di Bruno Rosati

Speciale linguaggi: dal Twin all'ARM-Assembler

prima parte

Dice il manuale dell'ARM-Assembler che per poter programmare efficientemente è necessario acquisire una buona conoscenza di tutta la documentazione annessa alla confezione di acquisto del nostro Archie: Welcome Guide, User Guide, Programmer's Reference Manual e chi più ne ha più ne metta.
Altro passo importantissimo è il rendersi estremamente abili nell'uso di un certo Twin! Ovvero lo screen editor «ufficiale» di casa Acorn. Un tool di lavoro così evoluto e potente da risultare assolutamente indispensabile a chi fa della programmazione la principale ragione di acquisto della Risc-machine in questione

Dedicandoci ad un breve ciclo di ricognizione sulle varie implementazioni «high level language» curate in grande stile dalla stessa AcornSoft, partiamo proprio con la presentazione del Twin. Procederemo con il Risc-Assembler. Un rapido assaggio delle caratteristiche peculiari dell'ARM (prima di tutti gli altri linguaggi) che spero faccia da incentivo per chi, meditando il passaggio al «RISC-hioso» computer, oltre che con le continue lodi alla velocità supersonica, vorrebbe essere tentato con la presentazione di ottimi linguaggi di programmazione.

Introduzione al Twin

Twin, ovvero: Two WINdow editor. Un acronimo che simpaticamente tradotto dall'inglese suona come «gemello» ma che nella realtà operativa significa un editor a due finestre. Una dedicata alla stesura dei nostri sorgenti, l'altra per il controllo di eventuali errori di compilazione; linee che vengono stornate dai file che le contengono per essere passate in rassegna.

La possibilità di editare testi su venti buffer interni di memoria (ma sempre e solo due, contemporaneamente visualizzabili); quella di poter girare in background task mentre il risultato viene visualizzato in una delle due finestre in screen (magari nello stesso momento in cui, nell'altra, si sta operando immissione di testo!); l'estrema elasticità nel poter copiare testi da una window all'altra o di trasferire testi già copiati come ulteriore input a task in retro-funzionamento ed altre peculiarità (tutte da vero e proprio word processor come il Goto e la ricerca/rimpiazzo di linea e/o pagine specifiche) ne fanno uno strumento decisamente versatile.

Twin riconosce tutti i comandi del sistema operativo, mantiene la stesura cronologica dei documenti sviluppati ed ovviamente ne offre la gestione della stampa. Come si evince già da queste righe introduttive, non si sta certo descrivendo uno screen editor qualsiasi.

Il package (a differenza di molte confezioni archimediane prodotte da terzi...) è di estrema raffinatezza. Il manuale spiralato è decisamente esauriente e fa sensazione sfogliare settanta pagine ben scritte e dalla trattazione decisamente particolareggiata sull'uso di uno screen editor. (Che c'è di così tanto importante da descrivere, su di uno screen editor?).

Nel depliant «Twin Release Note» sistemato sul fondo del contenitore, si leggono tra l'altro ulteriori istruzioni su come si opera il caricamento del Twin e della lista di file contenuti nel dischetto stesso. Prendiamo così immediata nota delle due versioni di Twin disponibili: quello «normale» ad ottanta colonne e la versione «super» a 132 colonne che comunque abbisogna di un monitor ad Alta Risoluzione; quindi della loro ulteriore evoluzione — HTwin ed HTwin132 — per customizzarli con l'aggiunta di STRUCTure da noi stessi costruite per le nostre personali necessità.

Insomma ce n'è già abbastanza per farvi capire che per scrivere i vostri programmi preferiti, in Assembler come in qualsiasi altro linguaggio High Level, è Twin il posto giusto per digitarli.

