Consiglio Nazionale delle Ricerche
ISTITUTO DI ELABORAZIONE DELL'INFORMAZIONE

ARCHITETTURA E SIMULAZIONE DI UN CALCOLATORE DIDATTICO

G. Pacini

Nota Tecnica C70/3
(Marzo 1970)

UNIVERSITA' DEGLI STUDI DI PISA
Facoltà di Scienze M.F.N.
- Corso di Laurea in FISICA -

TESI DI LAUREA:

ARCHITETTURA E SIMULAZIONE DI UN CALCOLATORE DIDATTICO.

Relatore:
Chiar.mo Prof.Antonio GRASSELLI

Candidato:
Giuliano PACINI

- Anno Accademico 1968-69 -

I N D I C E

-----====oooOooo=========

1. INTRODUZIONE

Il Calcolatore CANE (Calcolatore Automatico Numerico Educativo), la cui simulazione è argomento di questa nota, è un calcolatore ideale che è stato definito a scopi puramente didattici. Il CANE verrà illustrato durante un corso di introduzione ai calcolatori elettronici (corso di "Teoria ed applicazioni delle macchine calcolatrici" per la laurea in scienze dell'informazione), e verrà programmato dagli studenti nel parallelo corso di esercitazioni.

Il CANE è stato progettato con il preciso intendimento di mettere a disposizione degli studenti una macchina particolarmente adatta ad una prima introduzione alla programmazione in linguaggio macchina, e alla architettura e struttura logica delle macchine calcolatrici. Si è pensato che la maniera migliore di raggiungere questo scopo fosse la definizione di un calcolatore ideale che godesse delle seguenti due caratteristiche:

a) la macchina fosse abbastanza simile ad un calcolatore reale in modo da evitare una presentazione distorta della programmazione;

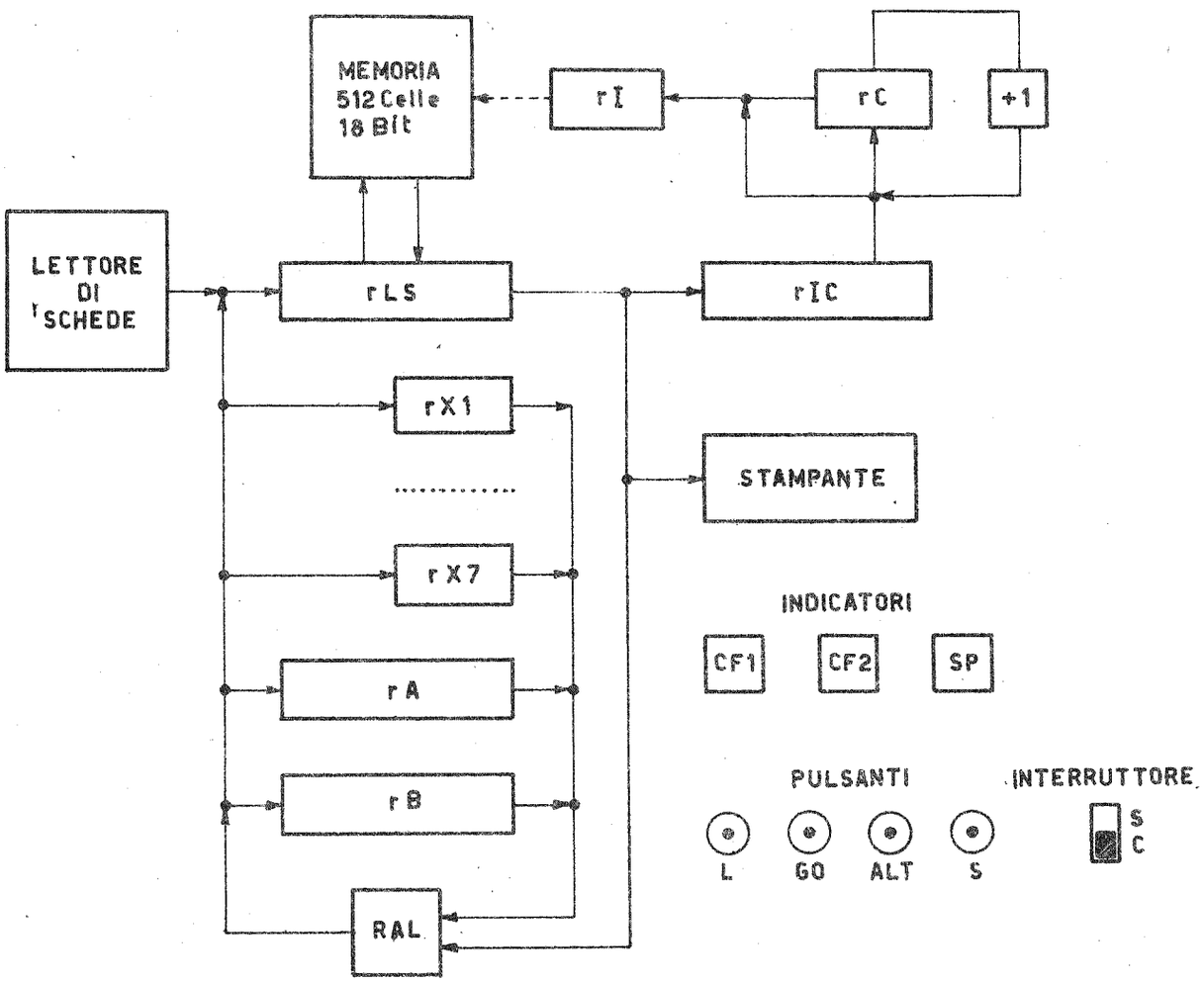
b) fossero eliminate tutte quelle complicazioni, molte volte di

carattere tecnico costruttivo, caratteristiche delle macchine reali, che potevano risultare inessenziali e forse addirittura dannose dal punto di vista formativo.

Si è cercato di restare coerenti con il punto a) dotando il CANE di un insieme di istruzioni paragonabile come varietà e completezza a quello di una macchina reale, e con il punto b) facendo in modo che l'insieme delle istruzioni fosse caratterizzato da una semplicità ed eleganza che non hanno riscontro di solito nei calcolatori reali, ma soprattutto strutturando le apparecchiature di ingresso e uscita in modo da permetterne un uso veramente semplice e lineare; nei calcolatori reali, invece, le difficoltà relative alle operazioni di ingresso-uscita in linguaggio macchina, sono così notevoli da impegnare anche un programmatore di provata esperienza. Comunque anche a questo proposito si è voluto che il CANE non fosse troppo dissimile da una macchina reale, e se ne è progettata l'architettura sfrondando unicamente le caratteristiche inessenziali. In particolare, la unità centrale continua a lavorare dopo aver lanciato una operazione di entrata-uscita (come nella realtà avviene per quasi tutte le macchine reali). Si è inoltre cercato di dotare il CANE di un software minimo (soprattutto per quanto riguarda l'entrata-u

scita ed i programmi di utilità), in modo che lo studente cominciasse a maturare fin dall'inizio il concetto di calcolatore inteso come insieme costituito dalla macchina fisica e dal software di cui essa è dotata.

Il nucleo di questa tesi è costituito dalla illustrazione del simulatore del CANE, realizzato dal candidato sul calcolatore IBM 7090 del CNUCE e del software di servizio, pure realizzato dal candidato. L'architettura del CANE, alla cui progettazione ha partecipato il candidato, è esposta nel Capitolo 2.



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.

2. Architettura del calcolatore CANE

Definiremo il calcolatore mediante la lista delle istruzioni, preceduta dalla lista degli elementi costituenti la macchina che sono direttamente richiamati nella definizione di una o più istruzioni, e dalla descrizione della codifica adottata per la rappresentazione interna dei vari tipi di informazione. Una tale descrizione della macchina è da considerarsi completa dal nostro punto di vista, perché illustra in sostanza tutte le caratteristiche architettoniche e strutturali che il nostro simulatore riproduce.

2.1 Schema architettonico.

Lo schema architettonico del calcolatore CANE è mostrato nella fig.1. Gli elementi che ci interessano per l'esposizione sono:

1) la memoria centrale di 512 parole
2) i due registri accumulatori rA e rB
3) i 7 registri indice rx1, rx2 ... rX7
4) l'indicatore di supero (overflow) SP
5) i due indicatori di confronto CF1 e CF2
6) il registro contatore istruzioni rC

7) l'unità di entrata (lettore di schede con due modalità di lettura: binaria e alfanumerica)

8) l'unità di uscita (stampante).

Le celle di memoria sono lunghe 18 bit, come pure i registri rA e rB, registri indice e registro contatore istruzioni sono invece lunghi 9 bit. Tutte le istruzioni sono codificate da una parola di 18 bit; il profilo della istruzione è mostrato nella fig. 2(a); la fig. 2(b) mostra un esempio di istruzione.

