

Consiglio Nazionale delle Ricerche

Interfaccia per un sistema VS2 sotto VM

P. Mogorovich

98

CNUCE

Divisione Servizio Elaborazione Dati

A cura di : Paolo Mogorovich
Copyright Gennaio 1976
by CNUCE - Pisa
Istituto del Consiglio Nazionale delle Ricerche

Presso questo Centro esiste un sistema IBM 370/168 su cui opera un sistema operativo "VM". Una delle macchine virtuali definite e' una macchina di nome simbolico "VS2" su cui gira un sistema operativo VS2/R1.7-HASP

Scopo di questa ricerca e' quello di fornire a VS2 un'interfaccia software, di modo che i files di stampa e perforazione prodotti, oltre ad essere chiusi automaticamente, possano essere inviati a qualsiasi altro Utente VM, senza interventi da parte dell'operatore.

La configurazione attuale e' formata da un sistema IFM 370/168 con 4 Mbyte di memoria centrale, due canali block multiplexor, due selector e due byte multiplexor.

Dischi, nastri e un tamburo sono connessi come indicato in fig.1.

Su questa macchina opera un sistema VM, PLC 16. (*)

A questa macchina sono collegati terminali di tipo start stop, e di tipo BSC. Poiche' il VM non gestisce i terminali BSC 2770 e 3780, e gestisce quelli tipo 2780 in maniera non efficiente, si fa uso di una macchina virtuale di nome RSCS1, gestita da un sistema operativo del tipo RSCS.

La macchina virtuale RSCS1 e' in grado di gestire anche terminali del tipo "intelligente" come IBM 1130, e ogni tipo della serie 360 e 370. Nel caso di terminali "intelligenti" su di essi gira un piccolo sistema operativo, generato usando una delle possibilita' di HASP.

In ambiente VM e' definita una macchina virtuale, di nome VS2, su cui gira un sistema operativo VS2/R1.7-HASP.

Ad essa sono dedicate due stringhe di dischi 3330: la prima consiste di quattro dischi 3330/1 di indirizzo da 250 a 253; la seconda di due dischi 3330/1 di indirizzo 258 e 259, e quattro dischi 3330/11 (da 25A a 25D).

(La seconda stringa e' "shared" con un'altro sistema IFM 370/158).

