UNPACKAGE A.D.E. per il L.M. del PC 128S

The ultimate assembly language development tool System Software ltd.

Programmare in linguaggio macchina con il PC128S è almeno in linea di principio, abbastanza facile (ammesso di conoscere il microprocessore 6502 ed i punti d'ingresso del sistema operativo): il BASIC BBC, già di per sè abbastanza strutturato, consente l'introduzione di codice assembly "in linea", semplicemente racchiudendo il tutto tra parentesi quadre; al momento dell'esecuzione del programma, l'Assemblatore incorporato nell'interprete BASIC provvederà a generare il codice oggetto e ad inserirlo in memoria all'indirizzo desiderato.

Questo modo di operare però risulta essere limitato in pratica, solamente a programmi di limitata estensione, in genere piccole routine da inserirsi all'interno di programmi BASIC. Non appena si intraprende la stesura di un lavoro maggiormente articolato, si finisce con lo scontrarsi con i limiti dell'Assemblatore BASIC; non che con esso sia impossibile scrivere grossi programmi, accade solamente che si comincia a sentire la mancanza di quelle facilitazioni presenti negli assemblatori più "seri" e che consentono di sbrogliare matasse aggrovigliate (o meglio, consentono, a chi ne fa buon uso, di evitare i grovigli!).

A questo punto entra in ballo ADE plus: si tratta di un pacchetto integrato per lo sviluppo di programmi in linguaggio assembly, comprendente, udite udite, un Text Editor, un Macro Assemblatore, un Linker, un Debugger, e per finire ma non certo ultima, una Memory Management Unit (MMU), che funge da interfaccia utente, ed in un certo senso, fa da "colla" per legare assieme tutte queste cose. Non tutti però sapranno esattamente che cosa siano un Macro Assemblatore od un Linker, vediamo dunque un po' più in dettaglio di che si tratta.

Gli strumenti

La funzione principale di un Assemblatore, è quella di tradurre i codici mnemonici che compongono un programma scritto in linguaggio assembly nei valori numerici corrispodenti, sì da produrre un "codice oggetto", direttamente eseguibile dal microprocessore (a questo proposito, ADE+ supporta i mnemonici e la sintassi standard della serie 65C00). Il "codice sorgente", ovvero il testo del prcgramma, viene scritto per mezzo di un text editor (ADE+ ne ha uno incorporato, ma si può usare anche VIEW), dunque salvato su di un file, oppure in un'apposita area di memoria (l'input buffer); a questo punto bisogna invocare l' Assembler, che compirà puntualmente il suo lavoro, segnalando eventuali errori incontrati: ADE+ dispone di una settantina di messaggi d'errore, che si dividono in fatali, non fatali e "warnings", cioè errori che non pregiudicano l'assemblazione, ma meritano comunque di essere investigati.

Al momento di lanciare l'Assemblatore, è possibile utilizzare delle "opzioni", per condizionare lo svolgimento dell'assemblazione (se effettuare o meno il listato del programma, se dirigere l'output sulla stampante, se utilizzare il set d'istruzioni ridotto....); è inoltre possibile specificare il nome del file dove si desidera che vada il codice oggetto prodotto, o quello dell'"output file", dove verranno registrati tutti i risultati del processo d'assemblazione, compreso l'elenco dei simboli incontrati ed un sommario degli errori.

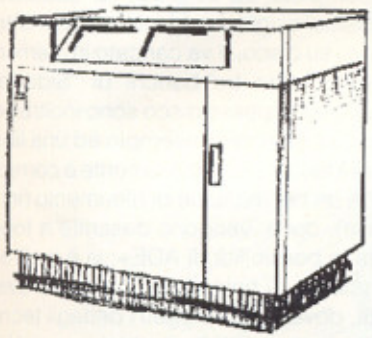
Già questo è molto più di quanto offra l'Assemblatore del BASIC: si ha l'impressione di essere trattati in guanti bianchi: ADE+ ci informa di tutto quello che è successo in macchina e del perché sia successo; ma il bello deve ancora venire. Un asssemblatore che si rispetti infatti, non deve solo riconoscere i codici mnemonici che corrispondono ad istruzioni del microprocessore, ma anche delle "pseudoistruzioni" o "direttive assembler", utilissime per controllare diverse operazioni connesse con il processo di assemblazione. Per la cronaca, la sola direttiva riconosciuta dall'asemblatore del BASIC e la "EQU" (nelle varianti EQUB, EQUW, EQUD, EQUS), utile per assegnare un valore (byte, word, double word o stringa) ad un simbolo definito dall'utente.