La consolle (vedi fig. 1) possiede 4 pulsanti ed un interruttore, e precisamente:

a) un pulsante L di lettura e avvio

b) un pulsante GO di avvio

c) un pulsante ALT

d) un pulsante S (singolo)

e) un interruttore a due posizioni (singolo,continuo) SC.-

Nel funzionamento normale, l'interruttore SC si trova nella posizione C (continuo).

Premendo il pulsante GO (avvio) la macchina inizia la esecuzione del programma prendendo come prima istruzione quella contenuta nella cella indicata dal contenuto del registro contatore

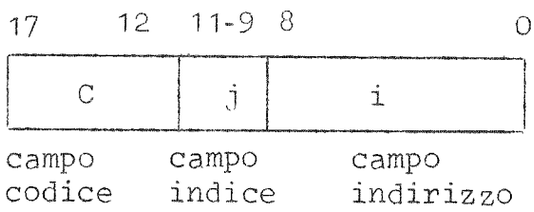
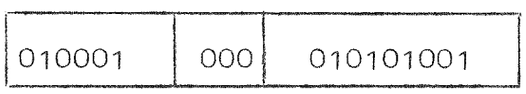


fig. 2 a)



codice della istruzione sottrazione ad rA

indirizzo

$010101001_{(2)} = 251_{(8)}$

fig. 2 b)

istruzioni rC. Premendo ALT, la macchina si ferma dopo aver completato l'esecuzione della istruzione corrente.
Premendo il pulsante L (lettura e avvio) a macchina ferma, viene letta una scheda secondo la modalità binaria, nelle celle 000, 001, 002, 003, e dopo il completamento della operazione di lettura (vedi 2.3) la macchina passa ad eseguire la istruzione contenuta nella cella 000.
Come vedremo nel capitolo 3 il pulsante L viene usato per far entrare in memoria le prime quattro istruzioni di un programma di caricamento.
Se l'interruttore SC è nella posizione S (singolo) il calcolatore esegue una istruzione tutte le volte che viene premuto il pulsante S (singolo).

2.2 Rappresentazione interna di numeri e caratteri.

Una parola può rappresentare un numero intero con segno, oppure una sequenza di 3 caratteri alfammerici. Il codice impiegato nella rappresentazione dei caratteri alfammerici è il normale codice B.C.D. a 6 bit/carattere in uso sul sistema I.B.M. 7090. I numeri sono rappresentati nel sistema di numerazione bi

nario, ed è stata impiegata la rappresentazione in complemento a 2 per i numeri negativi. Perciò in una parola del CANE si può rappresentare un qualsiasi intero i nell'intervallo:

$$-(2^{17}-1) \quad i \quad 2^{17}-1 \ .$$

2.3 Le istruzioni

Il funzionamento delle istruzioni è descritto con la seguente simbologia. Il contenuto di un registro o di una cella di memoria sarà indicato mettendo tra parentesi il nome del registro o l'indirizzo della cella: così, (rA) indica il contenuto del registro rA, (i) il contenuto della cella di indirizzo i, (i+(rXj)) il contenuto della cella il cui indirizzo si ottiene aggiungendo ad i il contenuto del registro indice j. Inoltre con (X)y (dove X è il nome di un registro o un indirizzo, y=c oppure j oppure i) si indicherà il contenuto del campo codice, indice od indirizzo del registro o della cella di memoria, infine con (rA,rB) il contenuto dei registri rA ed rB considerati come un unico registro.

Ricordando che una parola del CANE può contenere la codifica di tre caratteri alfanumerici indicheremo con $(i)_I$, $(i)_{II}$ e $(i)_{III}$ rispettivamente, partendo da sinistra, il primo il secon

do e il terzo carattere (gruppo di 6 bits) contenuti nella cella di indirizzo i. Analogo significato avranno $(rA)_I$, $(rA)_{II}$ e $(rA)_{III}$.

La lista delle istruzioni del CANE è la seguente :

codice ottale	codice mnemonico	
00	ALT	(rC)←i+(rXj) la macchina si ferma
01	TRA	(rA)←(i+(rXj))
02	TRAN	(rA)←-(i+(rXj))
03	TDA	(rA) ← i+(rXj)
04	TDAN	(rA) ← -i-(rXj)
05	TRB	(rB) ← (i+(rXj)
06	TRX	(rXj) ← $(i)_i$
07	TDX	(rXj) ← i
10	MEA	(i+(rXj)) ← (rA)
11	MEB	(i+(rXj)) ← (rB)
12	MEZ	(i+(rXj)) ← 0
13	MEX	$(i)_i$ ← (rXj)

14	ADA	(rA) ⟵ (rA)+(i+(rXj))
15	SOA	(rA) ⟵ (rA)-(i+(rXj))
16	ADDA	(rA) ⟵ (rA)+i+(rXj)
17	SODA	(rA) ⟵ (rA)-i-(rXj)
20	ADB	(rB) ⟵ (rB)+(i+(rXj))
21	SOB	(rB) ⟵ (rB)-(i+(rXj))
22	ADX	(rXj) ⟵ (rXj)+$(i)_i$
23	SOX	(rXj) ⟵ (rXj)-$(i)_i$
24	ADDX	(rXj) ⟵ (rXj)+i
25	SODX	(rXj) ⟵ (rXj)-i

skip di una istruzione se (rXj) = 0

26	MUL	(rA,rB) ⟵ (rB)X (i+(rXj))
27	DIV	(rB) ⟵ (rA,rB): (i+(rXj)),(rA) ⟵ resto
30	COP	(i+(rXj)) ⟵ $\overline{(i+(rXj))}$ complementazione bit a bit
31	AND	(rA) ⟵ (rA) $\cap$ (i+(rXj)) "and" bit a bit
32	OR	(rA) ⟵ (rA) $\cup$ (i+(rXj)) "or" bit a bit
33	ORX	(rA) ⟵ (rA) $\oplus$ (i+(rXj)) "or esclusivo" bit a bit

34	CFA	(CF1)⟵1 se	$(rA) \geq (i+(rXj))$	,(CF2)⟵1 $(rA) \leq (i+(rXj))$
35	CFB		$(rB) \geq (i+(rXj))$	$(rB) \leq (i+(rXj))$
36	CFX		$(rXj) \geq (i)_i$	$(rXj) \leq (i)_i$
37	CFXD		$(rXj) \geq i$	$(rXj) \leq i$

40	SLT	(rC) ← i+(rXj)	
41	SUB	(rXi) ← (rC) ,	(rC) ← i
42	SSP	(SP)=1	
43	SMIN	(CF1)=0, (CF2)=1	
44	SMAG	(CF1)=1 (CF2)=0	
45	SUG	(CF1)= (CF2) =1	
46	SAN	se (rA) < 0	(r C) ← i+(rXj)
47	SAP	(rA) > 0	
50	SAZ	(rA) = 0	
51	SBN	(rB) < 0	
52	SBP	(rB) > 0	
53	SBZ	(rB) = 0	

54	AVD	scorrimento aritmetico (rA) verso destra di i+(rXj) posti
55	AVS	" " " " sinistra "
56	ALD	" logico " " destra "
57	ALS	" " " " sinistra "
60	BVD	" aritmetico (rB) verso destra "
61	BVS	" " " " sinistra "
62	ABLD	" logico (rA, rB) " destra "
63	ABLS	" " " " " sinistra "

64 TCA $(rA) \longrightarrow 00 \quad 0$

$$(rX1)_n^{(1)} = \begin{cases} 01: & (rA)_{III} \longleftarrow (i+(rXj))_I \\ 10: & (rA)_{III} \longleftarrow (i+(rXj))_{II} \\ 11: & (rA)_{III} \longleftarrow (i+(rXj))_{III} \\ 00: & \text{nessuna operazione} \end{cases}$$

65 MCA

$$(rX1)_n = \begin{cases} 01: & (i+(rXj))_I \longleftarrow (rA)_{III} \\ 10: & (i+(rXj))_{II} \longleftarrow (rA)_{III} \\ 11: & (i+(rXj)_{III} \longleftarrow (rA)_{III} \\ 00 & (i+(rXj)) \longleftarrow \text{tre spazi.} \end{cases}$$

66 CAR Il valore assoluto del numero contenuto in rA, convertito in decimale viene scritto sotto forma di 6 cifre decimali in codice caratteri in rA (3 cifre più significative) ed rB.

67 NUM Le 6 cifre decimali in codice caratteri contenute in rA o rB vengono trattate come numero positivo, che convertito in binario viene scritto in rA.

70 ESG esegue l'istruzione nella cella i+(rXj)

---

1) Con $(rX1)_n$ si indicano i due bit meno significativi del registro indice 1.