(*) Al 10 settembre 1975

SCHEMA DI CONNESSIONE LOGICA DEI SISTEMI
DI CALCOLO IBM 370/150; IBM 370/160 DEL CUCE

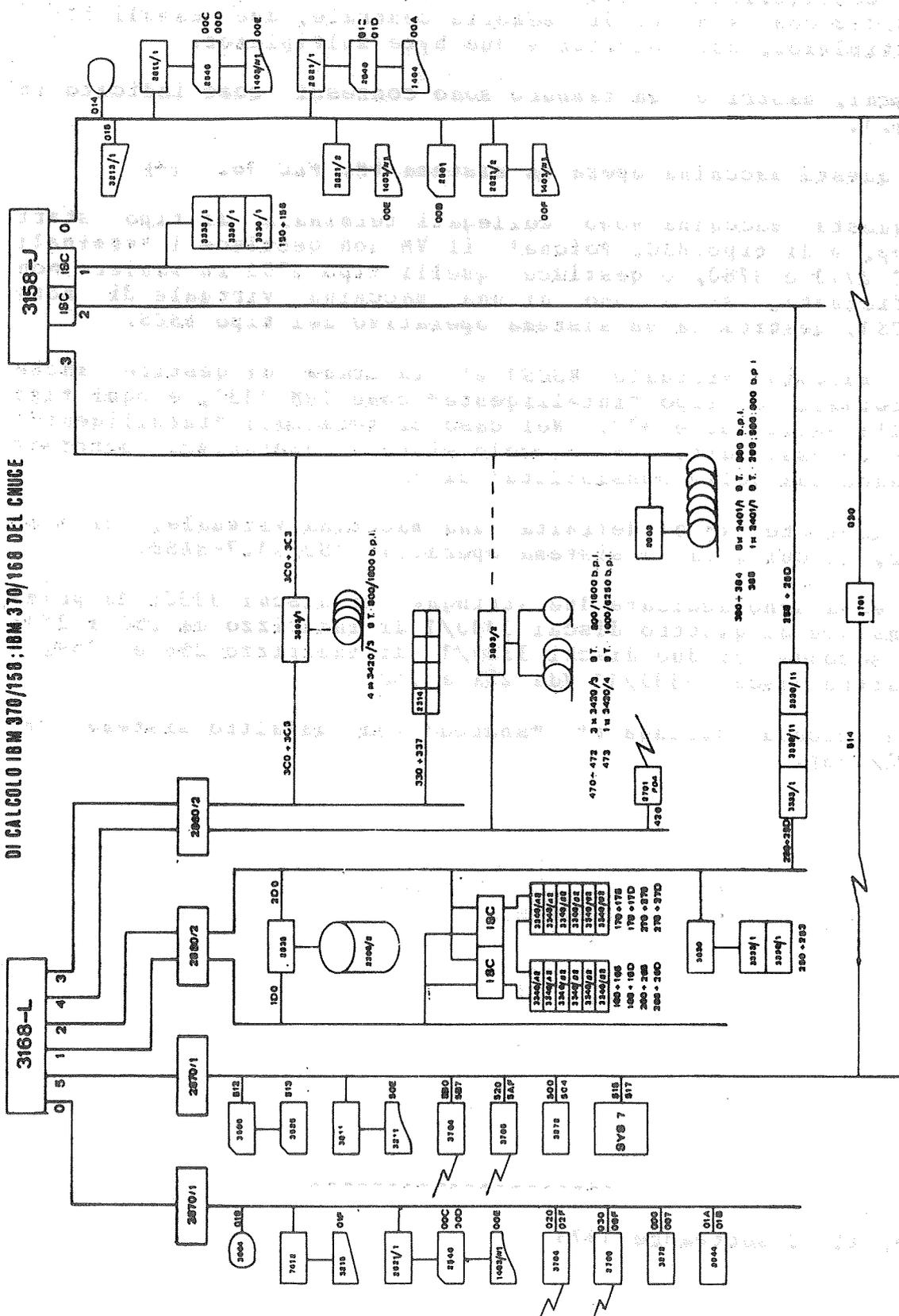


fig. 1

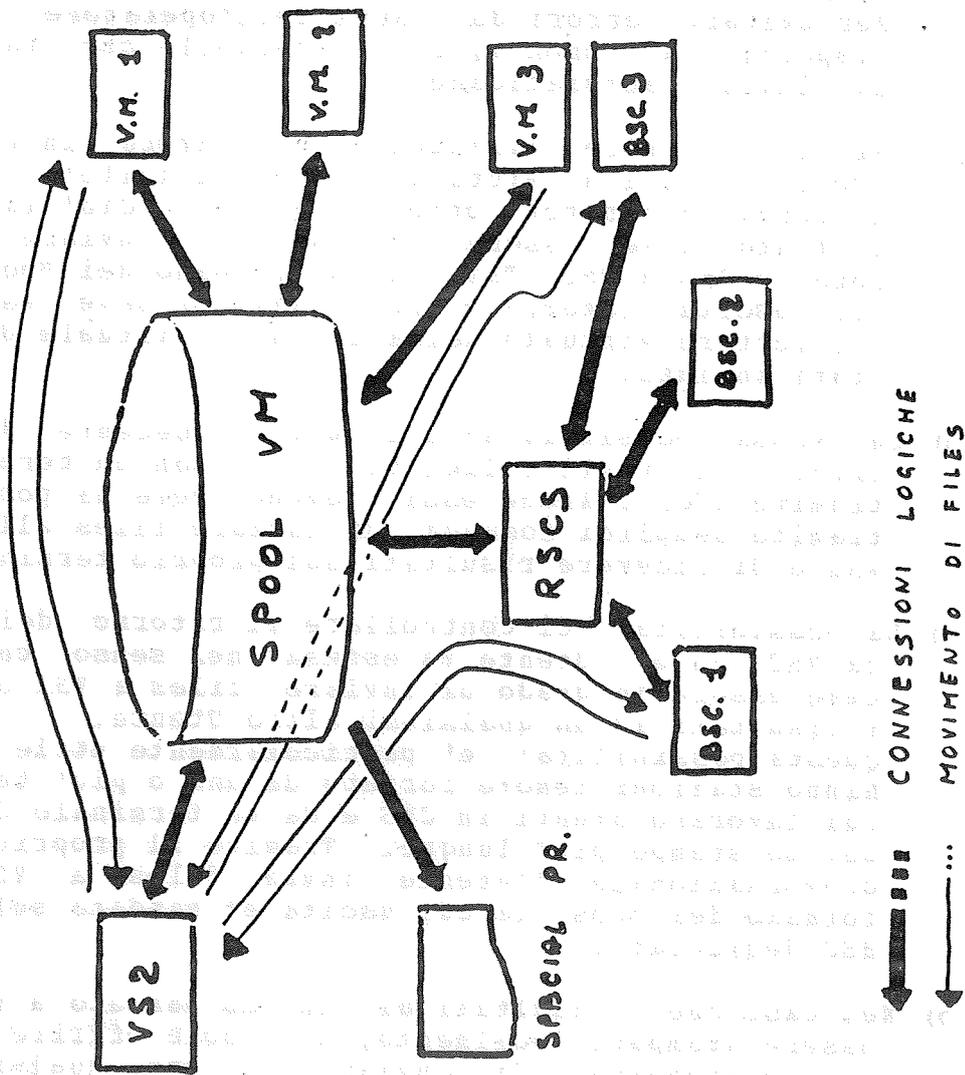


fig. 2

La fig. 2 mostra la connessione logica tra lo Spool del VM e alcuni differenti tipi di Utenti: VM1, VM2, VM3 rappresentano una serie di macchine virtuali gestite da un sistema operativo CMS; BSC1, BSC2, BSC3 rappresentano un qualsiasi terminale BSC (2770, 2780, 3780) connesso al VM tramite l'interfaccia di RSCS1.

Scopo di questa ricerca e' risolvere i seguenti problemi:

- 1) I files stampati e perforati da HASP, che opera sulla macchina virtuale VS2, per poter essere processati dal VM, hanno bisogno di essere chiusi ciascuno da un comando "CLOSE".
Per evitare errori da parte dell'operatore e per una risposta piu' pronta, e' desiderabile che tale comando sia inviato automaticamente.
- 2) Si vuol offrire all'Utente che lavora in CMS, sulla propria macchina virtuale, la possibilita' di accedere al servizio "batch" offerto da VS2. Cio' implica che l'Utente deve essere in grado di inviare files di formato CMS a VS2. Tali files, formano dei "job OS" e la loro uscita (stampa e perforazione) deve essere posta sul lettore virtuale della macchina virtuale da cui sono stati inviati.
