NOZIONI ESSENZIALI SULL'ELABORAZIONE ELETTRONICA

L'unità centrale - Le unità periferiche - I linguaggi utilizzati per la programmazione - Sistemi di numerazione - Diagrammazione a blocco e simbologia.

Un sistema di elaborazione è generalmente costituito da una macchina(l'elaboratore) dotata di un certi numero di organi preposti allo svolgimento di funzioni specializzate. Esso può essere descritto come una collezione di risorse a cui un utente può accedere secondo le modalità dettate dai linguaggi di programmazione e dai ~~sistemi operativi~~ disponibili sul sistema. Queste risorse vengono comunemente raggruppate in due classi che costituiscono il corredo HARDWARE, a cui spesso si riferisce la parola elaboratore, e SOFTWARE del sistema.

L'HARDWARE,da un punto di vista logico, è formato da componenti, dette memorie, atte a contenere programmi e dati,da componenti che realizzano la manipolazione aritmetica e logica delle informazioni e il succedersi dei vari eventi, e da mezzi, detti transduttori,capaci di convertire le informazioni da una forma fisica ad un'altra, per esempio da perforazioni su cartoncino a segnali elettrici.

Schematicamente nella struttura di un elaboratore si possano individuare:

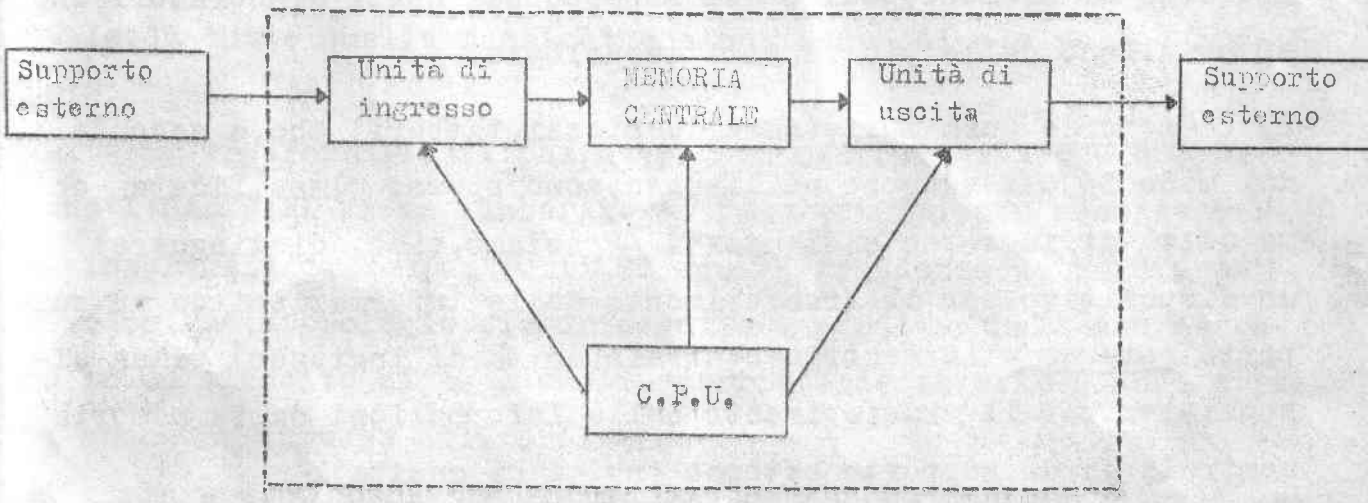


Fig. 1 - Struttura schematica di un elaboratore.

1) La memoria centrale, ovvero un supporto fisico costituito da un insieme di elementi atti a contenere informazioni codificate. Essa è il mezzo in cui, al momento di una effettiva elaborazione, vengono registrate le istruzioni da eseguire, i dati su cui queste devono operare e i risultati che producono. Tutte le informazioni in essa contenute sono codificate in forma binaria, ossia rappresentate con sequenze di cifre binarie(0 e 1) dette comunemente BITS(abbreviazione di binary digits).
Per quanto riguarda la sua organizzazione interna, la memoria centrale è composta da un certo numero di locazioni(o posizioni) che sono degli opportuni raggruppamenti di BITS(spesso 16 o 32), in ciascuno dei quali è contenuto un singolo elemento di informazione. Il contenuto di queste locazioni viene comunemente chiamato voce. Le locazioni sono numerate progressivamente e individuate in modo univoco dai numeri ad esse associati(indirizzi): ogni informazione contenuta nella memoria centrale viene quindi reperita facendo riferimento all'indirizzo della locazione in cui è registrata.
La memoria centrale è caratterizzata da un'alta velocità di accesso, ovverosia è possibile leggere le informazioni in essa contenute, o registrarci delle informazioni, in tempi che sono dell'ordine di microsecondi o dei nanosecondi negli elaboratori che si avvalgono delle più recenti tecnologie.

2) Le memorie ausiliarie(nastri, dischi, tamburi) che, a seconda dei modo in cui vengono utilizzate, sono spesso classificate come unità di ingresso e di uscita. Possiamo, cioè, distinguere: un dispositivo per il trasferimento delle informazioni da un supporto esterno alla memoria centrale(unità di ingresso) ed un dispositivo per il trasferimento delle informazioni dalla memoria centrale ad un supporto esterno (unità di uscita).

3) Una unità centrale di elaborazione(C.P.U.: Control Processing Unit) svolge un insieme di funzione intese a consentire l'esecuzione di un programma, cioè l'elaborazione di informazioni contenute nella memoria centrale e l'attivazione delle unità di ingresso e di uscita in conformità alle istruzioni fornite.

Nell'unità centrale si distinguono:

a) l'unità di controllo, fisicamente realizzata mediante circuiti elettronici, che dirige tutte le attività dell'elaboratore, determina la sequenza nella quale le istruzioni devono essere eseguite, interpreta le istruzioni, preleva dalla memoria i dati su cui queste operano e vi registra i risultati ottenuti, controlla che le istruzioni vengano eseguite dando gli opportuni comandi ai dispositivi interessati.

b) l'unità aritmetica, anch'essa costituita da componenti elettroniche, è un dispositivo capace di eseguire le quattro operazioni aritmetiche sui dati che le vengono trasmessi dalla memoria, mentre il tipo di operazione da eseguire viene determinato da un segnale trasmesso dall'unità di controllo.

