Consiglio Nazionale delle Ricerche
ISTITUTO DI ELABORAZIONE DELL'INFORMAZIONE

CANE, SIMULCANE e SYSTEMCANE

A. Grasselli - G. Pacini

Nota Tecnica C70/1

(Gennaio 1970)

INDICE

1. Introduzione

Il calcolatore CANE (Calcolatore Automatico Numerico Educativo), descritto in questa nota, è un calcolatore ideale, che è stato definito per insegnare i concetti elementari della programmazione in codice macchina e della architettura dei calcolatori; in particolare, il CANE viene impiegato nel corso di "Teoria ed applicazioni delle macchine calcolatrici" di Scienze dell'Informazione. Un simulatore del CANE (programma SIMULCANE) esiste per l'IBM 7090: le sue modalità di impiego sono descritte in questa nota.

Perchè insegnare la programmazione in codice macchina? Non vi è dubbio alcuno che, prima o poi, gli studenti del corso di laurea in Scienze dell'Informazione debbano acquisire una certa esperienza in codice macchina. E' tutto sommato preferibile che questa esperienza venga il più presto possibile, perchè in caso contrario (se, cioè, il primo contatto con il calcolatore avviene al livello di un linguaggio simbolico) rimane una larga ed insoddisfacente zona d'ombra, che rende oltretutto difficile, se non addirittura impossibile, un discorso anche solo introduttivo sull'architettura delle macchine.

L'insegnamento introduttivo della programmazione in codice macchine dovrebbe rispettare le seguenti condizioni:

a) il calcolatore scelto dovrebbe essere "abbastanza reale";

b) si dovrebbe cercare di non terrorizzare il principiante con dettagli inessenziali al livello di una prima esposizione.

La prima condizione consiglierebbe l'adozione di un calcolatore reale, se non fosse per le considerazioni che seguono. In tutti i casi, è a nostro parere sconsigliabile impiegare un calcolatore ideale molto semplice (dieci istruzioni, per esempio), perchè la semplicità della struttura scelta si riflette invariabilmente in una artificiosa macchi

nosità dei programmi: con poche istruzioni nel codice operativo, anche il più semplice dei programmi (somma di n numeri!) richiede parecchie istruzioni, memorizzazione di costanti, ecc. Il rischio è quello di presentare allo studente una visione distorta della programmazione in codice macchina.

D'altra parte, in un calcolatore reale, le idiosincrasie delle istruzioni di entrata-uscita scoraggiano di solito anche l'iniziato; in più, le manipolazioni alla consolle presentano una certa difficoltà; infine, di solito la struttura ed il codice operativo, frutto di compromessi di ogni tipo, violano talvolta la condizione b) di cui sopra.

Si è quindi ritenuto preferibile definire un calcolatore come il CANE, con un codice operativo ricco di possibilità ma di facile comprensione. Anche la struttura del CANE è tuttavia frutto di compromessi: la memoria è troppo piccola, mancano le istruzioni in virgola mobile, e le unità periferiche sono ridotte al lettore di schede ed alla stampante. A proposito delle dimensioni della memoria, noteremo che esse sono dettate da un lato dalla lunghezza della parola (a sua volta scelta in 18 bit perchè è allora possibile scrivere una istruzione mediante 6 cifre ottali), e dall'altro dal desiderio di codificare ognuno dei tre campi dell'istruzione mediante una o più cifre ottali: si noti che al livello di esercizi di programmazione elementare le dimensioni della memoria possono senz'altro essere giudicate adeguate. E' comunque possibile aumentare le dimensioni della memoria, riducendo a tre il numero di registri indice, ed usando un bit del campo indice per indicare indirizzamento diretto od indiretto; è anche possibile introdurre le istruzioni in virgola mobile, raccogliendo tutti gli spostamenti in un'unica istruzione (non indiciabile). Queste variazioni, accompagnate dalla introduzione di nastri magnetici, potranno dar luogo in futuro ad un fratello maggiore del CANE, il SUPERCANE, che dovrebbe a nostro parere es-

sere illustrato dopo il CANE nella progressione didattica. Gli autori saranno comunque grati ai lettori di questa nota per i commenti, soprattutto negativi, che essi vorranno inoltrare.

Concluderemo questa introduzione con un avvertimento: questa nota verrà distribuita anche agli studenti del corso di "Teoria ed applicazioni delle macchine calcolatrici". Tuttavia, la nota non è stata scritta per il principiante, che è in tutti i casi tenuto a frequentare le esercitazioni, ed alcuni passaggi del testo possono risultare, nel momento attuale, di comprensione problematica. Consigliamo la pazienza: il testo, si spera, diventerà cristallino dopo aver fatto abbaiare il CANE alcune volte. La nota non va intesa come una guida alla programmazione del CANE, ma come un punto di riferimento sul codice operativo e sulle modalità di impiego del software.

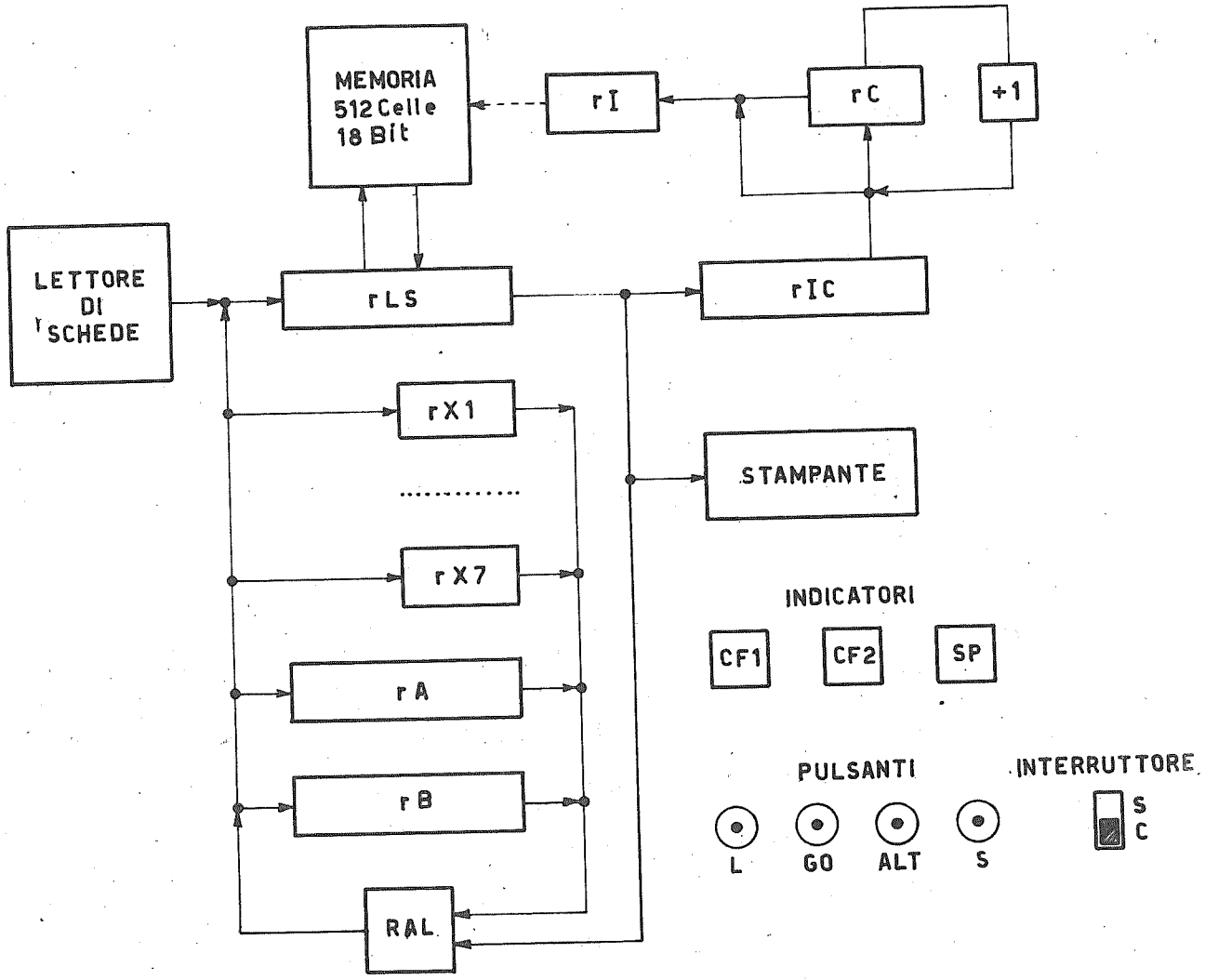
2. La struttura del CANE

L'architettura del CANE è illustrata dallo schema a blocchi della Fig. 1. La parola è di 18 bit, e la memoria consta di 512 celle. Il calcolatore possiede due registri operativi (accumulatori) rA ed rB, e 7 registri indice rX1, rX2, ...,rX7; rB funge da estensione di rA nelle operazioni di moltiplicazione e divisione. Vi sono due indicatori di confronto, CF1 e CF2, ed un indicatore di supero di capacità degli accumulatori.

I numeri negativi sono rappresentati in complemento a 2.

Al CANE sono collegate due sole unità periferiche: un lettore di schede, ed una stampante.

La consolle del CANE possiede quattro pulsanti, denominati L (lettura), GO (avvio), ALT, S(singolo), ed un interruttore a due posizioni SC (singolo/continuo).



rLS registro di lettura e scrittura
rI " indirizzamento
rC " contatore istruzioni
rIC " istruzione corrente
CF1,CF2 indicatori (flip-flop) di confronto
SP indicatore (flip-flop) di supero

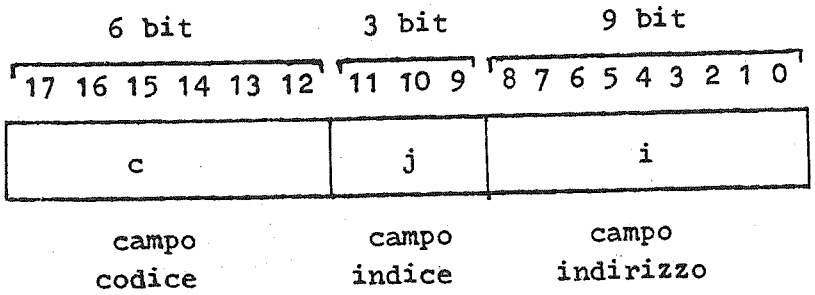
rA registro accumulatore
rB " moltiplicatore-quoziente
rX1,...,rX7 registri indice
RAL rete aritmetico-logica

Fig. 1 - Architettura del CANE.

Per quanto riguarda l'impiego dei registri indice, si noti che essi funzionano mod $777_{(8)}$: perciò, se nel calcolo dell'indirizzo effettivo (rXj)=770 ed i=333, (rXj)+i=770+333=323.

3. Le istruzioni

Tutte le istruzioni del CANE occupano una parola; il profilo dell'istruzione è il seguente:



Il contenuto del campo codice, campo indice e campo indirizzo, e, più in generale, contenuti di celle, registri, indirizzi, ecc., verranno nel seguito indicati in ottale, senza menzione esplicita della base.

Nel seguito, il contenuto di un registro o di una cella di memoria verrà indicato mettendo fra parentesi il nome del registro o l'indirizzo della cella di memoria: così, (rA) indica il contenuto del registro rA, (i) il contenuto della cella di indirizzo i, (i+(rX4)) il contenuto della cella il cui indirizzo viene ottenuto sommando i al contenuto del registro indice rX4. Inoltre, con $(X)_y$ (dove: X= un registro od un indirizzo, y = c oppure j oppure i) si indicherà il contenuto del campo codice, indice od indirizzo del registro o della cella X: per esempio, $(i)_i$ indica il contenuto del campo indirizzo della cella di indirizzo i. Infine, con (rA, rB) si indicherà il contenuto dei registri rA ed rB considerati come un unico registro (nella moltiplicazione e divisione).