70 ESG esegue l'istruzione nella cella i+ (rXj)

71 codice libero

72 ENB legge 4 parole binarie nelle celle i+(rXj),....., i+(rXj) + 3

73 ENA legge 24x3 caratteri nelle celle i+(rXj), ..., i+(rXj) + 23

74 USC scrive i 40x3 caratteri contenuti nelle celle i+(rXj), ..., i+(rXj) + 39

75 CEN se lettore occupato (rC) ← i+(rXj)

76 CUS se stampante occupata,(rC) ← i+(rXj)

77 NOP nessuna operazione.

2.4 Particolarità sulle istruzioni.

L'indicatore di supero SP può passare in posizione "On", in conseguenza di un supero di capacità di un registro, dopo la esecuzione di una qualunque istruzione aritmetica sui registri accumulatori (eccettuata la istruzione MOL), dopo la esecuzione di scorrimenti aritmetici verso sinistra e dopo la esecuzione della istruzione NUM. Dopo la istruzione SSP l'indicatore SP è sempre in posizione "off".

I numeri contenuti nei registri indice sono trattati sempre come numeri positivi. I registri indice funzionano modulo 1000_8, cioè per esempio, se il registro indice 1 contiene

$$760_8 \quad e$$

viene eseguita la istruzione

$$\text{ADDX} \quad 27_8,1 \ ;$$

dopo la esecuzione è

$$(rX1) = 007_8 \ .$$

Il risultato della operazione di moltiplicazione viene lasciato nei registri rA ed rB; la parte meno significativa (cioè i primi 17 bit a partire da destra) del risultato viene lasciata con il suo segno nel registro rB, la parte più significativa, anch'essa con il suo segno, in rA. In altri termini il risultato della moltiplicazione è dato dalla seguente espressione:

$$(rA) \times 2^{17} + (rB) \ .$$

Si noti che se una delle due parti è nulla i segni dei due registri dopo la moltiplicazione possono essere anche diversi [1].

---

1) Si tenga presente che nella rappresentazione in complemento 2 si ha una sola rappresentazione dello zero.

Analogamente la operazione di divisione assume come dividendo il numero:

$$(rA) \times 2^{17} + (rB).$$

La divisione viene effettuata mediante l'algoritmo non restaurativo descritto in: Chu Yaohan, Digital Computer Design Fundamentals pag. 39.

Tale algoritmo è stato scelto per la sua notevole semplicità.

Dalla lista delle istruzioni risulta evidente che l'ingresso uscita del CANE è, come preannunciato nella introduzione, di una estrema semplicità.

L'operazione di lettura può essere eseguita secondo due modalità: nella modalità binaria l'informazione contenuta in una scheda (consistente in una sequenza di 72 caratteri zero od uno) viene trasferita in 4 parole di memoria facendo corrispondere ad ogni colonna della scheda un bit in una delle quattro parole ; nella modalità alfanumerica l'informazione contenuta in una scheda viene trasferita in 24 parole del CANE facendo corrispondere ad ogni carattere sulla scheda un gruppo di 6 bit secondo il codice I.B.M. menzionato in 2.2.

Al contrario l'operazione di scrittura può essere eseguita con la sola modalità alfanumerica. All'atto della esecuzione di una

istruzione di ingresso-uscita vengono messe in moto le unità periferiche (stampante o lettore), dando così inizio all'ingresso o all'uscita delle informazioni; il calcolatore continua poi la elaborazione passando alle istruzioni successive.
Se una istruzione di ingresso o uscita viene incontrata prima che la relativa unità periferica abbia completato un'operazione precedentemente avviata, l'elaborazione viene sospesa fino a che l'unità abbia concluso l'operazione in corso.
Infine si tenga presente che la macchina si ferma se si tenta di leggere con modalità binaria una scheda contenente caratteri diversi da zero, uno, blank (1), e se viene incontrata una istruzione di entrata quando il contenitore del lettore di scheda è vuoto.

---

(1) Il carattere "blank" viene interpretato come zero.

3. SOFTWARE DI SERVIZIO

Il software del CANE è costituito da :

1) Caricatore di programmi codificati in binario.
2) Caricatore di programmi codificati in ottale.
3) Sottoprogramma di lettura e scrittura di numeri decimali con segno
4) Sottoprogramma di "dump parziale"
5) Sottoprogramma di "stampa registri".

3.1 Osservazioni generali

Ciascuno dei 5 programmi che compongono il Software del CANE è costituito da 48 istruzioni.
I programmi sono scritti in binario [1] ciascuno su 12 schede [2] ed essendo codificati in assoluto debbono essere caricati in memoria sempre nella stessa ben prestabilita posizione, precisamente, ognuno di essi deve essere posto a partire dalla

---

1) Si noti bene: non nella codifica binaria accettato dal caricatore binario.

2) Su ogni scheda binaria si possono rappresentare (scrivendo di seguito da colonna 1 a 72) 4 istruzioni.

cella di indirizzo 4_8 fino alla cella di indirizzo 63_8 (1). Questo fatto esclude che due dei cinque programmi possano essere adoperati contemporaneamente, ma permette di leggere in memoria uno qualunque di essi mediante il gruppo di 4 istruzioni :

000	TDX	720_8, 1
001	ENB	64_8, 1
002	ADDX	4, 1
003	SLT	1

Come si vede facilmente, in questa maniera vengono lette 12 schede binarie poi automaticamente il controllo passa alla cella successiva a quella contenente la istruzione SLT (2). Le quattro istruzioni possono essere caricate dalla cella 000 alla cella 003 per mezzo del dispositivo di lettura e avvio di cui il CANE è dotato, che è capace di leggere una scheda binaria siste

---

1) Ciascuno dei cinque programmi occupa in totale le celle da 004_8 a 137_8, in quanto 44 parole sono necessarie per i buffers di lettura e scrittura o per memorizzazioni temporanee.

2) Si ricordi che se dopo la esecuzione della istruzione ADDX il registro indice contiene zero il calcolatore salta la istruzione immediatamente successiva alla ADDX.

mando le quattro parole in essa contenute nelle celle 000-003 passando poi ad eseguire la istruzione contenuta nella cella 000.

Il sistema, nel caso di uno dei caricatori, funziona nella maniera seguente: ponendo inizialmente uno dei due caricatori preceduto dalla scheda contenente le quattro istruzioni di lettura, immediatamente di seguito il programma da eseguire, ed azionando infine il pulsante di lettura e avvio il caricatore viene letto in memoria ed entra automaticamente in funzione in quanto la voce 004 contiene, dopo la fine del ciclo di lettura, la prima istruzione del caricatore.

In tal modo il programma, codificato in uno dei due formati descritti nel seguito di questo capitolo, viene caricato e finalmente eseguito. Il funzionamento dei tre sottoprogrammi di servizio è illustrato nel paragrafo 3.4.

3.2 Caricatore binario.

La codifica binaria è descritta schematicamente nella figura 3 . Su ogni scheda trovano posto tre istruzioni, scritte di seguito da colonna 18 a 72, rispettivamente nei campi I_1, I_2; I_3.

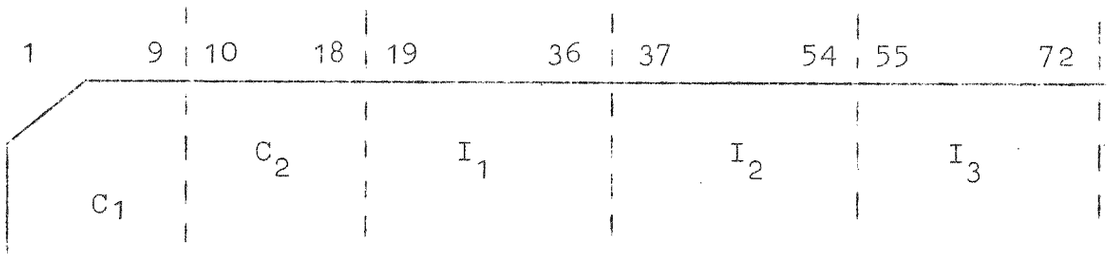


fig. 3

Le colonne da 10 a 18 debbono avere una delle due configurazioni seguenti:

1) 000000000

2) 000000100 .

Nel primo caso il caricatore pone in memoria le tre istruzioni a partire dalla cella il cui indirizzo è specificato nelle colonne 1 - 9, nel 2° caso il caricatore salta alla cella il cui indirizzo è indicato nelle colonne 1 - 9. In altre parole la specifica 000000100 provoca il termine del caricamento e l'inizio dell'esecuzione del programma.

Il caricatore binario è in grado di rivelare 2 errori:

1) C2 = 000000000

C1 $< 140_8$ oppure C1 $> 775_8$ (*)

2) C2 $\neq$ 000000000 C2 $\neq$ 000000100

(*) In caso contrario il caricatore caricherebbe su se stesso.

quando uno dei due errori viene riconosciuto il caricatore si ferma.

