C S. C. E.
BIBLIOTECA
Pos. ARCHIVIO

14 - Marzo 1956

(2)

C.S.C.E.P.

Sezione Logico-Matematica

NOTA TECNICA N° 2 Riservata

Presentazione e discussione della lista e del codice delle istruzioni.

(A. Caracciolo; E. Fabri)

1. Introduzione.

In questa Nota presentiamo la lista (provvisoria) delle istruzioni della macchina in progetto, che la caratterizza completamente dal punto di vista logico matematico. Necessari complementi per la comprensione di questa Nota sono le due Note relative allo schema strutturale e alla programmazione. La presentazione della lista delle istruzioni e la discussione del loro modo di operare richiede, tuttavia, un ampio cenno su tali questioni. Ciò è dovuto in particolare al fatto che, sebbene lo schema strutturale differisca apparentemente di poco da quello tradizionale, esso si ispira a una concezione essenzialmente nuova, alla cui origine si trovano importanti problemi da risolvere in fatto di programmazione.

Seguendo un orientamento largamente diffuso ci siamo attenuti fondamentalmente al tipo di macchine con istruzioni a un indirizzo. Come vedremo, tuttavia, nel nostro caso ogni istruzione conterrà l'indicazione di due indirizzi supplementari. Tali indirizzi sono, però, in relazione con il problema della modifica automatica delle istruzioni, di cui diremo in seguito, e svolgono una funzione essenzialmente diversa da quella che assolvono nelle attuali macchine a più indiriz

zi, ciò che non permette di classificare fra queste la macchina in progetto.

Nello schema classico delle macchine a un indirizzo ogni istruzione contiene due dati: il simbolo dell'operazione (in senso lato) da eseguire e l'indirizzo di una cella della memoria da cui estrarre un operando o in cui inviare un risultato. Prescindiamo, in questa discussione, da tutte le istruzioni riguardanti i trasferimenti con l'esterno (entrata, uscita) e con un eventuale tamburo magnetico. Talvolta, come ad es. nelle istruzioni di salto o di traslazione, le cifre dell'indirizzo hanno un significato diverso da quello indicato precedentemente, ma in ogni caso resta il fatto che una singola istruzione è caratterizzata da un primo gruppo di cifre che rappresenta, in opportuno codice, il simbolo d'operazione e da un secondo gruppo di cifre che, salvo alcune eccezioni, rappresenta un indirizzo.

Il fatto che le istruzioni vengono tutte rappresentate mediante simboli numerici e conservate nella memoria, permette di eseguire in modo automatico anche eventuali modifiche delle istruzioni. E' possibile cioè, mediante semplici istruzioni ordinarie, inviare nell'unità aritmetica parole che rappresentano istruzioni: sommando a tale parola il contenuto di un'altra cella della memoria e rispedendola nella memoria si può ottenere una modifica delle istruzioni di un dato tratto di programma. Come risulterà chiaro dal seguito, la necessità di apportare modifiche, lungi dall'essere un fatto eccezionale costituisce quasi la regola. La possibilità che queste macchine hanno di modificare, praticamente senza limiti, un programma originario, permette sia di condensare quest'ultimo in una specie di superprogramma che generi, per così dire, tutta

una serie di programmi che saranno poi effettivamente eseguiti uno per volta dalla macchina, sia di formulare programmi o parti di programma di uso corrente in forma assai generale, aggiungendo alcuni tratti di superprogramma il cui scopo è di adattare il programma generale al caso particolare in questione.

Nel primo caso rientrano, per es., tutti i calcoli in cui si debba ripetere un certo numero di volte una data successione di istruzioni su operandi di volta in volta diversi, cioè contenuti in celle diverse della memoria. Il programma originario conterrà istruzione per istruzione, oltre ai simboli delle operazioni da eseguirsi, certi indirizzi, per es. quelli delle celle in cui è contenuto il primo gruppo di operandi. Eseguito tale calcolo, il programma non conterrà in genere ripetute le stesse istruzioni, con gli indirizzi relativi al secondo gruppo di operandi, bensì un tratto di superprogramma eseguendo il quale la macchina stessa sostituisce al programma originario il programma modificato per la ripetizione del calcolo sul secondo gruppo e via di seguito.

Nel secondo caso rientrano tutti gli esempi di programmi che è possibile formulare in modo da coprire un certo numero di casi particolari che si ottengono precisando il valore di certi parametri che nella formulazione originaria sono lasciati indeterminati. Tali sono, ad es., il programma per la soluzione di un generico sistema di equazioni lineari in n incognite o di un'equazione differenziale di ordine n, come pure il calcolo di uno sviluppo fino a una data precisione ε e via dicendo. In tutti questi casi si formula un programma generale in forma standard, cioè indipendente dal valore dei parametri n, ε, etc.

Il più delle volte, tali due eventualità si presentano insieme: segue da tutto ciò la possibilità di preparare in forma standard programmi molto generali e molto condensati che la macchina stessa interpreta, modifica e sviluppa, magari in un modo o in un altro a seconda del verificarsi o meno di certe eventualità in dati punti del programma stesso.

Quanto detto vale praticamente per tutte le moderne calcolatrici elettroniche a cifre, indipendentemente dalle particolari caratteristiche di ciascuna. Turing, anzi, ha esposto già in un lavoro del 1936 un semplicissimo modello, teoricamente equivalente a qualsiasi calcolatrice automatica a cifre e universale nel senso di essere capace di calcolare qualsiasi grandezza, purché sia dato un procedimento aritmetico preciso per tale calcolo. Ciononostante, dal punto di vista pratico, la versatilità teorica, deve essere accompagnata da un elevato grado di efficienza, cioè dalla capacità di eseguire tutto ciò in modo semplice, rapido, e sicuro. La sicurezza riguarda problemi essenzialmente tecnici: dal presente punto di vista - quello logico-matematico - si può dire solo che essa implica la ricerca di uno schema strutturale tecnicamente semplice e col minor numero di componenti. La rapidità, invece, oltre che dalla rapidità intrinseca della macchina dipende, in modo assai complesso, anche dall'organizzazione della programmazione, poiché il tempo impiegato per un dato calcolo, dipende ovviamente dal numero di istruzioni che la macchina deve eseguire per giungere al risultato voluto. Oltre che breve, un programma deve essere anche il più possibile semplice. Benché, teoricamente, programmare un problema, una volta che esso sia stato espresso in

termini aritmetici precisi, non dovrebbe costituire una grossa difficoltà, in pratica le cose stanno assai diversamente. La necessità di ricorrere a programmi condensati e modificabili rende, infatti, piuttosto difficile l'opera del programmatore, ciò che è sovente causa di errori nella programmazione e di grande spreco di tempo e di lavoro. Al tempo stesso, le istruzioni da modificare costituiscono in pratica la quasi totalità delle istruzioni di un programma, onde gran parte del tempo di lavoro della macchina è necessariamente dedicato a quei tratti, che abbiamo chiamato di superprogramma, che effettuano la modifica dei tratti di programma veri e propri. Accade sovente che i primi siano molto più lunghi e complessi dei secondi, onde si impone la necessità di esaminare a fondo il problema della loro semplificazione e di studiare sistematicamente la teoria della programmazione e la tecnica della composizione dei programmi. E' in preparazione una prima Nota a questo riguardo, con riferimento all'organizzazione logico-strutturale della nostra macchina, il cui progetto è stato elaborato tenendo conto sin dall'inizio delle esigenze relative alla programmazione e alla modifica delle istruzioni.

Come è già stato osservato in uno dei Seminari diffuso a cura del Centro (Dott. E. Fabri - Considerazioni sulla progettazione di una calcolatrice elettronica) questo problema riveste vitale importanza al giorno d'oggi in cui il progresso tecnico è già in grado di fornire macchine con elevato coefficiente di sicurezza, mentre molto minore è stato il progresso dopo i fondamentali lavori di Von Neumann e Goldstine nel 1947-48 nel settore dell'organizzazione logica della programmazione e nella modifica dello schema classico del

le calcolatrici per agevolare la soluzione di tale problema.