La comunicazione fra l'unità centrale da una parte e le memorie ausiliarie e le unità di ingresso/uscita dall'altra viene effettuata tramite i canali. In realtà questi non effettuano una semplice trasmissione delle informazioni, ma gestiscono le unità a cui sono collegati controllandone la corretta esecuzione delle operazioni; possono anche seguire elaborazioni più o menó sofisticate sui dati trasmessi. In alcuni sistemi i canali sono veri e propri piccoli elaboratori, eventualmente collegati direttamente con la memoria centrale, e liberano completamente l'unità centrale da tutte quelle funzioni che non sono calcolo vero e proprio.

Tutte le trasformazioni, che avvengono nei circuiti che compongono l'HARDWARE di un elaboratore, sono temporificate secondo certi intervalli di tempo, tutti di uguale grandezza, da un dispositivo chiamato orologio, cioè in seguito a certi segnali che entrano in un circuito si ha l'uscita corrispondente solo quando arriva anche un impulso dell'orologio.

Si parla spesso anche di ciclo: con questo termine si intende un conveniente raggruppamento di intervalli necessari ad eseguire una certa operazione; per esempio con ciclo di memoria si intende il tempo necessario ad acquisire il contenuto di una locazione.

Per quanto riguarda il SOFTWARE di un sistema di elaborazione, questo è fornito normalmente dalla casa costruttrice del sistema ed è costituito da insiemi di programmi di tipi diversi ma tutti con l'intento di sfruttare in modo ottimale le risorse del sistema stesso.

Gli elementi che lo costituiscono possone essere suddivisi nei:

- traduttori, che convertono in linguaggio macchina i programmi scritti in linguaggi simbolici.
- programmi di utilità, di aiuto soprattutto nell'elaborazione di dati registrati su memorie ausiliarie,in quanto consentono, per esempio,di eseguire semplicemente trasferimenti di informazioni fra memorie dello stesso tipo o di tipo diverso,di accedere a determinati insieme di dati,di ordinarli secondo una sequenza prestabilita,di riunire più insiemi di dati in un unico insieme,secondo criteri prestabiliti.
- programmi applicativi,per risolvere in modo generalizzato problemi di tipo scientifico o gestionale,come,per esempio,la soluzione di problemi di programmazione lineare,problemi di trasporto, gestione delle scorte di un magazzino, elaborazione di paghe e stipendi e controllo della produzione.
- sistemi operativi, che hanno il compito di gestire in maniera "ottima" tutte le risorse "HARDWARE" e "SOFTWARE" sopraelencate, ripartendole eventualmente fra più di un utente in modo automatico.

DIAGRAMMAZIONE A BLOCCHI E SIMBOLOGIA.

L'esecuzione di un programma può essere intesa come un processo di trasformazione che elabora informazioni presenti in ingresso per produrne altre in uscita.

Da questo punto di vista un programma consiste nella definizione della seguenza di passi(algoritmo) che realizza la trasformazione delle informazioni in ingresso(i dati) nelle informazioni in uscita(i risultati).

Quando ci si propone di costruire un programma per ottenere la soluzione di un problema, è indispensabile condurre preliminarmente una attenta analisi allo scopo di definire completamente e dettagliatamente la sequenza di passi che porta al risultato.

Nella fase preliminare di analisi è conveniente cercare di suddividere il problema in unità elementari indipendenti, che si prestino ad essere affrontate separatamente; in tal modo un problema complesso può essere sotituito da un insieme di sottoproblemi più semplici e quindi di più facile soluzione. Nell'analizzare ciascun sottoproblema si può pensare di riutilizzare lo stesso procedimento fino ad arrivare ad un livello di semplicità tale da permettere la traduzione immediata di ciascuna unità elementare individuata, in istruzioni del linguaggio di programmazione usato.
Nella pratica è necessario individuare innanzitutto i possibili ingressi e precisare le uscite dell'algoritmo in esame; ciò è indispensabile per definire correttamente la trasformazione da realizzare. Successivamente si considera la possibilità di ridurre l'algoritmo in unità indipendenti.
Ciascuna delle unità individuate va a sua volta caratterizzata precisando i dati in ingresso e quelli in uscita, indi si valuta la convenienza di effettuare una ulteriore riduzione.
Come esempio di applicazione della tecnica di analisi descritta si illustra, qui di seguito, il procedimento di definizione di un semplice algoritmo per il calcolo delle soluzioni di una generica equazione di secondo grado: $AX^2 + BX + C = 0$.
L'analisi del problema va preceduta dalla sua esposizione che può essere fatta nei termini seguenti:

- esposizione del problema:

Nel caso in cui sia $A \neq 0$ le soluzioni sono due e sono fornite dalla formula $X = \frac{-B \pm \sqrt{B^2-4AC}}{2A}$; in caso contrario la soluzione è unica e vale $-\frac{C}{B}$ purchè sia $B \neq 0$; nell'ipotesi che anche B sia nullo l'equazione non ammette soluzione per $C \neq 0$, è indeterminata nel caso contrario.
Supposto $A \neq 0$, se il discriminante $D = B^2 - 4AC$ è non negativo, le radici sono reali e valgono:

$$X_1 = \frac{-B-\sqrt{D}}{2A} \qquad X_2 = \frac{-B+\sqrt{D}}{2A}$$

Se invece $D<0$ le radici sono complesse coniugate e hanno parte reale $R_x = -\frac{B}{2A}$ e coefficiente dell'immaginario $I_x = \frac{\sqrt{-D}}{2A}$

ANALISI DEL PROBLEMA.

a) individuazione degli ingressi e delle uscite.

Sulla base dell'esposizione precedente è possibile definire la trasformazione che si vuol realizzare, precisando che le informazioni in ingresso sono costituite dai coefficienti A,B,C, e che i risultati possono essere numeri(i valori delle soluzioni) oppure messaggi(equazione indeterminata o equazione incompatibile), in funzione dei valori dei coefficienti stessi(vedi fig. 2).

Fig.2 - Rappresentazione delle informazioni di ingresso e di uscita dell'algoritmo.

b) decomposizione.

L'algoritmo può essere decomposto in unità indipendenti corrispondenti ai diversi casi individuati, che conviene esaminare separatamente.

Negli schemi di Fig. 3 e 4, in cui ciascuna unità è rappresentata da un blocco rettangolare, è messo in evidenza l'unico legame esistente tra le diverse unità, costituito dal trasferimento dei dati dall'uscita di un blocco all'ingresso del blocco successivo.