La schèda di fine caricamento (C2 = 000000100) può contenere nelle colonne 1-9 un numero qualsiasi.

Ogni scheda letta durante il caricamento viene stampata, si ha così una lista del programma fino alla scheda di fine caricamento (o fino alla prima scheda errata) inclusa.

3.3. Caricatore ottale.

Il formato della codifica ottale è illustrato nella fig. 4 .

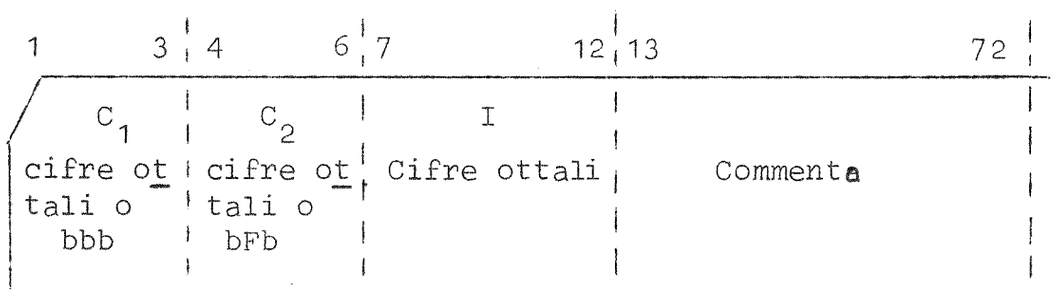


fig. 4

In una scheda trova posto una sola istruzione, che deve essere scritta da colonna 7 a 12, le colonne da 13 a 72 sono considerate dal caricatore come un commento, nelle colonne da 1 a 3 trova posto (sempre in ottale) l'indirizzo della voce in cui il caricatore deve sistemare le sei cifre scritte da colonna 7 a 12, (se questo campo è bianco il caricatore procede in sequenza).

Le colonne 4-6 devono essere bianche (nel qual caso avviene il normale caricamento) oppure contenere la specifica bFb ,in questo caso il caricamento cessa e il controllo passa alla locazione il cui indirizzo è indicato in ottale nelle colonne 1-3.

Esempio

300 bbb 123456 ⟵ viene caricato nella voce 300

bbb bbb 012345 ⟵ viene caricato nella voce 301

.....

300 bFb AB1234 fine caricamento e salto alla cella 300

Il caricatore ottale è in grado di rivelare 7 errori :

1) C2 = bbb C1 < 140

2) C1 ≠ bbb C1 ≠ cifre ottali

3) C2 = bbb I ≠ cifre ottali

4) Prima scheda con C2 = bbb C1 = bbb

5) C2 = bFb C1 = bbb

6) C2 ≠ bFb C2 ≠ bbb

7) C1 = bbb C2 = bbb e

precedente caricamento nella cella 777.

La scheda di fine caricamento (C2 = bFb) può contenere nelle colonne 1-3 un numero qualsiasi.

Qualora sia riconosciuto uno degli errori il caricatore si fer-

ma.

Ogni scheda letta durante il caricamento viene ristampata, si ha così una lista del programma e relativi commenti fino alla scheda di fine caricamento (o fino alla prima scheda errata) inclusa).

3.4 Sottoprogrammi di servizio.

Ciascuno dei tre sottoprogrammi può essere letto in memoria dal gruppo di 4 istruzioni già menzionato in 3.1 :

```
000 |  TDX    1  720
001 |  ENB    1  064
002 |  ADDX   1  004
003 |  SLT    0  001.
```

Il programma che vuole servirsi di uno dei tre sottoprogrammi può farlo entrare in memoria con le due istruzioni :

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

L'istruzione SUB salta alla cella 000; quando si esce dal ciclo di lettura viene eseguita la prima istruzione del sottoprogramma, che è :

SLT 7000

e si salta così alla istruzione CEN * che viene eseguita fino al completamento della lettura della ultima scheda del sotto-programma. I tre sottoprogrammi lasciano invariati i registri accumulatori e i registri indice (eccettuato rx7), modificano lo stato degli indicatori di confronto.

a) Sottoprogramma di Dump parziale.

Il sottoprogramma viene chiamato, nella ipotesi che sia già in memoria, mediante la istruzione

SUB 7005.

Ad ogni chiamata vengono stampati tre gruppi di 8 celle, a partire dalla cella il cui indirizzo è indicato negli ultimi 9 bits della parola immediatamente successiva alla istruzione di chiamata.

b) Sottoprogramma di Stampa registri.

Stampa ad ogni chiamata il contenuto di tutti i registri. La chiamata si effettua con la istruzione

SUB 7005.

c) Sottoprogramma di lettura e scrittura numeri decimali.

Il sottoprogramma ha due punti di ingresso L (lettura) ed S

(scrittura).

Chiamando L, esso legge un numero di sei cifre decimali e segno in uno dei campi indicati in fig. 5 ; una successiva chiamata di L provoca la lettura del numero nel campo successivo della stessa scheda,

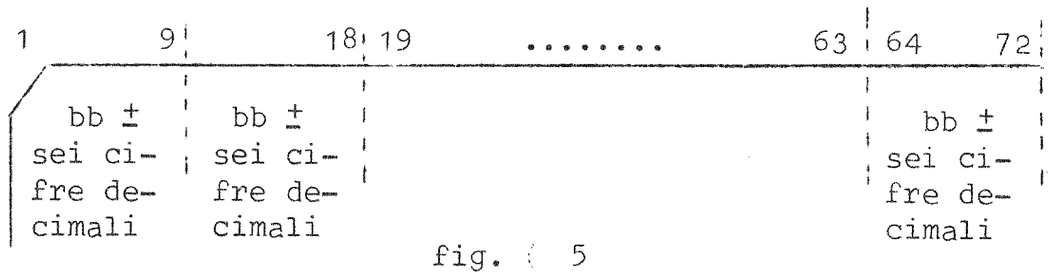


fig. 5

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 casi:

a) dopo aver chiamato il punto di ingresso L, viene chiamato il punto S (scrittura)

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

Il numero letto convertito in binario viene lasciato in rA.

Chiamando il punto di ingresso S viene stampato il contenuto di rA nel formato

± sei cifre decimali .

I punti L ed S vengono chiamati rispettivamente :

```
punto L      SUB   7034
punto S      SUB   7005 .
```

4. CONSIDERAZIONI GENERALI SULLA SIMULAZIONE IN RIFERIMENTO AL SIMULATORE DEL CANE.

In generale con il termine simulatore si indica un programma che posto nella memoria di una macchina A la mette in grado di eseguire programmi codificati per un'altra macchina B.
La necessità di usare un programma simulatore sorge ogni volta che si voglia far correre su una macchina A un programma scritto per una macchina B.
In pratica si ricorre all'uso di un programma simulatore principalmente nei due casi seguenti:

1) costruzione e progetto di un nuovo calcolatore
2) sostituzione di un vecchio calcolatore con una nuova macchina.

Nel primo caso il simulatore consente di iniziare lo studio del software prima che la costruzione e la messa a punto del calcolatore siano ultimate.
Nel secondo il simulatore, che è il più delle volte fornito direttamente dalla ditta costruttrice, consente di far correre sulla nuova macchina dei programmi che erano stati scritti per la vecchia.

La stesura di un programma simulatore può essere impostata secondo due metodi distinti: un metodo di tipo assemblativo e un metodo di tipo interpretativo.

I simulatori di tipo assemblativo funzionano in due passi successivi. Nel primo passo il simulatore analizza tutto il programma codificato nel linguaggio della macchina B e lo traduce, mediante l'uso di macroistruzioni, nel linguaggio della macchina A, nel secondo passo il programma tradotto viene eseguito.

I simulatori di tipo interpretativo considerano una istruzione alla volta, la interpretano e la eseguono, e passano quindi alla istruzione successiva.

In un simulatore interpretativo, tutti i registri, indicatori, e celle di memoria di B sono simulati da altrettante celle di memoria di A (che indicheremo con il termine di memoria e registri virtuali), il programma da eseguire (già codificato nel linguaggio macchina di B) deve essere posto senza alcuna modifica nella memoria virtuale. La simulazione procede in modo che dopo la interpretazione ed esecuzione di una qualsiasi istruzione, i registri e la memoria virtuale contengono esattamente quello che conterrebbero se la istruzione fosse stata eseguita dalla macchina B.

Caratteristiche del metodo assemblativo sono:

1a) il programma simulatore non deve essere necessariamente in memoria durante il secondo passo della simulazione.

2a) il programma da eseguire deve essere codificato in un linguaggio di tipo simbolico .

3a) anche nella ipotesi 2a è molto difficile simulare la esecuzione di programmi che si automodificano.

