COSTRUIAMOCI UN VERO MICROELABORATORE

HOME COMPUTER AMICO 2000

In possesso della scheda base, l'AMICO 2000/A che abbiamo costruito insieme lo scorso numero, scenderemo questa volta in alcuni particolari che riguardano sia l'hardware (ovvero la circuiteria elettronica vera e propria) sia il software esaminando il significato di altre importanti istruzioni oltre a quelle che fino ad ora abbiamo imparato. Alla fine di questo articolo passeremo direttamente alla fase pratica insegnandovi ad introdurre un simpatico programma che permetterà di utilizzare il nostro AMICO 2000/A come un orologio digitale di grande precisione; questo, insieme con il programma del gioco dei riflessi sarà il secondo della vostra biblioteca programmi che andrà arricchendosi di volta in volta.

a cura della A.S.E.L. - parte quarta

Tutti coloro che fino ad ora ci hanno seguito, o almeno la maggior parte di essi, saranno entrati in possesso dell'elaboratore AMICO 2000/A e cioè della scheda base di un sistema che andrà via via ingrandendosi.

Ci auguriamo che tutti abbiano portato a buon fine il cablaggio della piastra, che non abbiano difficoltà e che quindi siano pronti a seguirci con la solita attenzione ed entusiasmo anche questa volta.

Ricordiamo ai lettori che ci leggessero per la prima volta che questa serie di articoli sul microcomputer, molto orientati alla pratica, è cominciata nel numero 12/'78 di Sperimentare.

Questa volta non c'è niente da costruire, cercheremo, computer alla mano, di approfondire la conoscenza dal punto di vista software ed hardware riprendendo prima alcuni concetti fondamentali legati alle operazioni matematiche nel sistema di numerazione binario. Ci scusiamo con quelli che già sanno queste cose, ma, come abbiamo sempre detto, vogliamo essere sicuri che tutti siano in grado di seguirci tranquillamente senza tirarsi dietro dubbi di sorta che potrebbero compromettere la totale comprensione di questo affascinante argomento.

La somma nel sistema binario

Abbiamo già visto nel numero precedente una operazione elementare, la somma binaria di due numeri. Le stesse regole che ci permettono di eseguire una somma nel nostro solito sistema decimale, sono usate nel sistema binario. Inoltre nel sistema binario, dato che vi sono due sole cifre (lo 0 e l' 1) queste regole sono ancora più semplici.

Vediamole insieme e introduciamo il concetto di "Carry":

0 + 0 = 0	Riporto (Carry) = 0
0 + 1 = 1	Carry = 0
1 + 0 = 1	C = 0
1 + 1 = 0	C = 1

Assunte queste regole fondamentali supponiamo ora di dover eseguire la somma: 1 + 1 + 1.

Il risultato di questa operazione è 1 con il C = 1. Perchè?

Scomponendo abbiamo: (1 + 1) + 1 = 0 + 1 con C = 1 (per via della somma 1 + 1); ora 0 + 1 = 1. Il risultato della intera operazione quindi è 1 e il C rimane uguale a 1.

Utilizziamo queste regole per effettuare la somma:
$0A_{16} + 07_{16} = 11_{16}$.
(N.B. - Si tratta di cifre esadecimali).
Trasformandole in binarie abbiamo:

```
0 0 0 0   1 0 1 0 (0A)
0 0 0 0   0 1 1 1 (07)
---------------------
0 0 0 1   0 0 0 1 (11)
```

Partendo dalla cifra a destra si ha:

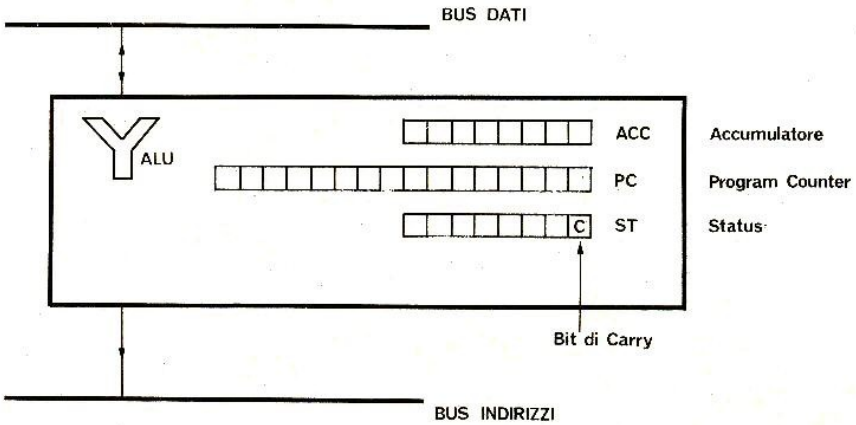
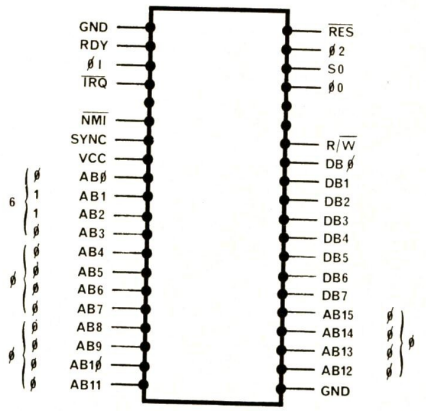


Fig. 1 - Alcuni particolari all'interno della CPU: Accumulatore, Program Counter e Status Register.