2. Critica del I° schema e criteri per la sua modifica.

Da alcuni studi condotti su tale argomento, risultò un primo schema di macchina, descritta nella Nota 1. Tale schema prevedeva l'aggiunta, nel controllo della calcolatrice elettronica di una piccola memoria supplementare (Registri di Controllo - R.C.) e di un Addizionatore separato mediante i quali effettuare automaticamente, durante la fase di istruzione, la modifica di un'istruzione; ciò si otteneva sommando, prima della decodificazione dell'istruzione, il contenuto di uno dei R.C. alla parola che la rappresenta. Questo procedimento è stato introdotto sulle macchine costruite dalla ditta Ferranti sul prototipo realizzato presso l'Università di Manchester. La soluzione da noi adottata differiva, però, sotto alcuni rispetti da quella della Ferranti, in quanto era prevista la possibilità di introdurre una modifica permanente nell'istruzione oltre a una semplice modifica temporanea. L'esame di alcuni programmi e il confronto con i metodi classici di programmazione ha rivelato che la via seguita era effettivamente buona.

La realizzazione fisica, dello schema di cui alla Nota 1, presentava, tuttavia una certa complessità; anche dal punto di vista logico si ravvisava l'opportunità di riesaminare lo schema per migliorarne ulteriormente le prestazioni.

Sono stati esaminati diversi possibili schemi cercando di giungere a una soluzione tecnicamente più semplice e al tempo stesso anche più flessibile dal punto di vista della programmazione; precisamente, semplificare lo schema strutturale eliminando i Registri e l'Addizionatore di Controllo affidan-

done le funzioni, rispettivamente, alla memoria principale e all'Addizionatore dell'Unità Aritmetica; aumentare il numero delle istruzioni a disposizione oltre alle 32 possibili col vecchio schema; e infine rendere possibile l'esecuzione anche di più modifiche, rinunciando alla modificabilità del simbolo d'operazione e alle modifiche permanenti, le quali, se permettono un certo risparmio di tempo, presentano alcuni svantaggi nel caso di cicli molto complessi. Il nuovo schema strutturale è ampiamente descritto nella Nota 3. In questa nota ci limiteremo a una presentazione schematica delle istruzioni, illustrando gli aspetti principali sulla base di un semplice programma aritmetico. Questo ci permetterà di istituire un confronto fra questa e le altre macchine a un indirizzo, e precisare, così, le considerazioni generali sopra esposte.

3. Notazioni.

Nel seguito converrà distinguere accuratamente fra celle, contenuti, nomi, indirizzi e via dicendo. Faremo ciò mediante il seguente simbolismo.

Innanzitutto distinguiamo fra nomi di celle o registri (che indicheremo con lettere maiuscole) e simboli numerici (che indicheremo con lettere minuscole o con cifre arabe). Questi ultimi costituiscono parole o parti di parole e possono rappresentare sia dati numerici che istruzioni o parti di queste (simboli d'operazione, indirizzi etc.). Per distinguere, nel caso delle istruzioni, fra le diverse parti di un'istruzione, conviene suddividere una cella in più cellule, ciascuna delle quali può essere indicata con il nome della cella munito di un indice. Per passare dalle celle alle cellule e viceversa faremo uso dei due operatori seguenti:

i : ... l'i.ma cellula della cella ..... seguito da un nome di cella.

: i ... la cella la cui i.ma cellula è... seguito da un nome di cellula.

Per passare dalle celle o dalle cellule ai rispettivi contenuti faremo uso delle seguenti coppie di operatori:

$\not\emptyset$ ... il contenuto de (la cella)..... seguito da un nome di cella.

$\emptyset$ ... la cella che contiene ..... seguito da un simbolo di parola.

$\not\beta$ ... il contenuto de (la cellula) ..... seguito da un nome di cellula.

β ... la cellula che contiene ..... seguito da un simbolo d'intero.

Per parola si intende un gruppo di 36 cifre binarie, per intero un gruppo di 12 cifre binarie (la ragione di tale ultima denominazione sta nel fatto che in genere essa rappresenta un numero intero modulo 2^{12}).

Per es. $a = \not\emptyset A$: il contenuto della cella A.

$A = \emptyset a$: la cella che contiene a.

$\not\beta A_i = 14$: il contenuto della i.ma cellula di A è 14.

$A_i = \beta 14$: A_i è la cellula che contiene 14.

$i : A = A_i$: la i.ma cellula di A.

$: i\ A_i = A$: A è la cellula di cui la i.ma cellula è A_i.

Gli operatori $\not\emptyset$ $\not\beta$ sono gli inversi di $\emptyset$ e β; i primi trasformano nomi (maiuscole) in contenuti (minuscole), i secondi viceversa.

Oltre a questi bisogna introdurre gli operatori che fanno passare dal nome all'indirizzo ($\not\alpha$) e viceversa (β).

$\not\alpha$ ... l'indirizzo de (la cella) ..... seguito da un nome di cella.

β ... la cella il cui indirizzo è ... seguito da un simbolo di intero.

Per es.: il contenuto della terza cellula della cella il cui indirizzo è contenuto nella seconda cellula del registro C,

si scrive : ∅ 3 : ∅ ∅ 2 : C

La suddivisione di una cella in cellule è relativa alla struttura delle istruzioni e al numero di simboli o gruppi di simboli che compaiono in una parola. Per es. in una macchina ordinaria a un indirizzo con due istruzioni per parola si dovrebbero distinguere quattro cellule, due per i simboli d'operazione e due per gli indirizzi. Per le nostre istruzioni, come si vedrà, conviene introdurre tre cellule (di 12 bit ciascuna).

Per ragioni connesse con l'uso delle celle parametriche, che fanno uso di indirizzi relativi, bisogna introdurre poi altri operatori:

∅ ... l'indirizzo L-relativo di ..... seguito da un nome di cella.

∅ ... la cella il cui indirizzo L-relativo è ... seguito da un simbolo d'intero.

∅ ... l'indirizzo M-relativo di ..... seguito da un nome di cella.

∅ ... la cella il cui indirizzo M-relativo è... seguito da un simbolo d'intero.

∅ ... l'indirizzo N-relativo di ..... seguito da un nome di cella.

∅ ... la cella il cui indirizzo N-relativo è... seguito da un simbolo d'intero.

Un operatore, infine, di uso generale è necessario per indicare la parola o la parte di parola che rappresenta un simbolo

mnemonico o simili.

$\not{\varepsilon}$ ... la rappresentazione di .....

Questo simbolismo, potrà apparire a prima vista complesso: tuttavia, esso è di grande aiuto nel dare forma precisa a complesse descrizioni verbali, sia dal punto di vista della programmazione che della descrizione del modo di operare della macchina.

4. Esempio di programma.

Passiamo ora ad illustrare le considerazioni svolte precedentemente sui vari tipi di istruzioni a un indirizzo, studiando un esempio concreto di programma per il calcolo del prodotto di una matrice $\underline{\underline{T}}$ per un vettore $\underline{X}$: $Y = \underline{\underline{T}}\,\underline{X}$. Indichiamo con t_{ji} gli elementi della matrice $\underline{\underline{T}}$, con x_i le componenti di $\underline{X}$ e con y_j le componenti di $\underline{Y}$.

Indichiamo poi con $t = \not{d}\, T = \not{d}\, \not{\emptyset}\, t_{oo}$ l'indirizzo della cella (T) che contiene l'elemento t_{oo} e supponiamo che gli elementi della matrice $\underline{T}$ siano contenuti, riga per riga, nelle celle di indirizzi da $t + j\,k$ a $t + j\,k + r - 1$ $(j = 0,1,\ldots r-1)$ sicché $\not{d}\, \not{\emptyset}\, t_{ji} = t + j\,k + i$. Analogamente detto x l'indirizzo della cella che contine x_o supporremo che $\not{d}\, \not{\emptyset}\, x_i = x + i$ $(i = 0,1\ldots r-1)$. Supponiamo infine che gli indirizzi delle celle in cui si devono formare le componenti y_j del vettore $\underline{Y}$ siano $y + j = \not{\emptyset}\, \not{d}\, y_j$.