L'affermazione 2a potrebbe essere argomento di discussione molto più approfondita, possiamo però senz'altro affermare che un programma simulatore di tipo assemblativo in grado di partire da un linguaggio tipo macchina, quantomeno, sarebbe molto più complesso di un programma simulatore elaborato per tradurre ed eseguire programmi codificati in un linguaggio di tipo simbolico.

Caratteristiche del metodo interpretativo sonò:

1b) il programma simulatore deve restare sempre in memoria;

2b) il programma simulato può essere codificato nel linguaggio macchina del calcolatore B;

3b) non vi sono restrizioni sulla natura del programma simulato.

Da questo rapido confronto si vede che i due metodi sono, sia praticamente che concettualmente, molto diversi. Concludendo si può dire che un simulatore di tipo assemblativo essenzialmente mette in grado la macchina A di eseguire programmi codificati nel linguaggio simbolico di B, mentre un simulatore di tipo interpretativo (come descritto sopra) posto nella memoria di A realizza "fisicamente" il calcolatore B. Infatti, per chiarire questa ultima affermazione, tutti i registri e la memoria di B sono fisicamente simulati da celle di memoria di A ed hanno passo a passo la configurazione che avrebbero sulla macchina B.
Si noti, a conclusione di queste brevi considerazioni generali, che affinché la macchina B sia simulabile sulla macchina A è necessario che la A abbia una "potenzialità" più elevata della B, dove con il termine "potenzialità" intendiamo qui indicare globalmente l'insieme delle caratteristiche della macchina, quali: numero di celle di memoria, numero di bits in ogni cella, numero di registri, apparecchiature di in/out , numero delle istruzioni base di macchina ecc.
Nel caso che la macchina A abbia "potenzialità" minore della B la simulazione, anche se non sempre impossibile in linea di principio, porterebbe certamente a difficoltà pratiche molto notevoli se non addirittura proibitive.

Passando ora a parlare della simulazione del CANE, senza soffermarsi ulteriormente sugli scopi del nostro lavoro che sono già stati chiariti nella introduzione, vogliamo solo puntualizzare che il nostro è senza dubbio un caso un po' particolare, infatti il CANE è un calcolatore ideale la cui realizzazione pratica è sconsigliabile e la cui struttura logica invece, studiata a scopi didattici è assai interessante. Questo fatto, se rende da una parte più semplice il lavoro di simulazione in quanto un calcolatore ideale è caratterizzato da una semplicità ed eleganza strutturale che una macchina reale di solito non possiede, ci pone d'altra parte dei problemi che sono essenziali dal nostro punto di vista e che invece possono essere ignorati nella maggior parte dei casi.

Questi problemi sorgono essenzialmente perché è nostro intendimento che il simulatore, più che permetterci semplicemente di passare sul 7090 programmi scritti in linguaggio CANE, riproduca tutte le caratteristiche della macchina, e sia strutturato in modo che i programmi possano essere passati e i risultati interpretati senza particolari accorgimenti dipendenti dal simulatore. In altre parole il simulatore deve essere tale che sia possibile programmare ed agire sotto tutti gli a-

spetti come se il CANE esistesse realmente.

Da queste ultime osservazioni e dal rapido confronto effettuato in questo paragrafo tra i due metodi di simulazione risulta evidente che l'unico dei due che si presta al nostro caso è il metodo interpretativo. Difatti il simulatore del CANE realizzato sul 7090 è un simulatore di tipo interpretativo.

5. IL PROGRAMMA SIMULATORE

Il programma è composto da due parti ben distinte anche dal punto di vista concettuale.

1) Simulatore del CANE

2) Programma supervisore.

Il programma supervisore non è essenziale per la simulazione ed ha il compito di rendere possibile la esecuzione automatica, da parte del simulatore di un numero qualsivoglia di programmi l'uno di seguito all'altro senza che tra di essi ci possa essere interferenza alcuna. Il simulatore a sua volta può essere diviso in tre parti, il simulatore propriamente detto e due programmi di servizio, uno di traccia e uno di dump ; il primo stampa tutti i registri e gli indicatori virtuali ogni volta che uno di essi viene modificato mentre il simulatore esegue un programma, il secondo stampa una immagine completa della memoria virtuale dopo che il simulatore ha terminato la esecuzione del programma.

Il simulatore può essere predisposto per funzionare normalmente oppure in traccia, per stampare oppure no la immagine finale della memoria.

Programma completo
- Supervisore
- Simulatore
  - Simulatore prop. detto
  - Programma di traccia
  - Programma di immagine

5.1 Funzionamento del simulatore

Il simulatore è di tipo interpretativo; tutte le celle di memoria, i registri e gli indicatori del CANE, sono simulati da altrettante celle di memoria del 7090, che indicheremo con i termini di memoria, registri e indicatori virtuali. Poiché le celle del 7090 sono costituite da 36 bits, la metà della parola è sufficiente per simulare una cella del CANE'. Il simulatore funziona in modo che dopo la esecuzione di una qualsiasi istruzione di macchina del CANE i registri gli indicatori e la memoria virtuale contengono quello che conterrebbero se la istruzione fosse stata eseguita dalla macchina simulata.

Come si è detto nel paragrafo precedente il simulatore è in grado di fornire a richiesta la traccia del programma che sta eseguendo e una immagine finale completa della memoria, l'una indipendentemente dall'altra.

La richiesta della immagine finale e della traccia si effettua predisponendo opportunamente due indicatori (celle del

7090); tale operazione non può essere effettuata dall'interno del programma simulatore e costituisce, come vedremo tra breve, una delle mansioni affidate al supervisore.
Il simulatore è dotato di un orologio, cioè di un contatore che viene incrementato ogni volta dopo la simulazione di una istruzione qualsiasi, e che contiene passo a passo il numero di cicli di macchina simulati trascorsi dall'inizio del programma. Formalmente è stato posto 1 ciclo = 1/300.000 di secondo. Per semplicità si è supposto che il CANE esegua ciascuna delle sue istruzioni in 2 cicli di macchina, eccezion fatta per la moltiplicazione e divisione che sono state considerate 5 volte più lente.
L'orologio permette di dotare il simulatore di un sistema di "supero orologio simulato". Possiamo cioè stabilire a priori il massimo numero di cicli simulati che vogliamo concedere ad un programma CANE, e fare in modo che il simulatore si arresti automaticamente se il contenuto dell'orologio raggiunge il massimo da noi stabilito. Il sistema di "supero orologio simulato" è molto semplice, basta infatti predisporre in un indicatore il numero di cicli massimo stabilito e confrontare il contenuto dell'orologio e dell'indicatore dopo ogni incremento dell'orologio. La predisposizione dell'indicatore di "supero orologio

simulato" rientra nelle mansioni affidate al programma supervisore.

Analizziamo ora il funzionamento del simulatore nella ipotesi che i tre indicatori di cui si è detto sopra siano stati predisposti dall'esterno.

1) Inizializzazione : si azzerano l'orologio e alcuni altri indicatori e contatori. Si fa un test sull'indicatore di traccia, alcune istruzioni del programma vengono modificate di conseguenza (*)

2) Analisi del contenuto della cella della memoria virtuale il cui indirizzo è indicato in rC (registro contatore istruzioni): i tre campi della istruzione vengono estratti e memorizzati separatamente.

3) Incremento del registro contatore istruzioni.

3a) Salvataggio del contenuto dei registri del CANE: si effettua solo se la routine di traccia è in funzione.

4) Riconoscimento codice operativo.

5) Calcolo indirizzo effettivo: non si effettua per le istruzio

(*) Il programma simulatore, salvo alcune subroutine, è stato scritto in Map.

ni di modifica e confronto dei registri indice.

6) Esecuzione della istruzione.

6a) Confronto del contenuto attuale dei registri del CANE con il contenuto degli stessi prima della esecuzione dell'istruzione: si effettua solo se la traccia è in funzione.

6b) Stampa di una riga di traccia: si effettua solo se la traccia è in funzione e se almeno uno dei registri è stato alterato durante la operazione 6.

7) Incremento orologio e controllo sul supero orologio simulato: se l'orologio ha raggiunto il valore contenuto nel relativo indicatore si procede alla operazione 8, altrimenti si torna alla operazione 2.

8) Stampa della immagine finale completa della memoria: si effettua solo se la immagine è stata richiesta.

Tutte queste operazioni sono sintetizzate nello schema dinamico seguente (fig.6).