ZOCCOLATURA DEL MICROPROCESSORE 6502

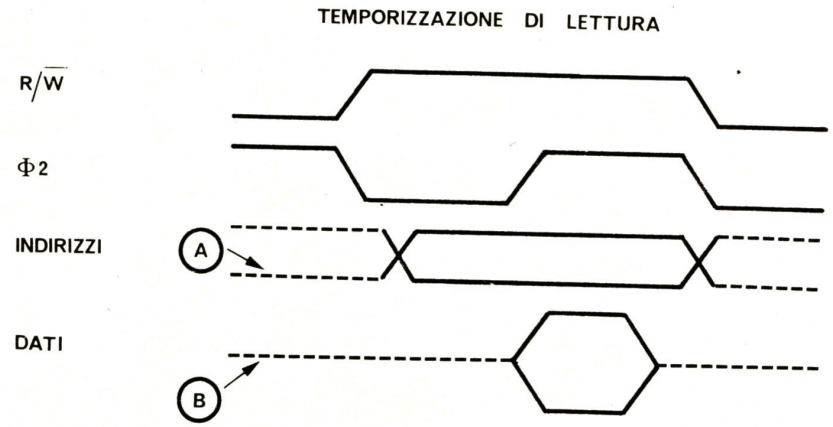


Fig. 2 - Piedini della CPU e loro funzioni. — **A.** *Nella zona tratteggiata è indifferente lo stato logico degli indirizzi.* **B.** *Nella zona tratteggiata non si deve prendere per valido il dato eventualmente presente sul bus indirizzi.*

C = 0 Carry uguale a 0 all'ingresso della operazione.

0 +1 +0 =1 C = 0

1 +1 +0 =0 C = 1

0 +1 +1 =0 C = 1

1 +0 +1 =0 C = 1 Carry intermedio.

0 +0 +1 =1 C = 0

0 +0 +0 =0 C = 0

0 +0 +0 =0 C = 0

0 +0 +0 =0 C = 0 Carry all'uscita dell'operazione.

Notiamo che la somma di ogni cifra viene fatta tenendo conto anche del Carry. Se ora ritorniamo al nostro elaboratore ci chiediamo: dove sta fisicamente il Carry?

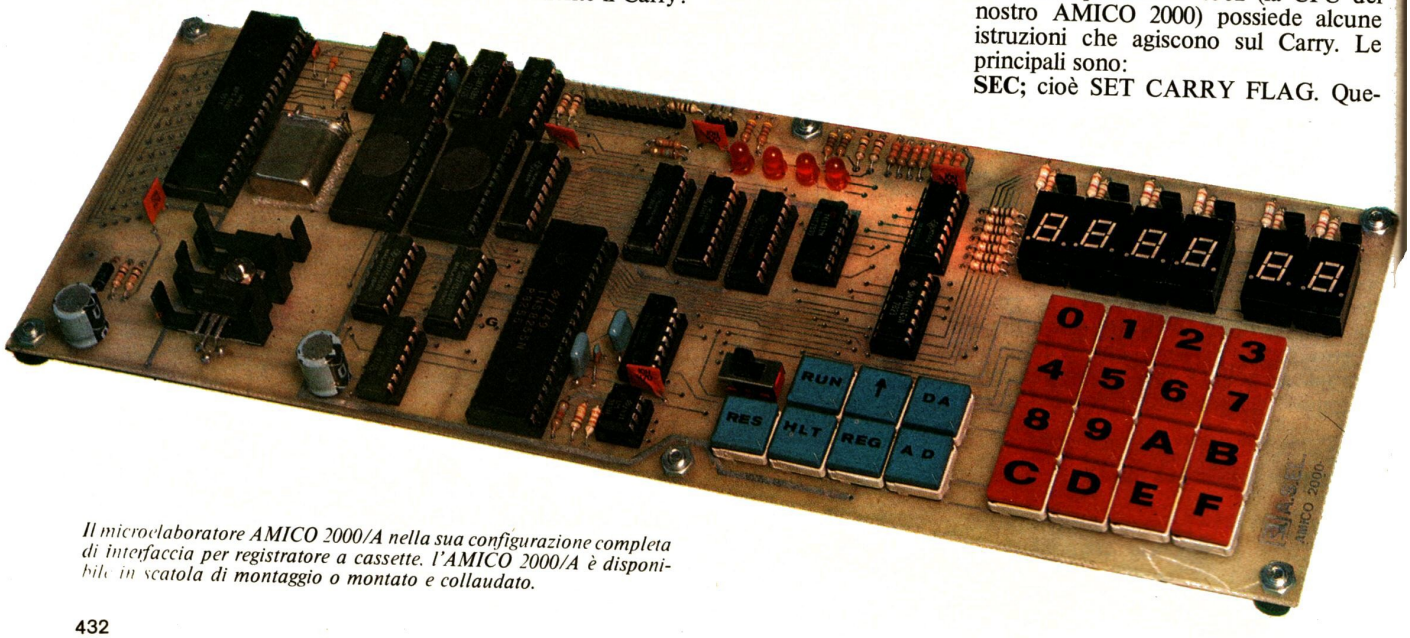
Per rispondere dobbiamo introdurre un nuovo registro presente nella CPU. Fino ad ora abbiamo incontrato l'ACCUMULATORE, il PROGRAM COUNTER e l'UNITÀ ARITMETICO-LOGICA (ALU).

Il REGISTRO DI STATO (Status in inglese) è il nuovo registro che contiene alcune informazioni sul progredire delle operazioni che il microprocessore sta eseguendo. Lo Status è formato da 8 bit di cui il primo è proprio il CARRY. Al momento attuale quindi la nostra CPU può essere rappresentata come in figura 1.

Vogliamo puntualizzare che: 1) Il Program Counter è un registro da 16 bit; 2) Dello ST (Status register) abbiamo definito solo il primo bit che è il Carry; gli altri bit dello Status verranno analizzati in seguito.

