P6060

Assembler

Manuale generale

olivetti

GR Code 3974450 Z (1)

PREFAZIONE

Scopo della presente pubblicazione è di illustrate l'uso dell'Olivetti P6060, un minicomputer programmabile in Assembler.

Questo manuale insegna come trattare vari tipi di file e come creare, eseguire ed archiviare programmi sul sistema P6060. Sono descritti ed illustrati con esempi i comandi, le istruzioni, i programmi di servizio e i modi operativi. Il manuale insegna inoltre come sfruttare le molte caratteristiche esclusive del sistema. Una parte del manuale è dedicata ai messaggi di errore; per ognuno di questi ultimi è illutrata anche la causa dell'errore. Un'altra parte è dedicata ad un esempio di programma. La pubblicazione contiene anche informazioni sull'operabilità del sistema.

Riferimenti : Nessuno

Distribuzione : Generale (G)

Prima edizione : Aprile 1978

Seconda edizione: Agosto 1978
Sostituisce il codice 3974450 Z (0) che era preliminare.

PUBBLICAZIONE EMESSA DA:

Ing. C. Olivetti & C., S.p.A.
Direzione Marketing Centrale
Servizio Documentazione
77, Via Jervis - 10015 IVREA (Italy)

INDICE

INDICE DELLE FIGURE

INDICE DELLE TABELLE

INTRODUZIONE

L'OLIVETTI P6060 è un sistema digitale che trova ampia possibilità di impiego principalmente nel campo delle applicazioni didattiche e scientifiche. Molto versatile, questo sistema può funzionare sia come elaboratore che come calcolatrice o terminale intelligente.

Il P6060 è semplice e facile da usare e risolve due esigenze fondamentali: ridurre i tempi necessari a risolvere un problema ed automatizzare al massimo tutte le operazioni. Per ottenere questi risultati, il P6060 offre una combinazione di hardware e software tale che può dirsi unica, in grado di aumentare la produttività, agevolando e semplificando al tempo stesso il lavoro.

Il sistema è dotato di due linguaggi: BASIC e Assembler. In questo manuale viene descritto il sistema con il linguaggio Assembler.

Nel sistema sono integrate una tastiera, un display, un'unità floppy disk e di una stampante termica seriale Assembler. L'utente dispone di comandi di sistema per creare programmi, correggere eventuali errori e dare risultati immediati; dispone inoltre di una gamma di programmi di utilità per gestire librerie di dati e di testi.

I programmi scritti in Assembler possono essere introdotti direttamente da tastiera e controllati immediatamente sul display. Il display aiuta ad identificare gli errori di sintassi e permette di controllare i dati introdotti. I programmi archiviati su floppy disk possono essere facilmente richiamati e modificati; i programmi che possono usufruire di questa facilità di editing sono solo quelli in formato sorgente, sotto forma di file testo. Si possono produrre listing di programmi che possono essere utilizzati come documentazione finale.

Per riassumere, le caratteristiche fondamentali del

sistema sono:

ESPANDIBILITA' — la capacità della memoria principale può essere ampliata e alla unità base si possono collegare un numero crescente di periferiche ottenendo configurazioni sempre più ampie

FLESSIBILITA' — per ogni tipo di problemi tecnici e scientifici si può scegliere la configurazione più adeguata (memoria e periferiche)

FACILITA' DI IMPIEGO — l'utente può controllare direttamente la correttezza delle sue operazioni ed il processo di elaborazione dei dati attraverso il display e la console.

Prestazioni

Le prestazioni più notevoli del sistema sono:

- operazioni semplici da eseguire per la generazione e la esecuzione dei programmi
- la possibilità di collegare automaticamente i programmi in formato sorgente tra loro
- rappresentazione dei numeri in forma binaria
- visualizzazione e stampa di testi con diversi formati predisposti dall'utente
- possibilità di impiego di una versatile gamma di periferiche

Impiego

Il sistema P6060, versione Assembler, può essere utilizzato essenzialmente per:

- eseguire programmi
- eseguire calcoli immediati.

1. IL SISTEMA P6060

Il sistema P6060 è composto da una unità base che rappresenta la configurazione minima dalla quale si ottengono, con espansioni successive, configurazioni via via più ampie.

Unità base

La unità base (vedi figura 1-1) è composta dalle seguenti parti:

- unità centrale
- tastiera
- console
- display
- unità floppy disk (ad un trascinatore)
- segnalatore acustico
- interruttore

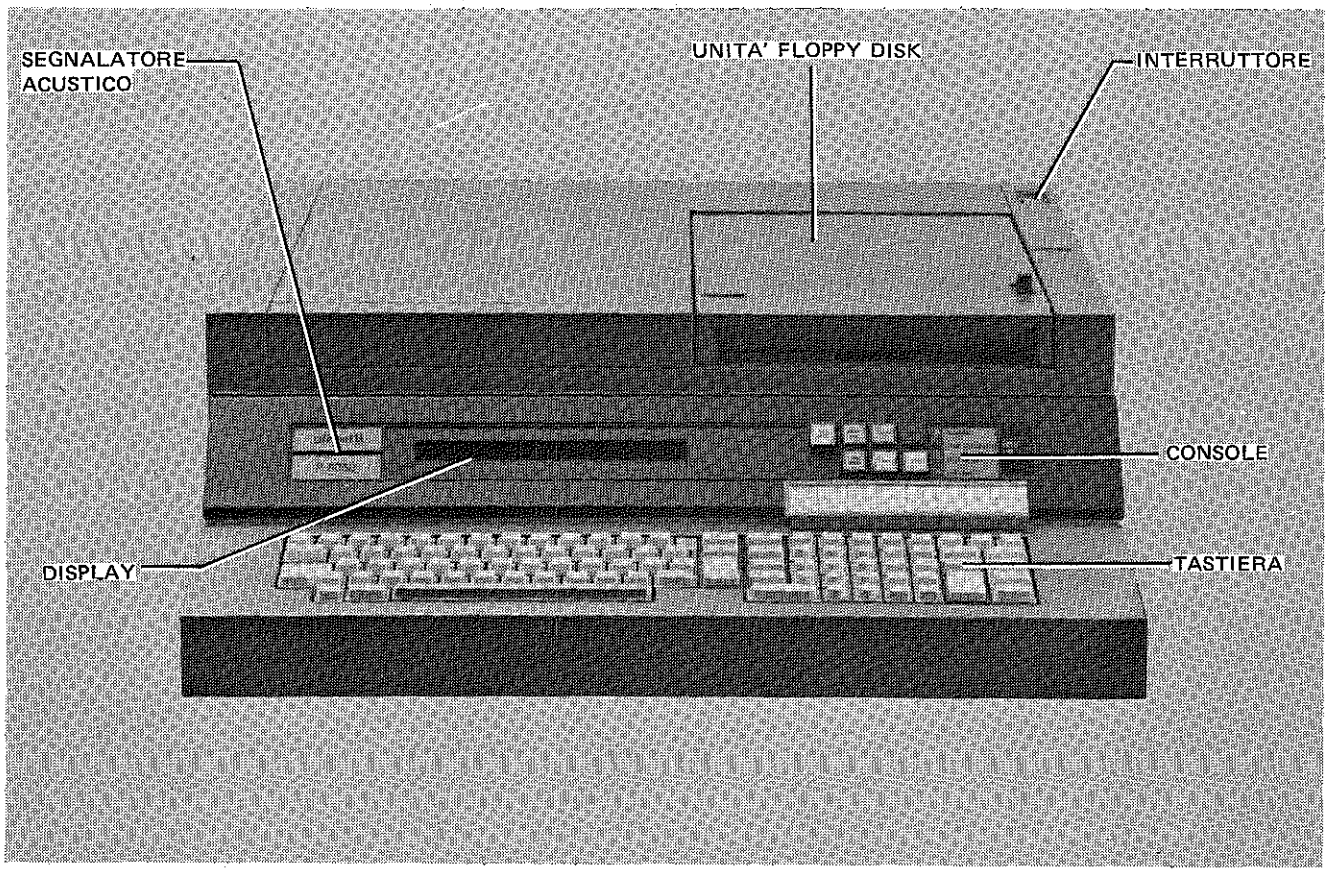


Figura 1-1 Il sistema P6060: unità base

Unità centrale

L'unità centrale è composta da:

- unità di controllo
- unità aritmetico-logica
- memoria principale (RAM)

1. L'unità di controllo coordina e controlla tutte le operazioni del sistema

2. L'unità aritmetico logica:

   - esegue le operazioni aritmetiche (addizione, sottrazione, ecc.)

   - esegue le operazioni logiche che permettono all'unità di controllo di attuare delle scelte tra diverse alternative di esecuzione.

3. La memoria principale, realizzata con circuiti integrati di tipo MOS, contiene:

   - i microprogrammi, ognuno dei quali è composto da un insieme di microistruzioni che esplodono le istruzioni del programma quando sono attivati dalle unità di controllo

   - i programmi di software di base che permettono di generare, eseguire e modificare i programmi di utente

   - i programmi utente

   - i dati da elaborare ed i risultati delle operazioni.

Possiamo quindi distinguere la memoria principale in due parti: una riservata al sistema e contenente il firmware ed il software di base (sistema operativo) ed una disponibile per l'utente. La memoria utente ha una capacità di 16384 byte di 8 bit ciascuno.

Tastiera

La tastiera è composta da 96 tasti monostabili; con essa si comunicano al sistema dati, comandi ed istruzioni attraverso un registro di transito (detto buffer) di 80 caratteri. La tastiera, vedi figura 1-2, è divisa nelle seguenti sezioni:

- sezione alfanumerica
- sezione di editing
- sezione algebrica
- sezione chiusura impostazioni
- sezione comandi
- sezione funzioni definibili

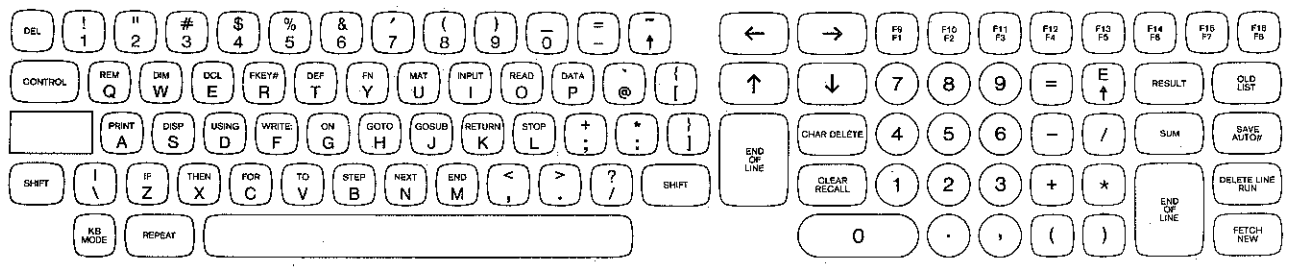


Figura 1-2 La tastiera

Sezione alfanumerica: La sezione alfanumerica è costituita dai tasti indicati in figura 1-3.

Nota: Questa sezione presenta istruzioni del linguaggio BASIC poichè è prevista per facilitare l'impiego del P6060 con il BASIC

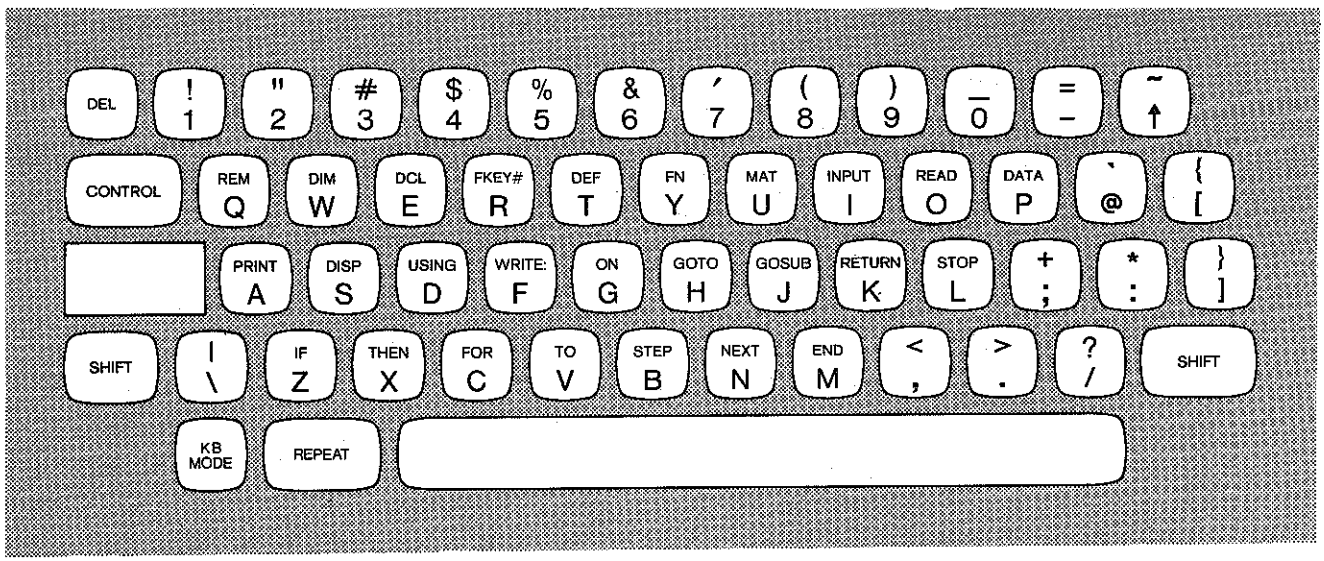


Figura 1-3 Sezione alfanumerica

Con i tasti della sezione alfanumerica si possono comporre e comunicare al sistema:

- istruzioni di programma es.
  [1][2][0] [M][V][C] [A],[B]

- istruzioni di calcolo immediato es.
  [1][2][3]*[C][O][S]([P])

- dati numerici es. [4].[5]

- stringhe di caratteri es. [V][O][L][U][M][E]

- comandi di sistema es. [0][L][D]

- comandi che richiamano ed eseguono programmi di utilità es. [E][X][E] [F][D][C][O][P][Y]

I tasti della sezione alfanumerica generano e trasmettono al sistema i 128 codici ISO che sono riportati nell'appendice C.

La tastiera alfanumerica è costituita dai seguenti tasti:

DEL — (DELETE) genera il codice ISO DEL che è operativo in terminal mode. Quando questo codice è inviato in stampa od in display ad esso corrisponde il simbolo

CONTROL — (CONTROL) è operativo in terminal mode ed è premuto insieme ad un tasto della sezione alfanumerica come indicato in appendice B

SHIFT — (SHIFT) premuto insieme ad un tasto con due chiavi introduce la chiave superiore.

KB MODE — (KEYBOARD MODE) quando è premuto permette di utilizzare la sezione alfanumerica come una macchina da scrivere: premendo [SHIFT] insieme ad un tasto si introduce la lettera maiuscola indicata nella parte inferiore del tasto. In questo caso un indicatore luminoso posto sull'estremità sinistra della tastiera è acceso. Per uscire dal keyboard mode si deve premere di nuovo [KB MODE]

LETTERE ALFABETICHE — Corrispondenti alle lettere maiuscole e minuscole dell'alfabeto inglese

CARATTERI NUMERICI — Le cifre da Ø a 9

CARATTERI ALFANUMERICI — L'unione dei due insiemi di caratteri suddetti

CARATTERI SPECIALI — Si possono generare i seguenti caratteri:

- le cifre da 0 a 9
- i segni di interpunzione: , . ; : ! ? " ' (apostrofo)
- le parentesi: (,) , [,] , { , }
- l'accento grave:
- i segni di confronto o assegnazione:
  < > o > <non uguale
  = > o = >maggiore di o uguale a
  = < o < =minore di o uguale a
  > maggiore di
  < minore di
  = uguale a o assegnazione
- i segni aritmetici: + , - , * , / , ↑
- i caratteri: # , $, % , & , _ , ~ , ~ , @ , | , \

REPEAT — (REPEAT) quando è premuto insieme ad un altro tasto viene ripetuta la generazione dello stesso codice.

BARRA SPAZIATRICE — Genera un codice che indica assenza di carattere grafico.

Sezione di editing: La sezione di editing è costituita dai 7 tasti indicati nella figura 1-4.

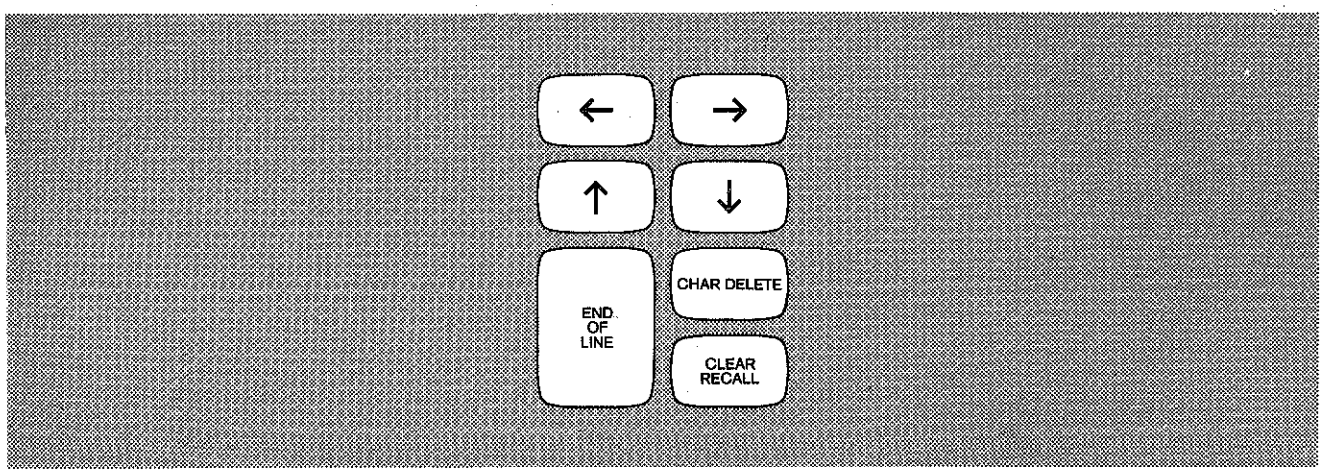
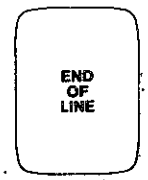
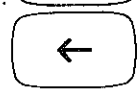


Figura 1-4 Sezione di editing



(END OF LINE) completa l'introduzione da tastiera



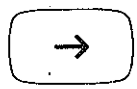
L'effetto prodotto premendo questo tasto dipende dalla posizione del pointer sul display (vedi cap. 1, par. "Il display")

- se il pointer è in una posizione compresa tra 1 e 31, il pointer si sposta di una posizione sulla sinistra

- se il pointer è in posizione 32, ma i caratteri visualizzati non sono l'inizio del testo, tutti i caratteri sul display si spostano di una posizione a destra; il pointer rimane nella posizione 32

- se il pointer è all'inizio del display, il tasto non provoca alcun effetto

Premendo [SHIFT] insieme a questo tasto, l'inizio del testo viene visualizzato a partire dalla seconda posizione ed il pointer è in prima posizione.



L'effetto prodotto premendo questo tasto dipende dalla posizione del pointer sul display

- se il pointer è in una posizione compresa tra 1 e 31 il pointer si sposta di una posizione verso destra

- se il pointer è in posizione 32, ma non alla fine del testo, tutti i caratteri sul display si spostano di una posizione a sinistra; il pointer rimane nella stessa posizione

- se il pointer è alla fine del testo, premendo il tasto non si ha alcun effetto

Premendo [SHIFT] insieme a questo tasto il pointer viene posizionato alla fine del testo (se il testo ha più di 31 caratteri vengono visualizzati gli ultimi 31 caratteri).

(CHARACTER DELETE) quando è premuto cancella il carattere che è visualizzato sul display alla sinistra del punto luminoso (pointer) e sposta di una posizione verso sinistra sia il pointer che gli eventuali caratteri alla destra di quest'ultimo.

↑ Permette di visualizzare sul display la linea di testo che precede, secondo il numero di linea, quella presente sul display prima che il suddetto tasto sia premuto. Se la linea visualizzata sul display è la prima di un file testo, premendo il suddetto tasto appare sul display l'ultima linea del file testo.

↓ Permette di visualizzare sul display la linea di testo che segue, secondo il numero di linea, quella presente sul display prima che il suddetto tasto sia premuto. Se la linea visualizzata sul display è l'ultima di un file testo, premendo il tasto suddetto appare sul display la prima linea del file testo.

CLEAR RECALL (CLEAR/RECALL)

- se premuto insieme al tasto SHIFT cancella tutti i caratteri introdotti prima di premere il tasto END OF LINE e sblocca la tastiera se era stata bloccata dopo una segnalazione di errore

- se premuto senza premere il tasto SHIFT visualizza al posto del testo attualmente sul display il testo precedente.

Sezione algebrica: La sezione algebrica è costituita dai 21 tasti evidenziati in figura 1-5.

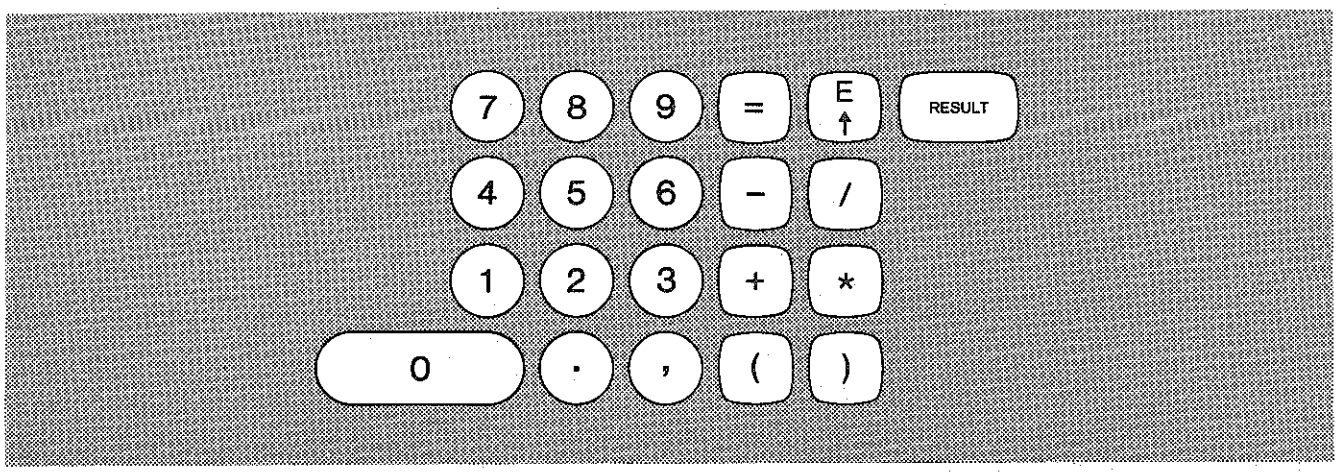


Figura 1-5 Sezione algebrica

TASTI NUMERICI Quando sono premuti generano le cifre da Ø a 9.

. Quando è premuto comunica al sistema che le cifre che verranno impostate successivamente sono da considerarsi decimali.

E ↑

Questo tasto è usato per introdurre numeri nel formato esponenziale. Se si usa E, il numero introdotto è elevato ad una potenza di 10. Se si usa ↑ il numero introdotto è elevato ad una potenza specificata dal programmatore. Per esempio premendo nell'ordine questi tasti: 1 2 SHIFT E 5 il numero fornito al sistema è 12 x 10^5. Se si premono nell'ordine questi tasti: 1 2 ↑ 5 il numero fornito al sistema è 12^5. Questo tasto è usato nello stato di sistema "calcoli immediati" (vedi capitolo 7).

TASTI ARITMETICI

Quando sono premuti richiedono al sistema di eseguire una operazione aritmetica di:

- addizione +
- sottrazione -
- moltiplicazione •
- divisione /
- elevazione a potenza ↑
- assegnazione =
- (e) sono utilizzati per dare una priorità univoca alla esecuzione delle operazioni aritmetiche contenute in una espressione.

Nota: Tutti i tasti precedentemente descritti sono riportati anche nella sezione alfanumerica.

RESULT

(RESULT) viene usato quando si comanda al sistema di eseguire calcoli immediati. Sul display viene visualizzato Φ . (Vedi esecuzione di calcoli immediati, capitolo 7).

Sezione chiusura impostazioni: La sezione "chiusura impostazioni" è composta dai due tasti di figura 1-6:

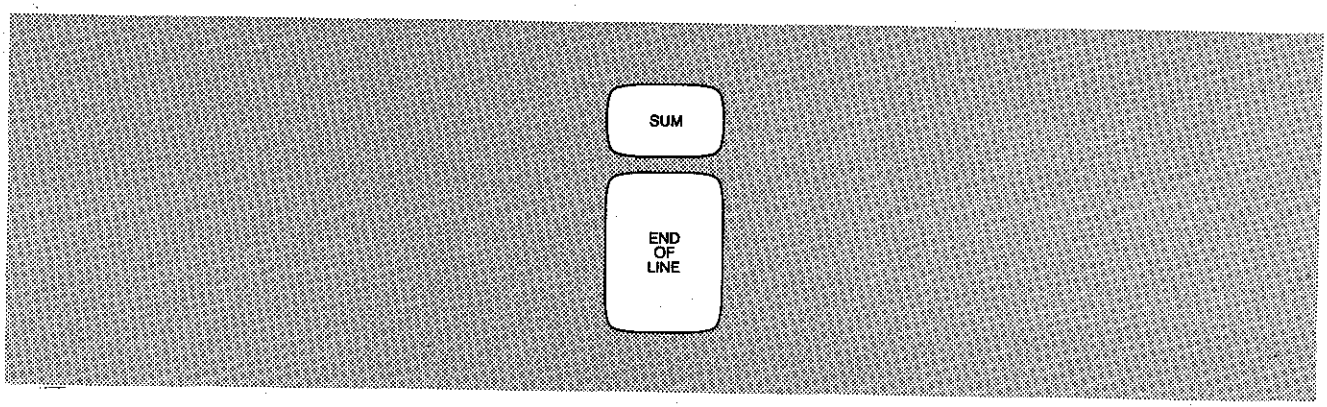


Figura 1-6 Sezione chiusura impostazioni

END OF LINE

(END OF LINE) è la duplicazione del tasto descritto nella sezione di Editing.

SUM

(SUM) è usato per totalizzare l'esecuzione di calcoli immediati (vedi capitolo 7).

Sezione comandi: La sezione comandi è costituita dai 4 tasti indicati in figura 1-7.

Nota: Questa sezione presenta comandi BASIC poichè è prevista per facilitare l'impiego del P6060 con il BASIC.

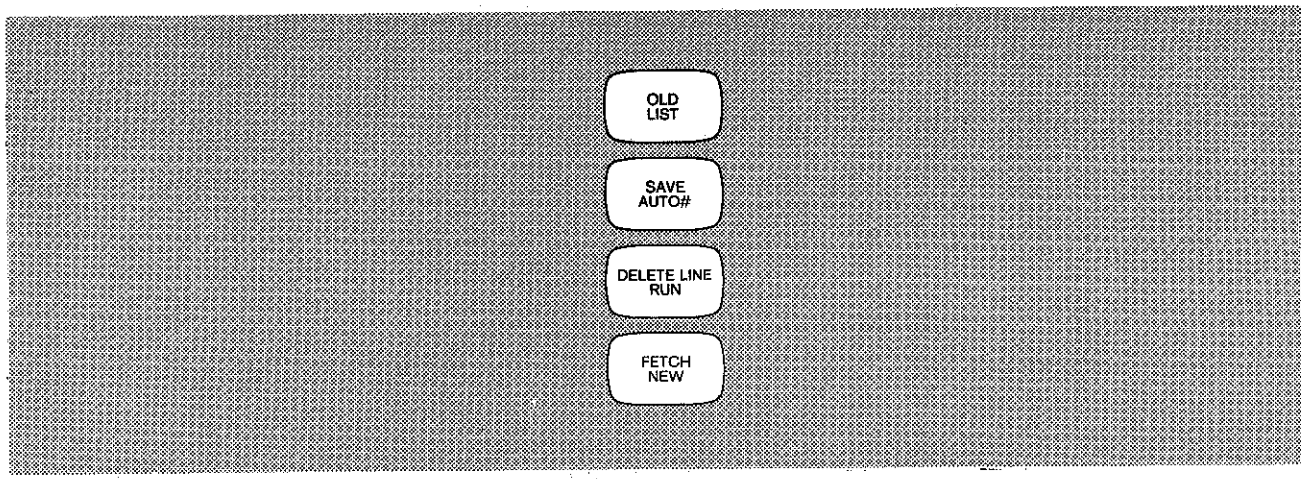


Figura 1-7 Sezione comandi

OLD LIST

(OLD LIST) con SHIFT comunica al sistema il comando OLD che carica in memoria principale un file memorizzato su floppy disk. Senza SHIFT comunica al sistema il comando LIST che stampa il contenuto del file o di parte del file presente in memoria principale.

SAVE AUTO#

(SAVE/AUTO #) con SHIFT comunica al sistema il comando SAVE che memorizza su floppy disk il file presente in memoria principale. Senza SHIFT comunica al sistema il comando AUTO # che genera la numerazione automatica dei numeri di linea durante la creazione di un file testo.

DELETE LINE RUN

(DELETE LINE/RUN) con SHIFT comunica al sistema il comando DELETE LINE che cancella in memoria principale la linea o le linee specificate nel comando.

FETCH NEW

(FETCH/NEW) con SHIFT comunica al sistema il comando FETCH che invia sul display e nel buffer di tastiera una linea di file testo presente in memoria principale.

Sezione funzioni definibili: E' composta dagli 8 tasti di figura 1-8:

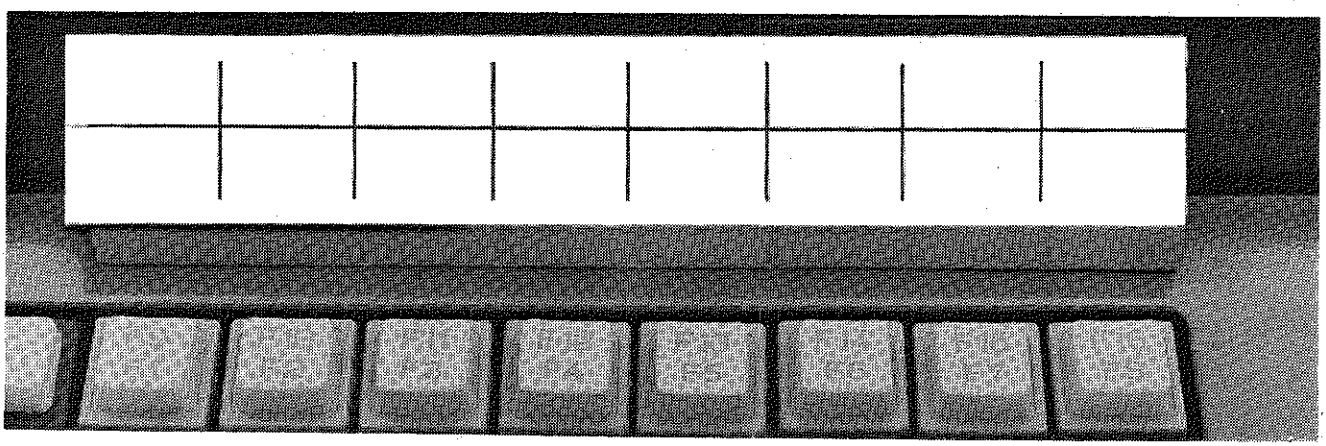


Figura 1-8 Sezione funzioni definibili

Si possono associare alle 16 funzioni F1 + F16 indicate in figura delle sequenze di caratteri prestabilite che vengono comunicate al sistema ogni volta che si preme il tasto corrispondente. Ad ogni tasto sono associate due funzioni e quella indicata nella parte superiore è abilitata con il tasto [SHIFT] (vedi capitolo 9 per l'impiego dei tasti funzione).

La console

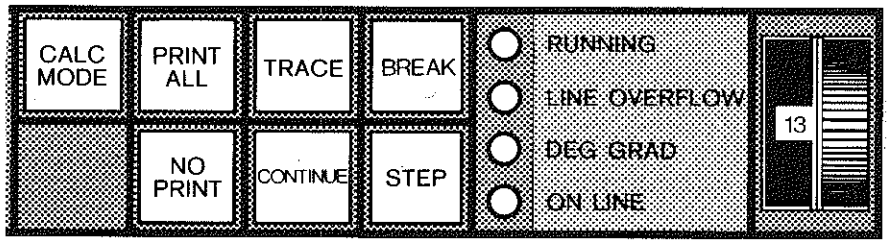


Figura 1-9 La console

La console è costituita da 7 tasti, 4 indicatori luminosi ed un dispositivo d'impostazione dei decimali come si vede in figura 1-9. I 7 tasti hanno incorpo-

rata una lampadina che si accende quando la funzione associata è attiva.

(CALCULATOR MODE) premendo il tasto si pone il sistema nello stato di esecuzione di calcoli immediati. Quando il sistema è nello stato di esecuzione di calcoli immediati la luce incorporata è accesa.

(PRINT ALL) premendo il tasto tutti i testi che appaiono sul display sono stampati. Quando la funzione suddetta è attiva la luce incorporata è accesa. Per disattivare la funzione PRINT ALL basta ripremere il tasto.

(NO PRINT) premendo il tasto vengono inibite tutte le operazioni di stampa riferite alla stampante integrata. Quando la suddetta funzione è attiva la luce incorporata è accesa. Per disattivare la funzione NO PRINT basta premere di nuovo il tasto.

(BREAK) premendo il tasto si termina l'esecuzione di un programma o del comando LIST o del comando CATALOG. La luce rimane accesa finchè la funzione BREAK è attiva. I seguenti 3 tasti sono utilizzati in debugging.

(TRACE) premendo il tasto viene stampato l'indirizzo e il codice oggetto delle istruzioni eseguibili di un programma durante la sua esecuzione. Gli indirizzi e il codice oggetto sono stampati secondo l'ordine di esecuzione. Quando la suddetta funzione è attiva la luce incorporata è accesa. Per disattivare la funzione TRACE si deve ripremere il tasto.

(STEP) premendo il tasto si interrompe l'esecuzione di un programma, in presenza dell'opzione DEB (vedi Comando OPTIONS capitolo 3). Ogni volta che il tasto è premuto successivamente il sistema esegue una istruzione. Quando la funzione suddetta è attiva la luce incorporata è accesa.

(CONTINUE) premendo il tasto viene ripresa l'esecuzione del programma interrotto. Quando la funzione suddetta è attiva la luce incorporata è accesa.

Se lampeggia indica che il sistema sta eseguendo una o più operazioni, se fissa indica che il sistema è in attesa di una introduzione da tastiera (i tasti END OF LINE e SUM sono abilitati).

Se accesa indica che sono già stati introdotti 80 caratteri nel buffer di tastiera.

Si accende quando si introduce il comando SDEG o SGRAD (esecuzione di calcoli immediati); indica che i valori degli angoli sono misurati in gradi sessagesimali o centesimali.

Se accesa indica che la tastiera funziona come periferica esterna (vedi modulo di sistema descritto nel capitolo 6).

(INDICATORE DEI DECIMALI) permette di predisporre il formato per la stampa e la visualizzazione dei numeri quando si eseguono calcoli immediati: (vedi il capitolo 7).

Il display

Il display permette la visualizzazione di 32 caratteri compresi nel set dell'appendice C.

Sul display appaiono:

- linee generate da tastiera
- messaggi da programma
- messaggi da sistema
- segnalazioni di errore
- dati introdotti da tastiera
- caratteri introdotti con i tasti funzione

Quandò si introduce da tastiera una linea, sul display compare anche un punto luminoso detto pointer che indica in quale posizione della linea si può introdurre un nuovo carattere. Quando il display è predisposto a visualizzare i caratteri introdotti da tastiera, il pointer è nella prima posizione di sinistra e si sposta di una posizione sulla destra, man mano che si introducono i caratteri; se si introducono più di 31 caratteri sul display sono visibili gli ultimi 31 caratteri introdotti: premendo [←] il pointer si sposta di una posizione a sinistra e sul display appaiono gli ultimi 32 caratteri introdotti.

Unità floppy disk

L'unità floppy disk è una unità con testina di lettura/scrittura mobile e dischi intercambiabili contenuta nel P6060 come si vede in fig. 1-10; essa comprende un trascinatore per inserire in esso un floppy disk.

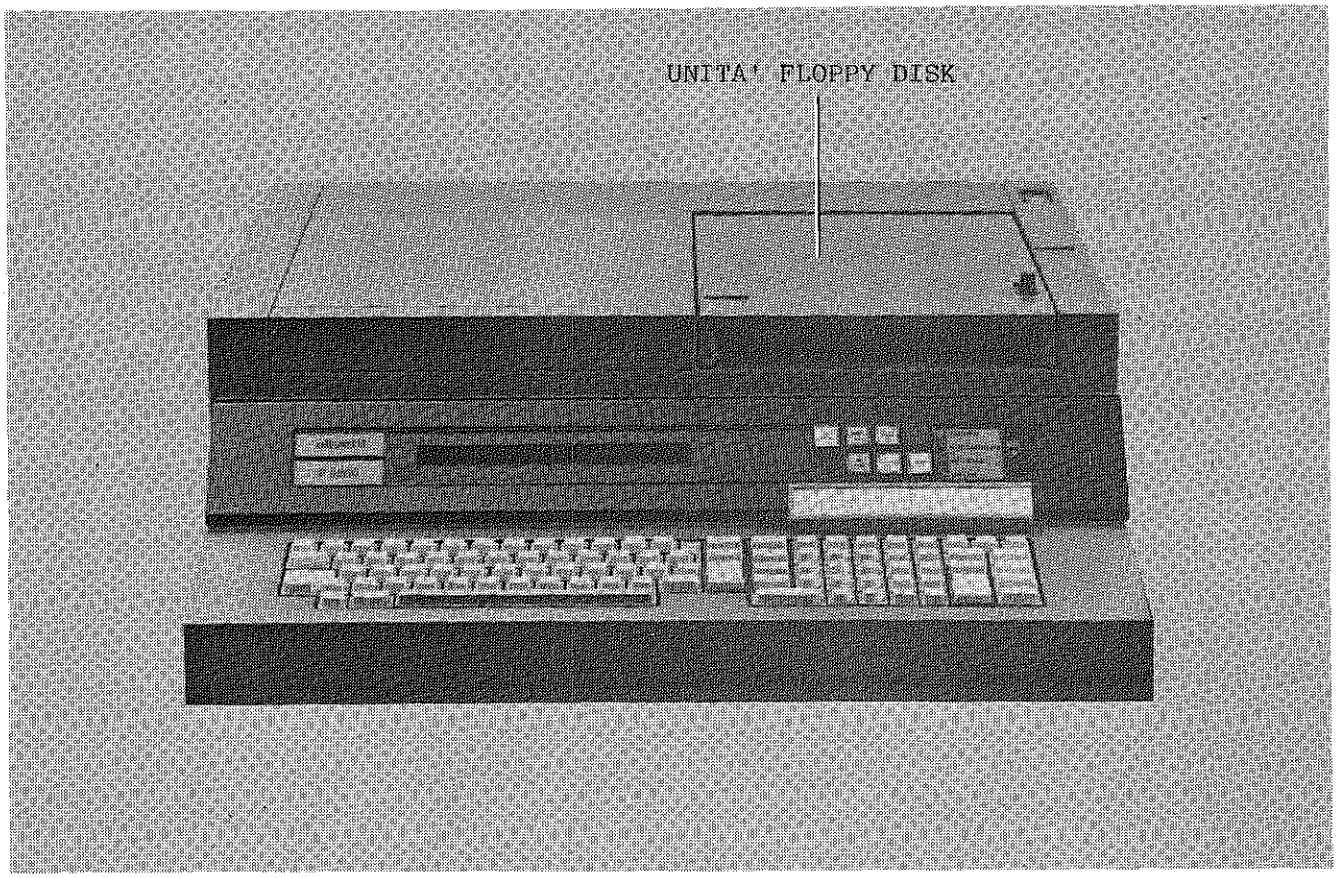


Figura 1-10 Unità floppy disk

Il floppy disk è un supporto di mylar, ricoperto di materiale magnetico ed inserito in una busta di carta dentro la quale esso è libero di ruotare. Il floppy disk è una memoria di massa ad accesso casuale che, ruotando ad una certa velocità, permette di leggere o registrare su di esso delle informazioni mediante una testina magnetica. Il floppy disk è inizializzato prima di essere spedito all'utente. Le informazioni sono memorizzate su sezioni circolari del disco (tracce). Ogni disco ha 77 tracce di cui 4 (tracce Ø, 74, 75 e 76) sono utilizzate in modo standard da qualunque sistema (vedi fig. 1-11). Ogni traccia è divisa in 26 settori di 128 byte ciascuno. Il floppy disk presente nella configurazione base contiene il sistema operativo ed eventualmente i programmi costituenti il software applicativo della libreria Olivetti. Nella parte rimanente l'utente può memorizzare dei file che vedre-

mo in seguito. L'utente può conoscere lo spazio disponibile per la registrazione di informazioni sul disco utilizzando il comando di sistema SPACE (vedi capitolo 3).

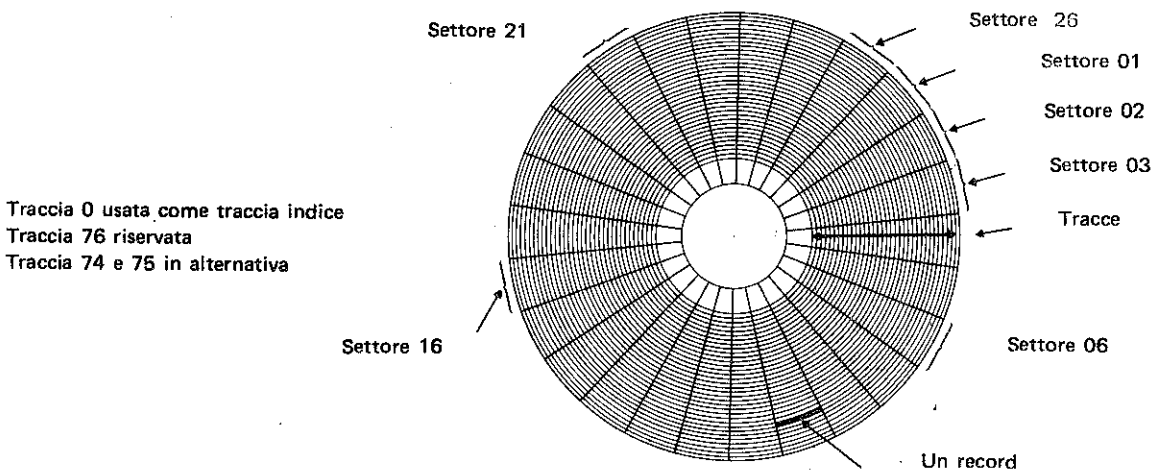


Figura 1-11 Divisione del disco in tracce e settori

Segnalatore acustico

Come indicato nella figura 1-1 il sistema è fornito di un segnalatore acustico che emette un suono ogni qualvolta l'operatore digita una linea non accettata dal sistema: ad esempio se si digita una linea mentre la luce di console RUNNING lampeggia, oppure si introducono più di 80 caratteri prima di premere END OF LINE.

Interruttore

L'interruttore indicato nella figura 1-1, se è premuto dalla parte ON, accende il sistema mentre, se è premuto dalla parte OFF, spegne il sistema.

Configurazioni estese

Dalla configurazione minima di sistema descritta nel paragrafo precedente si possono ottenere configurazioni sempre più estese con successivi ampliamenti.

Di seguito vengono descritti i moduli e le unità periferiche che possono essere collegate all'unità base.

Estensioni dell'unità base

1. Ampliamenti della memoria utente da un minimo di 16K byte (K=1024 byte) ad un massimo di 48K byte con le configurazioni seguenti: 16, 24, 32, 40, 48K byte.

2. Inserimento della stampante integrata (vedi figura 1-12)

Figura 1-12 Sistema P6060 con la stampante integrata

La stampante integrata è di tipo termografico con caratteri di stampa (vedi il set completo in appendice C) composti su una matrice di 5 per 7 punti. Su ogni riga di stampa si possono stampare fino ad un massimo di 80 caratteri con una velocità di stampa di 80 caratteri/sec.

3. Impiego di un secondo floppy disk (detto floppy disk utente) come indicato in figura 1-13. La capacità utilizzata dall'utente è di 240.590 byte; infatti oltre alle 4 tracce di impiego standard (vedi "Unità floppy disk") il sistema utilizza alcuni settori per una gestione automatica delle librerie. Per conoscere lo spazio disponibile per la registrazione di informazioni sul disco si deve utilizzare il comando SPACE (vedi capitolo 3).

Figura 1-13 Unità floppy disk con 2 floppy disk

4. Collegamento di unità periferiche esterne gestite da sistema (vedi comando CONFIGURE).

2. COME USARE IL SISTEMA

In questo capitolo sono introdotte le nozioni operative fondamentali che permettono l'impiego delle unità integrate del sistema P6060 e ne sono definiti i modi di funzionamento elencando le operazioni che l'utente può richiedere al minicomputer nelle diverse circostanze.

Accensione

1. Premere l'interruttore OFF/ON dalla parte ON (vedi figura 1-1).

2. Immediatamente dopo aver acceso il sistema tutte le luci di console si accendono ed il segnalatore acustico emette un suono. Se qualche luce non si accende la relativa lampadina non funziona. Se non si sente alcun suono il segnalatore acustico non funziona.

3. Dopo qualche secondo le luci di console possono assumere diverse configurazioni. Le configurazioni assunte hanno i seguenti significati:

LUCI DI CONSOLE ACCESE	SIGNIFICATO
BREAK, RUNNING, NO PRINT, ON LINE	Manca il floppy disk nell'unità
TRACE, RUNNING, NO PRINT, ON LINE	Floppy disk utente presente invece di floppy disk sistema
NO PRINT, ON LINE, DEG GRAD	Floppy disk sistema non corretto
BREAK, RUNNING, NO PRINT, DEG GRAD	Floppy disk sistema non corretto
NO PRINT, RUNNING, ON LINE	Sportello unità floppy disk aperto

Nota: Per configurazioni diverse da quelle specificate sopra si contatti il più vicino servizio tecnico di assistenza.

4. Se, per qualsiasi motivo, si è spento il sistema, è bene aspettare 5 secondi prima di riaccenderlo.

Spegnimento

1. Premere il tasto di console BREAK prima di spegnere il sistema se è in corso di esecuzione un programma che elabora file su floppy disk e quindi attendere che la luce incorporata nel tasto sia spenta.

2. Premere l'interruttore OFF/ON dalla parte OFF.

Come iniziare

Prima di iniziare ad usare il sistema l'utente può trovarsi nella condizione di dover eseguire delle operazioni di servizio quali: cambio del rullo di carta nella stampante integrata, inserimento del floppy disk nei relativi trascinatori.

Cambio del rullo di carta

Per estrarre il rullo di carta presente sulla stampante integrata e sostituirlo con uno nuovo si osservino le figure 2-1 e 2-2 e si eseguano le seguenti istruzioni.

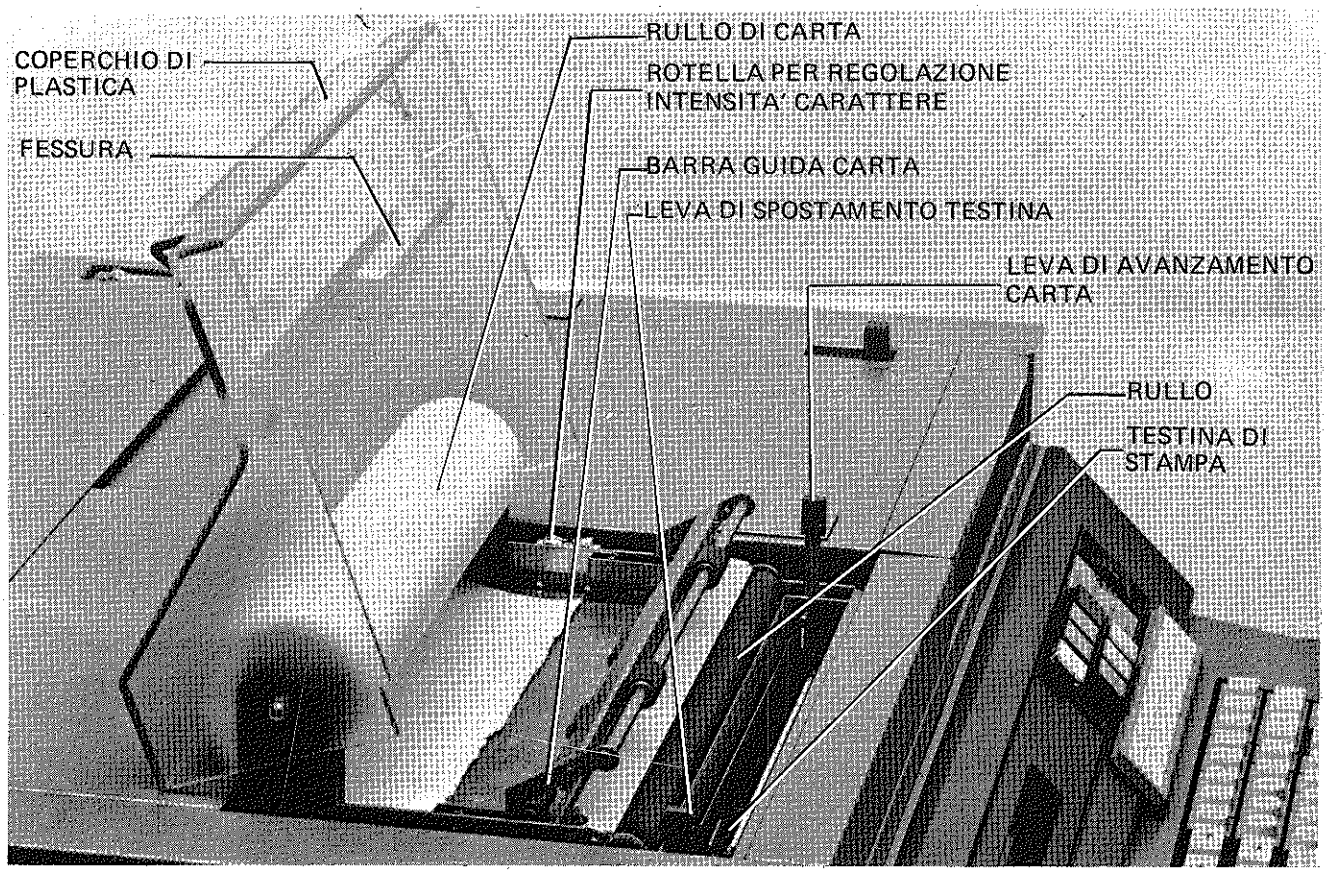


Figura 2-1 Parti componenti la stampante integrata

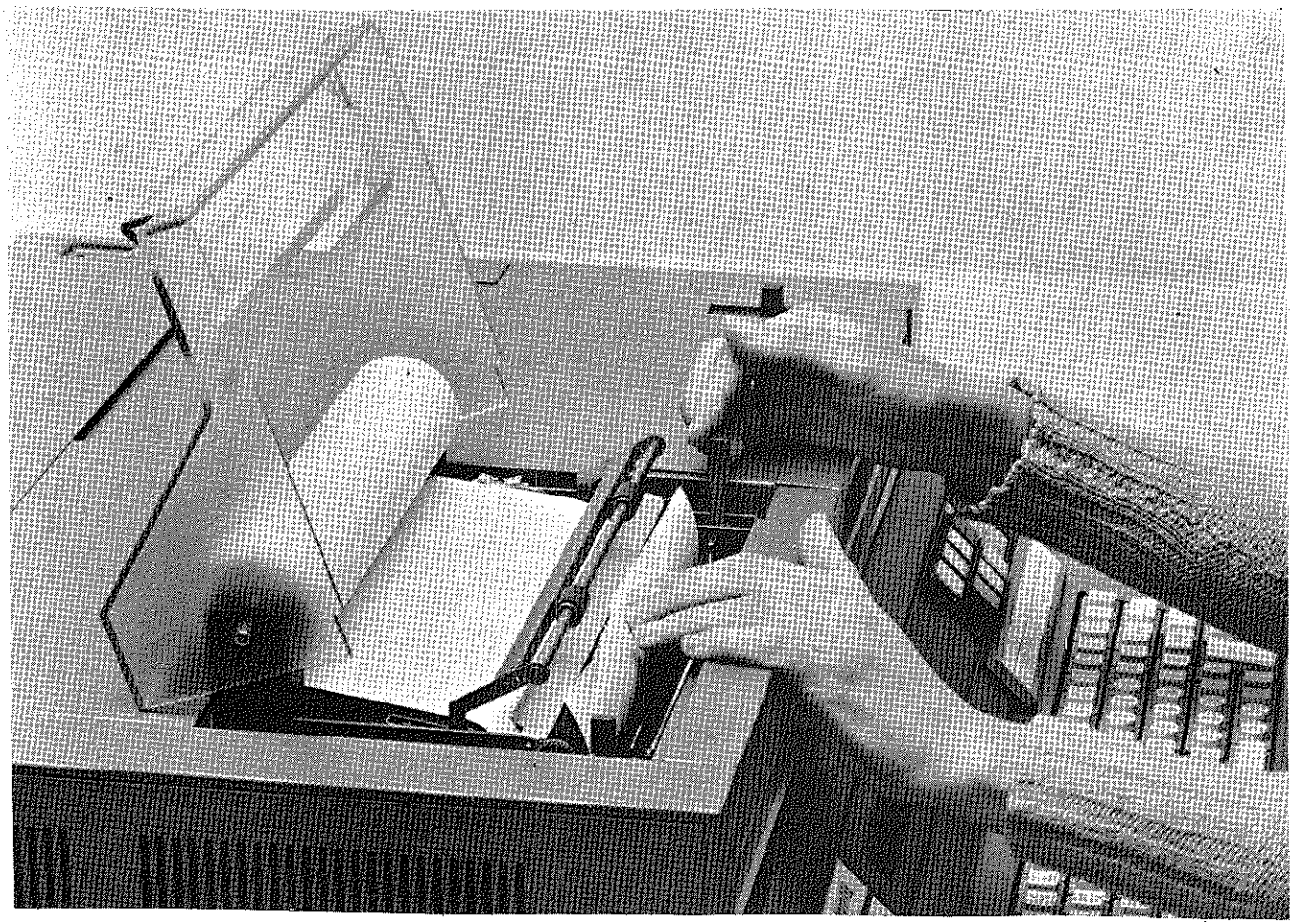


Figura 2-2 Avanzamento della carta nella stampante integrata

1. Sollevare il coperchio di plastica.

2. Allontanare la testina di stampa dalla carta tirando leggermente indietro la leva relativa.

3. Estrarre il rullo di carta dal suo alloggiamento ed estrarre dal rullo di carta il cilindro di metallo.

4. Introdurre il cilindro di metallo nel nuovo rullo di carta e poggiare il rullo nell'alloggiamento relativo della unità di stampa.