- 3) La stessa possibilita' del punto precedente deve essere offerta ad Utenti collegati al VM con un terminale BSC, tramite RSCS1. Anche essi devono avere la possibilita', tramite semplici comandi, di inviare files alla macchina VS2 e di ricevere risultati sul proprio terminale.
- 4) La possibilita' di controllare il ritorno dei risultati da VS2 ad un Utente va estesa nel senso che chiunque deve essere in grado di inviare files a VS2 e far avere i risultati ad un qualsiasi altro Utente.
Questa possibilita' e' particolarmente utile quando si hanno stazioni remote formate da uno o piu' terminali su cui lavorano Utenti in CMS e da un terminale di tipo BSC per le stampe piu' lunghe. Tramite il proprio terminale conversazionale l'Utente invia files a VS2. Questi formano dei jobs, la cui uscita e' mandata sul terminale BSC desiderato.
- 5) Nel caso che i risultati di un job mandato a VS2 debbano essere stampati localmente, si vuole offrire all'Utente la possibilita' di inviarli su una qualsiasi delle stampatrici.

I punti sopra discussi pongono due problemi: il primo e' l'invio di files alla macchina virtuale VS2; il secondo e' il controllo dei files-risultati.

Il primo problema si risolve molto facilmente usando la possibilita' standard del VM.