Il microprocessore 6502 (la CPU del nostro AMICO 2000) possiede alcune istruzioni che agiscono sul Carry. Le principali sono:

SEC; cioè SET CARRY FLAG. Que-



Il microelaboratore AMICO 2000/A nella sua configurazione completa di interfaccia per registratore a cassette. l'AMICO 2000/A è disponibile in scatola di montaggio o montato e collaudato.

sta istruzione mette a 1 il bit di Carry (Set in inglese). La sua traduzione in linguaggio macchina è **38.**

CLC; cioè CLEAR CARRY FLAG. Questa istruzione mette a Ø il bit di Carry (Clear in inglese). La sua traduzione in linguaggio macchina è **18.**

Riprendendo l'esercizio interrotto possiamo dire che il Carry all'ingresso della somma può essere da noi condizionato; lo possiamo infatti porre a Ø o a 1 a piacimento tramite una delle due istruzioni citate.

```
  1 Ø Ø 1 1 1 1 1    +
  1 1 Ø 1 Ø Ø 1 Ø
1 Ø 1 1 1 Ø Ø Ø 1
↓ Carry
```

Cioè 9F + D2 = 71 con il riporto di 1 (Carry di uscita = 1).

Gli esempi fino ad ora fatti sono stati sviluppati nel sistema Esadecimale. Le stesse cose però valgono anche nel nostro sistema Decimale che usiamo tutti igiorni.

Infatti se eseguiamo l'operazione:

$83_{10} + 41_{10} = 1\ 24_{10}$
sistema ↗ ↑riporto
decimale

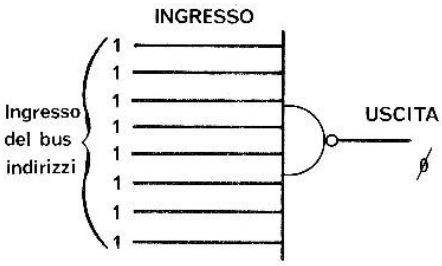
troviamo ancora il riporto, o carry, esattamente come abbiamo appena visto.

Esercizio con l'AMICO 2000/A

Cerchiamo ora di mettere in pratica le varie nozioni che abbiamo appena appreso.

Il calcolo da fare è ancora il solito: la Somma. però una volta la faremo con il Carry di ingresso a 1, la volta successiva con il Carry di ingresso a Ø.

Accendiamo la macchina e partiamo a scrivere il nostro programma per esempio dalla locazione di memoria 2Ø1.



L'uscita viene a Ø solo quando tutti gli ingressi sono a 1 in quanto la funzione logica impiegata è del tipo NAND.

Fig. 3 - Decodifica per l'indirizzo FF.

Il Carry all'uscita dell'operazione dipende dal risultato della operazione medesima.

Infatti nell'esercizio appena visto il Carry di uscita era uguale a zero, ma se avessimo fatto: 9F + D2 avremmo ottenuto:

Tasti	Indirizzi	Dati
AD	ØØØ6	
DA	ØØØ6	(primo dato, p.e. Ø2)
↑	ØØØ7	(secondo dato, p.e. Ø3)

Ora carichiamo il PC di partenza del programma. Diciamo cioè al calcolatore da che punto deve partire l'esecuzione dello stesso.

Allora premiamo AD Ø2Ø1 poi RUN sul display comparirà il risultato ØØØØ Ø5.

Il risultato è 5 perchè il Carry di ingresso è stato posto a Ø (prima istruzione di CLC).

Proviamo ora a porlo = 1. Per farlo sostituiamo CLC (18) con SEC (38).

Quindi sempre con la stessa procedura:

AD	Ø2Ø1	18
DA	Ø2Ø1	38

Tasti	(display) Indirizzi	Dati	Istruzione	Commenti
AD	Ø2Ø1	xx		
DA	Ø2Ø1	18	CLC	Azzerramento del Carry
↑	Ø2Ø2	A5	LDA Ø6	Carico l'addendo
↑	Ø2Ø3	Ø6		
↑	Ø2Ø4	65	ADC Ø7	Sommo all'accumulatore il contenuto della locazione Ø7 e il Carry.
↑	Ø2Ø5	Ø7		
↑	Ø2Ø6	85	STA ØØ	Metto il risultato nella locazione ØØØØ.
↑	Ø2Ø7	ØØ		
↑	Ø2Ø8	4C	JMP MONITOR	Istruzione di fine programma. Torno al programma di Monitor.
↑	Ø2Ø9	22		
↑	Ø2ØA	FE		

Ora che avete inserito il programma possiamo immettere i dati da elaborare.

RUN

Sul display apparirà come risultato Ø6. Il Carry in ingresso è stato posto da noi a 1.

Provate ora voi stessi a cambiare i dati e ripetere più volte l'esercizio.

Per riassumere abbiamo:

```
Ø2 +
Ø3 +
Carry
```

= Ø5 se il Carry di ingresso = Ø
= Ø6 se il Carry di ingressi = 1

Attenzione non spegnete a questo punto l'elaboratore perchè fra poco aggiungeremo alcune istruzioni al programma.

Somma esedecimale e somma decimale

Per ciò che riguarda la somma dobbiamo ancora spiegare la differenza fra somma esadecimale e somma decimale.

Abbiamo già visto nella parte prima che è:

$1Ø_{10} = ØA_{16}$.

Ora è evidente che se si esegue la operazione

Ø5 + Ø5

il risultato sarà 1Ø se espresso in forma decimale e ØA se espresso in forma esadecimale.

Ma attenzione! La matematica non è una opinione.

$Ø5_{10} + Ø5_{10} = 1Ø_{10}$ il che equivale a $ØA_{16}$

$Ø5_{16} + Ø5_{16} = ØA_{16}$ il che equivale a $1Ø_{10}$

Il problema è solo quello di sapere in che base si sta operando.

Il nostro AMICO 2000 può lavorare in entrambe le basi. Per farlo ha due istruzioni dedicate:

1) **SED** Set Decimal Mode. Codice operativo **F8**. Tutte le somme fatte dopo questa istruzione vengono eseguite in *decimale.*

2) **CLD** Clear Decimal Mode. Codice operativo **D8**. Tutte le somme fatte dopo questa istruzione vengono eseguite in *esadecimale.*

Aggiungiamo ora tornando al nostro computer queste ultime istruzioni al programma precedentemente introdotto. Vi abbiamo detto di non spegnere la macchina, se la avete spenta per qualsiasi ragione dovete reintrodurre il programma precedente prima di procedere alle modifiche.