Twin Assembler

Produttore:
Acorn Computers Ltd, Fulbourn Road, Cherry Hinton, Cambridge, CB1 4JN, UK
Distributore:
G. Ricordi & C. S.p.A.
Via Salomone 77 - 20138 Milano
Prezzi *(IVA esclusa):*

Twin	*L. 61.000*
Assembler	*L. 424.000*

Lavorare con Twin

Le modalità di caricamento del Twin (oltre che dagli usuali modi archimediani del DeskTop e dell'Arthur, sia per le ottanta colonne — per le quali è sufficiente digitare *Twin* — che per le 132: *Twin132*) possono essere variate a se-

conda delle nostre esigenze. Digitando ad esempio *Twin*, oltre al fatto di portarsi in modo 80 colonne, si disporrà il caricamento di una finestra sola e vuota. Scrivendo invece *Twin filèname*, verrà caricato il solo documento, «filename»; provvedendo infine con *Twin filename filename*, il programma caricherà i due file in due finestre separate.

Nel momento in cui viene caricato, Twin si mostra a pieno schermo dove sarà resa visibile la sola linea dedicata alle informazioni, i messaggi e i comandi di dialogo. La cosiddetta Linea di Stato.

Il taglio della finestra può essere ridotto abilitando lo schermo a mostrare altre finestre ed allo stesso tempo un pannello dedicato all'Help dove sono riportati anche i vari comandi assegnati ai tasti-funzione e a quelli di controllo. Tra l'altro con il comando SET MODE (SHIFT/F5) si possono selezionare fino a 6 differenti modi di schermo.

Una volta in Twin avremo finalmente la possibilità di creare i nostri preziosissimi file in tre differenti versioni: come Documenti, come File di Comando e come «purissimi» File Binari.

Appare chiaro che per «documents» sono da intendersi i codici-sorgente relativi al dato linguaggio col quale programmiamo (che Twin provvede a salvare marcandoli con la data di redazione) e che la possibilità del «binary files» è strettamente legata ad un Edit esclusivamente in «sovrascrittura» e su stringhe lavorate in ASCII. È sulla creazione dei «command-file» che urge qualche chiarimento.

Malgrado Twin riceva i comandi attraverso i tasti-funzione abilitati, è possibile guidarlo anche mediante l'esecuzione di un command-file che altro non è che un «semplice» insieme di comandi, precedentemente salvati, che formano una procedura ben precisa.

Spesso, programmando, capiterà di dover ripetere continuamente le stesse operazioni (passare da una finestra ad un'altra, forzare il contenuto di un buffer nello stack di memoria, dare lo start di prova ad un task in background, etc.). Costruendo opportune sequenze di comandi, comprensive del nome del linguaggio a cui si riferiscono, tutte queste alienanti manovre verranno tranquillamente eseguite dal nostro command-file debitamente costruito.

Chi programma sa quanto sia sempre agognata, e quasi mai esaudita, la possibilità di lavorare in un ambiente confortevole; con la massima manovrabilità possibile per la gestione di tutte le «cosacce» scritte sullo schermo. Da questo punto di vista, la predisposizione di Twin è davvero notevole, come tale è pure la «pedanteria» con la quale il manuale ne illustra tutte le molteplici potenzialità di Editing, una per una ed in mille combinazioni. Dalla basilare possibilità di scrivere in modo inserimento o in sovrascrittura (tasto-funzione INSERT/OVER corrispondente a SHIFT/F1) alla funzione di GO TO LINE (F10); dal FIND AND REPLACE (F4) al CHANGE MARGIN (SHIFT/F3) e poi via con la serie di opzioni per il CUT & PASTE del testo — marcatori di posizione e di copia —, con l'elencazione di tutte le funzionalità — TAB, ESC, tasti/cursore, etc. — della tastiera, sempre per la fase di editing, fino ad arrivare al comando INSERT FILE (SHIFT/F2), per poter fondere due file, prodotti separatamente, in un unico documento.

Twin e i tasti-funzione

Tutti i comandi assegnati ai tasti-funzione della tastiera di Archie oltre che sul pannello di Help, sono stampati sulla solita strip da inserire nella finestrella trasparente della tastiera stessa.