ANALISI SEPERATA DI CIASCUNA UNITA'

La prima unità ha il compito di individuare le diverse alternative possibili in dipendenza dei valori dei coefficienti in esame: l'ingresso di tale unità è costituito dai coefficienti stessi,le uscite sono diverse a seconda dei casi individuati.

Nei casi I e 2 l'algoritmo ha termine e il risultato è costituito dal corrispondente messaggio.

Nel caso 3 si individua una ulteriore unità il cui ingresso è costituito dai coefficienti B e C,la cui funzione è di calcolare la soluzione e la cui uscita è il valore della soluzione calcolata.

Al caso 4 corrisponde una unità che,a partire dai valori dei coefficienti porta al calcolo delle due soluzioni. Tale unità può

essere ulteriormente suddivisa tenendo conto del fatto che il procedimento di calcolo dipende dal valore del discriminante D. L'analisi effettuata porta quindi alla rappresentazione dello algoritmo illustrata nello schema di Fig. 4.

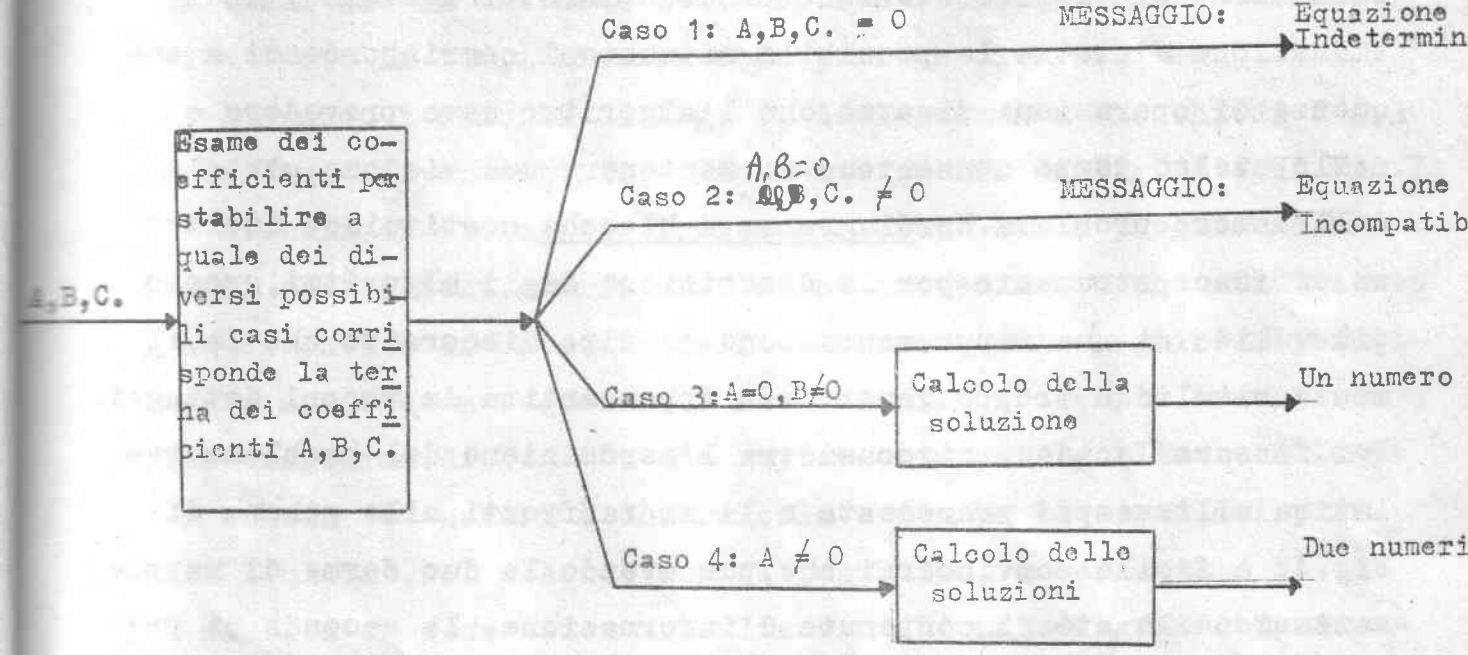


Fig. 3 - Decomposizione dell'Algoritmo in unità indipendenti.