Il calcolo si eseguirà ripetendo r volte il calcolo del prodotto scalare $y_j = \underline{T}_j\,\underline{X}$, che a sua volta si ottiene con un ciclo ripetuto r volte in base alla formula ricorrente :

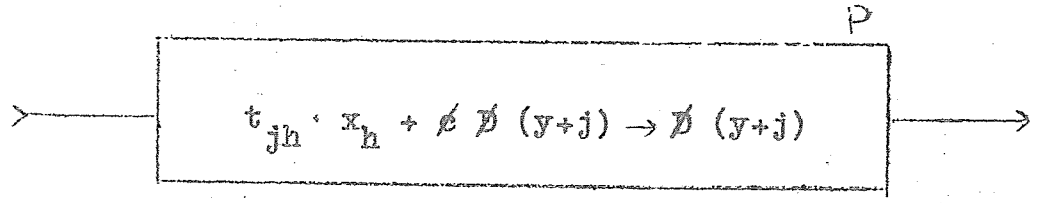
$$y_j^{(h+1)} = t_{jh} \cdot x_h + y_j^{(h)} \qquad (h = 0,1,\ldots r-1)$$

(1)

$$y_j^{(o)} = 0 \qquad\qquad y_j^{(r)} = y_j$$

Prima di formulare il programma vero e proprio, è opportuno disegnarne il relativo diagramma dinamico, che illustri con diversi "blocchi" le fasi del calcolo e il loro concatenamento. La teoria dei diagrammi dinamici verrà discussa nella Nota sulla programmazione. Nel caso presente si ottiene un diagramma dinamico piuttosto semplice e intuitivamente evidente.

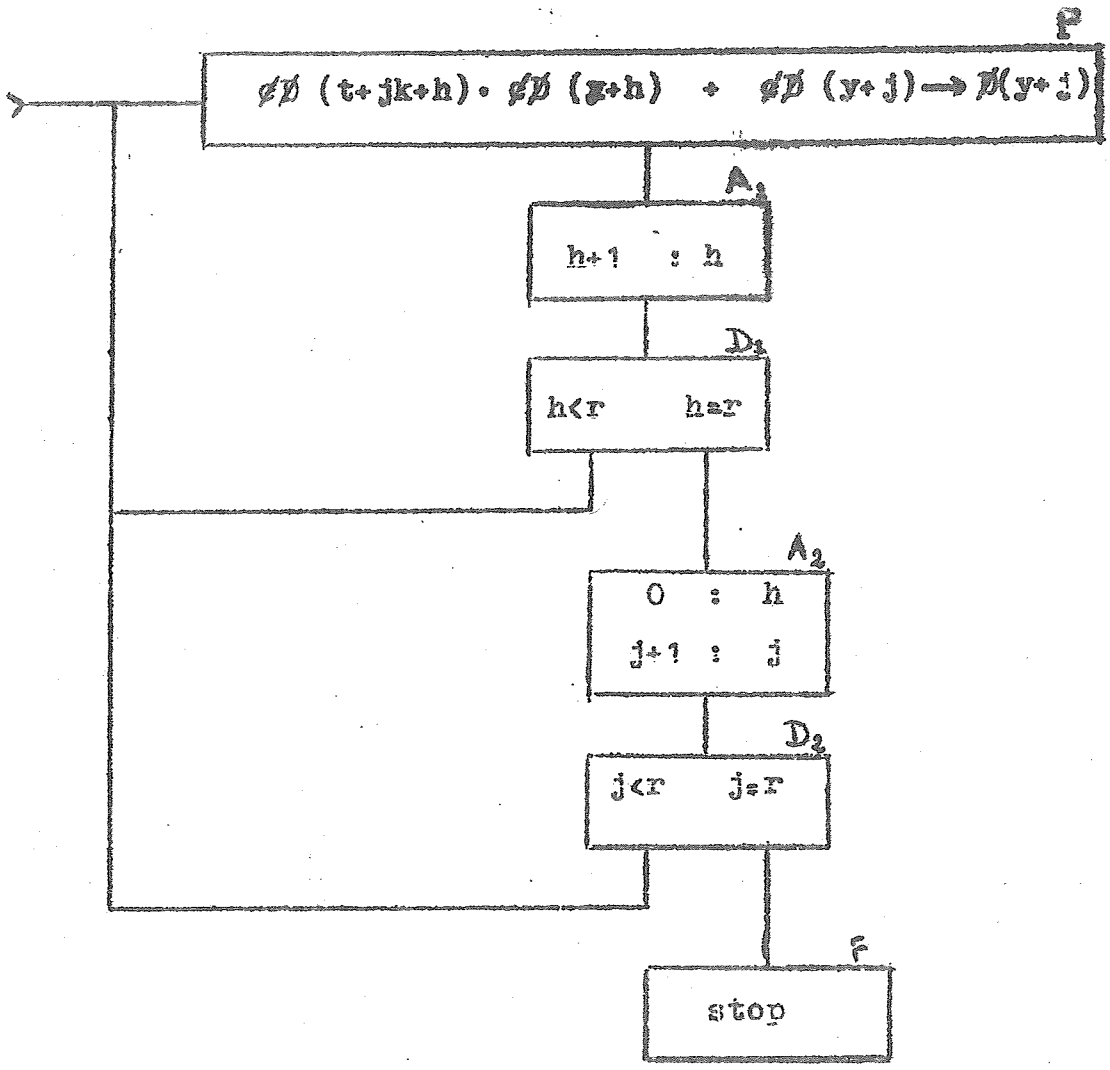
Indichiamo con P il blocco principale di operazioni, relativo all'esecuzione di un singolo passo della relazione (1) e cioè :



(o in parole: moltiplicare $t_{jh} = \not{\varepsilon}\, \not{D}\, (t+jk+h)$ per $x_h = \not{\varepsilon}\, \not{D}\, (x+h)$ e sommarlo al contenuto di $\not{D}\, (y+j)$ (che è $y_j^{(h)}$); trasferire il numero ottenuto nella cella di indirizzo $y+j$).

Ripetendo questo calcolo, per j fisso, r volte a partire da h = 0 fino a h = r , si ottiene la j.ma componente del vettore Y. Per ottenere la componente successiva y_{j+1}, basterà aumentare j di 1 e ripetere il ciclo precedente, nuovamente a partire da h = 0.

L'intero calcolo si comporrà perciò, come risulta chiaramente dal diagramma, di un doppio ciclo su h e su j.



Dal punto di vista logico i blocchi A_1, D_1, e A_2, D_2 indicano nel complesso le seguenti modifiche e decisioni : aumentare h di 1; se h+1 < r sostituire h+1 a h e ripetere, se h+1 = r porre h = 0 e aumentare j di 1; confrontare j+1 con r; se j+1 < r sostituire j+1 a j e ripetere, se j+1 = = r fermare il calcolo (stop). Ad essi corrispondono nella programmazione effettiva di una macchina, tratti di superprogramma, che modificano le istruzioni fondamentali del blocco P, in modo che esse assumano via via nello svolgersi dei cicli la forma di volta in volta necessaria.

Prima di esaminare e discutere tali blocchi, consideriamo il blocco P. Scriveremo le istruzioni corrispondenti fa

cendo uso del codice mnemonico usato nel nostro progetto e di cui trascriviamo qui i simboli che ci servono in questo caso:

n+A	x	cancellare l'Accumulatore A e sommarci ¢ Ø x.
n+B	x	cancellare il Moltiplicatore B e sommarci ¢ Ø x.
A+A	x	sommare ¢ Ø x a ¢ A ; ¢ Ø x + ¢ A → A
MC	x	moltiplicare ¢ Ø x per ¢ Ø: il risultato si forma in A ; ¢ Ø x · ¢ Ø → A .
AS	x	trasferire ¢ A nella cella di indirizzo x ; ¢ A → Ø x .
Z+A	x	se ¢ A < 0 passare all'istruzione successiva; se ¢ A ≥ 0 saltare all'istruzione ¢ Ø x, cioè all'istruzione contenuta nella cella di indirizzo x.
F	-	arresto (stop).

Il blocco P consiste allora della seguente successione di istruzioni :

P	n+B	t+jk+h
	Mc	x+h
	A+A	y+j
	AS	y+j

Tale blocco è praticamente identico per tutte le macchine a un indirizzo (tralasciamo come inessenziale a questo discorso il fatto che alcune macchine - generalmente in serie - posseggono un'istruzione di moltiplicare sommando che permette di abbreviare e organizzare diversamente il calcolo ed eventuali accorgimenti per il controllo del traboccamento e via dicendo). Le cose stanno invece assai diversamente, per quanto riguarda gli altri blocchi fra le macchine a schema classico

e quelle dotate di registri di controllo, o di sistemi simili (celle parametriche) come nella nostra organizzazione.