3.1 Istruzioni che si riferiscono alla memoria

Queste istruzioni trasferiscono il contenuto di una cella di memoria in uno dei registri (TRA,TRAN,TRB,TRX), memorizzano il contenuto di un registro (MEA,MEB,MEX) od una costante (MEZ), eseguono operazioni aritmetiche (ADA,SOA,ADB,SOB,MOL,DIV) o logiche (COP,AND,OR,ORX) per le quali uno degli operandi è in memoria.

codice ottale	mnemonico	operazione
01	TRA trasf. in rA	(rA) ← (i + (rXj))
02	TRAN trasf. in rA negativo	(rA) ← -(i + (rXj))
05	TRB trasf. in rB	(rB) ← (i + (rXj))
06	TRX trasf. in rXj	$(rXj) \leftarrow (i)_i$
10	MEA memorizzazione di rA	(i+(rXj)) ← (rA)
11	MEB memorizzazione di rB	(i+(rXj)) ← (rB)
12	MEZ memorizzazione di 0	(i+(rXj)) ← 00 ... 0
13	MEX memorizzazione di rXj	$(i)_i \leftarrow (rXj)$, vedi testo
14	ADA addizione ad rA	(rA) ← (rA)+(i+(rXj)),vedi testo

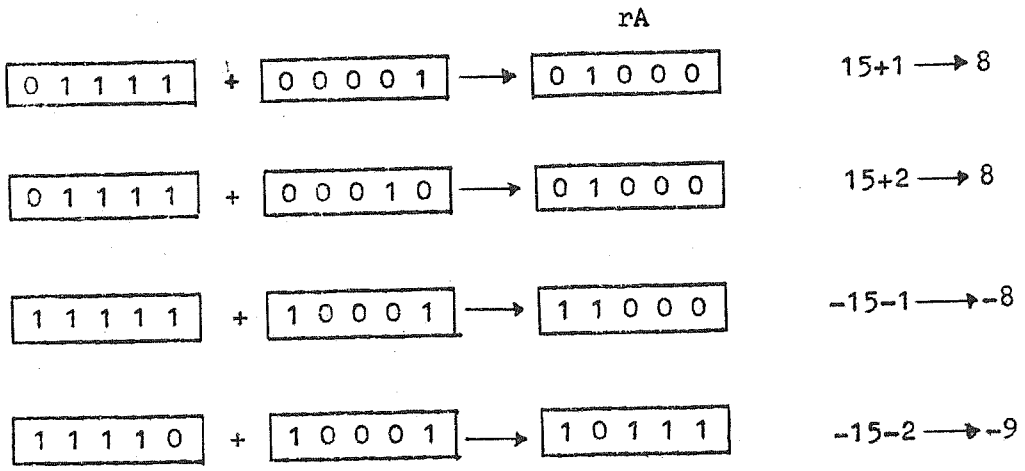
./.

codice ottale	mnemonico	operazione
15	SOA sottrazione da rA	$(rA) \leftarrow (rA)-(i+(rXj))$, vedi testo
20	ADB addizione ad rB	$(rB) \leftarrow (rB)+(i+(rXj))$, vedi testo
21	SOB sottrazione da rB	$(rB) \leftarrow (rB)-(i+(rXj))$, vedi note
22	ADX addizione ad rXj	$\left\{ \begin{array}{l} (rXj) \leftarrow (rXj)+(i)_i \\ \text{se } (rXj)=0,\ (rC) \leftarrow (rC)+2 \end{array} \right\}$ vedi note
23	SOX sottrazione da rXj	$\left\{ \begin{array}{l} (rXj) \leftarrow (rXj)-(i)_i \\ \text{se } (rXj)=0,\ (rC) \leftarrow (rC)+2 \end{array} \right\}$ vedi note
26	MOL moltiplicazione	$(rA,rB) \leftarrow (rB) \times (i+(rXj))$, vedi note
27	DIV divisione	$\left\{ \begin{array}{l} (rB) \leftarrow (rA,rB)/(i+(rXj)) \\ (rA) \leftarrow \text{resto} \end{array} \right\}$ vedi note
30	COP complemento	$(i+(rXj)) \leftarrow \overline{(i+(rXj))}$
31	AND prodotto logico	$(rA) \leftarrow (rA) \cap (i+(rXj))$
32	OR somma logica	$(rA) \leftarrow (rA) \cup (i+(rXj))$
33	ORX somma mod 2	$(rA) \leftarrow (rA) \oplus (i+(rXj))$

Note sulle istruzioni precedenti:

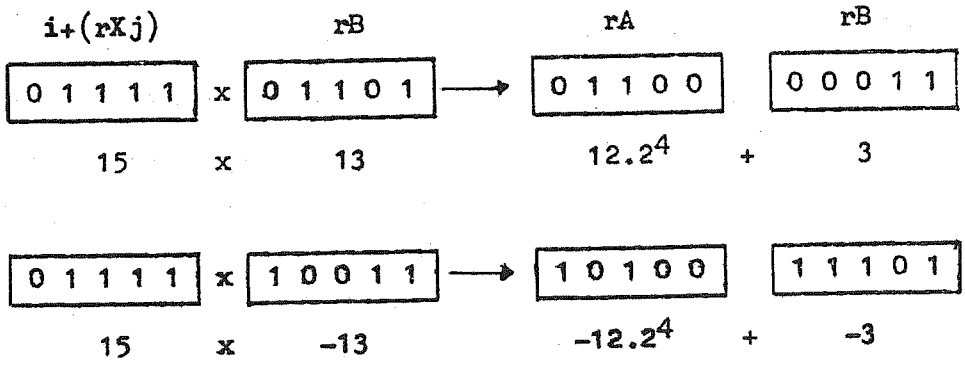
a) MEX: l'esecuzione dell'istruzione lascia invariati i campi c ed j della cella di memoria interessata;

b) ADA, SOA, ADB, SOB: se nella esecuzione di queste istruzioni si genera un supero di capacità dell'accumulatore (rA o rB):
- viene settato l'indicatore di supero SP, (SP) ← 1;
- il risultato viene diviso per 2, spostando l'accumulatore di un posto verso destra ed inserendo il bit più significativo in modo opportuno. Si noti quindi che il bit meno significativo del risultato si perde. Esempi (per semplicità si considerano solo 5 bit):

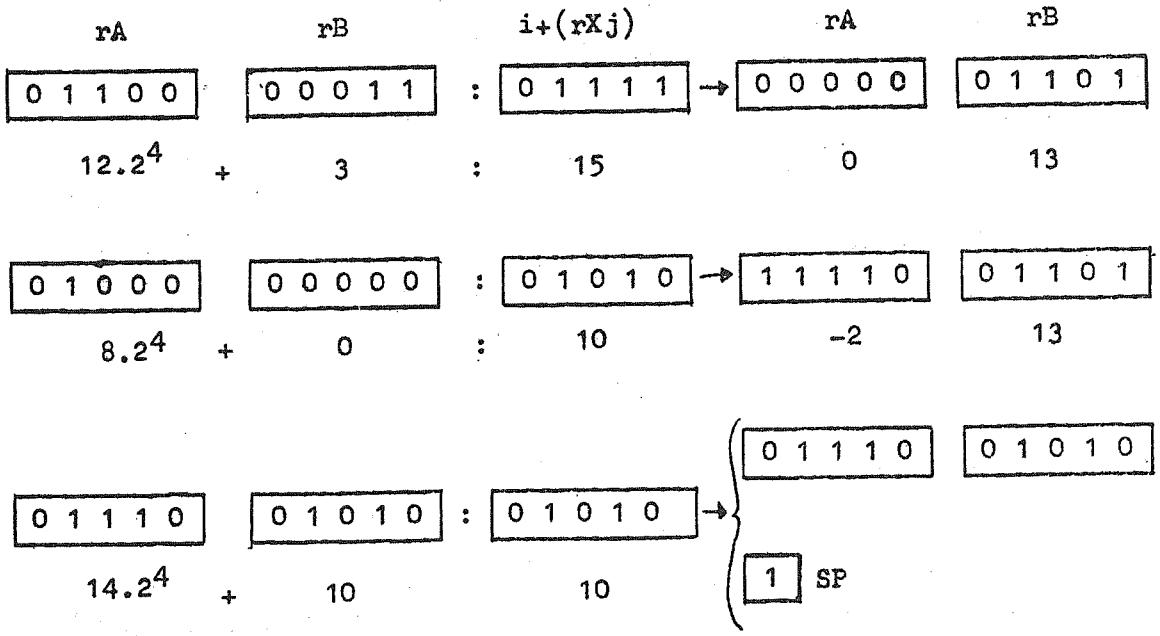


c) ADX, SOX: se, dopo l'esecuzione di queste istruzioni, il registro indice interessato contiene zero, viene eseguito uno skip, cioè viene saltata l'istruzione successiva ((rC) ← (rC)+2);

d) MOL: questa istruzione moltiplica il numero contenuto in rB per il numero in i+(rXj); la metà più significativa del prodotto va in rA, e la metà meno significativa in rB. Esempi (per semplicità si considerano solo 5 bit):



e) DIV: questa istruzione divide il numero contenuto in (rA,rB) (metà più significativa in rA, metà meno significativa in rB) per il numero in i+(rXj); il quoziente va in rB, ed il resto va in rA. Se $|(rA)| \geq |(i+(rXj))|$, la divisione non viene eseguita, e viene settato l'indicatore di supero, (SP) ←1. Esempi:



3.2 Istruzioni dirette

Queste istruzioni trasferiscono in un registro, o sommano e sottraggono al contenuto di un registro il contenuto del campo indirizzo dell'istruzione stessa (TDX, ADDX,SODX), o il contenuto del campo indirizzo più il contenuto del registro indice specificato (TDA,TDAN,ADDA,SODA).

codice ottale	mnemonico	operazione
03	TDA trasf. diretto in rA	(rA) ← i+(rXj)
04	TDAN trasf. diretto in rA negativo	(rA) ← -i-(rXj)
07	TDX trasf. diretto in rXj	(rXj) ← i
16	ADDA somma diretta ad rA	(rA) ← (rA)+i+(rXj), vedi note
17	SODA sottrazione diretta ad rA	(rA) ← (rA)-i-(rXj), vedi note
24	ADDX somma diretta ad rXj	{ (rXj) ← (rXj)+i se (rXj)=0, (rC) ← (rC)+2 } vedi note
25	SODX sottrazione diretta ad rXj	{ (rXj) ← (rXj)-i se (rXj)=0, (rC) ← (rC)+2 } vedi note

Note sulle istruzioni precedenti:

a) ADDA, SODA: il caso di supero di capacità viene trattato come illustrato nel paragrafo precedente;

b) ADDX, SODX: se dopo l'operazione (rXj)=0, viene effettuato lo skip.

3.3 Istruzioni di confronto

Lo scopo di queste istruzioni è quello di settare gli indicatori di confronto CF1 e CF2 in base al confronto fra il contenuto di un registro ed il contenuto di una cella di memoria. Gli indicatori di confronto vengono interrogati dalle istruzioni di salto (vedi 3.4). Nella tabella seguente, rY indica rA, oppure rB.

codice ottale	mnemonico	operazione
34	CFA confronto con rA	$\left\{\begin{array}{l}(CF1)\leftarrow 1 \text{ se } (rY)\geq(i+(rXj))\\ \quad\leftarrow 0 \text{ se } (rY)<(i+(rXj))\end{array}\right.$
35	CFB confronto con rB	$\left\{\begin{array}{l}(CF2)\leftarrow 1 \text{ se } (rY)\leq(i+(rXj))\\ \quad\leftarrow 0 \text{ se } (rY)>(i+(rXj))\end{array}\right.$
36	CFX confronto con rXj	$\left\{\begin{array}{l}(CF1)\leftarrow 1 \text{ se } (rXj)\geq(i)_i\\ \quad\leftarrow 0 \text{ se } (rXj)<(i)_i\\ (CF2)\leftarrow 1 \text{ se } (rXj)\leq(i)_i\\ \quad\leftarrow 0 \text{ se } (rXj)>(i)_i\end{array}\right.$
37	CFXD confronto con rXj diretto	$\left\{\begin{array}{l}(CF1)\leftarrow 1 \text{ se } (rXj)\geq i\\ \quad\leftarrow 0 \text{ se } (rXj)<i\\ (CF2)\leftarrow 1 \text{ se } (rXj)\leq i\\ \quad\leftarrow 0 \text{ se } (rXj)>i\end{array}\right.$

3.4 Istruzioni di salto

Queste istruzioni rimandano all'esecuzione dell'istruzione in i+(rXj) incondizionatamente (SLT,SUB), o condizionatamente agli indicatori (SSP,SMIN,SMAG,SUG) o ai segni dei registri rA ed rB (SAN,SAP,SAZ, SBN,SBP,SBZ).