Per inviare files da terminali del tipo BSC alla macchina virtuale VS2 e' necessario farli leggere dal lettore del terminale, dopo aver inserito in testa al pacco di schede una scheda ID col seguente formato:

```
col 1 2 3 4 5 6 7 8 8 10 11 12 13
   I D           V S 2
```

Per inviare files da terminale start-stop usando il sistema operativo CMS, alla macchina virtuale VS2 e' necessario inviare, in sequenza, i comandi:

```
SPOOL D TO VS2
PUNCH nome tipo
SPOOL D OFF
```

Per quanto riguarda il movimento dei files-risultati, si vede subito che solo la macchina virtuale VS2 e' in grado di controllare tale movimento, e che le informazioni necessarie per questo vanno prese dalle schede controllo del file che VS2 riceve, che, come abbiamo gia' detto, forma un job di tipo OS.

Dobbiamo distinguere tre possibilita':

- 1) -I risultati elaborati da VS2 vanno posti sul lettore virtuale di una macchina virtuale
- 2) -I risultati provenienti da VS2 vanno inviati ad un remoto di tipo BSC gestito dalla macchina virtuale RSCS1.
- 3) -I risultati provenienti da VS2 vanno inviati su una particolare stampante.

Dal punto di vista della macchina virtuale VS2 si soddisfano tutte queste richieste usando la possibilita' dei comandi di CP "TAG" e "SPOOL" nel modo seguente.

Se il file di stampa (perforazione) va posto sul lettore virtuale della macchina virtuale di nome VM1, e' necessario, dopo che JASP su VS2 ha stampato (perforato) l'ultima riga (scheda) del file, inviare i seguenti comandi di CP

```
SPOOL E(D) TO VM1
CLOSE E(D)
SPOOL E(D) OFF
```

Se il file va stampato (perforato) su un terminale del tipo

BSC (per esempio BSC1), dopo che HASP ha stampato (perforato) l'ultima riga (scheda) e' necessario inviare i seguenti comandi di CP

```
SPOOL E (D) TC BSCS1
TAG DEV E(D) TO BSC1
CLOSE E (D)
TAG DEV E(D)
SPOOL E (D) CFF
```

Per quanto riguarda il controllo dell'invio di files-risultati a stampanti diverse, queste sono abilitate a trattare files di alcune classi definite al momento dello "start" delle stampanti stesse.

Le classi per cui le stampanti sono abilitate, sono organizzate in modo tale che i file di tipo piu' comune (classe A) possono essere stampati indifferentemente su una stampante o su un'altra; esistono invece alcune classi definite univocamente per ciascuna stampante. Si vede cosi' che si puo' indirizzare una stampa su una particolare stampatrice semplicemente asseguando a quel file un'opportuna classe.

Se per esempio si vuole far stampare un file su una stampante abilitata univocamente per la classe 2 oltre che per altre classi, basta inviare, i seguenti comandi di CP, dopo che HASP sotto VS2 ha stampato l'ultima riga:

```
SPOOL E CLASS 2
CLOSE E
SPOOL E CLASS A
```

Abbiamo visto quali sono i comandi che vanno lanciati dalla macchina virtuale VS2 per rispettare le richieste dell'utente. Vediamo ora come tali richieste vanno formulate.

Senza aggiungere nuove schede controllo, il movimento dei files di stampa e perforazione si comanda codificando opportunamente la scheda JOB del job relativo.

Si sfrutta per questo il campo "programmer name". Esso e' lungo al massimo 20 caratteri e si trova dopo i parametri di account e prima di qualsiasi parola chiave racchiuso tra apici se contiene caratteri speciali diversi dal punto o blank.

Nel nostro caso esso va sempre chiuso tra apici.

Esso e' diviso in due sottocampi divisi da una virgola. Ogni

sottocampo e' a sua volta diviso in due parti: la parte "funzione" e la parte "indirizzo".

```
//NOME JOB (xxx,xxx,...),'$VM128,=REM62 ',xxxx...
```

fig.3

In figura vediamo un esempio di "programmer name".
Il '\$' e' la parte funzione del primo sottocampo e 'VM128'
la parte indirizzo; quindi il separatore. Poi il carattere
'=' come parte funzione del secondo sottocampo e 'REM62'
come parte indirizzo dello stesso sottocampo .