5. Far scorrere a mano la carta sotto il rullo di stampa e quindi verso l'alto mediante la leva di avanzamento (figura 2-2).

6. Quando la carta è avanzata di qualche centimetro abbassare il fermacarta.

7. Avvicinare la testina di stampa alla carta tirando

leggermente in avanti la leva relativa.

8. Scegliere l'intensità di stampa desiderata ruotando la relativa rotella. L'intensità di stampa varia da 0 a 9: 0 produce il carattere più scuro e 9 il carattere più chiaro.

9. Abbassare il coperchio di plastica inserendo la carta nella feritoia relativa; premere in avanti la leva di avanzamento carta per controllare che lo scorrimento della carta attraverso la feritoia sia regolare.

Inserimento dei floppy disk

La procedura da seguire dipende dal tipo di unità floppy disk disponibile.

Se si ha una unità ad un solo trascinatore:

1. Accendere il sistema.

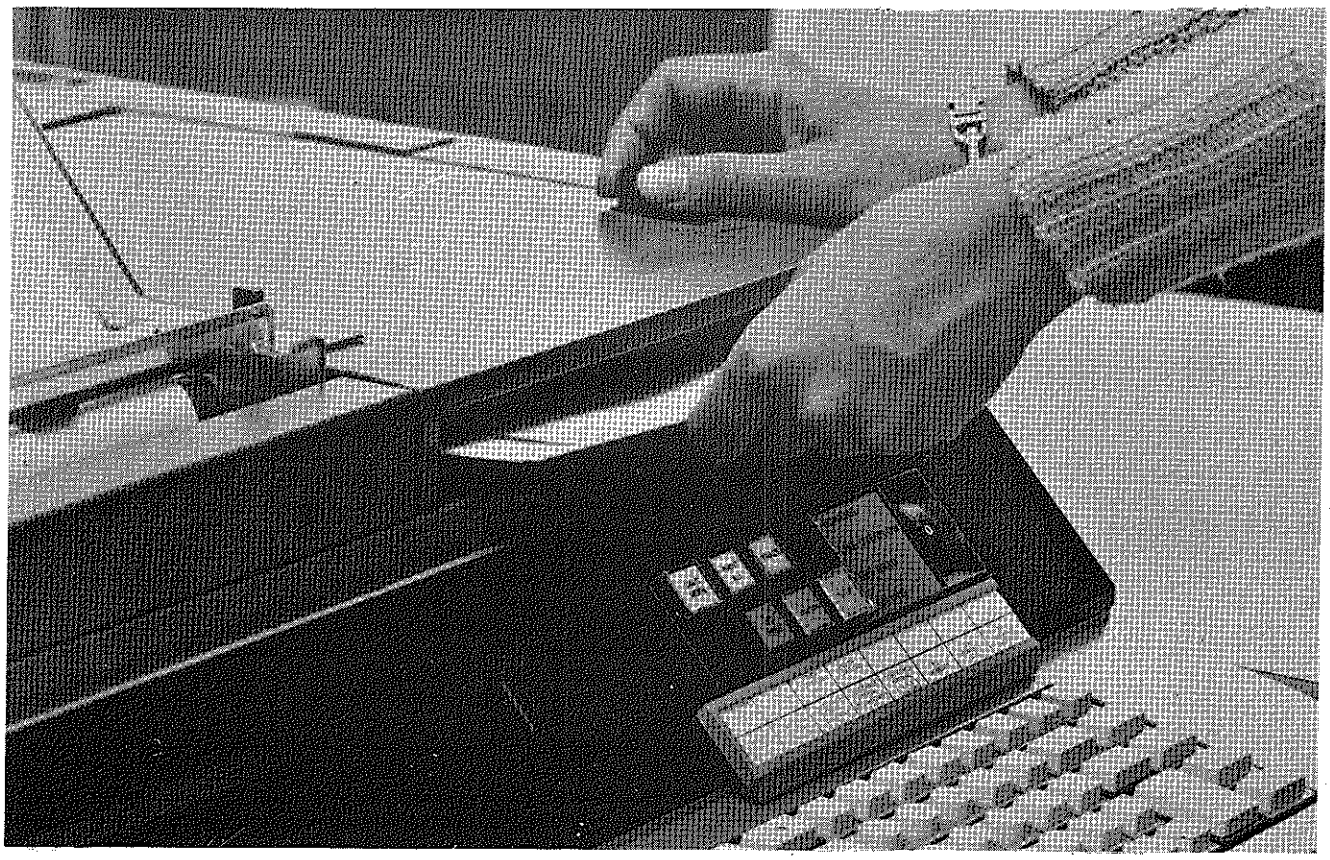


Figura 2-3 Inserimento del floppy disk in una unità monodisco

2. Inserire il disco nel trascinatore, con l'etichetta rivolta verso l'alto e verso l'operatore (come indicato in figura 2-3), finchè si sente un click.

3. Chiudere lo sportello tirando in avanti dolcemente la relativa leva.

   Attenzione: Prima di chiudere lo sportello, ci si assicuri che il sistema sia acceso.

Se si ha una unità con due trascinatori:

1. Accendere il sistema.

2. Sbloccare l'unità spingendo indietro la leva relativa. Sollevare l'unità. Aprire gli sportelli tirando indietro le relative leve.

3. Inserire il disco nel trascinatore superiore con l'etichetta rivolta verso l'alto; inserire il secondo disco con l'etichetta rivolta verso il basso, come indicato in figura 2-4.

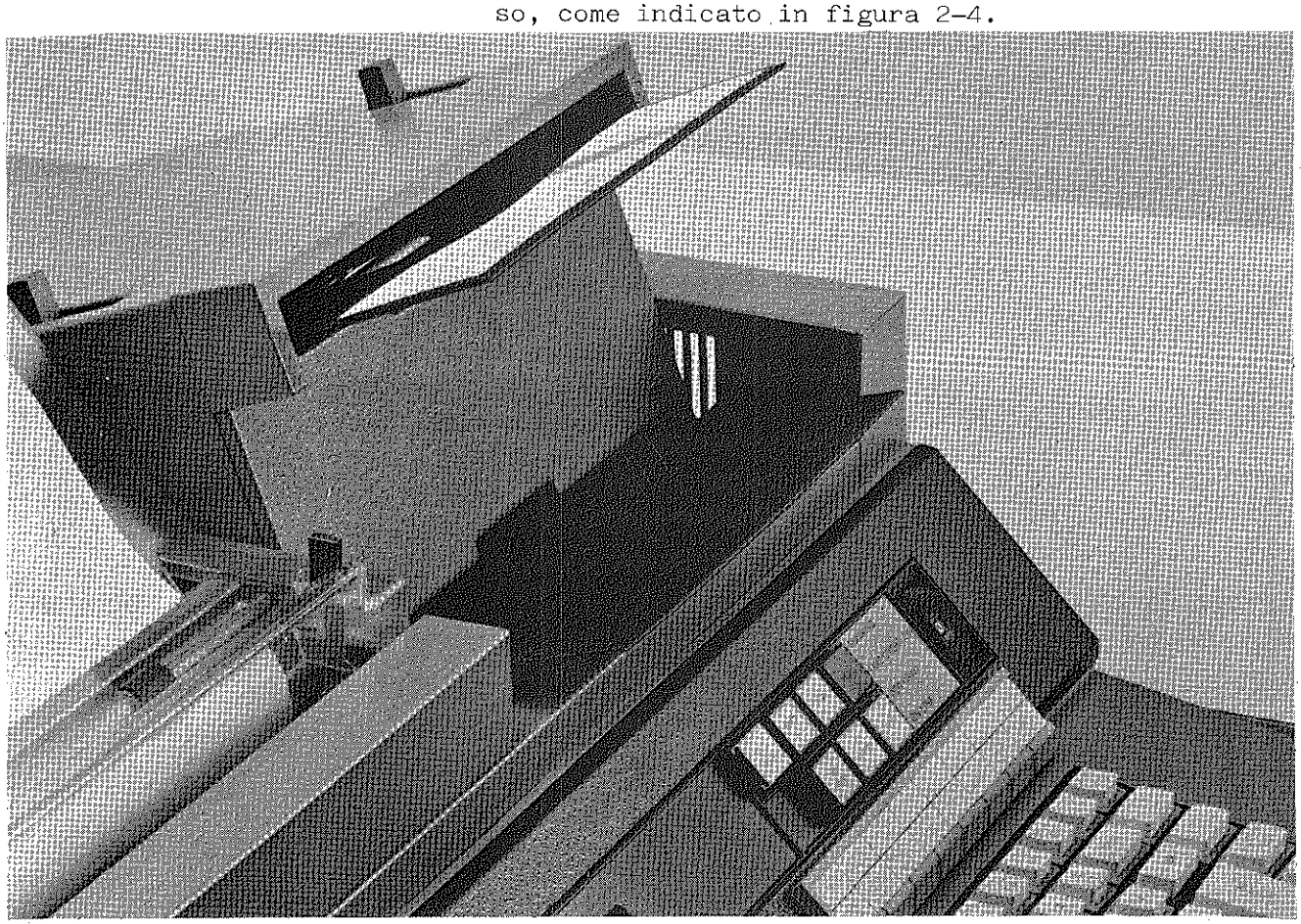


Figura 2-4 Inserimento del floppy disk in una unità bidisco

(E' utile inserire il floppy disk sistema -- che contiene il sistema operativo -- nel trascinatore inferiore). I dischi devono essere spinti verso l'interno finchè si sente un click.

4. Chiudere gli sportelli dell'unità tirando in avanti dolcemente le relativa leve.

   Attenzione: Prima di chiudere gli sportelli, ci si assicuri che il sistema sia acceso.

5. Abbassare l'unità e quindi bloccarla tirando in avanti la relativa leva.

Inizializzazione

Dopo aver inserito il floppy disk, il sistema inizia una fase detta inizializzazione. Durante l'inizializzazione, la parte di sistema operativo necessaria per interpretare quanto viene introdotto da tastiera è trasferita dal floppy disk sistema in memoria principale. Durante questa fase la luce di console RUNNING lampeggia. Quando l'inizializzazione è completata il messaggio READY è visualizzato sul display. La luce RUNNING smette di lampeggiare ma rimane accesa. Il sistema è ora pronto a ricevere i comandi e le istruzioni Assembler.

A seconda dei dischi inseriti nella unità floppy disk, il sistema si inizializza nella configurazione bidisco (floppy disk sistema e utente), o nella configurazione monodisco (solo floppy disk sistema). La configurazione con cui è stato inizializzato il sistema determina la possibilità o meno di effettuare alcune operazioni: ad esempio alcuni programmi di utilità non possono essere eseguiti se il sistema è stato inizializzato nella configurazione monodisco (vedi FDCOPY, FLCOPY, LIBCOPY nella appendice A).

Introduzione da tastiera

Da tastiera si possono introdurre:

- comandi di sistema
- stringhe di caratteri
- dati numerici
- linee di testo
- espressioni da eseguire immediatamente

I caratteri introdotti da tastiera sono memorizzati in un registro detto "buffer di tastiera" che ha una capacità di 80 caratteri. Il carattere introdotto è immediatamente visualizzato sul display alla destra della posizione indicata precedentemente dal pointer di display ed esso si sposta di una posizione verso destra.

Dal buffer di tastiera i caratteri sono trasferiti in memoria principale quando è premuto il tasto [END OF LINE] o [SUM] , a prescindere dalla posizione del pointer sul display. I caratteri sul display sono cancellati ed il pointer si pone nella prima posizione del display.

I caratteri possono essere digitati in tastiera anche mentre il sistema sta eseguendo elaborazioni, operazioni di stampa, operazioni di I/O su floppy disk, ma il comando END OF LINE o SUM è rifiutato dal sistema che emette una segnalazione acustica. Non appena la luce RUNNING è fissa i tasti [END OF LINE] e [SUM] sono abilitati.

Se si introducono, da tastiera, più di 80 caratteri l'ultimo carattere digitato è rifiutato, si ha una segnalazione acustica e si accende la luce di console LINE OVERFLOW.

Se si digitano contemporaneamente 2 caratteri la battuta è ignorata e si ha una segnalazione acustica.

Nella figura 2-5 si vede che il display può visualizzare i caratteri introdotti da tastiera oppure messaggi di programma o di sistema. I messaggi di programma o di sistema sono trasferiti dalla memoria principale ad un "buffer di display".

Se da tastiera viene digitato un carattere il messaggio sul display è cancellato ed è visualizzato il contenuto del buffer di tastiera. Quando il display visualizza il contenuto del buffer di tastiera è possibile rivedere il contenuto del buffer di display premendo [CLEAR RECALL] e viceversa premendo lo stesso tasto si collega il buffer di tastiera con il display se quest'ultimo era collegato con il buffer di display.

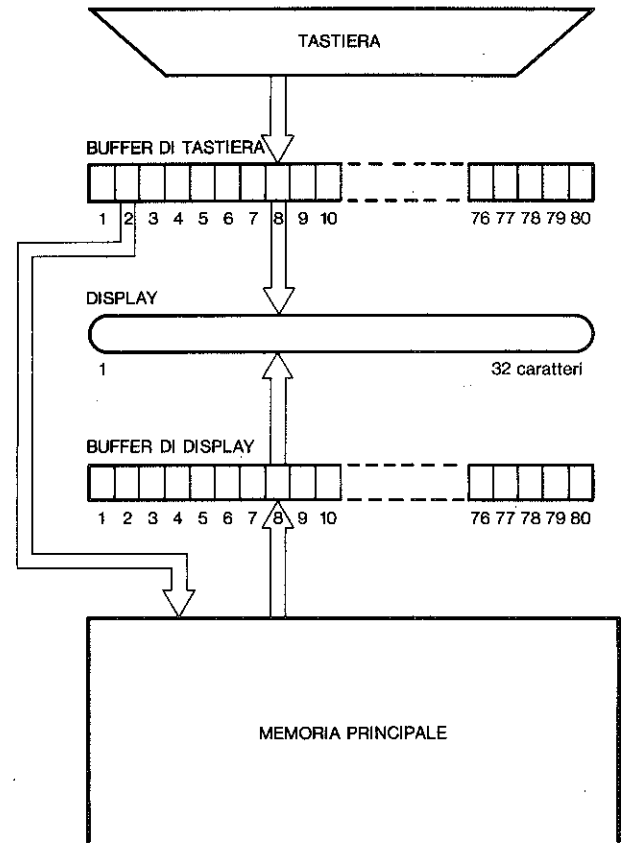


Figura 2-5 Visualizzazione sul display di caratteri introdotti da tastiera o generati da programma (o sistema)

Correzione delle introduzioni da tastiera

Prima di premere END OF LINE o SUM si possono cancellare, modificare o inserire caratteri nel buffer di tastiera.

1. Cancellazione:

   - per cancellare un carattere:

     . spostare il pointer di display nella posizione immediatamente a destra del carattere da cancellare utilizzando ← , → , SHIFT , REPEAT .

     . Premere CHAR DELETE . Il carattere è cancellato, il pointer ed i caratteri alla sua destra sono spostati di una posizione verso sinistra.

   - per cancellare tutti i caratteri presenti nel buffer di tastiera:

. premere contemporaneamente i tasti [SHIFT] e [CLEAR RECALL] a prescindere dalla posizione del pointer. I caratteri nel buffer sono tutti cancellati ed il pointer si sposta nella prima posizione del display.

2. Modificazione:

- per modificare un carattere:

. cancellare il carattere da modificare. (Vedi cancellare un carattere)

. digitare il carattere voluto. Il carattere digitato sostituisce nella posizione del buffer il carattere cancellato; il pointer è alla sua destra.

3. Inserimento:

- per inserire un carattere:

. spostare il pointer sul display nella posizione in cui si vuole inserire un nuovo carattere utilizzando [←] , [→] , [SHIFT] , [REPEAT] .

. Digitare il carattere da inserire nella stringa presente nel buffer di tastiera. Sul display il carattere viene visualizzato nella posizione in cui è inserito ed il pointer è visualizzato alla sua destra.

Introduzione della data

Le operazioni sui file possono essere datate dall'utente mediante il comando DATE. Il formato generale del comando è: DATE date, dove "date" sono 6 caratteri che esprimono nell'ordine il giorno, il mese e lo anno. Il sistema associa la data specificata a qualunque file che viene registrato su floppy disk. Così il comando DATE permette di ricordare quando un dato file è stato creato o modificato. (Tra i caratteri che compongono date non è ammesso lo spazio.)

Riconfigurabilità della memoria utente

Il sistema P6060 permette di eseguire programmi che impiegano periferiche seriali, che impiegano un video display, che impiegano una stampante IPSO in alternativa alla stampante integrata -- se l'utente lo spe-

cifica mediante un comando OPTIONS od un comando CONFIGURE. Ognuna delle prestazioni suddette richiede il caricamento in memoria utente di una specifica routine del sistema operativo.

Con il comando OPTIONS si può caricare in memoria utente la seguente routine del sistema operativo:

Nome	Funzione	Spazio di memoria utente richiesto
DEB	Permette di eseguire programmi sotto il controllo delle routine di debugging	3,5K byte

Per ulteriori informazioni si veda il comando OPTIONS nel capitolo 3.

Con il comando CONFIGURE si possono caricare in memoria utente le seguenti routine del sistema operativo:

Nome	Funzione	Spazio di memoria utente richiesto
EP	Permette l'impiego di una stampante, collegabile in alternativa ad una stampante integrata	512 byte
EVD	Permette l'impiego di un video display	1K byte

Per ulteriori informazioni si veda il comando CONFIGURE nel capitolo 3.

Ogni successiva inizializzazione del sistema, dopo la accensione, ricarica in memoria utente le routine del sistema operativo che erano state specificate con gli ultimi comandi OPTIONS e CONFIGURE eseguiti.

Stati del sistema

Per il sistema P6060 si hanno i seguenti "stati" o modi di funzionamento:

- stato comandi o di editing
- stato di esecuzione programma
- stato di debugging
- stato di esecuzione calcoli immediati

Stato comandi

Nello stato comandi si può:

- digitare un comando di sistema (vedi capitolo 3)
- digitare un comando che richiama in memoria ed esegue un programma di utilità (vedi appendice A)
- premere il tasto di console CALC MODE (vedi capitolo 7)
- introdurre un programma Assembler sotto forma di file testo

Il sistema è nello stato comandi:

- al termine della inizializzazione
- dopo la esecuzione di un comando di sistema
- dopo la esecuzione di un programma di utilità
- dopo la esecuzione di un programma utente
- se si richiede l'esecuzione di un programma che non può essere eseguito per insufficiente spazio in memoria principale
- se una operazione è stata interrotta dal comando di console BRK AR
- se è stato premuto il tasto di console CALC MODE mentre il sistema era nello stato di esecuzione calcoli immediati

Quando il sistema è nello stato comandi la luce RUNNING è fissa.

Stato di esecuzione programma

Quando il sistema è nello stato di esecuzione programma esegue un programma utente o un programma di utilità. Il sistema è nello stato di esecuzione programma dopo la introduzione di:

- un comando EXQ
- un comando EXEC

La luce running lampeggia quando il sistema è nello stato di esecuzione programma. Quando il sistema esegue un programma utente anche la luce del tasto CONTINUE è accesa.

Esecuzione: Per eseguire un programma, si deve introdurre il comando EXQ. Se durante l'esecuzione è rilevato un errore l'esecuzione del programma è interrotta ed è visualizzato il messaggio relativo (vedi appendice D). L'operatore può effettuare l'azione di recupero e quindi far riprendere l'esecuzione del programma premendo CONTINUE o STEP.

Interruzione della esecuzione di un programma: L'esecuzione di un programma è interrotta quando:

- è premuto il tasto STEP e l'opzione DEB è attiva

- è rilevato un errore

- si è verificata una anomalia nel funzionamento del sistema

Ripresa dell'esecuzione: L'esecuzione di un programma prosegue quando:

- è premuto il tasto STEP nello stato di debugging
- è premuto il tasto CONTINUE

Fine esecuzione: L'esecuzione di un programma termina quando:

- è eseguita l'istruzione END
- si preme il tasto BREAK

Quando è completata l'esecuzione di un programma il sistema è nello stato comandi.

Stato di debugging

E' lo stato del sistema che permette di verificare la correttezza di un programma. Per una spiegazione dettagliata si veda il capitolo 8.

Stato di esecuzione calcoli immediati

E' lo stato del sistema che permette di eseguire immediatamente delle espressioni algebriche introdotte da tastiera dopo che si sia premuto il tasto END OF LINE o SUM . Per una spiegazione dettagliata si veda il capitolo 7.

Working File

L'area di memoria principale occupata da un programma utente in formato eseguibile, un programma di servizio o da un file testo è definita working file. Nel working file può essere memorizzato un solo programma od un solo testo per volta. Il contenuto del working file viene cancellato quando si spegne il sistema. Dopo l'accensione del sistema, terminata la fase di inizializzazione, o dopo l'esecuzione dei comandi OPTIONS e CONFIGURE il sistema inizializza il working file per l'eventuale introduzione da tastiera. Se in memoria principale esiste già un programma o file testo per introdurre un nuovo file testo da tastiera si deve eseguire il comando TEXT.

Creazione ed editing di un file testo

Struttura di un file testo

Un file testo è composto da un insieme di linee ognuna delle quali contiene un numero di linea seguito da caratteri scelti nel set P6060 (vedi appendice C).Ogni linea può contenere al massimo 80 caratteri (compreso il numero di linea). Ecco un esempio di file testo:

```
LIST
FILE     PR2

0010 BEGIN  CSECT
0020          BALR   9,0
0030          USING   *,9
0040          ST      2,A
0050          BR      3
0060          B       B1
0070          DS      0F
0080          DS      CL1
0085 A        DS      CL4
0090 B1       LR      4,*
0100         BR      14
0110         END

END OF LISTING
```

I file testo possono essere usati per creare programmi in linguaggio Assembler da registrare su floppy disk formato sorgente al fine di essere successivamente tradotti nel formato eseguibile mediante le utility ASSEMBLY e LNK. La descrizione del formato sorgente Assembler è fatta al capitolo 4.

Introduzione ed editing di un testo

L'introduzione da tastiera di un file testo avviene linea per linea. Il sistema verifica che ogni linea sia preceduta da un numero di linea ed in caso affermativo la trasferisce dal buffer di tastiera nel working file. Se la linea non è preceduta da un numero di linea viene fornita una segnalazione di errore sul display ed emessa una segnalazione acustica. Il trasferimento della linea nel working file non avviene. Introducendo il numero di linea all'inizio della linea, una successiva pressione di END OF LINE trasferisce la linea nel working file. Il numero di linee che si possono introdurre dipende dalla capacità della memoria utente disponibile.

Quando le linee del file testo sono presenti nel working file si possono ulteriormente editare utilizzando i seguenti tasti e comandi:

- tasti: ← → ↑ ↓ SHIFT CHAR DELETE CLEAR RECALL END OF LINE
- comandi: DELETE LINE, FETCH, RESEQUENCE

Quando il file testo è pronto lo si può registrare permanentemente sul supporto esterno utilizzando il comando SAVE o REPLACE (vedi capitolo 3).

3. COMANDI DI SISTEMA

La comunicazione con il sistema è realizzata mediante un linguaggio costituito da comandi di sistema che permettono la creazione ed esecuzione di un programma Assembler in modo semplice ed immediato. Con i comandi disponibili si possono -- tra le altre cose:

- creare e mantenere file di lavoro su floppy disk e file testo
- eseguire programmi
- modificare programmi
- registrare programmi
- gestire librerie e sottolibrerie

Inoltre i comandi permettono lo scambio di informazioni tra la memoria principale e le unità esterne del P6060. Come usare i comandi di sistema è l'argomento di questo capitolo. Prima di entrare nei dettagli sui singoli comandi, tuttavia, sarà utile familiarizzare con i concetti ed i termini associati al linguaggio dei comandi. Questo è lo scopo dei paragrafi che seguono.

Floppy disk, librerie, sottolibrerie e file

I comandi di sistema permettono la gestione di floppy disk (sistema ed utente), della libreria di software applicativo (che può contenere tre sottolibrerie) di file (lavoro su floppy disk, testo, oggetto rilocabile ed oggetto eseguibile).

Floppy disk e librerie

Vi sono due tipi di floppy disk: sistema e utente.

Floppy disk sistema: I floppy disk sistema contengono le seguenti librerie: libreria di firmware residente (etichettata P6FWR), libreria di firmware opzionale (etichettata P6FWO), libreria di software di base (etichettata P6SW) e, opzionalmente, la libreria di

software applicativo (etichettata P6FSYS). Le prime tre librerie non sono accessibili all'utente e costituiscono il sistema operativo; la quarta libreria può essere inizializzata dall'utente e può contenere fino a tre sottolibrerie: la sottolibreria package, la sottolibreria comune e la sottolibreria utente. La sottolibreria package è così chiamata poichè i file ivi contenuti sono considerati finalizzati a svolgere funzioni applicative standard. La sottolibreria comune è così chiamata perchè abitualmente utilizzata per contenere file con cui l'utente lavora normalmente. La sottolibreria utente è così chiamata perchè abitualmente utilizzata per contenere file con cui l'utente crea e mette a punto i programmi. Se l'unità ha un solo trascinatore, si può utilizzare solamente il floppy disk sistema.

Sottolibrerie

Il P6060 permette la registrazione di file come membri di tre diversi tipi di sottolibrerie:

- sottolibreria package *
- sottolibreria comune +
- sottolibreria utente

La distinzione fondamentale tra le sottolibrerie consiste nel diverso grado di protezione. Per creare una sottolibreria su floppy disk si deve inizializzare il disco: (1) indicando il tipo di sottolibreria da creare, (2) specificando il numero di file che la sottolibreria potrà contenere. Questo viene attuato eseguendo il programma di utilità LBCREATE descritto in dettaglio nell'appendice A.

Sottolibreria package: La sottolibreria package può contenere i quattro tipi di file. Una sottolibreria package viene prodotta al termine delle seguenti operazioni:

1. Inizializzazione di un floppy disk eseguendo il programma di utilità LBCREATE.

2. Creazione dei file e loro registrazione sul disco

3. Esecuzione del programma di utilità LBPROTECT per fornire la necessaria protezione.

Al termine di questa sequenza di operazioni la sotto-

libreria package è protetta dall'azione dei seguenti comandi:

CREATE
MODIFY (non si può modificare il nome di un file)
PURGE
SAVE

e dei seguenti programmi di utilità:

FLCOPY
LIBCOPY

Si noti che i file delle sottolibrerie package sono protetti contro le operazioni di listing e di editing. La protezione può essere parziale nel senso che può operare a partire da un numero di linea (o da un certo dato) in poi; questo consente la personalizzazione dei file da parte dell'utente.

Sottolibreria comune: La sottolibreria comune può contenere i quattro tipi di file. Una sottolibreria comune viene prodotta al termine delle seguenti operazioni.

1. Inizializzazione di un floppy disk eseguendo il programma di utilità LBCREATE

2. Creazione dei file e loro registrazione sul disco

3. Esecuzione del programma di utilità LBPROTECT per fornire la necessaria protezione.

Dopo l'esecuzione di questa sequenza di operazioni, la sottolibreria comune è protetta contro l'azione dei seguenti comandi:

MODIFY (non si può modificare il nome di un file)
PURGE

Sottolibreria utente: La sottolibreria utente può contenere i quattro tipi di file. Una sottolibreria utente è creata eseguendo il programma di utilità LBCREATE. Il contenuto di una sottolibreria utente non è protetto; si possono modificare, cancellare o aggiungere file nella libreria usando i comandi appropriati

Nota: Si osservi che la protezione di una sottolibreria non implica la protezione delle informazioni contenute nei suoi file (che si ottiene con il comando SECURE).

Floppy disk utente: I floppy disk utente sono inizializzati dall'utente (l'inizializzazione viene realizzata dall'utility LBCREATE descritta nell'appendice A) e possono solo contenere la libreria di software applicativo; questa può contenere tre sottolibrerie a seconda di come è stato inizializzato il floppy disk utente. Le caratteristiche della libreria di software applicativo su floppy disk utente sono uguali a quelle della libreria su floppy disk sistema.

I file

I comandi di sistema possono fare riferimento a quattro tipi diversi di file contenuti in una delle sottolibrerie suddette: testo, lavoro su floppy disk, oggetto rilocabile e oggetto eseguibile.

File testo: Come già definito nel capitolo 2, un file testo è un insieme di linee numerate e memorizzate in memoria in formato sorgente -- il formato con cui sono introdotte. Ogni linea può contenere un qualsiasi carattere del set P6060.

File di lavoro su floppy disk: Un file di questo tipo viene creato dall'utente per mezzo del comando CREATE che alloca spazio su FDU. L'accesso a questo file viene realizzato dal sistema in modo automatico, mediante il richiamo di moduli di sistema READ e WRITE (vedi capitolo 6), purchè nel comando EXQ sia presente l'argomento W = filename1 (vedi comando EXQ).

Nota: Il file di lavoro su floppy disk non deve essere confuso con il working file di RAM descritto nel capitolo 2.

File oggetto rilocabile: Un file di questo tipo è composto da codice oggetto in formato rilocabile, di solito usato quale input al Linker. Un file oggetto rilocabile rappresenta l'output dell'assemblatore.

File oggetto eseguibile: Un file di questo tipo è composto da codice oggetto il formato eseguibile, usato quale argomento del comando EXQ al fine di mandare in esecuzione l'oggetto che contiene. Un file

oggetto eseguibile reppresenta l'output del Linker.

Nomi di file: Ogni file registrato su floppy disk è identificato da un nome. Il nome del file permette la ricerca di esso e la sua protezione da un accesso non autorizzato. Lo stesso nome può essere assegnato a due diversi file, ma i file devono essere registrati su diversi floppy disk. Si ricordi, comunque, che se si ha una configurazione bidisco ed esistono due file con lo stesso nome (uno su un floppy disk sistema e l'altro su un floppy disk utente), il sistema, per ogni comando che specifica il nome del file, si riferirà al floppy disk sistema.

I file registrati nella sottolibreria package o nella sottolibreria comune possono avere nomi composti da due a sette caratteri. Quelli registrati nella sottolibreria utente possono avere nomi da una a sei caratteri. Un file della sottolibreria package deve avere un asterisco (*) come primo carattere del suo nome; un file della sottolibreria comune deve avere un segno più (+) come primo carattere del suo nome; un file della sottolibreria utente deve avere un carattere alfabetico come primo carattere del suo nome.

Per le sottolibrerie package e comune, il secondo carattere del nome di un file deve essere alfabetico. I restanti caratteri dei nomi di tutti i file (appartenenti ad una sottolibreria package, comune od utente) devono essere alfanumerici. Nel nome di un file non ci possono essere uno o più spazi interni. Tutti i caratteri alfabetici del nome di un file devono essere maiuscoli.

Libreria package	- nome corretto	*SINES
	nome scorretto	* (meno di due caratteri
Libreria comune	- nome corretto	+G
	nome scorretto	+8G (secondo carattere non alfabetico
Libreria utente	- nome corretto	GRAPH2
	nome scorretto	GRAPH66 (più di sei caratteri)

Introduzione di un comando

Il comando è composto da una parola chiave seguita, di solito, da uno o più operandi. La parola chiave è un verbo inglese che descrive la funzione del comando. Gli operandi forniscono le informazioni specifiche che permettono al comando di eseguire l'operazione richiesta. Le parole chiave dei comandi più usati possono essere introdotti premendo un solo tasto della sezione comandi della tastiera. Questo rende più rapida l'introduzione da tastiera del comando e riduce la possibilità di errore. Tutte le parole chiave possono essere abbreviate nei loro primi tre caratteri. Infatti, il sistema analizza solamente questi caratteri per determinare quale comando deve essere eseguito. Per questo motivo il sistema accetta come parole chiave corrette quelle che contengono dei caratteri errati dopo le prime tre lettere.

Dopo aver introdotto la parola chiave, si digitano gli operandi carattere per carattere, e si completa l'introduzione premendo il tasto END OF LINE. Si devono inserire uno o più spazi tra la parola chiave ed il primo operando. Dopo essere introdotto, il comando è analizzato. Se è rilevato un errore sintattico viene immediatamente visualizzato un messaggio di errore; altrimenti il comando è eseguito. Un comando non deve mai terminare con una virgola prima di END OF LINE.

Notazioni

Le seguenti notazioni sono impiegate nella descrizione dei comandi di sistema:

{ } racchiude un insieme di parametri che non sono opzionali; uno di essi deve essere specificato.

[ ] racchiude un gruppo di parametri che sono opzionali; un parametro o nessun parametro può essere specificato.

[-] indica un parametro assunto implicitamente dal sistema; così se si sceglie un parametro sottolineato non è necessario digitarlo.

... indica che il precedente operando può essere ripetuto più di una volta.

, separa gli operandi di un comando

Nota: per parametro si intende il valore assegnato ad

un operando di un comando; es. in SAVE U MAT1, U è il valore (parametro) assegnato al primo operando.

I seguenti simboli sono usati per definire il formato di un comando, ma non devono essere digitati:

- trattino di unione
_ sottolineatura
{ } parentesi graffe
[ ] parentesi quadre
... puntini.

Le lettere minuscole, le parole con lettere maiuscole ed i seguenti simboli devono essere digitati esattamente come indicati nella definizione del comando:

segno di numero
* asterisco
+ segno più
: due punti
, virgola

Elenco e funzione dei comandi di sistema

I comandi di sistema e le loro funzioni sono elencati in ordine alfabetico come segue:

Nome	Funzione
AUTO #	Genera la numerazione automatica delle linee introdotte da tastiera
CATALOG	Stampa il catalogo del contenuto delle librerie
CONFIGURE	Definisce una specifica configurazione di sistema
CREATE	Alloca spazio su floppy disk per un file di lavoro su floppy disk
DATE	Registra su floppy disk sistema la data introdotta da tastiera per datare le operazioni sui file esterni
DCHANGE	Permette di sostituire un floppy disk con un altro mentre il sistema è acceso: così il programma presente nella memoria principale non è cancellato
DELETE LINE	Cancella una o più linee di file testo presenti in memoria principale

Nome	Funzione
EXEC	Carica in memoria principale ed esegue un programma di utilità (appendice A)
EXQ	Inizia l'esecuzione di un programma
FETCH	Trasferisce nel buffer di tastiera una linea di file testo presente in memoria principale
LDKEYS	Assegna ai tasti funzione il contenuto registrato sul floppy disk sistema
LIST	Stampa una o più linee di un file testo presente in memoria principale
MODIFY	Modifica il nome di un file e/o la dimensione di allocazione di un file di lavoro su floppy disk
OLD	Carica in memoria principale un file testo presente su floppy disk
OPTIONS	Definisce le OPZIONI del floppy disk sistema
PURGE	Cancella un file in una libreria su floppy disk
REPLACE	Sostituisce un file testo, presente su floppy disk, con un altro avente lo stesso nome, presente nella memoria principale
RESEQUENCE	Modifica la numerazione delle linee del file testo presente in memoria principale
SAVE	Registra un file testo su floppy disk
SHIFT	Modifica la numerazione delle linee del file testo presente in memoria principale, iniziando da una linea specificata
SPACE	Stampa lo spazio disponibile per ulteriori registrazioni su floppy disk
STKEYS	Registra su floppy disk sistema il contenuto assegnato ai tasti funzione
TEXT	Permette di introdurre un file testo da tastiera.

Comando AUTO#

Funzione

Genera la numerazione automatica delle linee di un file testo.

Formato

AUT [O#] [line-num] [,increment]

dove:

line-num
: è un numero intero positivo compreso tra 1 e 9999, che specifica il valore che sarà associato alla linea di testo introdotta successivamente

increment
: è un numero intero positivo che indica quale valore deve essere aggiunto ad ogni numero di linea per generare il numero di linea successivo.

Azione

Il comando comunica al sistema che alla linea di testo introdotta successivamente deve premettere il numero di linea line-num e alle successive linee deve premettere il numero che si ottiene aggiungendo all'ultimo generato l'incremento increment.

Ogni volta che una linea di testo è trasferita in memoria principale, dopo la pressione di (END OF LINE) , il sistema introduce nel buffer di tastiera il numero della linea successiva che è contemporaneamente visualizzato sul display.

Il comando privo della parte opzionale comunica al sistema che la numerazione deve iniziare dal numero che si ottiene aggiungendo 10 al più grande numero di linea presente in memoria principale e che l'incremento è 10. Se in memoria principale non vi è presente alcuna linea la numerazione iniza da 10.

Il comando privo dell'opzione line-num comunica al sistema che la numerazione delle linee introdotte

successivamente inizia dal numero ottenuto aggiungendo increment al più grande numero di linea presente in memoria principale. Se in memoria principale non vi è alcuna linea, la numerazione inizia dal numero specificato con increment.

Il comando privo dell'opzione increment comunica al sistema che la numerazione inizia dal valore specificato con line-num e che l'incremento è 10.

Note

1. Per interrompere la numerazione automatica si deve premere [CLEAR RECALL] con [SHIFT] ; il sistema passa nello stato comandi. Se si vuole ripristinare la numerazione automatica si deve introdurre nuovamente il comando AUTO #.

2. La parola chiave del comando può essere introdotta premendo il tasto [AUTO#] della sezione comandi.

Esempi

1. Richiedere la numerazione automatica per un nuovo testo iniziando dal numero di linea 5 con passo 20.

   Premere [A][U][T] [5][,][2][0] [END OF LINE]

2. Riprendere la numerazione automatica del seguente testo presente in memoria principale iniziando dal numero di linea 40 con passo 20. In memoria principale sono presenti le linee:

```
10 * ZERI DI UNA FUNZIONE REALE
20 BUFFER DS 80
```

   Premere [AUTO#] [4][0][,][2][0] [END OF LINE]

3. Inserire istruzioni di commento nel seguente file testo presente in memoria principale iniziando con il numero di linea 5 con passo 100.

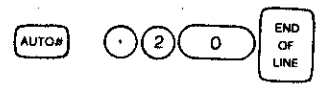
```
10  ...
20  ...
30  ...
...
...
...
500         END
```

Premere 

Si introducono le istruzioni * volute e come risultato si ha un programma del tipo:

```
5 * ...
10 ...
20 ...
30 ...
...
...
...
105 * ...
...
...
...
205 * ...
...
...
...
500          END
```

4. Richiedere la numerazione automatica di un file testo iniziando dal numero di linea 20 con passo 20.

Premere 

Comando CATALOG

Funzione

Stampa l'indice del contenuto dei floppy disk.

Formato

dove:

S indica il floppy disk sistema

U indica il floppy disk utente

* indica la sottolibreria package

\+ indica la sottolibreria comune

: indica tutte le sottolibrerie

filename indica il file specificato nella libreria di software applicativo

T indica file testo

E indica file oggetto eseguibile

O indica file oggetto rilocabile

W indica file di lavoro su floppy disk

F indica che sono richieste tutte le informazioni sui file

Azione

Il comando completo di tutti gli operandi (ad es. CATALOG U, :, T, F)comunica al sistema di stampare le seguenti informazioni per i file identificati mediante i primi tre operandi:

- nome del file

- tipo di file

- data di creazione del file

- data dell'ultima modifica del file

- lunghezza di allocazione del file (in byte)

- lunghezza attuale del file (in byte)

- codice di identificazione del file (se appartenente ad un package fornito dalla Olivetti S.p.A.)

Il comando privo del quarto operando comunica al sistema di stampare le seguenti informazioni abbreviate per i file identificati mediante gli altri operandi:

- nome del file
- tipo di file

Il comando privo del terzo operando comunica al sistema di stampare le informazioni relative a file di tipo testo, lavoro su floppy disk, oggetto eseguibile e rilocabile.

Il comando privo del secondo operando comunica al sistema di stampare le informazioni relative solo ai file della sottolibreria utente.

Note

1. Se il secondo operando è filename, il terzo ed il quarto operando non devono essere digitati; in questo caso sono stampate tutte le informazioni riguardanti il file di nome filename.

2. L'esecuzione del comando può essere terminata premendo il tasto di console BREAK

3. Se la configurazione di sistema installata è priva di stampante integrata e di stampante ausiliaria (vedi comando CONFIGURE), le informazioni fornite all'utente del comando CATALOG sono visualizzate sul display. In questo caso per leggere le informazioni prodotte in una linea si devono utilizzare i tasti [→], [REPEAT] e [SHIFT] come spiegato nel capitolo 1 § "Tastiera". Per visualizzare ogni linea si deve premere il tasto CONTINUE : ad ogni pressione appare sul display una diversa linea di catalogo e

quando, alla pressione di CONTINUE il sistema produce un segnale acustico significa che non vi sono più linee di catalogo da visualizzare.

Esempi

1. Si richieda la stampa del nome e tipo di file residenti sul floppy disk utente.

   Premere C A T U . : END OF LINE

   Il sistema stampa:

```
CAT U,:

       *  K0ASM1-R 1.0  *          VOLLABEL =
FILE    TYPE  CREAT     LAST MOD  MAX SIZE  USED SIZE   CODE       EXT



 TEQ2A    T
 TDC2A    T
 INIZM    T
 INIZ     T
 END      T
 START    T
 TR1GE    E
 START1   T
 TR1FE    E
```

2. Si richiedano le informazioni relative al file PROV1 residente su floppy disk utente.

   Premere C A T U , I N I Z END OF LINE

   Il sistema stampa:

```
CAT U,INIZ
FILE    TYPE  CREAT     LAST MOD  MAX SIZE  USED SIZE   CODE       EXT

 INIZ     T  20-02-78  20-02-78   384         384                    1
```

Comando CONFIGURE

Funzione

Definisce una dimensione logica della memoria utente inferiore a quella fisica effettivamente presente e permette l'impiego di periferiche esterne con funzione ausiliaria a quelle integrate

Formato

CON [FIGURE] [EVD] [, EP = n_1] [, MS = n_2]

dove:

EVD specifica che viene utilizzato un display video esterno (collegato mediante interfaccia RS 232 - C)

n_1 è un numero intero compreso tra zero e 31 che identifica una stampante esterna

n_2 è un numero intero compreso tra uno e 48, che indica la capacità logica, in K byte, da assegnare alla memoria utente.

Azione

Il comando completo di tutti gli operandi ordina al sistema di inizializzarsi configurando la memoria utente con la capacità specificata in byte con l'operando MS = n_2; ordina inoltre che le operazioni di stampa riferite alla stampante integrata vengano eseguite sulla stampante il cui nome logico è specificato con l'operando EP = n_1 e che sullo schermo del display video siano visualizzati tutti i testi che appaiono sul display integrato e sulla stampante integrata.

Se non è specificato l'operando MS, il comando CONFIGURE ordina al sistema di inizializzarsi configurando la memoria utente con la capacità fisicamente presente nell'unità centrale.

Se l'operando EP non è specificato, il comando CON-

FIGURE ordina al sistema che le operazioni di stampa comandate da sistema o da programma utente non siano più eseguite su di una stampante ma bensì sulla stampante integrata.

Se l'operando EVD non è specificato i testi inviati alla stampante integrata e al display integrato non vengono trasmessi all'unità video display esterno.

Note

1. L'esecuzione del comando CONFIGURE comporta una reinizializzazione del sistema con le modalità specificate dal comando che vengono registrate sul floppy disk sistema e rimangono valide finchè non è eseguito un nuovo comando CONFIGURE.

2. La possibilità di configurare il sistema assegnando capacità di memoria utente minore di quella reale permette di verificare la corretta esecuzione di programmi destinati ad essere eseguiti su sistemi con capacità di memoria utente pari a quella simulata.

3. La stampante, utilizzata in alternativa alla stampante integrata, può essere una delle seguenti:

   PR 1220
   PR 1230
   PR 1240
   PR 1350
   PR 1370

4. I caratteri che non rientrano nel set di quelli stampabili dalla stampante specificata con il comando CONFIGURE provocano la stampa del carattere ⁞⁞⁞. Alcuni dei caratteri compresi tra i primi 32 caratteri del set ISO sono, per le stampanti, dei comandi; per cui se uno di questi caratteri è compreso nella linea da stampare sarà interpretato dalla stampante come comando e la stampante eseguirà le azioni ad esso relative. Gli errori relativi alla stampante specificata nel comando CONFIGURE sono segnalati visualizzando ABN PRT, come per la stampante integrata.

5. Il video display esterno con interfaccia seriale, usato con funzioni ausiliarie alla stampante e al display integrati, può essere il video display

Olivetti TCV 451 (o un altro con set di caratteri di controllo compatibile con DEC VT52).

6. Se viene eseguito CONFIGURE con EVD, sul video display esterno sono visualizzate le informazioni che in condizioni normali vanno solo alla stampante integrata e al display integrato.

   Più precisamente, nella prima riga dal basso sono visualizzati i messaggi emessi dal sistema; nella seconda linea è visualizzata la linea introdotta da tastiera, nella terza riga non sono visualizzati caratteri e le righe superiori del quadro presentano le informazioni normalmente destinate alla stampante integrata, rispettando la logica di funzionamento del tasto PRINT ALL . Se il tasto NO PRINT non è attivo, le informazioni sono presentate contemporaneamente su stampante integrata e su video display esterno. Attivando il tasto NO PRINT si blocca il funzionamento della stampante.

   Le righe più in alto della terza compongono la "pagina" del display. La pagina viene riempita dal basso verso l'alto e quando è "piena" il tasto di console CONTINUE lampeggia per avvertire lo utente che vi sono altri caratteri che compongono l'informazione da visualizzare. Premendo il tasto CONTINUE la pagina viene riempita, dal basso verso l'alto, con i rimanenti caratteri dell'informazione ed i caratteri precedentemente visualizzati sono spostati verso l'alto mentre quelli che erano nella linea più in alto sono man mano sostituiti.

Comando CREATE

Funzione

Alloca lo spazio necessario a contenere un file di lavoro su floppy disk.

Formato

CRE [ATE] [S/U] ,filename,w [,n]

dove:

S indica il floppy disk sistema

U indica il floppy disk utente

filename indica il nome del file

W indica file di lavoro su floppy disk

n indica il numero di byte assegnati al file di lavoro su floppy disk; deve essere un numero intero compreso tra 1 e 240590.

Azione

Il comando completo con tutti gli operandi comunica al sistema di allocare, per il file di lavoro su floppy disk di nome filename, n byte sul floppy disk specificato.

Il comando privo del quarto operando comunica al sistema di allocare, per il file di lavoro su floppy disk di nome filename, 4096 byte sul floppy disk specificato.

Note

1. Su un floppy disk non vi possono essere due file con lo stesso nome.

2. Non si possono creare in una sottolibreria più file di quanti dichiarati per essa durante l'esecuzione del programma di utilità LBCREATE (vedi ap-

pendice A).

3. Il numero di byte richiesto viene arrotondato al successivo multiplo di 128.

Esempio

Si riservino 4096 byte nella sottolibreria utente, su floppy disk sistema, per il file di lavoro su floppy disk DATI.

Premere [C][R][E] [,][D][A][T][I][,][W] [END OF LINE]

DATE

Comando DATE

Funzione

Permette di datare le operazioni sui file registrati su floppy disk.

Formato

DAT [E] date

dove:

date è una sequenza di 6 caratteri che indica una data.

Azione

Il comando comunica al sistema di registrare sul floppy disk sistema la data introdotta da tastiera come operando del comando stesso.

Si devono sempre introdurre 6 caratteri e nessuno di essi può essere lo spazio.

Note

1. Durante l'inizializzazione del sistema, (dopo l'accensione o durante l'esecuzione dei comandi OPTIONS e CONFIGURE) la sequenza di caratteri registrata per ultima sul floppy disk sistema, presente nella unità, è caricata in memoria principale e quindi utilizzata per datare le operazioni sui file esterni.

2. La sequenza introdotta sarà, in generale: ggmmaa (giorno, mese ed anno).

Esempi

1. Si registri la data: 1 ottobre 1976

   Premere [D] [A] [T] [0] [1] [1] [0] [7] [6] [END OF LINE]

2. Si registri solo il mese (in lettere) e l'anno.

   Premere [D] [A] [T] [O] [T] [T] [O] [7] [6] [END OF LINE]

Comando DCHANGE

Funzione

Permette di sostituire un floppy disk mentre il sistema è in funzione senza cancellare il contenuto della memoria principale, oppure di modificare la configurazione del sistema da monodisco a bidisco.

Formato

DCH [ANGE] [S/U]

dove:

S indica floppy disk sistema

U indica floppy disk utente

Azione

Il comando comunica al sistema (se in configurazione bidisco) che si vuole sostituire il floppy disk indicato con l'operando.

Se il sistema è nella configurazione monodisco il comando, nella forma DCHANGE U, inizializza il sistema nella configurazione bidisco permettendo, così, l'impiego del disco utente. Introdotto il disco si deve premere il tasto di console CONTINUE.

Dopo aver introdotto il comando con l'operando S, o senza alcun operando, il sistema visualizza il messaggio:

INSERT DISK

e premendo il tasto [→] con [SHIFT] viene visualizzato il messaggio:

NEW SYSDIS ON DRIVE *

che indica di inserire il disco sistema nel trascinatore superiore dell'unità floppy disk; oppure il messaggio:

NEW SYSDIS ON DRIVE **

che indica di inserire il disco sistema nel trasci-

natore inferiore dell'unità floppy disk. Dopo aver introdotto il disco richiesto si deve premere il tasto di console CONTINUE .

Dopo aver introdotto il comando con l'operando U, il sistema visualizza il messaggio: INSERT DISK
e premendo il tasto → con SHIFT viene visualizzato il messaggio:

NEW USDIS ON DRIVE *

che indica di inserire il disco utente nel trascinatore superiore dell'unità floppy disk; appure il messaggio:

NEW USDIS ON DRIVE **

che indica di inserire il disco utente nel trascinatore inferiore dell'unità floppy disk. Dopo aver introdotto il disco richiesto si deve premere il tasto di console CONTINUE .

Note

1. Se si sostituisce un floppy disk sistema, il nuovo disco deve appartenere alla stessa release.

2. Il comando deve essere usato ogni qual volta si vuole sostituire un floppy disk con un altro oppure inserire un secondo disco, mentre la macchina è accesa.

3. Se si apre lo sportello di un trascinatore della unità floppy disk senza prima aver introdotto il comando DCHANGE il sistema visualizza il messaggio: ABN FD - DCH OMITTED, quando esegue una operazione di accesso al floppy disk presente nel relativo trascinatore. L'esecuzione dell'eventuale programma o comando è interrotta e riprende dal punto in cui è stata interrotta se si preme CONTINUE e poi

   Se il floppy disk introdotto non è stato inizializzato con il programma di utilità LBCREATE, oppure è un floppy disk sistema, viene visualizzato il messaggio: USDIS NOT INITLZD ed il sistema è nello stato comandi.

Esempio

Si introduca un floppy disk utente in sostituzione di un'altro già presente sull'unità, il cui spazio libero sia insufficiente a contenere un programma presente in memoria principale.

Premere D C H U END OF LINE

Introdurre il disco e quindi premere CONTINUE .

DELETE LINE

Comando DELETE LINE

Funzione

Cancella una o più linee del file testo presenti in memoria principale.

Formato

DEL [ETE LINE] [line-num$_1$ [,line-num$_2$]]

dove:

line-num$_1$ indica il numero della linea da cancellare o la prima linea di un insieme di linee da cancellare

line-num$_2$ indica l'ultima linea di un insieme di linee da cancellare

Azione

Il comando, completo di tutti gli operandi, comunica al sistema di cancellare le linee del file testo presente in memoria principale comprese tra i numeri di linea indicati con il primo e secondo operando (estremi inclusi).

Il comando, privo del secondo operando, comunica al sistema di cancellare la linea specificata con il primo operando.

Il comando, privo di operandi, comunica al sistema di cancellare una delle seguenti linee:

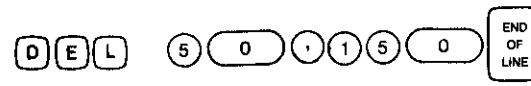
- l'ultima linea corretta introdotta da tastiera
- l'ultima linea visualizzata con il comando FETCH od i tasti [↑] e [↓]
- l'ultima linea stampata mediante un comando LIST.

Nota

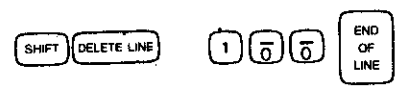
La parola chiave DELETE LINE può essere introdotta premendo il tasto [DELETE LINE RUN] insieme con [SHIFT].

Esempi

1. Si cancellino le linee di un testo, presente in memoria principale, dal numero di linea 50 al numero di linea 150.

   Premere 

2. Si cancelli la linea 100 del file testo presente in memoria principale.

   Premere 

Comando EXEC

Funzione

Carica in memoria ed esegue un programma di utilità.

Formato

EXE [C] utility [,parameter [,parameter] ...]

dove:

utility è il nome di un programma di utilità

parameter specifica un operando associato con il programma di utilità specificato.

Azione

Il comando comunica al sistema di caricare in memoria principale ed eseguire il programma di utilità di nome utility.

Nota

Nella libreria di sistema sono disponibili i seguenti programmi utilità:

ASSEMBLY
FDCOPY
FLCOPY
LBCREATE
LBPROTECT
LIBCOPY
LNK

Per ulteriori informazioni si veda l'appendice A.

Comando EXQ

Funzione

Comanda l'esecuzione di un programma

Formato

EXQ [filename [,W = filename1] [,D]]

dove:

filename è il nome di un programma, in formato oggetto eseguibile, presente su floppy disk

W = filename1 è l'operando che comunica al sistema il nome del file di lavoro su floppy disk (filename1) che si vuole utilizzare, nel corso della elaborazione

D è l'operando che comanda l'entrata in stato debugging sulla prima istruzione del programma.

Azione

Il comando, con filename come operando, comunica al sistema di caricare in memoria principale ed eseguire il programma di nome filename; gli oggetti eseguibili sono solamente contenuti su file di tipo E, ottenuti come output del Linker.

Il comando, con filename e W=filename1 come operandi, comunica al sistema di eseguire il programma di nome filename e di rendere disponibile al programma in esecuzione il file di lavoro su floppy disk di nome filename1

Il comando, con filename, W=filename1 e D come operandi, comunica al sistema di eseguire il programma di nome filename con un file di lavoro su floppy disk di nome filename1 e di entrare in debugging sulla prima istruzione del programma.

Il comando, senza operandi, comunica al sistema di

rieseguire il programma presente in memoria; un programma eseguibile può trovarsi in memoria solo a fronte di un precedente comando EXQ con operando flname.