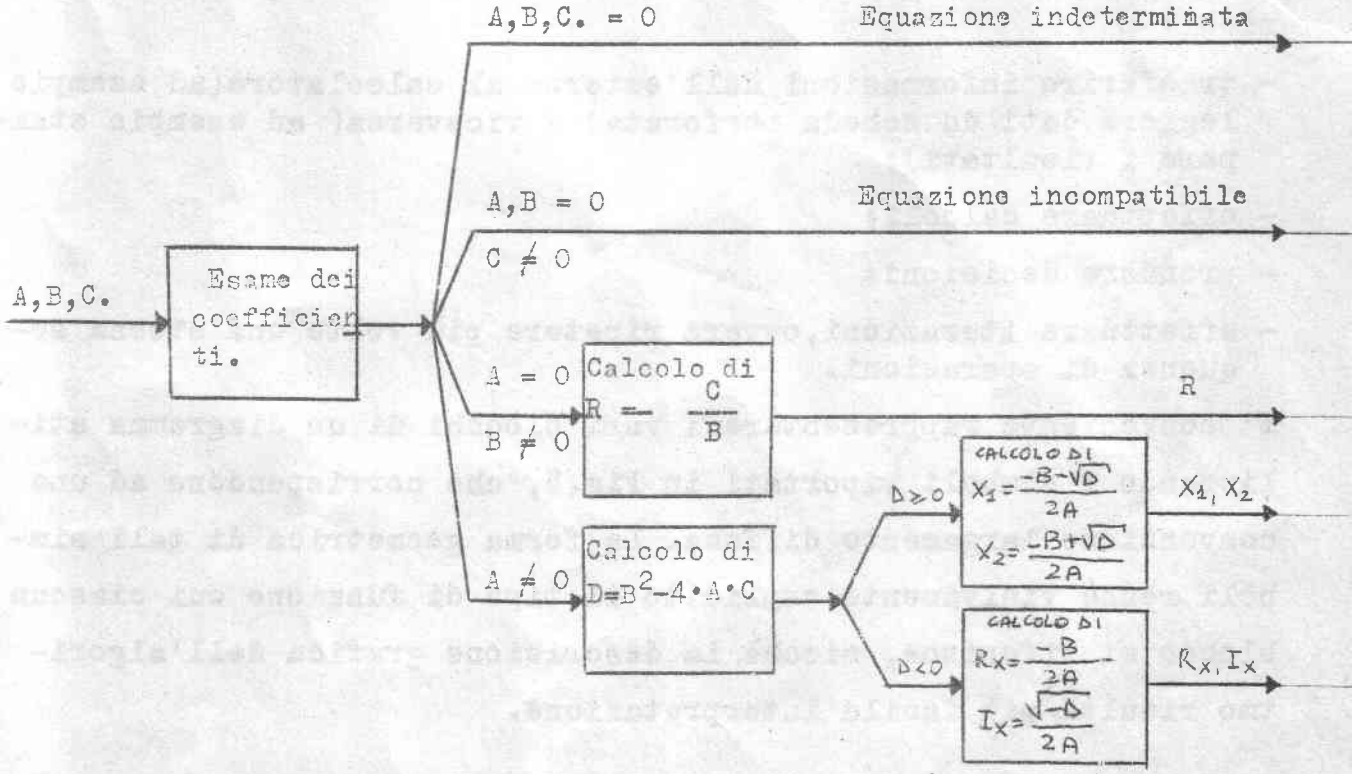


Fig.4 - Decomposizione dell'Algoritmo in unità elementari.

DIAGRAMMI A BLOCCHI

Le rappresentazioni grafiche di un algoritmo, come, ad esempio, quelle delle figure 2, 3 e 4, prendono il nome di diagrammi a blocchi; il ricorso ad esse è sempre consigliabile durante la fase di analisi del problema.

Tali forme di rappresentazione rendono infatti agevole l'individuazione di tutte le possibili situazioni corrispondenti a sequenze di operazioni diverse, che l'algoritmo deve prevedere e nello stesso tempo consentono di mantenere una visione globale dell'intero problema. Un diagramma a blocchi costituisce inoltre un efficace strumento per la descrizione degli algoritmi, spesso più valido di una rappresentazione di tipo discorsivo che facilmente risulta o troppo generica o appensantita da troppi dettagli.

Per fissare le idee, si consideri l'esposizione del problema relativa all'esempio precedente e la si raffronti allo schema di Fig.4: è facile convincersi che, pur avendo le due forme di rappresentazione lo stesso contenuto d'informazione, la seconda si presta ad una interpretazione immediata, mentre la prima richiede una lettura attenta ed un non trascurabile sforzo di concentrazione.

Un qualsiasi algoritmo, per quanto complesso, può essere decomposto in funzioni elementari corrispondenti ad una delle seguenti operazioni:

- trasferire informazioni dall'esterno al calcolatore (ad esempio leggere dati da scheda perforata) e viceversa (ad esempio stampare i risultati);
- effettuare calcoli;
- prendere decisioni;
- effettuare iterazioni, ovvero ripetere più volte una stessa sequenza di operazioni.

E' conveniente rappresentare i vari blocchi di un diagramma utilizzando i simboli riportati in Fig.5, che corrispondono ad una convenzione largamente diffusa. La forma geometrica di tali simboli rende visivamente esplicito il tipo di funzione cui ciascun blocco si riferisce, sicchè la descrizione grafica dell'algoritmo risulta di più facile interpretazione.

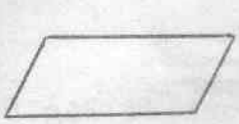
Un diagramma a blocchi costruito secondo le convenzioni di Fig.5 può essere compreso ed utilizzato da un qualsiasi programmatore per la traduzione dell'algoritmo in istruzioni di un linguaggio di programmazione.

Le fasi di analisi del problema e quella di stesura del programma risultano in tal modo nettamente distinte e possono essere effettuate in tempi diversi e da persone diverse.

Utilizzando i simboli di Fig. 5, i diagrammi a blocchi di Fig.3 e di Fig.4 diventano rispettivamente quelli di Fig. 6 e di Fig.7.

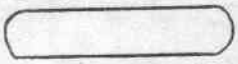
Il simbolo di iterazione, largamente usato nella programmazione degli elaboratori, corrisponde ad un procedimento che sfrutta vantaggiosamente la capacità dell'elaboratore di eseguire ripetutamente una stessa sequenza di istruzioni. Un esempio è costituito dal diagramma di Fig. 8, che rappresenta un algoritmo che legge da scheda 100 numeri e che calcola e stampa la somma del primo con l'ultimo, del secondo con il penultimo, e così via.

I simboli di connessione risultano utili nei diagrammi particolarmente complessi e consentono di evitare l'aggrovigliarsi delle linee che indicano il legame sequenziale e di continuare il diagramma stesso in altre pagine.



Trasferimento di informazioni dal calcolatore all'esterno(stampa) o viceversa(lettura).

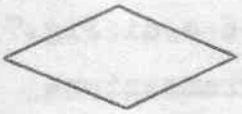
Operazioni di calcolo.



Inizio o fine un algoritmo.

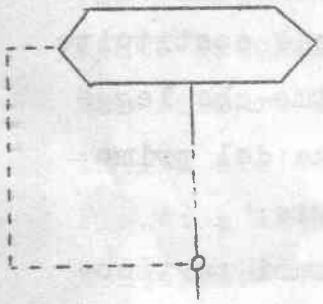
Riferimento ad un algoritmo descritto separatamente.

Connessione.



Operazione di decisione sulla base di un confronto.

Esame di sequenzialità.



Iterazione.

Fig. 5 - Simboli convenzionali per la costruzione di un diagramma a blocchi.