Tasti	Indirizzi	Dati	Commenti
AD	Ø2ØØ	xx	Apro la locazione Ø2ØØ
DA	Ø2ØØ	F8	Metto la macch. in calcolo decim.

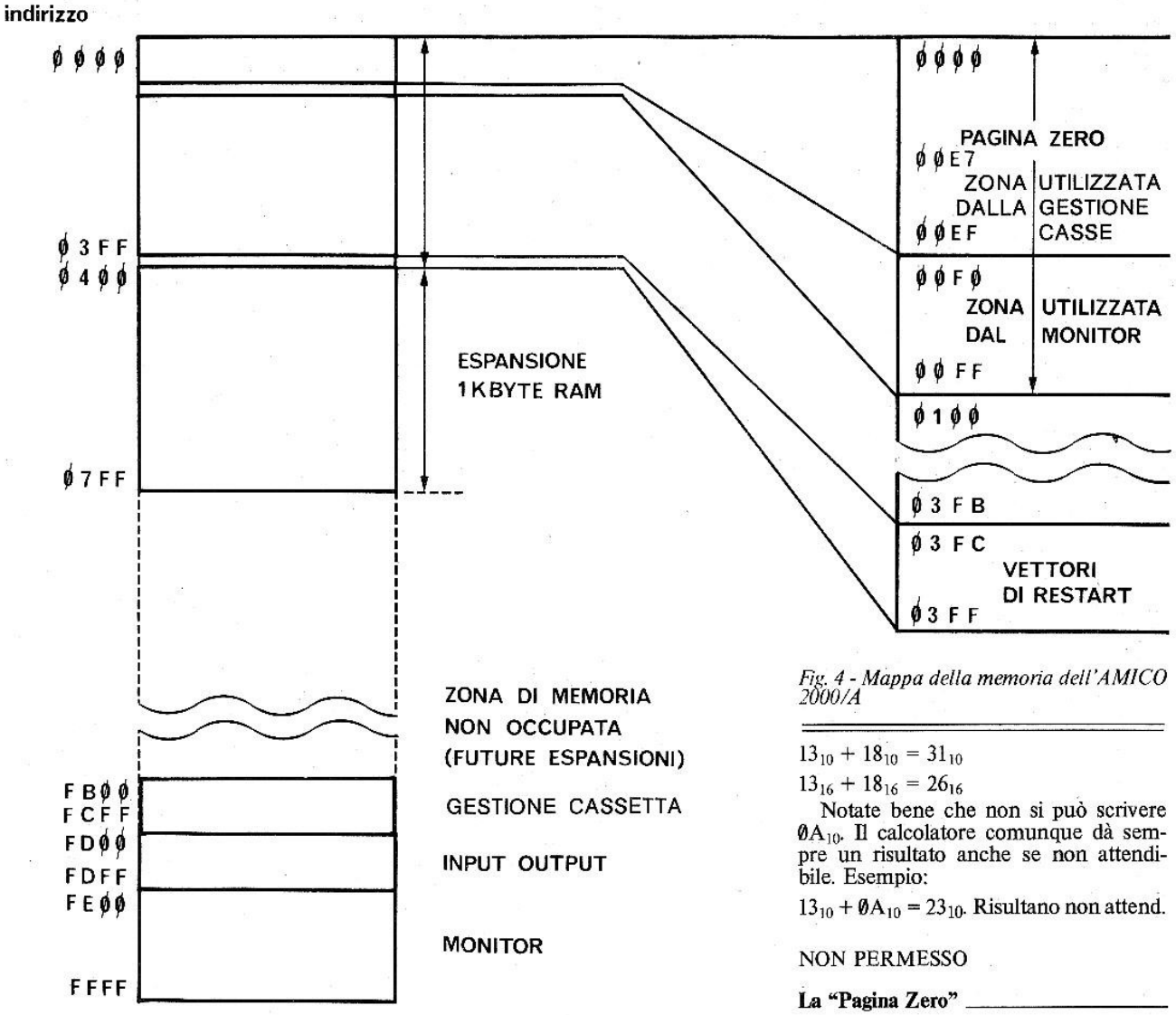


Fig. 4 - Mappa della memoria dell'AMICO 2000/A

↑	0201	18	CLC Ripristino l'istruzione di Clear Carry
AD	0006	xx	Apro la locazione 0006
DA	0006	05	Introducono l'addendo 05
↑	0007	05	Introducono l'addendo 05
AD	0200	F8	Mi riporto alla locazione 0200
RUN			Faccio partire il programma

Il risultato sarà 10
Ora cambiamo modo di funzionamento.

Tasti	Indirizzi	Dati	Commenti
AD	0200	F8	Mi riporto alla locazione 0200 il cui contenuto è da modificare.
DA	0200	D8	D8 è il codice operativo della nuova istruzione che sostituisco alla precedente (F8).
RUN			Partenza programma

Il risultato sarà 0A.
Provate ora a cambiare i dati e i modi di funzionamento. Qui di seguito riportiamo qualche esercizio risolto.

$13_{10} + 18_{10} = 31_{10}$
$13_{16} + 18_{16} = 26_{16}$
Notate bene che non si può scrivere $0A_{10}$. Il calcolatore comunque dà sempre un risultato anche se non attendibile. Esempio:
$13_{10} + 0A_{10} = 23_{10}$. Risultano non attend.

NON PERMESSO

La "Pagina Zero"

Abbiamo fin qui esaminato abbastanza approfonditamente le seguenti istruzioni:

Istruzione		Codice operativo
SEC	→	38
CLC	→	18
SED	→	F8
CLD	→	D8

Queste sono tutte istruzioni di un solo byte e che non richiedono altro per essere definite, sono cioè implicite.
Le istruzioni LDA e ADC invece necessitano di una ulteriore definizione. Si deve infatti precisare cosa bisogna caricare nell'accumulatore o cosa bisogna sommargli.
Nella tabella presentata nella seconda parte di questa serie (Sperimentare Gennaio '79 pagg. 35-36) si trova.

