alla solita, doverosa infarinatura.

Cosa questa che ho fatto sufficientemente convinto di rendervi l'informazione dovuta nella sua forma più corretta. D'altronde il piano di lavoro che mi ero riproposto — Speciale Linguaggi — aveva del titanico. Il fatto di averlo ridotto a sola informazione (logico che gli spazi e i limiti della rubrica imponevano tagli estremamente «leggeri» agli argomenti) è stata una scelta che dalla presentazione dell'Assembler, per finire con questa del Prolog, doveva servire solo a render note le caratteristiche dei vari linguaggi e di conseguenza le possibilità di Archie.

Alle solite, sottintendendo le piccole critiche di sempre, ciò va ad ulteriore elogio della Acorn, forsennatamente tesa a rendere al suo RISC i massimi servigi.

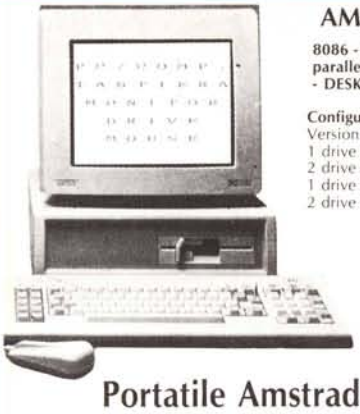
Manca, è vero, sempre un qualcosa che forse solo le grandi software-house tipo Borland potrebbero dare.

La tecnologia RISC è ancora troppo sola ed alla Acorn non si può certo chiedere di cantare e portare la croce. Quello che fa e produce è sempre di buon livello, per farlo diventare eccelso (e renderci di conseguenza il top delle caratteristiche congiunte: possibilità-macchina/qualità-software) non ci sarà certo bisogno di scippare le menti migliori alle altre software-house. Più semplicemente: commissionare o, meno dispendiosamente ancora, «convincere» a produrre per Archie! MC

MASTERBIT

VENDITA PER CORRISPONDENZA

MASTERBIT Viale dei Romagnoli, 35 — 00121 OSTIA LIDO - RM Cas. Pos. 301



AMSTRAD PC/IBM Comp.

8086 - 8 MHz - Drive 360K - Monitor - Interf. parallela seriale - mouse MS/DOS - 3.2 GEM - DESKTOP - GEM PAINT - BASIC 2.

Configurazione con 512K
Versioni e manuali in italiano

1 drive - monitor monocrom.	L. 1.210.000
2 drive monitor monocrom.	L. 1.560.000
1 drive monitor graf. col.	L. 1.560.000
2 drive monitor graf. col.	L. 1.960.000

Configurazione con 640K
Versioni e manuali in italiano

1 drive - monitor graf. monocrom.	L. 1.440.000
2 drive - monitor graf. monocrom.	L. 1.860.000
1 drive - H.D. 20MB monitor graf. monocrom.	L. 2.620.000

Portatile Amstrad a partire da L. 1.210.000

Configurazione 1640 ECD con monitor colore EGA.

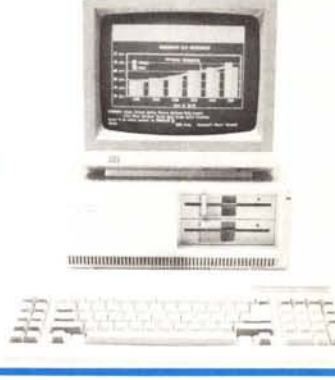
1 drive	L. 2.260.000
2 drive	L. 2.680.000
1 drive - H.D. 20MB	L. 3.410.000

MASTERBIT

MASTERBIT XT - DM 512

nuovo processore NEC V40, 5,5 e 8MHz, controller a 4 canali, 512 K, 2 drive, da 5"1/4, scheda Hercules e CGA, porta parallela e seriale, orologio, zoccolo per processore matematico, tastiera italiana, 4 slots, MS/DOS 3.2, GW BASIC 3.2, manuale in italiano, pulsante RESET esterno, altoparlante, monitor 12" fosfori verdi, doppia freq. basculante oppure 14" colore. Nuova Tecnologia

L. 1.570.000



BABY XT

Dim 26×26×8cm

8088 - 4.77/10MHz 512K - 2 drive da 3"1/2 - schede Hercules, CGA, porta ser. parall. mouse - tastiera 84 tasti - monitor monocrom. 12" doppia freq. - orologio con batt. - DOS 3.3 - man. **L. 1.770.000.**

PC PORTATILE OLIVETTI M15.