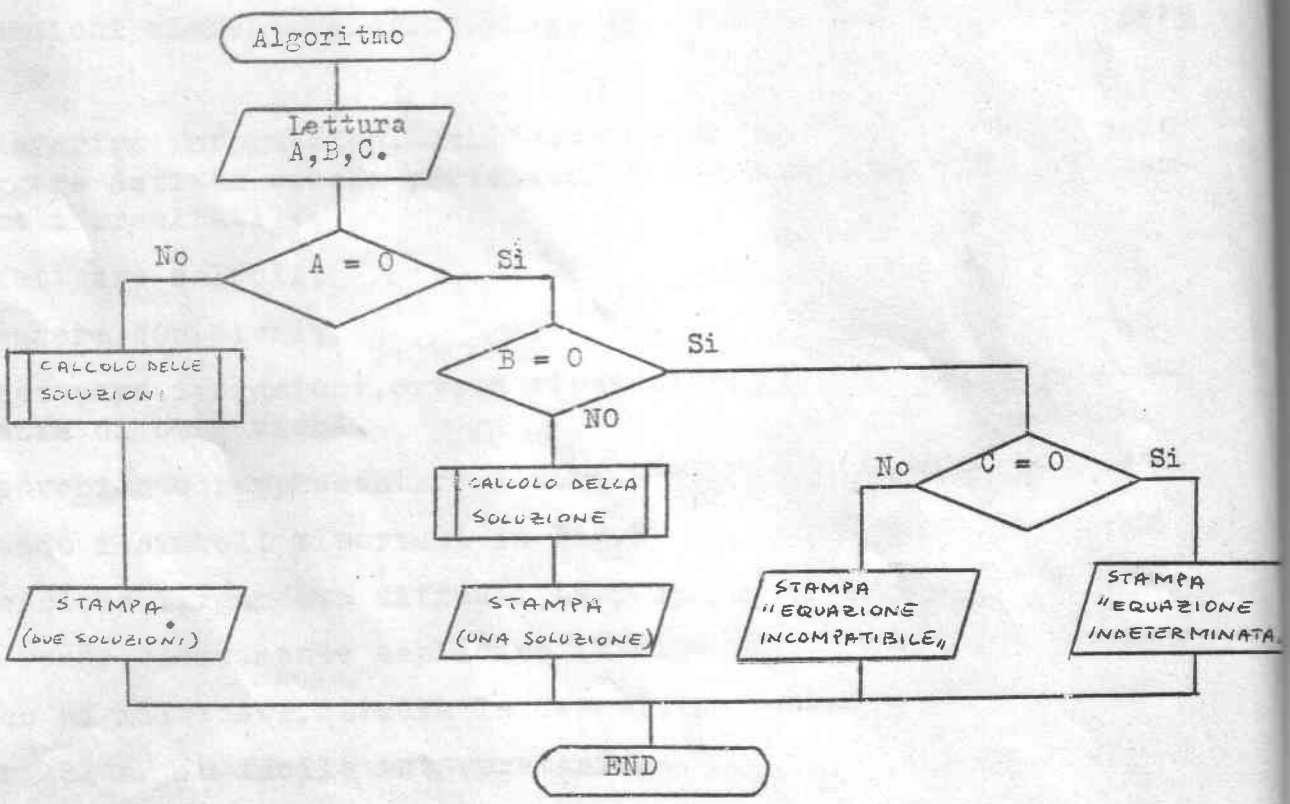


Fig. 6 Rappresentazione dell'algoritmo tramite diagramma a blocc

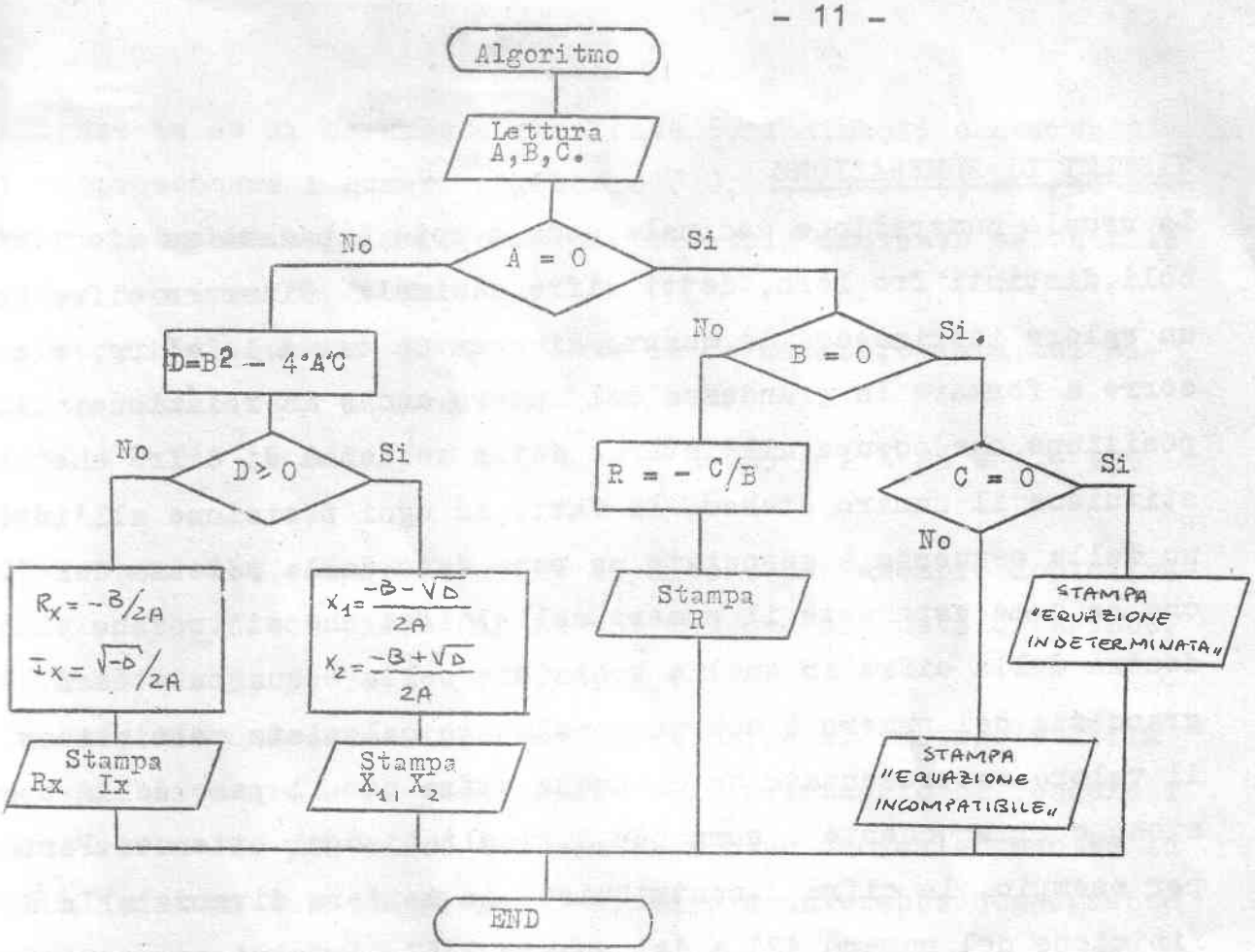


Fig. 7 - Rappresentazione dettagliata dell'Algoritmo.