Al termine dell'esecuzione di un programma, si stabilisce uno stato di macchina che accetta solo i seguenti comandi:

- EXQ — per rieseguire il programma
- EXQ (.....) — per eseguire un altro programma
- OLD/TEXT — per ripristinare la memoria utente

La memoria disponibile per l'utente, durante l'esecuzione, vale esattamente l'estensione dichiarata: 16, 24, 32, 40, 48 K byte a meno delle opzioni presenti.

Errori: Gli errori che si possono verificare sono i seguenti:

- file di lavoro su floppy disk assente

- area libera su disco insufficiente per poter rendere il programma rieseguibile

- opzione inesistente (specifica dell'argomento D e routine di debugging non caricate in memoria)

Comando FETCH

Funzione

Trasferisce nel buffer di tastiera una linea di file testo presente in\memoria principale.

Formato

FET [CH] [line-num]

dove:

line-num è il numero di linea della linea da trasferire

Azione

Il comando comunica al sistema di trasferire nel buffer di tastiera e visualizzare sul display la linea del file testo presente in memoria principale con numero line-num.

Il comando, senza operandi, comunica al sistema di trasferire nel buffer di tastiera e di visualizzare sul display una delle seguenti linee:

- l'ultima linea corretta introdotta da tastera
- l'ultima linea di un file caricato in memoria principale mediante il comando OLD
- l'ultima linea stampata con il comando LIST
- l'ultima linea richiamata nel buffer di tastera con il comando FETCH o con i tasti ↑ e ↓ .

Note

1. Se si introduce il comando FETCH con un operando line-num il cui valore non è presente in memoria principale, la linea con il numero di linea immediatamente inferiore a quella specificata viene trasferita nel buffer di tastiera. Se line-num è inferiore al più piccolo numero di linea presente

in memoria principale, nel buffer di tastiera viene trasferita la linea con il più grande numero di linea.

2. La linea visualizzata sul display può non avere lo stesso formato di quella introdotta da tastiera, perchè il sistema la modifica prima di trasferirla nel buffer di tastiera; così, ad esempio, quando si introduce la istruzione: 10* il comando FETCH la visualizzerà sul display come: 0010*. Se l'editing effettuato dal sistema produce una linea con più di 80 caratteri, la linea viene compattata eliminando tutti gli spazi.

3. Se l'eliminazione degli spazi non significativi non riporta la linea ad 80 caratteri viene visualizzato un messaggio di errore i la linea è stampata sulla stampante integrata. Premendo il tasto console RECALL la linea appare sul display (nel buffer di tastiera è troncata ai primi 80 caratteri) e si può editare; quando è introdotta nuovamente, premendo il tasto END OF LINE, sostituisce la linea precedente che era presente, integralmente, nel working file.

4. La parola chiave può essere introdottapremendo il tasto FETCH NEW con SHIFT .

Esempi

1. Trasferire nel buffer di tastiera la linea 50 di un testo presente in memoria principale.

   Premere F E T 5 0 END OF LINE

2. Trasferire nel buffer di tastiera l'ultima linea di un file testo caricato in memoria principale con il comando OLD.

   Premere SHIFT FETCH END OF LINE

LDKEYS

Comando LDKEYS

Funzione

Riassegna ai tasti funzione il contenuto assegnato precedentemente ed attualmente presente sul floppy disk sistema.

Formato

LDK [EYS]

Azione

Il comando comunica al sistema di riassegnare ad ogni tasto funzione la stringa di caratteri che ad esso era già stata assegnata in precedenza ed era stata successivamente registrata sul floppy disk sistema da un comando STKEYS.

Note

1. Il comando è usato perchè il contenuto dei tast funzione può essere modificato durante lo stato esecuzione calcoli immediati (vedi il capitolo 8).

2. Il contenuto presente su floppy disk sistema è riassegnato ai tasti funzione ogni volta che il sistema è inizializzato, quindi: (1) quando il sistema è acceso, (2) quando si esegue il comando OPTIONS e (3) quando si esegue il comando CONFIGURE.

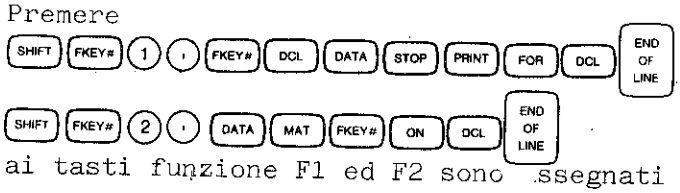
Esempio

Si registri sul floppy disk sistema un contenuto per i tasti funzione F1 ed F2. Dopo aver assegnato ai medesimi tasti funzione un contenuto diverso si ripristini il precedente.

Premere CALC MODE

Il sistema è nello stato esecuzione di calcoli immediati.

Premere



ai tasti funzione F1 ed F2 sono assegnati i caratteri a destra della virgola.

Premere CALC MODE

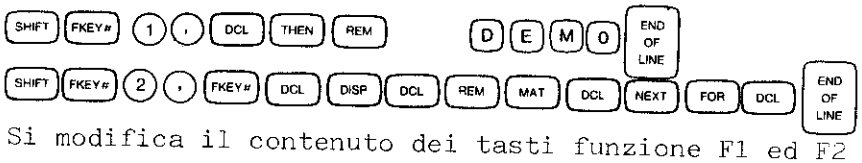
Il sistema è nello stato comandi.

Premere S T K END OF LINE

La suddetta associazione fra tasti funzione e relativo contenuto è registrata sul floppy disk sistema.

Premere CALC MODE

Premere



Si modifica il contenuto dei tasti funzione F1 ed F2

Premere CALC MODE

Il sistema è di nuovo nello stato comandi.

Premendo F1 sul display si vede: EXQ DEMO
Premendo F2 sul display si vede: RESEQUENCE

Premere L D K END OF LINE

Premendo F1 sul display si vede: REPLACE
Premendo F2 sul display si vede: PURGE

Come si vede è stato ripristinato il contenuto dei tasti funzione che era stato registrato sul floppy disk sistema.

Comando LIST

Funzione

Stampa una o più linee di un file testo presente in memoria principale.

Formato

$$\text{LIS[T] [line-num}_1\text{]} \left[,\left\{ \begin{matrix} \text{[line-num}_2\text{]} \\ \text{line-num}_2 \end{matrix} \right. \begin{matrix} \\ \left. \right\} \end{matrix} \right] \begin{matrix} \\ \text{,X} \end{matrix} \Bigg]$$

dove:

line-num$_1$ indica la linea da stampare o la prima linea di un insieme di linee da stampare

line-num$_2$ indica l'ultima linea di un insieme di linee da stampare

X indica che non devono essere stampati i numeri di linea di un file testo.

Azione

Il comando comunica al sistema di stampare le linee del file testo presente in memoria principale il cui numero di linea è compreso tra il numero di linea indicato con il primo operando e quello indicato con il secondo operando (estremi inclusi).

Il comando, privo degli ultimi due operandi, comunica al sistema di stampare le linee del file testo presente in memoria principale iniziando dalla linea il cui numero di linea è specificato con il primo operando.

Il comando, privo del primo operando, comunica al sistema di stampare le linee il cui numero di linea è inferiore o uguale a quello specificato.

Il comando, senza operandi, comunica al sistema si stampare tutte le linee del file testo presente in memoria principale.

Con l'operando X specificato il comando comunica al sistema di stampare le linee specificate del file testo presente in memoria principale, senza il numero di linea.

Note

1. La parola chiave del comando LIST può essere introdotta premendo il tasto [LIST].

2. Se la configurazione di sistema installata è priva di stampante integrata e di stampante ausiliaria (vedi comando CONFIGURE), le linee di programma o di file testo sono visualizzate sul display. In questo caso per leggere una linea completa si devono utilizzare i tasti [→], [REPEAT] e [SHIFT] come spiegato nel capitolo 1 § "Tastiera". Per leggere ogni linea si deve premere il tasto [CONTINUE]: ad ogni pressione appare sul display una diversa linea e quando alla pressione di [CONTINUE] il sistema produce un segnale acustico significa che non vi sono più linee da visualizzare.

Esempio

Si richieda la stampa delle linee di un file testo presente in memoria principale. Non si stampino i numeri di linea.

Premere [LIST] [,] [.] [X] [END OF LINE]

MODIFY

Comando MODIFY

Funzione

Modifica il nome di un file e/o il numero di byte riservati ad un file di lavoro su floppy disk.

Formato

MOD [IFY] old-filename, $\left\{ \begin{matrix} \textbf{new-filename} \text{ ,[n]} \\ \textbf{n} \end{matrix} \right\}$

dove:

old-filename indica il nome di un file presente su floppy disk

new-filename indica il nuovo nome da dare al file presente su floppy disk

n è il numero compreso tra 1 e 237130 che indica il numero di byte da riallocare su floppy disk per il file specificato.

Azione

Il comando, completo dei tre operandi, comunica al sistema di sostituire al file di lavoro su floppy disk con nome old-filename il nome di new-filename e di riservare ad esso n byte su floppy disk.

Il comando, con old-filename e new-filename come operndi, comunica al sistema di sostituire al file di lavoro su floppy disk con nome old-filename il nome new-filename.

Il comando, con n come secondo operando, comunica al sistema di riservare n byte al file di lavoro su floppy disk con nome old-filename.

Note

1. L'operando new-filename può esser costituito solamente da un massimo di 6 caratteri alfanumerici di cui il primo alfabetico.

2. Su uno stesso floppy disk non possono esistere file con lo stesso nome.

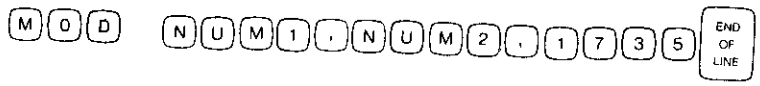
3. Non si può modificare il nome di un file della sottolibreria package o della sottolibreria comune dopo che sono state protette con il programma di utilità LBPROTECT.

4. Se n non è multiplo di 128 il sistema rialloca per il file specificato un numero di byte pari al successivo multiplo di 128.

Esempio

Modificare il nome del file di lavoro su floppy disk NUM1 in NUM2 e riallocare 1735 byte per esso.

Premere



Comando OLD

Funzione

Carica in memoria principale un file testo presente su floppy disk.

Formato

OLD filename

dove:

filename è il nome del file testo da caricare in memoria principale.

Azione

Il comando comunica al sistema di caricare in memoria principale il file testo registrato su floppy disk col nome specificato dall'operando.

Note

1. Se vi sono due floppy disk nel sistema la ricerca del file da caricare in memoria principale inizia dal floppy disk sistema. Se, quindi, si hanno due floppy disk aventi due file con lo stesso nome, si deve cambiare il floppy disk sistema con un altro che non abbia un file con lo stesso nome se si vuole caricare in memoria principale il file presente sul floppy disk utente.

2. La parola chiave del comando può essere introdotta premendo i tasti [SHIFT] e [OLD] contemporaneamente.

Esempio

Caricare in memoria principale il file testo +MAT, presente nella sottolibreria comune.

Premere [SHIFT] [OLD] [+] [M] [A] [T] [END OF LINE]

OPTIONS

Comando OPTIONS

Funzione

Definisce un modulo opzionale del sistema.

Formato

OPT [IONS] [DEB]

Azione

Il comunica al sistema di inizializzarsi in funzione degli operandi specificati.

L'operando DEB specifica che il sistema deve inizializzarsi in modo che si possano eseguire programmi sotto il controllo del modulo di debugging.

Il comando, senza operandi, comunica al sistema di cancellare in memoria utente le routine del sistema operativo relative al modulo DEB.

Note

1. La configurazione di sistema richiesta ogni volta che si esegue un comando OPTIONS è registrata sul floppy disk sistema, così che ogni volta che il sistema viene acceso si configura nel modo richiesto con l'ultimo comando OPTIONS eseguito.

2. Quando si esegue il comando OPTIONS, il precedente contenuto della memoria principale è cancellato. Se si vuole registrare il contenuto della memoria utente, prima di introdurre il comando OPTIONS si introduca il comando SAVE od il comando REPLACE.

3. E' possibile introdurre un programma che impiega le periferiche seriali, anche se la configurazione del sistema non ne permette l'esecuzione. Tale programma può essere registrato su floppy disk con i comandi SAVE o REPLACE ed eseguito dopo aver introdotto il comando OPTIONS con le opzioni necessarie.

Esempi

1. Reinizializzare il sistema per poter eseguire programmi sotto il controllo del debugging.

   Premere O P T D E B END OF LINE

2. Cancellare dalla memoria utente le routine del sistema operativo che impiegano periferiche seriali.

   Premere O P T END OF LINE

PURGE

Comando PURGE

Funzione

Cancella un file da una sottolibreria su floppy disk.

Formato

PUR [GE] filename

dove:

filename è il nome di un file da cancellare.

Azione

Il comando comunica al sistema di cancellare il file specificato con l'operando, dalla sottolibreria a cui appartiene su floppy disk.

Note

1. Se ci sono due floppy disk nell'unità, con due file che hanno lo stesso nome, il comando PURGE cancella il file presente sul floppy disk sistema.

2. Il comando non è accettato con i file che appartengono alla sottolibreria package ed alla sottolibreria comune, se queste sono state protette con il programma di utilità LBPROTECT.

Esempio

Si cancelli il file testo MAT dalla sottolibreria utente presente su floppy disk utente.

Premere [C][A][T] [.] [.][T][END OF LINE]

dal listing stampato dal sistema si verifica se sul disco sistema vi è un file di nome MAT. In caso affermativo si cambia il floppy disk sistema con un altro che non abbia nella sottolibreria utente un file con lo stesso nome, quindi si prema:

[P][U][R] [M][A][T][END OF LINE]

REPLACE

Comando REPLACE

Funzione

Sostituisce, su floppy disk, un file testo con un altro avente lo stesso nome.

Formato

REP [LACE]

Azione

Il comando comunica al sistema di registrare su floppy disk il file testo presente in memoria principale al posto del file testo già registrato sul floppy disk con lo stesso nome.

Note

1. Non si può utilizzare il comando REPLACE per registrare su floppy disk un file testo appena introdotto da tastiera, quindi senza nome. In questo caso si introduca il comando PURGE per cancellare il file che si vuole sostituire e quindi il comando SAVE per registrare il nuovo file.

2. Se vi sono due floppy disk, sul sistema, con due file con lo stesso nome, il comando REPLACE sostituisce il file presente sul floppy disk sistema.

Esempio

Si aggiorni il file testo EQUAZ presente sul floppy disk utente.

Premere [SHIFT] [OLD] [E] [Q] [U] [A] [Z] [END OF LINE]

Il file testo è trasferito in memoria principale.

Premere [LIST] [END OF LINE]

E' stampato il listing del file testo. Si introducano le modifiche desiderate.

Premere 

Il nuovo file testo EQUAZ è registrato sul floppy disk utente.

RESEQUENCE

Comando RESEQUENCE

Funzione

Modifica la numerazione delle linee del file testo presente in memoria principale.

Formato

RES [EQUENCE] [line-num] [,increment]

dove:

line-num indica il primo numero di linea da assegnare al file testo presente in memcria principale

increment è un numero intero positivo che indica il valore che è aggiunto ad ogni numero di linea per ottenere il numero di linea successivo.

Azione

Il comando comunica al sistema di assegnare alla prima linea del file testo presente in memoria principale il numero specificato con il primo operando e di aggiungere ad ogni numero di linea l'incremento specificato con il secondo operando, per assegnare il successivo numero di linea.

Il comando, con solamente il primo operando, comunica al sistema di assegnare alla prima linea del file testo, il numero specificato e di aggiungere 10 ad ogni numero di linea per assegnare il numero di linea successivo.

Il comando, con solamente il secondo operando, comunica al sistema di assegnare il numero increment alla prima linea del file testo e di aggiungere ad ogni numero di linea l'incremento specificato con il secondo operando per assegnare il successivo numero di linea.

Il comando, privo di operandi, comunica al sistema di

assegnare il numero 10 alla prima linea del file testo e di aggiungere 10 ad ogni numero di linea per assegnare il successivo numero di linea.

Esempio

Si rinumeri il seguente programma presente in memoria principale come file testo.

```
0007              CSECT
0009 CQTEST       PROC      PROLOG=NO
0030              BALR      10,0
0040              USING      *,10
0050 INIZ         LA         14,4096(10)
0060              LA         14,1(14)
0070              LA         3,IND
0080              LH         8,0(3)
0090              B          OUT
0100 NCONF        DC         F'0'
0110 IND          DS          0F
0120 L            DC          F'16'
0130 AREA1        DC         CL16'ABCDEFGHILMNOPQR'
0140 FLAG         DS         F
0150 OUT             CALEXS PRINT,(AREA1,L,FLAG)
0160              BR     14
0170              END
```

Premere R E S END OF LINE

Premere L I S END OF LINE

```
0010              CSECT
0020 CQTEST       PROC      PROLOG=NO
0030              BALR      10,0
0040              USING      *,10
0050 INIZ         LA         14,4096(10)
0060              LA         14,1(14)
0070              LA         3,IND
0080              LH         8,0(3)
0090              B          OUT
0100 NCONF        DC         F'0'
0110 IND          DS          0F
0120 L            DC          F'16'
0130 AREA1        DC         CL16'ABCDEFGHILMNOPQR'
0140 FLAG         DS         F
0150 OUT             CALEXS PRINT,(AREA1,L,FLAG)
0160              BR     14
0170              END
```

Comando SAVE

Funzione

Registra un file testo, presente in memoria principale, in una libreria su floppy disk.

Formato

SAV [E] [S/U], filename

dove:

S indica il floppy disk sistema

U indica il floppy disk utente

filename indica il nome con cui il file testo sarà registrato sul floppy disk.

Azione

Il comando comunica al sistema di registrare, con il nome filename, sul floppy disk specificato con il primo operando, il file testo presente in memoria principale.

Note

1. Il comando SAVE non può essere usato per registrare un file testo in una sottolibreria package che sia stata protetta mediante l'esecuzione del programma di utilità LBPROTECT.

2. Non si possono registrare su uno stesso floppy disk più file con lo stesso nome.

3. La parola chiave del comando può essere introdotta premendo [SHIFT] con [SAVE AUTO#].

Esempio

Si registri il file testo, presente in memoria principale, nella sottolibreria comune sul floppy disk sistema.

Premere [SHIFT] [SAVE] [.] [+] [C] [O] [M] [END OF LINE]

Comando SHIFT

Funzione

Modifica la numerazione delle linee di un file testo presente in memoria principale, iniziando dalla linea specificata.

Formato

SHI [FT] line-num, increment

dove:

line-num è un numero di linea che specifica da quale linea deve essere modificata la numerazione delle linee

increment è un numero intero positivo che specifica il valore di cui deve essere incrementato ogni numero di linea da modificare.

Azione

Il comando comunica al sistema di modificare i numeri di linea delle linee del file testo, presente in memoria principale, iniziando dalla linea con numero di linea line-num. I nuovi numeri di linea sono ottenuti aggiungendo il valore specificato con increment ai precedenti numeri di linea.

Esempio

Vediamo come i numeri di linea, dal 9 in poi, sono incrementati di 100 nella routine sottostante.

```
LIST
FILE     PROV1

0005             CSECT
0009 CQTEST      PROC      PROLOG=NO
0030             BALR      10,0
0040             USING      *,10
0050 INIZ        LA         14,4096(10)
0160              BR     14
0170              END

END OF LISTING


SHI 9,100
LIST
FILE     PROV1

0005             CSECT
0109 CQTEST      PROC      PROLOG=NO
0130             BALR      10,0
0140             USING      *,10
0150 INIZ        LA         14,4096(10)
0260              BR     14
0270              END

END OF LISTING
```

SPACE

Comando SPACE

Funzione

Stampa lo spazio disponibile (in byte) per registrazioni successive su floppy disk.

Formato

SPA [CE]

Azione

Il comando comunica al sistema di stampare lo spazio libero da registrazioni sul floppy disk presente nell'unità. Se si hanno due floppy disk nell'unità, viene stampato lo spazio disponibile per entrambi.

Nota

Le informazioni suddette sono anche visualizzate sul display.

Esempio

Si richieda la stampa dello spazio libero sui due floppy disk presenti nel sistema.

Premere S P A END OF LINE

Comando STKEYS

Funzione

Registra su floppy disk sistema il contenuto dei tasti funzione.

Formato

STK [EYS]

Azione

Il comando comunica al sistema di registrare sul floppy disk sistema l'attuale contenuto dei tasti funzione.

Nota

Ogni qual volta il sistema è inizializzato (dopo la accensione, eseguendo il comando CONFIGURE o il comando OPTICNS) o è eseguito il comando LDKEYS, ai tasti funzione è assegnato il contenuto registrato sul floppy disk sistema.

Comando TEXT

Funzione

Permette di introdurre un file testo, linea per linea, da tastiera.

Formato

TEX [T]

Azione

Il comando comunica al sistema che quanto verrà successivamente introdotto sono linee di un file testo. I file testo sono normalmente usati per contenere i programmi Assembler in formato sorgente.

Note

1. Quando il comando è eseguito il precedente contenuto della memoria utente è cancellato.

2. Utilizzando il comando AUTO si possono numerare automaticamente le linee del testo introdotte successivamente

3. Per la defirizione di file testo si veda il capitolo 2.

4. IL LINGUAGGIO ASSEMBLER

Introduzione

Un programma scritto in Assembler non può essere immediatamente eseguito. Ha bisogno della preventiva traduzione delle sue istruzioni dal linguaggio simbolico, utilizzato dal programmatore, al linguaggio assoluto. Questa traduzione viene chiamata "Assemblaggio" e viene svolta da un programma chiamato Assemblatore, il quale fà parte del sistema operativo P6Ø6Ø.

L'Assemblatore fornisce all'utente la possibilità di assemblare un modulo sorgente ottenendo in output un un modulo rilocabile (cioè che può essere assegnato in una zona qualunque di memoria).

Istruzioni Assembler

Nel linguaggio Assembler si distinguono essenzialmente 4 tipi di frasi:

- istruzioni esecutive
- istruzioni direttive per l'Assemblatore
- macro-istruzioni
- frasi commento

Istruzioni esecutive

Le istruzioni esecutive sono la rappresentazione simbolica "uno a uno" delle istruzioni macchina. A fronte di ogni frase esecutiva del programma sorgente, l'Assemblatore provvede a generare una istruzione macchina oggetto; sono definiti codici mnemonici per tutte le istruzioni macchina. Le istruzioni esecutive si dividono logicamente nelle seguenti aree funzionali:

- istruzioni standard
- istruzioni speciali

Le istruzioni standard sono elencate nella seguente tabella

MNEMONICO	DESCRIZIONE
A	ADD
AH	ADD HALFWORD
AL	ADD LOGICAL
ALM	ADD LOGICAL MEMORY
ALR	ADD LOGICAL REGISTER
ALRI	ADD LOGICAL REGISTER IMMEDIATE
AM	ADD MEMORY
AR	ADD REGISTER
B	BRANCH
BAL	BRANCH AND LINK
BALR	BRANCH AND LINK REGISTER
BC	BRANCH ON CONDITION
BCR	BRANCH ON CONDITION REGISTER
BCT	BRANCH ON COUNT
BCTR	BRANCH ON COUNT REGISTER
BE	BRANCH ON EQUAL
BH	BRANCH ON HIGH
BL	BRANCH ON LOW
BM	BRANCH ON MINUS
BNE	BRANCH ON NOT EQUAL
BNH	BRANCH ON NOT HIGH
BNL	BRANCH ON NOT LOW
BNM	BRANCH ON NOT MINUS
BNO	BRANCH IF NOT ONES
BNP	BRANCH ON NOT PLUS
BNZ	BRANCH ON NOT ZERO
BO	BRANCH ON OVERFLOW
BP	BRANCH ON PLUS
BR	BRANCH
BXH	BRANCH ON INDEX HIGH
BXLE	BRANCH ON INDEX LOW OR EQUAL
BZ	BRANCH ON ZERO
C	COMPARE
CH	COMPARE HALFWORD
CL	COMPARE LOGICAL
CLC	COMPARE LOGICAL CHARACTER
CLI	COMPARE LOGICAL IMMEDIATE
CLM	COMPARE LOGICAL MEMORY
CLR	COMPARE LOGICAL REGISTER
CM	COMPARE MEMORY
CR	COMPARE REGISTER
DM	DIVIDE MEMORY
EX	EXECUTE
IC	INSERT CHARACTER
IM	IMMEDIATE IN MEMORY

ISO	ISO TEST
L	LOAD
LA	LOAD ADDRESS
LC	LOAD COMPLEMENT
LCR	LOAD COMPLEMENT REGISTER
LH	LOAD HALFWORD
LM	LOAD MULTIPLE
LN	LOAD NEGATIVE
LNR	LOAD NEGATIVE REGISTER
LPR	LOAD POSITIVE REGISTER
LR	LOAD REGISTER
LT	LOAD AND TEST
LTR	LOAD AND TEST REGISTER
MLH	MULTIPLY LOGICAL HALFWORD
MLR	MULTIPLY LOGICAL REGISTER
MVC	MOVE CHARACTER
MVCR	MOVE CHARACTER ON REGISTER
MVI	MOVE IMMEDIATE
MVN	MOVE NUMERICS
MVO	MOVE WITH OFFSET
MVZ	MOVE ZONES
N	AND
NC	AND CHARACTER
NI	AND IMMEDIATE
NOP	NO OPERATION
NOPR	NO OPERATION
NR	AND REGISTER
O	OR
OC	OR CHARACTER
OI	OR IMMEDIATE
OR	OR REGISTER
S	SUBTRACT
SH	SUBTRACT HALFWORD
SL	SUBTRACT LOGICAL
SLA	SHIFT LEFT ALGEBRAIC
SLL	SHIFT LEFT LOGICAL
SLM	SUBTRACT LOGICAL MEMORY
SLR	SUBTRACT LOGICAL REGISTER
SLRI	SUBTRACT LOGICAL REGISTER IMMEDIATE
SM	SUBTRACT MEMORY
SR	SUBTRACT REGISTER
SRA	SHIFT RIGHT ALGEBRAIC
SRL	SHIFT RIGHT LOGICAL
ST	STORE
STC	STORE CHARACTER
STH	STORE HALFWORD
STM	STORE MULTIPLE
TM	TEST UNDER MASK

TR	TRASLATE
TRT	TRANSLATE AND TEST
X	EXCLUSIVE OR
XC	EXCLUSIVE OR CHARACTER
XI	EXCLUSIVE OR IMMEDIATE
XR	EXCLUSIVE OR REGISTER

Le istruzioni speciali sono suddivise ulteriormente nelle sottoclassi elencate di seguito.

Istruzioni per operazioni di gestione dello stack

MNEMONICO	DESCRIZIONE
ASA	ALLOCATE STACK AREA
FSA	FREE STACK AREA

Istruzioni per operazioni di conversione

MNEMONICO	DESCRIZIONE
CBS	BINARY TO ISO CONVERSION
CSBH	ISO TO BINARY CONVERSION

Istruzioni per operazioni di ricerca

MNEMONICO	DESCRIZIONE
DIS	DICOTOMIC SEARCH
LIE	LOOK FOR IMMEDIATE EQUAL
LINE	LOOK FOR IMMEDIATE NOT EQUAL
SES	SEQUENTIAL SEARCH
SESM	SEQUENTIAL SEARCH WITH MASK

Istruzioni per operazioni di gestione dinamica della memoria

MNEMONICO	DESCRIZIONE
ACT	ACTIVE MODULE
CALEXS	EXTERNAL SYSTEM MODULE
CALEXT	CALL EXTERNAL
RETEXT	RETURN FROM EXTERNAL MODULE
RLSEM	RELEASE MODULE

Istruzioni di richiamo subroutine

MNEMONICO	DESCRIZIONE
CALL	SUBROUTINE CALL
RETS	SUBROUTINE RETURN

Terminazione programmi

MNEMONICO	DESCRIZIONE
SVC (1)	SUPERVISOR CALL

Istruzioni direttive per l'Assembler

Le istruzioni direttive per l'Assemblatore specificano delle particolari funzioni ausiliarie da compiersi a cura del programma Assembler stesso. In questo contesto si distinguono le seguenti frasi:

- Frasi per l'indirizzamento del programma: Nel sistema P6Ø6Ø l'indirizzamento è effettuato tramite registro base (che contiene l'indirizzo base), un eventuale registro indice e un valore di spiazzamento (displacement), che sommato all'eventuale registro indice e al contenuto del registro base permette di ottenere l'indirizzo effettivo di memoria. L'indirizzamento

può essere di due tipi:

- esplicito
- implicito

Con il termine indirizzamento esplicito viene definito l'indirizzo tramite il registro base e il valore di spiazzamento. Con il termine indirizzamento implicito viene definito l'indirizzo mediante simboli (per maggiori informazioni vedere il capitolo 5). Di questo tipo di frase fanno parte le seguenti istruzioni:

USING

DROP

Frasi di controllo del programma: Un programma o modulo sorgente è formato da una sezione di controllo (Control Section), e eventualmente da una o più sezioni fittizie (Dummy Section).

1. Una sezione di controllo è un insieme di codice che al momento del caricamento in memoria per l'esecuzione può essere rilocato. L'Assemblatore assegna alla sezione di controllo un Location Counter (vedi pagina 4-18)il cui valore iniziale è zero poichè l'allocazione del modulo non è nota a priori all'Assembler ma viene definita dopo la fase di Linking e caricamento dello stesso in memoria; nel caso di istruzione START, può essere indicato un termine che specifica il valore iniziale da assegnare al Location Counter (valore significativo solo per il listing). Alla fine della sezione di controllo l'Assemblatore accoda la tavola contenente i riferimenti ai moduli esterni.

2. Una Dummy Section rappresenta una Control Section che viene assemblata ma non fà parte del programma oggetto. Essa è utilizzata per descrivere la struttura di un area di memoria, senza riservare realmente alcuna area. Più di una Dummy Section può essere definita in un assemblaggio, ma ciascuna di esse deve essere nominata (vedi DSECT). Si usa la Dummy Section per descrivere il formato di un'area di memoria, la cui locazione non è ancora determinata al momento dell'assemblaggio. L'effettivo indirizzamento di una Dummy Section avviene tramite l'istruzione USING:

Esempio:

```
          ...
          ...
          ...
          USING     DUM, 6
          ...
          ...
          ...
DUM       DSECT     CL3Ø
NAME      DS        CL5
COD       DS        CL4
IMPOR     DS
INIZ      CSECT
```

Al momento dell'esecuzione del programma, il formato dell'area definita nella DSECT con i relativi simboli, coincide con una reale area di memoria, caricando nel registro 6, specificato nella USING (vedi esempio precedente) l'indirizzo di quest'area. Ad ogni Dummy Section è allocato un Location Counter il cui valore iniziale è zero, e che viene incrementato ad ogni elemento della sezione. Di questo tipo di frase fanno parte le seguenti istruzioni:

CSECT

START

DSECT

ORG

END

Frasi di definizione di dati ed aree: Queste istruzioni non generano istruzioni macchina, ma riservano aree di memoria che al tempo dell'esecuzione conterranno costanti ed altri dati. Di questo tipo di frase fanno parte le seguenti istruzioni:

DC

DS

Frasi di definizione dei nomi: Queste istruzioni permettono di definire dei simboli. Di questo tipo di frase fanno parte le seguenti istruzioni:

EQU

EXT

Frasi di controllo listing: Queste istruzioni permettono di indicare il formato per il listing. Di questo tipo di frase fanno parte le seguenti istruzioni:

TITLE

EJECT

SPACE

PRINT

Le macro-istruzioni

Le macro-istruzioni comandano all'Assemblatore di inserire particolari sequenze di istruzioni macchina oggetto in base alle informazioni contenute nella stessa macro-istruzione. Le macro-istruzioni del sistema Assembler P6Ø6Ø sono:

LAEXT

PROC

PRRET

Formato sorgente

Un programma sorgente scritto in linguaggio Assembler P6Ø6Ø è costituito da una sequenza di frasi sorgente facenti parte di un file di tipo testo. Ogni frase sorgente può essere costituita da una o più linee. Si distingue quindi:

- la struttura della linea quale input fisico del programma
- la struttura della frase quale input logico del programma.

Struttura della linea

Le linee hanno lunghezza variabile, lunghezza che comunque non può superare gli 8Ø caratteri. Ogni linea è composta da 3 sottocampi (figura 4-1):

- campo sequenza
- campo continuazione
- campo frase sorgente

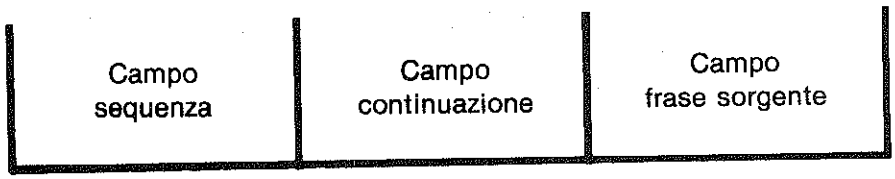


Figura 4-1 Struttura della linea

Campo sequenza: E' il primo campo della linea e contiene su 4 crt. il numero di linea, ordinato in senso ascendente e compreso tra ØØØ1 e 9999.

Campo continuazione: E' costituito da un carattere immediatamente successivo al campo sequenza e può assumere i seguenti valori:

- blank, quando lo statement non continua su linee successive

- un qualsiasi carattere alfabetico, quando lo statement continua su linee successive. Riguardo al campo continuazione l'utente deve soddisfare precise regole:

  . quando la frase sorgente non può essere contenuta completamente su di una linea, deve essere posto un qualsiasi carattere alfabetico nel campo continuazione

  . la continuazione della frase sorgente deve quindi essere scritta sulla linea successiva a partire da colonna 16. I primi 15 caratteri devono essere blank (vedi figura 4-2)

  . è ammesso un numero massimo di 2 linee di continuazione

  . il nome, se presente, e il codice operativo de-

vono comunque essere scritti sulla prima linea della frase sorgente

- non è ammessa la continuazione del campo commento
- non si possono porre delle frasi commento tra le linee di continuazione di una istruzione sorgente.

Campo frase sorgente: Questo campo è immediatamente successivo al campo continuazione e contiene la frase sorgente vera e propria.

Struttura della frase sorgente

Si distinguono 2 tipi di frasi sorgente:

- istruzioni
- frasi commento

Le istruzioni sono composte da 4 campi:

- campo nome
- campo codice operativo
- campo operandi
- campo commento

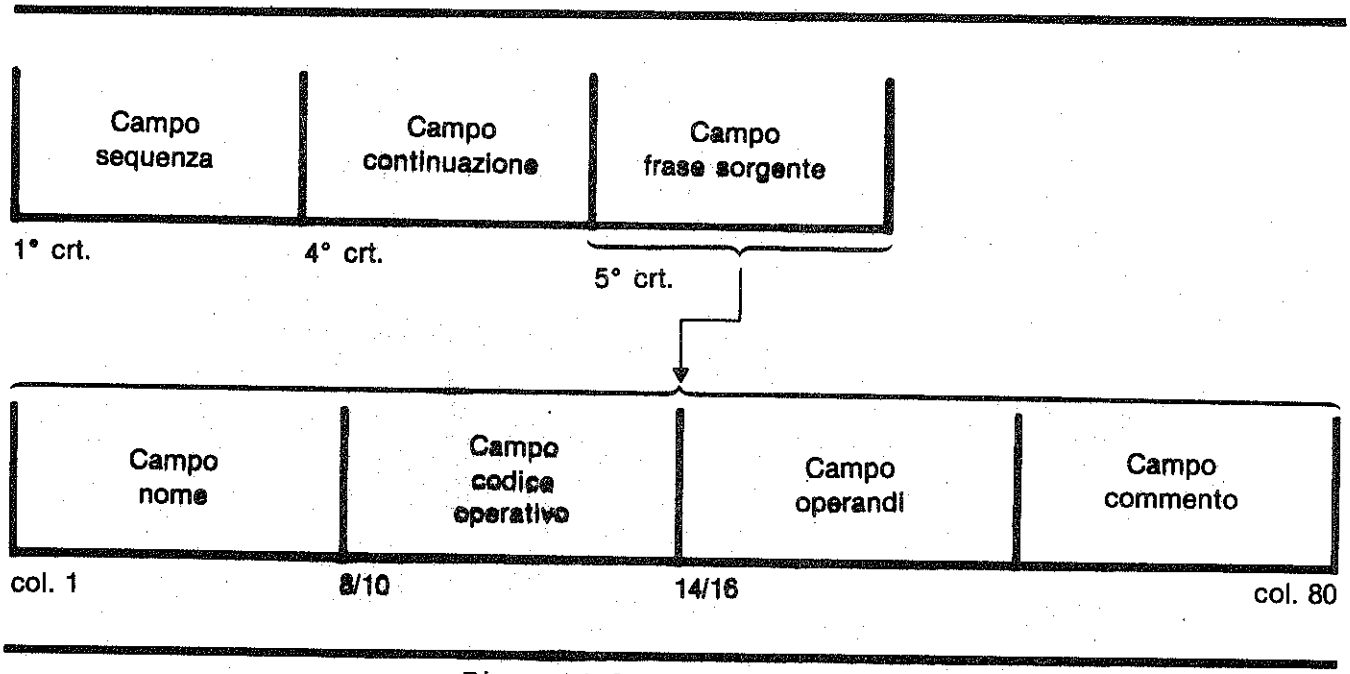


Figura 4-2 Struttura della frase

Il quinto carattere della linea è intesa come prima colonna della frase. Ognuno di questi campi deve essere scritto nell'ordine indicato in figura 4-2 e de-

ve essere separato da almeno un blank. L'unico campo obbligatorio è il campo operazione, gli altri campi sono obbligatori o facoltativi a seconda del tipo di istruzione usata dall'utente.

Campo nome: Il campo nome è usato per attribuire un nome simbolico alle frasi sorgenti. Deve sempre iniziare a colonna 1 ed è costituito al max di 8 crt. alfanumerici, il primo dei quali deve essere alfabetico. Il nome è opzionale, a meno delle istruzioni DSECT e EQU; se il primo crt. della frase sorgente è blank, l'Assembler assume tale frase come priva di nome.

Campo operazione: Questo campo contiene il codice operativo e deve comparire obbligatoriamente in tutte le istruzioni. Può avere uno dei seguenti significati:

a) Codice mnemonico di operazione macchina
b) Codice di una istruzione direttiva per l'Assemblatore
c) Codice di una macro-istruzione

Ha una lunghezza massima di 6 crt.

Campo operandi: Questo campo contiene gli operandi della istruzione. Tali operandi sono informazioni che identificano e/o descrivono:

- aree di memoria
- maschere
- lunghezza di aree o tipi di dati
- informazioni supplementari per l'Assembler.

In relazione al tipo di istruzione identificato tramite il codice operativo, possono o devono comparire o più operandi separati dalla virgola.

Campo commento: Questo campo è opzionale e contiene delle note che l'utente desidera includere nel tabulato emesso dal programma Assemblatore. Nel caso di istruzioni aventi il campo operandi opzionale, quando tale campo è omesso l'eventuale commento deve essere preceduto dalla virgola, a sua volta preceduta e seguita da uno o più blank.

Le frasi commento sono un ulteriore tipo di frasi sorgente, identificate dal crt. "*" posto in colon-

na 1. Queste frasi non generano alcuna azione particolare da parte dell'Assemblatore, ma sono semplicemente stampate nel tabulato in output del programma. Queste frasi possono contenere qualsiasi tipo di carattere accettato dall'Assemblatore. E' ammessa al massimo la continuazione di 2 linee e l'utente dovrà sottostare alla regola che impone di porre un qualsiasi carattere alfabetico nel campo continuazione e di scrivere sulla linea successiva solo oltre la colonna 15.

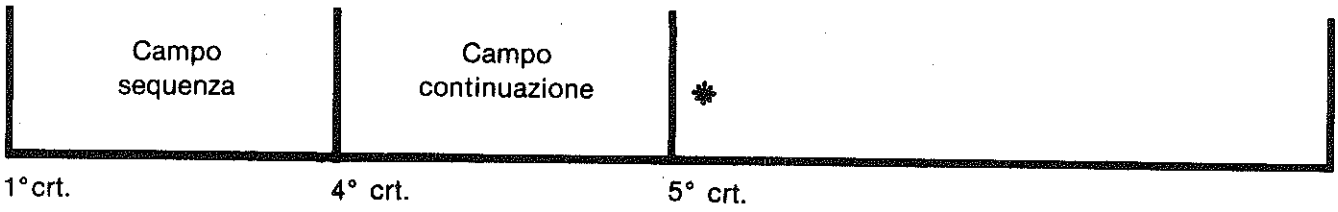


Figura 4-3 Struttura della frase commento

Caratteri accettati dall'Assembler

Le frasi sorgente Assembler sono scritte utilizzando i seguenti caratteri del codice ISO esteso:

- caratteri alfabetici: da A a Z

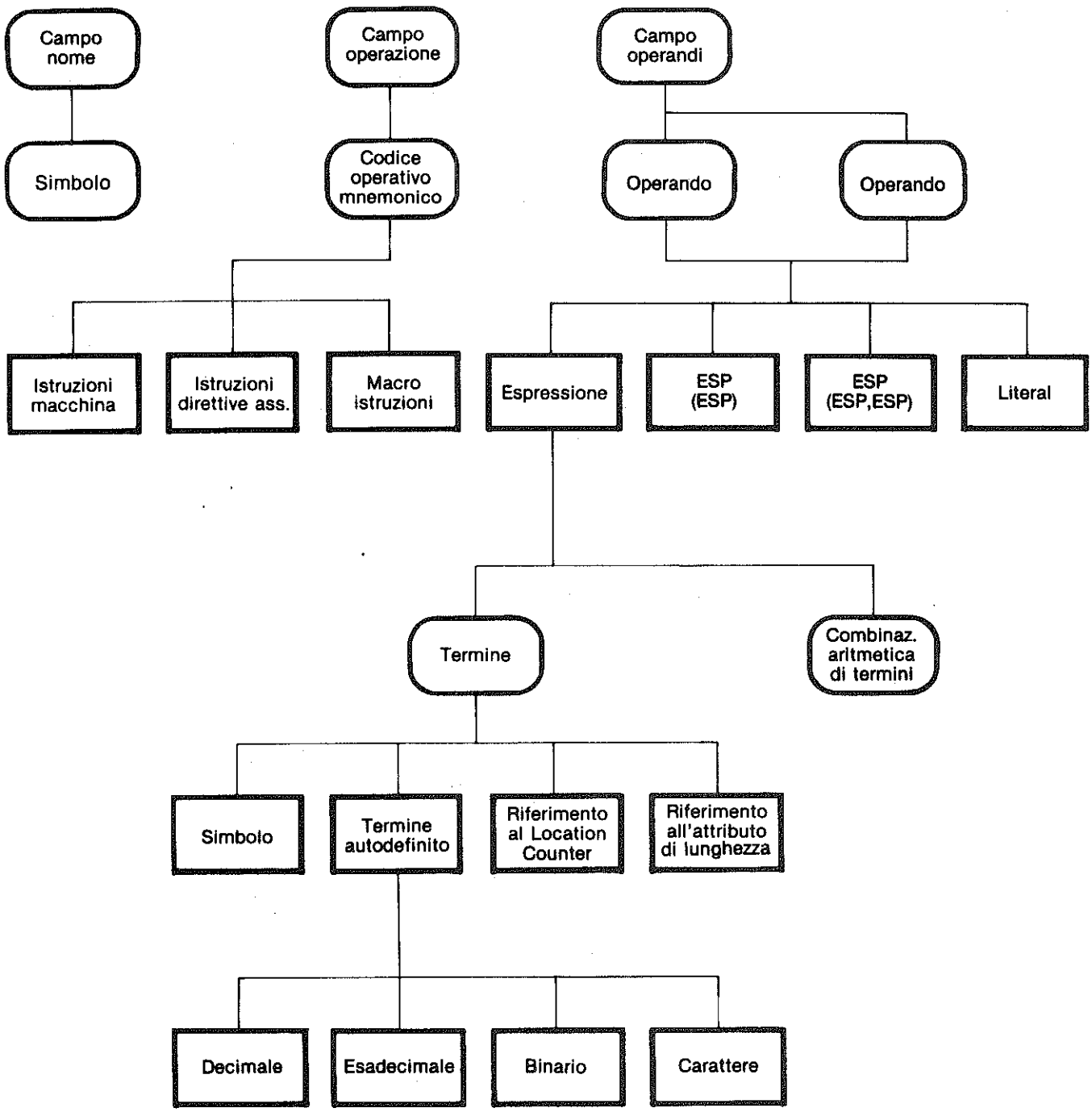
- caratteri speciali: vengono raggruppati in:

  . delimitatori, ()
  . blank
  . operatori + - * /
  . crt. speciali =

- caratteri numerici: da Ø a 9

Oltre ai caratteri sopra citati possono fare parte di una frase sorgente tutti i crt. del codice ISO esteso, purchè facciano parte di stringhe di dati tipo carattere.
Esempio:

```
DC      C'      ≠'
```


Campo nome
Campo operazione
Campo operandi
Simbolo
Codice operativo mnemonico
Operando
Operando
Istruzioni macchina
Istruzioni direttive ass.
Macro istruzioni
Espressione
ESP (ESP)
ESP (ESP,ESP)
Literal
Termine
Combinaz. aritmetica di termini
Simbolo
Termine autodefinito
Riferimento al Location Counter
Riferimento all'attributo di lunghezza
Decimale
Esadecimale
Binario
Carattere

Il campo operandi (vedi figura 4-2) è composto da una o più espressioni o literal; a sua volta una espressione è costituita da un termine o da una combinazione di termini.

I termini possono essere:

- rilocabili quando il loro valore durante la fase di esecuzione dipende dal punto di caricamento del programma; devono quindi essere basati su qualche registro che viene attualizzato a RUN-TIME dall'istruzione BALR

- assoluti quando il loro valore è comunque indipendente dal punto di caricamento del programma.

Si distinguono 4 tipi di termini:

- simboli
- termini autodefiniti
- riferimento al Location Counter
- riferimento all'attributo lunghezza di simboli

Simboli

I simboli sono definiti e usati dall'utente per riferirsi in modo simbolico a qualsiasi elemento del programma (aree di memoria, istruzioni, costanti ecc.). Un simbolo è costituito al massimo di 8 caratteri alfanumerici il primo dei quali deve essere alfabetico. Si distinguono 2 classi di simboli:

- simboli interni
- simboli esterni

Definizione di simboli interni: L'Assembler attribuisce ad ogni simbolo interno delle caratteristiche dette "attributi". Un simbolo viene detto "definito" quando compare nel campo nome di una istruzione sorgente; nell'ambito del modulo sorgente un simbolo è definibile una volta sola. Gli attributi che l'Assembler assegna ad un simbolo definito sono:

- valore
- lunghezza implicita
- tipo di simbolo

Il valore assegnato ad un simbolo è l'indirizzo del 1° byte di sinistra della zona di memoria contenen-

te l'elemento in tal modo nominato. Il valore di un simbolo non può essere negativo o comunque superare $2^{31} - 1$ questo perchè l'indirizzo di macchina viene espresso su 24 bit.

La lunghezza implicita di un simbolo è la lunghezza, in byte, del campo di memoria nominato.

Un simbolo può essere rilocabile o assoluto in relazione al tipo di espressione presente nel campo operandi della frase, che può per l'appunto essere rilocabile o assoluto.

Simboli precedentemente definiti: Il linguaggio Assembler richiede che nel campo operandi di particolari tipi di frasi compaiano simboli precedentemente definiti.
Esempio:

```
        B
        ...
        ...
        ...
A       EQU     B
```

Questo significa che i simboli in questione devono essere stati definiti in una frase sorgente posta fisicamente prima dell'istruzione in corso di elaborazione.

Simboli esterni: I simboli esterni sono quelli che servono per identificare il nome dei moduli esterni rispetto al modulo che si sta assemblando. Questi simboli sono definiti nel campo operandi di particolari istruzioni che gestiscono il collegamento tra i moduli (CALEXT, CALEXS, ACT, RLSEM, LAEXT). I nomi esterni vanno a far parte del dizionario dei nomi esterni nell'ordine in cui sono trovati nel programma sorgente. Un nome esterno anche se utilizzato più volte, dà luogo ad un unico elemento del dizionario. I simboli esterni si trasformano quindi in informazioni per il Linker e vengono di conseguenza trattati diversamente dagli altri simboli; a loro non vengono assegnati attributi. Lo stesso simbolo può essere utilizzato sia come interno che come esterno per indicare parti di programma completamente diverse.

Termini autodefiniti

Un termine autodefinito è quello il cui valore è costituito dal termine stesso; l'Assembler non assegna quindi alcun valore a questo termine. Tutti i termini autodefiniti sono considerati assoluti, poichè i loro valori non subiscono variazioni quando il programma di cui fanno parte viene caricato a differenti indirizzi di memoria. Il valore di un termine autodefinito non può superare i $2^{31}-1$ e il suo attributo lunghezza per definizione è 1; i termini autodefiniti servono per specificare dati immediati, maschere, registri, incrementi di indirizzi, lunghezze esplicite ecc.
Esempio:

```
MVI  A,  C'123'
```

l'attributo lunghezza del secondo operando vale 1.
I termini autodefiniti si dividono in 4 tipi:

- decimale
- esadecimale
- binario
- carattere

Termine autodefinito decimale: E' un numero decimale senza segno, scritto come una sequenza di caratteri numerici (max 8). Il programma Assembler provvede a convertire il dato decimale in dato binario.

Termine autodefinito esadecimale: E' costituito da un max di 6 caratteri esadecimali scritti tra apici (') e preceduti dalla lettera X.
Esempio:

```
X'A1BC'
```

Ogni carattere esadecimale è convertito in binario su 4 bit.

Nota: I caratteri numerici vengono allineati a destra e riempiti con bit a zero a sinistra. I caratteri alfabetici vengono allineati a sinistra e riempiti con byte a Ø (blank) a destra.

Esadecimale	Binario
Ø	ØØØØ
1	ØØØ1
2	ØØ1Ø
3	ØØ11
4	Ø1ØØ
5	Ø1Ø1
6	Ø11Ø
7	Ø111
8	1ØØØ
9	1ØØ1
A	1Ø1Ø
B	1Ø11
C	11ØØ
D	11Ø1
E	111Ø
F	1111

Tabella 4-1 Tabella di corrispondenza tra carattere esadecimale e binario (bit creati).

Termine autodefinito binario: E' una sequenza di caratteri zero e uno racchiusi tra apici e preceduti dalla lettera B.
Esempio:

B'11Ø1Ø1'

Questo tipo di termine viene normalmente usato per definire maschere per operazioni logiche. Ogni carattere numerico rappresenta un bit di memoria, e al massimo possono essere espressi 24 bit. Nel caso in cui il numero di caratteri a Ø o 1 non sia un multiplo di 8 la normalizzazione con Ø binari e fatta sulla sinistra.

Termine autodefinito carattere: Questo tipo di termine è costituito al massimo da 3 caratteri chiusi tra apici e preceduti dal carattere C.
Esempio:

C''
C'ABC'

Il valore di ogni carattere può essere o blank o una

delle 256 possibili configurazioni di un byte.

Nota: L'utente non può digitare il crt. EOL (255_{16}) come dato.

Riferimento al Location Counter

Il programma Assembler gestisce un contatore di memoria o Location Counter che contiene, in ogni momento dell'assemblaggio, l'indirizzo del primo byte di memoria disponibile. Quando viene elaborata una istruzione, sia che essa corrisponda ad una istruzione macchina, sia che essa definisca un'area dati, il Location Counter deve essere eventualmente aggiustato per esigenze di allineamento in modo tale che contenga l'indirizzo dell'elemento di programma che si sta considerando; infine viene incrementato di un numero di byte pari alla lunghezza dell'elemento di cui si tratta (lunghezza dell'istruzione macchina, lunghezza dell'area dati). Se l'istruzione ha un nome, a tale simbolo viene assegnato il valore del Location Counter dopo l'eventuale aggiustamento, ma prima dell'incremento della lunghezza dell'istruzione stessa. L'Assembler mantiene un Location Counter per ognuna delle sezioni di cui è composto il modulo sorgente cioè per.la Control Section e per ognuna delle Dummy Section (al max 127). Il valore massimo del Location Counter è $2^{31} - 1$. L'utente può riferirsi al valore corrente del Location Counter nel campo operandi delle frasi usando il carattere asterisco (*): usare tale termine nell'operando di una istruzione equivale a dare un nome all'istruzione e quindi usare tale nome nel campo operandi. Quindi il termine * è un termine rilocabile ed il suo valore corrisponde al valore corrente del Location Counter, cioè dell'indirizzo dell'istruzione macchina in cui il termine compare, anche quando il termine è utilizzato nell'indicazione di un literal, o all'indirizzo della costante o dell'area quando il termine è usato in una frase di definizione di dati.

Per quanto riguarda la determinazione dell'attributo lunghezza del termine *, si impone una distinzione:

- corrisponde alla lunghezza dell'istruzione macchina quando il termine compare in una tale istruzione, anche in un operando espresso tramite literal

- corrisponde alla lunghezza della costante quando il termine compare in una frase di definizione dati, in particolare nelle costanti di tipo indirizzo; al termine * viene assegnata lunghezza 1 quando compare in altri tipi di frasi direttive.

Il riferimento al Location Counter non deve essere utilizzato in una frase in cui si richieda l'uso di nomi predefiniti, salvo il caso di istruzioni ORG e EQU; comunque l'eventuale utilizzo del termine * in fase di definizione dati nelle espressioni che indicano il fattore di duplicazione o il modificatore di lunghezza non dà luogo a segnalazione di errore: al termine viene attribuito lunghezza Ø e il valore corrispondente al valore corrente del Location Counter senza l'eventuale aggiustamento per l'allineamento.

Riferimento all'attributo lunghezza dei simboli

L'attributo lunghezza di un simbolo può essere usato come termine mediante il prefisso L' (L apice) seguito dal simbolo stesso.
Esempio:

```
        ...
        ...
        ...
A       LA        2, 3
B       DS        F
MVC     A (L'B)   H'425'
        ...
```

L'attributo lunghezza implicito del simbolo B valorizza il suddetto termine (valore 4); si tratta quindi di un termine assoluto, il cui attributo lunghezza è 1.
Esempio:

```
L'L'B
```

L'attributo lunghezza risultante è 1.

Espressioni

Come risulta dalla figura 4-4 l'espressione può essere costituita da un termine o da una combinazione di termini. A questo punto si distinguono due tipi di espressioni:

- semplici

- composte.

Le espressioni composte devono seguire le normali regole del calcolo algebrico:

- ogni espressione può avere al max 16 termini

- gli operatori aritmetici ammessi sono: + - * /

- una espressione non può iniziare con un operatore aritmetico

- una espressione non può contenere 2 termini o 2 operatori aritmetici in successione

- combinazioni di termini racchiuse tra parentesi possono essere usate in combinazione con termini posti fuori dalla parentesi

- sono ammessi come numero max 5 livelli di parentesi.