In numero di 36, da F1 a F12, in combinazione con lo SHIFT e con il CTRL, abbiamo fra le mani tutte le potenzialità applicabili del Twin.

Oltre alla serie dei comandi riguardanti l'Editing che già abbiamo visto e fra le usuali opzioni di save, load, chiusura/apertura finestre ed altre utilità, è comunque il caso di porre un poco di attenzione su quei comandi che, per la loro importanza, possono ben dirsi il «cuore» del Twin.

Sto pensando a comandi come STRUCT (CTRL/F4), con il quale ci è consentito selezionare la struttura customizzata sul tipo di linguaggio che si vuole utilizzare.

Twin, tramite STRUCT, supporta i template di 16 differenti linguaggi; basta premere CTRL/F4 tante volte finché non appare la struttura desiderata. Del comando PUSH BUFFER (SHIFT/F11) anche se indirettamente, si è già detto, essendo questo il comando con il quale è possibile forzare i contenuti di un buffer nello stack della memoria. Tale funzionalità è molto utile per «liberare» i buffer delle due finestre visualizzabili, senza dover aprire altri buffer per caricare e vedere i file selezionati.

Altro «pezzo» del cuore di Twin è certamente il TASK STATUS (CTRL/F9) con il quale è possibile garantirsi il controllo di un task in background. Dettovi infine che con il comando TOGGLE WINDOW (F12) si apre il secondo buffer e quindi la seconda finestra in screen e che per i restanti comandi è il caso che vi compriate il Twin... vorrei chiude-

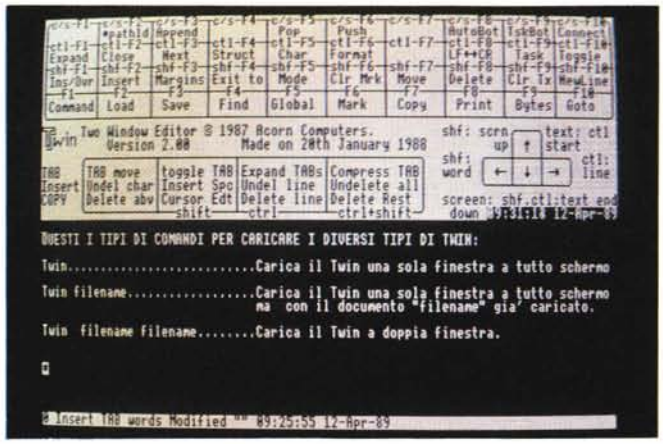

Figura 1 - Appena dato un comando SETMO DE D (che riduce la grandezza della finestra a metà schermo) ecco che il Pannello dell'Help e quello del memo dei tasti-funzione, ci appaiono completando la disposizione del Twin. Nella finestra abbiamo riportato la sintassi da usare per richiamare con le dovute modalità il tipo di Twin (80 colonne) e la seconda finestra.

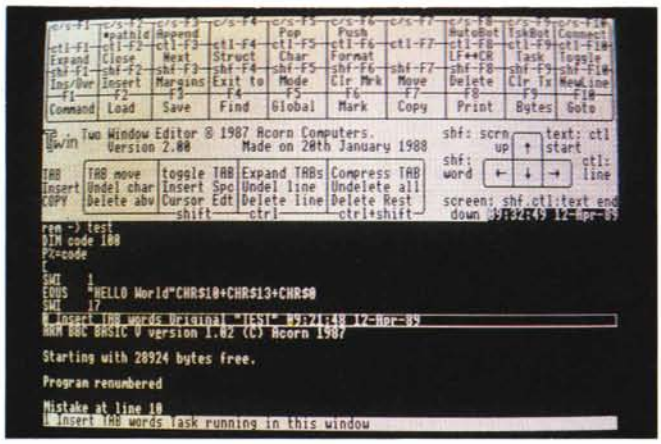

Figura 2 - Quello che vedete in figura (e che viene in parte commentato nell'articolo) è il piccolo programma-esempio presente sul manuale. Quelle nove righe di testo stanno per essere assemblate nella seconda finestra aperta in screen attraverso la chiamata del TASK STATUS, tramite l'ARM BBC BASIC.