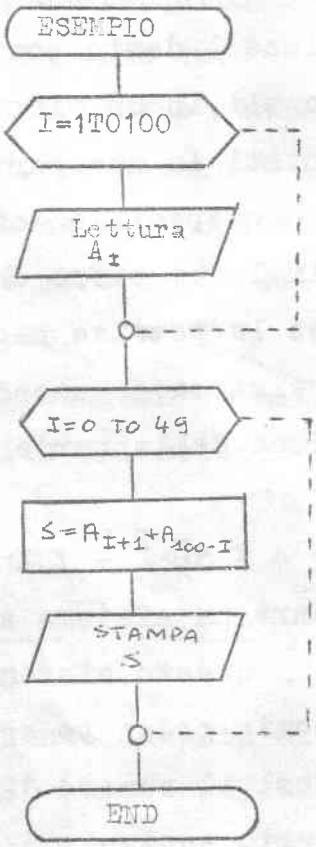


Fig. 8 Esempio di diagramma a blocchi.

SISTEMI DI NUMERAZIONE

La usuale numerazione decimale, come è noto, è basata su dieci simboli, distinti fra loro, detti cifre decimali. Ciascuna cifra ha un valore intrinseco, in quanto diversa da tutte le altre, e concorre a formare la grandezza del numero anche in relazione alla posizione che occupa all'interno della sequenza di cifre che costituisce il numero stesso. Infatti, ad ogni posizione all'interno della sequenza è associato un peso dato dalla potenza del 10 che ha come esponente il numero delle cifre che si trovano più a destra della cifra in quella posizione nella sequanza stessa. La grandezza del numero è convenzionalmente calcolata moltiplicando il valore rappresentato da ciascuna cifra per il peso della posizione corrispondente e sommando i risultati così ottenuti. Pertanto, per esempio, la cifra 1 contribuisce in maniera diversa alla definizione del numero 421 e del numero 1457. Infatti, per quanto detto sopra:

$$431 = 4 \times 10^{2} + 3 \times 10^{1} + 1 \times 10^{0}$$

e

$$1457 = 1 \times 10^{3} + 4 \times 10^{2} + 5 \times 10^{1} + 7 \times 10^{0}$$

Un tale sistema di numerazione è detto posizionale.

Analogamente i numeri razionali, in un sistema posizionale, sono convenzionalmente rappresentati in due parti, una intera e una fratta, separate da un simbolo speciale detto punto di separazione. Nella notazione decimale le cifre che rappresentano la parte fratta hanno come peso le potenze negative in ordine decrescente della radice 10 del sistema, assumendo 10^{-1} come potenza associata alla prima posizione della parte fratta del numero. Quindi, per esempio, si ha:

$3.147 = 3 \times 10^{0} + 1 \times 10^{-1} + 4 \times 10^{-2} + 7 \times 10^{-3}$

Si potrebbe, ad esempio, creare un sistema a base tre (valendosi delle cifre 0, 1, 2). Anche in questo sistema ogni numero si può esprimere mediante un polinomio ordinato secondo le potenze decrescenti della base tre. Così il numero 1201 scritto in forma abbreviata nel sistema ternario, equivale a:

$1 \times 3^{3} + 2 \times 3^{2} + 0.3^{1} + 1.3^{0}$ unità, cioè 27 + 18 + 1 = 46 unità.

Analogamente se si assumesse come base 5, i simboli occorrenti per rappresentare i numeri sarebbero (0, 1, 2, 3, 4) e ogni numero si potrebbe esprimere come un polinomio ordinato secondo le potenze decrescenti di 5.

Per esempio,il numero 3204 scritto in forma abbreviata nel sistema a base cinque equivale a:

$3 \times 5^3 + 2 \times 5^2 + 0 \times 5^1 + 4 \times 5^0 = 3 \times 125 + 2 \times 25 + 4 = 429$ unità.

La base potrebbe essere maggiore di dieci, per esempio dodici(valendosi delle dieci cifre già in uso più due di nuova creazione, cioé 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B).

E' chiaro che una base minore di dieci offre il vantaggio della necessità di usare solo poche cifre e lo svantaggio di rendere i numeri con molte più cifre dei numeri a base decimale, mentre la adozione di una base maggiore di dieci presenterebbe conseguenze perfettamente opposte a quelle dette sopra.

Come abbiamo visto, i numeri,come, in generale, ogni altra informazione, sono comunemente rappresentati internamente ad un elaboratore mediante delle sequenze di bits(cifre binarie). La numerazione, e l'aritmetica, usata,quindi, dai sistemi di calcolo, è fondamentalmente quella in base due.

Nel sistema binario i simboli che occorrono per scrivere i numeri sono soltanto due: 0 e 1. Il numero 1 si dirà unità del primo ordine e se a questa unità ne aggiungeremo un'altra avremo una unità del secondo ordine e zero unità del primo ordine e perciò potremo scrivere: 1 + 1 = 10.

Se a questo numero si aggiungerà un'altra unità si avrà il numero 11,formato da una unità del secondo ordine più una unità del primo e proseguendo:

11 +	100 +	101 +	110 +	111 +	
1 =	1 =	1 =	1 =	1 =	
100	101	110	111	1000	... ecc.

fino ad ottenere la seguente tabella di equivalenza:

numeri a base dieci		numeri a base due	numeri a base dieci		numeri a base due
1	=	1	16	=	10000
2	=	10	17	=	10001
3	=	11	18	=	10010
4	=	100	19	=	10011
5	=	101	20	=	10100
6	=	110	21	=	10101
7	=	111	22	=	10110
8	=	1000	23	=	10111
9	=	1001	24	=	11000
10	=	1010	25	=	11001
11	=	1011	26	=	11010
12	=	1100	27	=	11011
13	=	1101	28	=	11100
14	=	1110	29	=	11101
15	=	1111	30	=	11110

La difficoltà insita nella inusualità di tale rappresentazione non deve comunque preoccupare gli utilizzatori di un sistema di elaborazione in quanto è quest'ultimo a provvedere alle necessarie conversioni fra notazione binaria e decimale e viceversa, in maniera del tutto automatica.

Un altro fatto che porta a conseguenze ragguardevoli, e dovuto anch'esso alla natura stessa degli elaboratori, è che la loro aritmetica può trattare solo rappresentazioni finite di numeri. Ciò comporta, almeno in via teorica, che tutti quei risultati della matematica che derivano dall'aver supposto una precisione indefinita devono essere utilizzati con una certa cautela, se applicati per mezzo di un elaboratore.