Valutazione delle espressioni: Il calcolo delle espressioni viene eseguito da sinistra verso destra ponendo come priorità di esecuzione quella presente nelle normali espresioni algebriche parentetizzate. La divisione per Ø è ammessa e dà come risultato Ø.

Attributi delle espressioni: L'attributo valore viene assegnato alla espressione durante il calcolo della stessa. L'attributo lunghezza equivale all'attributo lunghezza del primo termine di sinistra della espressione.
Esempio:

```
MVC  A, B * C
```

se B si immagina lungo 3, l'attributo lunghezza è lungo 3.

Le espressioni inoltre possono essere assolute o rilocabili:

- nel primo caso il valore rimane immutato a prescindere dal punto di caricamente in memoria del programma

- nel secondo caso l'espressione subirà un mutamento pari a N, se il programma che la contiene è riloca-

to in memoria di N byte rispetto all'assegnazione originaria di memoria.

Un'espressione assoluta è costituita da un termine assoluto o da una espressione algebrica di termini assoluti.
Esempio:

```
A       DS      40
        ...
        ...
        ...
        MVI     A, X'A12' * 12 - B
                'ØØ1Ø1Ø11'
```

Può anche consistere di una combinazione aritmetica di termini rilocabili da soli o in combinazione con termini assoluti. In questo caso devono essere seguite le seguenti regole:

1. I termini rilocabili presenti nella espressione devono essere appaiati, cioè avere segno opposto e lo stesso attributo di rilocabilità (cioè devono essere definiti o nella stessa Control Section o nella stessa Dummy Section). I termini appaiati possono anche non essere contigui. Nel seguito, al fine di sintetizzare gli esempi, si riportano solo i componenti dell'espressione

   RT ⟶ termine rilocabile
   AT ⟶ termine assoluto

   Esempio:

   RT + AT + RT - RT - AT - RT

2. I termini rilocabili non possono entrare in divisioni e moltiplicazioni:
   Esempio:

RT - RT * AT	non valida
(RT - RT) * AT	valida

   Quando appaiono termini rilocabili di segno opposto e la stessa rilocabilità, si annulla l'effetto della rilocazione e il valore dei termini rimasti appaiati rimane costante. Un'espressione rilocabile è costituita da un termine rilocabile, op-

pure da una combinazione di termini rilocabili o frammisti a termini assoluti; quest'ultima combinazione però sottostà alle seguenti condizioni:

3. I termini rilocabili devono essere in numero dispari.

4. I termini rilocabili devono essere appaiati, tranne uno.

5. Il termine spaiato non deve essere preceduto dal segno meno.

A questo punto risulta chiaro che il valore dell'espressione corrisponde al valore del termine spaiato e l'attributo di rilocabilità del termine spaiato diventa l'attributo di rilocabilità dell'espressione.
Esempio:

RT - (RT + RT - RT) + RT

Literal

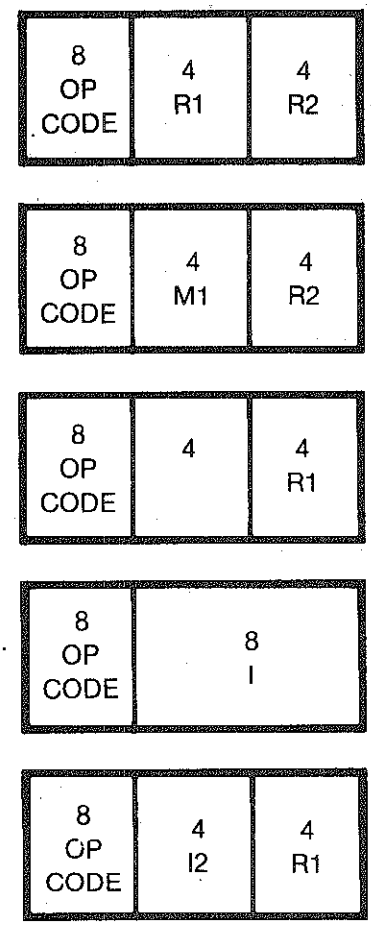
L'utente Assembler ha la possibilità di indicare nel campo operandi dell'istruzione delle reppresentazioni di dati invece che riferimenti a tali dati. I literal sono il mezzo più semplice per introdurre dei dati in un programma. Un literal è formato da una costante preceduta dal segno uguale (=). La presenza di un literal all'interno di una frase sorgente ordina all'Assemblatore di porre il valore del literal stesso (cioè la costante specificata) in una zona di memoria denominata tavola dei literal, e di sostituire ad esso, all'interno della corrispondente frase oggetto, l'indirizzo relativo. La definizione di un dato come literal segue le regole descritte più avanti circa la specificazioni di costanti (vedi DC).

Tavola dei literal: Tutti i literal definiti in un programma sorgente vengono raggruppati dall'Assemblatore in una zona di memoria detta tavola dei literal, e l'indirizzo in cui viene posto il literal viene assemblato nella frase in cui il literal è utilizzato. La tavola dei literal viene allocata alla fine della Control Section, a partire dall'indirizzo di Location Counter successivo o quello dell'ultima istruzione oggetto.

Tipi di Formati macchina

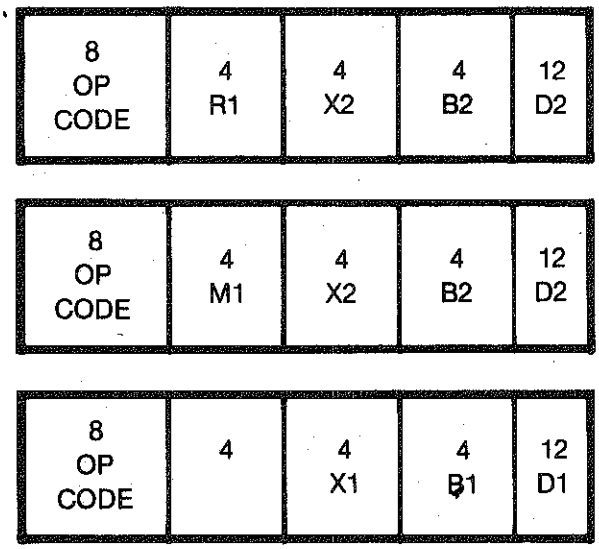
Ogni istruzione esecutiva del linguaggio dà luogo ad un codice oggetto, la cui lunghezza dipende dal formato dell'istruzione. Si distinguono infatti 5 formati fondamentali:

1. Formato RR



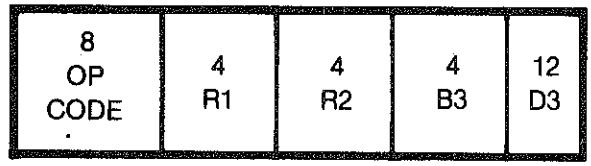
Lunghezza 2 byte

2. Formato RX



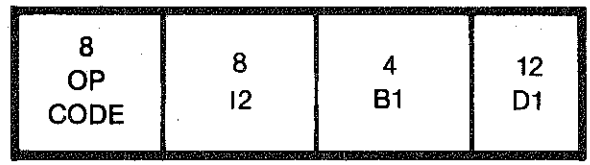
Lunghezza 4 byte

3. Formato RS



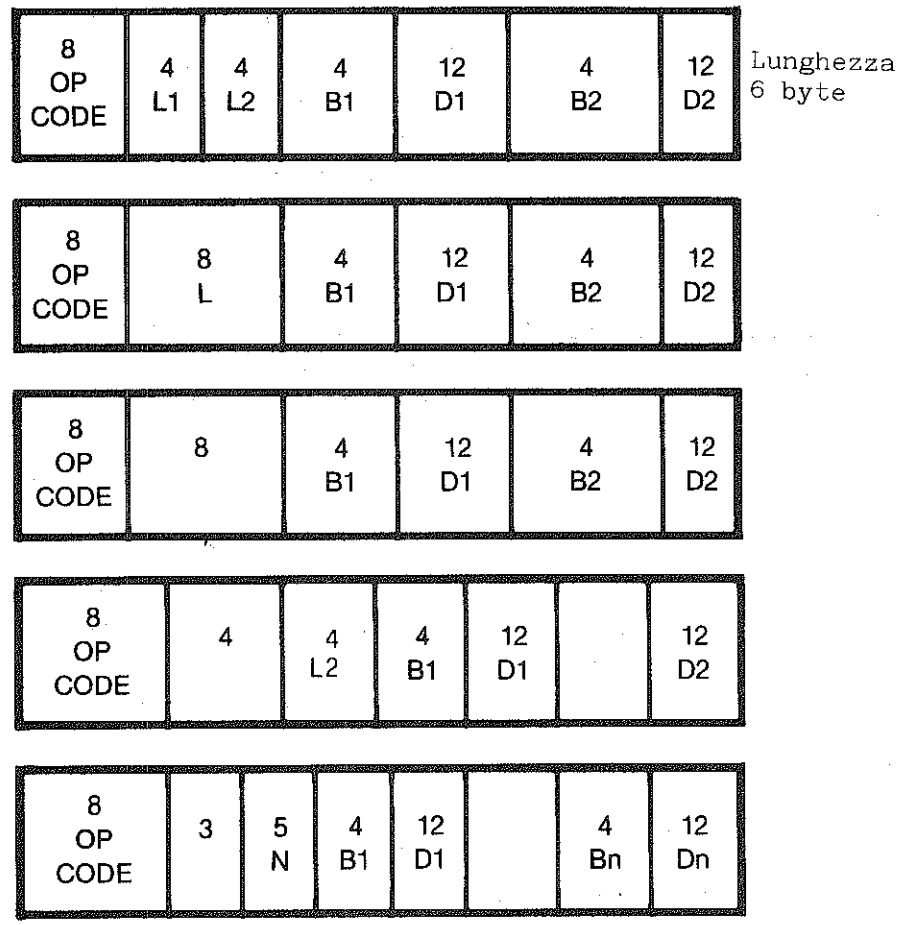
Lunghezza 4 byte

4. Formato SI



Lunghezza 4 byte

5. Formato SS



Dove:

OP CODE	Codice Operativo
M	maschera
R	registro
D	valore di scostamento
B	registro base
X	registro indice
S	indirizzo implicito (in genere un simbolo)
I	immediato (in genere termine autodefinito)
L	lunghezza
N	numero di argomenti
1, 2, 3...n	riferimento al 1°, 2°...ennesimo operando della frase sorgente

I termini sopra indicati vengono usati anche per la

stesura delle istruzioni del linguaggio Assembler.

Nell'ambito del formato SS (ALM, AM, CLM, CM, DIS, MVO, SES, SESM, SLM, SM) ci sono istruzioni a lunghezza variabile, quindi la lunghezza dell'oggetto prodotto dipenderà dal numero di operandi specificati nella frase sorgente stessa. L'allineamento delle istruzioni esecutive (macchina), viene svolto automaticamente dall'Assemblatore alla mezza parola.

Descrizione delle istruzioni Assembler

Nel seguito sono descritte dettagliatamente tutte le istruzioni e le macro-istruzioni del linguaggio Assembler P6Ø6Ø. Le istruzioni e le macro sono poste in ordine alfabetico. La descrizione di ogni istruzione e macro è realizzata secondo la medesima struttura, che è composta dai seguenti punti fondamentali:

1. Descrizione funzionale: è una breve descrizione della funzione della istruzione

2. Formato: è una descrizione del formato/i della istruzione

3. Descrizione: è la descrizione dei campi nome operazione e operandi che compaiono nel formato dell'istruzione

4. Espansione: è la descrizione delle istruzioni macchina generate dalla fase di macroespansione di una macro; la macroespansione avviene direttamente in codice oggetto.

Note: Le seguenti notazioni sono impiegate nella descrizione delle istruzioni ASSEMBLER, ma non appartengono al linguaggio in cui è scritta un'istruzione:

{ } indica la scelta di soltanto uno dei parametri indicati

[ ] indica la completa opzionalità, cioè la possibilità di scegliere K parametri, con $Ø \leqslant K \leqslant 1$

5. Esempi: è una raccolta di esempi di impiego della istruzione descritta che permette di chiarire eventuali dubbi riguardo la codifica corretta dell'istruzione.

Istruzioni direttive per l'assemblatore

Le istruzioni direttive per l'Assemblatore specificano delle particolari funzioni ausiliarie da compiersi a cura del programma Assembler stesso; i loro formati e le loro descrizioni sono i seguenti.

Descrizione Funzionale

Identifica l'inizio della sezione di controllo. Tutte le frasi che seguono sono assemblate come facenti parte di questa sezione, finchè non venga incontrata una frase che segni l'inizio di una nuova sezione.

Formato

nome	operazione	operando
[simbolo]	**CSECT**	

Descrizione

Nome

Se la frase è contrassegnata da un nome simbolico, questo indica l'indirizzo del 1° byte del modulo oggetto. A questo simbolo viene assegnato l'attributo lunghezza 1; esso è rilocabile.

Operandi

Il campo operandi non è usato e il Location Counter è settato a Ø. Eventuali caratteri presenti nel campo operandi sono trattati come commento.

Esempio

```
SEZ      CSECT
         ...
         ...
         ...
         LA          1Ø, AREA
         USING       DEF, 1Ø
         ...
         ...
         ...
AREA     DS          CL40
         ...
         ...
         ...
         END
```

DC

Descrizione Funzionale

Permette di introdurre in un programma dei dati sotto forma di costante.

Formato

nome	operazione	operando
[simbolo]	DC	[D] T [Ln] {'costante' / (costante)}

Descrizione

Nome

Se in questo campo è presente un simbolo, a quest'ultimo viene assegnato come valore l'indirizzo del 1° byte di sinistra della costante stessa e come attributo lunghezza la lunghezza implicita o esplicita della costante; si tratta di un simbolo rilocabile.

Operandi

Il campo operandi si compone di 4 sottocampi:

1. Sottocampo D - fattore di duplicazione; è un sottocampo opzionale; se presente provoca la riproduzione della costante per D volte, dopo che la costante è stata assemblata e completata nel suo esatto formato. Il fattore di duplicazione può essere espresso in due modi:

   - attraverso un termine autodefinito decimale

   - attraverso una espressione assoluta positiva chiusa tra parentesi, in cui gli eventuali simboli che compaiono siano precedentemente definiti.

Se il fattore di duplicazione è Ø, nessuna costante è

generata. L'istruzione DC con fattore di duplicazione Ø serve unicamente per assegnare al simbolo che compare nel campo nome un attributo lunghezza uguale alla lunghezza specificata nella frase, senza riservare l'area relativa. Tale fattore non è ammesso nella definizione delle costanti indirizzo.

2. Sottocampo T - tipo; è un sottocampo obbligatorio e indica all'assembler il tipo di costante da generare; questo sottocampo è costituito da una parola chiave.

Codice	Tipo di costante	Formato macchina
C	carattere	codice di 8 bit per ogni carattere
X	esadecimale	codice di 4 bit per ogni carattere esadecimale
B	binaria	formato binario
F	virgola fissa	formato binario in virgola fissa; una parola
H	virgola fissa	formato binario in virgola fissa; una mezza parola
S	indirizzo	valore del Registro base e dello spiazzamento; una mezza parola

3. Sottocampo Ln - modificatore di lunghezza; è un sottocampo opzionale e se presente definisce la lunghezza in byte della costante. Può essere indicato nella forma:

   - termine autodefinito decimale

   - espressione assoluta positiva chiusa tra parentesi; i simboli che compaiono nella espressione devono essere precedentemente definiti.

Se la lunghezza esplicita non è definita, l'Assembler assume la lunghezza implicita della costante, che può dipendere dal valore costante specificato X, B, C o dal tipo della costante stessa F, H. Nelle costanti che richiedono l'allineamento, specificare il modificatore implica il non allineamento della costante stessa.

4. Sottocampo valore costante; questo sottocampo è obbligatorio e indica il valore della costante. Le costanti di dati (B/C/X/ F/H) devono essere scritte tra apici; le costanti indirizzo (S) devono essere racchiuse tra parentesi.

Costante carattere (Tipo C) Per una costante carattere può essere utilizzata una qualsiasi delle 256 configurazioni possibili di un byte, con la sola eccezione della configurazione "FE". Il valore costante deve essere racchiuso tra apici; se in tale valore deve essere specificato il crt. apice (') ed esso deve essere indicato 2 volte: con la traduzione della stringa oggetto corrispondente apparirà un solo apice, poichè ogni carattere nelle costanti di questo tipo viene tradotto in byte. La lunghezza massima della costante è di 256 bytes. Se non è specificato il modificatore di lunghezza, l'Assembler assume come lunghezza della costante il numero di caratteri indicati tra apici. Se la lunghezza è espressa si possono avere le seguenti conseguenze:

- se il numero di caratteri della costante eccede la lunghezza specificata, la costante è troncata a destra

- se il numero di caratteri della costante è inferiore alla lunghezza specificata, la costante viene normalizzata a destra con blank.

Costante esadecimale (tipo X) Le costanti esadecimali sono costituite da una sequenza di caratteri esadecimali (da Ø a 9, da A a F), racchiusa tra apici. La lunghezza massima è di 256 byte: quindi possono essere indicati fino a 512 caratteri esadecimali, poichè ogni coppia di crt. esadecimali rappresenta un byte di memoria e viene quindi tradotta in un byte; se la lunghezza è dispari, il primo crt. esadecimale di sinistra darà origine ad un byte i cui 4 bits meno significativi contengono il digit esadecimale men-

tre i 4 bits più significativi sono a zero. Se non viene specificata la lunghezza, l'Assembler assume come lunghezza della costante il numero $l = \left[\frac{n+1}{2}\right]$, con n il numero di caratteri esadecimali specificati. Se la lunghezza è specificata si ottiene che:

- se il numero delle coppie di caratteri esadecimali eccede la lunghezza specificata, i crt. più a sinistra sono troncati

- se il numero delle coppie di cifre esadecimali è inferiore alla lunghezza specificata, i bytes necessari per raggiungere tale lunghezza sono inseriti a sinistra e posti a zero binario.

Costante binaria (Tipo B) Le costanti binarie sono rappresentate mediante una sequenza di Ø e 1, racchiusa tra apici. La lunghezza massima è di 256 byte. La lunghezza implicita è data dal numero di byte occupati dalla costante, inclusa l'eventuale necessaria normalizzazione con bit a zero, poichè ogni carattere Ø o 1 rappresenta un bit della memoria; tale lunghezza implicita si ricava in questo modo:

$\left[\frac{n+7}{8}\right]$ in cui n è il numero di caratteri Ø o 1 indicati.

La normalizzazione o il troncamento avvengono a sinistra.

Costanti in virgola fissa (tipo F e H) Contengono un numero decimale con o senza segno, racchiuso tra apici. Esso viene convertito nel corrispondente valore binario e messo in una parola (tipo F) o mezza parola (tipo H) con l'allineamento appropriato: la lunghezza implicita della costante tipo F è quindi 4, è due per la costante tipo H. Se è specificata la lunghezza, la quale non può essere superiore a 4, non viene eseguito l'allineamento. Se il valore del numero eccede la lunghezza esplicita od implicita della costante, il segno si perde e sono troncati i bit più a sinistra. Un valore negativo è posto nella forma di complemento a 2 del corrispondente valore positivo. I valori massimi delle costanti in virgola fissa sono:

lunghezza costante	Max	Min
4	$2^{31} - 1$	-2^{31}
2	$2^{15} - 1$	-2^{15}
1	$2^{7} - 1$	-2^{7}

Costanti indirizzo (S) Una costante indirizzo contiene un indirizzo di memoria, espresso in sorgente tramite una o più espressioni racchiuse fra parentesi. Il fattore di duplicazione non è ammesso.

Indirizzo tipo S: serve per ottenere indirizzi espressi nella forma registro base e scostamento; l'indirizzo può essere specificato:

1. Tramite una espressione rilocabile o assoluta.

2. Tramite 2 espressioni assolute, la prima delle quali rappresenta il valore di scostamento, la seconda il registro base.

Questo tipo di costante è assemblata in una mezza parola: i primi 4 bit di sinistra rappresentano il registro base; i rimanenti 12 bit esprimono lo scostamento. Se viene specificata una lunghezza esplicita, che non può essere diversa da 2, non viene effettuato l'allineamento.

Tabella riassuntiva delle costanti

Tipo	LL implicita BYTE	Allinea-mento	Duplica-zione	Range LL esplicita	Delimi-tatore	Valore della costante	Troncamento Normalizza-zione
C	-	Byte	ammessa	1 - 256^N	'	caratteri	destra
X	-	Byte	ammessa	1 - 256^N	'	crt esadecim.	sinistra
B	-	Byte	ammessa	1 - 256	'	crt binari 1/Ø	sinistra
F	4	Parola	ammessa	1 - 4	'	numero decim.	sinistra
H	2	Mezza parola	ammessa	1 - 4	'	numero decim.	sinistra
S	2	Mezza parola	non ammessa	solo 2	()	- una espressione assoluta o rilocabile - due espressioni assolute: exp (exp)	-

Nota Nelle istruzioni DS in cui sia specificato un valore costante di tipo C o X, il modificatore di lunghezza o la lunghezza implicita può essere anche superiore a 256, fino a un massimo di 65535.

Esempio

```
FILE   DC  C'END'          genera  END
COST1  DC  3C'AZ'          genera  AZAZAZ
COST2  DC  2CL5'AZ'        genera  AZ␣␣␣AZ␣␣␣
COST3  DC  2CL4'ABCDEF'    genera  ABCDABCD

COST1  DC  X'412A'         genera  4124 (2 byte)
COST2  DC  2XL4'A1F43'     genera  ØØØA1F43ØØØA1F43 (8
                                   byte)
COST3  DC  3XL2'A1F43'     genera  1F431F431F43 (6 byte)

COST   DC  B'11Ø1'           genera  ØØØØ11Ø1 (1 byte)
COST1  DC  BL1'11Ø111ØØ1'    genera 1Ø111ØØ1 (1 byte)

DC  H'3Ø'  genera ØØ1E (2 byte)
DC  F'-3'  genera FFFFFFFD (4 byte)

DC  S(BETA)
DC  S(4ØØ(13))
```

Descrizione Funzionale

Permette di escludere un certo numero di Registri generarali dal gruppo di registri precedentemente specificati quali registri base mediante frasi USING.

Formato

nome	operazione	operando
/*	DROP	R_1 [,R_2 .....R_n]

Descrizione

Nome

Non ammesso

Operandi

Questo campo contiene una successione di espressioni assolute, separate dalla virgola che specificano i registri base da sopprimere. Ogni espressione può avere valore da Ø a 15; il numero max di tali espressioni è 16.

Esempio

```
SEZ     START   1Ø
        ...
        ...
        ...
        USING   BASE, 1Ø
        ...
        ...
        ...
BASE    DS      ØH
        ...
        ...
        ...
        DROP    1Ø
        ...
        ...
        ...
        USING   BASE, 1Ø
        ...
        ...
        ...
        END
```

DS

Descrizione Funzionale

Permette di riservare aree di memoria, attribuendo loro eventualmente dei nomi per poterle indirizzare in modo simbolico.

Formato

nome	operazione	operando
[simbolo]	DS	[D] T [Ln] [{'costante' / (costante)}]

Descrizione

Nome

Se in questo campo è presente un simbolo, a quest'ultimo viene assegnato l'indirizzo del 1° byte di sinistra dell'area stessa e come attributo lunghezza la lunghezza implicita o esplicita dell'area; si tratta di un simbolo rilocabile.

Operandi

Il campo operandi si compone di 4 sottocampi:

1. Scttocampo D - fattore di duplicazione; è un sottocampo opzionale, se presente provoca la riproduzione dell'area di memoria per D volte. Il fattore di duplicazione può essere espresso in 2 modi:

   - attraverso un termine autodefinito decimale

   - attraverso una espressione assoluta positiva chiusa tra parentesi, in cui gli eventuali simboli che compaiono siano precedentemente definiti.

Se il fattore di duplicazione è Ø, l'istruzione può essere usata per assegnare un nome ad un'area di memoria, senza che tale area sia riservata effettivamente. Altre frasi DS possono riservare l'area stessa, assegnando nomi ai suoi sottocampi. Quindi il fattore di duplicazione Ø serve ad assegnare al simbolo che compare nel campo nome un attributo lunghezza uguale alla lunghezza specificata nella frase, senza riservare l'area relativa.

2. Sottocampo T - tipo; è un sottocampo obbligatorio e indica all'Assembler il tipo di costante da generare; questo sottocampo è costituito da una parola chiave.

3. Sottocampo Ln - modificatore di lunghezza; è un sottocampo opzionale e se presente definisce la lunghezza in byte dell'area. Può essere indicato nella forma:

   - termine autodefinito decimale

   - espressione assoluta positiva chiusa tra parentesi; i simboli che compaiono nella espressione devono essere precedentemente definiti.

   La lunghezza max permessa per le definizioni di aree in cui sia specificato il tipo C o X è 65535 byte.

4. Sottocampo valore costante; è un sottocampo opzionale, e se presente indica il valore della costante. Le costanti dati (B/C/X/F/H) devono essere scritte tra apici; le costanti indirizzo (S) devono essere racchiuse tra parentesi.

Nota. Per ulteriori informazioni sulle costanti nei loro vari tipi, vedere istruzione DC.

Nel complesso le istruzioni DS sono trattate dall'Assembler come quelle DC, con le seguenti differenze:

- a fronte di istruzioni DS viene lasciata un'area di memoria la cui lunghezza è pari a quella specificata dal modificatore di lunghezza o a quella desunta dal tipo o dal valore costante indicati con l'eventuale duplicazione; tale area è azzerata

- nel caso di istruzione DS del tipo C/X/B in cui non compaiano nè il modificatore, nè il valore costante, la lunghezza implicita associata è 1, e quindi viene riservata un'area di 1 byte.

Esempio

```
DS  CL5       viene riservata un'area di 5 byte
DS  C'ASMIS'  viene riservata un'area di 5 byte
DS  5X'AB'    viene riservata un'area di 5 byte
```

DSECT

Descrizione Funzionale

Identifica l'inizio di una Dummy Section. In un modulo sorgente sono ammesse come max 255 Dummy Section.

Formato

nome	operazione	operando
Simbolo	DSECT	

Descrizione

Nome

E' obbligatorio; è un simbolo rilocabile il cui valore equivale all'indirizzo del 1° byte di tale sezione. L'attributo lunghezza è 1.

Operandi

Questo campo non è usato, ed eventuali caratteri presenti sono trattati come commento.

Esempio

```
                ...
                ...
                ...
                USING     DUM, 6
                ...
                ...
                ...
DUM             DSECT     CL3Ø
NAME            DS        CL5
COD             DS        CL4
IMPORT          DS
INIT            DSECT
                ...
                ...
                ...
```

EJECT

Descrizione Funzionale

Provoca la stampa della linea successiva del listing su una nuova pagina. Se l'istruzione viene riconosciuta all'inizio di una pagina di stampa, non viene eseguita.

Formato

nome	operazione	operando
⌀	EJECT	

Descrizione

Nome

Non ammesso.

Operandi

Non sono richiesti operandi, e quindi qualsiasi carattere compaia in questo campo è considerato come commento.

Esempio

```
        START   256
INIT    BALR    11, Ø
        USING   *, 11
        BC      15, GO
        ...
        ...
        ...
        EJECT
        ...
        ...
        ...
        END
```

END

Descrizione Funzionale

Provoca la fine dell'assemblaggio, e quindi deve essere l'ultima del modulo sorgente.

Formato

nome	operazione	operando
	END	

Descrizione

Nome

Non ammesso

Operandi

Non sono usati, quindi qualsiasi carattere compaia in questo campo è considerato commento.

Esempio

```
SEZ     START   1Ø
        ...
        ...
        ...
        USING   BASE, 1Ø
        ...
        ...
        ...
BASE    DS      D
        ...
        ...
        ...
        END
```

EQU

Descrizione Funzionale

E' utilizzata per definire un simbolo a cui assegnare il valore e tutti gli attributi dell'espressione specificata nel campo operandi.

Formato

nome	operazione	operando
Simbolo	**EQU**	**Espressione**

Descrizione

Nome

E' obbligatorio.

Operandi

In questo campo sono ammesse qualsiasi espressione assolute o rilocabili. Ogni simbolo che compare in tale espressione, deve essere precedentemente definito. Al simbolo definito nel campo nome viene attribuito il valore e gli attributi lunghezza e di rilocabilità dell'espressione nel campo operandi.

Esempio

```
                START
RR      LR      3,4
RX      A       3, GIVEN
        ...
        ...
        ...
GIVEN   DC      F'33'
AREA    DS      XL2ØØØ
ONE     DS      CL24Ø
TWO     DS      CL8Ø
        ...
        ...
        ...
A       EQU     X'FF'
E       EQU     L'TWO
        ...
        ...
        ...
E       EQU     AREA+1ØØ
        ...
        ...
        ...
H       EQU     RX
        ...
        ...
        ...
        END
```

ORG

Descrizione Funzionale

Permette di modificare il contenuto del Location Counter della sezione a cui la frase appartiene.

Formato

nome	operazione	operando
b	**ORG**	**[Espressione rilocabile]**

Descrizione

Nome

Non ammesso

Operandi

Se in questo campo è presente una espressione rilocabile, il Location Counter di tale sezione che si sta assemblando è settato con il valore di tale espressione. Se il campo operando è omesso, l'istruzione non ha alcun effetto sul Location Counter.

Il simbolo che compare nell'espressione deve essere precedentemente definito, e deve essere definito nella stessa sezione in cui compare la ORG. Una istruzione ORG non può essere usata per specificare una allocazione precedente all'inizio della sezione in cui compare. Una nuova istruzione ORG con caratteristiche simili alla precedente, annulla l'effetto sul Location Counter della precedente, permettendo all'Assembler di resettarlo.

Esempio

```
                ...
CARD        DS      CL8Ø
COD         DS      CL1Ø
NAME        DS      CL4Ø
GIVEN       DS      CL3Ø
            ORG     * - 8Ø
ART         DS      CL3Ø
LARGE       DS      CL7
            DS      CL43
            ...
            ...
            ...
            END
```

PRINT

Descrizione Funzionale

Serve per controllare la stampa del listing dell'Assembler; più precisamente serve per specificare le modalità di stampa da quel punto in poi.

Formato

nome	operazione	operando
b	PRINT	[{ON / OFF}][{DATA / NO DATA}]

Descrizione

Nome

Non ammesso.

Operandi

Gli operandi possono essere indicati entrambi separati da virgola.

ON → stampa del listing
OFF → non avviene la stampa

DATA → le costanti vengono stampate per tutta la loro lunghezza nel listing

NO DATA → sono stampati solo i primi 8 byte a sinistra delle costanti

gli operandi di default sono: ON e NO DATA.

Quando si specifica OFF, il successivo operando non ha alcun effetto. Poichè in più punti del sorgente possono apparire istruzioni PRINT, se in una frase un operando è omesso, si assume che l'opzione relativa rimanga invariata rispetto all'ultima precedente specificazione.

Esempio

```
                ...
                ...
                ...
NUM             DS      D
TRAT            DC      PL6'Ø'
CIFRES          DC      PL5'Ø'
IMPON           DC      PL4'Ø'
ATTPRE          DC      PL7'Ø'
                ...
                ...
                ...
                PRINT   ON, NO DATA
                ...
                ...
                ...
                END
```

SPACE

Descrizione Funzionale

Ha la funzione di inserire nel listing una o più interlinee di spaziatura.

Formato

nome	operazione	operando
b	**SPACE**	**[valore decimale]**

Descrizione

Nome

Non ammesso

Operandi

Questo campo è opzionale; se non specificato viene eseguita una sola interlinea; se specificato vengono eseguite un numero di interlinee pari al valore decimale specificato nel campo. Se tale valore supera il numero delle righe rimanenti della pagina che si sta stampando, questa istruzione si comporta come l'istruzione EJECT.

Esempio

```
SEZ     START   1Ø
        ...
        ...
        ...
        USING   BASE, 1Ø
        SPACE   5
        ...
        ...
        ...
BASE    DS      D
        ...
        ...
        ...
        END
```

START

Descrizione Funzionale

Identifica l'inizio della sezione di controllo. Tutte le frasi che seguono sono assemblate come facenti parte di questa sezione, finchè non venga incontrata una frase che segni l'inizio di una nuova sezione o di un'altro modulo sorgente (CSECT, DSECT, END).

Formatc

nome	operazione	operando
[simbolo]	START	[termine autodefinito]

Descrizione

Nome

Se la frase è contrassegnata da un nome simbolico, questo indica l'indirizzo del 1° byte del modulo oggetto. A questo simbolo viene assegnato l'attributo lunghezza 1; esso è rilocabile.

Operandi

E' un campo opzionale; se presente specifica il valore iniziale da assegnare al Location Counter della sezione. Se tale valore non è divisibile per 4, l'Assembler lo arrotonda per eccesso in modo da garantire l'allineamento dei vari elementi oggetto della sezione. Se l'operando è omesso (vedi CSECT) il Location Counter è settato a zero.

L'istruzione START non identifica l'inizio dell'Assemblaggio e la prima sezione di controllo del programma ma semplicemente una sezione di controllo. L'istruzione START si differenzia dalla CSECT perchè in essa è consentito indicare il valore iniziale da assegnare al Location Counter (stampato nel listing).

Esempio

```
PROG1     START    2Ø4Ø
PROG2     START
PROG3     START    X'7F8'
          START
          ...
          ...
          ...
          END
```

TITLE

Descrizione Funzionale

Viene usata per identificare un listing Assembler. Provoca la stampa dell'intestazione voluta sulle pagine di tabulato che seguono fino alla lettura di un'altra frase TITLE. A fronte di questa frase viene eseguito il salto pagina.

Formato

nome	operazione	operando
b	**TITLE**	**'character list'**

Descrizione

Nome

Non ammesso

Operandi

Una sequenza di caratteri racchiusa tra apici. Possono essere espressi fino a 1ØØ caratteri, e se nella sequenza l'utente vuole esprimere il carattere apici deve indicarlo 2 volte.

Esempio

```
         START
PGM1     TITLE    'PRIMA COMPILAZIONE'
         ...
         ...
         ...
INIT     BALR     11,Ø
         USING    *, 11
         BC       15, GO
         ...
         ...
         ...
         END
```

USING

Descrizione Funzionale

Indica all'Assemblatore il Registro generale disponibile da utilizzare come Registro Base nelle istruzioni esecutive e il valore da caricare in tale registro.

Formato

nome	operazione	operando
b	**USING**	$V, r_1 [, r_2, \ldots . r_n]$

Descrizione

Nome

Non ammesso.

Operandi

La lettera V sta ad indicare una espressione assoluta o rilocabile; r1, r2,...rn sono espressioni assolute che designano i registri generali, quindi il valore di queste espressioni deve essere compreso tra Ø ÷ 15, e ne sono consentite al max 16.

In seguito a questa istruzione, l'Assembler assume che r1 sia caricato con il valore dell'espressione assoluta o rilocabile V, r2 sia caricato con V + 4Ø96... rn con V + (n - 1) * 4Ø96. Nell'istruzione USING in cui compaia un registro già menzionato in una USING precedente, annulla l'effetto della USING precedente per quanto riguarda quel registro. Se l'utente specifica il registro generale Ø come operando r1, l'Assemblatore assumerà che il contenuto di tale registro sia Ø, indipendentemente dal valore dell'espressione V. Inoltre nei registri r2...r16 vengono assunti contenenti V + 4Ø96, V + 8192.... Un programma che usi il registro Ø come registro base non è rilocabile.

Esempio

```
LARGE    START    Ø
         BALR     9, Ø
         USING    POD, 9, 1Ø, 12
POD      LM       1Ø, 11, BASE
         B        INIT
BASE     DC       A (POD + 4Ø96)
         DC       A (POD + 8192)
INIT     LA       3, 1 (14)
         ...
         ...
         ...
         END
```

Macro-istruzioni

Le macro-istruzioni comandano all'Assemblatore di inserire particolari sequenze di istruzioni macchina oggetto in base alle informazioni contenute nella stessa macro-istruzione; i loro formati e le loro descrizioni sono i seguenti.

PROC

Descrizione Funzionale

Questa macro-istruzione serve per generare il prologo del modulo, cioè per allocare al modulo in questione un'area LOCALE, e per il salvataggio in tale area dei registri di lavoro del modulo stesso. Se presente deve immediatamente seguire l'istruzione CSECT/START.

Formato

nome	operazione	operando
[simbolo]	**PROC**	**[RX], [RY] [, ENDLOC = simbolo 1] [, STARLOC = simbolo 2] [, PROLOG = {YES / NO}]**

Descrizione

Nome

E' opzionale; se presente corrisponde al nome del modulo oggetto. Se assente al modulo viene assegnato il nome della CSECT/START.

Operandi

- RX e RY sono operandi posizionali opzionali. Se presenti vengono espressi mediante termini autodefiniti decimali il cui valore deve essere compreso nell'intervallo Ø ÷ 15. Indicano gli estremi del gruppo di registri generali il cui contenuto deve essere salvato nell'area LOCALE (questi registri sono considerati in modo circolare).

- ENDLOC = simbolo 1; è un operando opzionale a parola chiave che specifica se presente il nome del 1° byte successivo all'area locale del modulo. Se non indicato l'Assembler assume che tale nome sia ENDLOCAL.

- STARLOC = simbolo 2; è un operando opzionale a parola chiave che specifica il nome dell'area locale. Se non indicato, viene assunto come nome dell'area locale LOCAL.

- PROLOG = $\left\{\frac{\text{YES}}{\text{NO}}\right\}$; è un operando opzionale che specifica all'Assembler se il prologo deve essere generato o no. Il valore di default è PROLOG = YES, quindi per esigenze normali questo operando deve essere omesso.

Espansione

```
[Simbolo]  LR     Ø, 13
           LA     13, {ENDLOCAL / Simbolo 1} - {LOCAL / Simbolo 2}
           ASA    13, 13
           ST     Ø, Ø (, 13)
          [STM    RX, RY, 4 (13)]
```

Le istruzioni macchina elencate nella espansione sono valorizzate dagli operandi espressi nella frase sorgente. Nel caso della frase STM, essa viene generata solo se nella frase sorgente sono indicati gli operandi RX e RY.

Esempi

1.
```
POP     PROC    ,,STARLOC = BEGLOC
```

Viene generata la seguente sequenza di istruzioni:

```
POP     LR      Ø, 13
        LA      13, ENDLOCAL - BEGLOC
        ASA     13, 13
        ST      Ø, Ø (, 13)
```

2.
```
POP     PROC     2, 3
```

Viene generata la seguente sequenza:

```
POP     LR      Ø, 13
        LA      13, ENDLOCAL - LOCAL
        ASA     13, 13
        ST      Ø, Ø (, 13)
        STM     2, 3, 4 (13)
```

PRRET

Descrizione Funzionale

Questa macro-istruzione serve per generare l'epilogo del modulo; essa viene espansa con le istruzioni complementari rispetto a quelle generate a fronte della PROC, che deallocano l'area LOCALE e ripristinano i registri di lavoro eventualmente salvati.

Formato

nome	operazione	operando
[simbolo]	**PRRET**	**[I]**

Descrizione

Nome

Può contenere un simbolo.

Operandi

Può essere indicato opzionalmente un termine autodefinito decimale il cui valore deve essere compreso nell'intervallo Ø ÷ 255. Questo termine indica il displacement di rientro. Se omesso si assume che il rientro abbia displacement Ø.

Espansione

```
[Simbolo] [LM      RX, RY, 4 (13)]
          L       13, Ø (, 13)
          LA      Ø, {ENDLOCAL | Simbolo 1} - {LOCAL | Simbolo 2}
          FSA     Ø, Ø
          RETEXT  [I]
```

Se precedentemente non è stata generato un prologo (PROC), l'espansione della PRRET è:

```
[nome]    RETEXT    [I]
```

Esempio

Se nel prologo generato dalla PROC non era incluso il salvataggio dei registri, viene omessa l'istruzione di ripristino dei registri (LM).

```
PUP     L       13, Ø (, 13)
        LA      Ø, ENDLOCAL - BEGLOC
        FSA     Ø, Ø
        RETEXT
```

Descrizione Funzionale — Somma del contenuto di una parola con il contenuto di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**A**	**R_1, D_2 (X_2, B_2)** **R_1, D_2 (, B_2)** **R_1, S_2 (X_2)** **R_1, S_2**	**RX**	**5A**

Azione — Il contenuto dell'area di memoria indirizzata dal 2° operando (una parola) è addizionato al primo operando ed il risultato è posto nel primo operando. L'operazione è eseguita sommando i 32 bit dei due operandi, segni compresi. Un overflow si verifica quando i riporti del segno e del bit di ordine più alto sono discordanti (riporto del segno uguale a 1 e riporto della posizione più alta uguale a Ø, o viceversa).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " < Ø
2 " " " " " " > Ø
3 se si verifica un'overflow

Esempio

A 2,100(,4)

dove:

1° operando reg. 2 il cui valore esadecimale è Ø1Ø8BA

2° operando 1ØØ(,4) " " " " " ØØ68

risultato reg. 2 " " " " " Ø1Ø922

1° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = parola

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

risultato = R1

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØ1 ØØ1ØØØ1Ø

ADD HALFWORD

AH

Descrizione Funzionale

Somma il contenuto di una mezza parola con il contenuto di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	AH	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	4A

Azione

Il secondo operando è addizionato al primo e la somma è posta nel primo operando. Prima dell'operazione il segno della mezza parola viene espanso per una lunghezza di 16 bit; l'operazione è quindi eseguita sommando i 32 bit dei due operandi segni compresi. Un overflow si verifica quando i riporti del segno e del bit di ordine più alto sono discordanti (riporto del segno uguale a 1 e riporto della posizione più alta uguale a Ø, o viceversa).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " "< Ø
2 " " " " " "> Ø
3 se si verifica un'overflow

Esempio

AH 2,50 (2,5)

dove:

1° operando reg. 2 il cui valore esadecimale è Ø1Ø8BA

2° operando 5Ø(2,5) " " " " " ØØ68

risultato reg. 2 " " " " " Ø1Ø922

1° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = mezza parola con espansione del segno

ØØØØØØØØ	ØØØØØØØØ	ØØØØØØØØ Ø11Ø1ØØØ

ESPANSIONE

risultato = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØ1 ØØ1ØØØ1Ø

Descrizione Funzionale

Somma logica del contenuto di una parola con il contenuto di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**AL**	**$R_1, D_2 (X_2, B_2)$** **$R_1, D_2 (, B_2)$** **$R_1, S_2 (X_2)$** **R_1, S_2**	**RX**	**5E**

Azione

Il secondo operando è addizionato al primo ed il risultato è posto nel primo operando. L'operazione è eseguita sommando i 32 bit dei due operandi, segni compresi. Se si verifica un riporto il Condition-Code viene posto ai seguenti valori.

C.C.: Ø se il risultato è = Ø e non c'è riporto
1 " " " " ≠ Ø " " " "
2 " " " " = Ø e c'è riporto
3 " " " " ≠ Ø " " "

Esempio

AL 3,100(,4)

dove:

1° operando reg. 3 il cui valore esadecimale è Ø1Ø8BA

2° operando 1ØØ(,4) " " " " " ØØ8Ø68

risultato reg. 3 " " " " " Ø18922

1° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = parola di memoria

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

risultato = R1

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØ1 ØØ1ØØØ1Ø

Descrizione Funzionale — Somma logica di due campi di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	ALM	$D_1 (L_1, B_1), D_2 (L_2, B_2)$ $S_1 (L_1), S_2 (L_2)$ S_1, S_2 $D_1 (, B_1), D_2 (, B_2)$	SS	FB

Azione

Il campo di memoria indirizzato dal secondo operando è sommato al campo di memoria indirizzato dal primo operando. Il risultato viene posto nel primo operando. Se il secondo operando è più corto del primo viene "logicamente" esteso con zeri a sinistra. Gli operandi vengono allineati a destra e l'operazione procede da destra a sinistra per il numero di byte indicato in L1.

C.C.: Ø se il risultato è = Ø senza riporto
1 " " " " ≠ Ø " "
2 " " " " = Ø con riporto
3 " " " " ≠ Ø " "

Esempio

ALM 50 (2,4), 20 (1,13)

dove:

1° operando 5Ø(2,4) il cui valore esadecimale è Ø8BA

2° operando 2Ø(1,13) " " " " " ØØ68

risultato 5Ø(2,4) " " " " " Ø922

1° operando = campo di memoria

ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = campo di memoria

ØØØØØØØØ	Ø11Ø1ØØØ

ESTENSIONE LOGICA

risultato = campo di memoria

ØØØØ1ØØ1	ØØ1ØØØ1Ø

Descrizione Funzionale

Somma logicamente il contenuto di un registro al contenuto di un altro registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**ALR**	**R_1, R_2**	**RR**	**1E**

Azione

Il contenuto del registro R2 è sommato al contenuto del registro R1 ed il risultato è posto in R1. L'operazione è eseguita sommando i 32 bit dei due operandi, segni compresi. Se si verifica un riporto dalla posizione del segno, il Condition Code viene posto ai seguenti valori.

C.C.: Ø se il risultato è = Ø e non c'è riporto
1 " " " " ≠ Ø " " " "
2 " " " " = Ø e c'è riporto
3 " " " " ≠ Ø " " "

Esempio

ALR 4,5

dove:

1° operando reg. 4 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 5 " " " " " ØØ8Ø68

risultato reg. 4 " " " " " Ø18922

1° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = R2

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

risultato = R1

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØ1 ØØ1ØØØ1Ø

ADD LOGICAL REGISTER IMMEDIATE

Descrizione Funzionale — Somma un immediato ad un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	ALRI	R_1, I_2	RR	01

Azione

Somma al registro R1 (1° operando) l'immediato indicato nei 4 bit costituenti il 2° operando. Questa somma non dà segnalazioni di overflow ma si segnala lo eventuale riporto.

C.C.: 1 se la somma senza riporto è $\neq \emptyset$
2 " " " con " " $= \emptyset$
3 " " " " " " $\neq \emptyset$

Esempio

ALRI 5,X'03

dove:

1° operando reg. 5 il cui valore esadecimale è Ø1Ø8BA

2° operando x'Ø3' " " " " " Ø3

risultato reg. 5 " " " " " Ø1Ø8BD

1° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = I2

ØØ11

risultato = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø1111Ø1

Descrizione Funzionale — Somma algebrica di due campi di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	AM	D_1 (L_1, B_1), D_2 (L_2, B_2) S_1 (L_1), S_2 (L_2) S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	FA

Azione

Il campo di memoria indirizzato dal secondo operando è addizionato al campo di memoria indirizzato dal primo operando. L'addizione è fatta in modo algebrico, cioè tenendo conto del segno e il risultato è posto nel primo operando. Il bit del segno è il bit più pesante del byte più pesante del campo su cui si opera. Il campo più corto viene logicamente esteso espandendo il bit del segno e l'operazione procede dal byte meno pesante a quello più pesante. Si verifica overflow:

- se i riporti del segno e del bit di ordine più alto sono discordanti

- se il campo in cui viene depositato il risultato non è sufficiente a contenerlo.

In caso di overflow gli L1 bytes meno pesanti del risultato vengono assegnati al campo di arrivo.

C.C.: Ø se il risultato è = Ø
1 " " " " < Ø
2 " " " " > Ø
3 se si verifica overflow

Esempio

AMI 100 (2,4), 50 (1,6)

dove:

1° operando 1ØØ(2,4) il cui valore esadecimale è Ø8BA

2° operando 5Ø(1,6) " " " " " ØØ68

risultato 1ØØ(2,4) " " " " " 8922

1° operando = campo di memoria

ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = campo di memoria

ØØØØØØØØ	Ø11Ø1ØØØ

ESPANSIONE LOGICA DEL SEGNO

risultato = campo di memoria

1ØØØ1ØØ1	ØØ1ØØØ1Ø

Descrizione Funzionale

Somma il contenuto di un registro al contenuto di un altro registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**AR**	**R_1,R_2**	**RR**	**1A**

Azione

Il contenuto del registro R2 è sommato al contenuto del registro R1 ed il risultato è posto in R1. L'operazione è eseguita sommando i 32 bit dei due operandi, segni compresi. Un overflow si verifica quando i riporti del segno e del bit di ordine più alto sono discordanti (riporto del segno uguale ad 1 e riporto della posizione più alta uguale a Ø, o viceversa). Ponendo R1 = R2 si ottiene come risultato il raddoppio del contenuto del registro.

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " < Ø
2 " " " " " " > Ø
3 se si verifica un overflow

Esempio

AR 5,4

dove:

1° operando reg. 5 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 4 " " " " " ØØ8Ø68

risultato reg. 5 " " " " " Ø18922

0° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = R2

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

risultato = R1

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØ1 ØØ1ØØØ1Ø

ALLOCATE STACK AREA

ASA

Descrizione Funzionale — Alloca un'area sullo stack.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**ASA**	**R_1, R_2**	**RR**	**23**

Azione

Alloca sullo stack un'area pari al numero di byte indicato dal contenuto del registro R2 e passa nel registro R1 l'indirizzo aggiornato della cima dello stack (TOP OF STACK). Se gli ultimi 4 bit meno significativi del registro R2 sono a Ø non viene allocata alcuna area ed in R1 si ottiene il valore della cima della stack.

C.C.: non gestito

BRANCH AND LINK

BAL

Descrizione Funzionale — Salto e memorizzazione di indirizzo.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	BAL	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	45

Azione

Memorizza nel registro R1 l'indirizzo della prossima istruzione, ed esegue un salto all'indirizzo espresso dal secondo operando.

C.C.: non gestito

BRANCH AND LINK REGISTER

BALR

Descrizione Funzionale — Salto e memorizzazione di indirizzo.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**BALR**	**R_1, R_2**	**RR**	**05**

Azione

Ha le stesse funzioni della BAL, con la caratteristica che l'indirizzo espresso dal secondo operando è il contenuto di un registro, invece che essere espresso in forma esplicita.

C.C.: non gestito

Descrizione Funzionale

Test del Condition Code e salto condizionato all'indirizzo definito nell'istruzione.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**BC**	**M_1, D_2 (X_2, B_2)** **M_1, D_2 (, B_2)** **M_1, S_2 (X_2)** **M_1, S_2**	**RX**	**47**

Azione

La maschera presente nella istruzione permette di testare il valore del Condition Code, e specifica per quali condizioni si esegue il salto. Se queste condizioni sono soddisfatte il controllo passa all'istruzione avente come indirizzo il secondo operando, altrimenti viene eseguita la prossima istruzione in sequenza.

La maschera è un campo formato da 4 bit aventi la seguente corrispondenza con il Condition Code:

C.C.		Maschera
0	→	8
1	→	4
2	→	2
3	→	1

Tabella di corrispondenza

Un branch condizionato esegue un salto solo se il valore del Condition Code (secondo la tabella) corrisponde alla maschera definita.

Esempio:

BC 8, COLM salta se C.C. = Ø

In una maschera si possono sommare più condizioni.
Esempio:

BC 9, COLM salta per C.C. = Ø oppure C.C. = 3

Un branch incondizionato equivale a una condizione di maschera uguale a 15 (salta sempre).

Nel seguito, per ragioni di sintesi, si dà un elenco dei codici mnemonici estesi, che permettono di scrivere i salti condizionati senza doverne specificare la condizione; essi rappresentano quindi sia la funzione di macchina che la condizione che provoca il salto stesso. I codici mnemonici estesi sono tradotti dall'Assemblatore nelle corrispondenti combinazioni di codice di operazione (EC o BCR) e maschera. La tabella che segue elenca appunto questi codici estesi e le corrispondenti istruzioni macchina:

CODICE ESTESO	SIGNIFICATO	ISTRUZIONE MACCHINA EQUIVALENTE
B D2(X2,B2)	Branch	BC 15,D2(X2,B2)
BR R2	Branch(RR format)	BCR 15,R2
NOP D2(X2,B2)	No Operation	BC Ø,D2(X2,B2)
NOPR R2	No operation(RR format)	BCR Ø,R2
BH D2(X2,B2)	Branch on High	BC 2,D2(X2,B2)
BL D2(X2,B2)	Branch on Low	BC 4,D2(X2,B2)
BE D2(X2,B2)	Branch on Equal	BC 8,D2(X2,B2)
BNH D2(X2,B2)	Branch on Not High	BC 13,D2(X2,B2)
BNL D2(X2,B2)	Branch on Not Low	BC 1Ø,D2(X2,B2)
BNE D2(X2,B2)	Branch on Not Equal	BC 6,D2(X2,B2)
BO D2(X2,B2)	Branch on Overflow	BC 1,D2(X2,B2)
BP D2(X2,B2)	Branch on Plus	BC 2,D2(X2,B2)
BM D2(X2,B2)	Branch on Minus	BC 4,D2(X2,B2)
BZ D2(X2,B2)	Branch on Zero	BC 8,D2(X2,B2)
BNP D2(X2,B2)	Branch on Not Plus	BC 13,D2(X2,B2)
BNM D2(X2,B2)	Branch on Not Minus	BC 10,D2(X2,B2)
BNZ D2(X2,B2)	Branch on Not Zero	BC 6,D2(X2,B2)
BNO D2(X2,B2)	Branch on Not Ones	BC 14,D2(X2,B2)

BRANCH ON CONDITION REGISTER

BCR

Descrizione Funzionale

Test del Condition Code e salto condizionato all'indirizzo definito nell'istruzione.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**BCR**	**M_1, R_2**	**RR**	**07**

Azione

La maschera presente nella istruzione permette di testare il valore del Condition Code, e specifica per quali condizioni si esegue il salto. Se queste condizioni sono soddisfatte viene eseguito un salto all'istruzione avente come indirizzo il contenuto di R2 (secondo operando), altrimenti viene eseguita la prossima istruzione in sequenza (per quanto riguarda le combinazioni di maschera, vedere istruzione BC).

Descrizione Funzionale

Salto per registro diverso da zero previo decremento di uno del suo contenuto.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	BCT	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	46

Azione

Il contenuto del registro R1 viene diminuito di 1; se il valore risultante è diverso da zero viene eseguito un salto all'istruzione il cui indirizzo è espresso dal secondo operando. L'indirizzo di salto viene determinato prima della diminuzione del registro R1. Il salto avviene anche se il registro R1 contiene un numero negativo. Nel caso che il registro contenga il max numero negativo rappresentabile in un registro, dopo la diminuzione conterrà il max numero positivo rappresentabile in un registro essendosi inoltre verificato il salto. L'overflow relativo non è segnalato. Se in partenza R1 contiene Ø, il salto si verifica ed il registro viene decrementato a -1.

C.C.: non gestito

BRANCH ON COUNT REGISTER

BCTR

Descrizione Funzionale

Salto per registro diverso da zero, previo decremento di uno del suo contenuto.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**BCTR**	**R_1, R_2**	**RR**	**06**

Azione

Il contenuto del registro R1 viene diminuito di 1; se il valore risultante è diverso da zero viene eseguito un salto all'istruzione il cui indirizzo è espresso dal secondo operando. L'indirizzo di salto viene determinato prima della diminuzione del registro R1. Se si pone R2 = Ø il salto non avviene, ed il contenuto del registro R1 è diminuito di 1. Il salto avviene anche se il registro R1 contiene un numero negativo. Nel caso che il registro contenga il max numero negativo, dopo la diminuzione conterrà il massimo numero positivo essendosi inoltre verificato il salto. L'overflow relativo non è segnalato. Se in partenza R1 contiene Ø, il salto si verifica ed il registro viene decrementato a -1.