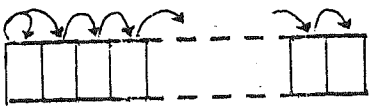
codice ottale	mnemonico	operazione
40	SLT salto	(rC) ← i+(rXj)
41	SUB salto a sotto-programma	(rXj) ← (rC)+1 (rC) ← i
42	SSP salto se supero	se (SP)=1, (rC) ← i+(rXj) (SP) ← 0
43	SMIN salto se minore	se (CF1)=0, (CF2)=1 ; (CF1)=1, (CF2)=0 ; (CF1)=(CF2)=1 , (rC) ← i+(rXj) vedi nota
44	SMAG salto se maggiore	
45	SUG salto se uguale	
46	SAN salto se (rA) negativo	se (rA) < 0 ; > 0 ; = 0 , (rC) ← i+(rXj)
47	SAP salto se (rA) positivo	
50	SAZ salto se (rA) zero	

codice ottale	mnemonico	operazione
51	SBN salto se (rB) negativo	se (rB) $\begin{cases} < 0 \\ > 0 \\ = 0 \end{cases}$, (rC) ← i+(rXj
52	SBP salto se (rB) positivo	
53	SBZ salto se (rB) zero	

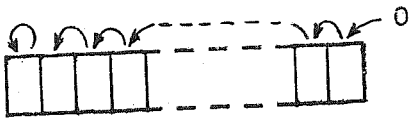
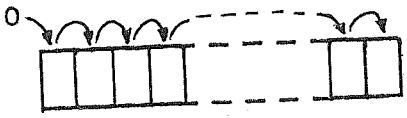
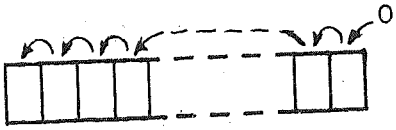
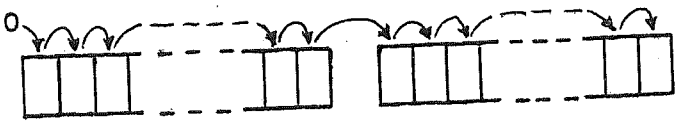
Nota: dopo l'esecuzione delle istruzioni SMIN, SMAG, SUG, lo stato degli indicatori CF1 e CF2 non è variato.

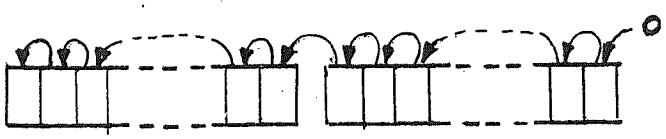
3.5 Istruzioni di spostamento

Queste istruzioni spostano di i+(rXj) posti il contenuto dei registri rA ed rB.

codice ottale	mnemonico	operazione
54	AVD rA aritmetico verso destra	 rA oppure rB ripetuto i+(rXj) volte
60	BVD rB aritmetico verso destra	

./.

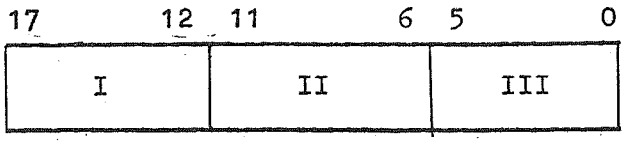
codice ottale	mnemonico	operazione
55	AVS rA aritmetico verso sinistra	 rA oppure rB ripetuto i+(rXj) volte
61	BVS rB aritmetico verso sinistra	
56	ALD rA logico verso destra	 rA ripetuto i+(rXj) volte
57	ALS rA logico verso sinistra	 rA ripetuto i+(rXj) volte
62	ABLD rA, rB logico verso destra	 rA rB ripetuto i+(rXj) volte

codice ottale	mnemonico	operazione
63	ABLS rA,rB logico verso sinistra	 rA rB ripetuto i+(rXj) volte

3.6 Istruzioni speciali

Queste istruzioni muovono caratteri fra rA e le celle di memoria (TCA,MCA), oppure trasformano rappresentazioni binarie in decimale e viceversa (CAR, NUM).

Nella tabella seguente, gli indici I, II e III indicano rispettivamente le prime 6 posizioni dalla sinistra, le seconde 6, e le ultime 6 di una cella di memoria o del registro rA; cioè:



Inoltre, $(rX1)_u$ indica gli ultimi due bit del registro indice rX1.

codice ottale	mnemonico	operazione
64	TCA trasferimento carattere	$(rA) \leftarrow 00 \ldots 0$ se $(rX1)_u =$ { 01: $(rA)_{III} \leftarrow (i+(rXj))_I$; 10: $(rA)_{III} \leftarrow (i+(rXj))_{II}$; 11: $(rA)_{III} \leftarrow (i+(rXj))_{III}$; 00: nessuna operazione
65	MCA memorizzazione carattere	se $(rX1)_u =$ { 01: $(i+(rXj))_I \leftarrow (rA)_{III}$; 10: $(i+(rXj))_{II} \leftarrow (rA)_{III}$; 11: $(i+(rXj))_{III} \leftarrow (rA)_{III}$; 00: $(i+(rXj)) \leftarrow$ tre spazi
66	CAR conversione in caratteri	il valore assoluto del numero in rA, convertito in decimale, viene scritto sotto forma di 6 cifre decimali in codice caratteri in rA (3 cifre più significative) ed rB (3 cifre meno significative).
67	NUM conversione in numero binario	le 6 cifre decimali in codice caratteri contenute in rA ed rB vengono trattate come numero positivo, che viene convertito in binario e scritto in rA. Se la conversione dà luogo a supero di capacità di rA, $(SP) \leftarrow 1$; ad operazione eseguita, se vi è stato supero, rA contiene le 18 cifre binarie meno significative.

3.7 Istruzioni di entrata-uscita

L'unità di entrata del CANE è un lettore di schede a 72 colonne, e l'unità di uscita una stampante a 120 colonne. Vi sono due modalità di entrata: binaria, e alfanumerica. Nella modalità binaria, la colonna generica può essere "bianca", oppure contenere uno 0 oppure un 1: tutti gli altri caratteri sono considerati errore, e la macchina si ferma sulla istruzione di lettura interessata. Lo spazio e lo 0 hanno uguale significato. Nella modalità binaria, il contenuto della scheda (72 colonne) viene letto in 4 celle successive: le prime 18 colonne nella prima delle 4 celle, ..., le ultime 18 colonne nell'ultima delle 4 celle.

Il codice di rappresentazione dei caratteri alfanumerici all'interno del CANE è quello della Tabella 1 dell'Appendice I. Una parola del CANE ospita quindi 3 caratteri. Nella modalità di lettura alfanumerica, il contenuto della scheda (72 colonne) viene letto in 24 celle successive: le prime 3 colonne nella prima cella, ..., le ultime 3 colonne nella ventiquattresima.

Vi è una sola modalità di uscita (alfanumerica), nella quale 120 caratteri, memorizzati in un gruppo di 40 celle successive, costituiscono una riga di stampa.

Le istruzioni di entrata-uscita vengono iniziate dalla unità di controllo, ma completate dalla unità di entrata e di uscita. Perciò, dopo aver iniziato una istruzione di entrata o di uscita, il calcolatore continua con le successive istruzioni del programma anche se l'esecuzione dell'istruzione non è completata. Naturalmente, se una nuova istruzione dello stesso tipo viene incontrata nel programma, il calcolatore si ferma fino a che l'esecuzione dell'istruzione precedente non è stata completata. Apposite istruzioni di controllo permettono di verificare il completamento della istruzione di entrata e della istruzione di uscita.

codice ottale	mnemonico	operazione
72	ENB entrata binaria	legge una scheda (72 colonne) binaria nelle 4 celle i+(rXj), i+(rXj)+1, i+(rXj)+2, i+(rXj)+3. La macchina si ferma se in una delle colonne della scheda è perforato un carattere diverso da: 0, 1, spazio.
73	ENA entrata alfanumerica	legge una scheda alfanumerica (72 colonne) nelle 24 celle i+(rXj), i+(rXj)+1,...,i+(rXj)+23.
75	CEN controllo entrata	se il lettore di schede non ha completato la lettura, (rC) ← i+(rXj).
74	USC uscita	scrive una riga di 120 caratteri sulla stampante; i 120 caratteri sono nelle celle i+(rXj), i+(rXj)+1, ..., i+(rXj)+39.
76	CUS controllo uscita	se la stampante non ha completato la stampa, (rC) ← i+(rXj).

3.8 Istruzioni di controllo

codice ottale	mnemonico	operazione
00	ALT	(rC) ← i+(rXj) la macchina si ferma
70	ESG esegui	viene eseguita l'istruzione nella cella i+(rXj)
77	NOP nessuna operazione	non viene eseguita alcuna operazione

Nota: le istruzioni TRX, TDX, MEX, ADX, SOX, ADDX, SODX, CFX, CFXD con j=0 hanno lo stesso effetto di NOP.

4. La consolle

Nel funzionamento normale, l'interruttore SC (singolo/continuo) si trova nella posizione C (continuo). Premendo il pulsante GO (avvio), la macchina inizia l'esecuzione del programma, prendendo come prima istruzione quella contenuta nella cella specificata dal contenuto del registro contatore istruzioni rC. Premendo ALT, la macchina si ferma dopo aver completato l'esecuzione dell'istruzione corrente.

Premendo il pulsante L (lettura) a macchina ferma, viene letta una scheda nella modalità binaria, nelle celle 000, 001, 002 e 003, e dopo

la lettura la macchina inizia ad eseguire il programma con l'istruzione nella cella 000. Normalmente, le quattro istruzioni così caricate sono le istruzioni iniziali di un programma di caricamento (vedi il capitolo 5).

Il pulsante L non ha alcun effetto quando il calcolatore è in attività.

Se l'interruttore SC si trova nella posizione S (singolo), il calcolatore esegue una istruzione del programma tutte le volte che viene premuto il pulsante S(singolo).

La consolle possiede indicatori luminosi che mostrano il contenuto di tutti i registri ed indicatori della macchina.

5. Software di servizio

Per il momento, il corredo di software del CANE comprende i seguenti programmi di servizio:

a) caricatore di programmi scritti in binario,
b) caricatore di programmi scritti in ottale,
c) sottoprogramma di stampa del contenuto di 24 celle,
d) sottoprogramma di stampa del contenuto dei registri,
e) sottoprogramma di lettura e scrittura di numeri decimali con segno.

5.1 Osservazioni generali

I cinque programmi a)-e) sono costituiti ognuno da 48 istruzioni, e vengono sempre caricati nelle celle 004-063. Perciò, non è possibile impiegare contemporaneamente due o più programmi di servizio. In totale ognuno dei cinque programmi di servizio occupa complessivamente 92_{10} parole. Le 44_{10} parole dalla cella 064 alla 137 sono necessarie per i

buffer di ingresso-uscita o per memorizzazione temporanea.

Un programma di servizio viene caricato facendo leggere al CANE, mediante il pulsante L, una scheda che contiene le seguenti quattro istruzioni, codificate in binario (chiamata caricatore minimo):

```
TDX  1  720
ENB  1  064
ADDX 1  004
SLT  0  001
```

Si ricorderà che le quattro istruzioni vengono caricate nelle celle 000, 001, 002 e 003, ed il calcolatore esegue poi l'istruzione contenuta nella cella 000. Queste quattro istruzioni caricano 48 parole binarie, perforate su 12 schede, nelle celle 004, 005, ... , 063.