Altre notazioni vengono comunemente usate, e che sono strettamente collegate a quella binaria, sono le ottale e le esadecimale. La loro relazione con la base due è dovuta al fatto che le loro basi sono potenze del due, 2^3 e 2^4 rispettivamente. Poichè ogni sistema di numerazione in base N ha bisogno di N. caratteri distinti, per rappresentare le cifre, la numerazione in base otto utilizza le prime otto cifre decimali da 0 a 7, mentre la numerazione in base sedici ricorre, oltre alle dieci cifre decimali, alle prime sei lettere dell'alfabeto. Avremo così la seguente corrispondenza tra

cifre esadecimali e numeri decimali:

0 I 2 3 4 5 6 7 8 9 A B C D E F

0 I 2 3 4 5 6 7 8 9 10 11 12 13 14 15

I numeri binari possono essere facilmente espressi anche in base 8 e 16, grazie al fatto che queste basi sono potenze del due. La regola per convertire un numero binario in ottale è la seguente: si raggruppano le cifre binarie in gruppi di tre partendo da destra e si valuta la grandezza di ciascuno di questi numeri binari così ottenuti; i valori ottenuti sono le cifre costituenti il numero ottale corrispondente al numero binario di partenza. Per la base 16 si procede in modo analogo raggruppando le cifre binarie quattro a quattro.

Viceversa per passare da una di queste due basi in binario è sufficiente convertire in binario ciascuna singola cifra e concatenare i risultati.

CONVERSIONE DA UNA BASE AD UN'ALTRA.

Sia un numero, supposto intero, rappresentato in una base N; vogliamo determinare la rappresentazione in base M.

Si tratta di determinare le incognite nell'equazione:

$$n = a_1 N^{p-1} + a_2 N^{p-2} + \ldots\ldots = x_1 M^{r-1} + x_2 M^{r-2} + \ldots\ldots$$

dove le x_i sono le cifre significative della rappresentazione in base M e verificano le condizioni:

$$0 \leq x_i < M \quad , \quad x_1 \neq 0$$

Allora:

$$n = x_1 M^{r-1} + x_2 M^{r-2} + \ldots\ldots x_{r-1} M + x_r =$$

$$= M(x_1 M^{r-2} + x_2 M^{r-3} + \ldots\ldots x_{r-1}) + x_r \quad \text{cioè} \quad n = M Q_1 + R_1 \quad \text{con}$$

$$Q_1 = x_1 M^{r-2} + \ldots\ldots + x_{r-1} \quad \text{e} \quad R_1 = x_r$$

cioè x_r si ottiene come resto della divisione di n per M, operazione che si intende essguita nella rappressntazione in base N nella quale è dato anche il numero M.

Dividendo Q_1 per M si ottiene un quoziente Q_2 ed un resto $R_2 = x_{r-1}$.

Così procedendo si giunge al quoziente $Q_r = 0$ e al resto $R_r = x_1$.

Nel caso del passaggio da base dieci a base due per n = 35 si ha:

././.

	Q	R
n = 35 : 2 =	17	1
17 : 2 =	8	1
8 : 2 =	4	0
4 : 2 =	2	0
2 : 2 =	1	0
1 : 2 =	0	1

Allora 35 in base due é : 100011(si deve leggere a rovescio).
Sia ora α il numero frazionario di cui vogliamo determinare la rappresentazione in base M(il numero è dato in base N).
Distinguiamo:
I (α) = parte intera di α F(α) = parte frazionaria di α
Per I (α) si procede come sopra. Per F(α) invece:

$$F(\alpha) = x_{r+1} M^{-1} + x_{r+2} M^{-2} + x_{r+3} M^{-3} + \dots\dots$$

moltiplicando per M si ha:

$$MF(\alpha) = x_{r+1} + x_{r+2} M^{-1} + x_{r+3} M^{-2} + \dots = I[MF(\alpha)] + x_{r+2} M^{-1} + x_{r+3} M^{-2} + \dots$$

cioè x_{r+1} è la parte intera del prodotto FM e quindi possiamo ottenere le cifre della parte frazionaria F ( ) moltiplicandola ripetutamente per M fino ad ottenere un numero maggiore di 1 e da questo punto in poi moltiplicando per M la parte frazionaria dei numeri ottenuti. Il procedimento ha termine solo se si arriva ad un prodotto con parte frazionaria nulla.
Sia,ad esempio, α = 25.45. Allora: I(α) = 25 F(α) = 0.45.
Allora,sempre nel caso del passaggio da base dieci a base due, si ha:

Per I(α): Q	R	Per F (α):	M.F.	I(MF)
25: 2 = 12	1	0,45X2 =	0,9	0
12: 2 = 6	0	0,9X 2 =	1.8	1
6 : 2 = 3	0	~~0,4X 2 = 0.8~~		~~0~~
3 : 2 = 1	1	0.8X 2 =	1.6	1
1 : 2 = 0	1	0.6X 2 =	1.2	1
		0.2X 2 =	0.4	0
		0.4X 2 =	0.8	0

Allora 25.45 è: 11001, $0\overline{11100}$, peiodico perchè le ultime 4 cifre continuano a ripetersi.

I LINGUAGGI UTILIZZATI PER LA PROGRAMMAZIONE.

L'insieme delle istruzioni che un elaboratore è in grado di eseguire (linguaggio macchina) è molto limitato; si tratta di operazioni elementari, per esempio trasferire una informazione da una posizione di memoria ad un'altra o di sommare il contenuto di una posizione di memoria al contenuto di un registro, la cui realizzazione è in genere ottenuta direttamente tramite opportuni circuiti.

Il numero ed il formato delle istruzioni variano al variare dell'elaboratore considerato, essendo, appunto, strettamente collegate alla realizzazione fisica degli elaboratori stessi, ma, da un punto di vista logico, possono sempre essere raggruppate in quattro classi, a seconda del tipo di funzione che esplicano.

In effetti ogni elaboratore è in grado di eseguire istruzioni aritmetiche e logiche; istruzioni di trasferimento dei dati da una posizione di memoria all'unità aritmetica e viceversa, o da una posizione di memoria ad un'altra; istruzioni dette di dontrollo che in seguito al verificarsi o meno di certe condizioni possono alterare la normale sequenza di operazioni da eseguire; istruzioni dette di ingresso/uscita che permettono lo scambio di informazioni fra l'unità di memoria e i mezzi periferici.