P	p	n+B	t+jk+h	p	I	n+B	t+	p	n+B	⊥T	02	/	0
	p+1	Mc	x+h	p+1	H	Mc	x+r	p+1	Mc	⊥X	02	/	0
	p+2	A+A	y+j	p+2	J	A+A	y+r	p+2	A+A	⊥Y	02	/	0
	p+3	A$S	y+j	p+3	J	A$S	y+r	p+3	AS	⊥Y	02	/	0
A_1	p+4	n+A	p	p+4	H	C+	p+12	p+4	C+	/	00	µH	1
	p+5	A+A	p+25										
	p+6	A$S	p										
	p+7	n+A	p+1	p+5	I	C+	p+12						
	p+8	A+A	p+25										
	p+9	A$S	p+1										
D_1	p+10	A+A	p+26	p+6	H	Z-	p	p+5	Z-	/	10	µH	-5
	p+11	Z+A	p										
A_2	p+12	A+A	p+28	p+7	H	C+	p+13	p+6	C+	⊥R	01	µH	0
	p+13	A$S	p+1										
	p+14	n+A	p	p+8	I	C+	p+14	p+7	C+	⊥K	01	µT	0
	p+15	A+A	p+28										
	p+16	A$S	p										
	p+17	n+A	p+2	p+9	J	C+	p+12	p+8	C+	/	00	µT	1
	p+18	A+A	p+25										
	p+19	A$S	p+2										
	p+20	A+A	p+28										
	p+21	A$S	p+3										

D_2	p+22	A+A	p+29	p+10	J	Z-	p	p+9	Z+	/	10	~~M~~J	-9
	p+23	Z+A	p										
F	p+24	F	—	p+11	—	F	—	p+10	F	/	/	/	/
	p+24		1	p+12		1							
	p+25	-~~X~~(Mc	x+r)	p+13		-r							
	p+26	~~X~~(Mc	x)	p+14		k-r							
	p+27		k-r										
	p+28	~~X~~(AS)-~~X~~(A+A)						T	~~X~~H-~~X~~T		0	(jk)+t+r	
	p+29	-~~X~~(AS	y+r)					X	~~X~~H-~~X~~X		0	x+r	
								Y	~~X~~J-~~X~~Y		0	y+r	
				I	(jk-h)			H	0		0	(h)-r	
				H	(h)-r			J	0		0	(j)-r	
				J	(j)-r			R	0		0	-r	
								K	0		0	R	

Riportiamo direttamente su tre colonne l'intero programma come viene scritto (I) per un'ordinaria macchina a un indirizzo, (II) per una macchina dotata di registri di controllo con la possibilità di una sola modifica temporanea (del tipo della Ferranti) e (III) per la macchina in progetto con celle parametriche a modifiche multiple. Si vede subito da un semplice confronto che gli ultimi due programmi risultano enormemente più brevi. La (III) colonna presenta poi un'istruzione in meno rispetto alla (II): in programmi molto più complicati da un punto di vista logico la differenza fra gli ultimi due tipi risulta assai più considerevole. Tuttavia, già in questo

programma figura fra le colonne (II) e (III) un'altra differenza meno appariscente, ma non meno importante: nell'ultima colonna di (III), che è quella riservata agli indirizzi figurano solo costanti, mentre nella colonna corrispondente di (II) compaiono delle lettere che indicano indirizzi particolari che possono assumere di volta in volta valori diversi (parametri posizionali e di lavoro). Ciò significa che mentre la (III) è espressa in forma invariante rispetto al valore di tali parametri, ciò non accade per la (II) che deve essere previamente adattata al caso in questione. Non è possibile entrare ora in dettagli su tale complesso argomento, che è connesso con l'uso di sottoprogrammi standard, e con i diversi sistemi escogitati per agevolare l'introduzione dei parametri all'inizio di un dato sottoprogramma. Basterà una sola considerazione: comunque si dispongano le cose il compito sarà tanto più agevole quanto più piccolo è il numero delle istruzioni o delle C.P. in cui si dovranno introdurre tali parametri. Come risulta dalla tabella, nelle prime due colonne bisogna introdurre dei parametri in quasi tutte le istruzioni del programma, mentre nella (III) colonna i parametri vanno introdotti solo nelle celle parametriche, ciò che, come sarà mostrato in una prossima nota, è un'operazione molto più semplice e che richiede molte meno istruzioni. Una considerazione speciale merita il parametro di posizione, indicato in questo caso con p, che rappresenta l'indirizzo della prima istruzione effettiva del programma. E' possibile assegnare, per ogni sottoprogramma un valore fisso a tale parametro, ciò che evita la sua introduzione in un gran numero di istruzioni: ma questo implica che ogni volta che si

faccia uso di tale sottoprogramma, esso deve sempre occupare le stesse celle della memoria, imponendo un vincolo spesso fastidioso sulla distribuzione delle varie informazioni nella memoria principale. Nel nostro progetto, sebbene sia possibile trattare assai semplicemente questo parametro come gli altri, abbiamo introdotto un particolare accorgimento, di cui si dirà in seguito, per eseguire nel modo più semplice l'introduzione del parametro di posizione.

Queste considerazioni generali, basate esclusivamente sul confronto esteriore della forma del programma, sono già sufficienti per illustrare nelle grandi linee l'opportunità di ricorrere a un'organizzazione logica della macchina che permetta di introdurre una modifica automatica delle istruzioni, scostandosi dallo schema tradizionale. Mostreremo come tutto ciò possa accompagnarsi con uno schema strutturale assai semplice, ciò che ha grandissimo peso dal punto di vista economico e tecnico-costruttivo.

Tuttavia prima di riportare lo schema strutturale e di presentare la lista delle istruzioni converrà precisare, con esempi tratti dal programma indicato, l'uso dei R.C. o delle parametriche.

5. Analisi della funzione dei Registri di Controllo.

Come risulta dal relativo diagramma dinamico le istruzioni del blocco P vanno ripetute diverse volte (in questo caso r^2), aumentando però di volta in volta le cifre corrispondenti all'indirizzo (sostituire h+1, j+1 rispettivamente a h e j). Nella colonna (I) ciò viene eseguito inviando praticamente ad una ad una le istruzioni da modificare nell'unità aritmetica, sommandoci l'opportuno contenuto di una cella della memoria e rispedendo al suo posto l'istruzione così modifi

cata. Occorrono cioè, salvo la possibilità di ricorrere a qualche artificio (si veda la modifica dell'istruzione ¢ Ø(p+3), circa 3 istruzioni per ogni istruzione da modificare. Questo è quanto fanno le istruzioni dei blocchi A_1 e A_2. Per le decisioni si procede nel modo seguente: consideriamo l'istruzione ¢ Ø(p+1) : (Mc x+h) : ogni ciclo su h inizia con h=0 e deve terminare con h=r, cioè quando in Ø(p+1) si è formato ¢(Mc x+r); ciò è esattamente quanto fanno le due istruzioni del blocco D_1, mediante un confronto da effettuarsi nell'unità aritmetica, fra l'istruzione ¢Ø(p+1) dopo ogni aumento dell'indirizzo e la forma che assume al termine di un ciclo.

Ambedue questi tipi di operazioni vengono semplificati dall'uso dei R.C. o delle C.P. Consideriamo dapprima il caso di una macchina con R.C. con programma del tipo descritto nella (II) colonna. Osservato che la modifica di un'istruzione consiste essenzialmente nel sommare qualcosa all'indirizzo (trascuriamo qui le considerazioni relative alle modifiche di simboli d'operazione) si organizzano le cose nel modo seguente. Ai due gruppi di cifre che costituiscono l'istruzione effettiva se ne aggiunge un terzo, che rappresenta l'indirizzo di uno dei R.C. In tale registro si scrive la parte variabile dell'indirizzo, che deve essere sommata ogni volta all'indirizzo che figura nell'istruzione, prima che l'istruzione stessa venga eseguita, in modo che l'indirizzo risultante sia proprio quello giusto. Così, ad es., la seconda istruzione della (II) colonna H, Mc, x+r, viene interpretata ed eseguita dalla macchina come Mc x+r+Ø2:H, ove Ø2:H indica il contenuto della 2.ª cellula del R.C. H che contiene h-r, cosicché l'indirizzo effettivo risulta x+h in accordo con l'istruzione nella colonna (I). Aumentare h di 1, significa ora semplicemente aumentare di 1 il contenuto del R.C.