Si noti che per caricare un programma binario di n istruzioni (con n multiplo di 4) nelle celle 004,005,...,n+3, ed eseguire il programma appena terminato il caricamento, le 4 istruzioni precedenti debbono essere modificate come segue:

```
TDX  1  777-(n-1)
ENB  1  n+4
ADDX 1  004
SLT  1  001
```

(1)

Poichè il calcolatore esegue lo skip sulla istruzione ADDX quando (rX1)=0, e salta all'esecuzione dell'istruzione in 004, è possibile che sia eseguita, prima che il calcolatore ne abbia ultimato la lettura, una delle istruzioni contenute nell'ultima scheda, oppure una istruzione il cui corretto funzionamento sia subordinato in un modo qualsiasi alla presenza in memoria di una delle parole contenute nell'ultima scheda. E' quindi necessario verificare che il programma sia tale da permettere il comple-

tamento della lettura dell'ultima scheda prima di usufruire di una delle istruzioni in essa contenute.

Qualora la verifica dia esito negativo o si presenti troppo laboriosa, è consigliabile inserire una istruzione controllo entrata CEN con indirizzo di salto a se stessa in posizione opportunamente scelta.

Le quattro istruzioni (1) caricano un programma di n istruzioni nelle celle 004,...,n+3. Questo sistema però può essere usato solamente per caricare programmi perforati in binario e per di più in una maniera che manca della necessaria elasticità. Infatti:

a) non è possibile caricare un programma in zone diverse da quella indicata: in particolare, non è possibile caricare insieme al programma dei sottoprogrammi, a meno che tali sottoprogrammi non vengano di volta in volta riscritti per adattarli al programma chiamante;

b) il numero n di istruzioni da caricare deve essere noto; infatti esso compare come una delle componenti degli indirizzi delle 4 istruzioni (1).

Si è quindi giudicato opportuno scrivere due caricatori, uno per programmi perforati in binario e l'altro per programmi perforati in ottale.

5.2 Il caricatore binario

Il caricatore binario legge schede contenenti tre istruzioni binarie, nel formato indicato nella fig. 2. Questo caricatore è del tipo "load and go": una scheda controllo segnala la fine del caricamento e l'indirizzo d'inizio del programma caricato; dopo la lettura della scheda controllo, il caricatore esegue un salto a tale indirizzo.

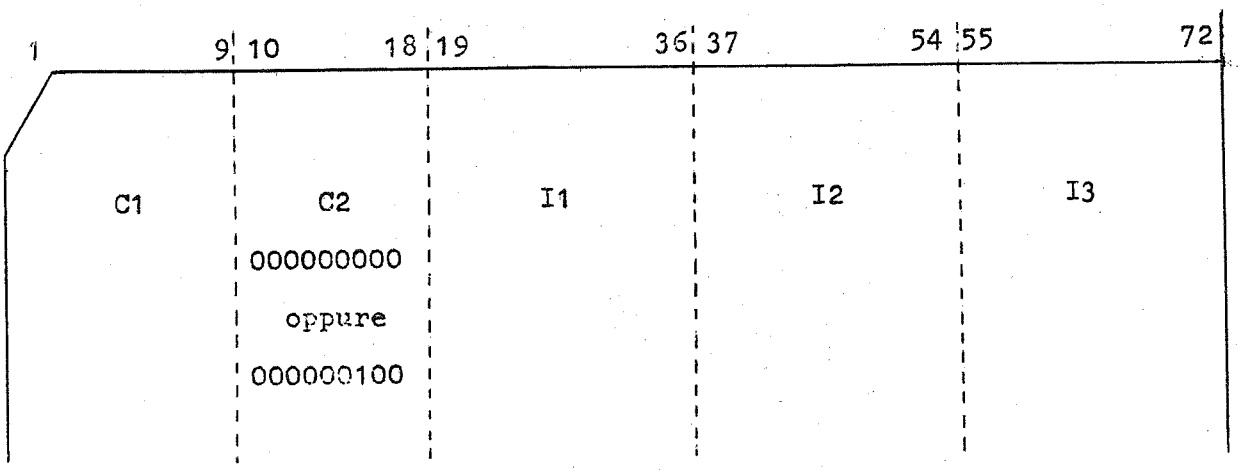


Fig. 2

Con riferimento alla fig. 2 (e ricordando che b e 0 vengono letti entrambi come 0):

a) se C2= 000000000 : scheda programma. Deve essere:
$140 \leq C1 \leq 775$. Le tre istruzioni I1, I2, I3 vengono caricate come segue:
(C1)$\leftarrow$ I1 , (C1+1)$\leftarrow$ I2 , (C1+2)$\leftarrow$ I3

b) se C2= 000000100 : scheda controllo (ultima scheda). Dopo aver letto tale scheda, (rC)$\leftarrow$ C1.

La fig. 3 mostra un esempio di scheda programma, e la fig. 4 un esempio di scheda controllo.

Il caricatore si ferma nei seguenti casi:

1) se C2 $\neq$ 000000000 e C2 $\neq$ 000000100 (ALT all'indirizzo 045);

2) se C2 = 000000000, e C1 $<$ 140 oppure C1 $>$ 775:
infatti, in questo caso il caricatore caricherebbe le istruzioni su se stesso (ALT all'indirizzo 050);

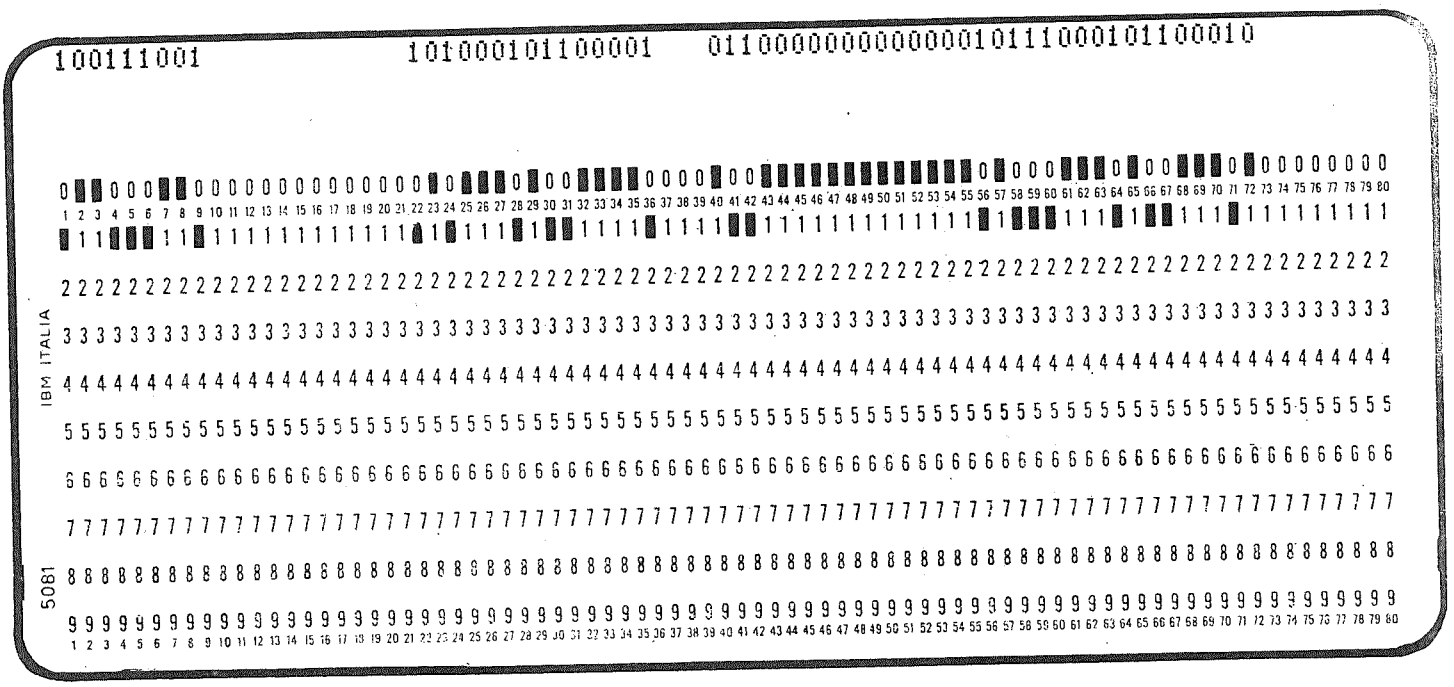


Fig. 3 - Esempio di scheda programma binaria.

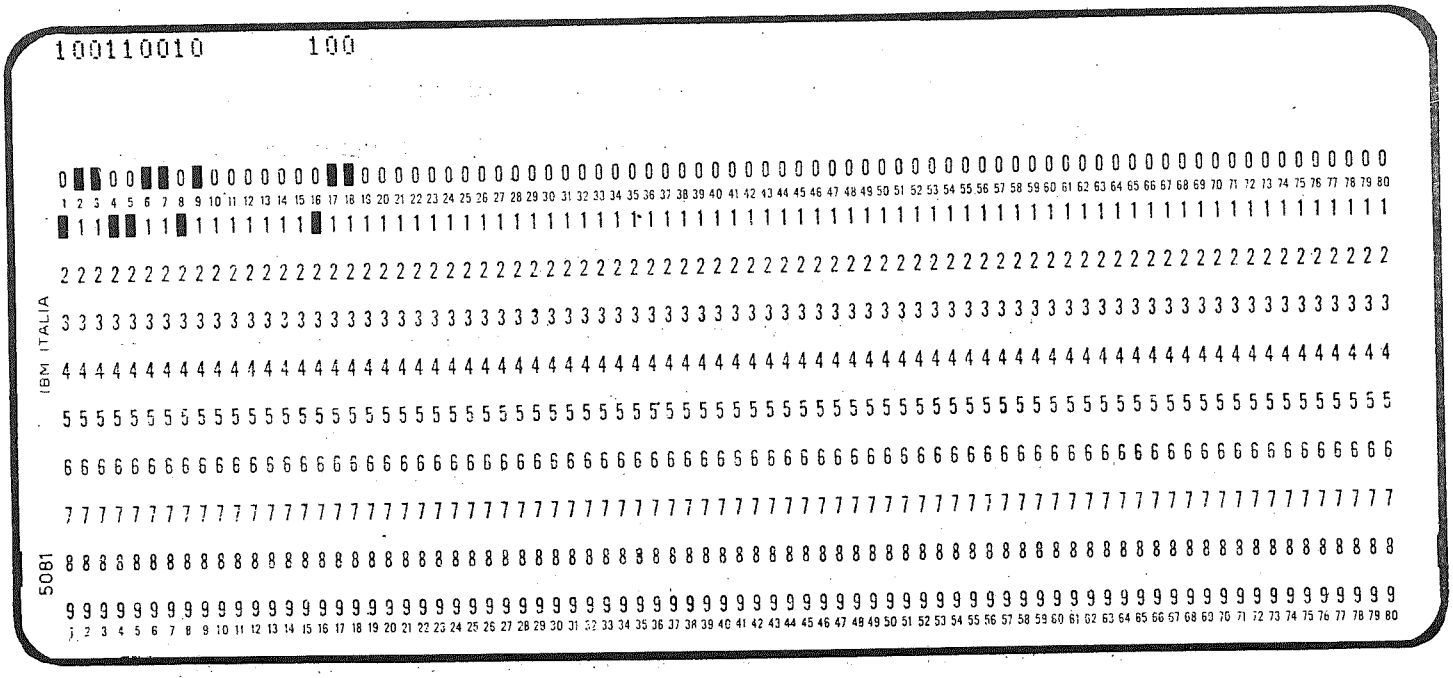


Fig. 4 - Esempio di scheda controllo binaria.

3) se viene letta una scheda contenente uno o più caratteri diversi da b, 0, 1.

Il programma caricato viene stampato dal caricatore (una scheda per riga).

Lo schema a blocchi del caricatore binario, ed il programma stesso, sono riportati nell'Appendice II.

5.3 Il caricatore ottale

Il caricatore ottale legge schede contenenti una istruzione ottale, nel formato indicato nella fig. 5. Anche questo caricatore è del tipo "load and go".