80C88 - 4,77/MHz - 512K - 2 drive da 3"1/2 - Display crist. Liq. 80 col., 25 righe, 640×200 - interf. ser. e parall. - Collegam. est. per drive da 5"1/4 - Batt. ricar., con 6 ore d'autonom. - tastiera 78 tasti - Aliment. con carica batt. - Borsa - 4 man. in it. - DOS 3.3. **L. 1.500.000**

COMPUTERS

NOVITÀ

BABY XT, dim. 26×26×8 1.770.000
8088, 4,77/10 MHz, 512K, 2 drive 3" 1/2, interf. ser. parall., mouse, tast. 84 tasti, mon. mono, doppia freq. Scheda Hercules e CGA.

MASTERBIT AT 3.350.000
512K, 80286, drive 1,2, HD 20 Mb, scheda EGA-MGA, monitor 12" fosfori verdi.

PC/AT 2.990.000
80286, 10 MHz, 512K esp. a 4 Mb, 1 drive 1,2 Mb. Hard 20 Mb, Hercules, CGA, EGA, Tast. 101 tasti, orologio, interf. parall.

PC 386 6.800.000
20 MHz, 512K esp. 8 Mb, 1 drive da 1,2 Mb, Hard 40 Mb/30MS, Hercules, CGA, EGA. Tasti 101 tasti, monitor mono dual.

PC ready 88 1.690.000
8088, 4,77/8 MHz, 1 drive da 5" 1/4, Hercules, Tast. 102 tasti, interf. ser. e parall. monitor mono basculante, DOS 3.3., man. it.

PC/AT READY 3.480.000
80286, 8/12 MHz, 512K esp. 4 Mb, 1 drive da 1,2 Mb, Hard 20 Mb, Tast. 102 tasti, interf. ser e parall., orologio, Hercules, monitor mono basculante, DOS 3.3 man. It.

PORTATILE HALIKAN 2.750.000
NECV20 4,77/10 MHz, 640 K, 2 drive da 3"1/2, display 640×200, uscita per monitor, RGB mono. Tast. 81 tasti, interf. ser e parall. batt. interna, alim. borsa, DOS 3.3 GWBasic, man.

PC WORD PROCESSOR AMSTRAD
256K 980.000
512K 1.450.000
PC BONDWELL 8 - Portatile, 512K, 1 drive 720K, 3"1/2 - Scheda grafica col. 1650.000

VIDEO WRITER PHILIPS 690.000
monitor monocrom. fosfori Ambra a 100 col. e 20 righe, tastiera 72 tasti, stampante incorporata termica a 24 aghi, cps da 25 a 50.

PC VEGAS 1.200.000
256K 1 Drive da 5,1/5 Hercules Monitor

SPECTRAVIDEO XIPRESS 16 1.350.000
256K, 8088, 2 drives, monitor 9", Joystick, MS-DOS 3.2

PORTATILE SPARK 1.990.000
NEC V 20, 4,7/9,45 MHz, 384K, 1 Drive 3"1/2, interf. ser. e parall. DOS 3.3

PORTATILE TOSHIBA T 1100+ 2.760.000

PC PHILIPS 1.480.000
8088, 4,77/8Mhz, 512 K 1 drive 3"1/2 Hercules CGA, orologio, interf. seriale parall. mon. monocrom.

PC PHILIPS 1.650.000
come sopra ma con 768 K e 2 drive

PC-PS/30 I.B.M. COMP. 1.870.000
8088, 4, 77/10 MHz, 256K 2 drive da 3"1/2, monitor monocrom. Tastiera 101 tasti, porta ser. e parall., schede hercules e CGA

PC ASEM 3011 1.830.000
Nec V20, 10 MHz, 256K, 2 drive, Hercules, monitor mono, Tastiera Dos 3.3

TRASPORTABILE XT 2.190.000
8088, 10 MHz, 256K, 2 drive da 5"1/4 CGA, display retro illum. Tastiera.

ATARI PC3 1.660.000
8088, 640K, 2 drive 5"/5, schede Hercules CGA, EGA, interf. ser. e parall., monitor ambra, mouse, DOS 3.2, man. it.

AT 80286 L. 2.550.000
512K, 12 MHz, 1 drive da 1,2, 1 hard disk 20 Mb, schede Hercules e CGA, interf. ser. e parall. tastiera, monitor 12" doppia freq., DOS 3.3

PORTATILE SHARP PC/4502 L. 2.760.000
80188, 7,16 MHz, 384K, 2 drive da 3"1/2, schermo retro illum., 88 tasti, porta ser. e parall., batt. ricar., DOS 3.21.

PORTATILE ZX88 L. 980.000
con alim. borsa, Ram da 128 K.

STAMPANTI

EPSON LX 800 539.000
80cd., 150cps, NLQ, Grafica, IBM compat.

EPSON LQ 500 839.000
24 aghi, 80 cd, 150 cps, LQ, Grafica, Ibm compat.