Il primo sottocampo serve ad indirizzare la stampa, il
secondo la perforazione.

Se la parte funzione e' un '\$' il file di stampa
(perforazione) va posto sul lettore virtuale della macchina
virtuale di nome "indirizzo".

Se la parte funzione e' un '=' il file di stampa
(perforazione) va inviato, tramite RSCS1 al terminale PSC di
nome "indirizzo".

Se la parte funzione e' un '\$' al file di stampa
(perforazione) va imposta la classe corrispondente al primo
carattere del campo indirizzo.

Nel caso della figura 3 il file di stampa va posto sul
lettore virtuale della macchina virtuale di nome VM128,
mentre il file di perforazione va perforato sul remote di
nome REM62, tramite RSCS1.

```
//NOME JOB (xxx,xxx,...),'2 ',xxxx...
```

fig.4

Nel caso della figura 4 al file di stampa va imposta la
classe '2' per una stampa locale.

All'interno della macchina virtuale VS2, per inviare i comandi descritti in precedenza, si fa uso di una istruzione di "DIAGNOSE", in particolare di quelle di tipo '8' che simula un comando di console.

Il formato di tale istruzione e' il seguente

```
      83      xy      ?      08  
byte 1 byte 2 byte 3 byte 4
```

Il primo byte contiene il codice operativo, il terzo e il quarto il tipo di DIAGNOSE, e il secondo contiene in ciascuno dei due semibyte, la specifica di un registro (registri x e y).

Il registro x contiene l'indirizzo (virtuale) di una stringa di caratteri, il registro y la lunghezza, per un massimo di 132. Questa stringa viene interpretata come un comando inviato da console.

Si intuisce quindi che all'interno della macchina virtuale VS2, analizzato il campo tra apici della scheda job, e stabilito quali azioni vanno intraprese, si dovranno costruire stringhe di caratteri che costituiscono i comandi opportuni e farle eseguire via la DIAGNOSE di tipo 8.

Occorre ora ricordare che la macchina virtuale VS2 e' definita con uno spazio di memoria reale, nel senso che essa non viene paginata dal VM, il quale le dedica una certa zona di memoria fisica, senza sapere in che modo viene usata.

Poiche' la macchina virtuale VS2 gestisce la memoria reale che le e' dedicata con un criterio di paginazione, ci troviamo nelle condizioni in cui una locazione di memoria virtuale della macchina virtuale VS2 non puo' essere raggiunta dal VM, il quale non sa in quale locazione fisica tale locazione e' stata posta.

Nel momento in cui la macchina virtuale VS2 lancia la DIAGNOSE di tipo '8', essa passa al VM un registro che contiene un indirizzo; il VM per eseguire correttamente la funzione richiesta, dovrebbe tradurre questo indirizzo in un altro, secondo le tabelle di paginazione della macchina virtuale VS2; in caso contrario il contenuto di tale registro e' senza alcun senso.

(Questo inconveniente non si verifica se la macchina virtuale VS2 e' definita "V=V" poiche' in tal caso entrano in funzione le "shadow tables". Cfr. IFF Virtual Machine Facility/370: Control Program - Program Logic - pg. 114 e seq.).

Vi sono diversi sistemi per ottenere un corretto risultato senza obbligare il VM ad accedere alle tabelle di paginazione della macchina virtuale VS2.

Quello che è stato usato in questo progetto è basato sul fatto che esiste una zona di memoria della macchina virtuale VS2 non dedicata alla paginazione. Il nucleo del sistema operativo VS infatti, negli indirizzi bassi di memoria, è gestito con un criterio $V=R$, cioè l'indirizzo virtuale corrisponde a quello reale.

Si intuisce quindi che la soluzione al nostro problema consiste nel preparare la stringa che rappresenta il comando dentro il nucleo del sistema operativo VS della macchina virtuale VS2.

Cio' può essere ottenuto facilmente mediante l'ausilio di una nuova SVC. (SVC n. 252).

Il criterio seguito è questo. Quando HASP ritiene che sia giunto il momento di lanciare un comando, secondo quanto eseguito precedentemente, prepara la stringa opportuna e lancia la SVC 252.