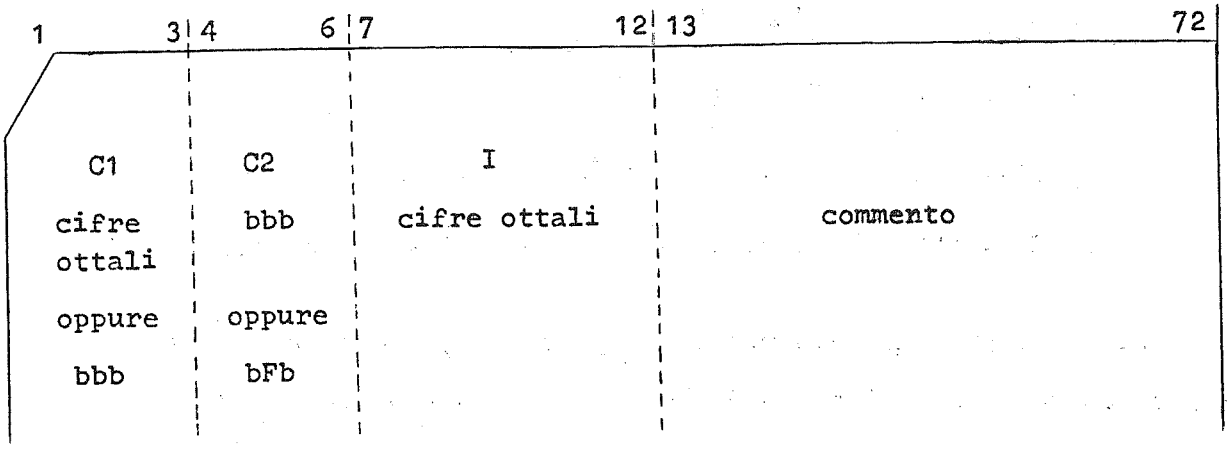


Fig. 5

Con riferimento alla fig. 5:

a) se C2 = bbb : scheda programma. Deve essere:

$140 \leq$ C1 (tre cifre ottali), oppure C1 = bbb. Se C1=3 cifre

ottali, l'istruzione I viene caricata in C1 ((C1)$\leftarrow$ I), altrimenti I viene caricata nella cella successiva a quella in cui è stata caricata l'istruzione precedente. La prima scheda del programma deve contenere un indirizzo in C1;

b) se C2 = bFb : scheda controllo (ultima scheda). Deve essere: C1 = 3 cifre ottali. Dopo aver letto tale scheda, (rC)$\leftarrow$C1.

La fig. 6 mostra un esempio di scheda programma, e la fig. 7 un esempio di scheda controllo.

Il caricatore si ferma nei seguenti casi (ALT all'indirizzo 042):

1) se C1 $\neq$ bbb e C1 $\neq$ 3 cifre ottali;

2) se C2 $\neq$ bbb e C2 $\neq$ bFb;

3) se I $\neq$ 6 cifre ottali;

4) se C2 = bbb e C1 $<$ 140;

5) se C2 = bFb e C1 = bbb;

6) se nella prima scheda è C1 = C2 = bbb;

7) se C1 = bbb e l'istruzione precedente è stata caricata in 777.

Il programma caricato viene stampato dal caricatore (una scheda per riga). Si noti che sulle schede le colonne 13-72 possono essere usate per scrivere commenti: questi vengono stampati assieme al resto della scheda.

Lo schema a blocchi del caricatore ottale, ed il programma stesso, sono riportati nell'Appendice II.

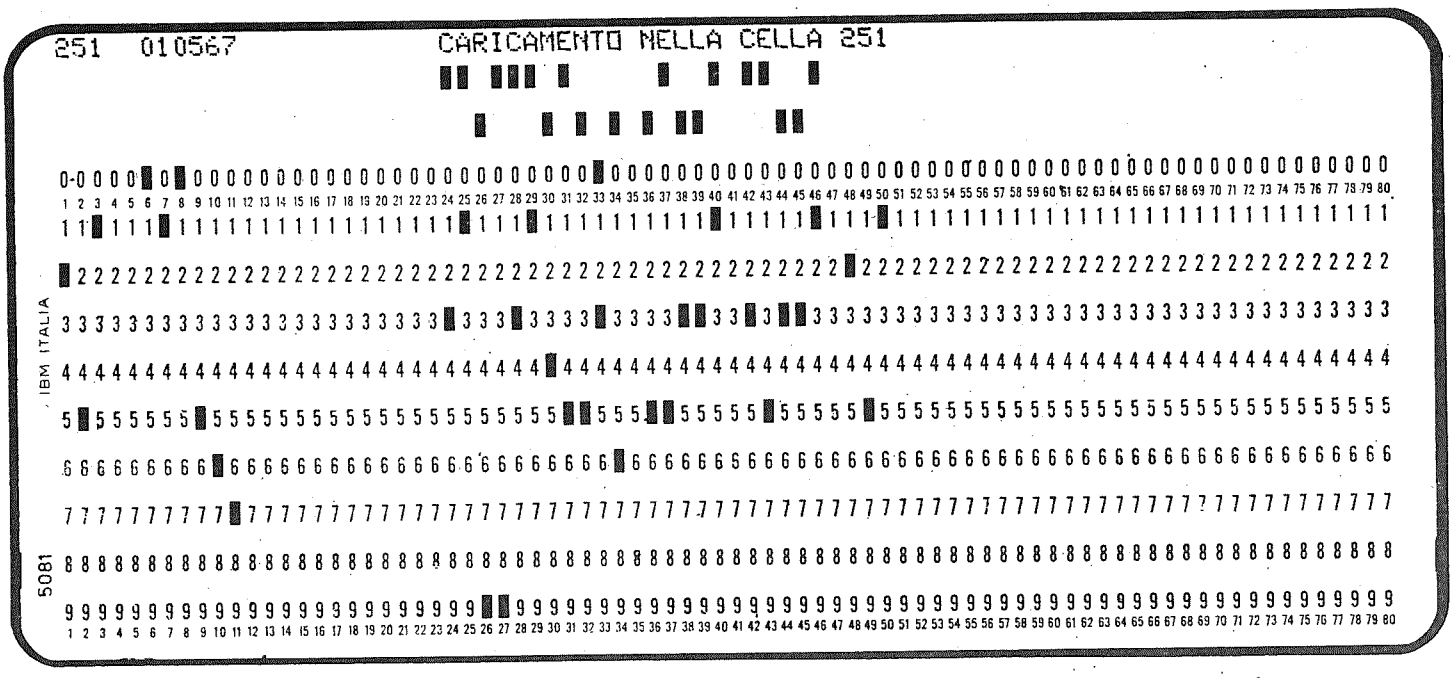


Fig. 6 - Esempio di scheda programma ottale.

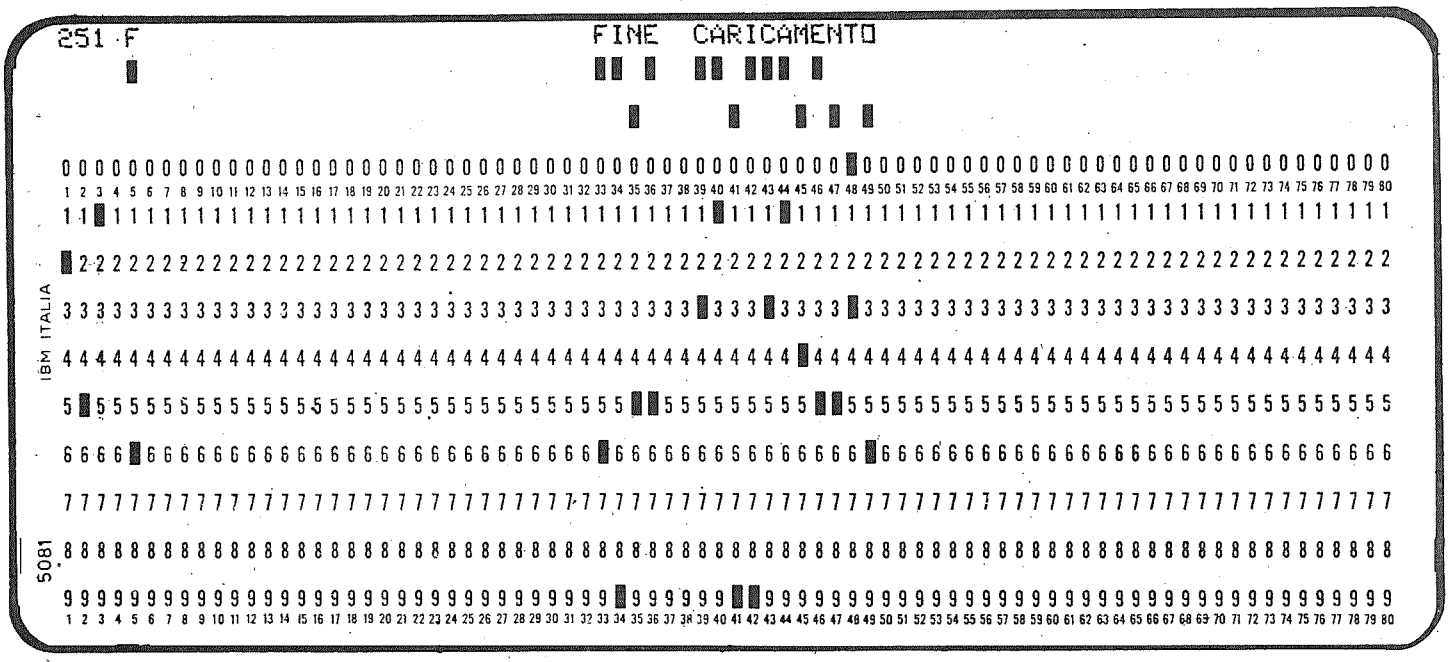


Fig. 7 - Esempio di scheda controllo ottale.

5.4 Impiego dei caricatori

Il caricatore minimo legge in memoria il caricatore e lo mette subito in funzione, poichè dopo la fine del ciclo del caricatore minimo la prima istruzione del caricatore (binario od ottale) si trova nella cella 004.

I due caricatori non possono essere usati contemporaneamente; possono però essere usati, e in modo vario, durante lo stesso programma.

Diamo alcuni esempi di composizione di un programma:

a) - Scheda asterisco (°)
   - Scheda caricatore minimo
   - Caricatore ottale
   - Programma ottale
   - Dati

b) - Scheda asterisco
   - Scheda caricatore minimo
   - Caricatore ottale
   - Programma ottale la cui scheda di fine caricamento (C2=bFb) contiene C1=000
   - Caricatore binario
   - Programma binario
   - Dati

c) - Scheda asterisco
   - Scheda caricatore minimo
   - Caricatore ottale

(°) Questa scheda è illustrata nei capitoli 6 e 7.

- Programma ottale che legge n schede e la cui ultima istruzione è:

  SLT 0 000

- n schede di dati
- Caricatore binario
- Programma binario.

5.5 Sottoprogramma di stampa di 24 celle

Stampa ad ogni chiamata i contenuti di 24 celle di memoria, 8 per riga di stampa. Le celle sono consecutive, e l'indirizzo della prima è specificato nella chiamata. Questo sottoprogramma lascia invariati tutti i registri salvo rX7; anche gli indicatori CF1 e CF2 vengono modificati.

Il sottoprogramma viene letto in memoria dal caricatore minimo. Il programma che vuole servirsi del sottoprogramma deve leggerlo con le istruzioni:

```
.........
SUB 7 000
CEN 0 *
.........
```

L'istruzione SUB salta alla prima istruzione del caricatore minimo. Quando si esce dal caricatore minimo, si esegue la prima istruzione del sottoprogramma in 004 che è:

```
SLT 7 000
```

e si salta perciò alla istruzione CEN del programma chiamante, che garantisce il completamento della lettura dell'ultima scheda del sottoprogramma.

Il sottoprogramma viene chiamato con l'istruzione:

SUB 7 005

La cella successiva alla chiamata contiene nella parte indirizzo la prima cella del gruppo di celle in cui si vuole ottenere la stampa.