MANNESMANN TALLY MT 80 PC 499.000
80 col. 135 cps bidirez. NLQ

MANNESMANN TALLY MT 85 789.000
80 col. 180 cps NLQ bidirez. interf. paral. o seriale IBM/comp.

MANNESMANN TALLY MT 86 959.000
136 col. 180 cps NLQ biderez. interf. parall. o ser. IBM/comp.

MANNESMANN TALLY MT 87 869.000
80 col. 200 cps NLQ, grafica IBM comp.

MANNESMANN TALLY MT 290 1.870.000
132 col. 200 cps NLQ interf. parall. IBM/comp.

MANNESMANN TALLY MT81 L. 370.000

MANNESMANN TALLY MT 905 LASER 3.350.000

STAR LC 10 519.000
80 col., 120 cps, NLQ, grafica IBM conp.

STAR LC 10CL 559.000
come la STAR LC 10 ma a colori

STAR LC 24-10 890.000
80 col. 24 aghi, 170 cps, LQ grafica e IBM comp.

AMSTRAD DMP 3160 490.000
80 col. 160 cps NLQ Graf. IBM comp.

AMSTRAD DMP 4000 790.000
132 col. 200 cps. NLQ, Grafica IBM comp.

AMSTRAD LQ 3500 900.000
24 aghi, 80 col. 200 cps. Grafica IBM comp.

AMSTRAD LQ 5000 di 1.310.000
132 col., 288 cps 24 aghi, grafica, I.B.M.comp. interf. ser. e parall.

OLIVETTI DM 100 580.000
80 col, 120 cps, NLQ, grafica, IBM compat.

NEC P2260 975.000
24 aghi, 80 col, 168 cps, grafica, IBM compat. 12 font residenti.

PANASONIC KX-P 1081 589.000
80 col. 120 cps, NLQ, IBM comp. grafica

PANASONIC KX-P 1540 1.720.000
136 col. 240 cps, LQ, 24 aghi

PHILIPS NMS 1432 519.000
80 col. 120 cps, I.B.M. comp. graf.

MONITORS

GM 1288 D 220.000
12" doppia freq. Hercules e CGA, fosf. verdi basculante per PC.

HANTAREX BOXER 12 229.000
12" fosf. verdi alta risoluzione

HANTAREX BOXER 14 doppia freq. 269.000

HANTAREX 14 499.000
14" colore standard risoluz. 80 col.

MONITOR QL 14
per QL, 85 COL., colore 399.000

MODEM

MODEM 300 baud per C64 175.000

MODEM 300 baud per RS 232 e IBM 199.000

MODEM 1200 RF 560.000
CCITT V21/V22 BELL 103/202 - 300/600-1200 Baud può allacciarsi a qualunque sistema di ricetrasmittenti, radiotelefonici - OM - CB.

MODEM COMMUNICATOR, 300/600/1200 e VIDEOTEL per C64/128/VIC 20 225.000

PER IBM - COMPAT. — OLIVETTI 248.000

TUTTO COME SOPRA MA CON AUTOANSWER PIÙ LIRE 20.000

MODEM FULL LINK — 300/1200 FULL DUPLEX — HAYES ESTESO — INTERF. SER. E CENTRONICS — OMOLOGATO 550.000

MODEM SU SCHEDA PC INTEGRAL 300/1200 FULL DUPLEX — HAYES ESTESO — OMOLOGATO 470.000

MODEM ECCLIPSE — 300/600/1200 — VIDEOTEL — INTER. SERIALE — AUTOANSWARE — OMOLOGATO 420.000

MICROSMART 339.000
V21 - V22, interf. ser. o TTL, AUTODIAL, AUTO ANSWER, HAYES esteso

AMSTRAD PC CARD 420.000
300/1200/75-1200/1200-75

JOYSTICK

DATALINE standard 9 PIN D 14.000

SPECTRAVIDEO QS II plus 25.000
SPECTRAVIDEO QS IV 20.000
SPECTRAVIDEO QS IX 25.000

SINCLAIR QL

QL vers. ingl. JS	329.000
2 ROM JS (trasf. il QL da JM a JS)	60.000
CONVERTITORE RS 232 per stampante	99.000
CAVO QL/RS232 stamp.	35.000
CAVO JOYSTICK per QL	19.000
CAVO SER 1 per QL	15.000
BOX per 20 Microdrive	15.000
Copritastiera per QL	12.000
Inter. disco + porta parallela + RAM disk + toolkit I	229.000
drive NEC singolo	259.000
drive NEC nudo	229.000
doppio drive NEC unico contenitore	519.000
Orologio residente	30.000
TUTTI I PEZZI DI RICAMBIO: es. Contattiera	30.000

SINCLAIR SPECTRUM

SPECTRUM PLUS 48K 260.000
MANU.IT. 5 progr. supercop.