H, ciò che può farsi direttamente con l'istruzione speciale H C+ p+12, che significa: sommare al R.C. il contenuto della cella p+12 (la quale in questo caso deve contenere la costante 1). La ragione per cui all'indirizzo x dell'istruzione considerata viene aggiunto il termine r, che viene poi sottratto in H è connessa con l'uso dei R.C. nelle decisioni, ovverosia all'uscita di un ciclo. Questa deve effettuarsi, nel nostro caso, quando h diventa = r. Mentre non sarebbe facile riconoscere ciò direttamente, il procedimento indicato porta a sostituire tale condizione con l'altra $\not{c}$ H = 0. La decisione avviene quindi con un salto condizionato (H Z+ p) sul R.C. H del tutto analogo a quelli ordinariamente previsti sull'Accumulatore dell'unità aritmetica.

E' noto il funzionamento ordinario di una macchina classica a un indirizzo. L'istruzione il cui indirizzo è contenuto nel numeratore N passa all'inizio della fase di istruzione nel registro delle istruzioni. Quindi viene decodificata ed eseguita (fase di esecuzione). Segue infine la fase di preparazione della nuova istruzione, che sebbene logicamente distinta, si può far rientrare senz'altro nella precedente.

Con l'introduzione dei R.C. (e delle C.P.) la prima fase viene spezzata in due tempi: la fase d'entrata, in cui l'istruzione giunge come prima dalla memoria al controllo, e la fase di modifica, durante la quale alla parola giunta dalla memoria viene sommato il contenuto di un R.C. il cui indirizzo è contenuto nell'istruzione stessa. Con la nostra organizzazione è possibile introdurre una modifica multipla, cioè sommare all'istruzione successivamente il contenuto di più C.P. secondo un procedimento che descriveremo fra breve. Terminata comunque la modifica, semplice o multipla, segue la fase

d'esecuzione praticamente identica a quelle delle macchine ordinarie.

Prima di parlare delle modifiche multiple, occorre però discutere un po' meglio le istruzioni speciali. Le due istruzioni speciali introdotte nel programma della (II) colonna sono H C+ z e H Z+ z. In entrambe queste istruzioni l'indirizzo[1] H del R.C. ha una funzione diversa da quella che svolge in un'istruzione ordinaria del tipo H Mc z, cioè di modifica dell'istruzione prima della sua esecuzione. In questa forma, le istruzioni speciali non sono modificabili, perché l'indirizzo del R.C. è usato per indicare rispettivamente in quale R.C. bisogna sommare il contenuto della cella Ø z o su quale R.C. effettuare la decisione di salto all'istruzione ø Ø z. Osserviamo incidentalmente che in questo caso non è possibile sostituire all'indirizzo z che figura nella prima di queste istruzioni l'indicazione diretta della quantità che si vuole sommare all'indirizzo (il che per certi versi sarebbe meglio) perché con questa organizzazione accade sovente, proprio per evitare gli inconvenienti dovuti alla non modificabilità delle istruzioni speciali di dover eseguire delle modifiche anche sul simbolo d'operazione, ciò che presenta degli svantaggi dal punto di vista della semplicità di programmazione. Proprio per evitare tutto questo noi abbiamo introdotto, sin dal primo progetto un secondo indirizzo di controllo per le istruzioni speciali, lasciando al primo indirizzo sempre la stes-

---

(1) Si noti che, per semplicità, usiamo eccezionalmente per i R.C. la stessa lettera maiuscola tanto per il nome quanto per l'indirizzo contrariamente a quanto stabilito nel N° 3.

sa funzione, cioè di modifica dell'istruzione in corso.

Poiché normalmente, nelle macchine a un indirizzo, la rappresentazione delle istruzioni risulta abbastanza corta, si suole raggrupparne due in un'unica cella della memoria. Ciò richiede l'introduzione di particolari istruzioni e rende alquanto più complicato il Controllo e il sistema di selezione della memoria, ma permette di risparmiare spazio nella memoria dimezzando il numero delle celle richieste per la conservazione di un dato programma.

Fu quanto facemmo anche noi per le istruzioni ordinarie nel primo progetto. Questa decisione faceva però anche sì che il ristretto numero di bit di una mezza parola lasciasse poco spazio per la rappresentazione dell'indirizzo di controllo, ciò che induceva a raccogliere i R.C. in una piccola memoria speciale. D'altro canto la decisione di includere un secondo indirizzo di controllo nelle istruzioni speciali imponeva di usare per tali istruzioni un'intera parola. Osservando allora che in un programma accade assai spesso che il numero delle istruzioni speciali è confrontabile, se non maggiore, di quello delle istruzioni ordinarie, si è visto che, tenuto conto della brevità che questa organizzazione permette di raggiungere nella formulazione dei programmi, risulta in definitiva conveniente usare un'intera parola per qualsiasi istruzione ordinaria o speciale. Ciò conduce di riflesso a molti altri vantaggi, quali : sostituire i R.C. con normali celle della memoria, aumento del numero delle istruzioni possibili da 32 a 64, diminuzione della parola da 40 a 36 bit, ciò che da solo rappresenta un risparmio del 10 % in quasi tutti gli organi della macchina, semplificazione di tutti i sistemi di connessione per la di-

stinzione fra prima e seconda metà di una parola. Su ciascuna di queste cose e su diverse altre questioni connesse, vi sarebbe molto da dire, sia dal punto di vista tecnico (discusso ampiamente nella Nota 3) che da quello logico-matematico, su cui non è possibile intrattenersi ora. Dagli esami sinora condotti su tali questioni è risultata per la rappresentazione delle istruzioni l'organizzazione che ora passiamo a descrivere.

6. Struttura delle istruzioni e uso delle C.P.

Ogni istruzione è rappresentata da una parola di 16 bit, suddivisa in 6 gruppi di 6 bit ciascuno (caratteri) che indicheremo nell'ordine con le lettere:

p x q y z' z"

p è il simbolo d'operazione; con 6 bit si possono avere $2^6=64$ combinazioni e quindi simboli d'operazione distinti.

x è l'indirizzo relativo della cella parametrica il cui contenuto va sommato all'istruzione nella fase di modifica.

q è il simbolo di modifica. Gli ultimi 5 bit indicano direttamente quante volte l'istruzione deve essere modificata prima della sua esecuzione. Il primo bit (q) viene usato in connessione con la modifica relativa al parametro di posizione.

y è l'indirizzo relativo di una cella parametrica, di cui si deve far uso nelle istruzioni speciali.

z' e z" rappresentano infine le cifre riservate all'indirizzo ordinario. Con i 12 bit a disposizione si possono indivi

duare fino a 4.096 celle di memoria, contro le 1.024 attualmente previste nel progetto. Per indicare complessivamente il numero intero individuato da tali due caratteri faremo uso semplicemente della lettera z.

Riportiamo in figura (pag. 24) lo schema a blocchi del nostro progetto, che ci servirà per discutere la "storia" di un'istruzione. La descrizione dettagliata di tale schema si trova nella Nota 3.

Poiché in questa organizzazione non sembra necessario ricorrere a modifiche del simbolo d'operazione (ove occorresse, ciò può sempre farsi nel modo ordinario attraverso i registri aritmetici) appena letta l'istruzione dalla memoria le cifre corrispondenti al primo (p) e terzo (q) carattere vanno direttamente al decodificatore (Df), e sono sostituite (nel combinatore Cb) da due gruppi di sei O ciascuno (che indicheremo con /), formando così la parola :

/ x, / y, z' z",

con cui è possibile, raggruppando i caratteri a due a due, rappresentare tre indirizzi distinti della memoria principale.