C.C.: non gestito

BRANCH ON INDEX HIGH

BXH

Descrizione Funzionale

Salto se la somma del contenuto di due registri è maggiore del contenuto di un terzo.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	BXH	$R_1, R_2, D_3 (B_3)$ R_1, R_2, S_3	RS	86

Azione

Il registro R1 (indice) viene incrementato col valore contenuto nel registro R2 (incremento). Il contenuto di R1 viene poi confrontato col contenuto di un terzo registro immediatamente seguente R2 (limite). Se il contenuto dell'indice è maggiore del limite, si salta all'istruzione il cui indirizzo è espresso dal terzo operando; se è minore o uguale si procede in sequenza. L'indirizzo di salto viene determinato prima dell'incremento del registro indice. Tutti i bit dei registri interessati partecipano all'operazione. Le quantità che intervengono sono considerate numeri binari (positivi o negativi).

C.C.: non gestito

BRANCH ON INDEX LOW OR EQUAL

BXLE

Descrizione Funzionale

Salto se la somma del contenuto di due registri è minore o uguale al contenuto di un terzo.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	BXLE	$R_1, R_2, D_3 (B_3)$ R_1, R_2, S_3	RS	87

Azione

Il registro R1 (indice) viene incrementato col valore contenuto nel registro R2 (incremento). Il contenuto di R1 viene poi confrontato col contenuto di un terzo registro immediatamente seguente R2 (limite). Se il contenuto dell'indice è $\leqslant$ del limite, si salta alla istruzione il cui indirizzo è espresso dal terzo operando; se è $>$ si procede in sequenza. L'indirizzo di salto viene determinato prima dell'incremento del registro indice. Tutti i bit dei registri interessati partecipano all'operazione. Le quantità che intervengono sono considerate numeri binari (positivi o negativi).

C.C.: non gestito

Descrizione Funzionale

Confronta algebricamente il contenuto di un registro con il contenuto di una parola.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	C	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	59

Azione

Il contenuto del registro R1 è confrontato algebricamente al contenuto della parola rappresentata dal secondo operando. Il risultato del confronto determina lo stato del Condition Code ed è riferito a R1.

C.C.: Ø se contenuto di R1 = contenuto secondo operando
1 " " " " < " " "
2 " " " " > " " "

Descrizione Funzionale

Salto con memorizzazione dell'indirizzo di rientro e aggiornamento della lista degli argomenti.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CALL	$D_1 (B_1), D_2 (L_2, B_2)$ $S_1, S_2 (L_2)$ S_1, S_2 $D_1 (B_1), D_2 (,B_2)$ $D_1 (B_1) [(.... D_n (B_n))]$ $S_1 [(.... S_n)]$	SI	9A

Azione

Il primo operando è l'indirizzo della istruzione a cui eseguire il salto. Il secondo operando (1 o più di 1) costituisce la lista degli argomenti. Dopo la esecuzione della istruzione il registro generale 1 punta ad un'area di stack n+2 parole (n è il numero degli argomenti 31) contenenti rispettivamente e nell'ordine:

1. prima parola: nel byte più pesante il numero di argomenti (n) e nei tre meno pesanti l'indirizzo della istruzione di rientro seguente la CALL.

2. seconda parola: informazioni per il sistema operativo.

3. terza parola: nel byte più pesante Ø e nei tre meno pesanti l'indirizzo del 1° argomento.

4. quarta parola: nel byte più pesante Ø e nei tre meno pesanti l'indirizzo del secondo argomento.

5. quinta ÷ n^{esima} parola: nel byte più pesante Ø e nei tre meno pesanti l'indirizzo del 5° ÷ n^{esimo} argomento.

Se la CALL viene eseguita a seguito di un EXEC l'in-

dirizzo memorizzato per primo nella lista di n+2 parole è quello della istruzione seguente l'EXEC.

Se n = Ø il registro generale 1 punta ad una lista di 2 parole contenente solo gli elementi dei punti 1 e 2.

C.C.: non gestito

CBS

Descrizione Funzionale

Converte il contenuto di una mezza parola nel corrispondente numero decimale in caratteri ISO.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CBS	D_1 (L, B_1), D_2 (B_2) S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	DB

Azione

Il primo operando contiene l'indirizzo del campo di memoria in cui andranno depositati i caratteri generati dalla conversione. Il secondo operando contiene l'indirizzo della mezza parola il cui contenuto è da convertire. Alla fine della conversione il registro generale RØ contiene il numero di caratteri generati. Se si esaurisce il campo di lunghezza L prima di aver terminato la conversione, il campo di arrivo contiene i primi L caratteri generati e il registro generale RØ il valore L.

C.C.: Ø conversione effettuata
1 " "
2 " "
3 esaurito il campo di lunghezza L prima della fine della conversione.

Esempio

2° operando mezza parola = 41

rappresentazione binaria ØØØØØØØØ Ø1ØØØØØ1
rappresentazione decimale 65
rappresentazione decimale in caratteri ISO A

risultato in 1° operando = 36 35

COMPARE HALFWORD

CH

Descrizione Funzionale

Confronta algebricamente il contenuto di un registro con una mezza parola.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CH	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	49

Azione

La mezza parola indirizzata dal secondo operando viene estesa a 32 bit mediante propagazione a sinistra del segno. Il contenuto del registro R1 è confrontato algebricamente con la mezza parola (estesa) contenuta nel secondo operando.

C.C.: Ø se contenuto di R1 = contenuto secondo operando
1 " " " " < " " "
2 " " " " > " " "

Descrizione Funzionale

Confronta logicamente il contenuto di un registro con il contenuto di una parola.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CL	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, D_2 (X_2)$ R_1, D_2	RX	55

Azione

Il contenuto di R1 è confrontato logicamente al contenuto della parola espressa dal secondo operando. Il risultato del confronto è riferito ad R1 e determina lo stato del Condition Code. I termini vengono confrontati bit a bit da sinistra verso destra (segno compreso); l'operazione si arresta quando vengono trovati due bit diversi. In tal caso viene considerato maggiore quello dei due termini al quale appartiene il bit 1.

C.C.: Ø se contenuto R1 = contenuto secondo operando
1 " " " < " " "
2 " " " > " " "

Descrizione Funzionale — Confronta logicamente due campi di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CLC	D_1 (L, B_1), D_2 (B_2) S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	D5

Azione

Il contenuto del primo operando è confrontato logicamente con il contenuto del secondo operando. Il confronto inizia da sinistra e procede, byte per byte, verso destra, per la lunghezza specificata al primo operando. Il confronto non tiene conto del segno degli operandi. L'operazione si arresta non appena vengono trovati due bit diversi. Il risultato del confronto determina lo stato del Condition Code.

C.C.: Ø se 1° operando = secondo operando
1 " " " < " "
2 " " " > " "

Descrizione Funzionale

Confronta un byte di memoria con un byte contenuto nell'istruzione.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CLI	$D_1 (B_1), I_2$ S_1, I_2	SI	95

Azione

Il contenuto del byte primo operando è confrontato logicamente con il secondo byte della istruzione. Il risultato del confronto determina lo stato del Condition Code. Il confronto avviene come per l'istruzione CLC.

C.C.: Ø se contenuto del 1° operando = I2
1 " " " " " < I2
2 " " " " " > I2

Descrizione Funzionale — Confronta due campi di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CLM	D_1 (L_1, B_1), D_2 (L_2, B_2) S_1 (L_1), S_2 (L_2) S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	F5

Azione

La zona di memoria indirizzata dal 1° operando viene confrontata con con la zona di memoria indirizzata dal 2° operando. Il risultato dell'istruzione determina lo stato del Condition Code. Gli operandi vengono allineati a destra, e l'operazione procede da sinistra a destra per il numero di byte indicato in L1. Se il secondo operando è più corto viene inteso "logicamente" esteso con zeri a sinistra.

C.C.: Ø i 2 operandi sono uguali
1 il 1° operando è < del 2°
2 il 1° operando è > del 2°

COMPARE LOGICAL REGISTER

CLR

Descrizione Funzionale — Confronta logicamente il contenuto di due registri.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**CLR**	**R_1, R_2**	**RR**	**15**

Azione

Il contenuto del registro R1 è confrontato logicamente con il contenuto del registro R2. Il risultato della operazione determina lo stato del Condition Code. I due termini vengono confrontati bit per bit da sinistra verso destra (segno compreso) e l'operazione si arresta in presenza di due bit diversi (Ø ed 1). Viene considerato maggiore quello dei due termini al quale appartiene il bit a 1.

C.C.: Ø se contenuto R1 = contenuto R2
1 " " " < " "
2 " " " > " "

COMPARE MEMORY

CM

Descrizione Funzionale — Confronto algebrico di due campi di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CM	$D_1 (L_1, B_1), D_2 (L_2, B_2)$ $S_1 (L_1), S_2 (L_2)$ S_1, S_2 $D_1 (, B_1), D_2 (, B_2)$	SS	F9

Azione

La zona di memoria indirizzata dal 1° operando è confrontata algebricamente con la zona di memoria indirizzata dal 2° operando; il risultato dell'operazione determina lo stato del Condition Code. L'operando più corto viene espanso logicamente con il bit del segno a sinistra. Gli operandi vengono allineati a destra e il confronto procede da sinistra a destra. Nella comparazione il bit più pesante del byte più pesante di un campo è considerato bit di segno del campo.

C.C.: Ø se i due operandi sono uguali
1 se 1° operando < 2° operando
2 " " " > " "

Descrizione Funzionale — Confronta il contenuto di due registri.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CR	R_1, R_2	RR	19

Azione — Il contenuto di R1 è confrontato algebricamente con il contenuto di R2. Il risultato del confronto determina lo stato del Condition Code.

C.C.: Ø se il contenuto di R1 = al contenuto di R2
1 " " " " " < " " " "
2 " " " " " > " " " "

Descrizione Funzionale

Converte una stringa di caratteri ISO numerici nel corrispondente numero binario.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CSBH	D_1 (B_1), D_2 (L_2, B_2) S_1 , S_2 (L_2) S_1, S_2 D_1 (B_1), D_2 (, B_2)	SS	F6

Azione

Converte in binario senza segno la stringa di caratteri numerici indirizzati dal secondo operando nella semiparola indirizzata dal primo operando. Il numero da convertire è quello specificato dal verificarsi della prima delle seguenti eventualità:

- dopo N cifre si incontra un carattere diverso da cifra; N < L2
- dopo L2 cifre.

Il risultato della conversione è modulo 2^{16} (65536) ed è posto nel primo operando. R1 alla fine dell'operazione contiene l'indirizzo del primo byte della stringa che non è stato convertito.

Se il primo carattere della stringa non è numerico, il primo operando non viene modificato e R1 riceve l'indirizzo del secondo operando. Il numero da convertire deve essere ⩽ 99999. Se è maggiore il primo operando non viene modificato e R1 riceve l'indirizzo del secondo operando.

C.C.: Ø se numero ⩽ 65535
1 " " > "

2 se numero > 99999
3 secondo operando non numerico

Esempio

2° operando = 36 35

rappresentazione decimale in caratteri ISO A
rappresentazione decimale ⟶ 65
rappresentazione binaria ØØØØØØØØ Ø1ØØØØØ1

risultato in 1° operando = 41

Descrizione funzionale

Ricerca tabellare eseguita in modo dicotomico su un sottocampo di un elemento della tabella.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	DIS	D_1 (L_1, B_1), D_2 (L_2, B_2) S_1 (L_1), S_2 (L_2) S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	F8

Azione

Viene fatto un confronto fra un sottocampo della tabella indirizzata dal primo operando e il campo di memoria indirizzato dal secondo operando. La ricerca è dicotomica e la relazione d'ordine è stabilita sul sottocampo da ricercare dalla numerazione binaria pura (senza segno). L1 è la lunghezza di un elemento della tabella. L2 è la lunghezza del secondo operando e quindi anche del sottocampo da confrontare. RØ contiene il numero totale degli elementi della tabella (solo gli ultimi 2 byte a destra sono significativi). R2 contiene l'offset del sottocampo nell'elemento. Si considerano solo i 4 bit meno pesanti di R2. Come risultato della ricerca viene caricato in R1 lo indirizzo dell'elemento ricercato se la ricerca ha esito positivo. Se la ricerca ha esito negativo viene caricato in R1 l'indirizzo dell'elemento immediatamente seguente quello ricercato. Se RØ = Ø si pone il Condition Code a 1 e in R1 l'indirizzo della tabella (1° operando).

C.C.: Ø se l'elemento è trovato =
1 " " " " ≠ e maggiore ultimo
2 " " " " ≠ e minore primo
3 " " " " ≠

Descrizione Funzionale

Divide il contenuto dei registri e memorizza risultato e resto.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**DM**	**R_1, R_2, D_3 (B_3)** **R_1, R_2, S_3**	**RS**	**85**

Azione

Il contenuto dei 2 byte meno significativi del registro R2, viene diviso per il contenuto dei 2 byte meno significativi del registro R1. Il risultato su 2 byte, seguito dal resto anch'esso su 2 byte, è posto in memoria all'indirizzo indicato dal terzo operando. Se i 2 byte meno significativi di R1 sono Ø, il risultato è 65535 ed il resto è il contenuto dei 2 byte meno significativi di R2. I contenuti di R1 e R2 non sono alterati dall'operazione.

C.C.: non gestito

EX

Descrizione Funzionale

Consente l'esecuzione di una singola istruzione situata al di fuori della normale sequenza.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	EX	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	44

Azione

L'istruzione il cui indirizzo è indicato nel secondo operando viene alterata dal contenuto (ultimi 8 bit) del registro R1 ed eseguita. L'alterazione riguarda il solo byte della istruzione che segue il codice operativo (bit 8 ÷ 15), e avviene mediante la somma logica del contenuto del byte suddetto e del contenuto dell'ultimo byte di destra del registro R1 (bit 24 ÷ 31). Questa somma sostituisce il contenuto del byte dell'istruzione solo agli effetti dell'interpretazione e dell'esecuzione, infatti l'istruzione resta inalterata in memoria. Se il contenuto di R1 = al contenuto di RØ non ha luogo alcuna alterazione.

Dopo l'esecuzione della EXECUTE, e della istruzione alla quale la EXECUTE rimanda, il programma procede con l'istruzione immediatamente seguente la EXECUTE. Tutto questo, a meno che l'istruzione fuori sequenza non sia un salto, nel qual caso il contenuto del contatore della istruzione conterrà l'indirizzo di branch, ed il programma procederà dalla istruzione alla quale il salto ha passato il controllo. Se l'istruzione, il cui indirizzo è indicato nel secondo operando e una BAL, BALR o CALL l'indirizzo che viene salvato è quello della istruzione seguente la EXECUTE.

C.C.: gestito dalla istruzione fuori sequenza

Descrizione Funzionale — Dealloca un'area dello stack.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	FSA	R_1, R_2	RR	22

Azione

Dealloca dalla cima dello stack un'area pari al numero di byte indicato dal contenuto di R2. Di R2 si considerano solo i 2 byte meno significativi. Se i 4 bit di minor peso del registro R1 non sono tutti Ø vi si carica il valore aggiornato della cima dello stack (TOS).

C.C.: non gestito

Descrizione Funzionale

Inserisce il contenuto di un byte di memoria in un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	IC	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	43

Azione

Gli 8 bit del carattere indirizzato dal secondo operando sono inseriti nei bit 24 ÷ 31 del registro R1. I rimanenti bit rimangono invariati.

C.C.: non gestito

Descrizione Funzionale

Muove in memoria un immediato per il numero di volte indicato nel registro generale Ø.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	IM	$D_1 (B_1), I_2$ S_1, I_2	SI	9D

Azione

Muove in memoria all'indirizzo espresso dal primo operando, l'immediato (secondo operando) per il numero di byte indicato dai due byte meno pesanti del registro generale Ø. Se i due byte meno pesanti del registro Ø contengono Ø l'immediato viene scritto 65536 volte.

C.C.: non gestito

Descrizione Funzionale — Determina la categoria a cui appartiene un carattere.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	ISO	$D_1 (X_1, B_1)$ $D_1 (, B_1)$ $S_1 (X_1)$ S_1	RX	82

Azione — Confronta il carattere indirizzato con la tabella ISO.

I caratteri alfabetici sono: A, B, C.....Z
I caratteri numerici sono : Ø, 1, 2.....9
Gli operatori sono : +, −, *, /, ↑

Dopo l'operazione il Condition Code viene posto ai seguenti valori:

C.C.: Ø carattere = alfabetico
1 " = numerico
2 " = operatore
3 caratteri rimanenti

Descrizione Funzionale — Caricamento del contenuto di una parola in un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	L	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	58

Azione — Il contenuto della parola, il cui indirizzo è specificato nel secondo operando, è caricato nel registro R1. Il secondo operando segue le normali norme per l'allineamento.

C.C.: non gestito

Esempio

L 5,50 (4,3)

dove:

1° operando reg. 5 il cui valore esadecimale è Ø1Ø8BA

2° operando 5Ø(4,3) " " " " " ØØ8Ø68

risultato reg. 5 " " " " " ØØ8Ø68

1° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = parola

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

risultato in R1 =

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

LOAD ADDRESS

LA

Descrizione Funzionale — Carica in un registro generale un indirizzo.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LA	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	41

Azione — L'indirizzo al secondo operando viene caricato per un massimo di 24 bit nel registro generale R1. Il registro viene caricato da destra; la parte non utilizzata è riempita con zeri.

C.C.: non gestito

LOAD COMPLEMENT

LC

Descrizione Funzionale

Carica un registro con il complemento a due del contenuto di una parola di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LC	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	53

Azione

Il complemento a due del contenuto della parola indirizzata dal secondo operando è caricato nel registro R1. Se il secondo operando contiene zero, tale valore viene caricato in R1 (con segno +). L'overflow è possibile solo nel caso si esegua il complemento del massimo numero negativo (-2^{31}).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " < Ø
2 " " " " " " > Ø
3 se si verifica un overflow

Esempio

LC 6,100 (,4)

dove:

1° operando reg. 6 il cui valore esadecimale è Ø1Ø8BA

2° operando 1ØØ(,4) " " " " " ØØ8Ø68

risultato reg. 6 " " " " " FF7F98

1° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = parola

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

11111111	11111111	Ø1111111	1ØØ11ØØØ

LOAD COMPLEMENT REGISTER

LCR

Descrizione Funzionale

Carica un registro con il complemento a due di un altro registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LCR	R_1, R_2	RR	13

Azione

Il complemento a due del contenuto del registro R2 è caricato nel registro R1. Se R2 contiene zero, tale valore viene caricato in R1 (col segno +). Se R1 = R2 il contenuto di R1 viene cambiato di segno. L'overflow è possibile solo nel caso si esegua il complemento a 2 del massimo numero negativo (-2^{31}) che può essere contenuto in un registro.

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " < Ø
2 " " " " " " > Ø
3 se si verifica un'overflow

Esempio

LCR 3,2

dove:

1° operando reg. 3 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 2 " " " " " ØØ8Ø68

risultato reg. 3 " " " " " FF7F98

1° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = R2

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

11111111	11111111	Ø1111111	1ØØ11ØØØ

LOAD HALFWORD

LH

Descrizione Funzionale

Carica il contenuto di una mezza parola in un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LH	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	48

Azione

Il contenuto della mezza parola, il cui indirizzo è dato dal secondo operando, è caricato nel registro R1. Il secondo operando deve essere allineato alla mezza parola. Il segno della mezza parola (bit di maggior peso) viene espanso fino alla posizione del segno del registro.

C.C.: non gestito

Esempio

LH 7,26 (2,10)

dove:

1° operando reg. 7 il cui valore esadecimale è Ø1Ø8BA

2° operando 26(2,1Ø)" " " " " ØØØØ68

risultato reg. 7 " " " " " ØØØØ68

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = mezza parola con espansione del segno

ØØØØØØØØ	ØØØØØØØØ	ØØØØØØØØ	Ø11Ø1ØØØ

ESPANSIONE ←

risultato in R1 =

ØØØØØØØØ	ØØØØØØØØ	ØØØØØØØØ	Ø11Ø1ØØØ

LOOK FOR IMMEDIATE EQUAL

LIE

Descrizione Funzionale — Ricerca di un carattere in un campo.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LIE	D_1 (B_1), I_2 S_1, I_2	SI	99

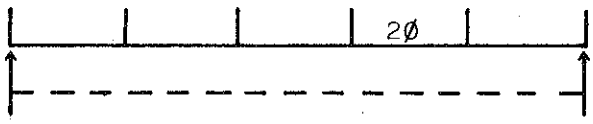
Azione

Viene ricercato il primo carattere uguale all'immediato sul campo di memoria il cui indirizzo è contenuto nel primo operando. La lunghezza massima per cui effettuare la ricerca è implicitamente espressa nel registro generale Ø, di cui si considerano solo i 2 byte meno pesanti. Se la ricerca ha esito positivo il Condition Code è posto a Ø, e viene passato nel registro l'indirizzo del carattere uguale all'immediato. Nel registro Ø si trova la lunghezza residua del campo di memoria. Se la ricerca si esaurisce senza esito positivo, il Condition Code viene posto a 3, il registro generale Ø viene posto a Ø ed il registro generale 1 contiene l'indirizzo del primo byte di memoria successivo al campo analizzato. Se la lunghezza di ricerca contenuta nel registro Ø è uguale a Ø, il Condition Code viene posto a 3, e nel registro generale 1 viene depositato l'indirizzo del campo di memoria specificato come primo operando.

C.C.: Ø se la ricerca dà esito positivo
1 " " " " " "
2 " " " " " "
3 se la ricerca dà esito negativo

Immaginiamo di avere:

RØ = 5 e LIE Ø(6), x'2Ø'



Viene passata in RØ la lunghezza residua del campo; benchè il puntatore punti all'inizio del sottocampo in cui risiede il carattere da ricreare, la lunghezza passata è 1 e non 2.

Descrizione Funzionale

Ricerca del primo carattere diverso da un valore specificato.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LINE	D_1 (B_1), I_2 S_1, I_2	SI	9B

Azione

Viene ricercato il primo carattere diverso dall'immediato nel campo di memoria il cui indirizzo è contenuto nel primo operando. La lunghezza massima per cui effettuare la ricerca è implicitamente espressa nel registro Ø, di cui si considerano solo i due byte meno pesanti. Se la ricerca ha esito positivo il Condition Code è posto a Ø, e in R1 viene depositato lo indirizzo del carattere trovato diverso dall'immediato. In RØ si trova la lunghezza residua del campo di memoria. Se la ricerca si esaurisce perchè si esaurisce il campo di memoria, il Condition Code viene posto a 3, il registro generale Ø viene posto a Ø ed il registro generale 1 contiene l'indirizzo del primo byte di memoria successivo al campo analizzato. Se la lunghezza di ricerca, contenuta nel registro Ø è uguale a Ø, viene posto:

- il Condition Code a 3
- nel registro R1 l'indirizzo del campo di memoria.

C.C.: Ø se la ricerca dà esito positivo
1 " " " " " "
2 " " " " " "
3 se la ricerca dà esito negativo

LOAD MULTIPLE

LM

Descrizione Funzionale

Carica N registri adiacenti con il contenuto di N parole consecutive.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LM	$R_1, R_2, D_3 (B_3)$ R_1, R_2, S_3	RS	98

Azione

Il gruppo di registri consecutivi aventi per estremi i registri R1 e R2 è caricato col contenuto di altrettante parole, a partire dall'indirizzo contenuto nel secondo operando. I registri sono caricati in ordine ascendente, tenendo conto che il registro Ø è consecutivo al registro 15. Se R1 = R2 solo un registro viene caricato.

C.C.: non gestito

Descrizione Funzionale

Carica un registro con il contenuto, reso negativo, di una parola.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LN	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	51

Azione

Il registro R1 viene caricato con il complemento a due del valore assoluto del contenuto della parola indirizzata dal secondo operando. Se il contenuto della parola è zero, questo valore resta immutato e dotato di segno positivo.

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

LN 11,50(,6)

dove:

1° operando reg. 11 il cui valore esadecimale è Ø1Ø8BA

2° operando 5Ø(,6) " " " " " ØØ8Ø68

risultato reg. 11 " " " " " FF7F98

1° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = parola

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

11111111	11111111	Ø1111111	1ØØ11ØØØ

LOAD NEGATIVE REGISTER

LNR

Descrizione Funzionale

Carica un registro con il contenuto, reso negativo, di un altro registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LNR	R_1, R_2	RR	11

Azione

Il registro R1 è caricato con il complemento a 2 del valore assoluto del contenuto del registro R2. Se R2 contiene Ø, questo valore resta immutato e dotato di segno positivo (anche R1 conterrà poi Ø positivo). Se R1 = R2, il contenuto di R1 viene reso negativo.

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " < Ø

Esempio

LNR 9,11

dove:

1° operando reg. 9 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 11 " " " " " ØØ8Ø68

risultato reg. 9 " " " " " FF7F98

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando R2 =

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

11111111	11111111	Ø1111111	1ØØ11ØØØ

LOAD POSITIVE REGISTER

LPR

Descrizione Funzionale

Carica un registro con il valore di un altro registro reso positivo.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LPR	R_1, R_2	RR	10

Azione

Il valore assoluto del contenuto di R2 viene posto in R1. Se il contenuto di R2 è negativo, l'operazione è preceduta da una complementazione a 2. L'overflow è possibile solo quando viene eseguita da una complementazione del massimo numero negativo contenuto in un registro (-2^{31}). Se R1 = R2 il contenuto di R1 viene reso positivo. Nel caso di massimo numero negativo l'istruzione non viene eseguita.

C.C.: Ø se il risultato dell'operazione è = Ø
2 " " " " " " > Ø
3 se si verifica un'overflow

Esempio

LPR 13,6

dove:

1° operando reg. 13 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 6 " " " " " ØØ8Ø68

risultato reg. 13 " " " " " ØØ8Ø68

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando R2 =

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

1ØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

Descrizione Funzionale

Carica un registro con il contenuto di un altro registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**LR**	**R_1, R_2**	**RR**	**18**

Azione

Il contenuto del registro R2 è caricato nel registro R1.

C.C.: non gestito

Esempio

LR 4,2

dove:

1° operando reg. 4 il cui valore esadecimale è ØlØ8BA

2° operando reg. 2 " " " " " ØØ8Ø68

risultato reg. 4 " " " " " ØØ8Ø68

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando R2 =

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

Descrizione Funzionale

Carica un registro con il contenuto di una parola di memoria e gestisce il Condition Code.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LT	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	52

Azione

Il registro R1 è caricato con il contenuto della parola di memoria indirizzata dal secondo operando. Il segno ed il valore assoluto di tale contenuto determinano lo stato del Condition Code.

C.C.: Ø il contenuto del secondo operando è = Ø
1 " " " " " " < Ø
2 " " " " " " > Ø

LOAD AND TEST REGISTER

LTR

Descrizione Funzionale

Carica un registro con il contenuto di un altro registro e gestisce il Condition Code.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LTR	R_1, R_2	RR	12

Azione

Il registro R1 è caricato con il contenuto del registro R2. Il segno ed il valore assoluto di tale contenuto determinano lo stato del Condition Code. Se R1 = R2 il contenuto di R1 resta immutato, ma viene gestito il Condition Code.

C.C.: Ø il contenuto del secondo operando è = Ø
1 " " " " " " < Ø
2 " " " " " " > Ø

MULTIPLY LOGICAL HALFWORD

MLH

Descrizione Funzionale

Moltiplica il contenuto di una mezza parola per il contenuto di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	MLH	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	83

Azione

Il prodotto avviene fra il contenuto della mezza parola allineata indirizzata dal secondo operando, ed il contenuto del registro R1. Il risultato viene messo nel registro R1. Un eventuale overflow non viene segnalato. Il bit più pesante di ciascuno dei due operandi, non viene interpretato come bit di segno. Se uno dei due operandi, o entrambi, sono uguali a zero, si ottiene come risultato uno zero positivo.

C.C.: non gestito

Esempio

MLH 5,60 (,6)

dove:

1° operando reg. 5 il cui valore esadecimale è Ø1Ø8BA

2° operando 6Ø(,6) " " " " " ØØØØØ3

risultato reg. 5 " " " " " Ø31A2E

1° operando R1 =

ØØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = mezza parola

ØØØØØØØØ	ØØØØØØ11

risultato in R1 =

ØØØØØØØØ	ØØØØØØ11	ØØØ11Ø1Ø	ØØ1Ø111Ø

MULTIPLY LOGICAL REGISTER

MLR

Descrizione Funzionale

Moltiplica in modo logico il contenuto di un registro per il contenuto di un altro registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**MLR**	**R_1, R_2**	**RR**	**26**

Azione

Il prodotto avviene fra i contenuti dei registri R1 e R2. Il risultato viene posto in R1. Un'eventuale overflow non viene segnalato. Se uno dei due operandi, o entrambi, sono uguali a zero, si ottiene come risultato uno zero positivo. Se R1 = R2 il risultato della operazione è il quadrato del contenuto del registro.

C.C.: non gestito

Esempio

MLR 4,1

dove:

1° operando reg. 4 il cui valore esadecimale è Ø1Ø8BA

2. operando reg. 1 " " " " " ØØØØØ3

risultato reg. 4 " " " " " Ø31A2E

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando R2 =

ØØØØØØØØ	ØØØØØØØØ	ØØØØØØØØ	ØØØØØØ11

risultato in R1 =

ØØØØØØØØ	ØØØØØØ11	ØØØ11Ø1Ø	ØØ1Ø111Ø

Descrizione Funzionale

Trasferisce il contenuto di una zona di memoria in un'altra zona di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**MVC**	**D_1 (L,B_1), D_2(B_2)** **S_1 (L), S_2** **S_1, S_2** **D_1 (, B_1), D_2 (, B_2)**	**SS**	**D2**

Azione

Un numero di byte pari alla lunghezza L viene mosso dal campo specificato come secondo operando nel campo specificato come primo operando. L è la lunghezza del campo di arrivo. L'istruzione sposta byte per byte, e procede da sinistra verso destra. Il massimo numero di byte che può essere trasferito è 256.

C.C.: non gestito

MVCR

Descrizione Funzionale — Trasferisce un campo di memoria in un altro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	MVCR	D_1 (B_1), D_2 (B_2) S_1, S_2	SS	F2

Azione

L'istruzione muove il campo indirizzato dal secondo operando nel campo indirizzato dal primo operando. Il numero di byte da trasferire è indicato nel registro Ø. Se il contenuto del registro Ø = Ø, l'operazione non ha nessun effetto. Il movimento tra i campi avviene in modo da garantire il salvataggio delle informazioni contenute nel campo di partenza.

C.C.: non gestito

Descrizione Funzionale

Trasferisce un byte dell'istruzione in una posizione di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	MVI	$D_1 (B_1), I_2$ S_1, I_2	SI	92

Azione

Un dato immediato I2 (1 byte), viene trasferito allo indirizzo di memoria specificato dal primo operando.

C.C.: non gestito

Descrizione Funzionale

Trasferisce dal secondo al primo operando solo i semibyte di destra (semibyte numerici).

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	MVN	D_1 (L, B_1), D_2 (B_2) S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	D1

Azione

Per ogni byte del campo indirizzato dal secondo operando, il semibyte di destra sostituisce il rispettivo semibyte dei byte che compongono il campo indirizzato dal primo operando. La lunghezza L che regola il trasferimento è quella del primo operando e può variare da 1 a 256 byte. I semibyte di sinistra del primo operando restano invariati.

C.C.: non gestito

Descrizione Funzionale

Il campo di memoria indirizzato dal secondo operando è trasferito a sinistra degli ultimi 4 bit di destra del campo di memoria indirizzato dal primo operando.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**MVO**	**D_1 (L_1, B_1), D_2 (L_2, B_2)** **S_1 (L_1), S_2 (L_2)** **S_1, S_2** **D_1 (, B_1), D_2 (, B_2)**	**SS**	**F1**

Azione

Il campo di memoria indirizzato dal secondo operando, viene trasferito con uno spostamento a sinistra di un semibyte nel campo indirizzato dal primo operando. Il trasferimento ha luogo da destra a sinistra e per la lunghezza massima di 16 byte. Se L1 < L2 si ha un troncamento. Se L1 > L2 l'eccedenza viene riempita con zeri binari.

C.C.: non gestito

Descrizione Funzionale

Trasferisce i semibyte di sinistra di un campo di memoria in un altro campo.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**MVZ**	**D_1 (L, B_1), D_2 (B_2)** **S_1 (L), S_2** **S_1, S_2** **D_1 (, B_1), D_2 (, B_2)**	**SS**	**D3**

Azione

Per ogni byte del campo indirizzato dal secondo operando, il semibyte di sinistra sostituisce il rispettivo semibyte dei byte che compongono il campo indirizzato dal primo operando. La lunghezza L che regola il trasferimento è quella del primo operando è può variare da 1 a 256 byte. I semibyte di destra del primo operando restano invariati.

C.C.: non gestito

Descrizione Funzionale

Operazione di AND del contenuto di un registro e di una parola.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**N**	**$R_1, D_2 (X_2, B_2)$** **$R_1, D_2 (, B_2)$** **$R_1, S_2 (X_2)$** **R_1, S_2**	**RX**	**54**

Azione

L'AND fra il contenuto del registro R1 e della parola allineata contenuta nel secondo operando, è posto nel registro R1. L'operazione segue bit per bit lo schema della operazione di AND sotto riportata:

AND	Ø	1
Ø	Ø	Ø
1	Ø	1

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio **N 2,50 (,10)**

dove:

1° operando reg. 2 il cui valore esadecimale è Ø1Ø8BA

2° operando 5Ø(,1Ø) " " " " " ØØØØØ3

risultato reg. 2 " " " " " Ø31A2E

1° operando R1 =

|1ØØØØØØØ|ØØØØØØØ1|ØØØ1ØØØ|1Ø111Ø1Ø|

2° operando = parola

|ØØØØØØØØ|ØØØØØØØØ|ØØØØØØØØ|ØØØØØØ11|

risultato in R1 =

|ØØØØØØØØ|ØØØØØØ11|ØØØ11Ø1Ø|ØØ1Ø111Ø|

Descrizione Funzionale — AND fra due campi di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	NC	D_1 (L,B_1), D_2 (B_2) S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	D4

Azione

L'AND FRA il contenuto del campo di memoria del primo operando, di lunghezza L byte, e del campo del secondo operando, pure di L byte, sostituisce il contenuto del primo operando. La lunghezza L è quella associata implicitamente o esplicitamente al campo contenuto nel primo operando. Qualunque configurazione di bit è valida. Le regole seguite sono quelle dell'operazione logica di AND (vedi istruzione N).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

NC 16 (4,11), 50 (4,13)

dove:

1° operando 16(4,11) il cui valore esadecimale è Ø1Ø8BA

2° operando 5Ø(4,13) " " " " " ØØØØØ3

risultato 16(4,11) " " " " " Ø31A2E

1° operando = campo di memoria

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = campo di memoria

ØØØØØØØØ	ØØØØØØØØ	ØØØØØØØØ	ØØØØØØ11

risultato in 1° operando

ØØØ1ØØØØ	ØØØØØØ11	ØØØ11Ø1Ø	ØØ1Ø111Ø

Descrizione Funzionale

AND fra il byte contenuto nella istruzione e un byte di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	NI	$D_1 (B_1), I_2$ S_1, I_2	SI	94

Azione

L'AND fra l'immediato I2 e il byte di memoria, sostituisce il contenuto del primo operando. Qualunque configurazione di bit è valida. Le regole seguite sono quelle della operazione logica di AND (vedi istruzione N).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

NI 30 (5), X' 05'

dove:

1° operando 3Ø(5) il cui valore esadecimale è 1F

2° operando x'Ø5' " " " " " Ø5

risultato 3Ø(5) " " " " " 9B

1° operando = byte di memoria

ØØØ11111

2° operando I2 =

ØØØØØ1Ø1

risultato in 1° operando

1ØØ11Ø11

NO OPERATION

NOP

Descrizione Funzionale — Nessuna esecuzione di operazioni.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**NOP**	**M_1, D_2 (X_2, B_2)** **M_1, D_2 (, B_2)** **M_1, S_2 (X_2)** **M_1, S_2**	**RX**	**470**

Azione — La maschera presente nella istruzione deve essere sempre posta a zero, quindi il Condition Code non viene testato e l'istruzione non esegue alcun salto. Viene quindi eseguita la prossima istruzione in sequenza. Il codice operativo è espresso in codice esadecimale lungo 1 byte e mezzo. Questa istruzione permette di creare spazio per l'introduzione futura di istruzioni (vedere istruzione BC).

C.C.: non gestito

NO OPERATION

NOPR

Descrizione Funzionale — Nessuna esecuzione di operazioni.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**NOPR**	**M_1, R_2**	**RX**	**70**

Azione

La maschera presente nella istruzione deve essere sempre posta a zero, quindi il condition code non viene testato e l'istruzione non esegue alcun salto. Viene quindi eseguita la prossima istruzione in sequenza. Per quanto riguarda codice operativo e maschera vedere istruzione NOP. Questa istruzione permette di creare spazio per l'introduzione futura di istruzioni (vedere istruzione BC).

C.C.: non gestito

NR

Descrizione Funzionale — AND fra il contenuto di due registri.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	NR	R_1, R_2	RR	14

Azione

L'AND fra il contenuto del registro R1 e R2 è messo al posto del registro R1. Qualunque configurazione di bit è valida. Le regole seguite sono quelle della operazione logica di AND (vedi istruzione N).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

NR 5,4

dove:

1° operando reg. 5 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 4 " " " " " ØØØØØ3

risultato reg 5 " " " " " Ø31A2E

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando R2 =

ØØØØØØØØ	ØØØØØØØØ	ØØØØØØØØ	ØØØØØØ11

risultato in R1 =

ØØØØØØØØ	ØØØØØØ11	ØØØ11Ø1Ø	ØØ1Ø111Ø

Descrizione Funzionale — OR fra il contenuto di un registro e di una parola.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**O**	**$R_1, D_2 (X_2, B_2)$** **$R_1, D_2 (, B_2)$** **$R_1, S_2 (X_2)$** **R_1, S_2**	**RX**	**56**

Azione — L'OR fra il contenuto del registro R1 e della parola allineata contenuta nel secondo operando sostituisce il contenuto di R1. Qualunque configurazione di bit è valida. L'operazione segue bit per bit lo schema dell'operazione logica di OR sotto riportata:

OR	Ø	1
Ø	Ø	1
1	1	1

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

O 13,100 (4,6)

dove:

1° operando reg. 13 il cui valore esadecimale è Ø1Ø8BA

2° operando 1ØØ(4,6) " " " " " ØØ8Ø68

risultato reg. 13 " " " " " Ø188FA

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = parola

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØØ	11111Ø1Ø

Descrizione Funzionale — OR fra il contenuto di due campi di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	OC	D_1 (L,B_1), D_2 (B_2) S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	D6

Azione

L'OR fra il contenuto del campo di memoria del primo operando, di lunghezza L byte, e del campo di memoria del secondo operando, pure lungo L byte, sostituisce il contenuto del primo operando. La lunghezza L è quella associata implicitamente o esplicitamente al campo contenuto nel primo operando. Qualunque configurazione di bit è valida. L'operazione segue bit per bit lo schema della operazione logica di OR (vedi istruzione O).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

OC 50 (4,7), 26 (4,11)

dove:

1° operando 5Ø(4,7) il cui valore esadecimale è Ø1Ø8BA

2° operando 26(4,11) " " " " " ØØ8Ø68

risultato 5Ø(4,7) " " " " " Ø188FA

1° operando = campo di memoria

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = campo di memoria

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in 1° operando

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØØ	11111Ø1Ø

Descrizione Funzionale

OR fra un byte contenuto nell'istruzione e un byte di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	OI	$D_1 (B_1), I_2$ S_1, I_2	SI	96

Azione

L'OR fra l'immediato I2 e il byte di memoria indirizzato dal primo operando, sostituisce il contenuto del primo operando. Qualunque configurazione di bit è valida. L'operazione segue bit per bit lo schema della operazione logica di OR (vedi istruzione O).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

OI 10 (10), X'68'

dove:

1° operando 1Ø(1Ø) il cui valore esadecimale è BA

2° operando x'68' " " " " " 68

risultato 1Ø(1Ø) " " " " " FA

1° operando = byte di memoria

1Ø111Ø1Ø

2° operando I2 =

Ø11Ø1ØØØ

risultato in 1° operando

11111Ø1Ø

Descrizione Funzionale — OR del contenuto di due registri.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	OR	R_1, R_2	RR	16

Azione — L'OR fra il contenuto dei registri R1 e R2 sostituisce il contenuto di R1. Qualunque configurazione di bit è valida. L'operazione segue bit per bit lo schema della operazione logica di OR (vedi istruzione O).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

OR 5,4

dove:

1° operando reg. 5 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 4 " " " " " ØØ8Ø68

risultato reg. 5 " " " " " Ø188FA

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando R2 =

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØØ	11111Ø1Ø

Descrizione Funzionale

Dealloca la lista degli argomenti e ne riceve il valore contenuto sulla cima della lista.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**RETS**	**I**	**RR**	**20**

Azione

Si dealloca dalla cima dello stack la lista degli argomenti che si era generata a seguito di una CALL, e ripassa il controllo all'istruzione successiva alla CALL stessa.

C.C.: non gestito

Descrizione Funzionale

Sottrae il contenuto di una parola al contenuto di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	S	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	5B

Azione

Il contenuto della parola allineata indirizzata dal secondo operando è sottratto al contenuto del registro R1, ed il risultato è posto in R1. L'operazione è svolta sommando il complemento a 2 della parola al registro. Tutti i bit dei due operandi (segni compresi) partecipano all'operazione. L'overflow si verifica quando i riporti del segno e del bit di ordine più alto sono discordanti.

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " < Ø
2 " " " " " " > Ø
3 se si verifica un'overflow.

Esempio

S 12,64 (4,6)

dove:

1° operando reg. 12 il cui valore esadecimale è Ø1Ø8BA

2° operando 64(4,6) " " " " " ØØ8Ø68

risultato reg. 12 " " " " " ØØ8852

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = parola

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

risultato in R1 =

1ØØØØØØØ	ØØØØØØØØ	1ØØØ1ØØØ Ø1Ø1ØØ1Ø

Descrizione Funzionale

Ricerca sequenziale di un elemento sul sottocampo di una tabella.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**SES**	**D_1 (L_1, B_1), D_2 (L_2, B_2)** **S_1 (L_1), S_2 (L_2)** **S_1, S_2** **D_1 (, B_1), D_2 (, B_2)**	**SS**	**F7**

Azione

Viene ricercata l'uguaglianza di un elemento indirizzato dal primo operando e il campo di memoria indirizzato dal secondo operando. La ricerca è effettuata in modo sequenziale a partire dall'elemento indirizzato e per indirizzi crescenti di memoria. L1 è la lunghezza dell'elemento di cui si cerca l'occorrenza. L2 è la lunghezza del secondo operando e quindi anche del sottocampo da confrontare. Il registro generale Ø contiene il numero massimo di occorrenze da ricercare (solo i 2 byte meno pesanti sono ritenuti significativi). Il registro generale 2 contiene lo spiazzamento del sottocampo nell'elemento. Si considerano solo i 4 bit meno pesanti del registro generale 2.

Se l'esito della ricerca è positivo, viene caricato nel registro generale 1 l'indirizzo del campo secondo operando uguale al primo campo. Nel registro generale 2 il numero ordinale (Ø ÷ N) dell'elemento nella tabella.

Se l'esito è negativo, si carica nel registro generale 1 l'indirizzo del byte successivo all'ultimo elemento, e nel registro generale 2 il numero totale di elementi (N + 1). Se il registro generale Ø = Ø si pone il condition code = 3 e si carica nel registro

generale 1 l'indirizzo della tabella, e nel registro generale 2 il valore Ø.

C.C.: Ø se la ricerca ha esito positivo
1 " " " " " "
2 " " " " " "
3 se la ricerca ha esito negativo

Descrizione Funzionale

Ricerca sequenziale di un elemento sul sottocampo mascherato di una tabella.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SESM	D_1 (L_1, B_1), D_2 (L_2, B_2) S_1 (L_1), S_2 (L_2) S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	F0

Azione

. Viene fatto il confronto fra il risultato dell'AND di un sottocampo della tabella indirizzata dal primo operando con gli L2 byte meno significativi del registro R1, e il campo di memoria indirizzato dal secondo operando. La ricerca è effettuata in modo sequenziale a partire dall'elemento indirizzato e per indirizzi crescenti di memoria. L1 è la lunghezza di un elemento di tabella. L2 è la lunghezza del secondo operando e quindi anche del sottocampo da confrontare. Se L2 < 4 si assume L2 = 4. Il registro generale Ø contiene il numero totale degli elementi della tabella da considerare (solo 2 byte meno pesanti sono ritenuti significativi). Il registro generale 2 contiene lo spiazzamento del sottocampo nell'elemento. Si considerano solo i 4 bit meno pesanti del registro generale 2.

Se l'esito della ricerca è positivo, si carica nel registro generale 1 l'indirizzo dell'elemento il cui sottocampo ha causato il confronto, e nel registro generale 2 il numero ordinale (Ø ÷ N) dell'elemento nella tabella.

Se l'esito è negativo, si carica nel registro generale

1 l'indirizzo del byte successivo all'ultimo elemento, e nel registro generale 2 il numero totale di elementi (N + 1). Se il registro generale Ø = Ø si pone il condition code a 3 e si carica nel registro generale 1 l'indirizzo della tabella e nel registro generale 2 il valore Ø.

C.C.: Ø se la ricerca ha esito positivo
1 " " " " " "
2 " " " " " "
3 se la ricerca ha esito negativo

Descrizione Funzionale

Sottrae il contenuto di una mezza parola dal contenuto di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SH	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	4B

Azione

Il contenuto della mezza parola allineata e indirizzata dal secondo operando è sottratto al contenuto del registro R1 ed il risultato è posto in R1. Prima dell'operazione il segno della mezza voce viene espanso per una lunghezza di 16 bit; l'operazione è eseguita sommando il complemento a 2 del valore così ottenuto (32 bit) al registro. Un'overflow si verifica quando i riporti del segno e del bit di ordine più alto sono discordanti.

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " < Ø
2 " " " " " " > Ø
3 se si verifica un'overflow

Esempio

SH 9,52 (,2)

dove:

1° operando reg. 9 il cui valore esadecimale è Ø1Ø8BA

2° operando 52(,2) " " " " " FF8Ø68

risultato reg. 9 " " " " " Ø18852

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = mezza parola

11111111	11111111	1ØØØØØØØ Ø11Ø1ØØØ

ESPANSIONE

risultato il R1 =

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØØ Ø1Ø1ØØ11

SUBTRACT LOGICAL

SL

Descrizione Funzionale

Sottrae in modo logico il contenuto di una parola dal contenuto di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SL	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	5F

Azione

Il contenuto della parola allineata indirizzata dal secondo operando è sottratta al contenuto del registro R1 ed il risultato è posto in R1. L'operazione è eseguita sommando il complemento a due del contenuto della voce al registro. Se si verifica un riporto della posizione del segno il condition code viene posto a determinati valori, come specificato in seguito. Si noti che il risultato Ø non può essere ottenuto senza che si verifichi un riporto; pertanto il condition code non sarà mai posto a Ø.

C.C.: Ø se il risultato è ≠ Ø e non c'è riporto
2 se il risultato è = Ø e c'è riporto
3 " " " " ≠ Ø " " "

Esempio

SL 5,17 (,8)

dove:

1° operando reg. 5 il cui valore esadecimale è Ø1Ø8BA

2° operando 17(,8) " " " " '" ØØ8Ø68

risultato reg. 5 " " " " " ØØ8852

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = parola

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

risultato in R1 =

1ØØØØØØØ	ØØØØØØØØ	1ØØØ1ØØØ Ø1Ø1ØØ1Ø

Descrizione Funzionale

Shift a sinistra del contenuto di un registro, segno escluso.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SLA	R_1, I_2	RR	02

Azione

Il contenuto di R1 è shiftato a sinistra di tante posizioni, quante specificate dall'immediato I2. Il segno resta immutato. I bit che escono a sinistra dai limiti del registro sono persi, mentre altrettanti Ø entrano da destra. Uno spostamento del contenuto di un registro a sinistra, equivale a moltiplicarlo per 2. Si verifica overflow quando viene perduto per lo shift un bit diverso dal bit del segno.

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " < Ø
2 " " " " " " > Ø
3 se si verifica un'overflow

SLL

Descrizione Funzionale — Shift a sinistra di un registro, segno compreso.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SLL	R_1, I_2	RR	04

Azione

Il contenuto di R1 è shiftato a sinistra di tante posizioni quante specificate dall'immediato I2. Tutti i bit di R1, segno compreso, partecipano all'operazione. I bit che escono a sinistra sono persi, e altrettanti Ø entrano da destra.

C.C.: non gestito

SUBTRACT LOGICAL MEMORY

SLM

Descrizione Funzionale

Sottrae il contenuto di un campo di memoria ad un altro campo di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SLM	D_1 (L_1, B_1), D_2 (L_2, B_2) S_1 (L_1), S_2 (L_2) S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	F4

Azione

Il contenuto del campo di memoria indirizzato dal secondo operando è sottratto al contenuto del campo di memoria indirizzato dal primo operando. La sottrazione viene ottenuta sommando il complemento a 2 del secondo operando al primo operando. L'operazione procede da destra a sinistra per il numero di byte indicato in L1. Se il secondo operando è più corto viene esteso "logicamente" con bit Ø.

C.C.: 1 se il risultato è ≠ Ø senza riporto
2 " " " " = Ø con riporto
3 " " " " ≠ Ø con riporto

Esempio

SLM 50 (,6), 26 (,7)

dove:

1° operando 5Ø(,6) il cui valore esadecimale è Ø1Ø8BA

2° operando 26(,7) " " " " " ØØ8Ø68

risultato 5Ø(,6) " " " " " ØØ8852

1° operando = campo di memoria

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = campo di memoria

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

risultato in 1° operando

1ØØØØØØØ	ØØØØØØØØ	1ØØØ1ØØØ Ø1Ø1ØØ1Ø

SUBTRACT LOGICAL REGISTER

SRL

SLR

Descrizione Funzionale — Sottrae in modo logico il contenuto di un registro al contenuto di un altro registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**SLR**	**R_1, R_2**	**RR**	**1F**

Azione

Il contenuto di R2 è sottratto al contenuto di R1 e il risultato è posto in R1. L'operazione è eseguita sommando il complemento a 2 del contenuto di R2 al contenuto di R1. Tutti i bit dei due operandi partecipano all'operazione. Se si verifica un riporto dalla posizione del segno, il condition code viene posizionato come descritto in seguito. Si noti che il risultato Ø non può essere ottenuto senza che si verifichi un riporto, pertanto il condition code non sarà mai impostato a Ø.

C.C.: 1 se il risultato è ≠ Ø e non c'è riporto
2 " " " " = Ø e c'è riporto
3 " " " " ≠ Ø e c'è riporto

Esempio

SLR 4,11

dove:

1° operando reg. 4 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 11 " " " " " 888Ø68

risultato reg. 4 " " " " " ØØ8852

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando R2 =

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

risultato in R1 =

1ØØØØØØØ	ØØØØØØØØ	1ØØØ1ØØØ Ø1Ø1ØØ1Ø

Descrizione Funzionale — Sottrae un immediato al contenuto di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**SLRI**	R_1, I_2	**RR**	**0C**

Azione

Sottrae al contenuto del registro R1 l'immediato I2. L'eventuale riporto viene segnalato dal condition code. La sottrazione avviene con la somma al registro R1 del complemento a 2 di I2.

C.C.: Ø se la differenza senza riporto è = Ø
1 " " " " " " ≠ Ø
2 se la differenza con riporto è = Ø
3 " " " " " " ≠ Ø

Descrizione Funzionale

Sottrae algebricamente il contenuto di un campo di memoria dal contenuto di un altro campo di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SM	$D_1 (L_1, B_1), D_2 (L_2, B_2)$ $S_1 (L_1), S_2 (L_2)$ S_1, S_2 $D_1 (, B_1), D_2 (, B_2)$	SS	F3

Azione

Il campo di memoria indirizzato dal secondo operando è sottratto al campo di memoria indirizzato dal primo operando; il risultato è posto nel primo operando. Nell'operazione il bit più pesante del byte più pesante di ciascun campo, è considerato come bit di segno. L'operazione viene eseguita espandendo logicamente con il bit del segno l'operando più corto, e sommando al primo operando il complemento a 2 del secondo operando. Si verifica overflow:

- se i riporti del segno e del bit di ordine più alto sono discordanti

- se L1 < L2 e gli L2 - L1 byte più pesanti del risultato non sono costituiti da bit uguali al bit più pesante del byte L1 del risultato.

C.C.: Ø se il risultato è = Ø
1 " " " " < Ø
2 " " " " > Ø
3 se si verifica overflow

Esempio

SM 48 (2,5), 64 (1,7)

dove:

1° operando 48(2,5) il cui valore esadecimale è Ø8BA

2° operando 64(1,7) " " " " " ØØ68

risultato 48(2,5) " " " " " Ø852

1° operando = campo di memoria

ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = campo di memoria

ØØØØØØØØ	Ø11Ø1ØØØ

ESPANSIONE

risultato in 1° operando

ØØØØ1ØØØ	Ø1Ø1ØØ1Ø

Descrizione Funzionale

Sottrae il contenuto di un registro al contenuto di un altro registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**SR**	**R_1, R_2**	**RR**	**1B**

Azione

Il contenuto di R2 è sottratto al contenuto di R1, e il risultato è messo in R1. L'operazione è eseguita sommando il complemento a 2 di R2 a R1. Ponendo R1 = R2 si ha come risultato l'azzeramento del registro. Un'overflow si verifica quando i riporti del segno e del bit di ordine più alto sono discordanti.

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " $<$Ø
2 " " " " " " $>$Ø
3 se si verifica un'overflow

Esempio

SR 8,3

dove:

1° operando reg. 8 il cui valore esadecimale è FFFFFF

2° operando reg. 3 " " " " " FFFFFF

risultato reg. 8 " " " " " ØØØØØØ

1° operando R1 =

11111111	11111111	11111111	11111111

2° operando R2 =

11111111	11111111	11111111	11111111

risultato in R1 =

ØØØØØØØØ	ØØØØØØØØ	ØØØØØØØØ	ØØØØØØØØ

SHIFT RIGHT ALGEBRAIC

SRA

Descrizione Funzionale — Shift a destra di un registro, segno escluso.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**SRA**	**R_1, I_2**	**RR**	**03**

Azione

Il contenuto di R1 è shiftato a destra di tante posizioni quante sono specificate dall'immediato I2. Il segno resta immutato. I bit che escono a destra sono persi, mentre quelli che entrano da sinistra sono uguali al segno.

