Consiglio Nazionale delle Ricerche

ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE

PISA

ARCHITETTURA SOFTWARE DI UN SISTEMA GRAFICO
AD ALTA PERFORMANCE

F. Fabbrini, L. Mastropietro, C. Montani

Nota interna B4-53
Convenzione Selenia S.p.A. - C.N.R.
Ottobre 1986

CONSIGLIO NAZIONALE DELLE RICERCHE
Istituto di Elaborazione dell'Informazione

ARCHITETTURA SOFTWARE DI UN SISTEMA GRAFICO AD ALTA PERFORMANCE(+)

F. Fabbrini, L. Mastropietro(++) e C. Montani

(+) Lavoro svolto nell'ambito della Convenzione Selenia S.p.A. - C.N.R.

(++) Borsista della Selenia S.p.A.

1. INTRODUZIONE

Il presente lavoro, svolto nell'ambito della Convenzione Selenia S.p.A. - C.N.R., descrive l'architettura software di un sistema grafico ad alte prestazioni e ne presenta la realizzazione emulata su personal computer.

Sia la descrizione dell'architettura che la realizzazione non sono perfettamente aderenti alla realizzazione finale approntata dalla Selenia S.p.A. ma rappresentano piuttosto le linee generali seguite nello sviluppo dell'intero sistema.

Le scelte operate e le soluzioni adottate possono essere comprese pienamente soltanto se si inquadrano le esigenze ed i vincoli delle due entita' che regolano e, in qualche modo definiscono l'architettura software di un sistema: da una parte l'utente, con le sue esigenze applicative, dall'altra l'hardware, con i suoi vincoli architetturali.

L'utente applicativo vede il sistema come una libreria di comandi grafici e di controllo. Tali comandi devono consentirgli di:

(a) inizializzare il sistema;

(b) inviare direttamente al processore grafico primitive grafiche bi- e tri-dimensionali, comandi per la selezione di attributi delle primitive, comandi per l'impostazione di trasformazioni geometriche (di modello, di vista, di proiezione e di workstation), istruzioni di input grafico e non, comandi per il salvataggio (mediante un meccanismo a stack) dello stato degli attributi e delle trasformazioni geometriche;

(c) creare oggetti grafici (segmenti) contenenti, oltre alle entita' descritte al punto precedente, comandi per l'esecuzione di altri segmenti ed etichette per il successivo editing del segmento grafico stesso. Tramite le chiamate ad oggetti all'interno di oggetti l'utente puo' costruire gerarchie di segmenti (strutture);

(d) editare segmenti cancellando, inserendo o sostituendo comandi grafici modificando cosi' l'apparenza o l'organizzazione gerarchica delle strutture;

(e) cancellare interi segmenti dalla memoria del sistema;

(f) eseguire strutture: forzare cioe' il traversal e

l'esecuzione dei comandi contenuti nella struttura semplicemente invocando l'esecuzione del segmento radice;

(g) impedire il traversal di sottoalberi di una struttura disabilitando l'esecuzione tra determinate coppie di label di un segmento (tag).

L'hardware grafico previsto non pone grossi vincoli per cio' che riguarda l'esecuzione dei comandi grafici, dei comandi di selezione degli attributi e delle trasformazioni geometriche, di salvataggio e ripristino dello stato e delle istruzioni di input.

L'unica vera limitazione, dal punto di vista dell'architettura software, imposta dall'hardware, e' che l'esecuzione (traversal) delle strutture non puo' essere espletata dalla CPU centrale della macchina: il software di gestione si deve limitare a preparare una opportuna lista di coppie del tipo indirizzo - dimensioni del blocco sulla base della quale un dispositivo DMA provvedera' a leggere dalla memoria centrale le sequenze di comandi grafici e di controllo da inviare al processore grafico.

Questo tipo di soluzione si e' resa necessaria in considerazione dell'alto numero di traversal cui una struttura e' sottoposta in confronto alle operazioni di creazione o di editing.

E' evidente che non disponendo di un traversal "interpretato" (esecuzione dei vari comandi e percorrimento della gerarchia secondo le chiamate ad oggetti impostate dall'utente) e' necessario che la struttura dati interna del sistema sia sofisticata al punto da consentire la linearizzazione della struttura senza rileggere, di volta in volta, il contenuto dei singoli oggetti.

Per meglio illustrare quanto sopra descritto ci avvarremo dell'esempio delle figure seguenti. Supponiamo che l'utente inserisca nel suo programma applicativo la sequenza di istruzioni mostrate in figura 1. L'esecuzione di tale sequenza porta alla creazione dei due oggetti mostrati in figura 2. Se il comando successivo lanciato dall'utente fosse callobj(FATHER) il sistema dovrebbe consentire al DMA incaricato del trasferimento di linearizzare la struttura in modo da inviare al processore grafico i comandi mostrati in figura 3a. Se il comando callobj(FATHER) fosse preceduto dall'istruzione disable(DAQUI,AQUI) la linearizzazione della struttura dovrebbe portare all'invio dei comandi di figura 3b.

```
...
init();                          /* initialize the system */
clears(0);                       /* clear the screen (color black) */
makeobj(FATHER);                 /* open segment 1 for creation */
   vwport(80,559,0,479);         /* define a square viewport */
   window(-10.,10.,-10.,10.);    /* define window extents */

   color(153);                   /* select current color (green) */
   prmfil(0);                    /* don't fill closed primitives */
   move(5.,5.);                  /* move current point to ... */
   callobj(2);                   /* execute segment 2 */

maketag(DAQUI);                  /* define tag */
   color(24);                    /* select current color (red) */
   prmfil(1);                    /* fill closed primitives */
   move(-5.,5.);                 /* move current point to ... */
   callobj(2);                   /* execute segment 2 */
maketag(AQUI);                   /* define tag */

   color(90);                    /* select current color (blue) */
   prmfil(0);                    /* don't fill closed primitives */
   move(-5.,-5.);                /* move current point to ... */
   callobj(2);                   /* execute segment 2 */

   color(120);                   /* select current color (cyan) */
   prmfil(1);                    /* fill closed primitives */
   move(5.,-5.);                 /* move current point to ... */
   callobj(2);                   /* execute segment 2 */

endobj();                        /* close object 1 */

makeobj(SON);                    /* open object 2 for creation */
   circle(2.);                   /* draw a circle centered at c.p. */
   rectr(3.,4.);                 /* draw a relative rectangle */

endobj();                        /* close object 2 */
...
```

Fig. 1

Segment FATHER

```
    vwport(80,559,0,479)
    window(-10.,10.,-10.,10.)
    color(153)
    prmfil(0)
    move(5.,5.)
    callobj(2)
tag(DAQUI)
    color(24)
    prmfil(1)
    move(-5.,5.)
    callobj(2)
tag(AQUI)
    color(90)
    prmfil(0)
    move(-5.,-5.)
    callobj(2)
    color(120)
    prmfil(1)
    move(5.,-5.)
    callobj(2)
```

Segment SON

```
    circle(2.)
    rectr(3.,4.)
```

Fig. 2

```
vwport(80,559,0,479)
window(-10.,10.,-10.,10.)
color(153)
prmfil(0)
move(5.,5.)
   circle(2.)
   rectr(3.,4.)
color(24)
prmfil(1)
move(-5.,5.)
   circle(2.)
   rectr(3.,4.)
color(90)
prmfil(0)
move(-5.,-5.)
   circle(2.)
   rectr(3.,4.)
color(120)
prmfil(1)
move(5.,-5.)
   circle(2.)
   rectr(3.,4.)
```

Fig. 3a

```
vwport(80,559,0,479)
window(-10.,10.,-10.,10.)
color(153)
prmfil(0)
move(5.,5.)
   circle(2.)
   rectr(3.,4.)
color(90)
prmfil(0)
move(-5.,-5.)
   circle(2.)
   rectr(3.,4.)
color(120)
prmfil(1)
move(5.,-5.)
   circle(2.)
   rectr(3.,4.)
```

Fig. 3b

Per consentire al lettore di familiarizzare con le potenzialita' del sistema forniamo l'esempio delle figure seguenti mostrando di volta in volta come le diverse funzioni di editing agiscono sulle strutture create dall'utente.

L'esempio consiste di un testo composto da un numero variabile di righe e contornato da un riquadro. Il segmento radice della struttura (FATHER) provvede a disegnare la linea alta del riquadro e, una volta eseguito il segmento che provvede al tracciamento delle righe di testo, la line bassa del riquadro.

Per ogni linea di testo il segmento TEXT attiva il segmento BORDER che provvede a tracciare due piccoli segmenti verticali ai lati di ciascuna linea di testo: tali segmenti formeranno, a traversal ultimato, i lati verticali del riquadro.

La figura 4b e' ottenuta (ovviamente in modo grafico e in colore rosso) lanciando il comando callobj(FATHER) dopo che la struttura di figura 4a e' stata creata. L'editing di figura 5a ed il successivo ritraversal portano alla figura 5b. In modo analogo le operazioni di editing mostrate nelle figure 6a, 7a, 8a, 9a e 10a ed i relativi comandi callobj(FATHER) producono i disegni delle figure 6b, 7b, 8b, 9b e 10b rispettivamente.

Una spiegazione dettagliata delle funzioni di controllo utilizzate nell'esempio (delete, insert, replace, etc.) puo' essere trovata nelle pagine seguenti l'esempio stesso.

```
...
init();                          /* initialize the system */
window(0.,639.,0.,479.);         /* define window like viewport */
color(24);                       /* select current color (red) */
tjust(2,2);                      /* center text on current point */
tsize(12.);                      /* select text size */

makeobj(FATHER);                 /* open segment for creation */
   clears(0);                    /* clear screen (black) */
   move(100.,470.);
   draw(539.,470.);              /* draw upper horizontal line */
   move(319.,470.);
   callobj(TEXT);                /* execute segment TEXT */
   mover(-219.,10.);
   drawr(439.,0.);               /* draw lower horizontal line */
endobj();                        /* close segment */

makeobj(TEXT);                   /* open segment for creation */
   callobj(BORDER);              /* execute segment BORDER */
   text("Questo");               /* draw graphical text */
   mover(0.,-20.);               /* move to the next line */
maketag(E);                      /* define label */
   callobj(BORDER);              /* execute segment BORDER */
   text("e'");                   /* draw graphical text */
   mover(0.,-20.);               /* move to the next line */
maketag(UN);                     /* define label */
   callobj(BORDER);              /* execute segment BORDER */
   text("un");                   /* draw graphical text */
   mover(0.,-20.);               /* move to the next line */
maketag(ESEMPIO);                /* define label */
   callobj(BORDER);              /* execute segment BORDER */
   text("esempio");              /* draw graphical text */
   mover(0.,-20.);               /* move to the next line */
endobj();                        /* close segment */

makeobj(BORDER);                 /* open segment for creation */
   mover(-219.,10.);
   drawr(0.,-20.);               /* draw left vertical segment */
   mover(439.,20.);
   drawr(0.,-20.);               /* draw right vertical segment */
   mover(-220.,10.);             /* current point to the center */
endobj();                        /* close segment */
...
```

Fig. 4a

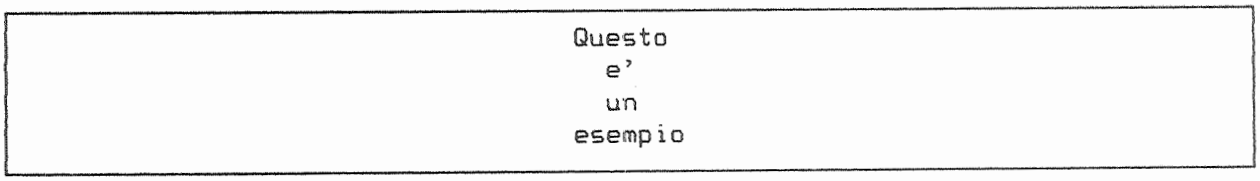


Fig. 4b

```
editobj(TEXT);                  /* open segment for editing */
   replace(0,1);                /* select replace mode */
   text("questo");
   insert(0,0);                 /* select insert mode (top) */
   callobj(BORDER);
   text("Forse");
   mover(0.,-20.);
endobj();                       /* close object */
```

Fig. 5a

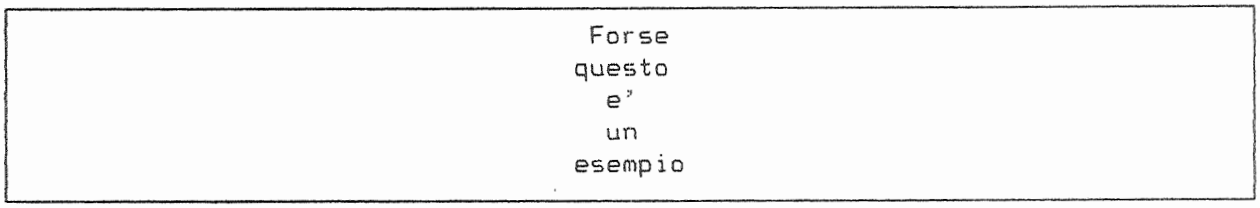


Fig. 5b

```
editobj(TEXT);                  /* open segment for editing */
   insert(UN,0);                /* select insert mode (tag UN) */
   callobj(BORDER);
   text("(ci pare)");
   mover(0.,-20.);
endobj();                       /* close object */
```