Le celle parametriche, in questo schema, sono ordinarie celle della memoria. Per ogni programma si può prefissare ad arbitrio l'indirizzo della prima cella che si vuole assumere come C.P. C'è una istruzione per questo di cui si dirà dopo. Indichiamo tale indirizzo con l . L'indirizzo assoluto di una generica cella parametrica si otterrà allora sommando l'indirizzo relativo x=X all'indirizzo assoluto della prima cella parametrica. Lo stesso dicasi per il secondo indirizzo relativo y. Può accadere, tuttavia, in alcune circo

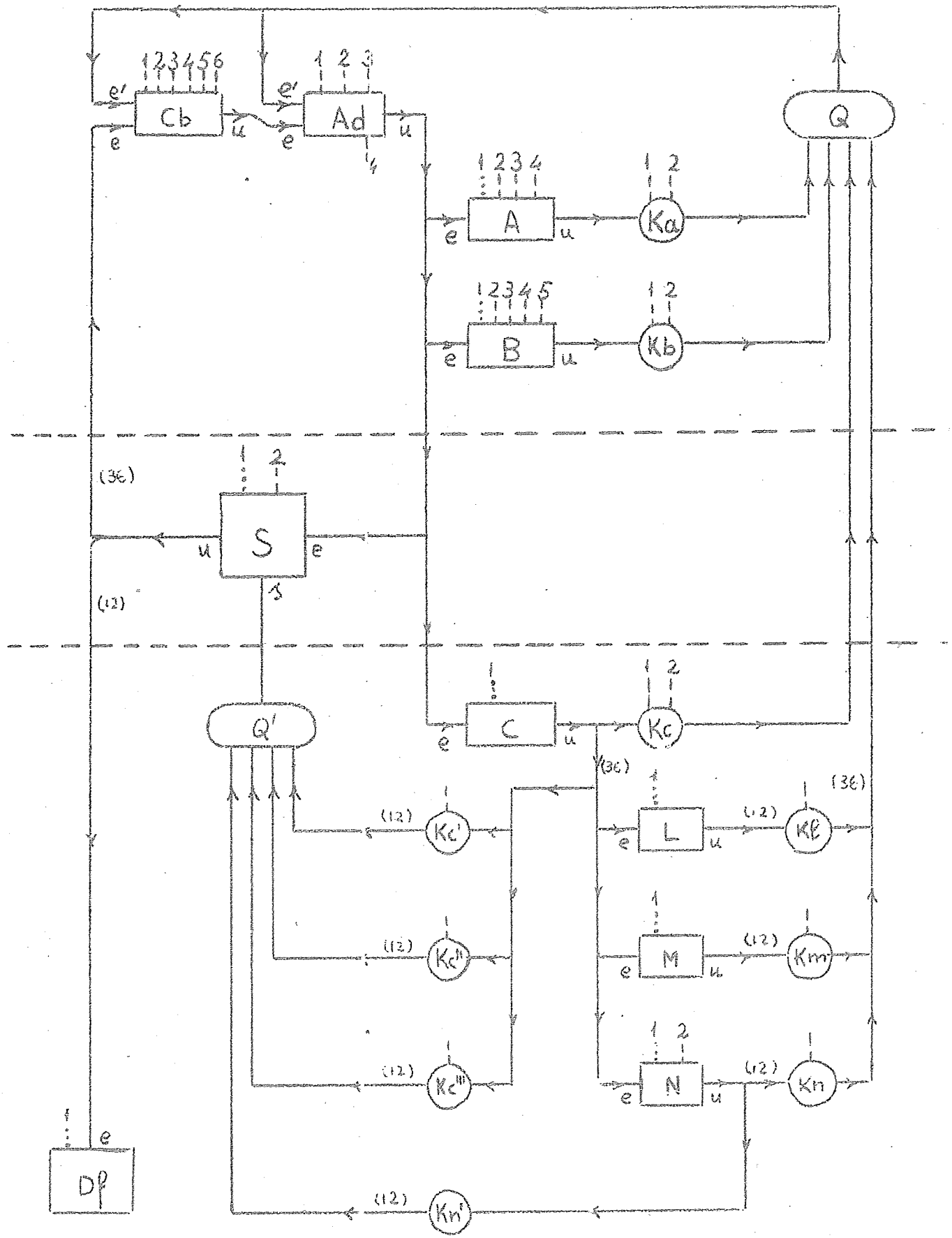


Fig. 1

stanze di doversi riferire, nelle istruzioni speciali, a un gruppo di celle parametriche diverse da quelle usate per la modifica delle istruzioni stesse. Ciò può farsi facilmente sommando al carattere y = ~~y~~ Y l'indirizzo assoluto m di tale secondo gruppo di celle parametriche. Come è descritto nella Nota 3, ciò viene direttamente eseguito attraverso lo Addizionatore Ad appena l'istruzione viene letta (fase d'entrata). Gli indirizzi indicati con l e m sono rispettivamente contenuti nella prima e seconda cellula di un opportuno registro D, di cui la terza cellula costituisce il numeratore N. Nel diagramma le tre cellule del registro D sono mostrate separate e indicate con le lettere L, M, N. L=1:D, M=2:D, N=3:D . Le istruzioni che adempiono la funzione di fissare gli indirizzi assoluti delle C.P. sono quelle indicate nel codice mnemonico con le lettere L, M. Si veda in proposito la lista completa delle istruzioni.

La coppia z' z" rappresenta l'indirizzo ordinario z. Abbiamo già notato che in gran parte delle istruzioni di un programma, si deve sommare a tale indirizzo il parametro di posizione, definito come l'indirizzo della cella che contiene la prima istruzione effettiva di quel tratto di programma. Ciò accade, per es. per tutte le istruzioni il cui indirizzo fa riferimento a celle in cui sono contenute istruzioni o costanti che fanno parte del programma stesso. Osservando, però, che anche l'indirizzo della cella, da cui è stata estratta tale istruzione, è rigidamente connesso col programma in questione, si può dare come riferimento per il paramentro di posizione, l'indirizzo dell'istruzione stessa, che di volta in volta è direttamente contenuto in N, cioè nella 3ª cellula del registro D. Il primo bit q_1 del simbolo di modifica q ser

ve appunto a questo scopo. Se $q_1 = 0$ al momento della lettura avviene solo la somma di l e m rispettivamente a $\not{l}X$ e a $\not{l}Y$; se $q_1 = 1$ oltre a tali somme viene effettuata anche la somma $z + \not{\phi} N$. Possiamo scrivere in generale che subito dopo la lettura dell'istruzione si forma sul registro C degli indirizzi la parola :

$$\not{l}X+\not{l}L, \qquad \not{l}Y+\not{l}M \qquad z+q_1 \cdot \not{l}N$$

Termina così la prima fase e comincia quella di modifica. Essa consiste nel sommare al contenuto di C, il contenuto della cella della memoria il cui indirizzo completo è contenuto nella prima cellula C_1 del registro C stesso. In formule :

$$\not{\phi}\, C\ (+)\ \not{\phi}\ \not{D}\ \not{l}\ C_1 \rightarrow C$$

Quest'operazione viene ripetuta tante volte quant'è il numero indicato dagli ultimi cinque bit di q. Se non occorre modificare un'istruzione basterà porre tale numero eguale a 0. Osserviamo qui incidentalmente che in tutte queste somme, per ragioni connesse con la rappresentazione dei numeri negativi bisognerà interrompere durante le operazioni di modifica i riporti fra le singole cellule. Il simbolo (+) indica appunto che la somma da effettuare va intesa a riporti interrotti. Ciò non costituisce alcuna difficoltà dal punto di vista tecnico (si veda la descrizione delle fasi I e II nella Nota 3).

Possiamo ora vedere facilmente, come si effettuano, a partire dalla costante 0 che figura nell'indirizzo della seconda istruzione del programma (III colonna) le 2 modifiche (q = 02) occorrenti per formare l'indirizzo giusto x+h. Indichiamo con X e H le celle parametriche che contengono rispettivamente il parametro x (più precisamente x+r) e l'indice h (o meglio h-r). L'istruzione originaria contiene come in-

dirizzo parametrico ⅊X ed è con tale cella parametrica che si effettua la prima modifica, cioè la somma di x+r nel posto dell'indirizzo ordinario. La seconda modifica deve però far riferimento alla cella parametrica di indirizzo ⅊ H. Ciò si ottiene assai facilmente ponendo :

$$\not{c}\,X = \not{\chi}\,H - \not{\chi}\,X \;,\; 0 \;,\; x + r$$
$$\not{c}\,H = 0 \;,\; 0 \;,\; h - r$$

L'istruzione originaria è :

Mc ⅊X , 02 / , 0

che viene letta come :

0,2 Mc : ⅊X + 1 , m , 0

Dopo la prima modifica sulla cella parametrica X, si ha :

0,2 Mc : ⅊H + 1 , m , x + r

Segue una seconda modifica su H, in base alla quale :

0,2 Mc : ⅊H + 1 , m , x + h

Avendo eseguito 2 modifiche (come prescritto dal valore di q) la macchina passa all'esecuzione dell'istruzione definitiva Mc, x+h. E' chiaro come questo procedimento di modifiche successive non è praticamente soggetto ad alcun limite, ciò che può tornare di grande utilità nella formulazione di sottoprogrammi generali invarianti specie se si tratta di programmi logicamente complessi. In un esempio di sottoprogramma di "riordinamento" che esporremo in una prossima nota si fa uso per alcune istruzioni anche di 4 modifiche, sebbene in media il numero di modifiche si aggira fra 1 e 2.