5.6 Sottoprogramma di stampa dei registri

Stampa ad ogni chiamata il contenuto dei registri rA,rB,rX1,...,rX7. Lascia invariati tutti i registri, salvo rX7; modifica lo stato degli indicatori CF1 e CF2.

Questo sottoprogramma viene letto in memoria come illustrato in 5.5; esso viene chiamato con l'istruzione:

SUB 7 005

5.7 Sottoprogramma di entrata-uscita di numeri decimali

Il sottoprogramma ha due punti d'ingresso: A (lettura) e B (scrittura). Chiamando A, esso legge un numero di 6 cifre decimali e segno in uno dei campi indicati in Fig. 8; una successiva chiamata di A provoca la lettura del numero nel campo successivo della stessa scheda,

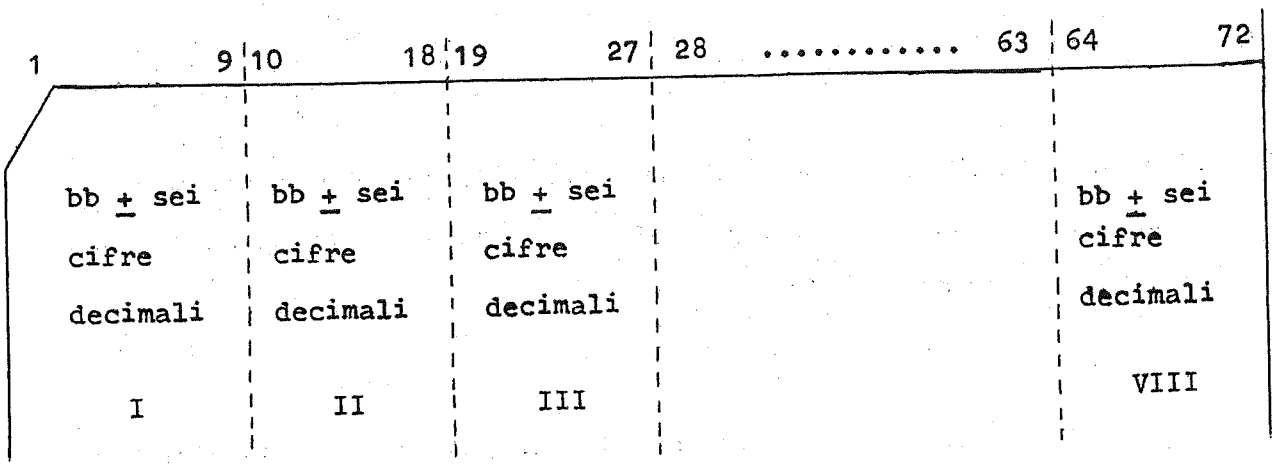


Fig. 8

oppure, se la scheda è stata letta completamente, del numero del primo campo della scheda successiva. Il salto alla scheda successiva viene effettuato automaticamente nei seguenti due casi:

a) dopo aver chiamato il punto di ingresso A, viene chiamato il punto d'ingresso B (scrittura);

b) i primi 3 caratteri di un campo sono bbb.

Il numero letto, convertito in binario, viene lasciato dal sottoprogramma in rA. Il numero letto deve essere compreso fra $-(2^{17}-1)$ e $+(2^{17}-1)$: in caso contrario, il registro rA contiene la parte meno significativa del numero convertito.

Chiamando il punto d'ingresso B, il sottoprogramma stampa il contenuto di rA nel formato:

± xxxxxx

dove gli "x" indicano cifre decimali.

Questo sottoprogramma viene letto in memoria come illustrato in 5.5; esso viene chiamato nel modo seguente:

punto A (entrata): SUB 7 034

punto B (uscita) : SUB 7 005

6. SIMULCANE e SYSTEMCANE

Il programma consta di due parti distinte:

1) simulatore del CANE (SIMULCANE)

2) programma supervisore (SYSTEMCANE).

Il programma supervisore non è essenziale per la simulazione ed ha il compito di rendere possibile l'esecuzione automatica di un numero qual-

siasi di programmi l'uno di seguito all'altro.

Il SIMULCANE è fornito di due programmi di servizio, uno di traccia e uno di immagine (dump). Il primo, se richiesto, stampa tutti i registri ed indicatori ogni volta che uno di essi viene modificato; il secondo, se richiesto, stampa una "immagine" finale completa della memoria (dump post-mortem).

Il simulatore è dotato di un orologio (ORG) e di un indicatore di supero orologio simulato (FTIME). L'orologio viene incrementato di una quantità opportuna dopo la esecuzione di ogni istruzione, e contiene nell'istante generico il numero di cicli trascorsi dall'inizio della simulazione del programma. Per semplicità, tutte le istruzioni durano 2 cicli, salvo MOL e DIV (10 cicli); è stato posto, formalmente, 1 sec = 300.000 cicli. Il completamento di una operazione di lettura o scrittura richiede:

lettura binaria	400 cicli
lettura alfanumerica	600 cicli
stampa	1000 cicli

Si noti che le operazioni di entrata-uscita sono, in tempo simulato, circa 50 volte più veloci che in realtà (lettura: 600 schede/min, e stampa: 600 righe al minuto): ciò è stato fatto con l'ovvia motivazione di risparmiare tempo macchina (reale!).

L'indicatore di supero orologio simulato FTIME contiene per ogni programma in corso di esecuzione il tempo massimo in cicli di esecuzione del programma stesso; quando il contenuto dell'orologio ORG raggiunge o supera il contenuto di FTIME:

$$(ORG) \geq (FTIME)$$

l'esecuzione del programma viene sospesa.

6.1 Struttura del pacco complessivo

Ogni programma CANE deve essere preceduto da una scheda iniziale (scheda asterisco) che ha il formato:

Colonne	1-6	* bbbbb
	7-51	Nome e cognome, nome del programma e annotazioni varie
	52-54	tre cifre decimali che vengono interpretate come richiesta del tempo in secondi
	64-66	TRC se si desidera la traccia
	70-72	IMG se si desidera l'immagine.

I programmi, ognuno con la sua scheda asterisco, vengono posti uno di seguito all'altro (°).

Il pacco complessivo deve essere preceduto da una scheda con il formato:

Colonne	1-9	nove cifre decimali
	10-12	bbb
	13-24	060000000000 oppure 060100000000

Il numero di nove cifre decimali scritto nelle colonne 1-9, che indicheremo con M, è interpretato come massimo numero di cicli da concedere ad ogni singolo programma del pacco indipendentemente dalle richieste contenute sulle schede asterisco. Più precisamente, suppo-

(°) Se le colonne 52-54 di una scheda asterisco contengono caratteri che non sono cifre decimali il relativo programma non viene eseguito.

niamo che una scheda asterisco richieda s secondi, ed indichiamo con S il numero di cicli richiesto:

$$S = 3.10^5 s$$

Al programma vengono concessi C cicli, dove:

$$C = \min(M,S)$$

All'inizio dell'esecuzione di un programma del pacco, perciò, SYSTEMCANE carica C nell'indicatore FTIME:

$$(FTIME) \leftarrow C.$$

Se le colonne 12-24 hanno la configurazione:

060000000000

la memoria ed i registri del CANE vengono azzerati all'inizio di ogni programma del pacco; se invece hanno la configurazione

060100000000

l'azzeramento non ha luogo.

L'ultima scheda del pacco deve essere del formato seguente:

Colonne 1-6 * FINE b

Il pacco deve venir preceduto dalla scheda $DATA.

La composizione del pacco è illustrata dalla fig. 9.

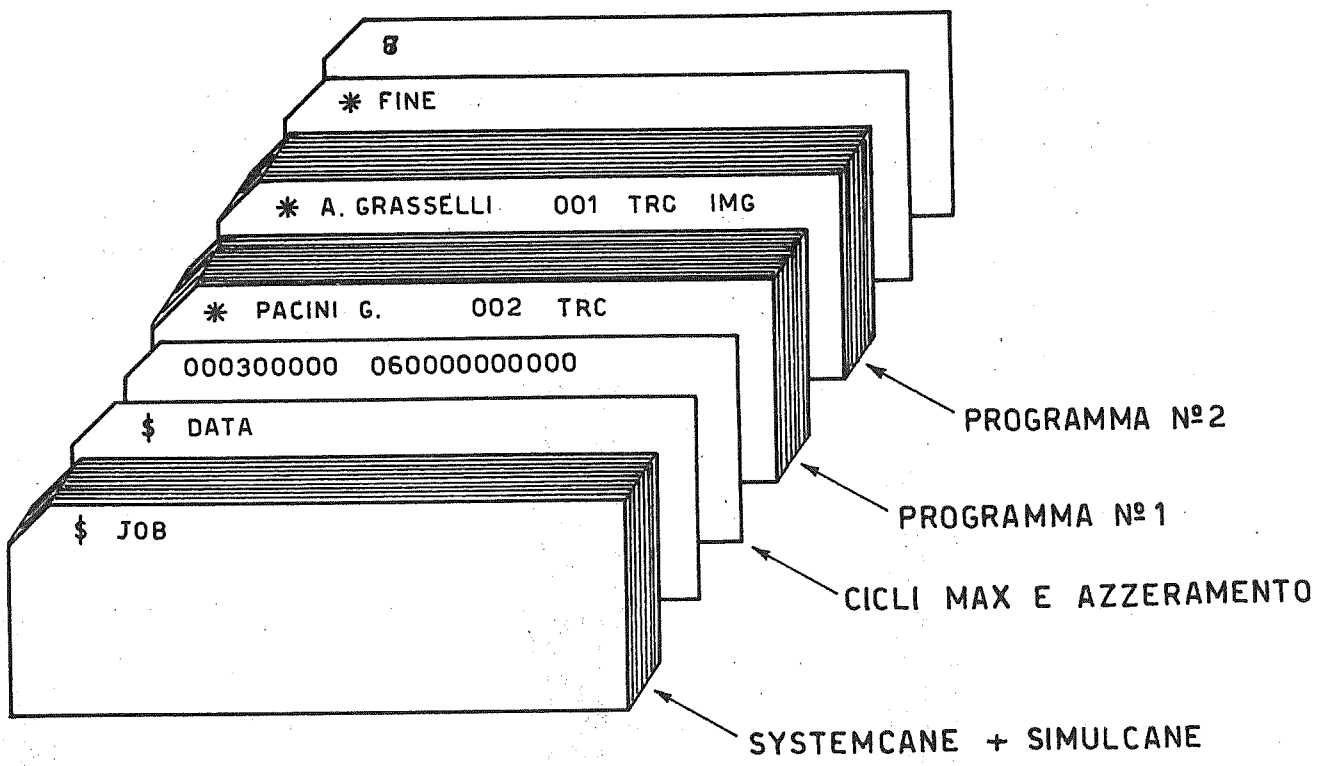


Fig. 9 - Struttura del pacco complessivo per il supervisore SYSTEMCANE.

6.2 Funzionamento del SYSTEMCANE

Il SYSTEMCANE effettua le operazioni seguenti:

1) riconoscimento scheda asterisco e stampa immediata della stessa;

2) analisi della scheda asterisco;

3) conseguente predisposizione degli indicatori di traccia, immagine e supero orologio simulato;

4) lettura di una scheda con la modalità binaria del CANE;

5) sistemazione delle quattro parole sulla scheda letta nelle celle 000, 001, 002 e 003;

6) azzeramento del registro contatore rC;

7) chiamata del programma SIMULCANE.

Le operazioni 4), 5), 6) e 7) simulano la esistenza del dispositivo L della consolle del CANE. Il riconoscimento della scheda *FINE provoca la stampa di una lista comprendente tutte le schede asterisco riconosciute a sinistra di ognuna delle quali appare, espressa in centesimi di secondo simulati, la durata del relativo programma. Se le schede asterisco sono più di 50, solo le prime 50 vengono stampate nella lista finale, i programmi invece sono eseguiti tutti normalmente.

Il simulatore si può fermare per i seguenti motivi:

1) istruzione di ALT;

2) supero orologio simulato;

3) istruzione di lettura quando siano terminate le schede dei dati;

4) lettura con modalità binaria di scheda contenente caratteri non binari.