ADE+ riconosce invece circa una cinquantina di pseudo operazioni, che si possono suddividere in direttive di assemblazione, di definizione dati, di assemblazione condizionale, di controllo file.

Le macro

Tra le direttive di assemblazione, di particolare interesse è la possibilità di gestire "Macro Istruzioni" (di qui la denomi-

nazione di Macro Assemblatore). Definire una Macro significa associare ad una nome inventato dall'utente (l'identificatore della Macro), un intero gruppo di istruzioni, cosicchè usare questo nome all'interno del programma assembly, equivale al riscrivere il gruppo d'istruzioni in questione. Vi è inoltre la possibilità di passare dei "parametri" alla Macro, in questo modo, facendo largo uso di Macro Istruzioni, si scrivono programmi più leggibili, quasi come se si usasse un linguaggio ad alto livello.



Un breve esempio

Vediamolo attraverso un esempio: supponendo che in un programma sia necessario richiamare più volte la funzione VDU, cosa che in L.M. si svolge con due operazioni: caricando A con il carattere da stampare, e chiamando la routine OSWRCH; risulta comodo definire una Macro, chiamandola per es. VDU:

```
VDU MACRO ; Definizione della Macro "VDU"
LDA #@1 ; carica in A il primo parametro
JSR OSWRCH ; chiama la routine di stampa
ENDM ; fine della definizione di Macro
```

Lo strano simbolo @1 sta per "il primo parametro della Macro" (@2 per il secondo, @3 per il terzo...); a questo punto, definita la Macro, se vogliamo stampare qualcosa nel nostro programma, non dobbiamo far altro che richiamarla: per scrivere una "W" basta fare: VDU "W"; per cambiare modo schermo : VDU 22 VDU modo Ora vogliamo di più: se il nostro programma deve cambiare spesso modo grafico, conviene definire una Macro per il modo:

```
MODE MACRO
VDU 22
VDU @1
ENDM
```

abbiamo cioè una Macro che richiama un'altra Macro! Ora per cambiare modo è sufficente scrivere ad es. MODE 3 e l'Assemblatore automaticamente si preoccuperà di sostituire la parola MODE con l' opportuna sequenza di codici definita nella Macro!

Qualcuno avrà notato che l'etichetta OSWRCH non è stata definita: ADE+ assegna all'inizio tutti i valori corretti ad identificatori come OSWRCH, OSBYTE, OSWORD, che pertanto sono tutti identificatori predefiniti; comodo vero? A questo punto nulla vieta di scrivere una raccolta di Macro di utile e frequente utilizzo: sarà possibile racchiuderle in un file che fungerà da libreria per il nostro Assemblatore. Ed a questo hanno già pensato i signori della System Software, infatti, sul disco di ADE+ oltre a dei programmi di esempio, esiste già pronta un' intera libreria di Macro nel file 'MACLIB'.

Il linker

Sempre fra le direttive di asssemblazione, ve ne sono alcune dedicate alla gestione di moduli di programma ed all'import/export di siboli esterni: questo discorso porta dunque a parlare del Linker.

Normalmente, quando si scrive un programma in linguaggio assembly, si specifica, tramite la direttiva ORG, l'indirizzo da dove questo programma dovrà partire in memoria; quindi si chiama l'Assemblatore, ed il programma è pronto per essere eseguito. Dovendo intraprendere però la stesura di qualcosa di più grande, appare logico suddividere questo mega programma in unità fondamentali, che chiameremo moduli; se tutto il programma non riesce a stare nella memoria disponibile per l'Assemblatore, sarà sempre possibile farlo stare un modulo alla volta. Tra i tanti moduli, ve ne potrà essere uno che funge da libreria, contenendo al suo interno tutta una serie di routine matematiche, grafiche, e così via; a questo faranno riferimento tutti gli altri moduli che necessitano di tali routine.