A5	LDA	Zero page
65	ADC	Zero page

Cosa significa?

La pagina zero (*zero page*) della memoria è per definizione una zona della memoria che comprende tutte le locazioni comprese fra gli indirizzi **0000** e **00FF** ed è quindi formata da 256 byte.

Vedremo quindi che è comodo lavorare con le locazioni di memoria in pagina zero. Perchè?

Perchè si sa già che il primo byte (la cosiddetta parte alta dell'indirizzo) è **00**. Allora analizzando meglio questa zona di memoria:

0 0 0 0 ÷ 0 0 F F
1° byte 2° byte 1° byte 2° byte

Dove il 1° byte è la **parte alta dello indirizzo** e il 2° byte è la **parte bassa dell'indirizzo.**

Ciò che cambia è *solo il secondo byte* ed è solo quello che dobbiamo precisare.

L'istruzione che carica il contenuto della locazione di memoria **0006** nello accumulatore si scrive:

A5
06

Abbiamo allora che **00** è sottointeso ovvero c'è perchè sto lavorando con una istruzione (A5) che è in pagina zero.

Ma se voglio caricare il contenuto della locazione di memoria **0306** nell'accumulatore devo scrivere:

AD
06 (parte bassa dell'indirizzo)
03 (parte alta dell'indirizzo)

L'istruzione AD nella solita tabella cui abbiamo fatto cenno, viene indicata come "assoluto" (vedi: AD - LDA - Absolute) cioè ad essa deve seguire l'indirizzo completo della locazione di memoria da cui devo prelevare il contenuto.

Ovviamente per la pagina zero esiste l'eguaglianza:

A5 = AD
06 = 06 (parte bassa dell'indirizzo)
00 (parte alta dell'indirizzo)

Però la seconda forma è *più lunga.* Il programma che ne risulta occupa più memoria, è quindi meno efficiente.

A questo punto è necessario fare un altro esercizio.

Trasportiamo un dato dalla locazione **06** alla locazione **00**. Cominciamo il programma a partire dalla locazione **0200**. Vedere tabella A.

Facciamo partire il programma (come al solito si carica l'indirizzo di partenza **0200** e poi si preme [RUN]) e vedremo il contenuto della locazione di memoria **06** copiato nella locazione **00**.

Per far ciò basterà come al solito selezionale [AD] **0006**, vedere il contenuto sul display dei dati, poi selezionate [AD] **0000** e verificate che in quella locazione si trovi lo stesso dato di prima.

Proviamo ora a modificare il programma come segue.

TABELLA A

Tasti	Indirizzi	Dati	Commenti
[AD]	0200	xx	Indirizzo di partenza
[DA]	0200	A5	LDA Pagina 0. Contenuto della locazione 06 in Accumulatore.
[↑]	0201	06	
[↑]	0202	85	STA Pagina 0. Contenuto dell'accumulatore nella locazione 00.
[↑]	0203	00	
[↑]	0204	4C	JMP START. Arresto del programma.
[↑]	0205	22	
[↑]	0206	FE	

Tasti	Indirizzi	Dati
[AD]	0200	A5
[DA]	0200	AD
[↑]	0201	06
[↑]	0202	00
[↑]	0203	85
[↑]	0204	00
[↑]	0205	4C
[↑]	0206	22
[↑]	0207	FE

Se facciamo partire il programma (premendo [AD] 0200 poi [RUN]) noteremo che il risultato è lo stesso, ma il programma è più lungo. Le stesse cose dette per la istruzione LDA valgono ovviamente per la istruzione STA. Anche per questa esiste un sistema di indirizzamento in pagina base (o pagina zero) con codice operativo 85 e un sistema di indirizzamento assoluto con codice operativo 8D.

Il precedente programma si può allora riscrivere anche così:

Tasti	Indirizzi	Dati
[AD]	0200	A5
[DA]	0200	AD
[↑]	0201	06
[↑]	0202	00
[↑]	0203	8D
[↑]	0204	00
[↑]	0205	00
[↑]	0206	4C
[↑]	0207	22
[↑]	0208	FE

Facendo partire il programma anche in questo caso vediamo che il risultato è sempre lo stesso.

Il metodo di indirizzamento

Abbiamo introdotto in sordina un concetto nuovo: IL METODO DI INDIRIZZAMENTO. Ne abbiamo visto due tipi: in pagina base e assoluto. Le *stesse istruzioni* cambiano codice operativo a seconda del metodo di indirizzamento usato. Dovrete esaminare con molta attenzione la tabella cui abbiamo fatto riferimento (quella pubblicata sul n. 1/79).

Sarà molto importante e utile quando si vorranno tradurre dei programmi da noi scritti in linguaggio simbolico in linguaggio macchina (cioè nei codici operativi compresi dal microprocessore 6502).

Per questa volta abbiamo finito con il software (il linguaggio di programmazione), cercate di impadronirvi delle poche, ma importanti nozioni e istruzioni che fino ad ora vi abbiamo insegnato provandole direttamente sul microcomputer. Tenete conto che la macchina fa esattamente ciò che voi scrivete sulla carta interpretando le istruzioni a una enorme velocità e soprattutto senza sbagliare. Per maggiori chiarimenti sui sistemi di conteggio vi consigliamo un volumetto molto valido di un po' di anni fa: "I sistemi di numerazione - D.A. Johnson, W. Glenn - Ediz. Zanichelli".

Approfondimento hardware: il clock

Esaminiamo ora qualche aspetto fisico dello scambio di segnali fra la CPU e l'integrato numero 1 a 40 piedini) e gli altri componenti dell'AMICO 2000/A.

Il problema è: come può il 6502 (la CPU) leggere nella ROM e leggere e scrivere nella RAM?