Per completare questa prima descrizione, conviene dire ancora una parola sulle istruzioni speciali. Per aumenta-

re di 1 il contenuto della terza cellula del C.P. H si fa uso dell'istruzione speciale.

C+ 0 , 00 ø H, 1

Essa non subisce nessuna modifica; la quantità da sommare è direttamente scritta, trattandosi di una costante indipendente dalle particolari applicazioni del programma al posto dell'ultimo indirizzo. Non è così invece per la prima delle istruzioni del blocco A_2 che effettua la "rimessa a zero" della C.P.H. Tale rimessa a zero consiste infatti nel sommare -r nella terza cellula di H (che è diventata 0 per h=r). Scrivendo direttamente tale parametro nell'istruzione anzidetta, ovviamente essa non risulta invariante. Una forma invariante si ottiene invece assoggettando tale istruzione ad una modifica rispetto a una cella parametrica che indicheremo con R e il cui contenuto sia ø R = 0, 0, -r. L'istruzione C+ ø R, 01 ø H, 0 viene letta come :

01 C+ : ø R + 1, ø H + 1, 0 e modificata in :

01 C+ : ø R + 1, ø H + m, ø 3 : ø (ø H + 1)

ove ø 3 : ø (ø H + 1) = -r

cioè sommare nella 3ª cellula della cella parametrica H, il numero -r. Si noti che in questo programma le istruzioni speciali fanno riferimento proprio alle C.P. di modifica: ciò richiede che sia m = 1. Consideriamo infine l'istruzione relativa alla decisione D_1. Tale istruzione porta l'indicazione ø H dell'indirizzo relativo della c.p.H su cui deve essere effettuato il salto. Nell'indirizzo figura la costante -5, mentre il primo bit del carattere q è 1, indicando che il salto, se deve essere effettuato, deve mandare all'istruzione contenuta nella cella di indirizzo ø N-5 ovverosia saltare 5 posti indietro.

Con questi dati è facile proseguire l'esame delle altre istruzioni del programma. Tuttavia, più che a questo scopo esse sono state date a semplice titolo indicativo, per illustrare, in un esempio concreto, anche se un po' artificialmente semplificato, di programma le caratteristiche generali di funzionamento della macchina in progetto. I vari problemi inerenti alla programmazione, sempre con riferimento a questo schema di macchina, verranno trattati in note successive.

7. Lista delle istruzioni.

Passiamo ora alla presentazione dell'intera lista delle istruzioni previste, descrivendo, secondo le notazioni già introdotte, e da un punto di vista essenzialmente logico, la III.ª fase dell'istruzione cioè quella di esecuzione, le prime due fasi che sono simili per tutte le istruzioni essendo già state descritte. Nella Nota 3 tale lista viene esaminata dal punto di vista del controllo.

Nella tabella seguente sono riportate le diverse istruzioni previste finora. Inutile dire che, anche a prescindere da una intera riorganizzazione della macchina sempre possibile, tale lista deve considerarsi provvisoria, la scelta di una lista definitiva essendo subordinata a ulteriori studi sulla programmazione.

Ogni istruzione è illustrata su 5 colonne.

Nella prima colonna figura il simbolo d'operazione codificato in base 64 mediante un gruppo di due cifre arabe da 0 a 7, assegnato convenzionalmente all'istruzione.

Nella seconda colonna figura il simbolo mnemonico dell'istruzione.

Nella terza è scritto in simboli il risultato della istruzione stessa. Si noti che qui intervengono essenzialmente i registri A, B, C, e che $\not{b}\ C_1$, $\not{b}\ C_2$, $\not{b}\ C_3$ rappresentano gli indirizzi effettivamente scritti nel registro C degli indirizzi dopo la fase di modifica. Nella descrizione verbale dell'istruzione, invece, indicheremo tali contenuti con le lettere x, y, z che figurano nella rappresentazione generale delle istruzioni (p x q y z' z").

Nella quarta colonna è indicata la preparazione dell'istruzione successiva. Essa consiste, a seconda dei casi, nell'aumentare di 1 il contenuto del numeratore N; $\not{b}N+1 \rightarrow N$, ovvero nel trascrivere in N il contenuto della cellula C_3; $\not{b}\ C_3 \rightarrow N$.

Nella quinta è data una breve descrizione verbale dell'istruzione.

Si noti che le operazioni indicate come sommare, sottrarre, moltiplicare, etc., data la natura finita dei registri aritmetici, non coincidono esattamente con le operazioni aritmetiche di egual nome. Ciò ha molta importanza dal punto di vista matematico, e verrà discusso in una prossima nota sull'aritmetica della macchina.

Si può anche rilevare la quasi completa simmetria fra i registri A e B. Questo fatto, che risulta assai comodo dal punto di vista della programmazione, è stato possibile per l'effettiva simmetria dello schema fisico dell'unità aritmetica.

Una discussione particolareggiata delle singole istruzioni, e del loro uso verrà fatta man mano che se ne presenterà l'occasione nel seguito del lavoro.

Le istruzioni 70-73 riguardano i trasferimenti (lunghi

e corti) dal tamburo alla memoria e viceversa. Questi trasferimenti vanno effettuati a gruppi di parole per volta. Una lunghezza ragionevole per i due tipi di trasferimenti potrebbe essere 8÷16 parole per quelli corti, 128÷256 per quelli lunghi. Tale questione, tuttavia, costituisce al momento solo un aspetto secondario che andrà esaminato in base a ulteriori studi sull'utilizzazione della macchina.

Per l'entrata e l'uscita abbiamo segnato solo due istruzioni; ES : leggere una parola dal nastro d'entrata (perforato o magnetico) e SU : scrivere una parola (o un carattere) sul veicolo d'uscita. Questo gruppo di istruzioni per l'entrata e l'uscita è ovviamente insufficiente : vi sono però diversi simboli di operazione liberi con cui è possibile provvedere a colmare questa lacuna. Tuttavia, la definizione esatta di tale gruppo di istruzioni non potrà essere fatta che quando saranno stati decisi gli organi di entrata e uscita di cui la macchina dovrà essere corredata.

00	S+Y	$\not{b}3:\not{D}\not{b}C_2 < \begin{matrix} < 0 \\ \geq 0 \end{matrix}$	$\not{b}N+1 \rightarrow N$ $\not{b}C_3 \rightarrow Y$	Saltare, per $\not{b}Y_3 \geq 0$, all'istruzione $\not{c}\not{D}z$
01	S-Y	$\not{b}3:\not{D}\not{b}C_2 < \begin{matrix} \geq 0 \\ < 0 \end{matrix}$	" "	Saltare, per $\not{b}Y_3 < 0$, all'istruzione $\not{c}\not{D}z$
02	C+Y	$\not{b}C_3 + \not{b}3:\not{D}\not{b}C_2 \rightarrow 3:\not{D}\not{b}C_2$	"	Sommare z a $\not{b}Y_3$
03	C-Y	$\not{b}3:\not{D}\not{b}C_2 - \not{b}C_3 \rightarrow 3:\not{D}\not{b}C_2$	"	Sottrarre z da $\not{b}Y_3$
04	CsY	$\not{b}C_3 \rightarrow 3:\not{D}\not{b}C_2$	"	Sostituire z a $\not{b}Y_3$
05	SY	$\not{c}\not{D}\not{b}C_3 \rightarrow \not{D}\not{b}C_2$	"	Trasferire $\not{c}\not{D}z$ in $\not{D}\not{b}Y$
06	L	$\not{b}C_3 \rightarrow L$ e M	"	Indirizzo assoluto celle parametriche.
07	M	$\not{b}C_3 \rightarrow M$	"	Indirizzo assoluto speciale.
10	ZoA	$\not{c}A < \begin{matrix} \neq 0 \\ = 0 \end{matrix}$	" "	Saltare, per $\not{c}A = 0$, all'istruzione $\not{c}\not{D}z$
11	ZoB	$\not{c}B < \begin{matrix} \neq 0 \\ = 0 \end{matrix}$	" "	Saltare, per $\not{c}B = 0$, all'istruzione $\not{c}\not{D}z$