É possibile dunque scrivere ed assemblare i diversi moduli uno alla volta, senza però specificare un indirizzo di partenza per mezzo della ORG: l'Assemblatore produrrà un "codice rilocabile", che ovviamente non potrà essere direttamente eseguito, non essendo stato scritto per lavorare in nessun punto preciso della memoria. Assemblati dunque tutti i moduli del nostro programma, ci troviamo ad avere tanti codici rilocabili che così come sono non possono essere di alcuna utilità. Ora entra in ballo il Linker: si tratta di un programma che "ricuce" tra di loro tutti i moduli che gli vengono dati in pasto. Un codice rilocabile contiene al suo interno anche tutti i riferimenti necessari per il collegamento di quel modulo con l'esterno, ed è compito del Linker sfruttare queste informazioni per effettuare tutti i collegamenti tra i vari moduli e con il modulo di libreria, assegnare al programma un opportuno indirizzo di inizio e rilocare i diversi moduli nelle posizioni che competono a ciascuno.

Alla fine di tutto ciò il programma è pronto per girare! Questo lavorio potrà sembrare macchinoso, ma porta ad innumerevoli vantaggi quando ci si trova a trattare con programmi che, scritti in unica soluzione, sembrerebbero interminabili. Bisogna fare una modifica? Non serve riassemblare tutto il programma da capo, basta alterare il solo modulo interessato. Vediamo un piccolo esempio: MODULE Scribacchino text EXT ; "text" è un simbolo esterno SYSEXEC ENT ; ENT sta per "Entry point" LDX #-1 :ciclo INX LDA text,X ; prende un carattere alla volta BEQ :fine ; se è CHR 0 allora esce dal ciclo JSR OSASCI ; chiama la routine di stampa JMP :ciclo :fine RTS

Questo moduletto prende qualunque stringa si trovi all'indirizzo 'text' e la mostra sullo schermo. Problema: text non è definito, o meglio, è definito come 'EXT' cioè come riferimento ad un modulo esterno. Da notare l'entrata del programma: SYSEXEC è un nome speciale che viene riconosciuto dal Linker come punto d'ingresso principale per il programma. Ci serve un altro modulo dove mettere il testo da scrivere: MODULE Gallina ; contiene il testo per il modulo Scribacchino text ENT STR "Meglio l'ovetto " STR "oggi che la " STR

"gallina domani! " BRK ; l'istruzione BRK produce un codice ASCII 0 Ogni modulo deve essere assemblato separatamente: bisogna dare ad ADE+ i seguenti comandi: ASM parte1=Scribacchino ASM parte2=Gallina Ed ora linkare: LINK progr=parte1,parte2

Il Linker fa sentire la sua presenza informando del successo avuto dalla operazione: sul disco ora è presente un file di nome "progr" contenente il codice oggetto eseguibile. Adesso il programma è finalmente pronto: lo si può lanciare con il comando *RUN progr ed ecco subito apparire sullo schermo una nota massima...

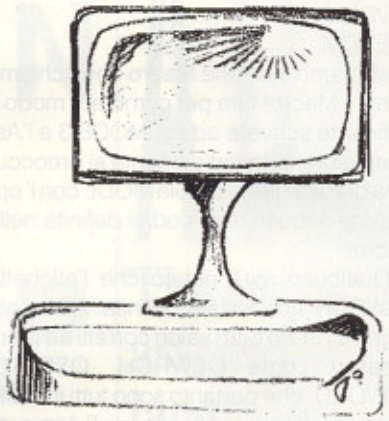
La MMU

Finora non se ne è parlato, ma tutte queste operazioni si sono svolte sotto lo sguardo vigile della MMU: la Memory Management Unit. Si tratta di un programma supervisore, che si occupa di sfruttare al meglio tutta la memoria a disposizione del PC128S, gestendo delle zone tampone dette appunto "buffer", ed impedendo che operazioni sconsiderate vadano ad alterare i contenuti delle zone riservate. Il primo contatto dell' utente con ADE+ si ha proprio con il livello comandi della MMU: da qui è possibile richiamare l'editor con il comando EDIT, l'Assemblatore con ASM, il Linker con LINK, il Debugger con DEBUG.