Avete notato che nell'AMICO 2000/A c'è un quarzo da 1 MHz; questo significa che nel nostro calcolatore c'è un oscillatore che, oltre a tutto, è preciso e stabile. Il *clock* (l'onda quadra) di uscita di questo oscillatore scandisce tutti i tempi principali della macchina. I segnali più importanti sono il Φ2 (OUT) cioè il piedino 39 dell'integrato e il $R/\overline{W}$

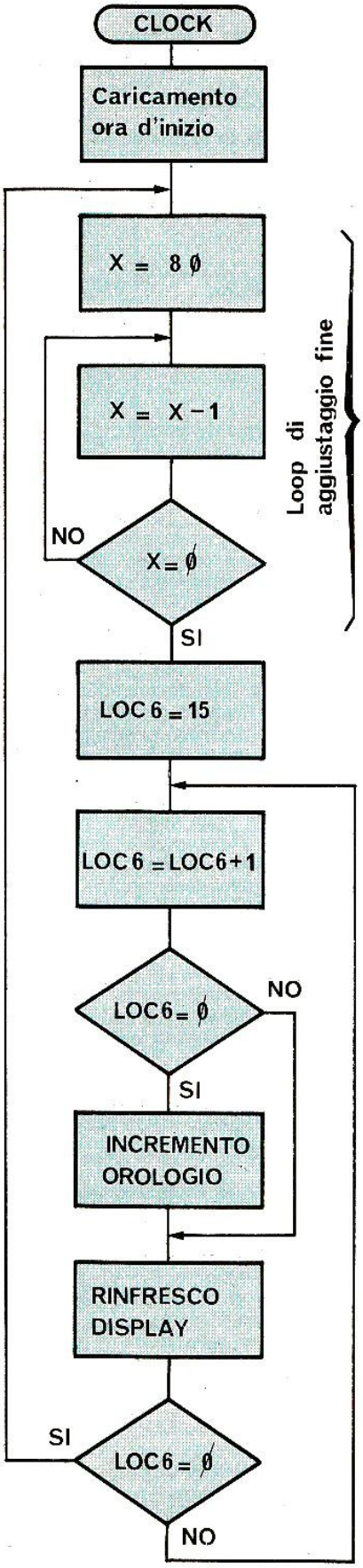


Fig. 5 - Flow chart del programma per realizzare un orologio digitale.

(cioè Read/Write leggi/scrivi) piedino 34 dell'integrato.

Gli altri fili interessati sono: i piedini 9÷20 e 22÷25 da cui la CPU fa uscire gli indirizzi della memoria interessata (AB0 = Address bit 0, cioè il bit meno significativo dell'indirizzo AB15 = Address bit 15, cioè il bit più significativo).

Per rinfrescare le idee sul concetto dell'indirizzamento rimandiamo il lettore a rileggere le pagine 1041 e 1047 del numero 12/78 di Sperimentare nel quale compare il primo articolo della serie dedicata al microelaboratore.

Spieghiamo ora cosa si intende per "bit più o meno significativo".

Se ci rifacciamo alla numerazione decimale quella che usiamo tutti i giorni, prendiamo in considerazione un numero qualsiasi ad esempio il numero 35.417.

La variazione in più o in meno di una unità assume evidentemente un valore (o peso) diverso a seconda della posizione della cifra: in questo caso la cifra meno significativa è rappresentata dal 7 (ovvero dalla cifra più a destra del numero) quella più significativa dal 3 (ovvero da quella più a sinistra del numero).

Trasferendo analogamente lo stesso discorso al concetto dei 16 bit (dal bit 0 al bit 15) degli indirizzi vediamo che il cambiamento di stato (da 0 a 1) del bit più significativo sposta l'indirizzamento di memoria dal primo blocco che va dalla 0 alla 32767_a locazione al secondo blocco che va dalla 32768_a alla 65535_a locazione.

Continuando la descrizione della CPU i piedini 26 ÷ 33 sono quelli tramite i quali essa presenta all'esterno un dato o lo preleva dall'esterno (DB0 = Data Bit 0, cioè il bit meno significativo del dato. DB7 = Data Bit 7 cioè il bit più significativo del dato).

Se la CPU vuole prelevare un dato dalla locazione di memoria 0006 (lo fa perchè noi glielo abbiamo indicato con un'istruzione) presenta il numero 0006 sui piedini di uscita dell'indirizzo mette a 1 il piedino R/$\overline{W}$ e quando la fase Φ 2 è alta (cioè a 1), legge sui fili dei dati quello che la memoria vi ha deposisato (vedi figura 2).

Vediamo ora di rispondere a questa domanda: come fa la memoria a immettere i dati sui fili del bus dati?

Per far ciò esiste una decodifica che esamina gli indirizzi che escono dalla CPU. Se questi indirizzi sono quelli voluti la CPU segnala alla memoria che la stanno interrogando e che può immettere i suoi dati sul bus. In figura 3 è rappresentato il più semplice tipo di decodifica che riconosce un indirizzo formato da tutti 1.

Se la Cpu vuole scrivere un dato nella cella di RAM 0006 presenta all'uscita gli stessi dati del caso precedente ma ora tiene basso (ovvero a 0) il piedino R/$\overline{W}$ per indicare alla memoria che vuole scrivere. Ed essa lo fa. Il tutto come sempre viene scandito dal clock.

Questi sono i segnali che ora principalmente interessano la nostra trattazione· in seguito approfondiremo le funzioni di altri piedini.

Suddivisione della memoria nell'AMICO 2000

In figura 4 viene riportata la mappa della memoria, cioè si mostra come è posizionata la memoria del nostro microelaboratore.

La figura è molto importante e verrà spiegata man mano nei dettagli durante il proseguimento degli articoli di questa serie.

Per ora si fa notare la pagina base (indirizzi 0000 ÷ 00FF), il Monitor e la gestione della cassetta che sono posizionati nelle ultime locazioni di memoria. A proposito, vi ricordate che cosa è il Monitor? È quel programma residente in ROM, che dà vita alla tastiera e al display che permette cioè alla CPU di accendere i LED di acquisire il tasto premuto e di interpretarne il significato. Esso permette inoltre l'esame del contenuto della memoria la modifica di questo contenuto se la cella esaminata è di memoria RAM la partenza del programma che abbiamo introdotto a mano.