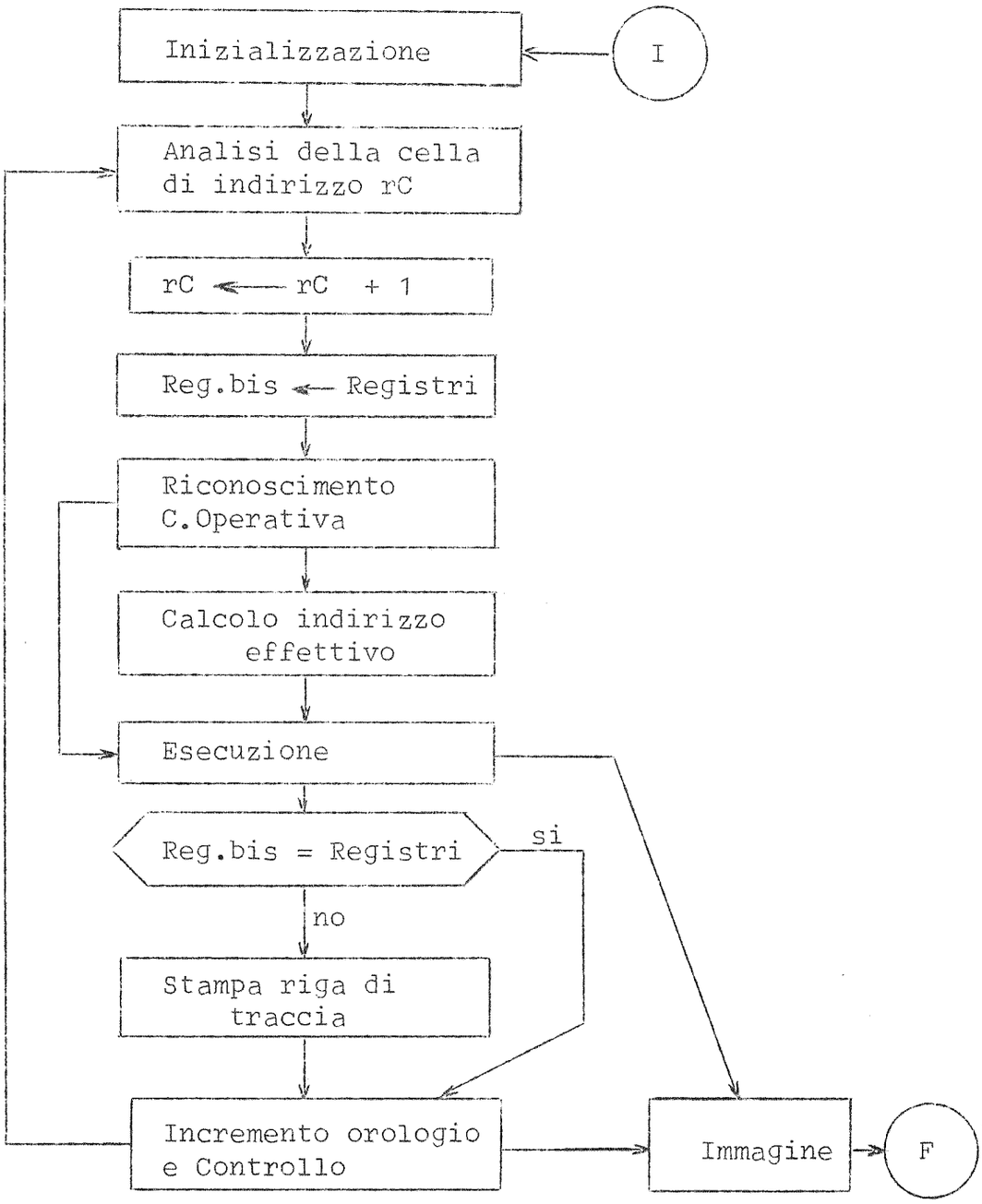

Inizializzazione
I
Analisi della cella di indirizzo rC
rC ← rC + 1
Reg.bis ← Registri
Riconoscimento C.Operativa
Calcolo indirizzo effettivo
Esecuzione
Reg.bis = Registri
si
no
Stampa riga di traccia
Incremento orologio e Controllo
Immagine
F

Fig. 6

Per quanto riguarda la velocità con cui la simulazione procede sul 7090, poiché la esecuzione di una istruzione di macchina del Cane richiede in media la esecuzione di una quarantina di istruzioni del 7090(corrispondenti a circa 80 cicli di macchina) si ha, ricordando che il ciclo base del 7090 è di 2.2 /usec e che il ciclo base del CANE è stato formalmente posto uguale a 3,3 /usec :

1 istruzione CANE $\simeq$ 2 cicli macchina CANE $\simeq$
200 /usec reali

cioè

1 ciclo macchina CANE = 3,3 /usec simulati $\simeq$ 100 /usec reali

e infine

1 /usec simulato $\simeq$ 30 /usec reali.

Concludendo si può dire il simulatore è circa 30 volte più lento della macchina intendendo in questo modo esprimere che se l'orologio del simulatore segna un numero di cicli corrispondenti formalmente a n secondi simulati, allora la simulazione ha tenuto impegnato il 7090 per 30xn secondi effettivi.

5.2 Funzionamento del supervisore

Il programma supervisore è stato in linea di massima elaborato per simulare un ipotetico operatore che pone volta a volta un singolo programma nel lettore di schede del CANE, ed aziona poi il dispositivo L di cui si è detto in 2.1 , mettendo in tal modo in funzione la macchina.

Infatti almeno in un primo tempo la esistenza e la funzione del supervisore non saranno illustrate agli studenti. Per il corretto funzionamento dei programmi supervisore e simulatore, ogni programma codificato in linguaggio CANE deve essere preceduto da una scheda (che diremo scheda asterisco) le cui prime colonne hanno il formato

*bbbbb

mentre nelle restanti colonne vengono scritte, oltre al nome del presentatore, le codifiche delle opzioni riguardanti la richiesta della traccia dell'immagine finale e del tempo simulato (TRC nelle colonne 64-66 per la traccia, IMG nelle colonne 70-72 per l'immagine, tre cifre decimali nelle colonne 52-54 che vengono interpretate come richiesta del tempo in secondi e in base alle quali viene predisposto l'indicatore di "supero

orol. S."). I programmi, ciascuno con la sua scheda asterisco vengono disposti uno sull'altro e il pacco complessivo, che deve essere terminato da una scheda le cui prime 6 colonne hanno la configurazione

*FINEb ,

viene inserito,in qualità di dati destinati alla elaborazione, dopo il programma simulatore-supervisore.

I compiti affidati al programma supervisore sono :

1) riconoscimento della scheda asterisco
2) stampa e analisi della scheda
3) predisposizione degli indicatori di Traccia di Immagine e di "supero orologio simulato".
4) simulazione del dispositivo L del CANE.
5) chiamata del simulatore.

Lo schema seguente illustra il funzionamento del supervisore e la maniera in cui dopo il termine della esecuzione di un singolo programma del pacco il simulatore richiama il supervisore.