re questa prima parte dell'articolo invitandovi ad osservare la figura 2 dove è immortalato un semplice esempio sulle doti pratiche del Twin. Tema trattato: come assemblare ed eseguire un ARM-program. La dinamica dei comandi richiamati è molto semplice. Per prima cosa, una volta scritto il testo, si è provveduto ad aprire una seconda finestra (TOGGLE WINDOW) e quindi, con il TASK STATUS, ci si è assunti il controllo del task più facile da caricare: il BBC BASIC bell'e pronto in ROM. (Sulla linea di stato che si è illuminata, subito dopo il prompt «command»: è stato sufficiente scrivere: Basic -chain test e le familiari informazioni dell'ARM BBC BASIC sono apparse). Il resto si è fatto da solo, con l'assemblaggio del breve testo rapidamente ricalcolato.

Questo è il Twin; più che un «gemello», un vero angelo custode.

Primi passi nell'ARM-Assembler

Che ARM sia l'acronimo di Acorn Risc Machine, ossia un computer dell'Acorn a Ridotto Set d'Istruzioni — e che poche istruzioni vuol dire massima velocità esecutiva (e tremendo lavoro in fase di programmazione...) — ormai lo sanno tutti. Ma che cosa sia realmente l'ARM-Assembler, a parte il solito gruppetto di «geniacci» che la fanno sempre da pionieri, sono ancora molti coloro che lo ignorano o, quantomeno, ne hanno un'idea confusa. Per questa seconda categoria è il caso di schematizzare meglio quali sono le caratteristiche principali di «Una CPU chiamata ARM». Così come fa il manuale, iniziamo col vedere come si struttura il famoso (e ridotto...) set d'istruzioni.

Com'è visibile in figura 3, composte per adattarsi ad una word a 32 — bit e rese di tipo condizionale (legate cioè allo stato dei flag) le istruzioni sono ripartite in sole nove classi. Le istruzioni di tipo Branch prendono le word disponendole a salti condizionati il cui raggio di azione va da +&2000004 a —&1FFFFF8, sufficiente ad indirizzare l'intera mappa della memoria.

Gruppo interessantissimo sono le sedici istruzioni dedicate al Data-Processing che producono operazioni tra un registro-sorgente (Rn) e un operatore, il risultato verrà convogliato verso il registro di destinazione Rd. (A riguardo di queste istruzioni osservare la figura 4).

Le istruzioni che seguono sono le Single-Data; ovvero, istruzioni usabili per muovere dati tra un registro e la memoria. LDR carica un registro da una locazione di memoria, STR lo immagazzina in un'altra locazione. Le cosiddette Block-Data, simili alle precedenti, sono preposte allo spostamento di più registri contemporaneamente. LMD carica i registri da un blocco di memoria e STM li trasferisce in un altro. L'azione di caricamento ed immagazzinamento deve essere preceduta o seguita dall'incremento o il decremento dell'indirizzo di memoria. Per quanto poi riguarda le chiamate al SuperVisor, si tratta di istruzioni che vengono estensivamente usate per comunicare con il sistema operativo ed i vari device annessi.

Le Multiply MUL e MLA sono istruzioni che utilizzano un algoritmo a 2-bit per produrre moltiplicazioni ad alta precisione fra due operandi a 32-bit.

Concludono il «ridotto set» le tre specie di istruzioni che permettono la comunicazione con il coprocessore. Le Data-Operation chiamano il coprocessore a produrre le operazioni interne; le Memory-Transfer si usano per trasferire una o più word di dati tra la memoria a cui queste sono assegnate al coprocessore; le Register-Transfer infine, trasferiscono una word da un registro dell'ARM direttamente al coprocessore.