Applicazione

Questa volta faremo funzionare il calcolatore da orologio. La calibrazione dell'orologio dovrete farla voi perchè dipende dalla precisione del vostro quarzo agendo essenzialmente su una locazione di memoria.

Diamo una breve descrizione del funzionamento di questo programma la cui "costruzione" è stata fatta mediante una cosiddetta "flow chart" o carta di flusso di cui parleremo più diffusamente in un prossimo articolo e che riportiamo in figura 5.

Il programma è costituito essenzialmente da un contatore contenuto in tre byte successivi di RAM, gli F9 - FA - FB di pagina base. In F9 sono contenuti i secondi, in FA i minuti, in FB le ore. Ogni secondo noi andremo a sommare 1 della locazione F9 (incremento dei secondi): poi ci chiediamo: siamo arrivati a 60 secondi? Se abbiamo raggiunto i 60 secondi, azzeriamo i secondi e sommiamo 1 ai minuti. Confrontiamo i minuti con 60 e ripetiamo l'operazione già

fatta per le ore; poi si torna all'inizio.

Come al solito il programma viene caricato da tastiera a partire dalla locazione 0300. Questo programma è stato scritto così come lo trovereste nei testi inglesi e come li scriveremo noi successivamente. In pratica si tratta di ripetere le solite operazioni iniziali selezionando la prima locazione di memoria (tasto AD), inserendo il primo dato o codice operativo dell'istruzione (tasto DA) procedendo poi ad inserire gli altri dati utilizzando il solito tasto con la freccia (↑).

Non analizzando a fondo questo programma in quanto vi mancano ancora alcune nozioni, per ora ci limitiamo a sottolineare alcune particolarità:
- nella prima colonna è indicata la locazione di memoria in cui si va a porre l'istruzione;
- nella seconda colonna c'è il byte (dato) da introdurre nella locazione di memoria indicata;
- nella terza colonna è riportata l'istruzione in linguaggio mnemonico;
- la quarta colonna è dedicata al commento.

Analizzando ora qualche simbolo. Il simbolo # sta ad indicare che quello che segue è un numero, non una locazione di memoria; quindi se si scrive LDA # 00 si indica che carichiamo il numero 00 nell'accumulatore, in maniera immediata come si dice in gergo Se però scriviamo LDA 00 invece carichiamo nell'accumulatore *il contenuto della locazione di memoria 0000.*

In linguaggio macchina le due istruzioni si traducono come segue:

```
LDA # 00    A9
            00
LDA 00      A5
            00      siamo in pagina base!
```

Il simbolo $ prima del numero sta ad indicare che il numero che segue è espresso in *Esadecimale.*

quindi sarà:

```
LDA   #$10   A9 10 (esadecimale)
LDA   # 10   A9 0A (esadecimale)
```

Ripetiamo che per caricare il programma si procede come al solito:

```
AD   0300          partiamo da questa
                   locazione)
DA   0300  F8
↑    0301  A5
```

e via di seguito alla fine.

A questo punto dobbiamo caricare il numero dei secondi in un minuto, dei minuti in un'ora e delle ore in un giorno:

```
AD   0003  xx
DA   0003  60   secondi in un minuto
↑    0004  60   minuti in un'ora
↑    0005  24   ore in un giorno
```

Ora carichiamo l'ora di partenza (qualche minuto avanti rispetto a quella esatta):

```
RES  0000  xx   La pressione del tasto RES ci ha portato ad aprire la
                locazione di memoria 0000.
DA   0000  (ss) Inserisco i secondi
↑    0001  (mm) Inserisco i minuti
↑    0002  (hh) Inserisco le ore
```

Si carica ora il PC di partenza del programma AD 0300; si preme RUN nel momento in cui scocca l'ora da noi programmata.

Volendo cambiare l'ora fermiamo l'orologio tramite il tasto RES ripetendo le operazioni succitate introducendo i nuovi valori. Se l'orologio non precede giusto si può cambiare il contenuto della locazione 0312: diminuendo il valore contenuto si accelera l'orologio, aumentando il valore si rallenta l'orologio.

Buon divertimento!

GLOSSARIO

MICROPROCESSORE – È il circuito integrato che contiene la logica necessaria a realizzare le funzioni di unità logica e aritmetica e i registri necessari per un corretto funzionamento.

LSI – È il termine che definisce un circuito integrato ad alta complessità circuitale.

MICROCALCOLATORE – È il termine che identifica un sistema costituito da un microprocessore, da una memoria e da una o più possibilità di ingresso e uscita.

HARDWARE – L'Hardware è costituito dall'insieme dei circuiti elettronici, dalle varie schede, dai circuiti di alimentazione e da tutto quanto è fisicamente visibile.

SOFTWARE – Il Software è costituito dall'insieme dei programmi e delle routines che possono essere caricate nella memoria del microcalcolatore.

BIT – Rappresenta l'informazione binaria elementare.

BYTE – Rappresenta l'informazione binaria costituita da 8 BIT.

BUS – È costituito da un insieme di segnali aventi una direzione è un significato ben preciso. Ad esempio il BUS indirizzi e rappresentato da un certo numero di fili che conducono i segnali dalla C.P.U. a tutta la memoria del Microcalcolatore.

RAM – Memoria ad eccesso casuale nella quale vengono depositati e/o prelevati i dati variabili durante l'esecuzione del programma; può anche essere utilizzata per contenere i programmi.

ROM – Memoria a sola lettura nella quale è normalmente contenuto un programma o delle routines.

EPROM - È una memoria ROM di tipo particolare in quanto può essere riprogrammata.

CPU – Spesso si identifica con il termine microprocessore anche se per precisione è costituita dall'unità centrale aritmetica.

I-O – Identifica una o più parti di un microcalcolatore destinate al colloquio con il mondo esterno.

LOCAZIONE – È un posto della memoria nel quale è contenuto un Byte.