Nel 1° caso viene stampata la dizione:

ALT ALLA LOC. ..... OROLOGIO = ... SEC

Nel 2° caso viene stampato:

TEMPO SCADUTO

Nel 3° caso viene stampato:

ALT PER FINE DEI DATI

Nel 4° caso viene stampato:

ERRORE IN SCHEDA BINARIA

Il verificarsi del 3° caso significa in realtà che è stata letta durante la simulazione di una istruzione di ingresso una scheda asterisco (o la scheda * FINE).

In tutti i quattro casi elencati, SIMULCANE chiama nuovamente SYSTEMCANE.

APPENDICE I

Carattere	Codice	Carattere	Codice
A	010001	0	000000
B	010010	1	000001
C	010011	2	000010
D	010100	3	000011
E	010101	4	000100
F	010110	5	000101
G	010111	6	000110
H	011000	7	000111
I	011001	8	001000
J	100001	9	001001
K	100010		
L	100011	b(spazio)	110000
M	100100	.	011011
N	100101	(	111100
O	100110	+	010000
P	100111	$	101011
Q	101000	*	101100
R	101001	)	011100
S	110010	-	100000
T	110011	/	110001
U	110100	,	001100
V	110101	=	001011
W	110110		
X	110111		
Y	111000		
Z	111001		

Tabella I - Codice caratteri

2^n	n	2^{-n}
1	0	1,0
2	1	0,5
4	2	0,25
8	3	0,125
16	4	0,0625
32	5	0,03125
64	6	0,015625
128	7	0,0078125
256	8	0,00390625
512	9	0,001953125
1024	10	0,0009765625
2048	11	0,00048828125
4096	12	0,000244140625
8192	13	0,0001220703125
16384	14	0,00006103515625
32768	15	0,000030517578125
65536	16	0,0000152587890625
131072	17	0,00000762939453125
262144	18	0,000003814697265625
524288	19	0,0000019073486328125
1048576	20	0,00000095367431640625

Tabella II - Potenze di 2.

n	8^n	16^n
0	1	1
1	8	16
2	64	256
3	512	4096
4	4096	65536
5	32768	1048576
6	262144	16777216

Tabella III - Potenze di 8 e di 16

APPENDICE II

Programmi e schemi a blocchi dei caricatori

In questa Appendice, sono riportati programmi e schemi a blocchi dei caricatori binario ed ottale. Per chiarezza, i programmi sono stati scritti in linguaggio assemblatore SYMBOLCANE (°): tale linguaggio verrà descritto in una nota futura, ma si ritiene che non vi siano difficoltà di comprensione per il non-principiante. Le pseudo-istruzioni del SYMBOLCANE che sono state impiegate in questi programmi sono:

ORIG (origine): stabilisce l'indirizzo nel quale viene caricata l'istruzione successiva;

BIS (blocco inizia con il simbolo): riserva un blocco di celle;

ALF (introduzione di caratteri alfanumerici): introduce tre caratteri alfanumerici in una cella.

1) Caricatore binario

```
        ORIG    4
        TDX     0,1
        TDX     40,2
LOOP4   MCA     BUFFER-1,2      } Buffer di uscita ← bbb
        SODX    1,2
        SLT     LOOP4
```

(°) I numeri nella parte indirizzo delle istruzioni sono scritti in notazione decimale.

```
START   ENB     BUFFIN          } lettura di una scheda
        CEN     *

        TDX     -4,2
        TDX     0,3
        CUS     *
LOOP1   TRB     BUFFIN+4,2
        TDX     2,4
LOOP3   TDX     3,5
LOOP2   ALS     3
        ABLS    3
        SODS    1,5
        SLT     LOOP2           } conversione per la stampa
        MEA     BUFFER,3
        ADDX    1,3
        SODX    1,4
        SLT     LOOP3
        ADDX    1,3
        ADDX    1,2
        SLT     LOOP1

        USC     BUFFER          stampa

        TRB     BUFFIN
        ABLS    9               } i ← C1
        MEA     BUFFIN
        TRX     BUFFIN,7

        CFB     FINE            } C2=004(8) ?
        SUG     FINIX
```

```
          CFB     * +2
          SUG     * +2          } C2=000(8) ?

          ALT     0             ALT

          CFXD    BUFFER+43,7   } i ≥ 140(8) ?
          SMAG    *+2           }

FINE      ALT     0,4           ALT

          CFXD    509,7         } i ≤ 775(8) ?
          SMAG    FINE          }

          MEX     LOOP5+1,7     }
LOOP5     TRA     BUFFIN+1,2    }
          MEA     **,2          } (i) ← I1
          ADDX    1,2           } (i+1) ← I2
          CFXD    3,2           } (i+2) ← I3
          SMIN    LOOP5         }
          SLT     START         }

FINIX     CUS     *             } (rC) ← I
          SLT     0,7           }

BUFFER    BIS     40

BUFFIN    BIS     4
```

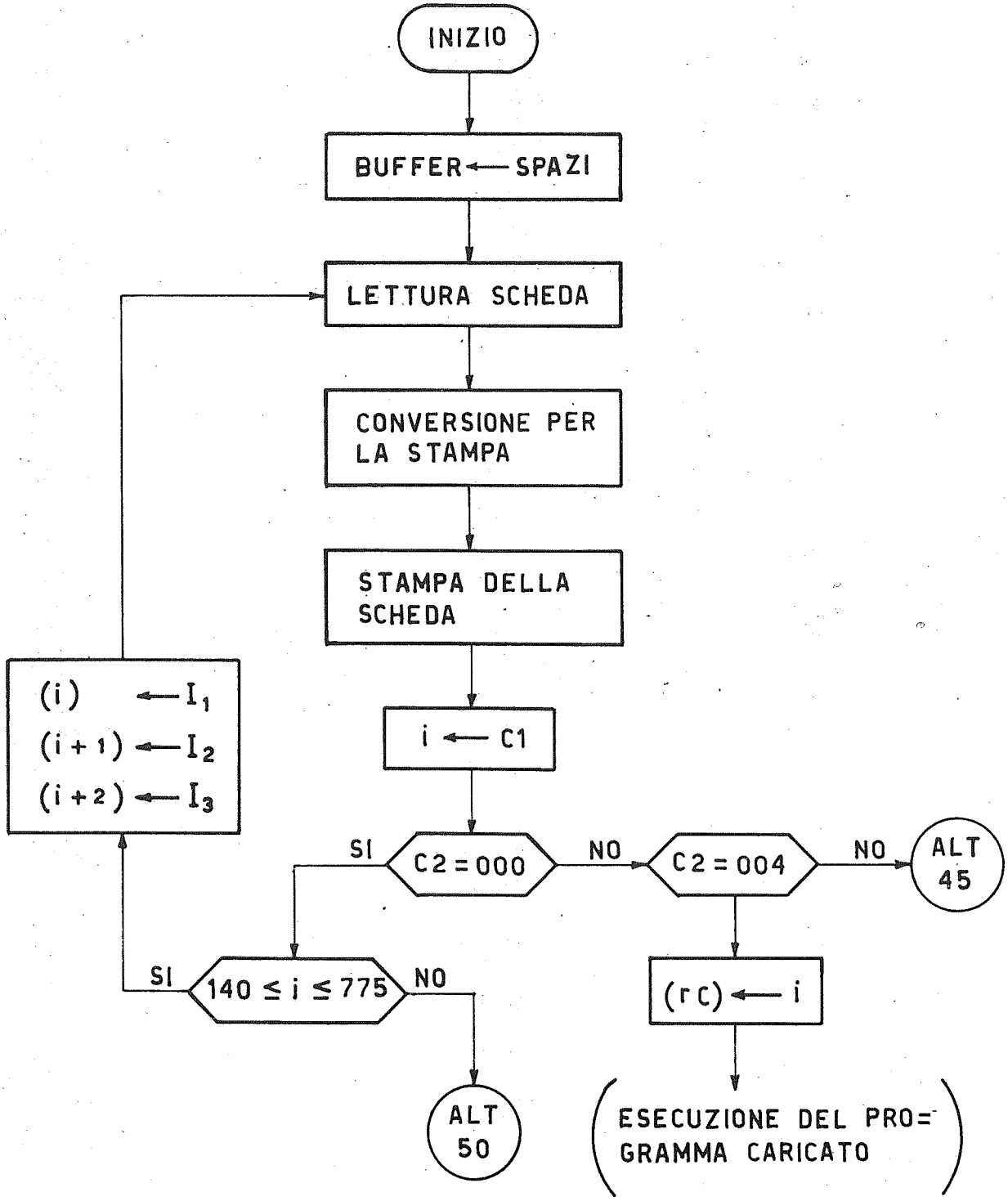


Fig. 10 - Schema a blocchi del caricatore binario.

2) Caricatore ottale

```
         ORIG    4
         TDX     0,1          }  BUFFER ← bbb
         TDX     16,7         }
LOOP4    MCA     BUFFER+23,7  }
         SODX    1,7          }  LC ← 0
         SLT     LOOP4        }

START    TDX     SETTE,6          ERRORE: ON

         CUS     *

         ENC     BUFFER       }  lettura della scheda
         CEN     *            }

         USC     BUFFER           stampa della scheda

         TRB     BUFFER
         CFB     BUFFER+24

         ADDX    1,7          {  LC ← LC+1
                                 LC = 0 ?

         SUG     C1B              C1 = bbb ?

         TDX     FINIX,6          ERRORE: OFF

         TDX     3,1          }
LOOP1    TCA     BUFFER       }
         CFA     SETTE        }
         SMAG    SETTE        }  C1 cifre ottali ?
         ABLD    3            }
         SODX    1,1          }
         SLT     LOOP1        }
```

```
        ABLD    9
        MEB     TR          } LC ← C1
        TRX     TR,7

C1B     TRB     BUFFER+1
        CFB     BUFFER+24   } C2 = bbb ?
        SUG     CAR

        CFB     FINE        } C2 = bFb ?
        SUG     0,6         } ERRORE: OFF ?

SETTE   ALT     7           ALT

CAR     CFXD    BUFFER+44,7 } LC ≥ 140(8) ?
        SMIN    SETTE

        TDX     2,2
LOOP3   TDX     3,1
LOOP2   TCA     BUFFER+1,2
        CFA     SETTE
        SMAG    SETTE
        ABLD    3           } I cifre ottali ?
        SODX    1,1
        SLT     LOOP2
        SODX    1,2
        SLT     LOOP3

        MEB     0,7         (LC) ← I

        SLT     START
```

```
FINE     ALF     bFb

FINIX    CUS     *
         SLT     0,7

BUFFER   BIS     40

TR       BIS     1
```