Fig. 6a

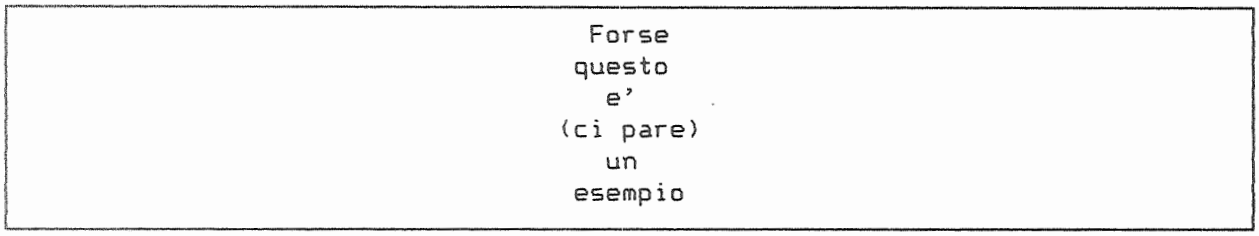


Fig. 6b

```
editobj(TEXT);                  /* open segment for editing */
   insert(ESEMPIO,30);          /* select insert mode (30 lines */
   callobj(BORDER);             /* below tag ESEMPIO) */
   text("bello.");
   mover(0.,-20.);
   callobj(BORDER);
   text("(Queste righe");
   mover(0.,-20.);
   callobj(BORDER);
   text("servono ad");
   mover(0.,-20.);
   callobj(BORDER);
   text("allungare l'esempio)");
   mover(0.,-20.);
endobj();                       /* close object */
```

Fig. 7a

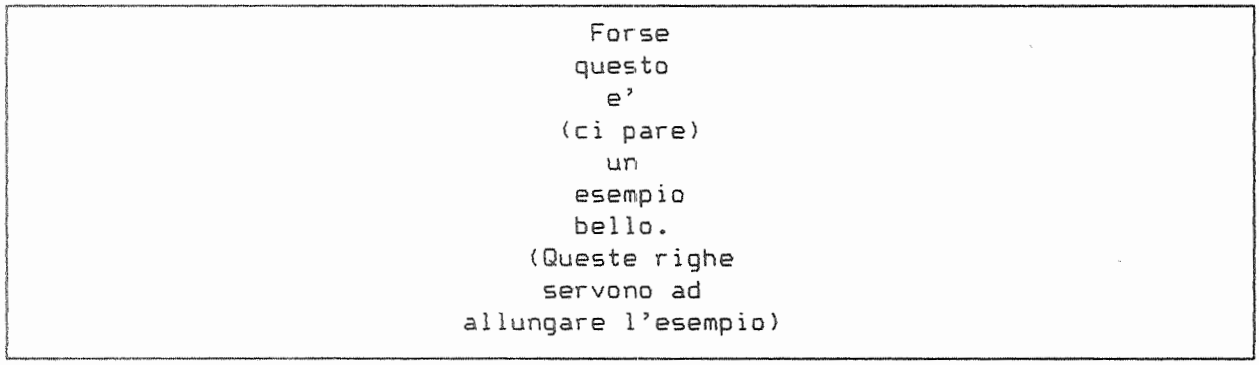


Fig. 7b

```
editobj(TEXT);                  /* open segment for editing */
   delete(0,0,3);               /* delete 3 statement on top */
   replace(0,1);                /* select replace mode */
   text("Questo");
endobj();                       /* close object */
```

Fig. 8a

```
                Questo
                  e'
                (ci pare)
                  un
                esempio
                bello.
              (Queste righe
               servono ad
            allungare l'esempio)
```

Fig. 8b

```
editobj(TEXT);                  /* open segment for editing */
   delete(UN,0,3);              /* delete 3 statement from UN */
endobj();                       /* close object */
```

Fig. 9a

```
                Questo
                  e'
                  un
                esempio
                bello.
              (Queste righe
               servono ad
            allungare l'esempio)
```

Fig. 9b

```
editobj(TEXT);                  /* open segment for editing */
   delete(ESEMPIO,5,9);         /* delete 3 statement from UN */
endobj();                       /* close object */
```

Fig. 10a

```
Questo
  e'
  un
esempio
bello.
```

Fig. 10b

2. GLI STATI INTERNI

Abbiamo visto, negli esempi precedenti, come il linguaggio grafico consti di funzioni di controllo (per la creazione, esecuzione e modifica delle strutture) e di funzioni grafiche vere e proprie. L'effetto che l'esecuzione di una primitiva di controllo o grafica provoca sul sistema e' dipendente dallo stato interno del sistema stesso.

Gli stati interni sono CLOSED, OPEN, CREATION ed EDITING. Lo stato EDITING e' a sua volta suddiviso nei due stati INSERISCI e RIMPIAZZA.

Per evidenziare i diversi comportamenti tenuti dal sistema nei diversi ambienti vediamo come esso reagisce ai comandi di controllo e grafici lanciati dall'utente.

callobj(obj)
- OPEN: provoca il traversal (esecuzione) della struttura la cui radice e' rappresentata dal segmento obj.
- CREATION: la primitiva callobj(obj) e' inserita nel segmento correntemente aperto in creazione.
- EDITING: la primitiva callobj(obj) viene inserita all'offset corrente del segmento che si sta editando (sottostato INSERISCI) oppure rimpiazza il comando callobj presente all'offset corrente del segmento che si sta editando (sottostato RIMPIAZZA).
- CLOSED: errore.

delete(tag,offset,count)
- EDITING: provoca la cancellazione di count comandi grafici a partire da disp comandi dopo la label tag.
- ALTRI: errore.

delobj(obj)
- OPEN: provoca la cancellazione del segmento obj.
- ALTRI: errore.

disable(btag,etag)
- OPEN: in seguito al lancio della funzione disable le esecuzioni delle strutture grafiche non porteranno al traversal delle parti di segmenti eventualmente comprese tra le label btag e etag.
- ALTRI: errore.

editobj(obj)
- OPEN: provoca l'ingresso del sistema nello stato EDITING del segmento obj.
- ALTRI: errore.

enable(btag,etag)
OPEN: ha la funzione opposta a disable.
ALTRI: errore.

endobj()
CREATION: provoca la chiusura del segmento in creazione e il ritorno allo stato OPEN.
EDITING: provoca la chiusura del segmento che si sta editando e il ritorno allo stato OPEN.
ALTRI: errore.

exit()
OPEN: provoca la chiusura del sistema e ritorno allo stato CLOSED (non implementata).
ALTRI: errore.

init()
CLOSED: provoca l'inizializzazione del sistema e l'ingresso nello stato OPEN.
ALTRI: errore.

insert(tag,offset)
EDITING: provoca lo spostamento del pointer corrente offset comandi dopo la label tag. Il sottostato di EDITING e' impostato a INSERISCI.
ALTRI: errore.

makeobj(obj)
OPEN: provoca l'apertura in creazione del segmento obj e l'ingresso del sistema nello stato CREATION.
ALTRI: errore.

maketag(tag)
CREATION: provoca l'inserimento della label tag nel segmento in creazione.
EDITING: provoca l'inserimento della label tag all'offset corrente del segmento che si sta editando.
ALTRI: errore.

replace(tag,offset)
EDITING: provoca lo spostamento del pointer corrente offset comandi dopo la label tag. Il sottostato di EDITING e' impostato a RIMPIAZZA.
ALTRI: errore.

Per quanto riguarda le funzioni grafiche va osservato che, da un punto di vista strettamente sistemistico, esse vengono gestite in modo identico indipendentemente dalla loro specifica funzione. Nel sistema queste primitive sono indicate con il termine generic.

generic(operands)
OPEN: la primitiva e' eseguita direttamente.

CREATION: la primitiva e' aggiunta al segmento correntemente in creazione.

EDITING: la primitiva e' inserita all'offset corrente del segmento che si sta editando (sottostato INSERISCI) oppure sostituisce la primitiva GENERIC di uguale lunghezza presente all'offset corrente del segmento che si sta editando (sottostato RIMPIAZZA).

CLOSED: errore.

Lo schema di figura 11 presenta un diagramma degli stati del sistema e le funzioni di controllo che provocano transazione di stato.

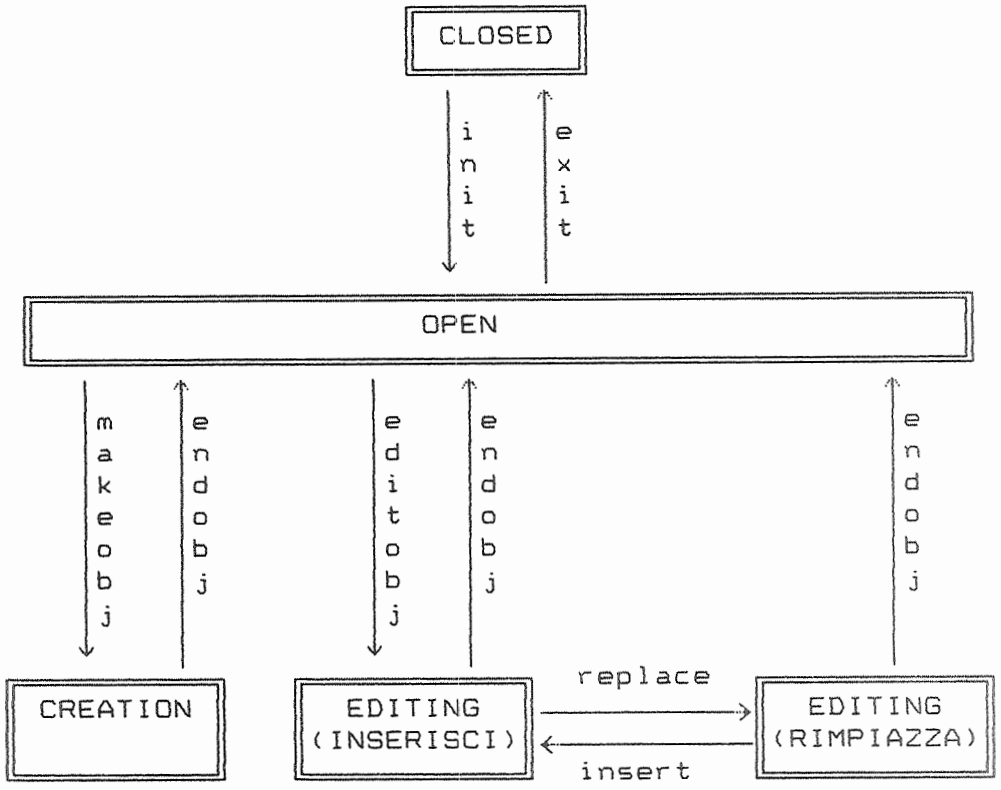


Fig. 11

3. LA STRUTTURA DATI

Come accennato nell'introduzione e mostrato negli esempi delle figure 1, 2 e 3 il problema piu' complesso da risolvere e' legato alla organizzazione interna delle strutture dati. I vincoli da rispettare (alcuni dei quali in netto contrasto tra loro) sono:

(a) necessita' di esecuzioni frequenti e rapide delle strutture;

(b) possibilita' di editing dei segmenti mediante funzioni di inserzione, cancellazione o modifica di comandi grafici;

(c) capacita' di "eseguire" le strutture senza interpretare il contenuto dei singoli comandi.

A questi vincoli, gia' di per se' onerosi, si aggiunge il problema, di natura sistemistica, della quantita' di memoria da allocare dinamicamente ai singoli segmenti senza conoscerne in anticipo la lunghezza.

Le scelte operate nell'architettura sono influenzate dalle seguenti considerazioni:

(a) in assenza di coppie attive disable il flusso sequenziale del traversal all'interno di un segmento e' interrotto dalla presenza di comandi callobj o dalla fine fisica del segmento;

(b) da un punto di vista statistico le operazioni di editing non allungano il segmento di una quantita' di istruzioni molto elevata.

Sulla base di queste considerazioni sono state adottate le seguenti scelte:

(a) ogni segmento e' suddiviso in uno o piu' sottosegmenti logici: il sottosegmento e' costituito dai comandi compresi tra due comandi callobj successivi (oppure tra l'inizio del segmento e un callobj, tra un callobj e la fine del segmento, etc.);

(b) ogni segmento inizia alla testa di una pagina fisica ed e' memorizzato su una o piu' pagine fisiche (le pagine, nella nostra realizzazione, sono di lunghezza fissa); ciascun sottosegmento puo' risiedere su una sola pagina oppure su piu' pagine fisiche;

(c) in fase di creazione viene riservata una certa quantita' di memoria (free area) alla fine di ciascuna pagina per le eventuali successive operazioni di editing;

Le figure 12, 13 e 14 mostrano singole istanze delle tabelle (allocate staticamente nel sistema) e delle liste (allocate dinamicamente) definite.

Segment Table (STAB): e' la tabella pricipale nel sistema. Ogni entrata non vuota descrive un oggetto (segmento) di nome NAME avente tanti figli di primo grado (comandi callobj al suo interno) quanti sono gli elementi della lista puntata da PATHPTR, organizzato in sottosegmenti come descritto dalla lista puntata da SEGPTR ed i cui tag sono descritti dalla lista puntata da TAGPTR.