PROGRAMMA – È la sequenza di istruzioni scritte per ogni calcolatore o microprocessore in grado di eseguire una determinata funzione.

ASSEMBLER – Programma in grado di tradurre delle istruzioni simboliche in un linguaggio comprensibile alla macchina.

MONITOR – È un programma che permette di controllare il funzionamento di un sistema contenente un microprocessore permettendo la scrittura, la lettura e la modifica delle varie locazioni di memoria.

REGISTRO – È una Memoria del tutto particolare utilizzata per specifici impieghi sia logici che di controllo o indirizzamento. Ad esempio l'Accumulatore è un registro che ha il compito di contenere i vari dati che si muovono all'interno del sistema e i risultati di operazioni matematiche. Il Program Counter è un registro che viene utilizzato per indicare il punto della memoria nel quale deve essere prelevata la prossima istruzione.

POINTER – È un registro particolare che viene utilizzato per indicare la locazione di memoria dove deve essere depositato o prelevato un Byte.

ISTRUZIONE – È una parte elementare del programma che dice al sistema quale funzione deve eseguire sin quel momento.

ALU – Questo termine indica l'unità logica e aritmetica cioè il vero e proprio cuore del microprocessore.

PROGRAMMA DI OROLOGIO

SI CARICA DALLA LOCAZIONE 0300

Locazione	Codice operativo	Istruzione		Commento
0300	F8	SED		Mi metto a contare in decimale
1	A5	LDA	00	Trasporto i secondi di partenza nel counter.
2	00			
3	85	STA	F9	
4	F9			Carico i minuti.
5	A5	LDA	01	
6	01			
7	85	STA	FA	
8	FA			Carico le ore.
9	A5	LDA	02	
A	02			
B	85	STA	FB	
C	FB			
D	A2	LDX	#$95	Contatore di aggiustaggio fine.
E	95			
F	CA	DEX		
0310	D0	BNE	030F	
1	FD			
2	A9	LDA	#$15	Contatore di aggiustaggio grosso.
3	15			
4	85	STA	06	Usiamo la locazione di memoria 06 per tenere il contatore.
5	06			
6	E6	INC	06	Incremento il contatore di attesa 1 secondo.
7	06			
8	D0	BNE	0334	Se non è passato 1 secondo vado a rinfrescare il display.
9	1A			
A	A2	LDX	#00	Inizio scansione.
B	00			
C	A0	LDY	#01	Incremento.
D	01			
E	18	CLC		Clear carry.
F	98	TYA		Prelevo l'incremento.
0320	75	ADC	F9, x	Sommo.
1	F9			
2	D5	CMP	03, x	
3	03			
4	D0	BNE	032A	
5	04			
6	A9	LDA	#00	
7	00			
8	F0	BEQ	032D	
9	03			
A	A0	LDY	#00	
B	00			
C	EA	NOP		
D	95	STA	F9, x	
E	F9			
F	E8	INX		Incremento pointer.
0330	E0	CPX	#03	
1	03			
2	D0	BNE	031F	
3	EB			
4	20	JSR	SCADS	Accensione display.
5	0C			
6	FF			
7	EA	NOP		
8	A5	LDA	06	Vedo se ho finito l'incremento.
9	06			
A	D0	BNE	0316	
B	DA			
C	4C	JMP	030D	
D	0D			
E	03			

MODULO DI ORDINAZIONE PER IL MICROELABORATORE "AMICO 2000/A"

Prego inviarmi a stretto giro di posta il seguente materiale, IVA compresa:

☐ quantità _____ "AMICO 2000/A" in scatola di montaggio **(Lit. 195.000)**

☐ quantità _____ "AMICO 2000/A" montato e collaudato **(Lit. 235.000)**

☐ quantità _____ Alimentatore 1A per "AMICO 2000/A" **(Lit. 15.000)**

(scrivere in stampatello)

Nome _____

Cognome _____

Via _____ Tel. _____

Codice Fiscale _____

CAP _____ Città _____

Per il pagamento scelgo la forma:

☐ **anticipato a mezzo assegno circolare o vaglia** (spese di spedizione a carico della A.S.E.L.);

☐ **parzialmente in contrassegno** (in questo caso è necessario inviare un anticipo di Lit. 57.000 a mezzo assegno circolare o vaglia, il resto verrà pagato alla consegna del pacco - spese di spedizione a carico del Committente).

IMPORTANTE: La merce viaggia a rischio e pericolo del Committente; è possibile assicurarla aggiungendo **Lit. 2.000** per ogni 50.000 di valore assicurato.

Inviare il presente modulo in busta chiusa con allegata copia della ricevuta del vaglia alla:

A.S.E.L. s.r.l. - Via Stadera, 18 - 20141 MILANO

ATTENZIONE: *chi ha già l'AMICO 2000 con i moduli pubblicati nei numeri 12/78 e 1/79 di Sperimentare dovrebbe aver ricevuto una lettera di conferma dalla A.S.E.L.*
Informiamo inoltre i lettori che gli ordini verranno evasi man mano che arrivano, si consiglia quindi di spedire immediatamente il modulo di ordinazione.

CARATTERISTICHE TECNICHE

CPU: microprocessore 6502

Memoria RAM: 1 K byte

Memoria ROM contenente il monitor

Tastiera esadecimale

Visualizzatore LED a 6 cifre

Interfaccia parallelo

Predisposto per interfaccia per telescrivente e per registratore a cassette

Regolatore di tensione incorporato

Alimentazione 5 V, 800 mA max (*)

Circuito stampato professionale doppia faccia in vetronite

(*) Per alimentarlo basta una tensione raddrizzata e filtrata compresa fra 7 e 12 V in grado di fornire 1000 mA.

(**) **IMPORTANTE:** *Il kit è comprensivo di una speciale garanzia per cui in caso di mal funzionamento o insuccesso nella realizzazione è possibile inviare la piastra, con tutti i componenti, al costruttore, che la sostituirà con una montata e collaudata dietro il pagamento di una quota fissa di* **Lit. 50.000.**