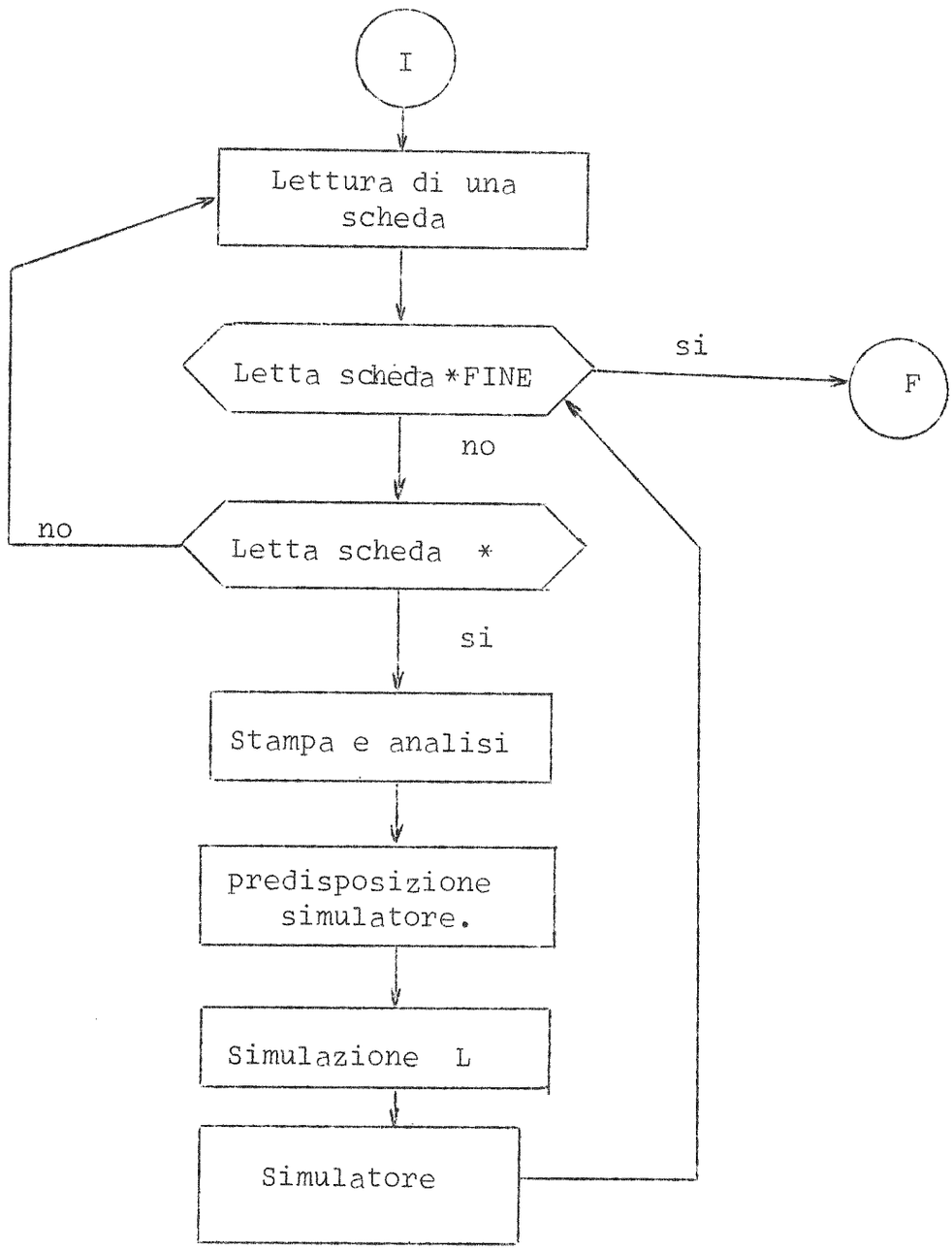

I
Lettura di una scheda
Letta scheda *FINE
si
F
no
no
Letta scheda *
si
Stampa e analisi
predisposizione simulatore.
Simulazione L
Simulatore

Fig. 7

Più in dettaglio le operazioni svolte dal supervisore dopo il riconoscimento della scheda asterisco sono :

1) stampa immediata della scheda asterisco

2) Analisi della scheda asterisco

3) Conseguente predisposizione degli indicatori

(Analisi della scheda)

4) Lettura di una scheda (*) con la modalità binaria del CANE, e sistemazione delle quattro parole in essa contenute nelle celle di indirizzo 000, 001, 002, 003.

5) Azzeramento del registro contatore istruzioni IC.

(simulazione del dispositivo L)

Si noti che la lettura della scheda asterisco (o della *FINE) può avvenire anche durante la simulazione qualora un programma tenti di leggere ancora dopo aver esaurito le sue schede di dati.

(*) Anche dopo questa operazione (come dopo ogni lettura di una scheda da parte del supervisore o da parte del simulatore) si controlla che le prime sei colonne non contengano nè *bbbbb nè *FINEb.

6. Operazioni di ingresso-uscita

In generale i calcolatori funzionano in modo da poter continuare normalmente la elaborazione durante le operazioni di ingresso uscita.

Cioè all'atto della esecuzione di una istruzione di ingresso-uscita da parte dell'unità centrale di processo viene messa in funzione la unità periferica richiamata e i dati cominciano ad entrare o ad uscire dalla memoria, mentre l'unità centrale passa ad eseguire le istruzioni successive in modo del tutto normale.

Nell'intervallo di tempo necessario per portare a termine una operazione di in-out l'unità centrale esegue circa 20.000 istruzioni, difatti in genere un calcolatore è in grado di effettuare circa 200.000 operazioni elementari di elaborazione al secondo mentre il lettore di schede legge circa 10 schede al secondo e la stampante stampa circa dieci righe al secondo.

Poiché il CANE continua la elaborazione dei dati in memoria durante le operazioni di ingresso-uscita e poiché d'altra parte il simulatore deve riprodurre tutte le caratteristiche della macchina, le operazioni di ingresso-uscita sono state simulate

in modo che dopo la esecuzione delle istruzioni di ingresso-uscita le istruzioni successive siano normalmente eseguite mentre i dati entrano o escono dalla memoria virtuale mano a mano che il contenuto dell'orologio aumenta.

Descriviamo ora in dettaglio come è organizzata la simulazione delle operazioni di ingresso uscita.

Supponiamo di aver fissato il numero di cicli di macchina da assegnare alle varie operazioni di ingresso-uscita del CANE. Sia ad esempio N il numero di cicli che competono alla lettura di una scheda in codice caratteri, (una scheda contiene 72 caratteri corrispondenti a 24 parole del CANE). Il simulatore funziona nel modo seguente: quando viene incontrata la istruzione ENA, il simulatore legge una scheda e sistema i 72 caratteri in essa contenuti in un'area di ingresso esterna alla memoria del CANE, poi passa ad eseguire le istruzioni successive, i caratteri vengono trasferiti nella memoria virtuale a gruppi di 6 (2 parole del CANE) mano a mano che il contenuto dell'orologio del simulatore aumenta. I singoli passi di ingresso avvengono successivamente quando l'orologio assume i valori:

$T_{in} + N/12$; $T_{in} + 2 \times N/12$ ....

.... $T_{in} + N$

essendo T_{in} il numero contenuto nell'orologio all'atto della esecuzione da parte del simulatore della istruzione di entrata ENA. La uscita dei dati avviene in modo del tutto analogo.I caratteri vengono trasferiti a gruppi di 6 in un'area di uscita esterna alla memoria virtuale; a conclusione della operazione i caratteri contenuti nella area di uscita vengono stampati. Lo schema seguente (fig.8) illustra come funziona la simulazione della istruzione e della operazione di uscita. N indica il numero di cicli formalmente assegnati alla operazione di uscita, n il numero di passi in cui il simulatore la esegue. Il significato degli indicatori Indout e clockout risulta dall'esame della figura. Il diagramma in fig. 8 è da inquadrare nel più ampio contesto di fig. 6 .In particolare il blocco "Incremento orologio e controllo" di fig. 8 corrisponde con il blocco analogo di fig. 6 . La parte del diagramma precedente il blocco "Incremento orologio e controllo" simula la esecuzione della istruzione USC, ed è quindi da pensare inserita nel blocco "Esecuzione" di fig. 6 .

La estensione al caso completo (ingresso-uscita) è molto semplice ed è illustrata nello schema successivo (fig.9).