L'indicatore TAGFLAG e' utilizzato nella fase di traversal di una struttura e (quando e' impostato al valore true) informa che per l'oggetto NAME si e' gia' provveduto a marcare i tag in esso contenuti con le marche di DISABLE o ENABLE eventualmente presenti (TAGFLAG evita che questa operazione di marcatura sia effettuata piu' volte per oggetti richiamati ripetutamente all'interno di un traversal).

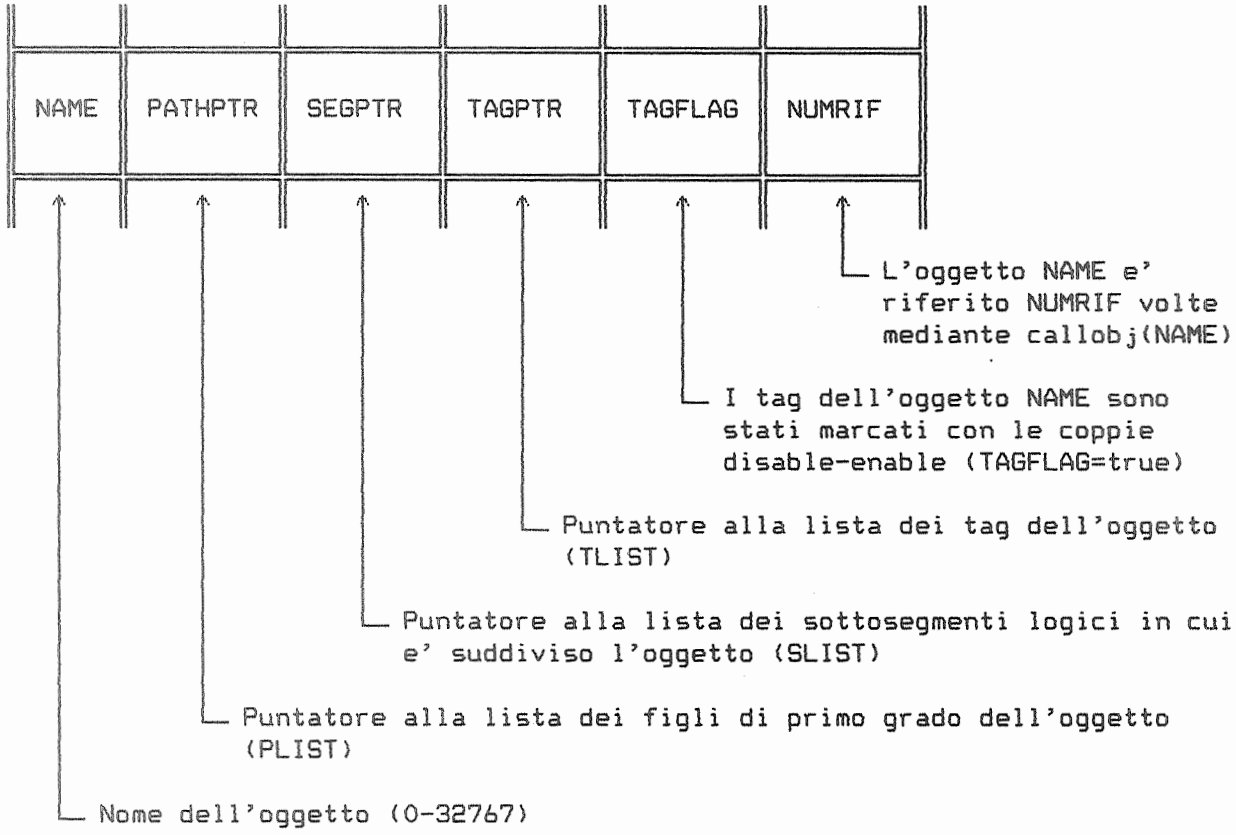
Il contatore NUMRIF indica i riferimenti di altri oggetti all'oggetto NAME ed ha la funzione di evitare che l'entrata relativa all'oggetto sia cancellata completamente dalla tabella quando l'oggetto, pur essendo stato cancellato, e' ancora riferito (mediante comandi callobj) da altri.

Per convenzione all'oggetto riferito da altri ma non creato corrispondera' in STAB una entrata costituita solo dal nome e dai campi TAGFLAG e NUMRIF; per l'oggetto creato (anche se vuoto) tramite il camando makeobj sara' pure inizializzata la lista dei sottosegmenti logici (puntatore SEGPTR).

Address Table (ATAB): Ogni segmento grafico e' memorizzato su una o piu' pagine fisiche di lunghezza variabile (ciascun segmento inizia sempre alla testa di una pagina fisica).

La tabella ATAB descrive le caratteristiche delle pagine fisiche allocate dal sistema. L'indice della i-esima entrata della tabella descrive la pagina di nome i, di indirizzo fisico ADDRPTR, di lunghezza complessiva PAGELEN e avente FREELEN bytes di memoria ancora utilizzabile per l'esecuzione di ulteriori comandi grafici.

Segment Table : STAB



Address Table: ATAB

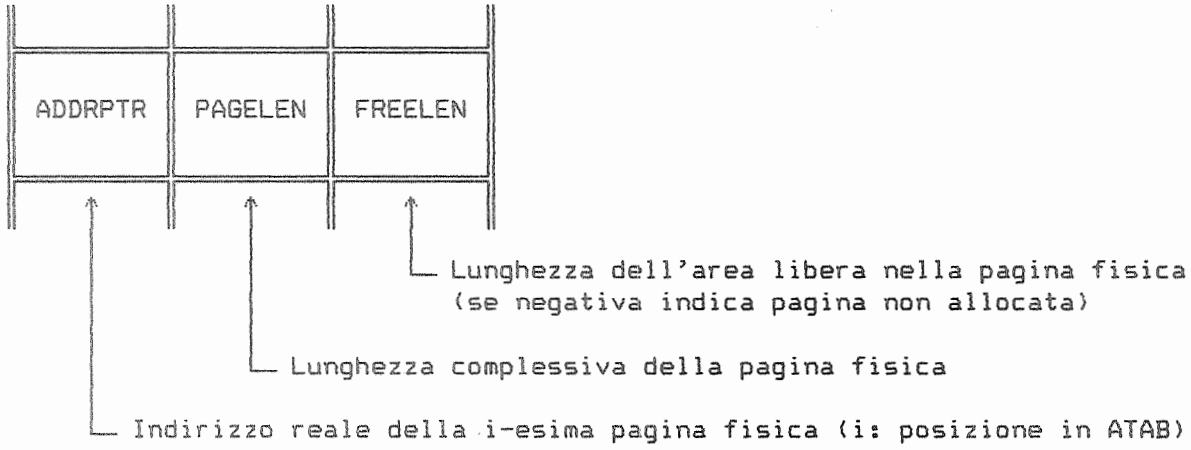
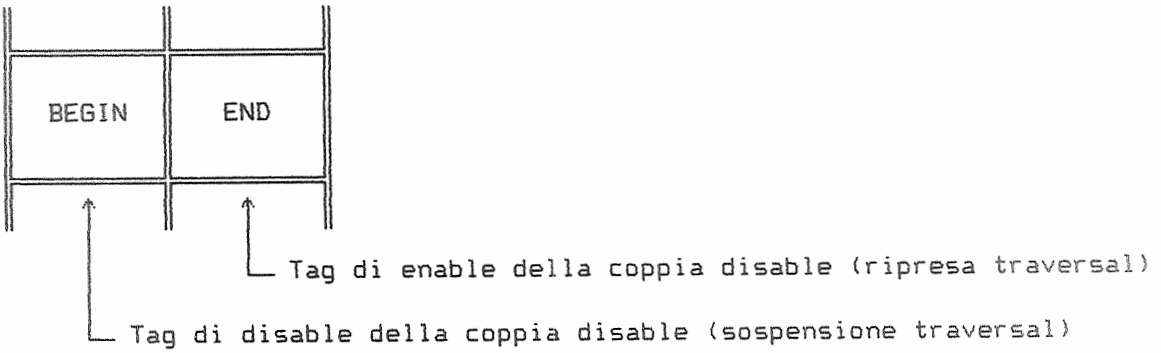
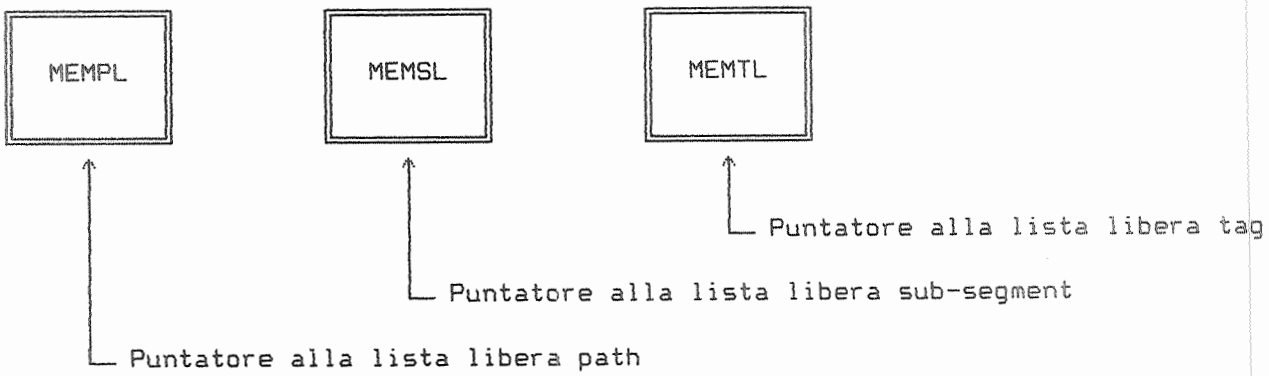


Fig. 12

Disable Tag Table: TTAB





Path List: PLIST

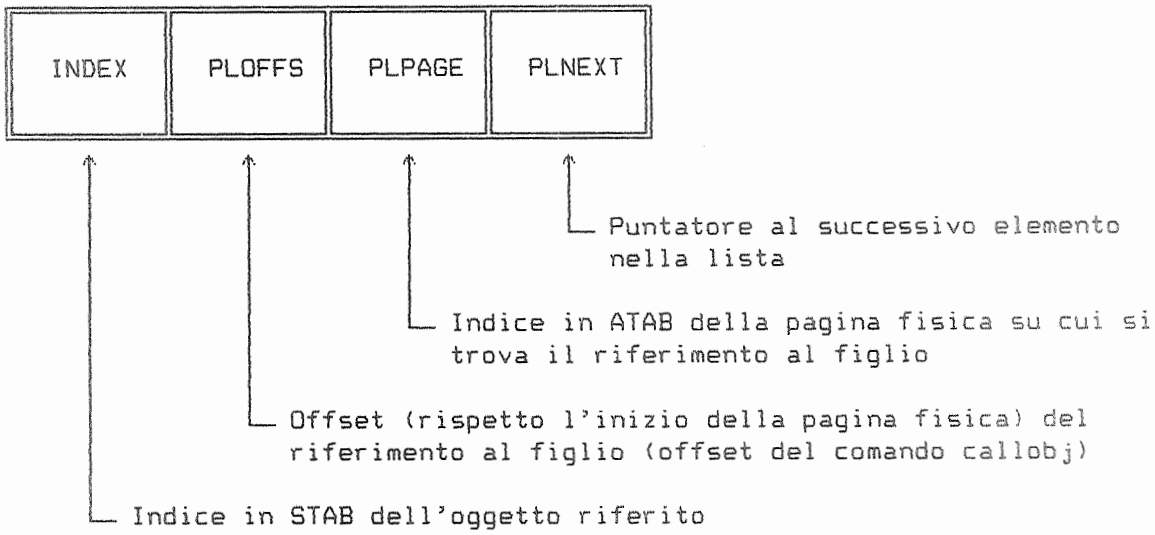
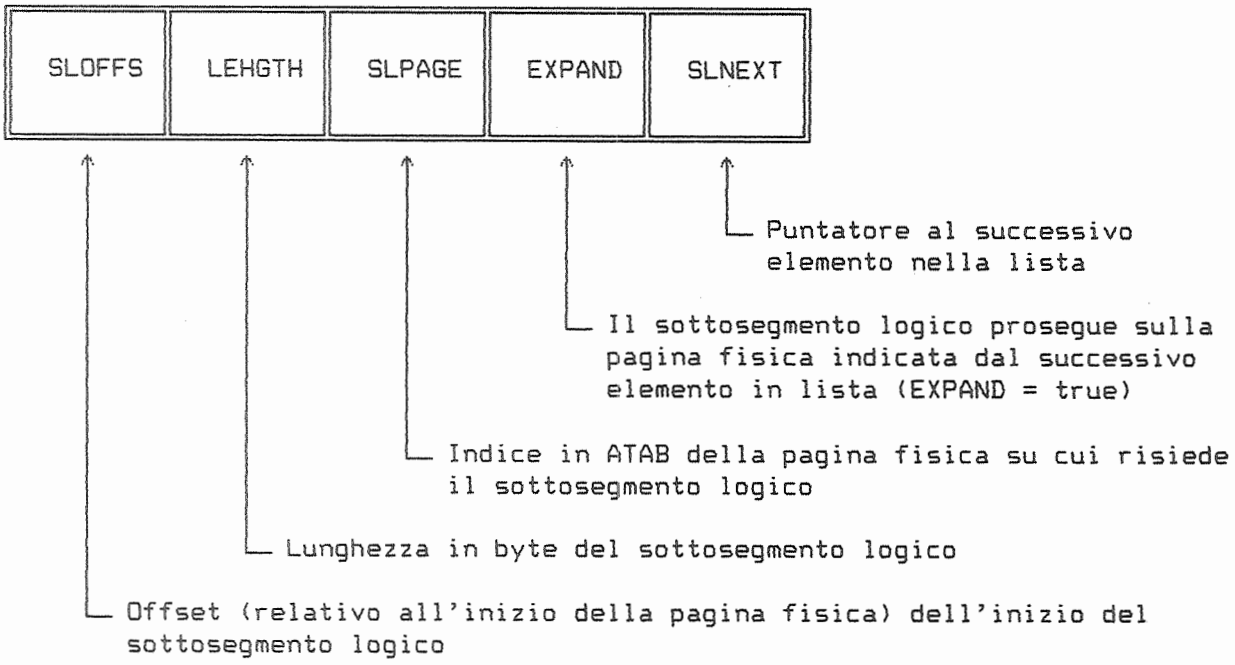