12	Z+A	$\not{c}A < 0$ / ≥ 0	$\not{b}N+1 \rightarrow N$ / $\not{b}C_3 \rightarrow N$	Saltare per $\not{c}A \geq 0$ all'istruzione $\not{c}\not{b}z$
13	Z+B	$\not{c}B < 0$ / ≥ 0	" / "	Saltare per $\not{c}B \geq 0$ all'istruzione $\not{c}\not{b}z$
14	Z-A	$\not{c}A \geq 0$ / < 0	" / "	Saltare per $\not{c}A < 0$ all'istruzione $\not{c}\not{b}z$
15	Z-B	$\not{c}B \geq 0$ / < 0	" / "	Saltare per $\not{c}B < 0$ all'istruzione $\not{c}\not{b}z$
16	Ztr	$\not{c}Tr = 0$ / $= 1$	" / "	Saltare su traboccamento all'istruzione $\not{c}\not{b}z$
17	Z	———	"	Saltare incondizionatamente all'istruzione $\not{c}\not{b}z$
20	AlS	$\not{c}A \rightarrow \not{b}\not{b}C_3$	"	Trasferimento da A ad S
21	BlS	$\not{c}B \rightarrow \not{b}\not{b}C_3$	"	Trasferimento da B ad S
22	AcS	$\not{b}3:A \rightarrow 3:\not{b}\not{b}C_3$	"	Trasferimento corto da A ad S
23	BcS	$\not{b}3:B \rightarrow 3:\not{b}\not{b}C_3$	"	Trasferimento corto da B ad S

24	A+S	$\not{c}A+\not{c}\not{D}\not{b}C_3 \rightarrow \not{D}\not{b}C_3$	$\not{b}N+1 \rightarrow N$	Sommare $\not{c}A$ in $\not{D}z$
25	B+S	$\not{c}B+\not{c}\not{D}\not{b}C_3 \rightarrow \not{D}\not{b}C_3$	"	Sommare $\not{c}B$ in $\not{D}z$
26				
27				
30	A+A	$\not{c}A+\not{c}\not{D}\not{b}C_3 \rightarrow A$	$\not{b}N+1 \rightarrow N$	Sommare $\not{c}Dz$ a $\not{c}A$ in A
31	A+B	$\not{c}A+\not{c}\not{D}\not{b}C_3 \rightarrow B$	"	Sommare $\not{c}Dz$ a $\not{c}A$ in B
32	A-A	$\not{c}A-\not{c}\not{D}\not{b}C_3 \rightarrow A$	"	Sottrarre $\not{c}\not{D}z$ da $\not{c}A$ in A
33	A-B	$\not{c}A-\not{c}\not{D}\not{b}C_3 \rightarrow B$	"	Sottrarre $\not{c}\not{D}z$ da $\not{c}A$ in B
34	B+A	$\not{c}B+\not{c}\not{D}\not{b}C_3 \rightarrow A$	"	Sommare $\not{c}\not{D}z$ a $\not{c}B$ in A
35	B+B	$\not{c}B+\not{c}\not{D}\not{b}C_3 \rightarrow B$	"	Sommare $\not{c}\not{D}z$ a $\not{c}B$ in B
36	B-A	$\not{c}B-\not{c}\not{D}\not{b}C_3 \rightarrow A$	"	Sottrarre $\not{c}\not{D}z$ da $\not{c}B$ in A

37	B-B	$\not{c}B-\not{c}\not{D}\not{b}C_3 \rightarrow B$	$\not{b}N+1 \rightarrow N$	Sottrarre $\not{c}\not{D}z$ da $\not{c}B$ in B.
40	n+A	$\not{c}\not{D}\not{b}C_3 \rightarrow A$	"	Sommare cancellando $\not{c}\not{D}z$ in A
41	n+B	$\not{c}\not{D}\not{b}C_3 \rightarrow B$	"	Sommare cancellando $\not{c}\not{D}z$ in B
42	n-A	$-\not{c}\not{D}\not{b}C_3 \rightarrow A$	"	Sottrarre cancellando $\not{c}\not{D}z$ in A
43	n-B	$-\not{c}\not{D}\not{b}C_3 \rightarrow B$	"	Sottrarre cancellando $\not{c}\not{D}z$ in B
44	A+m	$\not{c}A+\lvert\not{c}\not{D}\not{b}C_3\rvert \rightarrow A$	"	Sommare modulo $\not{c}\not{D}z$ in A
45	A-m	$\not{c}A-\lvert\not{c}\not{D}\not{b}C_3\rvert \rightarrow A$	"	Sottrarre modulo $\not{c}\not{D}z$ in A
46	n+m	$\lvert\not{c}\not{D}\not{b}C_3\rvert \rightarrow A$	"	Sommare cancellando modulo in A
47	n-m	$-\lvert\not{c}\not{D}\not{b}C_3\rvert \rightarrow A$	"	Sottrarre cancellando modulo in A
50	LpA	$\not{c}A \ \& \ \not{c}\not{D}\not{b}C_3 \rightarrow A$	"	Prodotto logico $\not{c}A$ e $\not{c}\not{D}z$ in A

51	LpB	$\not{\varnothing}B \;\&\; \not{\varnothing}\not{D}\not{b}C_3 \rightarrow B$	$\not{b}N \rightarrow N$	Prodotto logico $\not{\varnothing}B$ e $\not{\varnothing}\not{D}z$ in B
52	LiA	$\not{\varnothing}A \;!\; \not{\varnothing}\not{D}\not{b}C_3 \rightarrow A$	"	Incompatibilità $\not{\varnothing}A$ e $\not{\varnothing}\not{D}z$ in A
53	LiB	$\not{\varnothing}B \;!\; \not{\varnothing}\not{D}\not{b}C_3 \rightarrow B$	"	Incompatibilità $\not{\varnothing}B$ e $\not{\varnothing}\not{D}z$ in B
54	Tc			Preparare trasferimento corto tamburo sezione Z.
55	Tl		"	Preparare trasferimento lungo tamburo sezione Z.
56	TS		"	Trasferire dal tamburo alle cella $\not{D}z$.
57	ST		"	Trasferire dalla cella $\not{D}z$ al tamburo.
60	Vd	$\not{\varnothing}A \cdot 2^{\not{b}C_3} \rightarrow A$	"	Traslazione di Acc. di z posti a destra.
61	Vs	$\not{\varnothing}A \cdot 2^{-\not{b}C_3} \rightarrow A$	"	Traslazione di Acc. di z posti a sinistra.
62	Wd	$\not{\varnothing}(A,B) \cdot 2^{\not{b}C_3} \rightarrow (A,B)$	"	Traslazione di A e B di z posti a destra.
63	Ws	$\not{\varnothing}(A,B) \cdot 2^{-\not{b}C_3} \rightarrow (A,B)$	"	Traslazione di A e B di z posti a sinistra.

64	**Ml**	**$\not{c}B \cdot \not{c}\not{D}\not{b}C_3 \rightarrow (A,B)$**	**$\not{b}N+1 \rightarrow N$**	**Moltiplicazione completa con perdita del moltiplicatore.**
65	Mc	$\not{c}B \cdot \not{c}\not{D}\not{b}C_3 \dashrightarrow A$	"	Moltiplicazione abbreviata con conservazione del moltiplicatore.
66				
67				
70	ES		"	Leggere una parola (6 caratteri) e spedirla in $\not{D}z$.
71	SU		"	Scrivere la parola contenuta in $\not{D}z$.
72				
73				
74				
75				
76				
77	F		$\not{b}C_3 \rightarrow N$	Arresto (con salto)