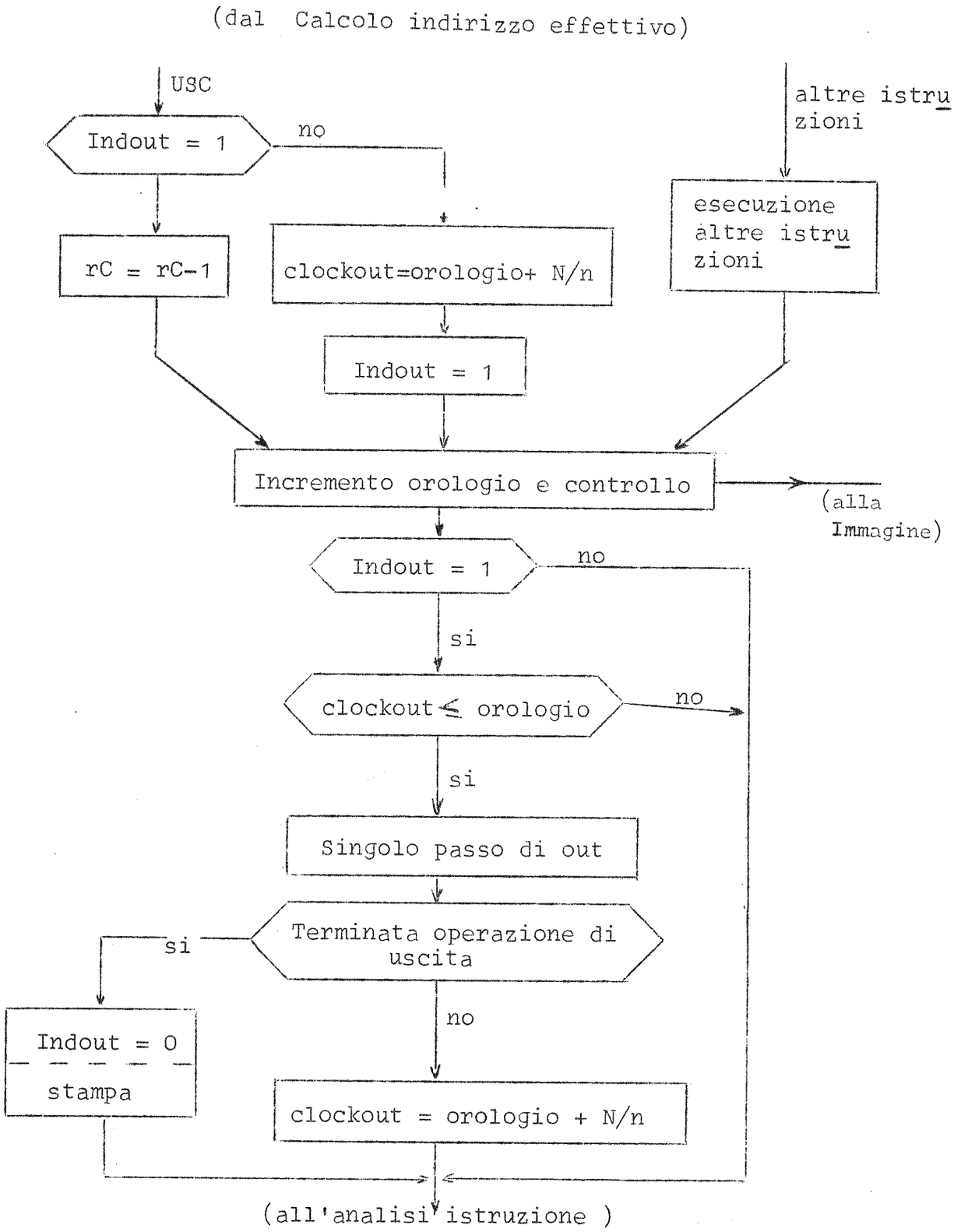

(dal Calcolo indirizzo effettivo)
USC
Indout = 1
no
rC = rC-1
clockout=orologio+ N/n
Indout = 1
altre istruzioni
esecuzione altre istruzioni
Incremento orologio e controllo
(alla Immagine)
Indout = 1
no
si
clockout ≤ orologio
no
si
Singolo passo di out
Terminata operazione di uscita
si
Indout = 0
stampa
no
clockout = orologio + N/n
(all'analisi istruzione)

Fig. 8

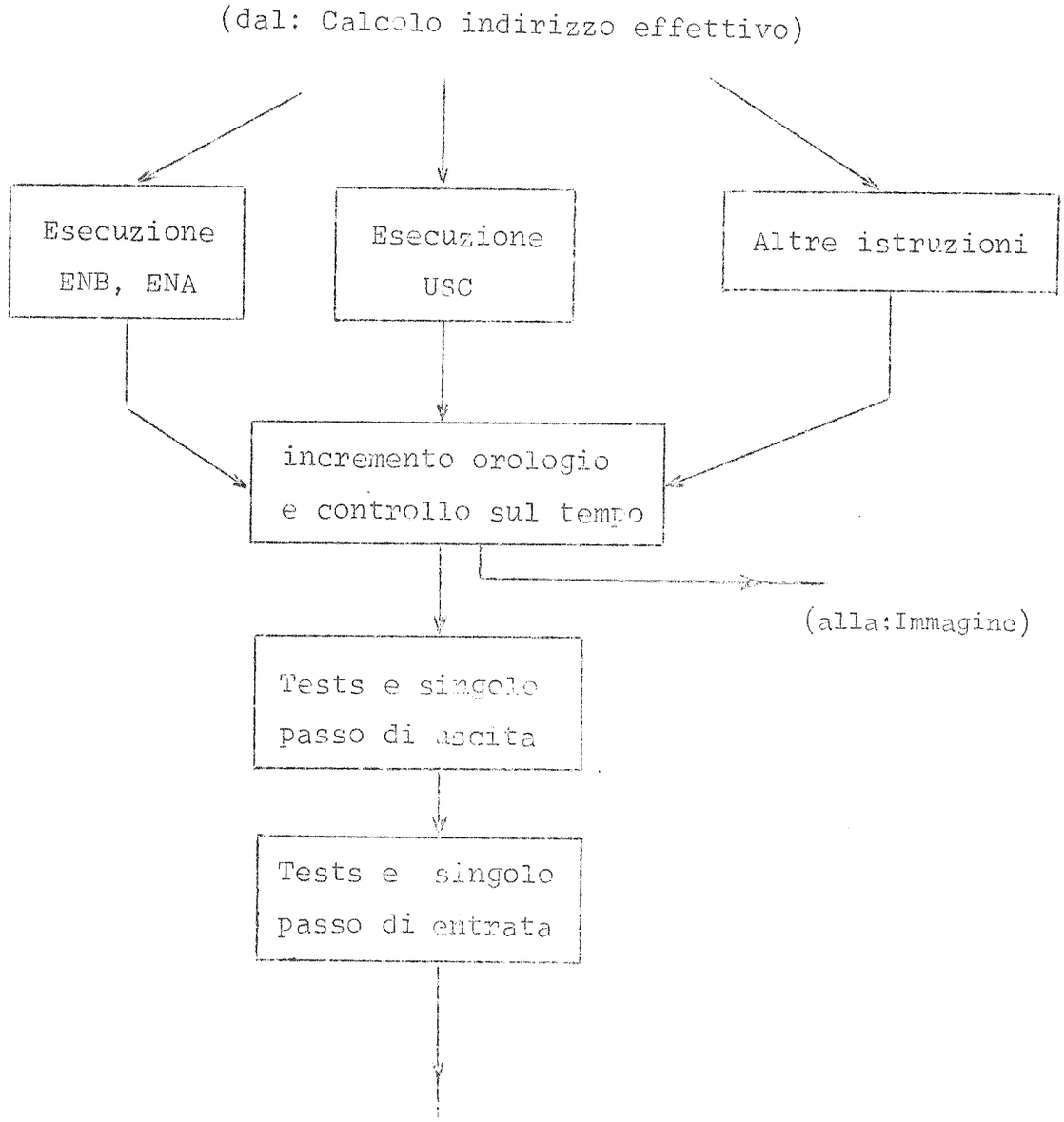

(dal: Calcolo indirizzo effettivo)
Esecuzione ENB, ENA
Esecuzione USC
Altre istruzioni
incremento orologio e controllo sul tempo
(alla:Immagine)
Tests e singolo passo di uscita
Tests e singolo passo di entrata
(all'analisi istruzione)

Fig. 9

Rimangono ancora da fare alcune osservazioni sulle operazioni di ingresso uscita, in particolare sul numero di cicli da assegnare loro, e sulle istruzioni di controllo CEN e CUS. Un lettore reale legge di solito circa 600 schede al minuto, e una stampante reale stampa circa 600 righe al minuto, poiché il ciclo base del CANE è stato assunto di 1/300.000 di secondo, il tempo impiegato dal lettore a leggere una scheda dovrebbe corrispondere a circa 30.000 cicli di macchina.

Cioè sarebbe da porre

$$N = 30.000 .$$

Perciò in un programma del tipo

```
.............
ENC     BUFFER
CEN        *
.............
```

supponendo di assegnare all'istruzione CEN due cicli di macchina il controllo dovrebbe essere eseguito 15.000 volte prima che il programma possa continuare.

Questo porta, ricordando che il simulatore è circa 30 volte più lento della macchina, che ogni operazione di ingresso-uscita duri circa 3 secondi effettivi sul 7090. E questo tempo trascorre in maniera del tutto inutile nei riguardi del proseguimento del

programma eseguito dal simulatore. Si è allora fatto in modo che le istruzioni di controllo CEN e CUS incrementino l'orologio di una quantità assai grande in modo che in pochi controlli il valore dell'orologio raggiunga il valore dell'indicatore clockin/out.Sussiste però ancora un caso in cui il funzionamento delle operazioni di ingresso-uscita e relativi controlli è inefficiente. Si supponga che il programmatore preveda di eseguire con il CANE 100 operazioni di ingresso-uscita, e chieda di conseguenza almeno 10 secondi di tempo (*), il programma non dura, se tutto procede come nelle intenzioni del programmatore, 5 minuti effettivi in quanto le operazioni di in-out del CANE sono state sveltite come detto sopra, ma se per errore il programmatore manda il simulatore in ciclo senza eseguire, nel caso più sfortunato, nessuna delle operazioni di ingresso-uscita previste, trascorrono effettivamente 5 minuti prima che l'orologio del simulatore raggiunga il valore 10 X 300.000.
Questo difetto è chiaramente dovuto al disquilibrio che è stato introdotto tra la elaborazione interna del CANE e le sue o-

---

(*) Qualora si chiedano n secondi di tempo il simulatore si ferma automaticamente quando l'orologio raggiunge il valore n x 300.000

perazioni di ingresso uscita, le quali nominalmente sono lente mentre in realtà sono eseguite assai velocemente dal simulatore.

Non rimane a questo punto altra soluzione che sveltire anche nominalmente le operazioni di ingresso-uscita del CANE.A dette operazioni è stato assegnato in definitiva un numero di cicli che corrisponde circa alla velocità con cui le informazioni verrebbero trasmesse da nastro a memoria centrale e viceversa su un calcolatore reale.