Fig. 13

Sub-segment list: SLIST



Tag list: TLIST

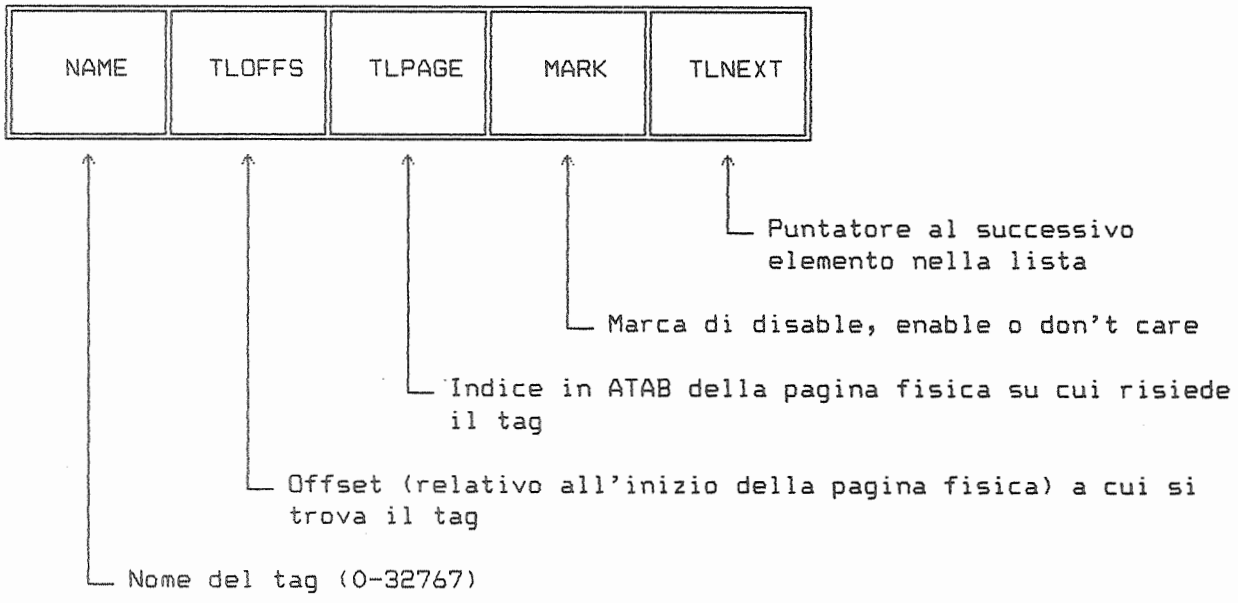


Fig. 14

Per motivi legati alla tecnica di allocazione di memoria utilizzata dal sistema operativo ospite le pagine eventualmente disallocate non vengono restituite al sistema bensi' marcate come riusabili (FREELEN = -1).

Disable Tag Table (DTAB): La tabella contiene, in ogni istante, le coppie di disable tag attive.

La presenza della coppia T1-T2 in tabella fa si' che i comandi grafici compresi tra T1 e T2 di tutti i segmenti di cui si effettua il traversal non siano inviati alla scheda grafica per l'esecuzione.

Il comando enable(T1-T2) fa si' che la coppia T1-T2 sia cancellata dalla tabella e che la tabella stessa sia ricompattata.

Subsegment List (SLIST): Come gia' accennato in precedenza la fase di esecuzione di una struttura deve evitare l'interpretazione dei comandi memorizzati nei singoli segmenti. E' necessario quindi identificare, in ogni oggetto, gruppi di comandi che non modificano il flusso sequenziale del traverser.

Nell'ipotesi di assenza di coppie DISABLE un oggetto e' eseguito sequenzialmente fino a quando questo non termina o non viene incontrato un comando callobj. Il concetto di sottosegmento e' legato a questa suddivisione logica di un oggetto in parti eseguibili sequenzialmente.

Un sottosegmento identifica il gruppo di comandi compreso tra:

[inizio segmento, fine segmento] oppure
[inizio segmento, callobj] oppure
(callobj, callobj] oppure
(callobj, fine segmento]

La lista SLIST descrive i sottosegmenti logici in cui e' suddiviso un oggetto in termini di offset relativo all'inizio della pagina fisica su cui risiede (SLOFFS), lunghezza in bytes (LENGTH), indice della pagina fisica su cui risiede (SLPAGE) e indicatore (EXPAND = true) del fatto che il sottosegmento descritto non termina con callobj bensi' prosegue sulla pagina fisica indicata dal prossimo elemento in lista.

Gli elementi di SLIST resisi liberi per modifiche apportate all'oggetto cui appartenevano sono organizzati in una lista libera ancorata a MEMSL.

Per comodita', in caso di oggetto terminante con callobj la lista dei sottosegmenti termina con un elemento indicante come offset l'indirizzo del primo byte successivo

al comando callobj e lunghezza nulla.

Tag List (TLIST): I tag locali ad ogni segmento sono descritti alla TLIST associata all'oggetto. Ogni elemento della lista contiene il nome del tag (NAME), la sua posizione nell'oggetto in termini di offset e pagina (TLOFFS, TLPAGE) ed una marca che, al momento del traversal dell'oggetto, varra' DISABLE (sospensione traversal), ENABLE (ripresa traversal) oppure DON'T CARE a seconda delle coppie disable presenti in DTAB.

Gli elementi di TLIST liberati per cancellazione dell'oggetto cui appartenevano (il sistema non prevede la cancellazione di singoli tag) sono organizzati in lista libera ed ancorati al puntatore MEMTL.

Path List (PLIST): E' la lista dei figli di primo grado dell'oggetto. Ogni elemento in lista indica, in pratica, la posizione in termini di offset (PLOFFS) e pagina (PLPAGE) del comando callobj relativo nel segmento. Il campo INDEX e' un indice in STAB alla entrata relativa al segmento chiamato.

La lista libera degli elementi inutilizzati di PLIST e' ancorata a MEMPL.

La funzione di PLIST diverra' evidente quando si descrivera' il funzionamento dell'algoritmo di traversal delle strutture.

Gli elementi di SLIST, TLIST e PLIST sono, in ogni momento, ordinati secondo la sequenzialita' dei relativi elementi nell'oggetto.

La figura 15 illustra, con un esempio, le strutture dati sopra descritte. La situazione descritta e' relativa all'istante precedente al traversal di una struttura (i campi TAGFLAG dei due segmenti A e B presenti nel sistema sono infatti impostati al valore true).

Nel sistema sono definiti i due segmenti A e B descritti alle entrate 7 e 35 di STAB rispettivamente. L'oggetto A risiede su due pagine fisiche (P17 e P05) e consta di quattro sottosegmenti logici (il primo, il terzo ed il quarto sono dovuti alla presenza di comandi callobj o alla fine dell'oggetto; il secondo e' determinato dal fatto che in creazione si e' deciso di riservare un certo numero di byte liberi su ogni pagina (FREELEN) e quindi si e' interrotto il sottosegmento indicando (EXPAND = true) che la sua prosecuzione logica si trova nella pagina successiva). L'oggetto B risiede su una singola pagina fisica (P27) ed e' costituito da un solo sottosegmento.

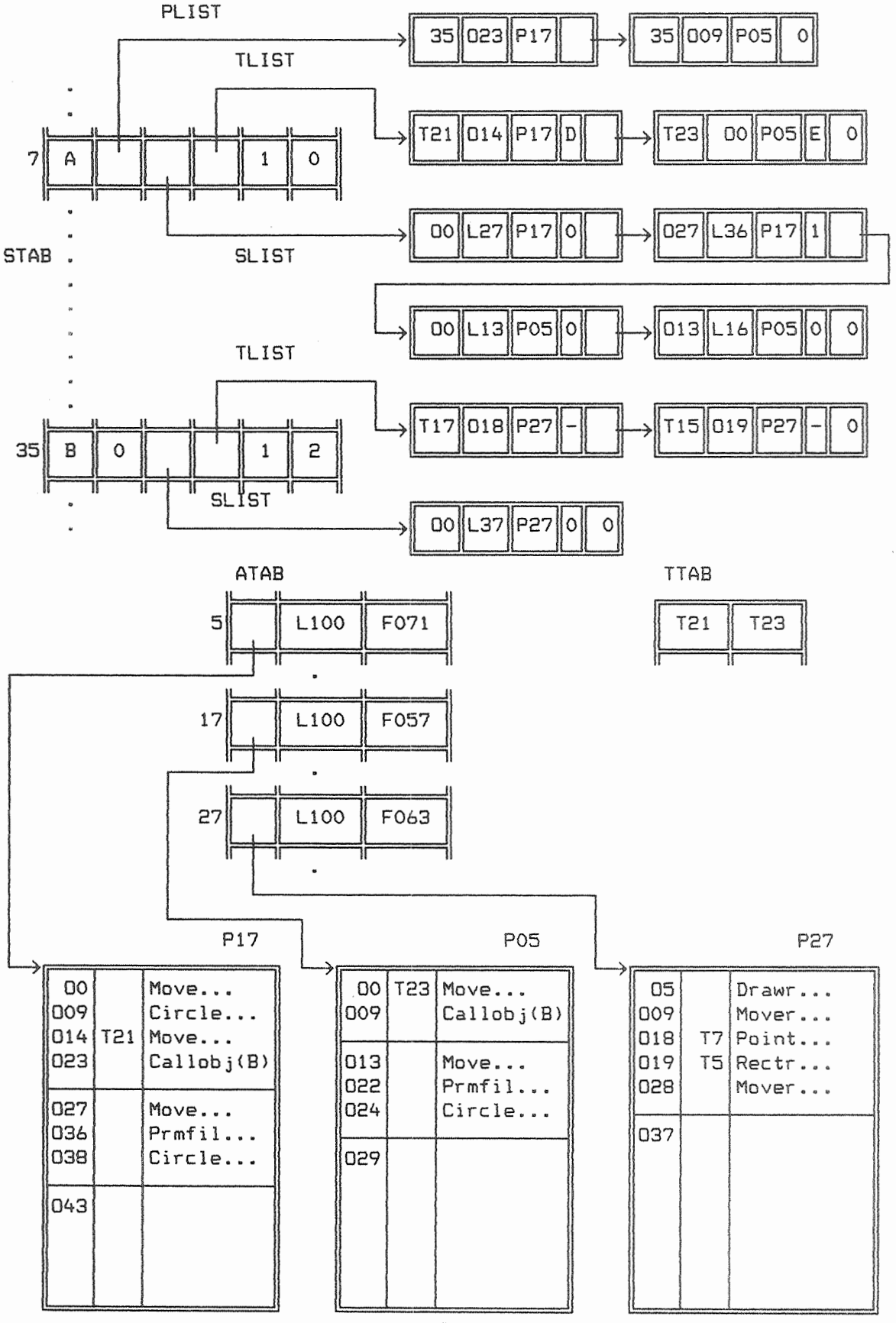

PLIST
TLIST
35 023 P17
35 009 P05 0
7 A 1 0
T21 014 P17 D
T23 00 P05 E 0
STAB
00 L27 P17 0
027 L36 P17 1
SLIST
00 L13 P05 0
013 L16 P05 0 0
TLIST
T17 018 P27 -
T15 019 P27 - 0
35 B 0 1 2
SLIST
00 L37 P27 0 0
ATAB
TTAB
5 L100 F071
T21 T23
17 L100 F057
27 L100 F063
P17
P05
P27
00 Move...
009 Circle...
014 T21 Move...
023 Callobj(B)
027 Move...
036 Prmfil...
038 Circle...
043
00 T23 Move...
009 Callobj(B)
013 Move...
022 Prmfil...
024 Circle...
029
05 Drawr...
009 Mover...
018 T7 Point...
019 T5 Rectr...
028 Mover...
037

Fig. 15

Il segmento A contiene due tag al suo interno (T21 e T23) che, per il traversal corrente, sono marcati con DISABLE ed ENABLE rispettivamente in quanto la stessa coppia e' presente in DTAB. I tag di B (T7 e T5) non portano alcuna marca (DON'T CARE).