Finita la carrellata dei nove tipi di istruzioni disponibili, due righe sui registri dell'ARM, che, normalmente, opera nella modalità User dove il programmatore è limitato a vedere un banco di 16 registri a 32-bit. Da R0 a R15. Gli altri undici registri vengono utilizzati solo quando l'ARM va in modo Interrupt, Fast Interrupt o in SuperVisor. Due dei sedici utilizzabili — R14 e R15 — hanno dei significati specifici. L'R15 ad esempio contiene il Program Counter a 24-bit e destina gli altri 8-bit al PSR (Processor Status Register). L'R14 è a sua volta usato come subroutine di Link ed è preposto a ricevere una copia di ritorno del Contatore e del PSR, quando vengono eseguite istruzioni di tipo Branch e Branch Link.

Direttive, Macro e Linker

Dopo le note riguardanti i registri e le istruzioni, arriva il turno delle Direttive. A «peso» saranno una cinquantina; da quelle per le assegnazioni simboliche (di valori numerici, registri, coprocessori e

```
                    CPU INSTRUCTION SET

B..................Branch

BL.................Branch con Link

                    Gruppo per il data-processing:
                    ADC  ADD  AND  BIC
                    CMN  CMP  EOR  MOV
                    MVN  ORR  RSB  RSC
                    SBC  SUB  TEQ  TST

LDS/STR............Trasferimento a SINGLE-data

LDM/STM............Trasferimento a BLOCK-data

SWI................Chiamate a SuperVisore

MUL/MLA............Moltiplicazioni High Precision

CDP................Coprocessore per data-processing

LDC/STC............Coprocessore per memory-trasfer

MCR/MRC............Coprocessore per register-trasfer
```

Figura 3 - Reduced Set Instruction. Le istruzioni dell'ARM sono dette condizionali, nel senso che saranno attivate ed eseguite solo nel caso in cui i vari flag N, Z, C e V siano nello stato di corretto funzionamento. La condizione di default è di «esecuzione».

Figura 4 - Istruzioni per il Data Processing. Questo sommario compendia il valore mnemonico, il significato ed il tipo di operazione che le 16 istruzioni preposte al processo possono eseguire; ovviamente sempre in corrispondenza del corretto stato dei flag.

```
                ISTRUZIONI PER IL DATA-PROCESSING

MEMO     SIGNIFICATO                                  OPERAZIONE

ADC......Add con Carry.....................Rd:= Rn+operand+C
ADD......Add...............................Rd:= Rn+operand
AND......And...............................Rd:= Rn AND operand
BIC......Bit Clear.................Rd:= Rn AND (NOT(operand))
CMN......Compare Negated...........................Rn+operand
CMP......Compare...................................Rn-operand
EOR......OR esclusivo..................Rd:= Rn EOR operand
MOV......Move.....................................Rd:= operand
MVN......Move Not.........................Rd:= NOT operand
ORR......OR logico.....................Rd:= Rn OR operand
RSB......Revers subtract...................Rd:= operand-Rn
RSC......Revers subtract with carry......Rd:= operand-Rn-1+C
SBC......Subtract with carry.............Rd:= Rn-operand-1+C
SUB......Subtract.............................Rd:= Rn-operand
TEQ......TestEQuivalence.....................Rn EOR operand
TST......TeST and mask.......................Rn AND operand


Nota:

 Rd è il registro di destinazione. Rn è quello di sorgente.
```

registri-coprocessori attribuiti ad equivalenti simbolici) a quelle di STORE-LOADING, seguite quindi da Direttive addette ad allineare (ALIGN) sulla locazione dell'istruzione seguente dopo una store-loading appena eseguita. Altre Direttive sono quelle delle aree di immagazzinamento e poi, importantissime quanto potenti, quelle delle variabili.