SPECTRUM 128K 299.000
2 cassette con giochi

SPECTRUM 128+2 415.000
con registratore incorporato.

Interfaccia Stampante su ROM 99.000

Interfaccia joystick tipo Kempston 1 presa 25.000

Interfaccia parlante CURRAH 60.000

Int. Ram Print. 120.000
RAM Writer incorporato + porta Jostick

INTERF. DISCIPLE 199.000
interf. disco, porta parallela per stampante 2 porte joyst., 2 network, magic bottom compat. con drive da 3"1/2, 5"1/4 e interf. 1

INTERF. DRIVE con magic bottom 119.000

DRIVE NEC 3" 1/2, 720K formattati 280.000

Multiface 1, magic bottom 105.000

Cartucce per Microdrive 5.500

Music Machin con cuffia, microfono e cassetta demo 129.000

TUTTI I PEZZI DI RICAMBIO: es. Ula 48.000

VARIE

1500 prog. per PC/comp.
10 FLOPPY POLAROID 5"1/4 26.000
con custodia cartone
10 FLOPPY POLAROID 5",1/4 30.000
con custodia di plastica
FLOPPY POLAROID da 3"1/2 5.000
FLOPPY VERBATIM PLUS formatt. da 5"1/4 2.500
FLOPPY NEUTRI 5"1/4 con etichetta 1.200
FLOPPY SENTINEL. 3"1/2 3.500
SENTINEL. 5"1/4 2.000
10 pz. con contenitore plastica nera.
MOUSE GENIUS GM6 PLUS 130.000
VOICE CARD (Fa parlare il vostro PC) 180.000
MOUSE PER PC 120.000
SCHEDA JOYSTICK PER PC 65.000
INTER. TRANSCOPY PER PC 385.000
SCHERMO ANTIRIFL. POLAROID 120.000
HARD DISK MINISCRIBE 699.000
32 Mb con controller e cavi
HARD DISK MINISCRIBE in scheda 760.000
32 Mb con controller e cavi
HANDY SCANNER 105 mm 519.000
HRC, GCA, EGA per XT/AT/PS-2
FAX MURATA M1 1.599.000
FAX per XT e AT su scheda 860.000
DRIVE 3"1/2 pr PC interno con Kit per 5"1/5 TOSHIBA da 720K L. 239.000
DRIVE come sopra ma da 1,44 Mb L. 289.000
Nuovo processore NEC V20 60.000
Scheda PARADISE, compatibile 335.000
ACCESSORI E PERIFERICHE PER COMP. IBM
INTERF. per PPC AMSTRAD e TV Domestico 449.000
SCHEDA CGA/TV
Collega un XT ad un monitor CGA o ad un TV a colori o ad un VIDEOREGISTRATORE
INTERF. PER TV A COLORI CON PRESA
SKART E COMPAT. CON SCHEDA COLORI CGA 110.000
GRUPPI DI CONTINUITÀ
BOX PER 80 FLOPPY 3"1/2 25.000
BOX PER 100 FLOPPY DA 5"1/4 25.000
KIT DI PULIZIA 5"1/4 6.500
KITI PULIZIA 3"1/2 10.000
FOTOCOPIATRICE CANON
PLOTTER GRAFITEC A TAGLIO
MULTIDRIVE esterno 580.000
1 drive da 5"1/4 e 1 drive da 3"1/2 per AMIGA 500, 2000, ATARI
INTERF. DIGITALIZZ. AUDIO 140.000
Regolabile con progr. e man. per AMIGA 500
PHILIPS macchine da scrivere elettr. da 290.000
COORDINA, programmi modulari contab. gen., magaz. e fatturaz.
REGISTRATORE DI CASSA Misuratore fiscale
INDESIT, collegabile al PC per gestione magazzino
FLOPPY KONICA da 5"1/4 1.950
FLOPPY KONICA da 3"1/2 3.750
FLOPPY KONICA alta dens. 5"1/4 4.150
FLOPPY KONICA alta dens. 3"1/2 8.450

AVVERTENZE - Tutti i prezzi sono compresivi di IVA e spese postali, per ordini inferiori alle 50.000 lire aggiungere L. 8.000 per contributo spese di spedizione - pagamento contrassegno al ricevimento del pacco. (È gradito il contatto telefonico).
SCONTI QUANTITÀ

ORDINI TELEFONICI
ORE 8.30/20.30 - Tel. 06/5621265

Garanzia 48H - la MASTERRBIT si impegna a sostituire quegli arti li riscontrati malfunzionanti entro 48H dal ricevimento, inoltre o articolo è fornito di regolare garanzia.
MASTERBIT Viale dei Romagnoli 35 - 00121 OSTIA LIDO RM - C POST. 3016