L'oggetto A contiene due callobj all'oggetto B quindi la sua descrizione in STAB prevede anche una PLIST di due elementi (i cui campi INDEX puntano entrambi alla entrata 35 di STAB) testimoniante questo fatto.

La tabella ATAB descrive le caratteristiche delle pagine fisiche attualmente in uso riportandone lunghezza complessiva e byte ancora utilizzabili.

Il traversal della struttura A, nella situazione corrente, porta all'invio alla scheda grafica dei comandi:

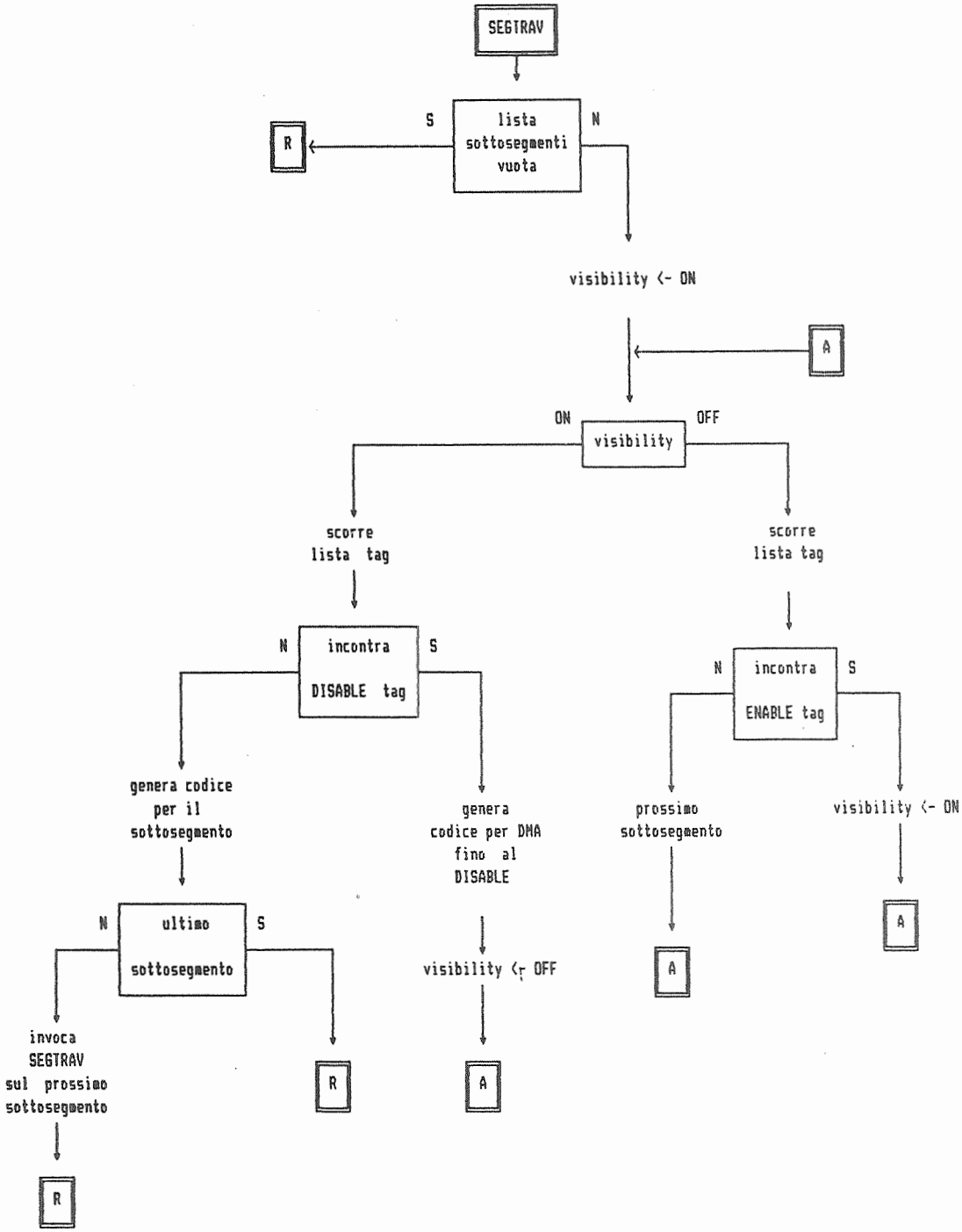
```
Move...
Circle...
Move...
(Callobj(B))
     Drawr...
     Mover...
     Point...
     Rectr...
     Mover...
Move...
Prmfil...
Circle...
```

evitando quindi di eseguire il traversal per i comandi:

```
Move...
(Callobj(B))
     Drawr...
     Mover...
     Point...
     Rectr...
     Mover...
Move...
Prmfil...
Circle...
```

compresi tra i tag T21 e T23.

Il diagramma seguente (Fig. 16) schematizza la logica di funzionamento dell'algoritmo di traversal di un segmento.


SEGTRAV
lista sottosegmenti vuota
S
N
R
visibility <- ON
A
ON
visibility
OFF
scorre lista tag
scorre lista tag
incontra DISABLE tag
N
S
incontra ENABLE tag
N
S
genera codice per il sottosegmento
genera codice per DMA fino al DISABLE
prossimo sottosegmento
visibility <- ON
ultimo sottosegmento
N
S
visibility <- OFF
A
A
invoca SEGTRAV sul prossimo sottosegmento
R
R
A
R

4. LA REALIZZAZIONE

Lo schema di figura 17 mostra l'organizzazione del sistema in termini dei moduli separati e comunicanti in esso presenti.

INTERF: rappresenta il modulo di interfaccia tra il programma applicativo (l'utente) ed il sistema. E' una libreria di funzioni di controllo e grafiche direttamente richiamabili dall'utente. Per ogni funzione il programma si limita a:

(a) controllare che il comando sia stato lanciato in uno stato ammissibile;

(b) convertire il comando in un formato opportuno (per i comandi grafici la forma adottata e' direttamente quella comprensibile dal processore grafico);

(c) attivare il modulo di trattamento del comando, dipendentemente dallo stato interno del sistema.

La figura 18 mostra le funzioni di controllo realizzate ed il modulo attivato in dipendenza dello stato.

L'analogo per le funzioni grafiche e' mostrato in figura 19. E' opportuno ricordare che le funzioni grafiche, da un punto di vista del sistema, sono tutte trattate allo stesso modo ed indicate con il termine generic.

I moduli di sistema (quelli con cui INTERF colloquia) sono:

OUTDMA: nella nostra realizzazione l'invio delle informazioni grafiche al processore grafico non e' effettuato da un dispositivo DMA. E' il modulo OUTDMA che riceve stringhe di byte direttamente da INTERF (comandi grafici da eseguire direttamente) o da SEGTRAV (sottosegmenti grafici o parti di essi da eseguire) e le invia al processore grafico.

GENTRAV: e' attivato da un comando callobj lanciato in stato di OPEN. GENTRAV controlla la validita' del traversal invocato dall'utente ed innesca la routine ricorsiva SEGTRAV per il traversal dei singoli segmenti che costituiscono la struttura da eseguire.

CREATOR: e' il modulo che sovrintende alla creazione di un segmento grafico gestendo quindi i comandi utente makeobj (transizione dallo stato OPEN allo stato CREATION), callobj, maketag, generic ed endobj.

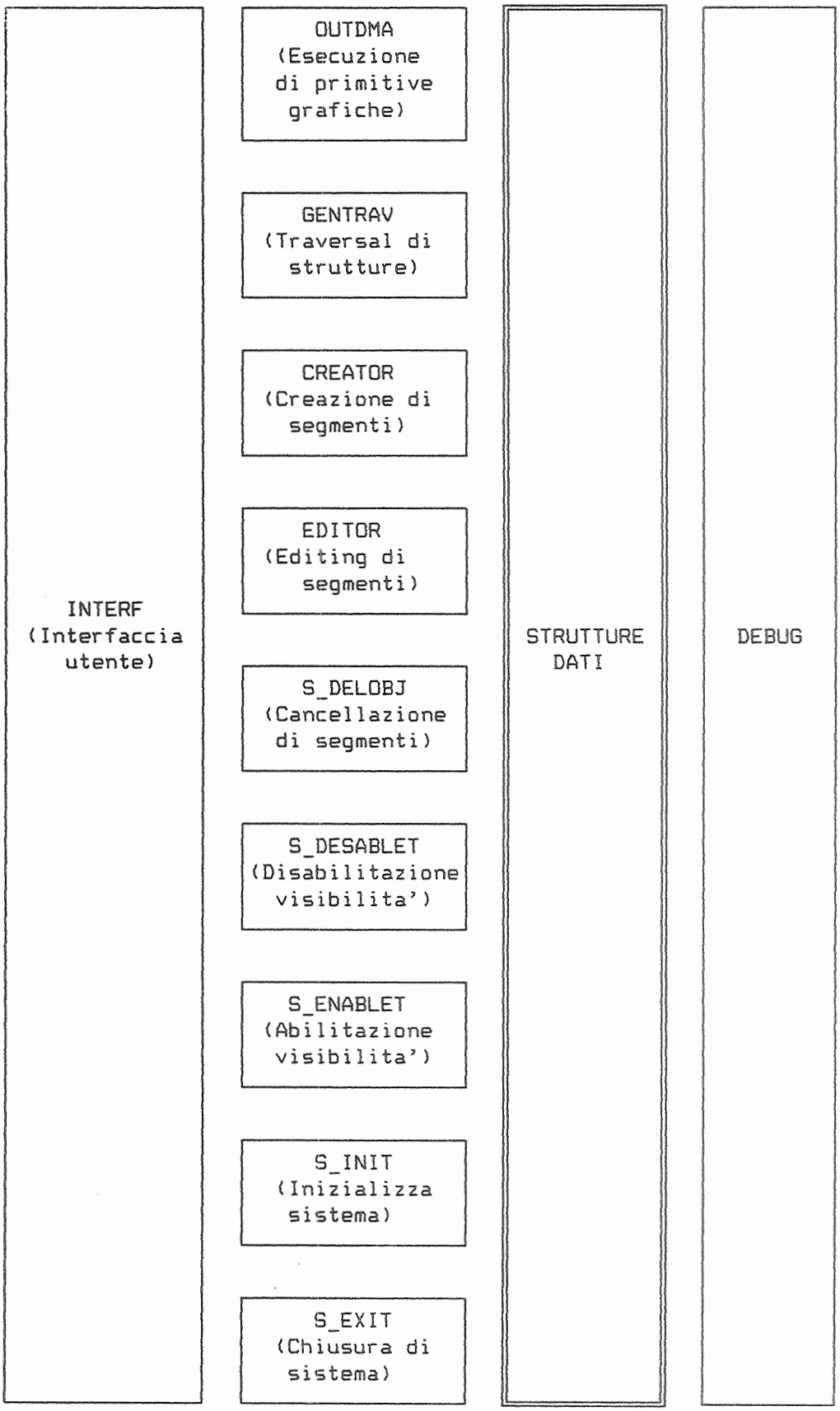

INTERF
(Interfaccia utente)
OUTDMA
(Esecuzione di primitive grafiche)
GENTRAV
(Traversal di strutture)
CREATOR
(Creazione di segmenti)
EDITOR
(Editing di segmenti)
S_DELOBJ
(Cancellazione di segmenti)
S_DESABLET
(Disabilitazione visibilita')
S_ENABLET
(Abilitazione visibilita')
S_INIT
(Inizializza sistema)
S_EXIT
(Chiusura di sistema)
STRUTTURE DATI
DEBUG

Fig. 17

Control Functions			
		--- open ------->	GENTRAV
	CALLOBJ	--- creation --->	CREATOR
		--- editing ---->	EDITOR
	DELETE INSERT REPLACE	--- editing ---->	EDITOR
	DELOBJ	--- open ------->	S_DELOBJ
	DESABLET	--- open ------->	S_DESABLET
	EDITOBJ	--- open ------->	EDITOR
	ENABLET	--- open ------->	S_ENABLET
	ENDOBJ	--- creation --->	CREATOR
	MAKETAG	--- editing ---->	EDITOR
	EXIT	--- open ------->	(n. i.)
	INIT	--- closed ----->	S_INIT
	MAKEOBJ	--- open ------->	CREATOR

Fig. 18

Graphics Functions	CIRCLE CLEARS COLOR DISTAN DRAW DRAWR DRAWR3 DRAW3 LINFUN LUT MDIDEN MDROTZ MOVE MOVER --- open -------> OUTDMA MOVER3 --- creation ---> CREATOR MOVE3 --- editing ----> EDITOR POINT POINT3 POLY POLYR POLYR3 POLY3 PRMFIL RECT RECTR TANGLE TEXT TJUST TSIZE VWPORT WINDOW

Fig. 19

EDITOR: sovrintende alle operazioni di editing di un segmento. In seguito al comando editobj lo stato di sistema diventa EDITING (sottostato INSERISCI), il segmento da editare e' aperto ed il puntatore corrente e' posto alla testa del segmento. In EDITING il modulo gestisce le funzioni generic, delete, callobj, maketag, insert, replace ed endobj lanciate dall'utente.