Divise «modernamente» in locali e globali, queste vengono ulteriormente selezionate in aritmetiche (GBLA), logiche (GBLL) e stringa (GBLS). Mentre le Global possono operare per tutto il file ove sono comprese, le Local sono costrette nell'ambito dei confini stabiliti da una Macro-istruzione che le usa. Rimandando un breve approfondimento nel momento in cui parleremo delle possibilità di utilizzo di Macro, saltiamo dalle variabili all'ASSERT che è una Direttiva per il controllo degli errori in espressioni logiche ed infine, sorvolando su di un bel gruppetto di «directivies» che vi andrete a studiare per conto vostro (anche perché non è proprio possibile passarle tutte in rassegna!) eccoci balzare al gruppo delle OBJASM. L'ObjAsm sarebbe quel particolare Assembler che non crea file-oggetto eseguibili, bensì singole porzioni di programmi che andrano poi linkate. Fra le dedicate ai codici prodotti in tale Assembler, troviamo la direttiva AREA che assume un nome con annesso un attributo e l'allineamento all'area nella quale il codice, o l'informazione seguente la Direttiva stessa, è stato inserito. Quindi le Direttive IMPORT, EXPORT, STRONG e KEEP che sono tutte impostate, ovviamente, ai riferimenti che si dovranno avere con altri prodotti di ObjAsm al momento del linkaggio.

Ma che inutile spreco di meningi e polpastrelli se dopo tutte le istruzioni di questo mondo e le tormentose Direttive non ci fosse quel qualcosa di magico che riuscisse ad unirle fra di loro! Ovvero: la Macro-istruzione; ovvero ancora: l'assemblaggio di istruzioni e Direttive che potremo via via creare, inventariare e richiamare alla bisogna. Lo sapete il detto informatico che «... *chi scrive prima non scrive dopo*»? Nella creazione di Macro verrà fatto un uso massiccio delle variabili locali LCLA (aritmetica), LCLL (logica) e LCLS (stringa). I valori mutanti in codeste variabili potranno esser inseriti nelle globali, usando i raccordi di altre direttive quali SETA, SETL e SETS.

Ma ora è tempo di Linker, di file-oggetto e librerie varie da linkare. Il Linker indubbiamente è un programma indispensabile per chi sviluppa il proprio software con compilatori ad Alto Livello. Le caratteristiche di un Linker difatti sono quelle di poter combinare i contenuti di una serie di file-oggetto (prodotti da un compilatore come dall'ObjAsm) con quelli di librerie, producendo un programma finale funzionante.

Il fatto che il Linker sia compreso nel nostro pacchetto, ci permette quindi un utilizzo estremamente proficuo dell'ObjAsm, introducendo tra l'altro in questa parte finale del discorso quello che è il concetto emergente nell'ambito dei metodi di programmazione. Ovvero: sviluppare il software utilizzando compilatori ad Alto Livello e ricorrere all'Assembler — operando tra l'altro più facili conversioni — solo per sciogliere, migliorandoli in velocità, nodi particolarmente congestionati.

Liberi di programmare in «puro» Assembler se volete, quello che voglio comunque evidenziare è l'estrema validità di simile metodo e, di conseguenza, l'estrema importanza della variante ObjAsm presente nell'Acorn Assembler. Se un programma scritto in AAsm sarà sempre il più veloce del mondo, dall'altra, un linkato, lo sarà infinitamente di più in fase di sviluppo. L'ObjAsm ed il Linker dell'Acorn Assembler sono a vostra disposizione.

Note di riferimento

Introduzione, Riferimenti all'ARM-CPU, Registri, Istruzioni... e poi le Direttive ed appresso le Macro ed il Linker, esempi di programmi assemblati ed accenni al Floating Point. Nelle 180 pagine dell'Acorn Assembler Guide c'è spazio solo per informare e neanche un rigo per un pur timido abbozzo di tutorial. Ma d'altronde, un manuale che apre la prima pagina, quella dell'*About this manual*... pregando di venir considerato né più né meno di una Reference Guide, non può non disporsi che al raggiungimento dell'unico scopo che una Reference Guide deve porsi: l'istruzione generale sulle caratteristiche del microprocessore in questione e del suo Assembler; istruzioni, variabili, direttive, sintassi, esempi e particolarità come i messaggi di errore. E questo viene fatto con estrema chiarezza, malgrado l'argomento sia complesso per sua natura (ancora di più nel caso di un Risc). Un plauso quindi alla AcornSoft che non sviluppa i suoi manuali solo perché questi debbono in qualche modo accompagnare i dischetti e riempire la bella confezione.