(rC) ⟵ C1

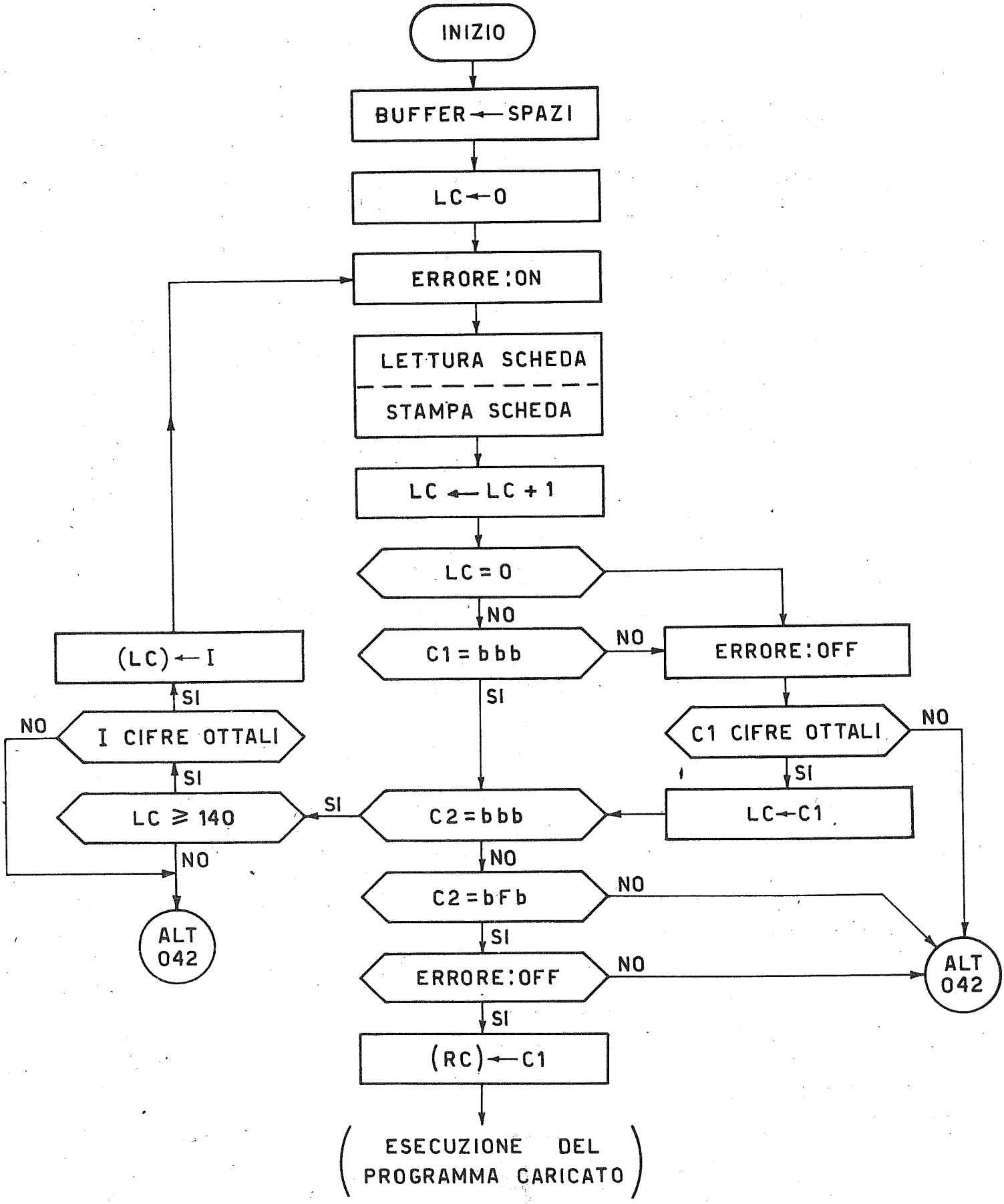


Fig. 11 - Schema a blocchi del caricatore ottale.

APPENDICE III

Struttura di un pacco programma CANE

Un pacco programma deve contenere (vedi l'esempio in fig. 12):

1) scheda asterisco
   (per un esempio di scheda asterisco, vedi la fig. 13);
2) scheda caricatore minimo;
3) caricatore binario (od ottale);
4) programma nel formato binario (o nel formato ottale);
5) eventuali sottoprogrammi di servizio;
6) eventuali dati.

L'ordine degli ingredienti 5) e 6) non è prefissato, ma è determinato dal programma. Altre composizioni più complesse sono naturalmente possibili, come del resto è già stato illustrato nel paragrafo 5.4.

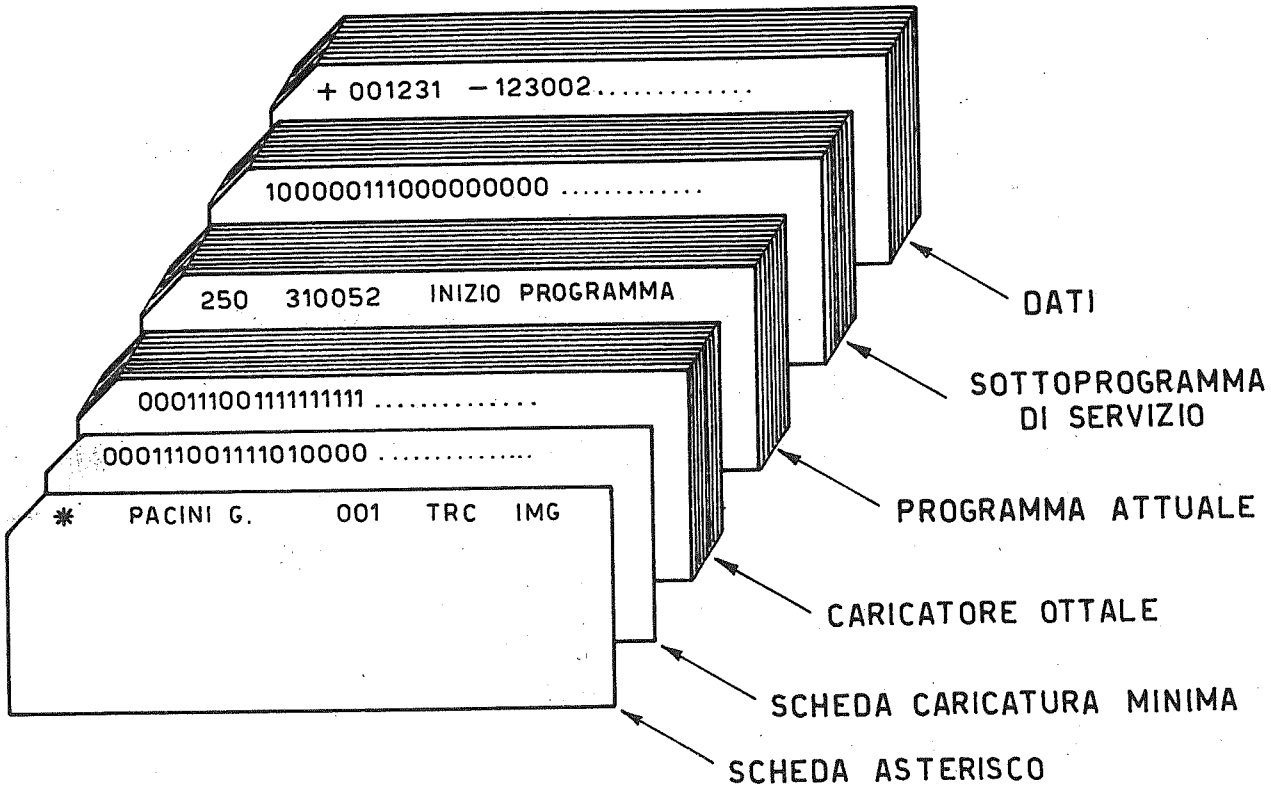


Fig. 12 - Struttura di un programma CANE.

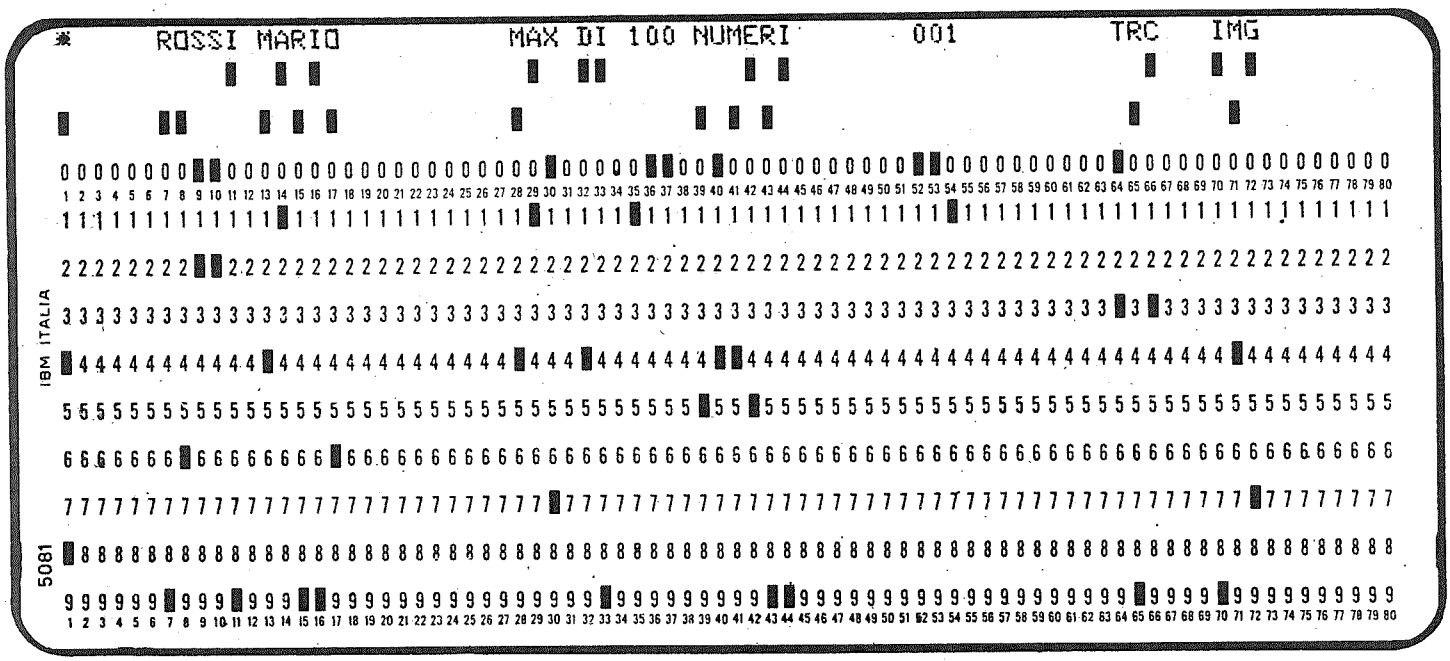


Fig. 13 - Esempio di scheda asterisco.

APPENDICE IV

Elenco delle istruzioni

00	ALT		
01	TRA	trasferimento	in rA
02	TRAN		in rA negativo
03	TDA		diretto in rA
04	TDAN		diretto in rA negativo
05	TRB		in rB
06	TRX		in rXj
07	TDX		diretto in rXj
10	MEA	memorizzazione	di rA
11	MEB		di rB
12	MEZ		di zero
13	MEX		di rKj
14	ADA	addizione ad rA	
15	SOA	sottrazione da rA	
16	ADDA	addizione diretta ad rA	
17	SODA	sottrazione diretta da rA	
20	ADB	addizione ad rB	
21	SOB	sottrazione da rB	
22	ADX	addizione ad rXj	
23	SOX	sottrazione da rXj	
24	ADDX	addizione diretta ad rXj	
25	SODX	sottrazione diretta da rXj	
26	MOL	moltiplicazione	

27	DIV	divisione
30	COP	complemento
31	AND	prodotto logico
32	OR	somma logica
33	ORX	somma modulo 2
34	CFA	confronto con rA
35	CFB	confronto con rB
36	CFX	confronto con rXj
37	CFXD	confronto diretto con rXj
40	SLT	salto incondizionato
41	SUB	salto a sottoprogramma
42	SSP	salto se supero
43	SMIN	salto se minore
44	SMAG	salto se maggiore
45	SUG	salto se uguale
46	SAN	salto se rA negativo
47	SAP	salto se rA positivo
50	SAZ	salto se rA zero
51	SBN	salto se rB negativo
52	SBP	salto se rB positivo
53	SBZ	salto se rB zero
54	AVD	spostamento aritmetico di rA verso destra
55	AVS	spostamento aritmetico di rA verso sinistra
56	ALD	spostamento logico di rA verso destra
57	ALS	spostamento logico di rA verso sinistra
60	BVD	spostamento aritmetico di rB verso destra
61	BVS	spostamento aritmetico di rB verso sinistra
62	ABLD	spostamento logico di rA ed rB verso destra
63	ABLS	spostamento logico di rA ed rB verso sinistra

64	TCA	trasferimento carattere
65	MCA	memorizzazione carattere
66	CAR	conversione in caratteri
67	NUM	conversione in numero binario
70	ESG	esegui
71	GULP	
72	ENB	entrata binaria
73	ENA	entrata alfanumerica
74	USC	uscita
75	CEN	controllo entrata
76	CUS	controllo uscita
77	NOP	nessuna operazione