S_DELOBJ: e' attivato dal comando delobj e gestisce la cancellazione dei segmenti grafici.

S_DESABLET, S_ENABLET: sono attivati dai comandi disable ed enable rispettivamente. Gestiscono l'inserimento e la cancellazione delle copiie disable nella disable tag table.

S_INIT, S_EXIT: i moduli sono preposti alla inizializzazione e chiusura del sistema rispettivamente. In pratica si limitano ad inviare (tramite OUTDMA) al processore grafico informazioni di controllo per la inizializzazione e la chiusura.

DEBUG: trattandosi di una realizzazione sperimentale il sistema e' dotato di un modulo di debugging. Il modulo, che ha accesso alle strutture dati, prevede funzioni per la visualizzazione del contenuto delle tabelle, liste e pagine fisiche presenti nel sistema. DEBUG e' corredato inoltre di una procedura (SEGREAD) che interpreta il contenuto di un segmento grafico e ne visualizza una descrizione "ad alto livello".

5. I MODULI DEL SISTEMA

Le pagine seguenti contengono una descrizione delle funzioni presenti nel sistema. Ciascuna descrizione presenta informazioni sui parametri in ingresso ed in uscita e alcune note sul modo di operare e sullo scopo della funzione stessa.

Le funzioni sono organizzate in ordine alfabetico e, quando la complessita' lo richieda, sono corredate da opportuni disegni esemplificativi.

Le funzioni appartenenti ai moduli INTERF e DEBUG non sono commentate: ne viene soltanto fornito un listato in appendice.

Procedura ALLOPAG

/in/ dim: dimensione della pagina da allocare
/out/ indice del descrittore (in ADDTAB) della pagina allocata. -1 se non c'e' spazio in ADDTAB.

La procedura ALLOPAG ha la funzione di allocare una pagina fisica, della dimensione - in byte - specificata dal parametro in ingresso.
Inizializza il descrittore della pagina allocata.
La procedura riutilizza le pagine gia' allocate e logicamente rilasciate (campo FREELEN del descrittore uguale a -1) quando la dimensione coincide con quella specificata.
Restituisce l'indice nella tabella ADDTAB al descrittore della pagina allocata.
Restituisce -1 se la tabella ADDTAB e' piena.

Procedura ALLOPL

/in/ pun: puntatore all' elemento in coda al quale fare l' inserzione
/out/ puntatore all' elemento inserito

La procedura ALLOPL alloca un nuovo elemento nella lista dei figli di primo grado; se la lista libera corrispondente non e' vuota, l' elemento viene prelevato da questa.
L' elemento allocato viene inserito dopo quello puntato dal parametro in ingresso.
Restituisce il puntatore all' elemento inserito.

Procedura ALLOSL

/in/ pun: puntatore all' elemento in coda al quale fare l' inserzione
/out/ puntatore all' elemento inserito

La procedura ALLOSL alloca un nuovo elemento nella lista dei sottosegmenti; se la lista libera corrispondente non e' vuota, l' elemento viene prelevato da questa.
L' elemento allocato viene inserito dopo quello puntato dal parametro in ingresso.
Restituisce il puntatore all' elemento inserito.

Procedura ALLOTL

/in/ pun: puntatore all' elemento in coda al quale fare l'

inserzione
/out/ puntatore all' elemento inserito

La procedura ALLOTL alloca un nuovo elemento nella lista dei tag; se la lista libera corrispondente non e' vuota, l' elemento viene prelevato da questa.
L' elemento allocato viene inserito dopo quello puntato dal parametro in ingresso.
Restituisce il puntatore all' elemento inserito.

Procedura BTOI

/in/ bytes: puntatore alla coppia di byte da convertire
/out/ risultato della conversione

La funzione BTOI trasforma la coppia di byte in ingresso in un intero. I byte in ingresso sono nella forma:

byte[0] = low
byte[1] = high

Procedura CHECKTAG

/in/ ind: indice del segmento in SEGTAB
/out/ nessuno

La funzione CHECKTAG controlla la presenza di disabilitazioni e abilitazioni di visibilita' dei tag per il segmento specificato in ingresso; controlla il bilanciamento delle coppie disabilitazione-abilitazione e inizializza ai valori corrispondenti il campo MARK nella lista dei tag.

Procedura COPYPL

/in/ pun: puntatore sorgente
pun1: puntatore destinazione
/out/ nessuno

La funzione COPYPL copia l' elemento puntato dal primo parametro nell' elemento puntato dal secondo parametro.
Il campo "puntatore al successivo" dell' elemento destinazione viene lasciato inalterato.
La funzione lavora sulla lista dei figli di primo grado.

Procedura COPYSL

/in/ pun: puntatore sorgente
pun1: puntatore destinazione
/out/ nessuno

La funzione COPYSL copia l' elemento puntato dal primo parametro nell' elemento puntato dal secondo parametro.
Il campo "puntatore al successivo" dell' elemento destinazione viene lasciato inalterato.
La funzione lavora sulla lista dei sottosegmenti.

Procedura COPYTL

/in/ pun: puntatore sorgente
pun1: puntatore destinazione
/out/ nessuno

La funzione COPYTL copia l' elemento puntato dal primo parametro nell' elemento puntato dal secondo parametro.
Il campo "puntatore al successivo" dell' elemento destinazione viene lasciato inalterato.
La funzione lavora sulla lista dei tag.

Procedura CREASEG

/in/ namseg: nome del segmento da creare
/out/ indice del descrittore del segmento nella tabella dei segmenti (SEGTAB); -1 se la tabella dei segmenti e' piena

La procedura CREASEG ricerca un descrittore vuoto (cioe' con il nome nullo) nella tabella dei segmenti: se lo trova, inserisce il nome specificato in ingresso, inizializza gli altri campi e restituisce l' indice al descrittore; se non esiste un descrittore vuoto nella tabella dei segmenti, restituisce -1.

Procedura CREATOR

/in/ cmd: codice comando
lng: lunghezza stringa comando
cmdstr: stringa comando
/out/ reply code

La funzione CREATOR elabora il codice operativo in ingresso e compie le seguenti funzioni:

- Aggiorna le strutture dati, nel caso che il codice operativo corrisponda a MAKEOBJ, CALLOBJ, GENERIC E MAKETAG.

- Memorizza nella pagina fisica la stringa in ingresso contenente il comando in formato scheda, nel caso che il codice operativo corrisponda a CALLOBJ o GENERIC.

- Effettua un cambiamento di stato, nel caso che il codice operativo corrisponda a ENDOBJ.

- Restituisce un codice di completamento della funzione (OK oppure codice di errore).

Procedura EDITOR

/in/ cmd: codice comando
lng: lunghezza comando
cmdstr: stringa comando
/out/ reply code

E' il modulo di gestione di tutte le funzioni di editing di un segmento. Data la complessita' della procedura e' opportuno analizzare il suo comportamento in funzione dei diversi comandi in ingresso.

editobj: se il segmento da editare non e' stato ancora creato restituisce un codice di errore altrimenti entra nello stato di EDITING, posiziona l'offset alla testa del segmento e si predispone all'inserimento di nuovi comandi grafici (sottostato INSERISCI).

generic: se il sottostato e' RIMPIAZZA vengono effettuati gli opportuni controlli che il comando da rimpiazzare sia della stessa lunghezza e natura (generic) del comando in ingresso e, se superati, si effettua il rimpiazzamento. Se il sottostato e' INSERISCI si tenta di inserire il comando nella pagina di accumulo (allocata se non esiste); se l'aggiunta in pda del nuovo comando dovesse portare ad un trabocco della freearea corrente la pda e' svuotata (vedi procedura LONG_SPLIT) ed il comando inserito in pda.

delete: scaricata l'eventuale pda (mediante la procedura SHORT_SPLIT) presente viene spostato l'offset corrente al punto di inizio della cancellazione. Dopo gli opportuni controlli ha inizio il ciclo di cancellazione. La cancellazione fisica dei comandi avviene soltanto in occorrenza di uno dei seguenti eventi:
(a) scorrendo le istruzioni da cancellare si e' pervenuti alla fine di un sottosegmento;

(b) termina il ciclo di cancellazione.

callobj: ci si comporta come per l'inserimento (o il rimpiazzamento) di comandi generic ad eccezione del fatto che la pda contenente il comando (ed eventuali suoi predecessori) viene immediatamente svuotata.

maketag: si scarica l'eventuale pagina di accumulo allocata e si inserisce il tag in Tag List (se gia' non esiste).

insert: si scarica l'eventuale area di accumulo e si sposta l'offset corrente al punto indicato. Il sottostato diventa (o rimane) INSERISCI.

replace: si scarica l'eventuale pda e si sposta l'offset corrente al punto indicato. Il sottostato diventa (o rimane) RIMPIAZZA.

endobj: si scarica l'eventuale pda all'offset corrente e si ritorna allo stato OPEN.

Procedura ESISTSEG

/in/ name: nome del segmento
/out/ indice del segmento nella tabella dei segmenti, se trovato -1 altrimenti

La funzione controlla l' esistenza, nella tabella dei segmenti, del segmento specificato in ingresso.

Procedura ESISTTAG

/in/ name: nome del tag da inserire
/out/ reply code

Controlla che il tag da inserire non esista gia' nella Tag List del segmento.

Procedura FILLPL

/in/ pun: puntatore all' elemento da riempire
ind, offs, page: valori da inserire nell' elemento
/out/ nessuno

La funzione assegna i valori in ingresso ai campi dell' elemento specificato.
La funzione lavora sugli elementi della lista dei figli di primo grado.

Procedura FILLSL

/in/ pun: puntatore all' elemento da riempire
offs, len, page, exp: valori da inserire nell' elemento
/out/ nessuno

La funzione assegna i valori in ingresso ai campi dell' elemento specificato.
La funzione lavora sugli elementi della lista dei sottosegmenti.

Procedura FILLTL

/in/ pun: puntatore all' elemento da riempire
nam, offs, page, mar: valori da inserire nell' elemento
/out/ nessuno

La funzione assegna i valori in ingresso ai campi dell' elemento specificato.
La funzione lavora sugli elementi della lista dei tag.

Procedura FREEPAG

/in/ indpag: indice della pagina da rilasciare
/out/ nessuno

La funzione effettua il rilascio logico (-1 nel campo FREELEN della corrispondente entry nella tabella delle pagine) della pagina specificata in ingresso.

Procedura FREEPL

/in/ pun: puntatore all' elemento da disallocare
/out/ puntatore all' elemento successivo a quello disallocato

La funzione toglie un elemento dalla lista dei figli di primo grado e lo inserisce in testa alla lista libera corrispondente.

Procedura FREESL

/in/ pun: puntatore all' elemento da disallocare
/out/ puntatore all' elemento successivo a quello disallocato

La funzione toglie un elemento dalla lista dei sottosegmenti e lo inserisce in testa alla lista libera corrispondente.

Procedura FREETL

/in/ pun: puntatore all' elemento da disallocare
/out/ puntatore all' elemento successivo a quello disallocato

La funzione toglie un elemento dalla lista dei tag e lo inserisce in testa alla lista libera corrispondente.

Procedura GENTRAV

/in/ name: nome del segmento grafico da attraversare
/out/ codice di errore

La funzione scandisce la tabella dei segmenti alla ricerca del descrittore del segmento il cui nome e' specificato come parametro d' ingresso.
Se il segmento non esiste la funzione restituisce un codice d' errore (ERTRAV).
Se il segmento esiste la funzione chiama il traverser (segtrav) e inizializza (a zero) i tagflag sui descrittori nella tabella dei segmenti.

Procedura ITOB

/in/ integer: valore da trasformare
bytes: puntatore alla stringa risultato
/out/ puntatore alla coppia di byte contenenti il valore trasformato

La funzione trasforma un intero in una coppia di byte, nella forma:

byte[0] = low

byte[1] = high

Procedura LONG_SPLIT

/in/ pagea: indice in address table della pagina di accumulo;
offseta: offset corrente all'interno della pagina di accumulo.
/out/ restituisce un valore compreso tra 1 e 8 indicante il tipo di situazione in cui e' avvenuto lo split lungo.

La procedura e' invocata da EDITOR (sottostato INSERISCI) quando (a) l'inserimento del comando GENERIC in ingresso (e degli eventuali comandi GENERIC precedentemente inseriti nella pda) provocherebbe esubero oltre la dimensione della pagina corrente e (b) l'inserimento del comando CALLOBJ in ingresso provoca esubero oltre la dimensione della pagina corrente.

La procedura esegue lo split lungo come se nella pda fossero presenti una o piu' istruzioni GENERIC indipendentemente dal fatto che la pda possa contenere nessuna o alcune istruzioni GENERIC (caso a) oppure una istruzione CALLOBJ (caso b). E' cura del chiamante (EDITOR) sistemare opportunamente le cose.