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " Ø
2 " " " " " " Ø

Descrizione Funzionale — Shift a destra di un registro, segno compreso.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
(simbolo)	SRL	R_1, I_2	RR	08

Azione

Il contenuto di R1 è shiftato a destra di tante posizioni quante specificate dall'immediato I2. Tutti i bit segno compreso partecipano all'operazione. I bit che escono da destra sono persi; da sinistra entrano degli Ø.

C.C.: non gestito

Descrizione Funzionale — Memorizza il contenuto di un registro in una parola.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	ST	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	50

Azione

Il contenuto del registro R1 è memorizzato nel campo di memoria specificato come secondo operando. Il campo di memoria è costituito da una parola che deve seguire le norme per l'allineamento descritte nel capitolo 5 di questo manuale.

C.C.: non gestito

STORE CHARACTER

STC

Descrizione Funzionale

Memorizza in un byte di memoria gli ultimi 8 bit di destra di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**STC**	**R_1, D_2 (X_2, B_2)** **R_1, D_2 (, B_2)** **R_1, S_2 (X_2)** **R_1, S_2**	**RX**	**42**

Azione

I bit 24 ÷ 31 del registro R1 sono memorizzati all'indirizzo espresso dal secondo operando.

C.C.: non gestito

STORE HALFWORD

STH

Descrizione Funzionale

Memorizza il contenuto dei due byte di destra di un registro in una mezza parola.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	STH	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	40

Azione

Il contenuto della metà di destra del registro R1 è memorizzato alla mezza parola allineata indirizzata dal secondo operando. La metà sinistra del registro non partecipa all'operazione.

C.C.: non gestito

Descrizione Funzionale

Memorizza il contenuto di N registri adiacenti in N parole consecutive.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	STM	$R_1, R_2, D_3 (B_3)$ R_1, R_2, S_3	RS	90

Azione

Il contenuto dei registri compresi tra R1...R2 è memorizzato in altrettante parole consecutive, a partire dall'indirizzo espresso dal secondo operando. Se R1 = R2 il solo registro R1 è memorizzato. Se R1 > R2 il trasferimento ha comunque luogo, tenuto conto che il registro RØ è considerato consecutivo al registro R15. Le N parole consecutive devono seguire le norme di allineamente descritte nel capitolo 5 di questo manuale.

C.C.: non gestito

Descrizione Funzionale

Test di un byte di memoria usando una maschera inserita nell'istruzione.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	TM	D_1 (B_1), I_2 S_1, I_2	SI	91

Azione

Il secondo byte dell'istruzione (contenente il dato immediato) funge da maschera. Per ogni bit = 1 ad esso appartenente, viene controllato il bit corrispondente del byte di memoria indirizzato dal primo operando. Il test avviene bit per bit. Qualunque configurazione di bit è valida.

C.C.: Ø se tutti i bit controllati sono = Ø oppure
se tutti i bit della maschera sono = Ø
1 se tutti i bit controllati sono misti
3 se tutti i bit controllati sono = 1

Descrizione Funzionale Ricerca tabellare.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**TR**	**D_1 (L, B_1), D_2 (B_2)** **S_1 (L), S_2** **S_1, S_2** **D_1 (, B_1), D_2 (, B_2)**	**SS**	**DC**

Azione

I byte contenuti nel campo indirizzato dal 1° operando sono considerati come una lista di argomenti; quelli del campo indirizzato dal 2°operando come una lista di funzioni. Vengono presi in considerazione uno alla volta i byte la lista degli argomenti; il contenuto di ognuno di essi preso come valore intero non segnato viene sommato in modo aritmetico all'indirizzo del secondo operando. Il nuovo indirizzo così ottenuto corrisponde ad un byte della lista delle funzioni. Il contenuto del byte funzione sostituisce quello del byte argomento. L'operazione continua fino all'esaurimento del campo argomenti.

C.C.: non gestito

TRT

Descrizione Funzionale — Ricerca tabellare con test della funzione.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	TRT	D_1 (L, B_1), D_2 (B_2) S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	D0

Azione

I byte contenuti nel campo indirizzato dal primo operando sono considerati come una lista di argomenti; quelli contenuti nel campo indirizzato dal secondo operando sono considerati come una lista di funzioni. Partendo da sinistra e per un numero di byte pari ad L (1 ÷ 256) vengono presi in considerazione uno alla volta i byte formanti la lista degli argomenti. Il contenuto di ognuno di essi viene sommato all'indirizzo del secondo operando; il nuovo indirizzo così ottenuto corrisponde ad un byte della lista delle funzioni. Se l'AND di quest'ultimo byte con il byte meno pesante del registro generale Ø è uguale a Ø, viene preso in esame il successivo argomento; se invece è diverso da Ø, lo stesso byte viene inserito nelle 8 posizioni binarie di destra del registro generale 2 (il resto del registro resta immutato), lo indirizzo dell'argomento corrispondente, viene inserito nel registro generale 1 (il resto del registro resta immutato) e l'operatore si arresta. I byte formanti la lista degli argomenti restano immutati.

C.C.: Ø se tutti i byte controllati dalla lista delle funzioni sono = Ø
1 se almeno un byte controllato dalla lista delle funzioni è ≠ Ø
2 se l'ultimo byte controllato selezionato nella lista delle funzioni è ≠ Ø

Descrizione Funzionale

OR esclusivo fra il contenuto di un registro e di una parola.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**X**	**$R_1, D_2 (X_2, B_2)$** **$R_1, D_2 (, B_2)$** **$R_1, S_2 (X_2)$** **R_1, S_2**	**RX**	**57**

Azione

L'OR esclusivo fra il contenuto del registro R1 e della parola allineata contenuta nel secondo operando, sostituisce il contenuto di R1. Qualunque configurazione di bit è valida. L'operazione segue bit per bit lo schema dell'OR esclusivo sotto riportata:

XOR	Ø	1
Ø	Ø	1
1	1	Ø

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " $\neq$ Ø

Esempio

X 12,37 (4,6)

dove:

1° operando reg. 12 il cui valore esadecimale è Ø1Ø8BA

2° operando 37(4,6) " " " " " ØØ8Ø68

risultato reg. 12 " " " " " Ø188D2

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = parola

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØØ	11Ø1ØØ1Ø

EXCLUSIVE OR CHARACTER

XC

Descrizione Funzionale — OR esclusivo fra il contenuto di due campi di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	XC	D_1 (L,B_1), D_2 (B_2) S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	D7

Azione

OR esclusivo del contenuto del campo di memoria del primo operando, di lunghezza L byte, e del campo di memoria del secondo operando, pure lungo L byte, sostituisce il contenuto del primo operando. La lunghezza L è quella associata implicitamente o esplicitamente al campo contenuto nel primo operando. Qualunque configurazione di bit è valida. L'operazione segue bit per bit lo schema della operazione di OR esclusivo (vedi istruzione X).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

XC 97 (,5), 84 (,9)

dove:

1° operando 97(,5) il cui valore esadecimale è Ø1Ø8BA

2° operando 84(,9) " " " " " ØØ8Ø68

risultato 97(,5) " " " " " Ø188D2

1° operando = campo di memoria

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = campo di memoria

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in 1° operando =

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØØ	11Ø1ØØ1Ø

EXCLUSIVE OR IMMEDIATE

XI

Descrizione Funzionale

OR esclusivo di un byte contenuto nell'istruzione e di un byte di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	XI	D_1 (B_1), I_2 S_1, I_2	SI	97

Azione

OR esclusivo del secondo byte dell'istruzione contenente I2 e del byte di memoria contenuto del primo operando, vanno a sostituire il contenuto del primo operando. Qualunque configurazione di bit è valida. L'operazione segue bit per bit lo schema della operazione di OR esclusivo (vedi istruzicne X).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

XI 43 (5), X'6E'

dove:

1° operando 43(5) il cui valore esadecimale è BB

2° operando x'6E' " " " " " 6E

risultato 43(5) " " " " " D5

1° operando = byte di memoria

1Ø111Ø11

2° operando I2 =

Ø11Ø111Ø

risultato in 1° operando

11Ø1Ø1Ø1

EXCLUSIVE OR REGISTER

Descrizione Funzionale — OR esclusivo del contenuto di due registri.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	XR	R_1, R_2	RR	17

Azione

OR esclusivo del contenuto dei registri R1 e R2 sostituisce il contenuto di R1. Qualunque configurazione di bit è valida. L'operazione segue bit per bit lo schema della operazione di OR esclusivo (vedi istruzione X).

C.C.: $\emptyset$ se il risultato dell'operazione è = $\emptyset$
1 " " " " " " $\neq \emptyset$

Esempio

XR 12,2

dove:

1° operando reg. 12 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 2 " " " " " ØØ8Ø68

risultato reg. 12 " " " " " Ø188D2

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando R2 =

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØØ	11Ø1ØØ1;

5. DESCRIZIONE DELLE PRESTAZIONI ASSEMBLER

Sono di seguito descritte le modalità per il caricamento e l'esecuzione di un programma Assembler.

Il programma sorgente viene introdotto da tastiera, sotto forma di file testo, e salvato su floppy disk.

Il file testo viene dato in input all'assemblatore che produce un oggetto rilocabile che viene salvato su floppy disk.

Al fine di eseguire il programma si deve ancora trasformare l'oggetto rilocabile in formato eseguibile: questa funzione viene svolta dal linker.

Introduzione programmi

L'introduzione dei programmi, in linguaggio sorgente, avviene da tastiera digitando le istruzioni quali linee di un file testo.

I programmi, in formato sorgente, devono essere memorizzati nelle librerie del sistema (file di tipo T), al fine di essere successivamente richiamati e assemblati.

La dimensione di un programma sorgente può essere superiore alla massima estensione di un file testo (il file testo ha come limite la dimensione della memoria utente disponibile); in tal caso si può memorizzare il programma sorgente su più file testo. Si veda come l'utility ASSEMBLY (appendice A) accetta come input un programma su più file testo.

L'introduzione delle linee di file testo, può essere preceduta dal comando AUTO# che assegna automaticamente il numero di linea.

Le linee introdotte sono modificabili come descritto al punto "correzione dell'introduzione da tastiera" del capitolo 2.

Assemblaggio programmi

L'assemblaggio dei programmi viene realizzato mediante l'esecuzione dell'utility ASSEMBLY descritto nell'appendice A.

Il passo di assemblaggio trasforma i programmi da formato sorgente (input) in formato oggetto rilocabile (output).

Input l'input dell'assemblatore è costituito da uno o più file testo (file di tipo T); l'assemblatore tratta tutte le frasi presenti, dall'inizio del primo file all'incontro di una frase END, o in assenza di questa, alla fine dell'ultimo file.

La frase sorgente è costituita da una o più linee di testo caratterizzate da un numero di linea e chiuse con l'introduzione del tasto (END OF LINE) . L'assemblatore non esegue alcun controllo sulla sequenza dei numeri di linea.

Output l'output dell'assemblatore è così costituito:

Codice oggetto: contiene le linee oggetto in formato rilocabile (il file è di tipo O) ed è strutturato nelle seguenti sezioni:

Testata: Contiene le seguenti informazioni:

1. Nome modulo corrisponde al nome della macro PROC, se presente; se assente o se non ha nome, corrisponde al nome della frase CSECT/START.

2. Spiazzamento delle parole d'indirettezza, rispetto alla fine della testata (lunghezza due byte).

3. Spiazzamento del dizionario, rispetto alla fine della testata (lunghezza due byte).

4. Numero di riferimenti esterni (lunghezza un byte); se nel modulo non è stato definito alcun nome esterno questo campo è a Ø e i due campi precedenti non sono significativi.

5. Un byte per garantire l'allineamento alla mezza parola.

La testata occupa i primi 14 caratteri del modulo oggetto.

Oggetto istruzioni: Contiene le stringhe oggetto prodotte dall'assemblatore a fronte di istruzione macchina e di definizioni dati. Sono presenti zone generate da frasi DS e zone generate dall'assemblatore, per garantire l'allineamento alla mezza parola, che sono inizializzate a zero binario.

Oggetto literal: contiene le stringhe oggetto generate a fronte di literal nel programma sorgente.

Parole d'indirettezza: Contengono i riferimenti ai moduli esterni e di sistema chiamati dal programma sorgente. I riferimenti, a livello di assemblaggio, vengono valorizzati a zero o a uno, a seconda se il modulo riferito è un modulo esterno o di sistema. Le parole di indirettezza, a livello di linker, vengono valutate in base al loro valore:

1. Se questo è zero si va a cercare nella lista dei moduli esterni disponibili e se il modulo esterno esiste il linker valorizza la parola d'indirettezza con l'indirizzo fisico del modulo; se non esiste si segnala errore.

2. Se questo è uno si va a cercare nella lista dei moduli di sistema e si ha un comportamento analogo al punto 1.

Dizionario nomi esterni: contiene i nomi dei moduli esterni e di sistema richiamati dal programma sorgente. I nomi sono su 8 crt. Se il programma sorgente fa riferimento più volte allo stesso nome esterno, nel dizionario viene memorizzato una sola volta. Le parole d'indirettezza e gli elementi del dizionario sono in corrispondenza biunivoca.

Messaggi: Sono relativi a situazioni di errore riscontrate durante la fase di assemblaggio che impediscono la continuazione dell'assemblaggio stesso. Il sistema segnala il messaggio relativo all'anomalia (vedi appendice D) e o torna nello stato comandi dopo aver deallocato il flusso scratch e l'eventuale flusso oggetto o prosegue l'elaborazione dell'utility ASSEMBLY.

Listing: Ad assemblaggio ultimato viene fornito il listing del sorgente assemblato. Il listing è così costituito:

1. Una sezione contenente le opzioni di assemblaggio ed informazioni sull'output.

2. Una sezione diagnostica contenente, se si rilevano errori, informazioni relative al numero di statement che ha provocato l'errore e una segnalazione estesa dell'errore.(vedi, per l'elenco delle segnalazioni diagnostiche, l'appendice D).

3. Una sezione contenente l'elenco dei flussi testo che compongono il sorgente con l'indicazione del numero di statement minimo e massimo contenuto in ciascun flusso testo.

4. Una sezione, prodotta su opzione (XREF), relativa alla Cross Reference; contiene l'elenco di tutti i nomi definiti nel programma sorgente, in ordine alfabetico e con il valore, la lunghezza e il numero della frase sorgente in cui il nome è stato definito e i numeri degli statement che vi si riferiscono. Per i nomi esterni e di sistema viene semplicemente stampato l'elenco degli statement in cui tali nomi sono stati riferiti.

5. Una sezione, prodotta su opzione (LIST), relativa al Listing delle frasi sorgenti con lo spiazzamento associato e l'eventuale oggetto prodotto; in coda viene stampata la lista dei literal presenti nel programma sorgente. Se l'opzione LIST non è presente, tra le opzioni di assemblaggio, vengono comunque listati gli statement con errori diagnostici. Lo spiazzamento non viene stampato per le frasi commento e per le frasi USING, DROP, EXT, EQU e END. Durante questa fase l'assemblatore controlla ed esegue le funzioni richieste tramite le frasi direttive del listing: TITLE, SPACE, EJECT e PRINT.

6. Una sezione composta da informazioni riassuntive:

   - il nome del modulo oggetto prodotto

   - occupazione del modulo oggetto rilocabile, comprensiva del dizionario e della testata

   - occupazione del modulo oggetto prodotto

   - il numero degli statement errati

- codice d'errore di peso maggiore.

Esempio

```
                                                                    PAGE 1

                          ASSEMBLY OPTIONS                  DATE   24/05/78

   LIST =YES
   XREF =YES
   OUT  = PROV10 (USLIB )









                                                                    PAGE 2

                        ASSEMBLER DIAGNOSTICS

FILENAME LINE STMT COL. CODE MESSAGE                          DATE : 24/05/78

 PROV1   0050    5   11 A009 INCORRECT SPECIFICATION OF REGISTER OR MASK









                                                                    PAGE 3

                          ASSEMBLY OPTIONS                  DATE : 24/05/78

   SOURCE FILE NAME          FROM STM    TO STM

        PROV1                    1          17



                                                                    PAGE 4

                          CROSS REFERENCES

SYMBOL     LEN   VALUE   DEFN  REFERENCES                       DATE : 24/05/78

AREA1       16   000020    13    15
CBTEST       1   000000     2
FLAG         4   000030    14    15
IND          4   00001C    11     7
INIZ         4   000002     5
L            4   00001C    12    15
NCONF        4   000018    10
OUT         10   000034    15     9
```

EXTERNAL SYMBOL DICTIONARY

SYMBOL	REFERENCES
PRINT	15

DATE : 24/05/78


```
LINE  LOC       OBJECT CODE      STMT  SOURCE STATEMENT        DATE : 24/05/78

0010 000000                         1          CSECT
0020 000000                         2  CBTEST  PROC    PROLOG=NO
0030 000000 05A0                    3          BALR    10,0
0040                                4          USING   *,10
0050 000002 0000 0000               5  INIZ    LA      14,4095(20)
    *** ERROR ***
0060 000006 41EE 0001               6          LA      14,1(14)
0070 00000A 4130 A01A               7          LA      3,IND
0080 00000E 4883 0000               8          LH      8,0(3)
0090 000012 47F0 A032               9          B       OUT
0100 000018 00000000               10  NCONF   DC      F'0'
0110 00001C                        11  IND     DS      0F
0120 00001C 00000010               12  L       DC      F'16'
0130 000020 4142434445464748       13  AREA1   DC      CL16'ABCDEFGHILMNOP
                                                                        R
0140 000030                        14  FLAG    DS      F
0150 000034 9C03 A03E A01E A01A    15  OUT       CALEXS PRINT,(AREA1,L,FLAG)
             A02E                                    '
0160 00003E 07FE                   16          BR    14
0170                               17          END
```

ASSEMBLER STATISTICS

DATE : 24/05/78

```
PROGRAM NAME = CBTEST
NUMBER OF STATEMENT FLAGGED = 1
HIGHEST SEVERITY WAS  8
LINK OBJECT LENGTH = 0000005A
OBJECT MODUL LENGTH = 0044
```

Caratteristiche dell'assemblatore

Nel seguito vengono date alcune informazioni inerenti alle funzioni assolte dall'assemblatore ed alle sue prestazioni. L'assemblatore assolve alle seguenti funzioni:

1. Traduce in istruzioni macchina le istruzioni simboliche.

2. Esegue le funzioni ausiliarie richieste dal programmatore tramite frasi direttive all'assemblatore.

3. Calcola, per ogni indirizzo espresso mediante una espressione, il corrispondente indirizzo di memoria in termini di registro base e di spiazzamento.

4. Gestisce il suo output (oggetto rilocabile) in modo che può essere caricato in una zona qualunque della memoria.

5. Calcola l'occupazione di memoria del modulo oggetto per mezzo di un contatore di posizioni (Location Counter).

6. Gestisce, all'interno del programma sorgente, delle dummy section sino a un massimo di 127. Le dummy section permettono all'utente di riconfigurare zone di memoria o riferirsi a zone, la cui allocazione non è nota a priori.

7. Gestisce una tavola di literal: sono usati per definire dati costanti.

8. Realizza le richieste di allineamento dell'utente e realizza le condizioni di allineamento per le istruzioni macchina.

9. Gestisce una tavola dei nomi (Symbol Table) nella memoria utente. La tavola dei nomi contiene tutte le etichette di frasi direttive, esecutive o di dichiarazione dati, con associato il loro valore.

10. Genera, a fronte di moduli esterni o di sistema (ad esempio READ) che compaiono nel programma sorgente come operandi di istruzioni (ad esempio CALEXS) che gestiscono il collegamento trai moduli, delle tavole di informazioni per il Linker:

- dizionario dei nomi esterni: è costituito dalla lista dei nomi esterni definiti nella Control Section nell'ordine in cui compaiono; nomi esterni uguali generano un unico elemento di dizionario

- tavola delle parole di indirettezza: contiene, in corrispondenza di ogni nome di dizionario, un elemento (parola d'indirettezza) che contiene l'indirizzo del modulo.

11. Genera codice oggetto a fronte delle macro (PROC, PRRET, LAEXT), in quanto queste non subiscono una fase di macroespansione, non essendo ancora definito un linguaggio per le macro.

L'assemblatore ha le seguenti prestazioni:

1. Tempo di compilazione medio: da 0,5 a 1 secondo per una istruzione senza stampa.

2. Il numero massimo di simboli, nomi esterni e literal che possono comparire in un programma sorgente, sono in dipendenza della capacità dell'area utente del sistema; se nel sistema viene caricata l'opzione DEB l'area utente del sistema viene decrementata dall'area occupata dall'opzione e arrotondata per difetto a moduli di 4 K. Nel seguito si dà un prospetto di tali limiti in dipendenza dell'area utente disponibile; in corrispondenza dell'area utente che coincide con la configurazione base della macchina viene posto un *:

	<16K	16K *	20K	24K *	28K	32K *	36K	40K *	44K	48K *
N. simboli	100	302	515	657	875	1013	1215	1368	1590	1724
N. nomi esterni	20	25	25	25	30	51	60	51	51	76
N. byte per tavola literal	256	256	256	512	512	1024	1024	1536	1536	1792

3. Gestisce il flusso di output, allocandosi l'area.

4. Utilizza in modo temporaneo un file scratch costituito dall'area non utilizzata del floppy.

In un sistema monodisco i due flussi (output e scratch) sono concorrenti; l'assemblatore suddivide l'area disponibile di disco dinamicamente, tra i due flussi in base alla loro reale occupazione.

In un sistema bidisco l'assemblatore alloca il flusso di scratch sul floppy utente, a meno che l'utente non specifichi una particolare opzione tra i parametri di lancio (WSP = $\left\{\begin{matrix}\text{USLIB}\\\text{SYSLIB}\end{matrix}\right\}$).
Se entrambi i flussi sono sullo stesso floppy la situazione è analoga al sistema monodisco. In caso contrario non esiste antagonismo e l'allocazione è fatta in base all'allocazione dei due flussi.

A fine assemblaggio rimane allocato il flusso di output contenente il modulo oggetto generato; se il programma sorgente non ha generato alcun oggetto, oltre la testata, il file viene deallocato.

Link degli oggetti

Il link degli oggetti esegue la trasformazione del programma da oggetto rilocabile (file di tipo O) a oggetto eseguibile (file di tipo E).

Il link degli oggetti viene realizzato per mezzo dell'esecuzione dell'utility LNK (si veda la descrizione nell'appendice A)

Il passo di Linker risolve i riferimenti degli eventuali moduli di sistema presenti nel programma, in modo che i richiami siano riconosciuti a run time e siano risolti in modo automatico.

Esecuzione programmi

L'esecuzione del programma avviene dopo il suo trasferimento, in formato oggetto eseguibile (file di tipo E), da libreria in memoria.

L'esecuzione di un programma viene realizzata mediante l'esecuzione del comando EXQ (si veda la sua descrizione al capitolo 3).

Gestione degli errori

Un programma utente in esecuzione può generare delle condizioni di errore, che si manifestano all'utente sotto forma di interruzioni interne del sistema.

Le interruzioni interne arrestano il programma in esecuzione ed il sistema si blocca segnalando su display il tipo di errore; l'utente può, premendo il pulsante BREAK, ritornare nello stato comandi. Se nella memoria sono caricate le routine di debugging, il sistema provvede a segnalare su display altre informazioni sull'errore e l'utente è abilitato ad operare la tecnica del debugging (si veda la descrizione al capitolo 7).

Le condizioni di errore che si possono verificare, ed i rispettivi codici di ritorno, sono descritte nell'appendice D.

File di lavoro su floppy disk

Il file di lavoro su floppy disk permette al programma utente in esecuzione, di utilizzare un'area di lavoro su FDU, alla quale può accedere automaticamente mediante il richiamo dei moduli di sistema READ e WRITE (vedi capitolo 6).

E' inteso come un'estensione della memoria utente.

Non è accessibile in ambiente diverso da quello di esecuzione.

Deve essere appositamente creato dall'utente per mezzo del comando CREATE: il file è di tipo W.

Può risiedere indifferentemente su floppy disk di sistema o utente.

Il programma utente in esecuzione ha a disposizione il file di lavoro su floppy disk, se detto file è presente nelle librerie del sistema e se nel comando EXQ è presente l'operando "W= filename1" che lo riferisce (vedi comando EXQ capitolo 3).

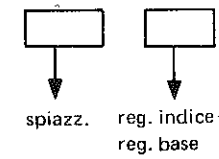
Indirizzamento

La tecnica di indirizzamento su P6Ø6Ø richiede l'uso di registri base. I registri disponibili sul sistema sono 16, riferiti dai numeri compresi da Ø a 15.

L'indirizzamento di un programma viene realizzato mediante un registro base, che contiene l'indirizzo di inizio del programma, e uno spiazzamento; l'indirizzo di inizio del programma, nelle istruzioni di tipo RX, è costituito dalla somma del registo indice più il registro base. Il programmatore può esprimere gli indirizzi in forma esplicita o in forma implicita.

Indirizzamento esplicito: è espresso specificando lo spiazzamento e il registro base (eventualmente sommato al registro indice)
Esempio:

```
Indirizzo
Esadecimale
     Ø        1Ø BEGIN      STAPT    Ø
     ...                    ...
     ...                    ...
     ...                    ...
     ...      80            LA       10, X'40' (0,12)
     ...
     ...
     ...                    ...
     ...                    ...
     ...                    ...
     4Ø       220 ADDRESS   DC       C'PPPP'
                             ...
                             ...
                             ...
```



Come si può vedere dall'esempio, l'indirizzamento esplicito richiede, da parte del programmatore, un notevole sforzo, poichè deve calcolare per ogni riferimento ad un simbolo la sua posizione di memoria.

Indirizzamento implicito: è espresso specificando un indirizzo attraverso dei simboli (label) e senza l'indicazione del registro base. L'assemblatore risolve gli indirizzi impliciti tenendo conto di un registro base, purchè il programmatore gli indichi quali sono i registri disponibili e quale valore deve assumere per essi. Queste informazioni vengono passate all'assemblatore per mezzo della frase USING.

Questa prestazione dell'assemblatore elimina possibili errori nella stesura del sorgente (non si devono calcolare tutti gli spiazzamenti in numero di byte, ma si può indicare direttamente il simbolo interessato).

Esempio:

```
Indirizzo
Esadecimale
     Ø        1Ø BEGIN      START    Ø
    ...                     ...
    ...       40            USING    *, 1
    ...                     ...
    ...       70            LA       10, ADDRESS
    ...                     ...
    ...                     ...
    ...                     ...
    4Ø       180 ADDRESS    DC       C'PPPP'
                            ...
```

Come si può vedere nell'esempio l'indirizzamento implicito è molto più semplice; il programmatore può anche non sapere che il simbolo ADDRESS si trova al valore 40 del Location Counter.

Sezioni di un programma: Un programma è costituito da una sezione di controllo (Control Section) ed eventualmente da una o più dummy section.

1. La sezione di controllo è identificata da una delle seguenti istruzioni:

   - START non deve essere preceduta da nessuna istruzione che modifichi il Location Counter; può avere un argomento, nel qual caso deve esprimere un valore assoluto. L'argomento può anche essere una espressione di simboli; in questo caso i simboli devono essere valorizzati in precedenza mediante istruzioni EQU. Il valore dell'argomento della START viene assegnato al Location Counter, un tale valore è significativo solo per il listing.
     Esempio:

```
10      A       EQU     100
20      B       EQU     2
30              START   A + B
                ...
                ...
                ...
```

   - CSECT è l'istruzione usata in alternativa alla START; non deve essere preceduta da nessuna i-

struzione che modifichi il Location Counter.

2. Possono esistere sezioni fittizie, definite da DSECT (dummy section) che dervono per mappare aree definite all'interno della sezione controllo del programma. La fine di una sezione è determinata da un'istruzione END o dall'inizio di una nuova sezione.
Esempio:

```
10      FIRST     START     Ø
20                BALR      3, Ø
30                USING     *, 3
                  ...
                  ...
                  ...
70                LA        10, BUFFER
80                USING     DUMMY, 10
                  ...
                  ...
                  ...
150               CLI       KEY, C'X'
                  ...
200     BUFFER    DS        CL80
.                 ...
230               LA        9, BUFFER
240               USING     DUMMY1, 9
                  ...
300               CLI       SECT1, C'XOD.M'
                  ...
500     DUMMY     DSECT
510     KEY       DS        C
520     CODE      DS        CL3
530     NAME      DS        CL20
540     ADDR      DS        CL20
550     WAGES     DS        CL10
560     HRS       DS        CL8
570     DEDUCT    DS        CL6
580     PAY       DS        CL12
590     DUMMY1    DSECT
600     SECT1     DS        CL30
610     SECT2     DS        CL50
620               END
```

Nell'esempio precedente di vede che, nella control section (definita da START), l'istruzione di nome BUFFER definisce un'area di 80 crt. mentre i sottocampi di quest'area vengono definiti nella se-

zione fittizia DUMMY; all'interno della control section, dopo aver basato la sezione fittizia DUMMY con il registro 10, si può far riferimento ai campi definiti nella sezione fittizia (vedi istruzione 150). Nel caso che il programma utente abbia la necessità di avere per la stessa area (BUFFER) due formati diversi, si può definire una seconda sezione fittizia (DUMMY1) che mappa l'area BUFFER nel secondo formato desiderato. Per utilizzare i campi di questa seconda sezione fittizia bisogna caricare un nuovo registro (vedi istruzioni 230 e 240). La questo punto in poi, si può fare riferimentc all'area definita nell'istruzione di nome BUFFER in modo simbolico, specificando i nomi delle due sezioni fittizie secondo le esigenze. L'assemblatore non produce codice oggetto per le istruzioni che si trovano dentro a una dummy section e non riserva area di memoria.

Riassumendo, quando si usa una dummy section bisogna:

- definire un'area all'interno della control section (BUFFER)
- stabilire l'indirizzamento della dummy section in relazione con l'area di memoria assegnata.

Indirizzamento di una sezione controllo: L'indirizzamento viene specificato a due livelli:

1. Livello di assemblaggio - viene realizzato per mezzo della frase USING che specifica qual è il registro base e specifica qual è l'indirizzo da cui si calcolano gli spiazzamenti dei vari campi di memoria del programma. L'indirizzo base può essere espresso da un simbolo, da un valore assoluto o dal valore del Location Counter (riferito da *) in quel momento; si ricordi che il Location Counter viene riferito all'inizio del programma.

2. Livello di esecuzione - viene realizzato per mezzo dell'istruzione BALR che carica in un registrc l'indirizzo attuale dell'istruzione eseguita (valore del program counter). Il registro viene a contenere l'indirizzo della posizione fisica di memoria dell'istruzione eseguita.

Esempio 1:

```
10    INIZ     START
20             BALR      12, 0
30             USING     *, 12
40    BEGIN    L         3, A
50             A         3, B
                ...
                ...
                ...
220   A        DS        F
240   B        DS        F
                ...
                ...
                ...
```

Nell'esempio precedente abbiamo specificato la USING in posizione tale che tutti i riferimenti a simboli si possano esprimere in modo simbolico. L'istruzione BALR è specificata in posizione tale che in esecuzione carichi nel registro 12 l'indirizzo dell'istruzione che segue (l'istruzione con simbolo BEGIN, e non USING che è una dichiarativa e come tale non crea codice).

Esempio 2:

```
10    BEGIN    START     Ø
20    A        EQU       10
30             AR        3, 4
40    NORIL    SR        8, 7
50             BALR      9, 0
60             USING     *, 9
70             L         7, RIL
                ...
                ...
                ...
100            B         NORIL
                ...
200   RIL      DS        F
```

Si noti che nel caso il programmatore voglia eseguire un salto in forma implicita all'istruzione di nome NORIL (istruzione 100), tale istruzione non è eseguibile, poichè il nome NORIL si trova a monte del punto in cui (istruzione 70) l'assemblatore ha iniziato a calcolare gli spiazzamenti dei simboli.

Nel caso che la sezione sia più lunga di 4096 byte si debbono specificare, per quella sezione, più registri base, (vedi istruzione USING) poichè lo spiazzamento massimo riferito ad un registro è 4096 (nel codice di un'istruzione ci sono 12 bit disponibili per il valore binario dello spiazzamento).

Note

1. Il processor di esecuzione richiede che le istruzioni dell'oggetto eseguibile siano allineate alla mezza parola (2 byte); l'assemblatore, a fronte di istruzioni oggetto non allineate, si comporta come segue:

   - provvede a realizzare l'allineamento sul codice operativo delle istruzioni
     Esempio:

```
         CSECT
         BALR      9, Ø
         USING     *, 9
         BR        3, 4
         B         B1
         DS        ØF
         DS        CL1
A        DS        CL4
B1       LA        4, *
         ...
         ...
         ...
```

     l'assemblatore allinea l'istruzione di nome B1.

   - non provvede all'allineamento su un indirizzamento implicito riferito ad un'area non allineata alla mezza parola e specificato in un'istruzione "registro memoria" (tipo RX, RR e RS)

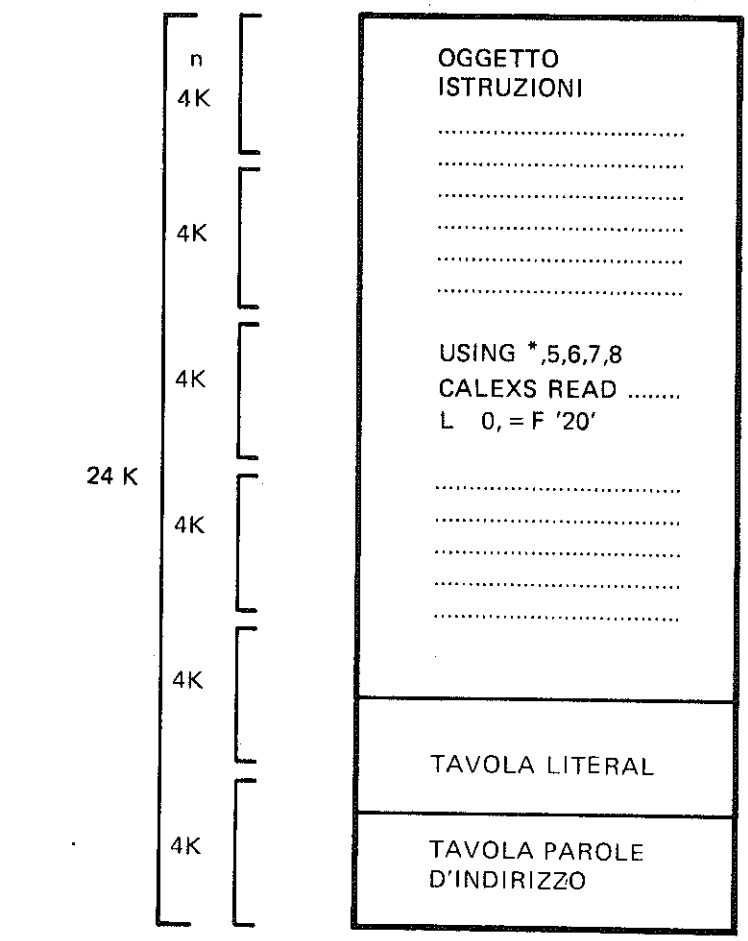
Esempio:

```
        CSECT
        BALR    9, Ø
        USING   *, 9
        ST      2, A
        BR      3, 4
        B       B1
        DS      ØF
        DS      CL1
A       DS      CL4
B1      LA      4, *
        ...
        ...
        ...
```

l'assemblatore segnala un messaggio, nel listing della diagnostica, relativo all'istruzione ST.

2. E' opportuno ricordare che la tavola dei literal e la tavola delle parole di indirèttezza vengono poste in coda all'oggetto delle istruzioni nel codice oggetto (vedi capitolo 5 output dell'assemblatore). Questo implica che a fronte di un riferimento ad un modulo esterno o di sistema oppure ad un literal, nel programma Assembler, l'Assemblatore risolve questi riferimenti specificando gli indirizzi (espressi in termini di registro base e spiazzamento) delle tavole relative. E' quindi cura dell'utente assicurarsi che, quando si richiama un modulo esterno o si usa un literal, le tavole siano indirizzabili con i registri base; in altre parole si deve calcolare la dimensione dell'oggetto istruzioni e delle tavole e determinare quanti registri base sono necessari per indirizzare le tavole stesse.

Esempio



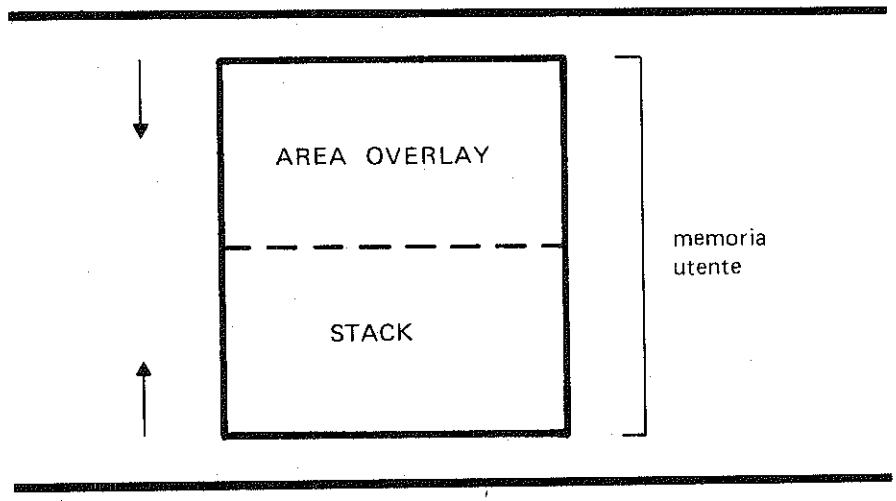
Supponendo che i riferimenti alle tavole vengano solo fatti nel terzo modulo di 4 K dell'oggetto istruzioni, le istruzioni che contengono questi riferimenti devono essere precedute dalla specificazione di almeno 4 registri base; infatti 4 registri base possono indirizzare 4096 x 4 posizioni di memoria.

Memoria utente

La memoria utente è suddivisa in due parti: la prima è l'area di overlay dove viene caricato il programma utente, la seconda è lo stack che rappresenta una "pila" sulla quale vengono inserite ed estratte informazioni da parte del programma utente.

L'area di overlay e lo stack sono adiacenti. Ad essi viene riservata una zona di memoria la cui estensione varia al variare dell'area disponibile dell'utente. L'area di overlay viene occupata dall'alto verso il basso, lo stack in senso opposto. Si ha condizione di errore solo quando una delle due aree cresca fino ad invadere la parte effettivamente impegnata dall'altra.

La rappresentazione grafica della memoria utente si può sintetizzare nel seguente modo:



La linea tratteggiata indica che il taglio tra lo stack e l'area overlay è dinamico in funzione della dimensione del programma utente.

L'area di overlay contiene di volta in volta gli overlay in cui possono essere strutturati i programmi; il singolo overlay identifica il segmento di programma.

6. INPUT/OUTPUT CON PIOCS DI SISTEMA

I programmi utente hanno la possibilità di utilizzare alcune funzioni standard del PIOCS (Phisical Input Output Control Section) di sistema, per realizzare le operazioni di lettura e scrittura su periferiche.

Le funzioni standard del PIOCS di sistema vengono svolte da moduli di sistema che vengono richiamati, da programma utente, con il seguente formato:

CALEXS modulo-sistema (argomenti)

L'esecuzione di questi moduli provoca la variazione dei registri generali Ø e 1. Il programma utente che utilizza questi moduli di sistema deve controllare le condizioni di errore presenti nell'ultimo argomento; l'assenza di errore viene segnalata col valore zero.

Elenco e funzioni dei moduli di sistema

I moduli di sistema e le loro funzioni sono i seguenti:

Nome	Funzione
WRITE	Scrive su un file di lavoro su floppy disk
READ	Legge da un file di lavoro su floppy disk
RKB	Richiede l'input da tastiera
DISP	Esegue un output su display
PRINT	Esegue la stampa di una linea su stampante integrata

Il passaggio degli argomenti avviene per indirizzo; dove non specificato ulteriormente gli argomenti sono descritti su campi di quattro byte.

WRITE

Funzione

Scrittura su un file di lavoro su floppy disk.

Formato

CALEXS WRITE, (buffer, offset, length, flag)

dove:

buffer indica l'indirizzo del buffer di memoria; il buffer di memoria deve essere dichiarato dall'utente

offset indica lo spiazzamento sul file di lavoro su floppy disk (in byte)

length indica la lunghezza del trasferimento (in byte)

flag rappresenta il flag d'errore; i valori che può assumere ed i corrispondenti tipi di errore sono i seguenti:

1 errore fisico
2 overflow di sistema
3 file di lavoro inesistente

Azione

Il modulo comunica al sistema di trasferire il contenuto del buffer di memoria (argomento buffer) nel file di lavoro su floppy disk, specificato nel comando EXQ, alla posizione indicata dall'argomento offset per una lunghezza pari a quanto specificato dall'argomento length. Il test sulla correttezza dell'operazione viene fatto sull'argomento flag.

READ

Funzione

Lettura da file di lavoro su floppy disk.

Formato

CALEXS READ (buffer, offset, length, flag)

dove:

buffer indica l'indirizzo del buffer di memoria; il buffer di memoria deve essere dichiarato dall'utente

offset indica lo spiazzamento sul file di lavoro su floppy disk (in byte)

length indica la lunghezza del trasferimento (in byte)

flag rappresenta il flag di errore; i valori che può assumere ed i corrispondenti tipi di errore sono i seguenti:

1 errore fisico
2 overflow di sistema
3 file di lavoro inesistente

Azione

Il modulo comunica al sistema di trasferire nel buffer di memoria (argomento buffer) il contenuto del file di lavoro su floppy disk, specificato nel comando EXQ, dalla posizione indicata dall'argomento offset per una lunghezza pari a quanto specificato dall'agomento length. Il test sulla correttezza dell'operazione viene fatto sull'argomento flag.

RKB

Funzione

Permette l'input da tastiera.

Formato

CALEXS RKB, (buffer, length-set)

dove:

buffer — indica l'indirizzo del buffer di memoria; il buffer di memoria deve essere dichiarato dall'utente

length-set — indica la lunghezza del trasferimento massimo (in byte); a trasferimento avvenuto, contiene la lunghezza effettiva del dato trasferito.

Azione

Il modulo comunica al sistema di trasferire nel buffer di memoria (argomento buffer), per una lunghezza specificata dall'argomento length-set, i dati introdotti da tastiera.

DISP

Funzione

Permette l'output su display.

Formato

CALEXS DISP, (buffer, length, offset, flag)

dove:

buffer indica l'indirizzo del buffer di memoria; il buffer di memoria deve essere dichiarato dall'utente

length indica la lunghezza del trasferimento

offset indica lo spiazzamento rispetto all'origine del buffer display, in cui la stringa di output viene posta; esistono le seguenti possibilità:

- offset = Ø comanda la visualizzazione dalla prima colonna
- offset ≠ Ø comanda la visualizzazione in coda ai dati già presenti

flag rappresenta il flag di errore; assume il valore 1 a fronte di un display fuori margini (> 80).

Azione

Il modulo comunica al sistema di trasferire il contenuto del buffer di memoria (argomento buffer), per una lunghezza pari all'argomento length, sul display nella posizione indicata dall'argomento offset. Il test sulla correttezza dell'operazione viene fatto sull'argomento flag.

PRINT

Funzione

Permette la stampa di una linea sulla stampante.

Formato

CALEXS PRINT, (buffer, length, flag)

dove:

buffer indica l'indirizzo del buffer di memoria;

length indica la lunghezza dell'area da stampare

flag rappresenta il flag di errore; assume il valore 1 se la stampante è fuori servizio.

Azione

Il comando comunica al sistema di trasferire il contenuto del buffer di memoria (argomento buffer), per una lunghezza specificata dall'argomento length, sulla stampante. Il test sulla correttezza dell'operazione viene fatto sull'argomento flag.

Nota

La stampante, alla quale la print si riferisce, è dipendente dalla configurazione del sistema (vedi comando CONFIGURE).

7. STATO CALCOLI IMMEDIATI

Il P6060 può essere utilizzato per eseguire calcoli istantaneamente, quando il sistema è nello stato calcoli immediati. Si può commutare il sistema dallo stato comandi allo stato calcoli immediati, premendo il tasto di console CALC MODE . Il sistema rimane nello stato suddetto finchè non si preme di nuovo il tasto CALC MODE , che ricommuta il sistema nello stato comandi.

Molti programmi che sono eseguiti possono fornire dei risultati errati a causa di errori nella loro struttura logica. Fornendo la possibilità di verificare immediatamente i risultati di un programma -- mentre il programma è ancora nella memoria principale -- lo stato calcoli immediati offre un mezzo estremamente rapido e conveniente per scoprire degli errori di programmazione.

Quando il sistema P6060 è nello stato calcoli immediati si può introdurre:

- un comando di sistema
- un comando di richiamo di un programma di utilità
- una espressione da eseguire immediatamente

I comandi di sistema sono introdotti ed eseguiti come spiegato nel capitolo 3. Quando è eseguito un comando, il sistema ricommuta automaticamente nello stato comandi.

I comandi di richiamo dei programmi di utilità sono introdotti ed eseguiti come spiegato nella appendice A. Quando il programma è eseguito, il sistema ricommuta automaticamente nello stato comandi.

Le espressioni tipiche dello stato calcoli immediati sono descritte nei paragrafi che seguono.

Introduzione ed esecuzione di espressioni nello stato calcoli immediati

Quando il P6060 è nello stato calcoli immediati, si può richiedere al sistema di:

- eseguire una espressione numerica
- assegnare agli argomenti delle funzioni trigonometriche una particolare unità di misura (radianti, gradi sessagesimali o centesimali)
- assegnare un contenuto predefinito ai tasti funzione.

Espressioni numeriche nello stato calcoli immediati

Una espressione numerica è composta da un insieme di operandi uniti da uno o più operatori. La valutazione di una espressione numerica fornisce come risultato un numero. Un operando numerico può essere una:

- costante numerica
- variabile numerica
- funzione numerica di sistema
- funzione numerica definita dall'utente
- espressione numerica racchiusa tra parentesi.

Costanti numeriche

Una costante numerica, nel sistema P6060, è un numero espresso nel sistema decimale. Come si può vedere negli esempi seguenti, una costante numerica può essere introdotta, visualizzata o stampata in uno dei tre formati: intero, virgola fissa o virgola mobile.

Formato intero	In virgola fissa	In virgola mobile
5	5.0	5.0 E +0
-5	-5.0	-5.0 E +0
+5	+5.0	+5.0 E +0

Formato intero

Una costante numerica espressa nel formato intero consiste in una o più cifre precedute, opzionalmente, dal segno. Da tastiera si può introdurre una costante numerica intera con non più di 13 cifre. Il massimo numero che si può introdurre è 9999999999999.

Formato in virgola fissa

Una costante numerica espressa nel formato in virgola fissa consiste in una o più cifre separate dal punto decimale e precedute, opzionalmente, dal segno. Una

costante numerica nel formato in virgola fissa può essere introdotta da tastiera con non più di 13 cifre.

Nota: La costante π è una costante interna del sistema. Al contrario delle altre costanti numeriche essa è richiamata da tastiera mediante il nome PI. Il suo valore è 3.141592653590.

Formato in virgola mobile

Una costante numerica nel formato in virgola mobile (noto anche come "notazione scientifica") consiste in un numero intero od in virgola fissa seguito da E e da una o due cifre, opzionalmente precedute dal segno. Il massimo numero che si può introdurre da tastiera è: 9.999999999999E+99.

Rappresentazione interna

La rappresentazione interna di un numero è la forma con cui è rappresentato in memoria principale. Tutti i numeri sono rappresentati in memoria principale con due parti: la mantissa e l'esponente. Per esempio 5665 è rappresentato in memoria principale come: 5.665000000000E+03 di cui 5.665000000000 è la mantissa e 03 è l'esponente. La mantissa viene normalizzata in modo che essa sia sempre composta da una cifra compresa tra 1 e 9, seguita da 12 cifre decimali. Il valore rappresentato in memoria principale è dato dal prodotto della mantissa per 10 elevato all'esponente indicato.

Campo della rappresentazione interna

I numeri che possono essere elaborati devono rientrare in campo di definizione da -9.999999999999E+99 a -1E-99, lo zero, da +1E-99 a +9.999999999999E+99, come indicato nella figura 7-1; la parte tratteggiata indica i numeri non definiti nella rappresentazione interna.

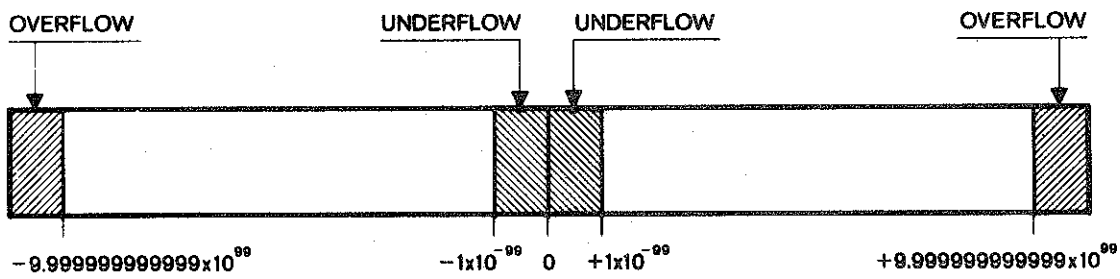


Figura 7-1 Campo della rappresentazione interna dei numeri

I numeri minori di -9.999999999999E99 e maggiori di 9.999999999999E99 appartengono alle zone di OVERFLOW, ossia esprimono grandezze in valore assoluto più grandi di quelle rappresentabili in memoria principale.

I numeri maggiori di -1E-99 ed inferiori a +1E-99, ma diversi da zero, appartengono alle zone di UNDERFLOW, ossia esprimono grandezze in valore assoluto più piccole di quelle rappresentabili in memoria principale.

Se durante l'elaborazione di una espressione si ottiene un risultato parziale con valore compreso in una zona di OVERFLOW, l'esecuzione della espressione è portata a termine assumendo come valore intermedio -9.999999999999E99 oppure 9.999999999999E99. Il risultato finale è stampato, se il tasto di console PRINT ALL è illuminato, ed il sistema visualizza il messaggio di errore ERROR 3.

Se durante l'elaborazione di una espressione si ottiene un risultato parziale con valore compreso in una zona di UNDERFLOW, l'esecuzione della espressione è portata a termine assumendo come valore intermedio zero. Il risultato finale è stampato, se il tasto di console PRINT ALL è illuminato, ed il sistema visualizza il messaggio di errore ERROR 4.

Variabili numeriche

Una variabile numerica è un dato numerico, identificato da un nome, il cui valore è soggetto a cambiare durante l'esecuzione di una espressione. Nello stato calcoli immediati si possono utilizzare quattro variabili numeriche il cui nome è: Φ, ΦØ, Φ1, Φ2.

Per assegnare, da tastiera, un valore numerico ad una variabile, si deve premere il tasto [RESULT] ed, eventualmente, una delle tre cifre: Ø, 1 o 2; quindi il tasto [=] seguito dal numero da assegnare alla variabile.

Vediamo, ad esempio, come si assegna il valore 10 ad cgnuna delle 4 variabili e cosa appare sul display come risultato di cgni assegnazione:

Quando si preme	Sul display appare
[RESULT] [=] [1] [0] [END OF LINE]	Φ=10
[RESULT] [0] [=] [1] [0] [END OF LINE]	ΦØ=1Ø
[RESULT] [1] [=] [1] [0] [END OF LINE]	Φ1=1Ø
[RESULT] [2] [=] [1] [0] [END OF LINE]	Φ2=1Ø

Per assegnare un valore ad una variabile come risultato dell'esecuzione di una espressione, si deve introdurre il nome della variabile seguita dal segno uguale e dall'espressione. Per esempio, premendo:

[RESULT] [2] [=] [RESULT] [+] [RESULT] [0] [+] [RESULT] [1] [-] [RESULT] [2] [END OF LINE]

il risultato della esecuzione dell'espressione Φ + ΦØ + Φ1 - Φ2 è assegnato alla variabile Φ2.

Vi sono due modi per visualizzare il valore corrente di una variabile. Con il primo si visualizza il valore della variabile è contemporaneamente si assegna tale valore alla variabile Φ; con il secondo si visualizza soltanto il valore della variabile.

1. Per visualizzare il valore di una variabile ed assegnare il suo valore alla variabile Φ, si introduca il nome della variabile e si prema [END OF LINE].

2. Per visualizzare il valore di una variabile senza variare il valore della variabile Φ, si utilizzi il seguente formato: variabile=variabile e quindi

si prema [END OF LINE] . Per esempio, per visualizzare il valore della variabile Φ Ø, si prema [RESULT] [0] [END OF LINE]

Nota: Ogni volta che si commuta il sistema nello stato calcoli immediati (premendo [CALC MODE]), le quattro variabili suddette sono eguagliate a zero. Se il sistema è nello stato calcoli immediati e si vogliono eguagliare a zero le quattro variabili contemporaneamente, si prema due volte [CALC MODE] -- la prima volta si esce dallo stato calcoli immediati, la seconda volta vi si rientra.

La variabile Φ come totalizzatore

Se si introduce un valore numerico o una espressione numerica da tastiera premendo il tasto [SUM], invece del tasto [END OF LINE], la variabile Φ si comporta come un totalizzatore (accumulatore). Il valore introdotto o il risultato della espressione è aggiunto al valore precedente della variabile Φ.

L'impiego della variabile Φ come totalizzatore è mostrato negli esempi che seguono (si ricordi che quando si preme il tasto [CALC MODE] per passare nello stato calcoli immediati, tutte le variabili sono automaticamente eguagliate a zero.

Quando si preme	Si pone
[CALC MODE]	Φ=Ø, ΦØ=Ø, Φ1=Ø, Φ2=Ø
[RESULT] [=] [3] [END OF LINE]	Φ=3
[RESULT] [=] [9] [SUM]	Φ=12

Come si vede, premendo [SUM] si aggiunge 9 al precedente valore di Φ.

Quando si preme	Si pone
[CALC MODE]	Φ=Ø, ΦØ=Ø, Φ1=Ø, Φ2=Ø
[RESULT] [=] [1] [0] [END OF LINE]	Φ=1Ø ΦØ=Ø
[RESULT] [0] [=] [1] [0] [SUM]	Φ=2Ø ΦØ=1Ø

Come si vede, premendo [SUM] per assegnare il valore 1Ø a Φ Ø, si aggiunge anche 1Ø al precedente valore di Φ.