Il Debugger

Parliamo dunque di quest'ultimo: Debugger è uno strumento praticamente indispensabile per curiosare nella memoria di un computer, visualizzando blocchi di byte, ed alterando contenuti di locazioni. Quello che appare all'utente è una specie di pannello di comando colorato, dove in alto sono visualizzati tutti i registri del processore con il loro contenuto, il Program Counter con l'istruzione corrente, lo stato dei vari flag. Al centro vi è una sorta di finestra aperta sulla memoria del PC128S: vi sono visualizzati 64 byte, a partire dall'indirizzo contenuto in un puntatore che può venir spostato a piacimento. In basso vi è una riga libera per l'introduzione dei comandi.

Si tratta per lo più di comandi che si attivano con la pressione di un solo tasto: è possibile andare "su e giù" con il puntatore di memoria, esplorando diverse zone di RAM o ROM, è possibile scegliere il formato di visualizzazione, infatti i contenuti delle locazioni di memoria possono essere mostrati sotto forma di numeri esadecimali o attraverso i codici ASCII corrispondenti (molto utile per la ricerca di frasi e messaggi).

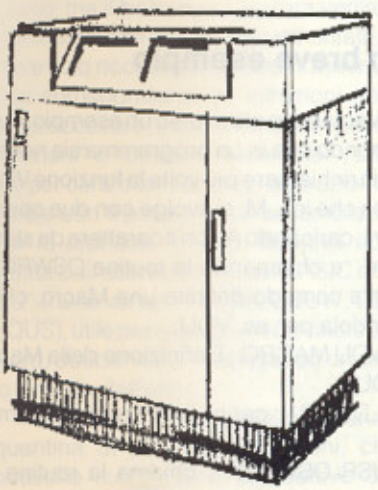


Di fondamentale importanza è la possibilità di "disassemblare" il contenuto della memoria: i valori numerici vengono ritrasformati in codici mnemonici, ottenendo una specie di programma in linguaggio assembly (senza però le etichette ed i commenti!); come se non bastasse vi sono comandi per la modifica di locazioni o gruppi di byte: quanto basta per accontentare gli smanettoni. Debugger però è soprattutto uno strumento per la verifica del funzionamento di routine in linguaggio macchina, non manca dunque la possibilità del "single step", dell'eseguire cioè un programma una istruzione alla volta, stando a guardare quello che succede. Si possono inserire inoltre fino ad otto "break points", cioè punti di controllo all'interno di un codice macchina: si lancia il programma, e quando questo arriva ad un break point, esso si arresta temporaneamente, ritornando il controllo all'utente che può modificare il contenuto di variabili o registri, per poi eventualmente proseguire.

Quindi...

Debug è proprio quanto basta per chiudere in bellezza con ADE+, sebbene questo pacchetto risulti sempre aperto a nuove eventuali espansioni (disponendo di qualche coprocessore, si potrebbe collegare un cross assembler...). Ovviamente quanto visto fino ad ora è solamente un assaggio delle potenzialità di ADE+, assaggio che avrà lasciato scontento sia il lettore principiante, che quello super esperto-mega informato (che di certo starà sbadigliando), ma avrà speriamo lasciato l'acquolina in bocca a tutti coloro i quali, già in possesso dei punti di partenza, desiderino fare un passo in avanti nell' utilizzo del linguaggio macchina.

Per questi, ma anche per i più esperti, ADE+ si rivelerà uno strumento veramente completo ed in grado di soddisfare qualsiasi esigenza. L' intero pacchetto è fornito su disco, e va caricato in memoria, dove occupa tre banchi di "Sideways RAM"; sullo stesso disco sono inoltre presenti programmi di esempio ed una libreria di Macro. Il tutto ovviamente è corredato da un bel manuale di riferimento (in inglese), dove vengono descritte a fondo tutte le possibilità di ADE+; vi è inoltre la possibilità di richiedere una guida avanzata, dove sono spiegati i dettagli tecnici sulla struttura dell'Assemblatore e del Linker.



Un elogio dunque alla System Software Ltd. per aver prodotto uno strumento degno di reggere il paragone con i più evoluti assemblatori presenti per macchine ad otto bit, e buon lavoro a tutti quelli che si accingeranno ad adoperarlo!