La procedura distingue 8 possibili casi di split lungo rappresentati dagli schemi seguenti:

1: Inserimento in sottosegmento nullo su pagina vuota (e' il caso di un segmento creato vuoto o i cui comandi grafici sono stati interamente cancellati): la pagina corrente e' disallocata, ad essa viene sostituita la pda. I tag eventualmente presenti sono spostati alla testa di pda.

2: Inserimento in sottosegmento nullo su pagina non vuota (e' il caso del sottosegmento che chiude un segmento terminante con CALLOBJ): la pda e' inserita logicamente dopo la pagina corrente. I tag eventualmente presenti sono spostati alla testa di pda.

3: Inserimento in testa al primo sottosegmento di una pagina: la pda e' inserita logicamente prima della pagina corrente. I tag eventualmente presenti all'offset corrente sono spostati alla testa della pda.

4: Inserimento in testa ad un sottosegmento interno alla pagina corrente: il sottosegmento corrente (ed i suoi eventuali successori di pagina) e' splittato su una nuova pagina. Pda e' inserita tra la pagina corrente e

quella appena allocata. I tag eventualmente presenti all'offset corrente sono spostati alla testa di pda; quelli seguenti sono opportunamente spostati sulla nuova pagina.

5: Inserimento nel corpo di un sottosegmento: la parte del sottosegmento corrente seguente l'offset (e gli eventuali sottosegmenti successori di pagina) e' splittata su una nuova pagina. La testa del sottosegmento corrente da luogo ad un nuovo sottosegmento. Pda e' inserita tra la pagina corrente e la nuova. I tag eventualmente presenti all'offset corrente sono spostati alla testa di pda; quelli seguenti sono opportunamente spostati sulla nuova pagina.

6: Inserimento in fondo ad un sottosegmento che splitta logicamente: pda e' inserita dopo la pagina corrente ed il suo contenuto diventa la prosecuzione logica del sottosegmento corrente.

7: Inserimento in fondo ad un sottosegmento che termina con CALLOBJ ed e' l'ultimo della pagina corrente: pda e' inserita dopo la pagina corrente ed il suo contenuto diventa testa del sottosegmento successivo al corrente.

8: Inserimento alla fine dell'ultimo sottosegmento dell'oggetto (l'oggetto non termina con CALLOBJ): pda e' inserita dopo la pagina corrente ed il suo contenuto diviene l'ultimo sottosegmento dell'oggetto. I tag eventualmente presenti all'offset corrente sono spostati alla testa di pda.

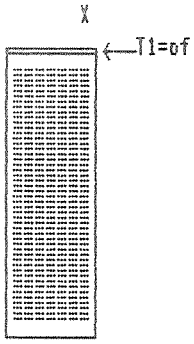
Nei casi 6 e 7 non si considerano i tag presenti a fine pagina in quanto si assume che gli eventuali tag si trovino sempre alla testa della pagina successiva. Il caso 8 non prevede il caso di oggetto terminante con CALLOBJ perche' questa condizione e' coperta dal caso 2.

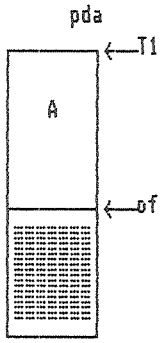
Procedura MOVSTR

```
/in/   page:   indice di pagina
       offset: offset nella pagina
       lng:    lunghezza della stringa da trasferire
       str:    stringa da trasferire
/out/  nessuno
```

La funzione inserisce nella pagina fisica specificata la stringa da trasferire per una lunghezza e a partire dall' offset specificati in ingresso.
La funzione non aggiorna l' offset.

Caso 1

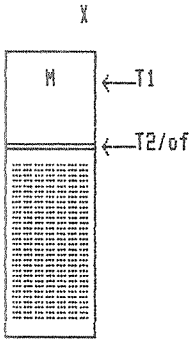


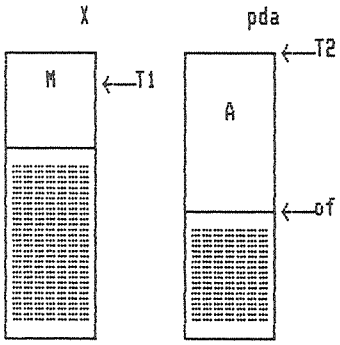


O	L	P	E
a(M)	0	X	0	0

O	L	P	E
a(M)	l(A)	pda	0	0

Caso 2





O	L	P	E
a(M)	l(M)	X	0

→

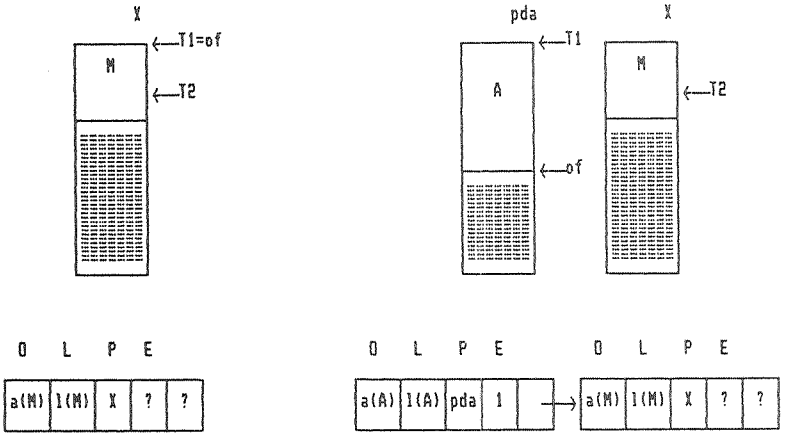
O	L	P	E
a(N)	0	X	0	0

O	L	P	E
a(M)	l(M)	X	0

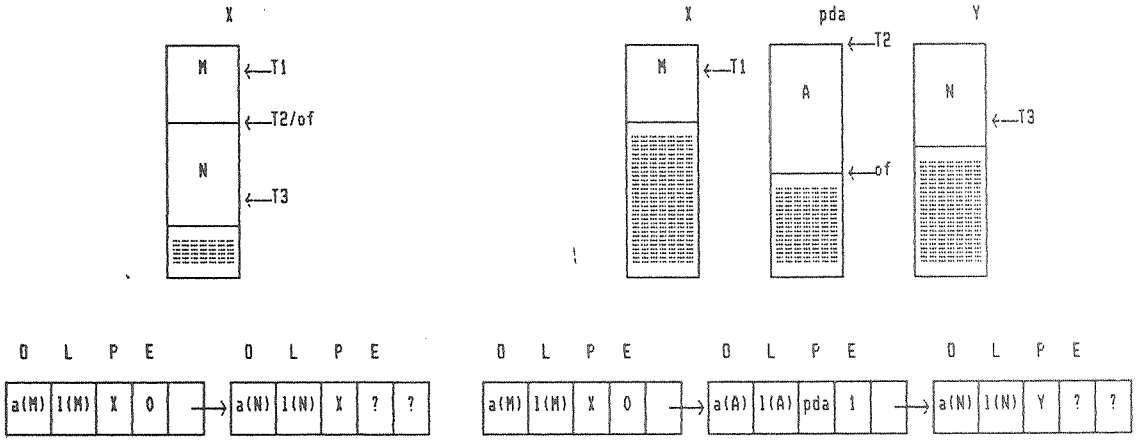
→

O	L	P	E
a(A)	l(A)	pda	0	0

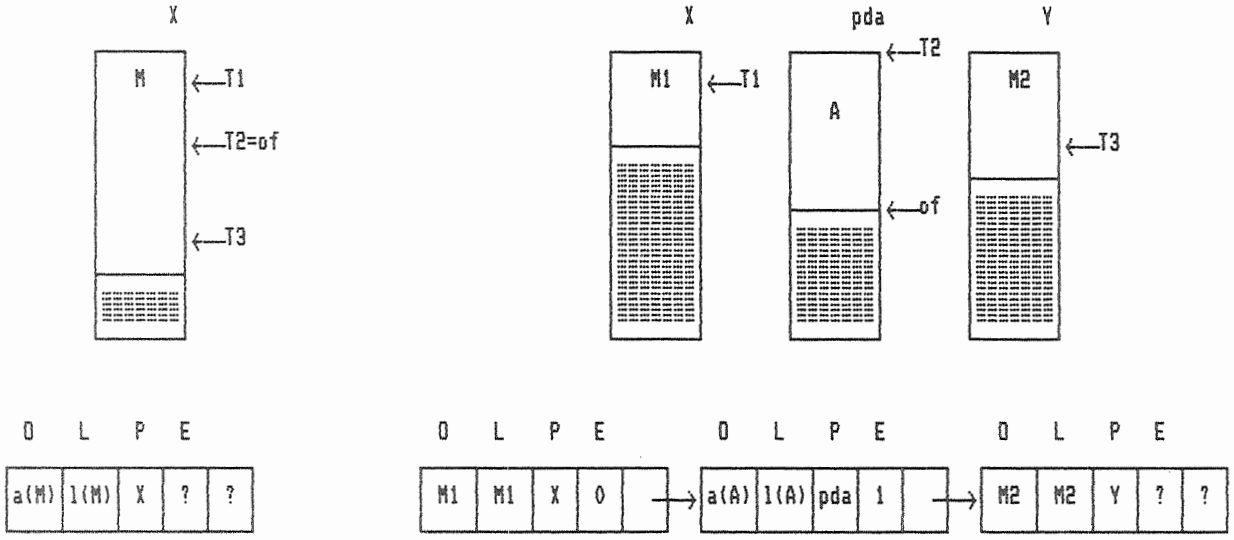
Caso 3


X
T1=of
M
T2
pda
T1
A
of
X
M
T2
O L P E
a(M) l(M) X ? ?
O L P E
a(A) l(A) pda 1
O L P E
a(M) l(M) X ? ?

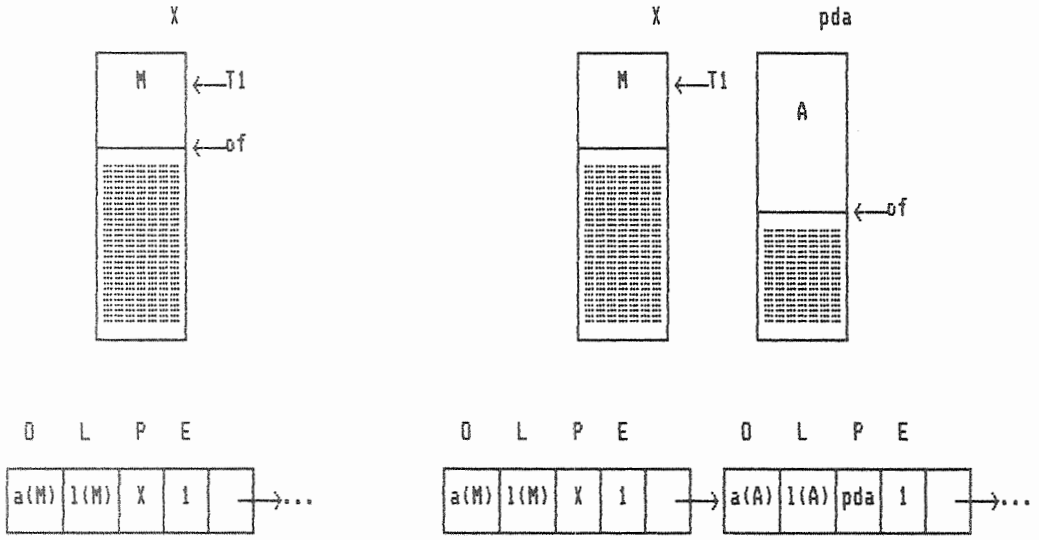
Caso 4


X
M
T1
T2/of
N
T3
X
pda
Y
M
T1
T2
A
N
T3
of
O L P E
a(M) l(M) X 0
O L P E
a(N) l(N) X ? ?
O L P E
a(M) l(M) X 0
O L P E
a(A) l(A) pda 1
O L P E
a(N) l(N) Y ? ?

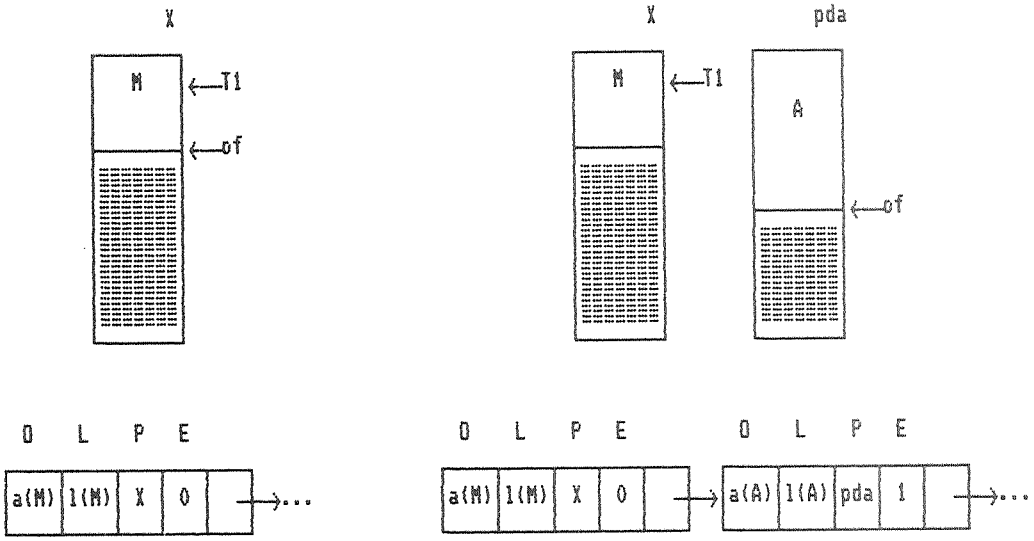
Caso 5


X
M
T1
T2=of
T3
X
pda
Y
M1
T1
A
T2
of
M2
T3
O L P E
a(M) l(M) X ? ?
O L P E
M1 M1 X 0
O L P E
a(A) l(A) pda 1
O L P E
M2 M2 Y ? ?