Con questi tipi di istruzione è possibile descrivere qualsiasi procedura; trattandosi però di operazioni strettamente semplici è necessario frazionare l'algoritmo in questione in una successione di passi molto elementari che poco a che vedere con le descrizioni logico-scientifiche dei metodi di soluzione dei problemi di cui uno è abituato.

Nella stesura di un programma di linguaggio macchina, alle difficoltà di "pensare" in termini di comportamente dell'elaboratore, si aggiunge poi la difficoltà intrinseca della scrittura delle singole istruzioni.

Le istruzioni, come del resto tutte le informazioni che può trattare un elaboratore, sono codificate in forma binaria, e la loro lunghezza, come pure il numero di voci che occupano, è strettamente dipendente sia dal tipo stesso di istruzioni, sia dal tipo di elaboratore a cui si riferiscono.

Gli elementi che concorrono alla loro formazione sono generalmente la specificazione del tipo di istruzione di cui si tratta, espressa tramite il cosidetto codice operativo, il nome di uno o due registri interessati all'operazione specificata o perchè contenenti il dato o i dati su cui operare, o perchè contenenti informazioni utili alla determinazione dell'effettivo indirizzo di memoria in cui è il dato o l'istruzione a cui ci si vuole riferire, e a cui si riferisce anche un'altra parte dell'istruzione stessa,detta campo indirizzo.
Questi elementi possono o no essere tutti presenti nella specificazione di una istruzione a seconda sia del tipo di elaboratore sia dell'istruzione in questione.
Nella scrittura di programmi in linguaggio macchina deve quindi essere prestata particolare attenzione agli indirizzi delle posizioni di memoria in cui ciascuna istruzione che compone il programma si troverà al momento dell'esecuzione dello stesso e delle posizioni in cui si troveranno i dati da elaborare,ciò per poter specificare correttamente il campo indirizzo.
Le difficoltà insite nella programmazione in linguaggio macchina hanno portato alla definizione di linguaggi in cui fosse più semplice descrivere delle procedure di calcolo, e alla realizzazione quindi di corrispondenti programmi traduttori a cui fosse delegato il compito di tradurre programmi scritti in questi linguaggi, in programmi equivalenti al linguaggio macchina, che,come abbiamo detto,è il solo che può essere interpretato ed eseguito direttamente da parte di un elaboratore.
I linguaggi di questo tipo sono chiamati simbolici e possono essere divisi in due categoriz: compilativi ed assemblativi.
Da un punto di vista molto generale un linguaggio programmativo è un insieme di convenzioni per rappresentare o comunicare algoritmi; un programma,poi, scritto in un qualsiasi linguaggio programmativo, è la descrizione di un particolare algoritmo.
Ogni linguaggio programmativo presenta due aspetti:sintattico e semantico. La sintassi riguarda la forma e la struttura delle frasi del linguaggio; la semantica il loro significato.Più precisamente:

- la sintassi di un linguaggio è un insieme di regole per formare le frasi corrette(sintatticamente) del linguaggio;
- la semantica è un insieme di regole che definiscono,invece, il significato di tali frasi(spesso mediante una interpretazione).

Da un punto di vista formale poi(astraendo cioè dal significato delle sue espressioni e dalle regole di interpretazione),un linguaggio può essere considerato come un insieme di sequenze finite di simboli appartenenti ad un insieme finito,detto alfabeto(terminale). Tali sequenze sono dette parole(o stringhe).

Se consideriamo due parole, cioè due sequenze di simboli appartenenti ad un certo alfabeto, concatenando l'una all'altra otteniamo ancora una sequenza finita di simboli, cioè un'altra parola.L'operazione di concatenazione di due parole è quindi una operazione interna all'insieme di tutte le possibili sequenze di simboli generabili a partire da un dato alfabeto.

I problemi fondamentali della teoria dei linguaggi formali sono due:

1) Il problema della definizione:consiste nel determinare un insieme di regole che caratterizzano un linguaggio, che,cioè, individuano un sottoinsieme dell'universo linguistico.Tale insieme di regole costituisce una grammatica.

2) Il problema del riconoscimento: consiste nel determinare una procedura che riconosca se una parola appartiene o meno ad un dato linguaggio.

Questi due problemi sono connessi fra di loro.Infatti una procedura di riconoscimento opera una ripartizione degli elementi(sottoinsiemi)dello universo linguistico in due classi:quella dei linguaggi accettati e quella dei linguaggi non accettati.L'inverso,invece, non è sempre vero: si possono infatti definire dei linguaggi per i quali non si può definire una procedura di riconoscimento.

Ritornando alla distinzione tra linguaggi compilativi e assemblativi, i primi sono definiti allo scopo di rendere più agevole la stesura di programmi relativi alla soluzione di problemi di una certa classe e in modo del tutto indipendente dalla struttura dei particolari elaboratori su cui verranno usati; il collegamento fra questi linguaggi e le singole macchine è costituito unicamente dai programmi traduttori, detti compilatori.

I secondi sono invece definiti in modo da rispecchiare la struttura interna dell'elaboratore per cui sono progettati.In molti di essi si ha addirittura una corrispondenza uno a uno fra le istruzioni simboliche e quelle in linguaggio macchina generate, a partire da queste,da traduttori(detti assemblatori). Essi sono praticamente insostituibili per tutti quei problemi che implicano,da parte del programmatore,un uso dettagliato e completo di tutte le possibilità offerte dalla macchina e un controllo diretto dei dispositivi interni fra cui, ad esempio, i registri.

I linguaggi assemblativi, anche se sono composti da istruzioni di tipo elementare e nonsvincolano il programmatore il programmatore dal pensare in termini della logica della macchina a cui si riferiscono, hanno il tipico vantaggio di permettere codici operativi simbolici,non numerici,e quindi spesso mnemonici, e soprattutto di associare ai dati e alle istruzioni dei nomi simbolici,dette etichette, che identificano in modo univoco le posizioni di memoria in cui verranno a trovarsi al momento della esecuzione.

I linguaggi che allo stato attuale sono di uso più diffuso sono:

- L'Assembler: è un linguaggio base orientato verso la macchina.
- Il Cobol(Common business oriented language): è un linguaggio orientato verso problemi commerciali.
- Il Fortran(Formula translation): è un linguaggio orientato verso i problemi scientifici.
- Il Pert(program evaluation and review technique): è un programma standard che utilizza la ricerca operativa.