Per il resto, e cioè l'insegnamento alla programmazione, come una più o meno particolareggiata guida ai «trucchi», si è inevitabilmente costretti a cercare altrove. Ma è così per ogni prodotto. I manuali di riferimento vanno completati da guide all'utilizzatore. Testi, nel caso specifico, sul tipo dell'*ARM Assembly Language Programming* di Peter Cockrell. Una pubblicazione che, tra l'altro, viene consigliata nello stesso manuale dell'Assembler. Riprendendo il filo logico del manuale, il testo di Cockrell accompagna il programmatore nel nuovo mondo passo dopo passo.

Conclusioni

Anche se non è tradizione di MC microcomputer, questo articolo non è altro che la prima parte di un lungo dossier dedicato ai linguaggi di Archie. Qualche volta capita di fare articoli a puntate. Di iniziare discorsi che se non sono lunghi come *Dallas* o peggio ancora: *Sentieri*... debbono fisicamente subire un taglio mensile. L'importante è dare il senso compiuto all'argomento che si sta affrontando, e noi credo che lo abbiamo dato. Non finisce il «dossier Linguaggi», ma finisce, — e per quella che ritengo solo un'indagine conoscitiva — sufficientemente esaustivo, l'argomento legato al Twin e all'Assembler. Anche la suddivisione dello spazio disponibile fra i due non è casuale. Se è vero, come è vero, che in un articolo di Assembler non se ne può certo fare un corso, ma di un Twin se ne può dare un quadro decisamente esauriente, si è optato per una semplice ma opportuna presentazione. Doverosa tra l'altro per chi, avendo intenzione di comperare pacchetti che non costano certo quanto un videogame, gode del diritto di venire informato. (Sono a conoscenza del fatto che alcuni utenti hanno comperato l'ARM-Assembler senza il Twin e si sono poi messi a piangere...). MC

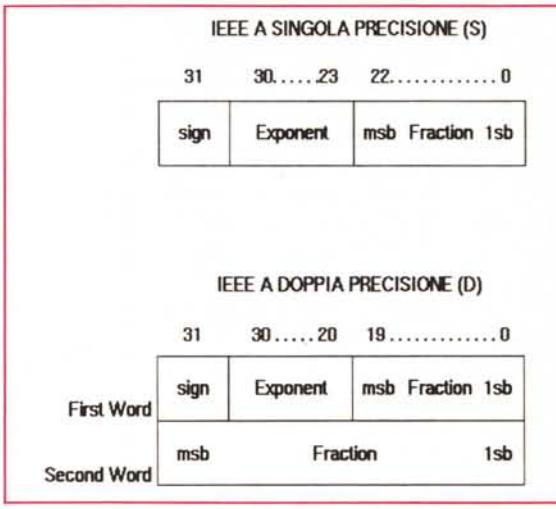

Figura 5 - Floating Point Emulator. Il sistema dell'ARM, perfettamente aderente allo standard IEEE, si struttura su otto registri di coprocessore ad alta precisione. Così come le istruzioni dell'ARM, anche le operazioni di Data Processing svolte dal Floating Point, si riferiscono ai registri, piuttosto che alle locazioni di memoria. I valori ottenuti possono quindi essere immagazzinati nella memoria dell'ARM in quattro differenti formati: a Singola Precisione (S); a Doppia Precisione (D); a Doppia Precisione Estesa (E) e a Decimale (P). Quella in figura è una esemplificazione grafica della IEEE a Singola e Doppia Precisione.