Quando si preme	Si pone
CALC MODE	Φ=Ø, ΦØ=Ø, Φ1=Ø, Φ2=Ø
RESULT 1 = 1 0 END OF LINE	Φ=Ø Φ1=1Ø
RESULT 2 = 1 0 0 END OF LINE	Φ=Ø Φ1=1Ø Φ2=1ØØ
RESULT = 2 0 END OF LINE	Φ=2Ø Φ1=1Ø Φ2=1ØØ
RESULT = RESULT 1 + RESULT 2 SUM	Φ=13Ø Φ1=1Ø Φ2=1ØØ

Come si vede, premendo SUM si aggiunge il valore di Φ1 + Φ2 11Ø al precedente valore di Φ(2Ø).

Quando si preme	Si pone
CALC MODE	=Ø, ΦØ=Ø, Φ1=Ø, Φ2=Ø
RESULT 1 = 1 0 END OF LINE	Φ=Ø Φ1=1Ø
RESULT 2 = 1 0 0 END OF LINE	Φ=Ø Φ1=1Ø Φ2=1ØØ
RESULT = 2 0 END OF LINE	Φ=2Ø Φ1=1Ø Φ2=1ØØ
RESULT 2 = RESULT 1 + RESULT 2 SUM	Φ=13Ø Φ1=1Ø Φ2=11Ø

Come si vede, premendo SUM non solo si assegna il valore di Φ1 e Φ2 a Φ2, ma si aggiunge anche tale valore al precedente valore di Φ.

Per riassumere brevemente, il tasto END OF LINE serve per assegnare un valore ad una variabile; il tasto SUM serve sia per assegnare un valore ad una variabile che, contemporaneamente, per aggiungere tale valore al precedente valore della variabile Φ.

Operatori numerici

Gli operatori numerici difiniscono quale operazione deve essere eseguita sui valori numerici degli operandi specificati. Essi producano come risultato un numero. Gli operatori numerici che si possono utilizzare sono:

Operatore numerico	Funzione
↑	Elevamento a potenza
/	Divisione

*	Moltiplicazione
+	Addizione e segno più
-	Sottrazione e segno meno

Le espressioni numeriche sono eseguite secondo il livello di priorità degli operatori che le costituiscono. Le operazioni con il più alto livello di priorità sono eseguite per prime; quelle con lo stesso livello di priorità sono eseguite nell'ordine da sinistra a destra.

Il P6060 osserva le regole dell'algebra per la definizione del livello di priorità di esecuzione di una operazione nell'ambito di una espressione numerica, per cui i livelli di priorità sono:

Operatore numerico	Livello di priorità
↑	Il più alto
* e /	↓
+ e -	il più basso

Le parentesi (e) possono essere usate per cambiare l'ordine di esecuzione delle operazioni nell'ambito di una espressione numerica. Una espressione racchiusa tra parentesi è trattata come un singolo elemento numerico: viene valutata per ottenere il suo valore numerico, quindi tale valore è utilizzato per la valutazione della parte restante di una espressione più complessa di cui essa fà parte. Se più di una espressione è compresa tra parentesi, il calcolo inizia con la valutazione delle parentesi più interne. Nel seguito diamo ulteriori informazioni relative agli operatori numerici.

Elevamento a potenza

RESULT 1 ↑ RESULT 2	Indica che il valore di Φ1 deve essere elevato alla potenza di esponente dato da Φ2
Se Φ1= Φ2=Ø	Φ1 ↑ Φ2 è uguale ad uno
Se Φ1=Ø e Φ2<Ø	Si ha una segnalazione di OVERFLOW
Se Φ1<Ø e Φ2 non è intero	Si ha una segnalazione di errore

Se Φ1=Ø e Φ2=Ø Φ1 ↑ Φ2 è uguale ad uno

Se Φ1=Ø e Φ2>Ø Φ1 ↑ Φ2 è uguale a zero

Moltiplicazione ed addizione

RESULT 1 * RESULT 2 equivale a RESULT 2 * RESULT 1

RESULT 1 + RESULT 2 equivale a RESULT 2 + RESULT 1

Come si vede, per la moltiplicazione e l'addizione, vale la proprietà commutativa.

RESULT 1 * (RESULT 2 * RESULT)

non equivale sempre a

(RESULT 1 * RESULT 2) * RESULT

RESULT 1 + (RESULT 2 + RESULT)

non equivale sempre a

(RESULT 1 + RESULT 2) + RESULT

perchè l'operazione tra parentesi in alcuni casi può dare un risultato troncato od arrotondato.

Divisione e sottrazione

RESULT 1 / RESULT 2 Indica Φ1 diviso per Φ2

Se Φ2=Ø Si ha segnalazione di OVERFLOW

RESULT 1 - RESULT 2 Indica Φ1 meno Φ2

Segno

- RESULT + (- RESULT 1) + RESULT 2 - (- RESULT 0)

è ammesso

RESULT 1 + - RESULT 2 non è ammesso

Il segno + ed il segno - possono essere usati dopo una parentesi aperta o prima di una espressione numerica.

Nota: Due operatori numerici non possono essere usati uno di seguito all'altro ma devono essere separati da parentesi.

Funzioni

Nello stato calcoli immediati si possono utilizzare funzioni numeriche di sistema o definite dall'utente.

Funzioni numeriche di sistema

Le funzioni numeriche di sistema possono essere utilizzate introducendo da tastiera il loro nome e (tra parentesi) l'argomento su cui si applicano. L'argomento può essere una costante numerica, una variabile numerica o comunque, in generale una espressione numerica.

Nella tabella seguente sono riassunte, in ordine alfabetico, le funzioni numeriche di sistema disponibili; la lettera X indica l'argomento della funzione.

Funzione numerica	Descrizione
ABS(X)	Valore assoluto di X
ACS(X)	Arcocoseno (in radianti) di X
ASN(X)	Arcoseno (in radianti) di X
ATN(X)	Arcotangente (in radianti) di X
COS(X)	Coseno di X radianti
COT(X)	Cotangente di X radianti
EXP(X)	Esponenziale in base e di X
HCS(X)	Coseno iperbolico di X radianti
HSN(X)	Seno iperbolico di X radianti
HTN(X)	Tangente iperbolica di X radianti
INT(X)	Parte intera di X
LGT(X)	Logaritmo in base 10 di X
LOG(X)	Logaritmo naturale di X
RND	Numero casuale compreso tra zero ed uno
SGN(X)	Segno di X (+1 per X positivo, Ø per Ø,-1 per X negativo)

Funzione numerica	Descrizione
SIN(X)	Seno di X radianti
SQR(X)	Radice quadrata di X
TAN(X)	Tangente di X radianti

Tabella 7-1 Funzioni numeriche di sistema

Si possono inoltre utilizzare le seguenti funzioni:

LEN (string-exp) che ritorna il numero di caratteri di string-exp

SCN (string-exp, substring, n-occurrence, start-position) che ritorna la posizione dell'ennesima occorrenza il string-exp della sottostringa substring a partire dalla posizione specificata con start-position.

Funzioni definite dall'utente

Il P6060 permette di definire le funzioni di frequente impiego e di assegnarle ad uno degli 8 tasti della sezione funzioni definibili.

Si possono assegnare due funzioni per ogni tasto suddetto (utilizzando il tasto SHIFT) quindi si ha la possibilità di avere contemporaneamente fino a 16 funzioni. Ogni funzione può essere definita con al massimo 73 caratteri, ma il numero totale dei caratteri che definiscono tutte le funzioni assegnate ai tasti funzione non può essere maggiore di 238.

Vediamo con un esempio come si definisce ed assegna una funzione ad un tasto funzione, mentre il sistema è nello stato calcoli immediati.

Supponiamo di voler calcolare le radici reali di una equazione di secondo grado del tipo ax^2+bx-c utilizzando i tasti funzione. Si deve procedere nel seguente modo. Alla variabile Φ2 si assegneranno i coefficienti di tipo a. Alla variabile Φ1 si assegneranno i coefficienti di tipo b. Alla variabile ΦØ si assegneranno i coefficienti di tipo c. Quindi alla chiave F1 si assegni la stringa:
(- Φ1 + SQR(Φ1 * Φ1 - 4 * Φ2 * ΦØ))/(2 * Φ2)
premendo i tasti SHIFT FKEY# 1 . e di seguito i tasti

equivalenti alla stringa suddetta, seguiti da [END OF LINE] .
Alla chiave F2 si assegni la stringa:

(- Φ1 - SQR(Φ1 * Φ1 - 4 * Φ2 * ΦØ))/(2 * Φ2)

premendo i tasti [SHIFT] [FKEY#] (2) [.] e di seguito i tasti equivalenti alla stringa suddetta, seguiti da [END OF LINE] .

Per calcolare le radici reali di una equazione particolare, si introducano da tastiera i valori dei coefficienti assegnandoli nell'ordine alle variabili:

Φ2 = a
Φ1 = b
ΦØ = c

Dopo di che premendo: [RESULT] [=] [F1] [END OF LINE]

si ottiene, assegnandolo a Φ , il valore di una radice.

Premendo: [RESULT] [=] [F2] [END OF LINE]

si ottiene il valore della seconda radice.

Con analoghe procedure si possono assegnare ai tasti funzione altre espressioni numeriche. Premendo un solo tasto tali espressioni sono richiamate nel buffer di tastiera ed eventualmente inserite in espressioni più complesse.

Nota: Nello stato calcoli immediati si possono utilizzare, premendo i tasti funzione, le stesse espressioni assegnate ad essi dal programma presente in memoria principale.

Visualizzazione dei risultati

Il risultato della esecuzione di una espressione è visualizzato immediatamente sul display e, se il tasto di console [PRINT ALL] è acceso, stampato sul tabulato della stampante integrata. I risultati possono essere visualizzati (o stampati) in diversi formati che dipendono dalla posizione dell'indicatore dei decimali (vedi capitolo 1)

1. Quando l'indicatore dei decimali è nella posizione Ø (lo Ø è visibile nella relativa finestrella) non

vengono visualizzate (stampate) le cifre decimali.

2. Quando l'indicatore dei decimali è in una posizione da 1 a 13 (la relativa cifra è visibile nella suddetta finestrella), vengono visualizzate (stampate) da 1 a 13 cifre dopo il punto decimale se ciò è possibile.

3. Quando l'indicatore dei decimali è nella posizione Flt (Flt è visibile nella relativa finestrella), il numero è visualizzato nel formato interno ossia nella notazione scientifica (vedi paragrafo "Rappresentazione interna).

4. Quando l'indicatore dei decimali è nella posizione ST (ST è visibile nella finestrella):

   - i numeri interi sono visualizzati (stampati) con al massimo 8 cifre significative

   - i numeri con valore assoluto compreso tra 0.0999999995 e 99999999.4 sono visualizzati (stampati) con al massimo 8 cifre significative (se il numero è minore di 1 viene tralasciato lo zero che precede la parte decimale) nel formato in virgola fissa

   - i numeri con valore assoluto minore di 0.0999999995 che possono essere rappresentati con 8 cifre significative, sono visualizzati (stampati) nel formato in virgola fissa

   - tutti gli altri numeri sono rappresentati con al massimo 8 cifre significative nel formato in virgola mobile.

Nota: L'indicatore dei decimali controlla anche il formato di visualizzazione e stampa del contenuto delle variabili numeriche Φ, ΦØ, Φ1, Φ2.

L'unità di misura degli angoli

Nel sistema P6060 gli angoli sono misurati normalmente in radianti; quindi, quando si forniscono gli argomenti alle funzioni trigonometriche, i valori introdotti sono assunti dal sistema come espressi in radianti. (Anche il valore ritornato della funzione ATN è espresso in radianti.)

Nello stato calcoli immediati è possibile scegliere l'unità di misura degli angoli, comunicando al sistema quella voluta, mediante la digitazione di uno dei seguenti comandi di predisposizione: SDEG, SGRAD, SRAD. (Quando si introduce SDEG o SGRAD, la luce di console DEG GRAD si accende. Essa rimane accesa finchè si introduce SRAD, si esce dallo stato calcoli immediati, oppure il P6060 è spento.) Se si preme (S) (D) (E) (G), i valori assegnati come argomenti alle funzioni trigonometriche sono misurati in gradi sessagesimali. (Anche il valore ritornato dalla funzione ATN è espresso in gradi sessagesimali.)

Si noti che i valori assegnati agli argomenti delle funzioni trigonometriche, dopo aver introdotto SDEG, non possono essere espressi in gradi, primi e secondi di grado, ma in termini di gradi, decimi, centesimi, millesimi etc. di grado.

Se si preme (S) (G) (R) (A) (D), i valori assegnati come argomenti alle funzioni trigonometriche sono interpretati in gradi centesimali. (Anche il valore ritornato dalla funzione ATN è espresso in gradi centesimali).

Se si preme (S) (R) (A) (D), i valori assegnati agli argomenti delle funzioni trigonometriche sono interpretati in radianti. (Anche ATN ritorna un valore espresso in radianti.) Se si preme due volte CALC MODE si ottiene lo stesso risultato, ma le variabili Φ, ΦØ, Φ1 sono eguagliate a zero.

Esempi di calcoli immediati

1. I seguenti esempi mostrano come l'impiego delle parentesi influisce sulla valutazione di una espressione:

Premendo	Sul display appare
(4) (+) (3) (*) (2) (−) (6) (/) (2) END OF LINE	7
((4) (+) (3)) (*) (2) (−) (6) (/) (2) END OF LINE	11
((4) (+) (3) (*) (2) (−) (6)) (/) (2) END OF LINE	2

Premendo	Sul display appare
((4 + 3) * 2 − 6) / 2 END OF LINE	4
2 ↑ 3 ↑ 2 END OF LINE	64
5 ↑ (3 ↑ 2) END OF LINE	512

2. L'esempio seguente mostra come si può calcolare la media di cinque numeri:

Premendo	
RESULT END OF LINE	Φ è uguagliata a <u>zero</u>
5 SUM	Φ = 5
6 SUM	Φ = 11
1 4 SUM	Φ = 25
3 SUM	Φ = 28
1 2 SUM	Φ = 4Ø
RESULT / 5 END OF LINE	Φ = 8

3. Negli esempi seguenti si mostra l'effetto prodotto posizionando l'indicatore dei decimali nelle posizioni indicate:

Premendo	Con l'indicatore dei decimali su	Sul display appare
P I END OF LINE	Ø	3
P I END OF LINE	1	3.1
P I END OF LINE	2	3.14
P I END OF LINE	13	3.1415926535900
P I END OF LINE	ST	3.1415927
P I END OF LINE	Flt	3.141592653590E+00

8. LO STATO DI DEBUGGING

Premessa

Lo stato di debugging del P6060 offre un insieme completo di strumenti per l'analisi e la verifica dei programmi. Nello stato di debugging, l'operatore ha un completo controllo sulla esecuzione di un programma. Si può fermare e riprendere l'esecuzione del programma, si possono visualizzare e modificare registri e aree di memoria, si può controllare il flusso logico del programma, si può vedere la frequenza di esecuzione delle istruzioni del programma, si può, insomma, vedere e tracciare l'esecuzione del programma.

Come accedere allo stato debugging

Il sistema può eseguire programmi sotto il controllo del debugging solo se l'opzione di debugging è caricata in memoria.

Il caricamento delle routine in memoria viene realizzato con il lancio del comando OPT DEB (vedi comando OPTIONS). Dopo l'esecuzione del comando OPT DEB le routine di debugging sono caricate in memoria, ma non sono attivate; vengono attivate quando si specifica, nel comando EXQ l'operando D, quando durante l'esecuzione del programma si preme il tasto STEP oppure a fronte di un errore.

Quando un programma viene eseguito sotto il controllo del modulo di debugging, il suo tempo di esecuzione viene aumentato di circa un 20%.

Entrata in stato debugging: Il sistema commuta nello stato debugging quando: si esegue il comando EXQ con l'operando D, si rileva un errore durante l'esecuzione di un programma, quando si incontra un alt indirizzato, oppure si preme il tasto di console STEP. Quando il sistema è nello stato di debugging, il tasto di console STEP è illuminato.

1. Il comando EXQ con l'operando D (vedi capitolo 3), può essere introdotto durante lo stato comandi op-

pure durante lo stato calcoli immediati. Il programma presente in memoria principale viene lanciato ed il sistema commuta nello stato di debugging. Il tasto di console STEP si illumina.

2. Se, durante l'esecuzione di un programma, viene rilevato un errore, l'esecuzione del programma è interrotta, il sistema commuta nello stato di debugging e viene visualizzato un appropriato messaggio di errore sul display.

3. Quando si preme il tasto di console STEP, l'esecuzione di un programma viene interrotta. Il sistema commuta nello stato di debugging e sul display appare il messaggio. Se si preme di nuovo il tasto STEP, viene eseguita l'istruzione successiva e il sistema commuta di nuovo nello stato di debugging.

4. Al riconoscimento di un ALT INDIRIZZATO (vedi i comandi da tastiera del presente capitolo)

Se dopo la segnalazione d'errore su display, non si preme il tasto STEP, l'unica altra possibilità che rimane all'utente è di premere il tasto BREAK che manda in terminazione il programma e pone il sistema nello stato comandi.

Nota: Non è possibile utilizzare le facilities del debugging sull'oggetto degli eventuali moduli di sistema richiamati dal programma utente.

Quando si entra in stato debugging, a fronte di un errore del programma utente, vengono visualizzate su display le seguenti informazioni:

- il valore del program counter

- l'istruzione corrente (quella che deve essere eseguita)

- il condition code

Nel seguente formato:

iiiiii oooo oooo oooo CC=x

dove:

iiiiii = program counter
oooo = codice oggetto in esadecimale
x = valore del condition code

Se si verifica un errore durante l'esecuzione di un programma utente e non è stato lanciato il comando OPT DEB, il sistema segnala l'errore relativo su display ed il programma si ferma; l'unico modo per sbloccare il sistema è di premere il tasto BREAK che manda in terminazione il programma.

Strumenti dello stato debugging

Quando il programma è nello stato di debugging, si possono utilizzare i seguenti strumenti per ricercare la causa degli errori di un programma presente in memoria principale:

1. Comandi da tastiera nello stato debugging.
2. Tasti di console

Comandi da tastiera

Nello stato debugging è possibile introdurre da tastiera dei comandi che visualizzano o modificano aree di memoria e il contenuto dei registri. I comandi vengono introdotti da tastiera, carattere per carattere e ogni linea di comando viene chiusa dal tasto END OF LINE. La lunghezza della linea di comando non deve superare i 32 crt; in caso contrario verrà rifiutata con segnalazione acustica. Nel caso in cui il comando introdotto risulti sintatticamente scorretto viene data segnalazione di comando errato; premendo il tasto RECALL è possibile riottenere la linea in modo da poterla correggere e reintrodurre.

I comandi che si possono introdurre in stato debugging sono i seguenti:

- display istruzione corrente
- display registro
- modifica registro
- stampa registri
- somma un immediato ad un registro
- display memoria
- modifica memoria
- stampa memoria
- avanzamento in display
- modifica program counter
- alt indirizzato.

Errori in stato debugging: Introducendo i comandi da tastiera si può incorrere in errori di introduzione che possono essere così sintetizzati:

- comando errato, il comando appena introdotto non è sintatticamente corretto

- modifica errata, la modifica non è stata eseguita correttamente, più precisamente:

  . è stato modificato il campo indirizzo o il campo indicante il registro

  . i campi sono stati introdotti non nello stesso formato di output

  . sono stati introdotti caratteri non esadecimali.

A fronte di questi errori il sistema visualizza questa segnalazione:

INCORRECT FORMAT

e si mette in attesa dell'intervento dell'operatore; questi premendo il tasto RECALL può riottenere il comando errato in modo da poterlo correggere e reintrodurlo.

A fronte di un comando specificante un indirizzo fuori della memoria il sistema visualizza questa segnalazione:

MEMORY ADDRESSES EXCEEDED

e si mette in attesa dell'intervento dell'operatore; questi premendo il tasto RECALL può riottenere il comando errato in modo da poterlo correggere e reintrodurlo.

Nel seguito si da la descrizione funzionale dei comandi da tastiera.

Display istruzione corrente:

Funzione: visualizza l'istruzione corrente; l'istruzione corrente viene pure visualizzata ad ogni entrata in stato debugging.

Formato: (EOL)

Output: *iiiiii* oooo oooo oooo CC=x

dove:

iiiiii = program counter
oooo = codice oggetto in esadecimale
x = valore del condition code
CC=** = caratteri alfanumerici fissi

Messaggi di errore: *INCORRECT FORMAT*

Display registro:

Funzione: visualizza, in esadecimale, il contenuto del registro indicato.

Formato: Ry [y] (EOL)

Output: *Reg.yy* xxxx xxxx

dove:

yy = numero del registro richiesto
xxxx = contenuto, in esadecimale, del registro
Reg.= caratteri alfanumerici fissi

Messaggi di errore: *INCORRECT FORMAT*

Modifica registro:

Funzione: Viene visualizzato, in esadecimale, il contenuto del registro indicato, con il pointer posizionato in ultima colonna. In questa situazione è possibile editare soltanto sul campo che indica il contenuto del registro. Per rinunciare alla modifica è necessario digitare il tasto

Formato: Ry [y] M (EOL)

Output: *Reg.yy* xxxx xxxx

dove:

yy = numero del registro richiesto

xxxx = contenuto, in esadecimale del registro
Reg. = caratteri alfanumerici fissi

Messaggi di errore:=*INCORRECT FORMAT*

Stampa registri:

Funzione: stampa, in esadecimale, il contenuto di tutti i registri.

Formato: R (EOL)

Output:
```
*Reg.00* xxxx xxxx ..... *Reg.12* xxxx xxxx
*Reg.01* xxxx xxxx ..... *Reg.13* xxxx xxxx
*Reg.02* xxxx xxxx ..... *Reg.14* xxxx xxxx
*Reg.03* xxxx xxxx ..... *Reg.15* xxxx xxxx
```

dove:

xxxx = contenuto, in esadecimale, del registro
Reg.nn = caratteri alfanumerici fissi
nn = numero del registro

Messaggi di errore: *INCORRECT FORMAT*

Somma immediato a registro:

Funzione: esegue la somma binaria fra il contenuto di un registro (24 bits meno significativi) e una costante esadecimale, visualizzando su display il risultato. Il contenuto del registro non viene modificato.

Formato: Ry [y] + xxxxxx (EOL)

Output: xxxxxx

dove:

yy = numero del registro richiesto
xxxxxx = costante esadecimale (nel formato) e risultato dell'operazione (nell'output)

Messaggi di errore: *INCORRECT FORMAT*

Display memoria:

Funzione: visualizza 8 bytes contigui di memoria

Formato: iiiiii (EOL)

Output: *iiiiii* xxxx xxxx xxxx xxxx

dove:

iiiiii = indirizzo del primo byte
xxxx = contenuto, in esadecimale, della memoria
** = caratteri alfanumerici fissi

Messaggi di errore: *INCORRECT FORMAT*
MEMORY ADDRESSES EXCEEDED

Modifica memoria:

Funzione: visualizza 8 bytes contigui di memoria. Il pointer sarà in ultima posizione. Dopo tale comando sarà possibile editare sul contenuto della memoria ma non sul campo indirizzo. Per rinunciare alla modifica è necessario digitare il tasto CLEAR

Formato: iiiiii M (EOL)

Output: *iiiiii* xxxx xxxx xxxx xxxx

dove:

iiiiii = indirizzo del primo byte
xxxx = contenuto, in esadecimale, della memoria
** = caratteri alfanumerici fissi

Messaggi di errore: *INCORRECT FORMAT*
MEMORY ADDRESSES EXCEEDED

Stampa memoria:

Funzione: stampa, in esadecimale, il contenuto della memoria per tanti bytes quanti sono quelli indicati nel comando con arrotondamento a 16. Se tale parametro è stato omesso viene assunto il valore 16.

Formato: iiiiii L [yyy] (EOL)

Output: *iiiiii* xxxx xxxx xxxx xxxx -------- xxxx

dove:

iiiiii = indirizzo del primo byte
yyy = numero di bytes da stampare (in decimale)
xxxx = contenuto in esadecimale, della memoria
** = caratteri alfanumerici fissi

Messaggi di errore: *INCORRECT FORMAT*
MEMORY ADDRESSES EXCEEDED

Avanzamento in display: (memoria o registro)

Funzione: permette di visualizzare il registro o i byte successivi a quelli appena visti. Questo comando ha effetto solo dopo una visualizzazione o una visualizzazione con modifica.

Formato: N (EOL)

Output: Es.: se utilizzato dopo la visualizzazione di un registro (es.R2), provoca la visualizzazione del registro successivo (es.R3). Il registro successivo a R15 viene considerato il registro R0. Se utilizzato dopo la visualizzazione di memoria, provoca la visualizzazione degli 8 byte successivi di memoria.

Messaggi di errore: *INCORRECT FORMAT*
MEMORY ADDRESSES EXCEEDED

Modifica program counter:

Funzione: modifica il valore del program counter. In pratica opera come un branch incondizionato a un'altra parte del programma.

Formato: iiiiii J (EOL)

Output: *iiiiii* oooo oooo oooo CC=x

dove:

iiiiii = program counter modificato
oooo = codice oggetto in esadecimale
x = valore del condition code
CC=** = caratteri alfanumerici fissi

Messaggi di errore: *INCORRECT FORMAT*
MEMORY ADDRESSES EXCEEDED

Alt indirizzato:

Funzione: predispone un particolare indirizzo di BREAK-POINT all'interno del programma utente, al riconoscimento del quale la debugging viene invocata e si entra in stato debugging. Una nuova impostazione rimpiazza il BREAK-POINT precedente. Per annullare tale BREAK-POINT basta impostare il comando OS(EOL). L'indirizzo predisposto dovrà essere puntato sul codice operativo dell'istruzione, diversamente verrà ignorato. La ricerca del BREAK-POINT avviene dalla istruzione successiva a quella corrente.

Formato: iiiiii S (EOL)

Output: *iiiiii* oooo oooo oooo CC=x

dove:

iiiiii = program counter
oooo = codice oggetto in esadecimale
x = valore del condition code
CC=** = caratteri alfanumerici fissi

Messaggi di errore: *INCORRECT FORMAT*

Comandi da console

Tre tasti di console sono particolarmente utili come strumenti di verifica dei programmi: CONTINUE , STEP e TRACE .

Nel seguito è spiegato il loro impiego e sono dati alcuni suggerimenti che permettono di utilizzare, nello stato di debugging, le prestazioni offerte dai tasti di console BREAK e NO PRINT .

Se, mentre il sistema è nello stato di debugging, è premuto il tasto di console CONTINUE, riprende l'esecuzione del programma che è in memoria principale. Il tasto CONTINUE si illumina quando è premuto mentre il sistema è nello stato di debugging. La funzione del tasto è attiva solamente quando il sistema si trova nello stato di debugging.

Se, mentre il sistema è nello stato di debugging, è premuto il tasto di console STEP, si può eseguire passo a passo (ossia istruzione per istruzione) il programma presente in memoria principale. Ogni volta che una istruzione del programma è eseguita il sistema visualizza sul display l'istruzione logicamente successiva, che specifica quale istruzione del programma sarà eseguita alla successiva pressione del tasto STEP. Dopo di che, ogni volta che si preme il tasto STEP, viene eseguita la successiva istruzione.(Si ricordi che il tasto di console STEP si illumina ogni volta che il sistema è nello stato di debugging.)

Il tasto di console TRACE permette di visualizzare il codice oggetto di ogni istruzione eseguita, nell'ordine con cui essa è eseguita. (Il tasto TRACE è acceso quando la funzione omonima è attiva.) Per utilizzare la funzione TRACE nello stato di debugging:

1. Premere il tasto TRACE
2. Premere il tasto CONTINUE.

Come conseguenza, si riprende l'esecuzione del programma e vengono visualizzati i codici oggetto delle istruzioni eseguite. Si noti che non vengono stampati i codici delle istruzioni non esecutive, come * o DC, e l'esecuzione del programma è realizzata come quando la funzione TRACE non è attiva (la funzione TRACE può essere attivata anche durante lo stato comandi, premendo TRACE e quindi introducendo il comando EXQ).

Il codice oggetto di ogni istruzione eseguita viene visualizzato nel seguente formato:

iiiiii oooo oooo oooo CC=x

dove:

iiiiii = program counter
oooo = codice oggetto in esadecimale
x = valore del condition code
CC=** = caratteri alfanumerici fissi

Se si vuole tenere traccia delle visualizzazioni si prema il tasto PRINT ALL che fa stampare tutte le informazioni visualizzate.

Quando la funzione del tasto NO PRINT è attiva, la stampante integrata è inibita. Questa prestazione può essere utile per risparmiare tempo di stampa e carta. Quando la funzione NO PRINT è attiva, il tasto NO PRINT è illuminato e in questo caso viene disabilitata la eventuale funzione di PRINT ALL

Il tasto BREAK, premuto nello stato di debugging, permette di terminare l'esecuzione di un programma e commuta il sistema nello stato comandi. Nello stato comandi, si può modificare il programma presente in memoria principale oppure introdurre un qualsiasi comando di sistema. Quando la funzione BREAK è attiva, il tasto BREAK è illuminato.

Uscita dallo stato debugging

Si può uscire in seguito a una delle seguenti operazioni:

- premendo il tasto STEP in questo caso viene eseguita un'istruzione del programma utente, dopo di che si ritorna in stato debugging

- premendo il tasto CONTINUE ; in questo caso si riprende l'esecuzione del programma

- premendo il tasto BREAK ; in questo caso si passa direttamente alla terminazione del programma.

Output del debugging

I messaggi emessi, durante lo stato di debugging, vengono visualizzati su display a meno di quelli emessi a fronte di comandi di stampa da tastiera (stampa registro e memoria).

Se il sistema P6060 ha nella sua configurazione il Video, i messaggi, oltre a comparire su display compaiono anche su Video.

Nel caso che l'utente voglia tenere traccia dei messaggi emessi, può premere il tasto PRINT ALL che comanda al sistema di mandare su stampa i messaggi che compaiono sul display.

Se il sistema ha nella sua configurazione una stampante IPSO, la stampa dei messaggi viene effettuata sulla stampante integrata o IPSO a seconda di come si è configurato il sistema (vedi comando CONFIGURE).

Luci di console

Le lampadine, in stato debugging hanno il seguente significato:

- luce running

  . fissa quando il debugging è in attesa di un comando

  . pulsante mentre il debugging sta elaborando un comando

- luce overflow:

  . accesa quando si cerca di introdurre da tastiera più di 80 crt.

- de-grad:

  . sempre spenta

- on-line:

  . sempre spenta

- TRACE/NO PRINT/PRINT ALL:

  . accesa quando la predisposizione è in atto

Si da di seguito una tabella sintetizzante la situazione delle lampadine nei vari stati del sistema:

LUCE	STATO OFF	ESECUZIONE PROGRAMMA	DEBUGGING
RUNNING	0	*	1
STEP	0	0	1
CONTINUE	0	1	0
BREAK	0	0	0
CALC.MODE	0	0	0

dove:

0 = luce spenta
1 = luce accesa
* = luce pulsante

9. IMPIEGO DEI TASTI FUNZIONE

Agli 8 tasti funzione della sezione funzioni definibili, vedi figura 9-1, si possono assegnare, utilizzando anche il tasto [SHIFT], fino a 16 funzioni definite dall'utente. Questo capitolo riprende in modo organico i diversi aspetti sull'impiego dei tasti funzione, già descritti nei capitoli precedenti (vedi capitolo 3, comando LDKEYS e capitolo 7), per permettere al lettore di ritrovare facilmente le informazioni che gli permettono di sfruttare in modo completo questa prestazione importante del sistema P6060.

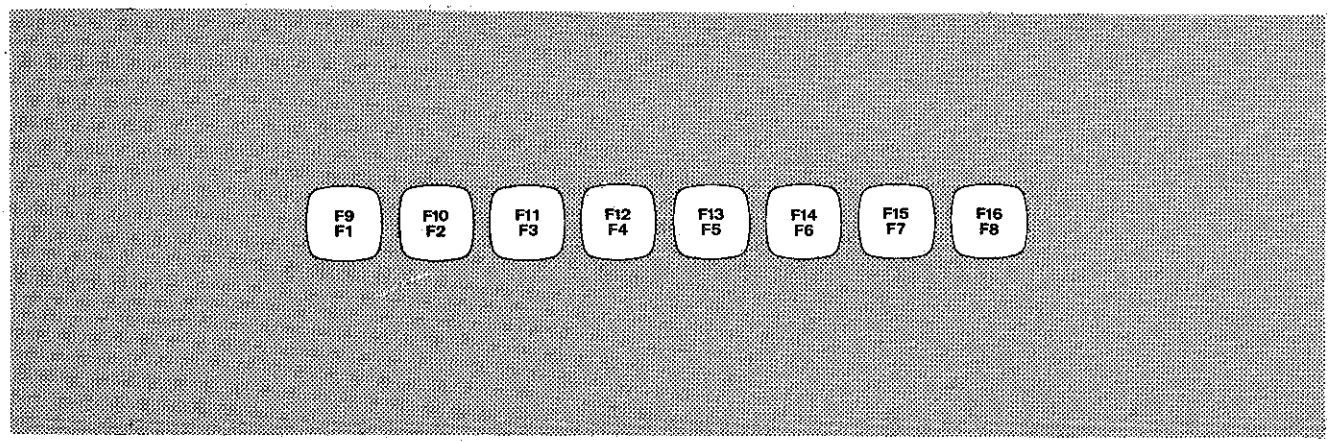


Figura 9-1 Tasti funzione

Assegnazione di una funzione ad un tasto funzione

L'assegnazione di una funzione ad un tasto funzione può essere effettuata durante due diversi stati del sistema, ma la struttura dell'assegnazione è sempre la seguente:

FKEY# n, stringa

dove

n
è un numero intero, da 1 a 16, che indica a quale tasto funzione (F1 ÷ F16) deve essere assegnata la stringa specificata

stringa
indica quale espressione numerica o stringa di caratteri deve essere assegnata al tasto funzione specificato.

L'impiego della assegnazione di una funzione ad un tasto funzione varia in relazione allo stato del sistema. Gli stati del sistema in cui si può assegnare ai tasti funzione una funzione specifica sono:

- lo stato calcoli immediati
- lo stato comando -- mediante il comando LDKEYS

Ad ogni tasto funzione si può assegnare una stringa di caratteri, od una espressione numerica, con non più di 73 caratteri. Le stringhe di caratteri, o le espressioni numeriche, assegnate ai tasti funzione, sono registrati in un registro (buffer) di 238 caratteri (byte); quindi il numero totale di caratteri assegnati a tutti i tasti funzione non può essere maggiore di 238.

Nota: Per ricordare l'assegnazione fatta ai tasti funzione, si può utilizzare una striscia di carta in cui vi sono 16 caselle nelle quali si possono scrivere dei codici mnemonici relativi ai tasti funzione sottostanti (vedi figura 9-1).

Assegnazione di una funzione ai tasti funzione durante lo stato calcoli immediati

Quando il sistema è nello stato calcoli immediati si possono assegnare ai tasti funzione delle espressioni o costanti utilizzate comunemente. L'assegnazione è effettuata come già descritto nel capitolo 7.

Esempio: Si usino i tasti funzione per l'assegnazione delle utility, dei comandi e delle istruzioni usate più comunemente.

```
FKEY#1,EXE ASS,IN=USLIB,START-,OUT=SYSLIB,TR--O,LIST,XREF
FKEY#2,EXE LNK,IN=SYSLIB,TR--O,OUT=USLIB,TR--E
FKEY#3,EXQ TR--E
```

Impiego di tasti funzione nello stato comandi

Durante lo stato comandi si possono registrare sul floppy disk sistema le stringhe di caratteri od espressioni numeriche che sono state assegnate ai tasti funzione. In questo modo si possono ripristinare ai tasti funzione le funzioni desiderate. Per poter ripristinare l'assegnazione delle funzioni ai tasti funzione, si ricorre al seguente comando di sistema:

STKEYS

come si è già visto nel capitolo 3 questo comando richiede al sistema di caricare in memoria principale il contenuto del buffer relativo ai tasti funzione. Quando il sistema è successivamente inizializzato il buffer relativo ai tasti funzione viene caricato dal floppy disk sistema con il contenuto ivi registrato mediante il comando STKEYS. Se tale contenuto associato ai tasti funzione è modificato da nuove assegnazioni ai tasti funzione effettuate durante lo stato calcoli immediati, esso può essere ripristinato mediante il comando LDKEYS (vedi capitolo 3). I comandi suddetti sono utili quando si ha una notevole mole di gestione delle sottolibrerie. In questi casi si assegnano ai tasti funzione i comandi che richiamano in memoria principale ed eseguono i programmi di utilità che gestiscono le sottolibrerie. Quindi si registrano le assegnazioni suddette sul floppy disk sistema mediante il comando STKEYS. Quando le assegnazioni registrate sul floppy disk sistema sono necessarie si possono richiamare in memoria principale mediante il comando LDKEYS. In questo modo non è necessario digitare ogni volta i comandi richiesti perchè basta premere l'appropriato tasto funzione per richiamare in memoria principale il programma di utilità richiesto.

A. PROGRAMMI DI UTILITA'

Il sistema P6060 è corredato di un insieme di programmi che permettono l'esecuzione di operazioni di servizio quali: inizializzazione dei floppy disk, gestione delle sottolibrerie presenti sul floppy disk sistema e sul floppy disk utente etc. Questi programmi sono detti di utilità e risiedono sul floppy disk sistema come parte del sistema operativo. I programmi di utilità sono richiamati in memoria principale ed eseguiti mediante il comando EXEC seguito dal nome del programma. I nomi e le funzioni dei programmi di utilità disponibili nel sistema sono:

Nome del programma	Funzione del programma
ASSEMBLY	Traduce un sorgente Assembler in oggetto rilocabile
FDCOPY	Copia il contenuto di un floppy disk su di un altro
FLCOPY	Copia un file da una sottolibreria in un'altra
LBCREATE	Inizializza la libreria software applicativo
LBPROTECT	Protegge la sottolibreria package e la sottolibreria comune presenti su di un floppy disk
LIBCOPY	Copia un'intera sottolibreria da un floppy disk ad un altro
LNK	Trasforma un oggetto rilocabile in oggetto eseguibile.

Quando uno dei suddetti programmi è caricato in memoria principale, il precedente contenuto della memoria utente è perduto. Se si vuole registrare il contenuto della memoria utente, si deve utilizzare il comando

SAVE o REPLACE. Si osservi che, quando viene caricato un programma di utilità in memoria principale, non viene alterato il contenuto relativo ai tasti funzione. I programmi di utilità, come abbiama visto, permettono di realizzare diverse funzioni tra cui, fondamentalmente, è quella di copiare le informazioni -- sottolibrerie, file e programmi -- da un floppy disk ad un altro. Nel predisporre il sistema per eseguire delle operazioni di copia, talvolta si determina nella unità floppy disk una configurazione di dischi tale da non permettere altre operazioni. Per esempio, per copiare il contenuto di un floppy disk su di un altro, ad un certo punto, si avranno nell'unità floppy disk due floppy disk utente. (Il funzionamento normale prevede la presenza di un solo floppy disk di tipo sistema nell'unità relativa.) Naturalmente, quando l'operazione richiesta è stata completata, nell'unità vi dovrà essere una configurazione valida di dischi, perchè si possa proseguire nel lavoro. Per rendere consapevole l'utente della presenza di una configurazione non valida di dischi, il sistema emette delle segnalazioni quando sono eseguiti i relativi programmi di utilità. Seguendo tali messaggi, l'utente non si troverà mai in situazione non ammessa per il corretto funzionamento del sistema. La descrizione dei programmi di utilità, che segue nelle pagine successive, contiene un elenco di messaggi che possono essere visualizzati dal sistema quando i programmi suddetti sono eseguiti. Di ogni programma è descritta la funzione, il formato del comando che lo carica in memoria utente e ne inizia l'esecuzione, le azioni eseguite dal sistema e gli eventuali messaggi emessi durante l'esecuzione. Per l'interpretazione della sintassi dei comandi che caricano in memoria utente ed eseguono i programmi di utilità si veda il paragrafo "Notazioni" cap. 3. Si osservi che, il nome del programma di utilità, può essere digitato abbreviato nei suoi primi tre caratteri. Quando si digita un comando EXEC, non si deve digitare la virgola prima di premere END OF LINE .

Programma di utilità
ASSEMBLY

Funzione

Traduce un modulo scritto in linguaggio Assembler sorgente in oggetto rilocabile.

Formato

EXE [C] ASS [EMBLY], IN = [SYSLIB / USLIB], filename$_1$, OUT =

{/ / [SYSLIB / USLIB], filename$_2$}, [LIST], [XREF], [WSP = {SYSLIB / USLIB}]

dove:

IN= indica l'operando a parola chiave relativo a informazioni sull'input

SYSLIB/USLIB indica se l'input risiede su floppy disk sistema o utente

filename1 indica il file contenente il sorgente da assemblare, oppure il file contenente la lista di nomi di file che contengono consecutivamente il programma sorgente

OUT= indica l'operando a parola chiave relativo a informazioni sull'output

SYSLIB/USLIB indica se l'output risiede su floppy disk sistema o utente

filename2 indica il file, di tipo oggetto rilocabile, su cui viene trasferito l'output dell'assemblatore

/ indica che l'output dell'Assemblatore non viene registrato su floppy disk

LIST indica la richiesta del listing in output; è un parametro a parola chiave

XREF indica la richiesta della cross reference in output; è un operando a parola chiave

WSP= indica la richiesta del file temporaneo scratch; è un operando a parola chiave. Se non specificato il file viene ugualmente riservato dall'assemblatore sul floppy disk utente se il sistema è bidisco, altrimenti sul floppy disk sistema

SYSLIB/USLIB indica se lo scratch file risiede su floppy disk sistema o utente

Gli operandi sopra descritti sono posizionali.

Azione

Il programma di utilità traduce un modulo scritto in linguaggio Assembler sorgente, residente quale file di tipo testo su floppy disk, in un oggetto rilocabile; l'output del programma di utilità è trasferito su un file di tipo oggetto rilocabile e la sua allocazione avviene in modo dinamico durante l'esecuzione dello Assemblatore.

Il modulo, scritto in linguaggio Assembler sorgente, può risiedere su più file testo; in questo caso il file testo specificato quale input dell'Assemblatore (filename1) contiene solo i riferimenti ai file testo che costituiscono il modulo sorgente.
Esempio:

```
10      COPY    FILE A
20      COPY    FILE B
        ...
        ...
```

dove FILE A e FILE B sono file testo che contengono le istruzioni Assembler sorgenti da assemblatore. Se nell'esecuzione dell'utility si verificano situazioni di errore il sistema segnala il messaggio relativo (vedi appendice D) e passa allo stato comandi. Se l'Assemblatore rileva nel programma sorgente degli errori, emette delle segnalazioni diagnostiche (vedi appendice D).

1. Lo scratch file è allocato, utilizzato e deallocato dall'assemblatore; esso viene allocato con il nome @WORKF e viene comunque cancellato al ter-

mine dell'assemblaggio. Se per qualche caso accidentale (errore fisico, power off) il file rimane presente su floppy disk, è bene operare in uno dei due seguenti modi:

1) eseguire un successivo passo di assemblaggio che cancella automaticamente il file scratch preesistente

2) copiare la libreria, a cui la scratch file appartiene, su di un altro disco; in questa operazione lo scratch file spurio non viene trasferito e si evita, nelle successive operazioni, di trovarsi in presenza di spazio insufficiente su FDU.

Se non esiste spazio sufficiente ad allocare lo scratch file il sistema segnala errore (vedi appendice D) e l'utente deve indicare un altro supporto e creare spazio sul disco in questione; lo spazio minimo richiesto per lo scratch file è di due settori.

2. Se filename2 esiste già, il sistema controlla se è di tipo oggetto rilocabile; in caso affermativo, dato che si presume essere un riassemblaggio dello stesso programma, il file viene rimpiazzato; in caso contrario viene segnalato errore (vedi appendice D).

3. Se filename1 non esiste o non è tipo testo, oppure se le direttive contenute in filename1 fanno riferimento a un file che non esiste o non è di tipo testo o il numero di file riferiti, nelle direttive, è > di 16 il sistema interrompe l'esecuzione dell'utility e segnala l'errore relativo (vedi appendice D).

4. La fase di assemblaggio può essere interrotta in qualunque istante premendo il tasto BREAK .

Esempi

1. Assemblare il file testo PROV, contenente un programma Assembler sorgente e residente su floppy disk utente, e scaricare l'output dell'assemblatore sullo stesso floppy disk, assegnandogni il nome PROVO.

Premere: EXE ASS.IN=USLIB.
PROV,OUT=USLIB,PROVO END OF LINE

2. Assemblare un file testo ART, contenente un programma sorgente Assembler e residente su floppy disk sistema, senza scaricare l'oggetto, output dell'assemblatore, su floppy disk. Viene richiesto il listing in output.

Premere: EXE ASS IN=,ART,
OUT=/,LIST END OF LINE

FDCOPY

Programma di utilità
FDCOPY

Funzione

Copia il contenuto di un floppy disk su di un altro floppy disk.

Formato

EXE [C] FDC [OPY] [,[S/U]]

dove:

S indica il floppy disk sistema

U indica il floppy disk utente

Azione

Il programma di utilità, copia il contenuto del floppy disk specificato con l'operando su di un altro floppy disk.

Se la parte opzionale non è specificata, il programma copia il contenuto del floppy disk sistema su di un altro floppy disk.

Note

1. Il programma FDCOPY può essere usato solamente con un sistema nella configurazione bidisco.

2. Se il disco ricevente contiene sottolibrerie di sistema o applicative (package comune o utente) ancora valide, la loro esistenza è segnalata all'utente mediante la stampa di un relativo messaggio (vedi Messaggi). Sul display compare il messaggio: CONTINUE? Se l'utente decide di effettuare ugualmente la copia, deve premere il tasto console CONTINUE. Altrimenti si prema BREAK.

3. Se si specifica U come operando, viene visualizzato il seguente messaggio:

INSERT DISK (vedi Messaggi).

4. Al termine della esecuzione della copia il sistema ha due dischi utente, o due dischi sistema, viene visualizzato il seguente messaggio:

   END - ILLEGAL STATUS >(vedi Messaggi).

5. Se la copiatura del disco è interrotta per cause accidentali (caduta di tensione, unità floppy disk funzionante in modo anomalo...) il contenuto del disco ricevente non è attendibile, per cui è bene ripetere le operazioni di copia dall'inizio.

6. Se in qualsiasi momento, si richiede la terminazione dell'esecuzione del programma premendo il tasto di console **BREAK**, il sistema richiede di verificare che nell'unità floppy disk vi sia uno ed un solo floppy disk sistema con il messaggio:

   BREAK OCCURRED > (vedi Messaggi)

Messaggi

CONTINUE?

Azione del sistema: L'esecuzione del programma di utilità è interrotta; viene stampato uno o più dei seguenti messaggi:

FILE "P6FWR" VALID
FILE "P6FWO" VALID
FILE "P6SW" VALID
FILE "P6FSYS" VALID
FILE "XXXXXX" VALID

I primi tre messaggi indicano che il disco su cui si vuole copiare è un disco di tipo sistema. Il quarto messaggio indica che su disco ricevente sono state inizializzate le sottolibrerie applicative (package, comune e utente). Se il quarto messaggio appare da solo, indica che il disco suddetto è di tipo utente. Il quinto messaggio indica che il disco ricevente non è compatibile con il sistema P6060.

Risposta dell'utente: Per far continuare la esecuzione del programma di utilità, si deve premere **CONTINUE**

Per terminare l'esecuzione del programma di utilità si deve premere BREAK . In questo caso il sistema è reinizializzato.

Se è stampato il messaggio "P6FSYS" VALID per verificare il contenuto delle sottolibrerie si preme BREAK e dopo che è stato visualizzato il messaggio READY si introduce il comando GATALOG.

BREAK OCCURRED >
CHECK DISK STATUS
(vedi nota)

Azione del sistema: E' terminata l'esecuzione del programma di utilità perchè è stato premuto BREAK .

Risposta dell'utente: Si verifichi che nella unità floppy disk vi sia una configurazione corretta di dischi, per il normale funzionamento del sistema. Se nell'unità non vi è un floppy disk sistema, inserirne uno al posto di un disco utente. Se nell'unità vi sono due floppy disk sistema, toglierne uno e sostituirlo, se necessario, con un floppy disk utente. Quindi premere CONTINUE .

END-ILLEGAL STATUS >
REARRANGE DISKS
(vedi nota)

Azione del sistema: L'esecuzione della copia è stata effettuata correttamente. I dischi presenti nell'unità floppy disk non compongono una configurazione corretta per il normale funzionamento del sistema.

Risposta dell'utente: Se nell'unità floppy disk non vi è alcun floppy disk sistema, inserirne uno. Se nell'unità vi sono due floppy disk sistema, toglierne uno è sostituirlo, se necessario, con un floppy disk utente. Quindi premere CONTINUE .

ERROR n

Azione del sistema: L'esecuzione del programma di utilità è terminata a causa dell'errore il cui codice numerico è n (vedi appendice D). Il sistema commuta nello stato comandi.

Risposta dell'utente: Premere [SHIFT] e [CLEAR RECALL] contemporaneamente per sbloccare la tastiera. Ripetere l'operazione dopo aver rimosso la causa d'errore.

ERROR n-ILLEGAL STATUS >
REARRANGE DISK
(vedi nota)

Azione del sistema: L'esecuzione del programma è terminata a causa dell'errore con codice numerico n (vedi appendice D). La combinazione di dischi attualmente nell'unità floppy disk non è corretta per il funzionamento normale del sistema.

Risposta dell'utente: Si rimuova la causa di errore. Se nell'unità non vi è alcun floppy disk sistema, inserirne uno. Se nell'unità vi sono due floppy disk sistema, toglierne uno e sostituirlo, se necessario, con un floppy disk utente. Quindi premere [CONTINUE]; il sistema passa nello stato comandi. Si ripeta l'operazione.

INSERT DISK
RECEIVING DISK ON DRIVE > { * / * * }
(vedi nota)

Azione del sistema: L'esecuzione del programma di utilità è interrotta. Il sistema richiede di introdurre il disco ricevente nel trascinatore superiore (se specificato *) o nel trascinatore inferiore (se specificato **) della unità floppy disk.

Risposta dell'utente: Inserire il disco ricevente nel trascinatore specificato e premere [CONTINUE].

READY

Azione del sistema: L'esecuzione del programma di utilità è terminata. Il sistema si è reinizializzato con la configurazione di dischi presente nell'unità ed è nello stato comandi.

Risposta dell'utente: Nessuna.

Nota: Quando l'ultimo carattere di un messaggio che appare sul display è >, il messaggio non è completo; per visualizzare la restante parte del messaggio si devono premere [SHIFT] e [→] contemporaneamente.

Esempi

1. Copiare un floppy disk sistema avendo il floppy disk sistema in un trascinatore ed il disco ricevente nell'altro trascinatore.

   Premere [E] [X] [E] [F] [D] [C] [END OF LINE]

   Sul display appare il messaggio: CONTINUE? e viene stampato il messaggio FILE "P6FSYS" VALID. Si prema CONTINUE.

   Il contenuto del floppy disk sistema è copiato sul floppy disk presente nell'altro trascinatore. Al termine della copia sul display compare il messaggio: END-ILLEGAL STATUS>; premendo [SHIFT] con [→] sul display appare la continuazione del messaggio: REARRANGE DISKS. Si tolga uno dei due floppy disk sistema ed, eventualmente, si introduca un floppy disk utente. Si prema CONTINUE. Sul display appare il messaggio: READY ed il sistema è reinizializzato secondo la configurazione attuale di dischi.

2. Copiare il floppy disk utente su di un altro floppy disk, avendo il floppy disk sistema nel trascinatore superiore ed il floppy disk utente da copiare nel trascinatore inferiore.

   Premere [E] [X] [E] [F] [D] [C] [,] [U] [END OF LINE]

   Sul display appare il messaggio: INSERT DISK>; premendo [SHIFT] con [→] sul display appare la conti-

nuazione del messaggio: RECEIVING DISK ON DRIVE *. Si cambi il floppy disk sistema con il floppy disk su cui si vuole copiare e si prema il tasto di console CONTINUE. Sul display è visualizzato il messaggio CONTINUE? e viene stampato il messaggio: FILE "P6FSYS" VALID. Si prema CONTINUE. Al termine della copia sul display appare il messaggio: END-ILLEGAL STATUS >; premendo contemporaneamente i tasti SHIFT e → si può leggere la continuazione del messaggio: REARRANGE DISKS. Si sostituisca uno dei floppy disk utente con un floppy disk sistema e si prema CONTINUE. Sul display appare il messaggio: READY ed il sistema è nello stato comandi.

FLCOPY

Programma di utilità
FLCOPY

Funzione

Copia un file da una sottolibreria in un'altra sullo stesso floppy disk o su di un altro floppy disk.

Formato

EXE [C] FLC [OPY], IN = [SYSLIB / USLIB], filename$_1$, OUT = [SYSLIB / USLIB] [, {filename$_2$ / * / +}]

dove:

IN=SYSLIB indica che il file da copiare è su un floppy disk sistema

IN=USLIB indica che il file da copiare è su un floppy disk utente

filename1 indica il nome del file da copiare

OUT=SYSLIB indica che il file è copiato su un floppy disk sistema

OUT=USLIB indica che il file è copiato su un floppy disk utente

OUT= indica che il file deve essere copiato da una sottolibreria in un'altra sullo stesso floppy disk

* indica una sottolibreria per la quale è stato specificato l'operando * durante l'inizializzazione del floppy disk, effettuata con il programma di utilità LBCREATE, ma che non è stata ancora protetta mediante il programma di utilità LBPROTECT

\+ indica una sottolibreria comune

filename2 indica il nome con cui il file deve essere copiato.

Azione

Il programma di utilità copia il file filename1, dal floppy disk indicato con l'operando IN=, nella sottolibreria specificata con il primo carattere di filename2 (*, + o ø), sul floppy disk specificato con lo operando OUT=, assegnando al file il nome filename2.

Se l'operando OUT= non ha assegnato alcun valore, il programma di utilità ricopia il file specificato con filename1 nello stesso floppy disk in cui è residente (specificato con IN=), assegnando ad esso il nome specificato con filename2.

Se * è specificato come ultimo operando, il programma di utilita copia il file nella sottolibreria package, assegnandogli il nome costituito da * seguito dai caratteri alfanumerici di filename1.

Se + è specificato come ultimo operando, il programma di utilità copia il file nella sottolibreria comune, assegnandogli il nome costituito da + seguito dai caratteri alfanumerici di filename1.

Se l'ultimo operando è omesso, il file è copiato nella sottolibreria utente, del floppy disk specificato con l'operando OUT=, assegnandogli come nome i caratteri alfanumerici di filename1.

Note

1. Nel caso di sistemi inizializzati con la configurazione monodisco, FLCOPY può essere usata solo per copiare un file sullo stesso floppy disk.