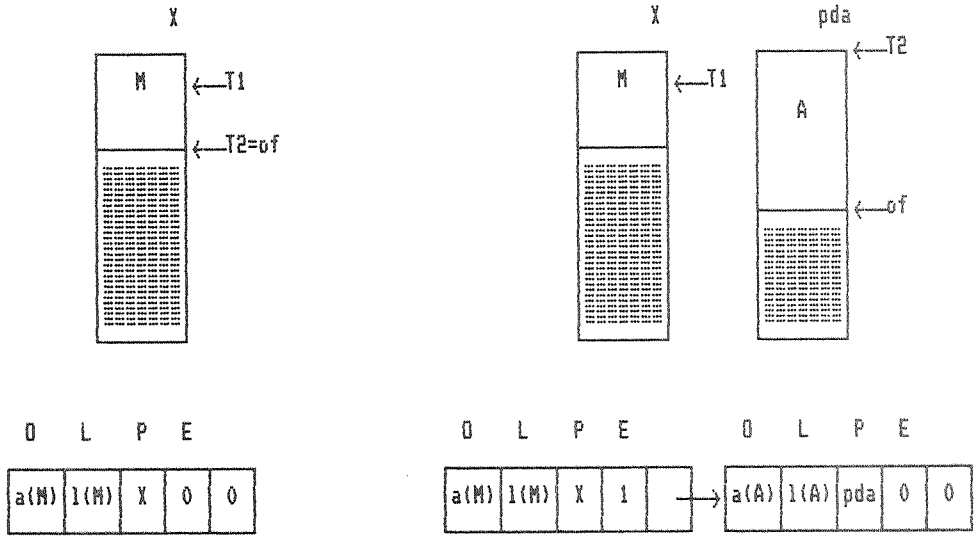
Caso 6


X
M
T1
of
X
pda
M
T1
A
of
O L P E
a(M) l(M) X 1 ...
O L P E
a(M) l(M) X 1
O L P E
a(A) l(A) pda 1 ...

Caso 7


X
M
T1
of
X
pda
M
T1
A
of
O L P E
a(M) l(M) X 0
O L P E
a(M) l(M) X 0
O L P E
a(A) l(A) pda 1

Caso 8


X
M
T1
T2=of
X
pda
M
T1
A
T2
of
O L P E
a(M) l(M) X 0 0
O L P E
a(M) l(M) X 1
O L P E
a(A) l(A) pda 0 0

Procedura OUTDMA

/in/ begin: indirizzo di inizio del trasferimento
length: lunghezza del trasferimento
/out/ nessuno

La funzione simula la presenza di un DMA grafico.

Procedura PATHMOVE

/in/ ind: indice in segment table del segmento corrente,
oldpag: indice in address table della pagina su cui risiedeva il comando CALLOBJ,
newpag: indice in address table della pagina su cui e' stato spostato il comando CALLOBJ,
source: offset a cui si trovava il comando CALLOBJ,
dest: offset relativo a cui attualmente si trova il comando CALLOBJ.
/out/ nessuno.

La procedura PATHMOVE, invocata da LONG_SPLIT, SHORT_SPLIT e PRESS, ha la funzione di aggiornare gli elementi di path list relativi a comandi CALLOBJ che hanno subito uno spostamento (per effetto, appunto, di uno split lungo, corto o di una cancellazione).

Procedura POSIX

/in/ tag: label rispetto cui si effettua il posizionamento,
disp: offset (relativo al tag precedente) del posizionamento.
/out/ assume un valore corrispondente al tipo di posizionamento effettuato oppure un codice di errore.

La procedura POSIX, invocata da DELETE (EDITOR), INSERT (EDITOR) e REPLACE (EDITOR), ha la funzione di posizionare il puntatore corrente (in termini di offset e pagina) all'interno del segmento corrente. Il posizionamento avviene dopo [disp] comandi grafici a partire dal [tag] specificato. Sono ammessi valori nulli per tag o disp. La procedura restituisce un codice di errore (-1) soltanto nel caso che la label [tag] specificata non sia presente nell'oggetto.

La procedura assume un valore compreso tra 1 e 8 per indicare la situazione raggiunta nel posizionamento:

1: posizionamento alla testa di un segmento vuoto;
2: posizionamento alla testa del sottosegmento vuoto di chiusura di un oggetto terminante con un comando CALLOBJ;

3: posizionamento alla testa di un sottosegmento che e' il primo della sua pagina;
4: posizionamento alla testa di un sottosegmento che non e' il primo della sua pagina;
5: posizionamento all'interno di un sottosegmento;
6: posizionamento alla fine dell'ultimo sottosegmento di una pagina (caso di sottosegmento che splitta sulla prossima pagina;
7: posizionamento alla fine dell'ultimo sottosegmento di una pagina (caso di sottosegmento terminante con comando CALLOBJ);
8: posizionamento alla fine dell'oggetto (l'oggetto non termina con un comando CALLOBJ).

Procedura PRESS

/in/ off1: offset, rispetto l'inizio della pagina fisica, del primo comando grafico da cancellare;
off2: offset, rispetto l'inizio della pagina fisica, del comando successivo all'ultimo da cancellare.
/out/ nessuno.

La procedura PRESS, invocata da DELETE (EDITOR), ha la funzione di cancellare il blocco di comandi grafici compresi tra [off1] e [off2] all'interno del sottosegmento corrente.
La procedura compatta il sottosegmento corrente e quindi gli eventuali sottosegmenti successori di pagina, aggiorna gli elementi di path list e di subsegment list eventualmente movimentati, sposta i tag agli indirizzi opportuni.

Procedura READCMD

/in/ offset: indirizzo, relativo alla pagina fisica, del comando da esaminare
/out/ rcmd: tipo della primitiva esaminata
lunghezza della primitiva

La funzione esamina il contenuto della pagina fisica corrente all' indirizzo specificato come parametro d' ingresso e restituisce il tipo di primitiva incontrata, che puo' essere GENERIC (per tutte i codici operativi corrispondenti alle primitive grafiche delle schede) o CALLOBJ (corrispondente al codice operativo PSEUDOC, interpretato dalle schede grafiche come un WAIT 0). E' previsto un codice errore per codici operativi non riconosciuti.
La funzione restituisce inoltre la lunghezza del comando grafico incontrato, che e' data dalla lunghezza della

corrispondente primitiva per le schede grafiche.

Procedura RICORS

/in/ p: indice, nella tabella dei segmenti, del segmento padre
f: indice, nella tabella dei segmenti, del segmento figlio
/out/ codice di risposta: -1 in caso di ricorsione, OK altrimenti

La funzione controlla che non esista un ciclo di chiamate che dal segmento figlio riporti al segmento padre, generando ricorsione.

Procedura RTOB

/in/ real: numero da trasformare
/out/ bytes: sequenza di byte contenente il risultato della trasformazione

Trasforma un numero in formato double real in una sequenza di quattro byte, nella forma:

```
byte[0] = low_int
byte[1] = high_int
byte[2] = low_fract
byte[3] = high_fract
```

Procedura SEARCHPL

/in/ ind: indice del segmento corrente nella tabella dei segmenti
offset: offset corrente nella pagina fisica corrente
/out/ codice di risposta:
0: se la lista dei figli e' vuota
1: inserimento dopo PL
-1: inserimento prima di PL

La funzione scorre la lista dei figli di primo grado e, se non e' vuota, identifica l' elemento sul quale effettuare l' inserzione di un nuovo elemento.
La funzione, nel caso di lista non vuota, aggiorna il list pointer (PL) a puntare all' elemento identificato.

Procedura SEARCHTL

/in/ ind: indice del segmento corrente nella tabella dei segmenti
offset: offset corrente nella pagina fisica corrente
/out/ codice di risposta:
0: se la lista dei tag e' vuota
1: inserimento dopo TL
-1: inserimento prima di TL

La funzione scorre la lista dei tag e, se non e' vuota, identifica l' elemento sul quale effettuare l' inserzione di un nuovo elemento.
La funzione, nel caso di lista non vuota, aggiorna il list pointer (TL) a puntare all' elemento identificato.

Procedura SEGTRAV

/in/ ind: indice del segmento da attraversare
/out/ nessuno

La funzione effettua l' attraversamento della struttura grafica, partendo dal segmento specificato in ingresso e inviando i comandi grafici (contenuti nelle pagine fisiche) alle schede grafiche.
La funzione gestisce la visibilita' delle parti grafiche comprese tra tag e tag: la variabile KMAX contiene il numero totale di coppie di tag a visibilita' disabilitata, che sono contenute nella tabella TAGTAB.

Procedura SHORT_SPLIT

/in/ pagea: pagina di accumulo
offseta: offset sulla pagina di accumulo
/out/ reply code

La funzione prevede gli stessi ingressi e restituisce gli stessi valori (corrispondenti alle stesse situazione) della procedura LONG_SPLIT. In questo caso non viene provocato uno spli su piu' pagine ma il contenuto di pda e' scaricato all'offset corrente della pagina fisica corrente.

Procedura SSEGMOVE

/in/ sll: puntatore all'elemento di Segment List da spostare
oldpag: pagina fisica di provenienza
newpag: pagina destinazione
dest: spostamento relativo dell'offset
/out/ nessuno

La procedura sposta un elemento di Sub Segment List da una pagina ad un'altra conseguentemente allo spostamento fisico del relativo sottosegmento.

Procedura STACCASL

/in/ nessuno
/out/ nessuno

La procedura ha la funzione di staccare l'elemento corrente di Sub Segment List.

Procedura S_DELOBJ

/in/ name: nome del segmento grafico da cancellare
/out/ codice d' errore

La funzione provvede alla cancellazione di un segmento grafico:

- cancella il segmento dalla tabella dei segmenti, azzerando il campo relativo al nome

- rilascia le pagine fisiche occupate dal segmento

- recupera gli elementi della lista dei figli di primo grado associata al segmento cancellato, reinserendoli in testa alla lista libera corrispondente

- recupera gli elementi della lista dei sottosegmenti associata al segmento cancellato, reinserendoli in testa alla lista libera corrispondente

- recupera gli elementi della lista dei tag associata al segmento cancellato, reinserendoli in testa alla lista libera corrispondente.

La funzione restituisce un codice di errore se il segmento specificato non esiste nella tabella dei segmenti.

Procedura S_DESABLET

/in/ tag1: tag da cui ha inizio la disabilitazione
tag2: tag su cui termina la disabilitazione
/out/ codice di errore

Inibizione della visibilita' delle strutture grafiche comprese tra la coppia di tag specificati in ingresso.

La funzione aggiorna la variabile KMAX (numero totale di coppie disabilitate) e la tabella (TAGTAB) che memorizza le coppie di tag che definiscono le sezioni non visibili: KMAX e TAGTAB vengono utilizzate dalla procedura SEGTRAV.

Se la tabella e' piena, la funzione restituisce un codice di errore.

Procedura S_ENABLET

/in/ tag1: tag da cui ha inizio la riabilitazione
tag2: tag su cui termina la riabilitazione
/out/ codice di errore

Riabilitazione della visibilita' - precedentemente disabilitata - delle strutture grafiche comprese tra i tag specificati in ingresso.

La funzione aggiorna la variabile KMAX (numero totale di coppie disabilitate) e la tabella TAGTAB che memorizza le coppie di tag che definiscono le sezioni non visibili: KMAX e TAGTAB vengono utilizzate dalla procedura SEGTRAV.

Se la visibilita' tra i tag specificati non e' stata in precedenza disabilitata, e quindi la coppia di tag non esiste in TAGATAB, la funzione restituisce un codice di errore.

Procedura S_EXIT

La procedura non e' stata implementata.

Procedura S_INIT

/in/ nessuno
/out/ nessuno

La funzione effettua l' inizializzazione dell' hardware grafico e dello stato del sistema grafico, che assume il valore OPEN.

Procedura TAGMOVE

/in/ ind: indice in Segment Table
oldpag: pagina di provenienza del tag
newpag: pagina di destinazione del tag
source: offset sulla pagina di provenienza
dest: offset relativo allo spostamento
type: tipo di spostamento (per tag all'offset corrente o

successivi
/out/ nessuno

La procedura viene utilizzata per lo spostamento di tag da una pagina fisica ad un'altra.

Le liste complete dei programmi in linguaggio "C" sono raccolte nella Nota Tecnica B4-54 (Ottobre '86) dell'Istituto di Elaborazione dell'Informazione.