2. Il programma di utilità FLCOPY non può copiare un file in una sottolibreria package che sia stata protetta mediante il programma LBPROTECT.

3. Se il file da copiare è stato protetto con il comando SECURE, il file copiato mantiene la stessa protezione.

4. Se il sistema è nella configurazione bidisco e si vuole copiare un file da un floppy disk su di un altro, non presente nell'unità floppy disk, il sistema visualizza il seguente messaggio: INSERT DISK > (vedi Messaggi).

5. Se con il sistema nella configurazione bidisco, al termine della esecuzione della copia, sono presen-

ti nel sistema due floppy disk utente, viene visualizzato il seguente messaggio: END ILLEGAL STATUS > (vedi Messaggi).

Messaggi

BREAK OCCURRED >
CHECK DISK STATUS

Azione del sistema: L'esecuzione del programma di utilità è terminata, perchè è stato premuto BREAK .

Risposta dell'utente: Si verifichi che nella unità floppy disk vi sia una configurazione corretta di dischi, per il normale funzionamento del sistema. Se nell'unità non vi è un floppy disk sistema, inserirne uno al posto di un disco utente. Se nell'unità vi sono due floppy disk sistema, toglierne uno e sostituirlo con un floppy disk utente. Quindi premere CONTINUE .

END-ILLEGAL STATUS >
REARRANGE DISKS
(vedi nota)

Azione del sistema: L'esecuzione della copia è stata effettuata correttamente. I dischi presenti nella unità floppy disk non compongono una configurazione corretta per il normale funzionamento del sistema.

Risposta dell'utente: Se nell'unità floppy disk non vi è alcun floppy disk sistema, inserirne uno. Se nell'unità vi sono due floppy disk sistema, toglierne uno e sostituirlo, se necessario, con un floppy disk utente. Quindi premere CONTINUE .

ERROR n

Azione del sistema: L'esecuzione del programma utilità è terminata a causa dell'errore il cui codice numerico è n (vedi appendice D). Il sistema commu-

ta nello stato comandi.

Risposta dell'utente: Premere (SHIFT) e (CLEAR RECALL) per sbloccare la tastiera. Ripetere l'operazione dopo aver rimosso la causa d'errore.

ERROR n -ILLEGAL STATUS >
REARRANGE DISKS
(vedi nota)

Azione del sistema: L'esecuzione del programma è interrotta perchè è stato rilevato l'errore con codice numerico n (vedi appendice D). Nell'unità floppy disk non c'è un disco sistema o vi sono due dischi sistema.

Risposta dell'utente: Introdurre il disco sistema, se manca, ponendolo eventualmente al posto di un disco utente; oppure togliere un disco sistema se nell'unità ve ne sono due. Quindi premere CONTINUE. Il sistema è nello stato comandi.

INSERT DISK

RECEIVING {SYSDIS / USDIS} ON DRIVE { * / * * }

Azione del sistema: L'esecuzione del programma è interrotta. Il sistema richiede di introdurre un disco sistema (se specificato SYSDIS) o un disco utente (se specificato USDIS) nel trascinatore superiore (se specificato *) o nel trascinatore inferiore (se specificato **), della unità floppy disk.

Risposta dell'utente: Inserire il disco ricevente nel trascinatore specificato e premere CONTINUE.

READY

Azione del sistema: L'esecuzione del programma di utilità è terminata. Il sistema si è reinizializzato con la configurazione di dischi presente nell'unità

ed è nello stato comandi.

Risposta dell'utente: Nessuna.

Nota: Quando l'ultimo carattere di un messaggio che appare nel display è >, il messaggio non è completo; per visualizzare la restante parte del messaggio si devono premere [SHIFT] e [→] contemporaneamente.

Esempi

1. Copiare il file dati SOK dalla sottolibreria comune di un floppy disk sistema nella sottolibreria utente del medesimo floppy disk, assegnandogli il nome SOK.

   Premere [E][X][E] [F][L][C][,][I][N][=][,][+][S][O][K][,] [O][U][T][=][END OF LINE]
   Al termine della copia il sistema visualizza il messaggio: READY. Il sistema è nuovamente disponibile, nello stato comandi.

2. Copiare il file PIF della sottolibreria utente di un floppy disk utente in un altro floppy disk utente, assegnandogli il nome DAS (il file deve essere registrato in un'altra sottolibreria utente). Si introduca il floppy disk sistema nel trascinatore superiore. Si introduca nel trascinatore inferiore il floppy disk utente contenente il file da copiare.

   Premere [E][X][E] [F][L][C][,][I][N][=][U][S][L][I][B][,] [P][I][F][,][O][U][T][=][U][S][L][I][B][,][D][A][S][END OF LINE]

   Sul display appare il messaggio: INSERT DISK > ; premendo contemporaneamente [SHIFT] e [→] , sul display appare la seconda parte del messaggio: RECEIVING USDIS ON DRIVE *. Si sostituisca, nel trascinatore superiore, il floppy disk sistema con il floppy disk utente su cui si vuol copiare il file del disco presente nel trascinatore inferiore.

   Premere [CONTINUE] .

   La copia del file suddetto è effettuata. Sul display appare il messaggio: END - ILLEGAL STATUS >; premendo contemporaneamente [SHIFT] e [→] , sul display

appare la restante parte del messaggio: REARRANGE DISKS. Si introduca il floppy disk sistema nel trascinatore superiore o inferiore al posto del disco attualmente presente. (Se per le operazioni successive è sufficiente il floppy disk sistema, si può estrarre il floppy disk utente.)

Si prema CONTINUE .

Sul display appare il comando READY. Il sistema è nello stato comandi, inizializzato con la configurazione di dischi attualmente presente nell'unità.

LBCREATE

Programma di utilità
LBCREATE

Funzione

Inizializza la libreria software applicativo.

Formato

EXE [C] LBC [REATE] [, [S/U]] [, * = n1] [, + = n2] [, NP = n3]

dove:

S indica il floppy disk sistema

U indica il floppy disk utente

*=n_1 specifica un numero intero positivo o nullo usato dal sistema per allocare il massimo numero di nomi di file, che possono essere registrati nella sottolibreria package

+=n_2 specifica un numero intero positivo o nullo usato dal sistema per allocare il massimo numero di nomi di file, che possono essere registrati nella sottolibreria comune

NP=n_3 specifica un numero intero positivo o nullo usato dal sistema per allocare il massimo numero di nomi di file, che possono essere registrati nella sottolibreria utente

$n_1 + n_2 + n_3$ deve essere minore di 15.

Azione

Il programma di utilità inizializza la libreria software applicativo del floppy disk specificato con il primo operando, assegnando alle sottolibrerie package, comune ed utente, il numero di nomi di file definibili dall'utente ottenuto moltiplicando per 13 rispettivamente n_1, n_2, n_3; a tal fine il sistema alloca sul disco rispettivamente n_1, n_2, ed n_3 settori (128 byte).

Se nessuno degli operandi *, + ed NP è specificato nel comando, il programma di utilità LBCREATE assegna alle sottolibrerie package, comune ed utente rispettivamente 65, 52 e 65 nomi di file definibili dall'utente. Altrimenti le sottolibrerie per le quali non si è specificato il relativo operando, assumono un numero di nomi di file uguale a zero.

Note

1. Se il sistema è nella configurazione con 8K byte di memoria utente, per poter utilizzare il programma di utilità LBCREATE, si deve prima reinizializzare il sistema con il comando OPT, per essere sicuri che vi sono in memoria utente le routine che trattano le opzioni.

2. Gli operandi *, + ed NP non sono posizionali, quindi si possono specificare in qualsiasi ordine.

Messaggi sul display

ACTION ON UNIT FDU * ? oppure
ACTION ON UNIT FDU ** ?
Il floppy disk da inizializzare contiene già delle informazioni. Premendo CONTINUE l'esecuzione del programma di utilità prosegue, mentre premendo BREAK l'esecuzione termina ed il sistema commuta nello stato comandi.

READY: l'esecuzione del programma di utilità è terminata; il sistema è nello stato comandi.

Esempi

1. Si inizializzi un floppy disk utente assegnando 52 file alla sottolibreria comune, 130 file alla sottolibreria utente e nessun file alla sottolibreria package.

   Premere E X E L B C , U , C + M = 4 , N P = 1 0 END OF LINE

2. Si inizializzi un floppy disk sistema assegnando 65 file alla sottolibreria package, 52 file alla sottolibreria comune e 65 file alla sottolibreria utente.

   Premere E X E L B C END OF LINE

LBPROTECT

Programma di utilità
LBPROTECT

Funzione

Protegge le sottolibrerie specificate dall'azione di alcuni comandi di sistema.

Formato

EXE [C] LBP [ROTECT] [, [**SYSLIB** / **USLIB**] [, { * / + }]]

dove:

SYSLIB indica il floppy disk sistema

USLIB indica il floppy disk utente

* indica la sottolibreria package

\+ indica la sottolibreria comune.

Azione

Il programma di utilità protegge dall'azione di alcuni comandi di sistema la sottolibreria specificata dall'ultimo operando, residente sul floppy disk specificato con il primo operando.

Se l'ultimo operando non è specificato, vengono protette sia la sottolibreria package che quella comune.

La sottolibreria package è protetta contro l'azione dei comandi:

CREATE (non è possibile creare altri file)

MODIFY (non è possibile modificare il nome dei file)

PURGE (non è possibile cancellare dei file)

SAVE (non è possibile registrare altri file testo)

e dei programmi di utilità

FLCOPY
LIBCOPY.

La sottolibreria comune è protetta contro l'azione dei comandi:

MODIFY (non è possibile modificare il nome dei file)
PURGE (non è possibile cancellare dei file).

Nota

Quando una sottolibreria è protetta non si può rimuovere la protezione.

Messaggi su display

READY: l'esecuzione del programma di utilità è terminata; il sistema è nello stato comandi.

LIBCOPY

Programma di utilità
LIBCOPY

Funzione

Copia su di un altro disco tutti i file della sottolibreria specificata.

Formato

EXE [C] LIB [COPY], IN = {SYSLIB / USLIB} , [* / + / :] , OUT = {SYSLIB / USLIB}

dove:

IN=SYSLIB indica che la sottolibreria da copiare è sul floppy disk sistema

IN=USLIB indica che la sottolibreria da copiare è sul floppy disk utente

* indica che i file di una sottolibreria package devono essere copiati in un'altra sottolibreria package

\+ indica che i file di una sottolibreria comune devono essere copiati in un'altra sottolibreria comune

: indica che devono essere copiate tutte le sottolibrerie applicative (package, comune e utente) presenti sul floppy disk specificato con l'operando IN=

OUT=SYSLIB indica che la copia deve essere fatta su un floppy disk sistema

OUT=USLIB indica che la copia deve essere fatta su un floppy disk utente.

Azione

Il programma di utilità copia tutti i file della sottolibreria o delle sottolibrerie indicate con il secondo operando, dal floppy disk specificato con l'o-

perando IN=, nello stesso tipo di sottolibreria, sul floppy disk indicato con l'operando OUT=. Nella sottolibreria ricevente i file sono aggiunti in coda a quelli preesistenti in un'unica estensione.

Se l'operando che indica la sottolibreria da copiare è omesso vengono copiati i file della sottolibreria utente, dal floppy disk specificato con l'operando IN=, nella sottolibreria utente, sul floppy disk specificato con l'operando OUT=.

Note

1. Se la sottolibreria package di un disco ricevente è stata protetta, eseguendo il programma di utilità LBPROTECT, in essa non si possono copiare altri file mediante il programma di utilità LIBCOPY.

2. Il programma di utilità è eseguibile solo su di un sistema inizializzato nella configurazione bidisco.

3. Si noti che, naturalmente, il floppy disk ricevente deve essere inizializzato dichiarando per la sottolibreria ricevente, al momento in cui è inizializzata mediante il programma di utilità LBCREATE, un numero di file uguale o maggiore del numero di file che si avranno nella stessa sottolibreria al termine della copia.

4. Se si vuol copiare una sottolibreria di un floppy disk sistema su di un altro floppy disk sistema, viene visualizzato il seguente messaggio: INSERT DISK> (vedi Messaggi).

5. Se si vuol copiare una sottolibreria di un floppy disk utente su di un altro floppy disk utente, viene visualizzato il seguente messaggio: INSERT DISK> (vedi Messaggi).

6. Se l'operazione di copiatura della sottolibreria specificata viene interrotta (perchè viene a mancare la tensione, perchè l'unità floppy disk è malfunzionante, etc.), il contenuto del disco ricevente non è attendibile.

BREAK OCCURRED >
CHECK DISK STATUS
(vedi nota)

Azione del sistema: E' terminata l'esecuzione del programma di utilità perchè è stato premuto BREAK.

Risposta dell'utente: Si verifichi che nella unità floppy disk vi sia una configurazione corretta di dischi, per il normale funzionamento del sistema. Se nell'unità non vi è un flopy disk sistema, inserirne uno al posto di un disco utente. Se nell'unità vi sono due floppy disk sistema, toglierne uno e sostituirlo, se necessario, con un floppy disk utente. Quindi premere CONTINUE.

END-ILLEGAL STATUS >
REARRANGE DISKS
(vedi nota)

Azione del sistema: L'esecuzione della copia è stata effettuata correttamente. I dischi presenti nell'unità floppy disk non compongono una configurazione corretta per il normale funzionamento del sistema.

Risposta dell'utente: Se nell'unità floppy disk non vi è alcun floppy disk sistema, inserirne uno. Se nell'unità vi sono due floppy disk sistema, toglierne uno e sostituirlo, se necessario, con un floppy disk utente. Quindi premere CONTINUE.

Error n

Azione del sistema: L'esecuzione del programma di utilità è terminata a causa dell'errore il cui codice numerico è n (vedi appendice D). Il sistema commuta nello stato comandi.

Risposta dell'utente: Premere SHIFT e CLEAR RECALL per sbloccare la tastiera. Ripetere l'operazione dopo aver rimosso la causa d'errore.

ERROR n-ILLEGAL STATUS >
REARRANGE DISKS
(vedi nota)

Azione del sistema: L'esecuzione del programma è terminata a causa dell'errore il cui codice numerico è n (vedi appendice D). Nell'unità floppy disk non c'è un disco sistema o vi sono due dischi sistema.

Risposta dell'utente: Si rimuova la causa di errore. Se nell'unità non vi è alcun floppy disk sistema, inserirne uno. Se nell'unità vi sono due floppy disk sistema, toglierne uno e sostituirlo, se necessario, con un floppy disk utente. Quindi premere CONTINUE . Il sistema passa nello stato comandi. Si ripeta l'operazione.

INSERT DISK { SYSDIS / USDIS } >
RECEIVING { SYSDIS / USDIS } ON DRIVE { * / ** }

Azione del sistema: L'esecuzione del programma è interrotta. Il sistema richiede di introdurre un disco sistema (se specificato SYSDIS) od un disco utente (se specificato USDIS), nel trascinatore superiore (se specificato *) o nel trascinatore inferiore (se specificato **), dell'unità floppy disk.

Risposta dell'utente: Inserire il disco ricevente nel trascinatore specificato e premere CONTINUE .

READY

Azione del sistema: L'esecuzione del programma di utilità è terminata. Il sistema si è reinizializzato con la configurazione di dischi presente nell'unità ed è nello stato comandi.

Risposta dell'utente: Nessuna.

Nota: Quando l'ultimo carattere di un messaggio che appare sul display è >, il messaggio non è completo; per visualizzare la restante parte del messaggio si devono premere [SHIFT] e [→] contemporaneamente.

Esempi

1. Copiare la sottolibreria package da un floppy disk utente nel floppy disk sistema.

   Premere [E][X][E] [L][I][B],[I][N]=[U][S][L][I][B],·
   ,[O][U][T]=[S][Y][S][L][I][B] [END OF LINE]
   Al termine della copia il sistema visualizza il messaggio: READY. Il sistema è nello stato comandi.

2. Copiare la sottolibreria utente dal floppy disk utente, montato nel trascinatore inferiore, in un altro floppy disk utente. Il floppy disk sistema è montato nel trascinatore superiore.

   Premere [E][X][E] [L][I][B],[I][N]=[U][S][L][I][B],
   ,[O][U][T]=[U][S][L][I][B] [END OF LINE]
   Sul display appare il messaggio: INSERT DISK >
   Premendo contemporaneamente [SHIFT] con [→], sul display appare la restante parte del messaggio: RECEIVING USDIS ON DRIVE *.

   Si introduca il floppy disk utente, nel quale si vuole copiare la sottolibreria suddetta, nel trascinatore superiore. Premere [CONTINUE]. La sottolibreria è copiata ed al termine viene visualizzato il messaggio: END-ILLEGAL STATUS >.

   Premendo [SHIFT] e [→] contemporaneamente, viene visualizzata la restante parte del messaggio: REARRANGE DISKS >.

   Inserire il floppy disk sistema al posto di uno dei floppy disk utente. Premere [CONTINUE]. Sul display appare il messaggio: READY. Il sistema è nello stato comandi.

Programma di utilità
LNK

Funzione

Trasforma un oggetto rilocabile in un oggetto eseguibile.

Formato

EXE [C] LNK , IN = [SYSLIB / USLIB] , filename 1 , OUT = [SYSLIB / USLIB] , filename 2

dove:

IN= indica l'operando a parola chiave relativo allo input

SYSLIB/USLIB indica se l'input risiede su floppy disk sistema o utente

filename1 indica il file di tipo oggetto rilocabile, input al programma di utilità, da cui si ottiene il file di tipo oggetto eseguibile

OUT= indica l'operando a parola chiave relativo all'output

SYSLIB/USLIB indica se l'output risiede su floppy disk sistema o utente

filename2 indica il file di tipo oggetto eseguibile, output del programma di utilità; è allocato dinamicamente

Azione

Il programma di utilità trasforma un modulo output dell'assemblatore, residente quale file di tipo oggetto rilocabile su floppy disk, in un oggetto eseguibile; l'output del programma di utilità è trasferito su un file di tipo oggetto eseguibile e la sua allocazione avviene in modo dinamico durante l'esecuzione del Linker. Se nell'esecuzione dell'utility si verificano situazioni di errore il sistema segna-

la il messaggio relativo (vedi appendice D) e passa allo stato comandi.

Note

1. I richiami ai moduli di PIOCS di sistema (vedi capitolo 6) sono sviluppati in modo che siano riconosciuti a Run time e siano risolti in modo automatico.

2. Se filename2 esiste già, il sistema controlla se è di tipo oggetto eseguibile; in caso affermativo il file viene rimpiazzato (si presume che il file già esistente sia l'output di un precedente passo di Linker); in caso contrario viene segnalato errore (vedi appendice D).

3. Se filename1 non esiste o è di tipo 0 (oggetto rilocabile il sistema interrompe l'esecuzione dell'utility e segnala l'errore relativo (vedi appendice D).

Esempio

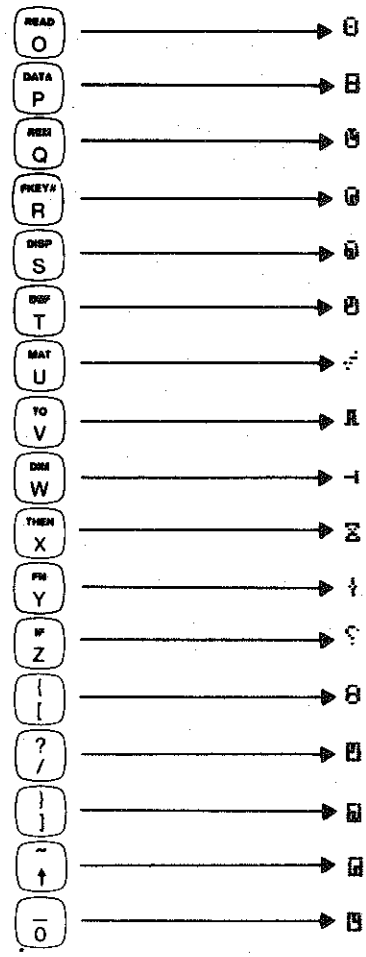
Linkare il file oggetto PROVO, contenente un programma in formato oggetto rilocabile e residente su floppy disk utente, e scaricare l'output del linker sullo stesso floppy disk, assegnandogli il nome PROVE.

Premere EXEC LNK,IN=USLIB,
PROVO,OUT=USLIB,PROVE END OF LINE

B. CARATTERI SPECIALI DISPONIBILI DA TASTIERA

Utilizzando il tasto CONTROL è possibile ottenere i caratteri documentati nella seguente tabella.

Tasto premuto insieme con il tasto CONTROL	Carattere grafico visualizzato e/o stampato	Codice ISO corrispondente
@		NUL
PRINT A		TC1
STEP B		TC2
FOR C		TC3
USING D		TC4
DCL E		TC5
WRITE F		TC6
ON G		BEL
GOTO H		FE0
INPUT I		FE1
GOSUB J		FE2
RETURN K		FE3
STOP L		FE4
END M		FE5
NEXT N		SO



SI
DLE
DC1
DC2
DC3
DC4
NAK
SYN
ETB
CAN
EM
SUB
ESC
IS4
IS3
IS2
IS1

C. SET DI CARATTERI DEL SISTEMA P6060

Nella tabella seguente sono elencati tutti i caratteri che possono essere stampati o visualizzati sul sistema P6060. Ogni carattere è evidenziato in relazione al corrispondente valore decimale, valore binario e codice ISO.

0	00000000	NUL	
1	00000001	SOH	
2	00000010	STX	
3	00000011	ETX	
4	00000100	EOT	
5	00000101	ENQ	
6	00000110	ACK	
7	00000111	BEL	
8	00001000	BS	
9	00001001	HT	
10	00001010	LF	
11	00001011	VT	
12	00001100	FF	
13	00001101	CR	
14	00001110	SO	
15	00001111	SI	
16	00010000	DLE	
17	00010001	DC1	
18	00010010	DC2	
19	00010011	DC3	
20	00010100	DC4	
21	00010101	NAK	
22	00010110	SYN	
23	00010111	ETB	
24	00011000	CAN	
25	00011001	EM	
26	00011010	SUB	
27	00011011	ESC	
28	00011100	FS	
29	00011101	GS	
30	00011110	RS	
31	00011111	US	
32	00100000	Space	
33	00100001	!	!
34	00100010	"	"
35	00100011	#	#

36	00100100	$	$
37	00100101	%	%
38	00100110	&	&
39	00100111	'	'
40	00101000	(	(
41	00101001	)	)
42	00101010	*	*
43	00101011	+	+
44	00101100	,	,
45	00101101	-	-
46	00101110	.	.
47	00101111	/	/
48	00110000	0	0
49	00110001	1	1
50	00110010	2	2
51	00110011	3	3
52	00110100	4	4
53	00110101	5	5
54	00110110	6	6
55	00110111	7	7
56	00111000	8	8
57	00111001	9	9
58	00111010	:	:
59	00111011	;	;
60	00111100	<	<
61	00111101	=	=
62	00111110	>	>
63	00111111	?	?
64	01000000	@	@
65	01000001	A	A
66	01000010	B	B
67	01000011	C	C
68	01000100	D	D
69	01000101	E	E
70	01000110	F	F
71	01000111	G	G
72	01001000	H	H
73	01001001	I	I
74	01001010	J	J
75	01001011	K	K
76	01001100	L	L
77	01001101	M	M
78	01001110	N	N
79	01001111	O	O
80	01010000	P	P
81	01010001	Q	Q
82	01010010	R	R
83	01010011	S	S
84	01010100	T	T
85	01010101	U	U
86	01010110	V	V
87	01010111	W	W
88	01011000	X	X
89	01011001	Y	Y
90	01011010	Z	Z

91	01011011	[	[
92	01011100	\	\
93	01011101	]	]
94	01011110	↑	↑
95	01011111	←	←
96	01100000	`	`
97	01100001	a	a
98	01100010	b	b
99	01100011	c	c
100	01100100	d	d
101	01100101	e	e
102	01100110	f	f
103	01100111	g	g
104	01101000	h	h
105	01101001	i	i
106	01101010	j	j
107	01101011	k	k
108	01101100	l	l
109	01101101	m	m
110	01101110	n	n
111	01101111	o	o
112	01110000	p	p
113	01110001	q	q
114	01110010	r	r
115	01110011	s	s
116	01110100	t	t
117	01110101	u	u
118	01110110	v	v
119	01110111	w	w
120	01111000	x	x
121	01111001	y	y
122	01111010	z	z
123	01111011	{	{
124	01111100	|	|
125	01111101	}	}
126	01111110	~	~
127	01111111	DEL	▒
128	10000000		▒
129	10000001		▒
...			
...			
...			
255	11111111		▒

Tabella C-1 Set di caratteri del sistema P6060

D. MESSAGGI DEL SISTEMA P6060

Il sistema P6060 produce i seguenti tre tipi di messaggi che ne facilitano l'impiego e che permettono una rapida identificazione degli errori di programmazione:

1. messaggi di avvertimento

2. messaggi informativi

3. messaggi di errore

Ognuno dei tre tipi di messaggio verrà descritto brevemente e si potrà così osservare che i messaggi di avvertimento e i messaggi informativi, sono comprensibili senza ulteriore spiegazione, mentre verrà fornito un elenco completo dei messaggi di errore.

Messaggi di avvertimento

I messaggi di avvertimento sono quei messaggi che avvertono l'operatore che è stato fornito un dato non corretto; per esempio, se si introduce nello stato debugging, un indirizzo che sconfina la memoria, il sistema avverte l'operatore con il messaggio:

MEMORY ADDRESSES EXCEEDED

così se un comando è digitato con sintassi scorretta, in stato debugging, sul display appare il messaggio:

INCORRECT FORMAT

ed attende che venga introdotto il comando corretto.

Messaggi informativi

I messaggi informativi forniscono informazioni sullo stato del sistema come, ad esempio:

READY

che indica che il sistema è pronto ad accettare un comando.

Messaggi di errore

Questi messaggi identificano eventuali errori che si verificano durante l'introduzione o l'esecuzione dei comandi di sistema, dei programmi di utilità o delle istruzioni Assembler. I tipi di errori identificati dai messaggi suddetti si possono classificare in due categorie: errori di sintassi, errori di esecuzione.

1. Errori di sintassi: si riferiscono alla struttura di un comando (ad esempio, un operando non coerente con il tipo di comando richiesto).

2. Errori di esecuzione: sono errori rilevati durante l'esecuzione di un programma.

Il sistema rileva gli errori di sintassi quando viene introdotto un comando e permette all'operatore, dopo la pressione del tasto RECALL, di effettuare le opportune correzioni. Il sistema rileva gli errori di esecuzione dopo che è stato introdotto un comando EXQ.

Gli errori di esecuzione possono essere corretti durante la fase di esecuzione di un programma. Quando si verifica un errore, viene interrotta l'esecuzione del programma ed il sistema si commuta nello stato di debugging se nella memoria sono presenti le routine di debugging (comando OPT DEB). Nello stato di debugging si può modificare opportunamente il codice oggetto e riprendere l'esecuzione del programma (vedi capitolo 8). Se le routine di debugging non sono presenti in memoria l'unico modo per sbloccare il sistema dalla situazione di errore è di premere il tasto BREAK: il sistema ritorna nello stato comandi e quindi si possono effettuare le necessarie correzioni sul programma sorgente.

I messagi di errore sono normalmente identificati da un codice. I codici da 1 a 22 si riferiscono a errori che si possono verificare durante l'esecuzione di un programma. I codici che iniziano con "AØ" si riferiscono a segnalazioni diagnostiche rilevate dall'assemblatore sul sorgente Assembler. I codici da 114 a 117 sono relativi ad errori che possono verificarsi quando il sistema è nello stato calcoli immediati o debugging. I codici da 151 a 156 si riferiscono ad errori che si possono verificare durante una operazione di accesso ad un floppy disk. I codici da 172 a 212 si riferiscono ad errori che si possono veri-

ficare durante l'introduzione o l'esecuzione di un comando di sistema. I codici da 232 a 235 e il 142 si riferiscono ad errori che si possono verificare durante l'introduzione dei comandi che richiamano un programma di utilità o durante l'esecuzione del programma di utilità stesso. Se viene segnalato un errore durante l'esecuzione di un comando di sistema o di un programma di utilità la tastiera può essere inibita; in questo caso si prema [SHIFT] con [CLEAR RECALL] oppure [CLEAR RECALL] e la tastiera viene disabilitata. Si è anche specificata una lista di errori che vengono segnalati quando il sistema funziona in condizioni anormali. Quando questi ultimi sono segnalati si prema il tasto [CONTINUE] e si ripetano le operazioni precedenti dopo che sul display è apparso il messaggio: READY. Se la segnalazione di errore persiste dopo diversi tentativi ci si rivolga al più vicino servizio Olivetti evidenziando il messaggio visualizzato dal sistema.

Messaggi di errore
Gruppo A

Questo paragrafo descrive i messaggi di errore bloccanti che si possono verificare durante l'esecuzione dell'utility ASSEMBLY; vengono segnalati dall'utility.

Errore	Descrizione
INSUFFICIENT SPACE ON UNIT {SYSLIB / USLIB} FOR WORK FILE	Gli extent liberi di disco assegnati allo scratch file non sono sufficienti a contenere la stringa intermedia e l'eventuale reference table: l'utente deve creare più spazio sul disco indicato.
LOCATION COUNTER OVERFLOW	Il location counter assegnato ad ognuna delle sezioni di controllo del programma sorgente supera $2^{31}-1$: l'utente deve modificare in modo opportuno il proprio programma sorgente.
TOO MANY DEFINED NAMES	L'utente ha definito nomi in eccesso rispetto al numero consentito dalle dimensioni della memoria: l'utente deve o diminuire il numero dei nomi nel proprio programma sorgente o passare ad una macchina di capacità superiore.
TOO MANY DSECT	Il numero di dummy section definite supera 127:

	l'utente deve modificare opportunamente il proprio programma sorgente.
TOO MANY EXTERNAL NAMES	L'utente ha definito troppi nomi esterni, nel programma sorgente, rispetto alla capacità assegnata al dizionario: l'utente deve o diminuire il numero dei nomi esterni nel proprio programma sorgente o passare ad una macchina di capacità superiore.
TOO MANY LITERALS	L'utente ha definito literal tali che il codice oggetto generato, a fronte dei literal, supera la dimensione alla tavola dei literal: l'utente deve o diminuire il numero dei literal nel proprio programma sorgente o passare ad una macchina di capacità superiore.

Messaggi di errore
Gruppo B

Questo paragrafo descrive i messaggi di errore non bloccanti che si possono verificare durante l'esecuzione dell'utility ASSEMBLY; vengono segnalati dall'utility.

Errore	Descrizione
INSUFFICIENT SPACE ON UNIT {SYSLIB / USLIB} FOR OBJECT FILE	Sul supporto non ci sono extent sufficienti per allocare il codice oggetto: l'utente deve creare più spazio sul disco o fare scelte più opportune, a livello argomenti di lancio, nell'indicare l'unità di residenza del flusso di lavoro e del codice oggetto.
REF OVERFLOW	Overflow nella cross reference. Tabella stampata in modo incompleto.
OBJECT LENGTH Ø - FILE OBJECT NOT ALLOCATED	Codice oggetto non allocato perchè, a fronte del programma sorgente, non è stato generato alcun codice oggetto.
SOURCE LITERAL TABLE OVERFLOW	Tabella literal stampata in modo incompleto.

Messaggi di errore
Gruppo C

Questo paragrafo descrive le condizioni di errore bloccante che si possono verificare durante l'esecuzione di un programma ordinata dal comando EXQ.

Codice di errore	Descrizione
75	Opzione debuggirg assente.
150	Area di disco insufficiente per rendere il programma rieseguibile.

A fronte dei seguenti errori il sistema può entrare in stato debugging.

Codice di errore	Descrizione
1	Overflow di stack.
2	Execute di una execute.
4	Indirizzamento fuori memoria.
5	Formato istruzione errato degli operandi. Istruzione RLSEM di segmento non attivo.
8	Errore imputabile al hardware.
9	Underflow di stack.
11	Opzione inesistente.
12	Il floppy disk di sistema è danneggiato.
13	Errore di allineamento.
14	Codice operativo inesistente.

Messaggi di errore
Gruppo D

Questo paragrafo descrive le condizioni di errore bloccante che si possono verificare durante l'esecuzione di un programma ordinata dal comando EXQ; a fronte di questi errori il sistema manda in terminazione il programma e rientra nello stato comandi.

Codice di errore	Descrizione
21	CALEXS con numero di argomenti errato.
22	CALEXT o ACT ad un modulo non esistente.

Messaggi di errore
Gruppo E

Questo paragrafo descrive l'elenco delle segnalazioni diagnostiche emesse dall'Assemblatore, in fase di assemblaggio di un programma sorgente, a fronte di errori nel sorgente. L'apposizione del segno (*), in coda al testo dell'errore, indica che il messaggio è accompagnato dall'indicazione della colonna, rispetto a inizio operandi, presso la quale si è rilevato l'errore. L'indicazione della colonna,in certi casi, per ragioni inerenti alla logica del sistema, perde di significato.

Errore	Descrizione
AØØ1 ILLEGAL NAME FIELD	Spiegazione Nome assente in frasi in cui è obbligatorio indicare il nome frase DSECT e frase EQU , oppure nome presente in frasi in cui il nome non è ammesso frasi EXT/USING/DROP/END/ORG/frasi di controllo listing . Severity code Ø8 Sviluppo La frase in cui è rilevato questo errore è comunque analizzata e tradotta; la frase DSECT crea comunque una sezione fittizia; la frase EQU senza nome è praticamente inoperante; i nomi non ammessi non sono posti nella tavola dei nomi.
AØØ2 INVALID NAME	Spiegazione Il campo nome contiene un nome scorretto; contiene più di 8 crt, oppure non inizia con un carattere alfabetico, oppure contiene dei caratteri speciali.

	Severity code Ø8 Sviluppo Il nome scorretto non viene posto nella tavola dei nomi; la frase in cui è rilevato questo errore è comunque analizzata e tradotta.
AØØ3 INVALID OR MISSING OPERATION CODE	Spiegazione Nel campo operazione della frase non è stato indicato un codice operativo, oppure il codice operativo non è correttamente isolabile, perchè costituito più di 6 caratteri. Severity code 12 Sviluppo La frase non viene presa in considerazione dall'Assembler; l'eventuale nome presente non è posto nella tavola dei nomi.
AØØ4 UNDEFINED OPERATION CODE	Spiegazione La stringa di caratteri indicata nel campo operazione della frase sorgente non corrisponde al codice mnemonico di una istruzione macchina, al codice di una istruzione direttrice o di una macro-istruzione. Severity code 12 Sviluppo La frase non viene presa in considerazione dall'Assembler; l'eventuale nome non è posto nella tavola dei nomi.

AØØ5 ERROR IN LOGICAL SEQUENCE	Spiegazione Frase fuori sequenza logica: frase PROC non segue immediatamente la frase START/CSECT di inizio Control Section; frasi, diverse dalle frasi commento, o comunque trattate come commento, diverse dalle frasi di controllo listing e dalle frasi EQU, situate prima dell'inizio della sezione di programma. Severity code 12 Sviluppo La frase fuori sequenza non è analizzata e tradotta dall'Assembler.
AØØ6 MISSING OPERAND FIELD	Spiegazione Campo operandi assente in frase in cui è obbligatorio. Severity code Ø8 Sviluppo Se si tratta di istruzione macchina, viene sviluppato l'oggetto a zero binario; l'eventuale nome viene posto nella tavola dei nomi. Se si tratta di frase direttiva all'Assembler, essa non viene eseguita; in particolare nel caso di frasi di definizione dati con campo operandi assente non viene sviluppato alcun oggetto, e all'eventuale nome viene attribuita lunghezza 1.
AØØ7 ILLEGAL CONTINUATION	Spiegazione Errore di continuazione: più di due linee di continuazione; i primi 15 caratteri delle linee di continuazione non a Ø; il campo operandi sulle linee di continuazione non inizia alla colonna 1 di continuazione del campo commento; l'ultima linea sorgente con

	l'indicativo di continuazione. Severity code Ø8 Sviluppo L'assembler isola il campo operandi fino a che è stato rilevato l'errore di continuazione. Si precisa che l'assembler considera facenti parte di una stessa frase sorgente tutte le linee di continuazione indicate nel campo continuazione, anche oltre le due linee permesse in base alle specifiche del linguaggio: il campo operandi si considera comunque concluso nelle prime due linee di continuazione.
AØØ8 ILLEGAL CHARACTER OR INVALID DELIMITER (*)	Spiegazione Carattere illegale alla posizione indicata: l'Assembler si aspetta un termine, un operatore o uno dei due delimitatori che separano i sottocampi degli operandi e gli operandi stessi (virgola, parentesi aperta o chiusa, apice di apertura della stringa caratteri nelle pseudo di definizione dati) e non lo trova. Severity code Ø8 Sviluppo L'analisi e traduzione della frase si interrompe. Nel caso di istruzione macchina, l'oggetto viene generato a Ø binario; nel caso di pseudo di definizione dati viene generata la corrispondente stringa oggetto o a Ø binario o viene allocata la corrispondente area solo se è stato possibile valutarne la lunghezza.
AØØ9 INCORRECT SPECIFICATION OF REGISTER OR MASK (*)	Spiegazione L'espressione usata per indicare un registro o una maschera non ha valore assoluto, o comunque un valo-

	re assoluto compreso tra Ø e 15. Severity code Ø8 Sviluppo L'Assembler prosegue l'analisi sintattica e semantica della frase; la stringa oggetto corrispondente è comunque posta a Ø binario.
AØ1Ø SYNTAX ERROR IN OPERAND FIELD (*)	Spiegazione La sintassi della frase prevede la fine del campo operandi e invece quest'ultimo continua scorrettamente. Severity code Ø8 Sviluppo L'Assembler interrompe l'analisi della frase, la cui corrisponderte eventuale stringa oggetto è posta a Ø binaric.
AØ11 INCORRECT SPECIFICATION OF IMMEDIATE (*)	Spiegazione L'espressione usata per indicare un valore immediato non è assoluto, o non ha valore compreso tra Ø e 255, oppure tra 1 e 16. Severity code Ø8 Sviluppo L'Assembler prosegue l'analisi sintattica e semantica della frase; la stringa oggetto corrispondente è comunque posta a Ø binario.

AØ12 PREMATURE END OF THE OPERAND FIELD	Spiegazione Fine prematura del campo operandi: sulla frase sorgente non è stato indicato il numero di operandi previsto, oppure la specificazione di un operando è incompleta. Severity code Ø8 Sviluppo L'analisi della frase termina; l'oggetto relativo è posto a Ø binario; nel caso di pseudo definizione dati viene generata la corrispondente stringa oggetto a Ø binario o viene allocata la corrispondente area solo se è stato possibile valutarne la lunghezza.
AØ13 ALIGNMENT ERROR	Spiegazione Errore di allineamento: la frase richiede che l'operando sia allineato alla parola o alla semi-parola; questo errore viene segnalato solo quando l'indirizzo è espresso in modo implicito e tale indirizzo è dispari; in particolare viene segnalato nel caso in cui l'operando è espresso tramite literal il cui indirizzo risulti appunto dispari. Severity code Ø4 Sviluppo L'Assembler continua l'analisi e la trasmissione della frase.
AØ14 TOO MANY OPERANDS (*)	Spiegazione Nella frase sorgente è stato specificato un numero di operandi eccessivo: . più di 16 registri specificati in una frase USING

	o DROP . più di 31 parametri specificati in una frase CALL o CALEXT o CALEXS . più di 31 nomi esterni indicati in una frase EXT . troppi operandi specificati in una qualsiasi istruzione-macchina; questo è il caso ad esempio in cui la posizione di inizio del literal eventualmente indicato non coincida con la posizione di fine dell'operando precedente. Severity code Si interrompe l'analisi e la traduzione della frase: la stringa oggetto è generata a zeri binari.
AØ15 INVALID LENGTH IN MACHINE INSTRUCTION (*)	Spiegazione La lunghezza implicita o esplicita indicata in una istruzione-macchina non ha valore compreso tra 1 e 256, oppure tra 1 e 16. La lunghezza esplicita è stata espressa tramite una espressione non assoluta. Severity code Ø8 Sviluppo L'analisi della frase sorgente prosegue; la stringa oggetto viene comunque generata a zeri binari.
AØ16 INVALID DISPLACEMENT SPECIFICATION (*)	Spiegazione In una istruzione-macchina il campo scostamento è stato indicato tramite una espressione non assoluta, o comunque tramite una espressione il cui valore non è compreso tra Ø e 4Ø95. Severity code Ø8

	Sviluppo L'analisi della frase sorgente prosegue; la stringa oggetto è comunque generata a zeri binari.
AØ17 REGISTER NOT PREVIOUSLY USED (*)	Spiegazione Un registro specificato in una frase DROP non è stato precedentemente definito come registro base in una opportuna frase USING. Severity code Ø4 Sviluppo L'Assembler ignora il registro specificato e prosegue nell'analisi della frase DROP.
AØ18 ADDRESSABILITY ERROR (*)	Spiegazione L'Assembler non riesce a risolvere l'indirizzo implicito specificato nella frase sorgente, poichè nessun registro base definito con una precedente frase USING è disponibile, cioè nessun registro base ha lo stesso attributo di rilocabilità dell'espressione che esprime l'indirizzo implicito o comunque quest'ultima ha un valore tale da dar luogo a uno scostamento 4Ø95. Severity code Ø8 Sviluppo L'analisi della frase sorgente prosegue; la stringa oggettc è generata a zeri binari.

AØ2Ø TOO MANY TERMS IN EXPRESSION (*)	Spiegazione Espressioni con più di 16 termini. Severity code Ø8 Sviluppo L'analisi della frase termina.
AØ21 MORE THAN 5 LEVELS OF PARENTHESES (*)	Spiegazione Espressicne con più di 5 livelli di parentesi. Severity code Ø8 Sviluppo L'analisi della frase termina.
AØ22 RIGHT PARENTHESES OMITTED	Spiegazione Parentesi di chiusura omesse in una espressione. Severity code Ø8 Sviluppo L'analisi della frase sorgente si interrompe.

AØ23 UNDEFINED OR NOT PREVIOUSLY DEFINED SYMBOL (*)	Spiegazione Nella frase sorgente è indicato un simbolo non definito oppure un simbolo non precedentemente definito quando la previa definizione è richiesta (ad esempio frasi EQU, ORG.....). Severity code Ø8 Sviluppo L'espressione non può essere valorizzata; l'analisi della frase prosegue sugli operandi successivi.
AØ24 INVALID SELF-DEFINING TERM (*)	Spiegazione Termine autodefinito scorretto: i caratteri indicati nella stringa non sono conformi al tipo del termine stesso. Severity code Ø8 Sviluppo L'espressione non può essere valorizzata; l'analisi della frase comunque prosegue.
AØ25 SYNTAX ERROR IN SELF-DEFINING TERM (*)	Spiegazioni Termine autodefinito non isolabile perchè non trova l'apice di chiusura della stringa in posizione corretta. Severity code Ø8 Sviluppo Si interrompe l'analisi sintattica della frase sorgente.

AØ26 VALUE OF SELF-DEFINING TERM TOO LARGE (*)	Spiegazione Termine autodefinito decimale con valore superiore a 16.777.215. Severity code Ø3 Sviluppo L'espressione non può essere valorizzata; l'analisi della frase comunque prosegue.
AØ27 PREMATURE END OF EXPRESSION (*)	Spiegazione Espressione troncata; ad esempio espressione che termina con un operatore aritmetico. Severity code Ø8 Sviluppo L'analisi della frase è interrotta.
AØ28 ARITHMETIC OVERFLOW IN EXPRESSION EVALUATION (*)	Spiegazione Overflow nel calcolo dell'espressione: il risultato intermedio non è compreso tra $2^{31}-1$ e -2^{31}. Severity code Ø8 Sviluppo L'analisi della frase prosegue.
AØ29 FINAL RESULT OF EXPRESSION OUT OF RANGE (*)	Spiegazione Risultato dell'espressione negativo o comunque supe-

	riore a $2^{24}-1$. Severity code Ø8 Sviluppo L'analisi sintattica e semantica della frase sorgente prosegue.
AØ3Ø RELOCABILITY ERROR (*)	Spiegazione Non è possibile valutare l'attributo di rilocabilità dell'espressione perchè più di un termine rilocabile risulta spaiato oppure il termine spaiato è preceduto dall'operatore. - . Severiy code Ø8 Sviluppo L'espressione non è valorizzata; l'analisi della frase prosegue.
AØ31 FIRST OPERAND OF DIV./MULT. IS RELOCATABLE (*)	Spiegazione Espressione in cui compare un termine rilocabile come primo operando in una divisione o di una moltiplicazione. Severity code Ø8 Sviluppo L'espressione non è valorizzata; l'analisi della frase prosegue.

AØ32 SECOND OPERAND OF DIV./MULT. IS RELOCATABLE (*)	Spiegazione Espressione in cui il secondo operando di una divisione o moltiplicazione è rilocabile. Severity code Ø8 Sviluppo L'espressione non è valorizzata; comunque l'analisi della frase prosegue.
AØ33 BOTH THE OPERAND OF DIV./MULT. ARE RELOCATABLE (*)	Spiegazione Espressione in cui compare una divisione o moltiplicazione fra termini rilocabili. Severity code Ø8 Sviluppo Non è possibile calcolare il valore dell'espressione; l'analisi della frase prosegue.
AØ34 EXPRESSION START WITH OPERATOR (*)	Spiegazione Il primo termine dell'espressione è preceduto dall'operatore aritmetico. Severity code Ø8 Sviluppo L'analisi dell'espressione e della frase termina.

AØ35 DIVISION BY ZERO IN EXPRESSION (*)	Spiegazione Nell'espressione compare una divisone con valore Ø a divisore. Severity code Ø4 Sviluppo Il risultato della divisione di un termine per un altro termine con valore Ø e posto a Ø.
AØ36 UNKNOWN TYPE (*)	Spiegazione Scorretta indicazione del tipo in una frase DC/DS o in un literal. Severity code 12 Sviluppo L'analisi della frase viene interrotta; non viene generata alcuna stringa oggetto.
AØ37 DUPLICATION FACTOR ERROR (*)	Spiegazione scorretta indicazione del fattore di duplicazione di una frase DC/DS o in un literal: il fattore di duplicazione è stato indicato con una espressione non assoluta, o comunque ha valore superiore a 65.535; in un literal è stato indicato un fattore di duplicazione uguale a zero. Severity code 12 Sviluppo Prosegue l'analisi della frase; non viene generata alcuna stringa oggetto.

AØ38 DUPLICATION FACTOR NOT ALLOWED (*)	Spiegazione Errata indicazione del fattore di duplicazione in frasi DC o DS o in literal di tipo indirizzo. Severity code 12 Sviluppo L'analisi della frase termina; non viene generata alcuna stringa oggetto.
AØ39 S-TYPE CONSTANT IN LITERAL (*)	Spiegazione Costante indirizzo di tipo S indicata in un literal. Severity code 12 Sviluppo L'analisi della definizione del literal si interrompe; non viene generata alcuna stringa oggetto.
AØ4Ø INVALID LENGTH MODIFIER (*)	Spiegazione Errata indicazione del modificatore di lunghezza in una frase DC o DS o in un literal: il modificatore di lunghezza è stato espresso con una espressione non assoluta, o comunque ha valore fuori range rispetto al tipo di costante. Severity code 12 Sviluppo L'analisi della frase prosegue; non viene generata alcuna stringa oggetto.

AØ41 PREVIOUSLY DEFINED NAME	Spiegazione Nella frase è definito un nome già definito in una precedente frase sorgente. Severity code Ø8 Sviluppo Rimane valida la prima definizione del nome.
AØ42 INVALID OPERAND IN ORG STATEMENT	Spiegazione Nella frase ORG l'operando è espresso tramite una espressione non rilocabile, o una espressione il cui attributo di rilocabilità è diverso rispetto alla sezione in cui compare la frase stessa. Severity code Ø8 Sviluppo Il location counter rimane immutato; la frase è quindi inoperante.
AØ43 CONSTANT TRUNCATED-HIGH ORDER DIGITS LOST	Spiegazione I bit di ordine superiore di una costante o literal di tipo H/F sono persi poichè il campo definito è troppo piccolo per contenere l'intero valore delle costanti. Severity code Ø4 Sviluppo Viene generata una stringa oggetto per la lunghezza implicita o esplicita indicata con il valore troncato.

AØ44 INVALID CONSTANT (*)	Spiegazione Errata indicazione del valore della costante in frasi DC e DS o in literal: i caratteri indicati non sono compatibili con il tipo di costante. Severity code Ø8 Sviluppo La stringa oggetto viene generata a zeri binari se la lunghezza della costante è indicata esplicitamente; altrimenti nessuna stringa viene generata.
AØ45 QUOTES NOT PAIRED IN DC/DS/LITERAL (*)	Spiegazione L'indicazione del valore costante in frasi di definizione dati o in un literal non ha l'apice di chiusura; apici non pari nelle costanti tipo C. Severity code Ø8 Sviluppo La stringa oggetto viene generata a zeri binari se la lunghezza della costante è indicata esplicitamente altrimenti nessuna stringa è generata.
AØ46 INCORRECT OR DUPLICATE KEYWORD OPERAND (*)	Spiegazione Operando a parola-chiave compare più di una volta nell'ambito di una macro-istruzione oppure parola-chiave scorretta alla posizione indicata. Severity code 12

	Sviluppo L'Assembler prosegue nell'analisi sintattica della macro-istruzione; non viene però generata alcuna espansione.
AØ47 INVALID OPERAND IN MACRO-INSTRUCTION (*)	Spiegazione Operando invalido in una macro-istruzione perchè non è riconosciuto nè come simbolo nè come termine autodefinito decimale. Severity code 12 Sviluppo L'Assembler prosegue l'analisi sintattica della frase; non viene generata alcuna espansione.
AØ48 INVALID USING STATEMENT	Spiegazione Nella frase using è stato indicato come registro base il registro generale Ø non in prima posizione. Severity code Ø8 Sviluppo L'analisi e l'esecuzione della frase si interrompe.

Messaggi di errore
Gruppo F

Questo paragrafo descrive gli errori che possono verificarsi mentre il sistema è nello stato calcoli immediati o di debugging.

Codice di errore	Descrizione
114	La definizione di funzione monolinea è ricorsiva.
115	Nello stato calcoli immediati si introduce un nome di variabile diverso da quelli accettabili; oppure nello stato di debugging si introduce il nome di una variabile semplice, o con indice, od un nome di una funzione che non sono specificati nel programma presente in memoria principale.
117	Lo spazio disponibile in memoria principale non è sufficiente per contenere anche l'ultima linea introdotta.

Messaggi di errore
Gruppo G

Questo paragrafo descrive gli errori che si possono verificare durante una operazione di accesso al floppy disk.

Codice di errore	Descrizione
151	Sull'unità ● il disco è stato montato male o è rovinato; oppure la stessa unità è fuori uso.
152	Sull'unità ● il disco è stato montato male o è rovinato; oppure la stessa unità è fuori uso.
156	Nell'unità floppy disk non c'è un floppy disk sistema.

Messaggi di errore
Gruppo H

Questo paragrafo descrive gli errori che si possono verificare durante l'introduzione o l'esecuzione di un comando di sistema.

Codice di errore	Descrizione
172	E' riferito un canale RS232 non esistente nella configurazione di sistema installata.
175	La dimensione specificata per la memoria utente, nel comando CONFIGURE, è superiore alla dimensione real-

	mente installata.
176	E' stato specificato l'impiego di una stampante IPSO, nel comando CONFIGURE, ma nel sistema non è presente il relativo canale.
181	Lo spazio disponibile in memoria utente non è sufficiente per eseguire l'operazione richiesta.
183	La sottolibreria specificata non è stata inizializzata (PK=Ø o COM=Ø o US=Ø nel relativo comando EXEC LBCREATE) oppure contiene già il numero di file per essa, definito durante l'esecuzione del programma LBCREATE.
184	Il floppy disk specificato come utente non è stato inizializzato come tale.
185	Il floppy disk sistema non è stato inizializzato per contenere le sottolibrerie (package e/o comune e/o utente).
186	C'è già un file con questo nome.
187	Non esiste un file con questo nome.
188	Il numero di file allocati per la libreria è stato superato oppure sul floppy disk lo spazio disponibile non è sufficiente per eseguire l'operazione richiesta.
190	Il comando non può essere accettato nel presente stato del sistema.
191	Non è presente il nome del file.
192	C'è un carattere non corretto.
193	Mancano uno o più operandi.
194	Il numero di linea specificato non esiste.
196	Un operando non è valido.
198	Lo spazio richiesto supera la capacità del floppy disk.
199	L'operazione richiesta non è permessa per file pro-

	tetti.
200	L'operazione richiesta non è permessa per sottolibrerie protette.
201	L'operazione richiesta non è permessa per sistemi nella configurazione monodisco.
202	La registrazione o creazione del file su disco non è possibile per insufficienza di spazio nella sottolibreria o su disco.
203	Il primo operando è maggiore del secondo operando.
205	L'operazione richiesta non è permessa per linee protette.
207	Il tipo di file non è coerente con l'operazione richiesta.
208	L'opzione specificata non è disponibile nel sistema.
209	E' stato generato un numero di linea maggiore di 9999.
211	Nella memoria principale non c'è nè un programma nè un file testo.
212	La linea o le linee da stampare non ci sono.

Messaggi di errore Gruppo 1

Questo paragrafo descrive gli errori che possono accadere durante un richiamo o l'esecuzione di un programma di utilità.

Codice di errore	Descrizione
145	Il file di direttive di input contiene più di 16 file di tipo testo.
232	$n_1+n_2+n_3$ è maggiore di 14.
234	Manca il nome del programma di utilità.
235	Il programma di utilità specificato non esiste.

Messaggi di errore
Gruppo J

Questo paragrafo descrive gli errori prodotti da condizioni anormali del sistema.

Codice di errore	Descrizione
4 *A	La memoria principale è danneggiata; il suo contenuto è cancellato.
12 *A	Il floppy disk sistema è danneggiato.
16 *A	L'etichetta di flusso è danneggiata.
ABN FD*	Il trascinatore superiore è in condizioni anormali. Verificare che uno sportello non sia aperto.
ABN FD**	Il trascinatore inferiore è in condizioni anormali. Verificare che uno sportello non sia aperto.
ABN PRT	La stampante integrata è in condizioni anormali. Il contenuto della memoria principale non è alterato. Verificare che la testina di stampa non sia alzata.

Nota

Ogni altro codice di errore non compreso in quelli elencati, denuncia una condizione anormale del sistema. Quando vengono segnalati questi tipi di errore si premano contemporaneamente i tasti SHIFT e CLEAR RECALL e quindi il tasto di console CONTINUE ; se il messaggio READY appare sul display si può utilizzare il sistema di nuovo. Si provino a ripetere le operazioni precedenti.

Printed in Italy