Questa SVC prende tutta la stringa e la muove in un'area dentro il suo stesso modulo, e poiché tale SVC è residente, quest'area si trova nel nucleo. A questo punto, importati i registri opportuni, viene lanciata la DIAGNCSF.

Riportiamo di seguito i listing delle modifiche apportate ad HASP e la SVC scritta.

Le modifiche segnate appartengono ad un'altro progetto ma non abbiamo ritenuto di doverle eliminare in quanto sono sviluppi successivi a questa stessa ricerca.

CHGEN	MVC	2 (R1), JCTPNAME	MOVE PPG. FIELD	MOGOL45	P3467000
	MVI	21 (R1), X'40'	MOVE BLANK	MOGOL45	P3467001
	MVI	21 (R1), Y'40'	MOVE BLANK	MOGOL45	P3467002
	MVC	22 (R1), JCTACCTN	MOVE ACCT. NUMBER	MOGOL45	P3467003
	MVI	25 (R1), X'40'	MOVE BLANK	MOGOL45	P3467004
	MVC	27 (R1), JCTROOMN	MOVE ROOM NUMBER	MOGOL45	P3467005
	MVI	31 (R1), X'40'	MOVE BLANK	MOGOL45	P3467006
	MVI	32 (R1), X'40'	MOVE BLANK	MOGOL45	P3467007
	LA	R1, 34 (R1)	SET POINTER	MOGOL45	P3467008
CHLOOP	MVI	0 (R1), X'6A'	MOVE 11/12 PUNCH	MOGOL45	P3467009
	LA	R1, 1 (R1)	INCREMENT POINTER	MOGOL45	P3467010
	CL	R1, =P'41'	TEST IF END	MOGOL45	P3467011
	BL	PCHLOOP	LOOP IF NOT	MOGOL45	P3467012
	LA	R1, 4	ADJUST SECOND POINTER	MOGOL45	P3467013
CHGEN1	MVI	0 (R1), Y'6A'	INITIATE WITH 11/12 PUNCH	MOGOL45	P3468000
	BL	PCHGEN1	LOOP IF NOT LAST CHAP.	MOGOL45	P3468001

```

*
MVI PPMFLAG,X'0C' RESET IL FLAG INDICATORE
TM PCEID,PCERJEID E' UN REMOTO?
BO PPMEND1 ESCI SE E' UN REMOTO
*
LA R1,JCTPNAME PUNTA AL PROG.NAME
*
TM PCEID,PCFPRSID SE E' UNA STAMPANTE
BO PPMPRINT R1 E' GIA' PUNTATO
*
LA R15,11 MASS.DISTANZA DFL CAMPO PERF.
PPMLOOP1 CLI C(R1),C', ' SE IL CAMPO STAMPA E' FINITO
BE PPMPUYES CERCA IL CAMPO PERFORAZ.
CLI C(R1),C' ' SE NON CI SONO ALTRI CAMPI
BE PPMNOPUN INDICA TALE SITUAZIONE
LA R1,1(R1) INCREMENTA R1
BCT R15,PPMLOOP1 E CONTINUA LA RICERCA
PPMNOPUN MVI PPMFLAG,X'0F' RIDONDANTE.MANCA IL CAMPO PERF.
B PPMEXEC VA SUBITO ALLA 'CLOSE'
*
PPMPUYES LA R1,1(R1) R1 PUNTA AL PRIMO CARATTERE
*
PPMPRINT CLI C(R1),C'=' SE NON E' UN UGUALE
BNE PPMSPCL VA A VEDERE SE E' UN $
*
MVI PPMFLAG,X'08' INDICA RICHIESTA DI TAG
B PPMTRANS VA A TRADURRE L'INDIRIZZO
*
PPMSPCL CLI C(R1),C'$' GUARDA SE E' UN DOLLARO
BE PPMSPCL1 SALTA SE E' UN $
*
CLI C(R1),C'&&' GUARDA SE E' &
BNE PPMCLSE VA A INDICARE SOLO LA CLOSE
MVI PPMFLAG,X'02' INDICA SOLO CLASS
B PPMTRANS VA A TRADURRE L'INDIRIZZO
PPMCLSE MVI PPMFLAG,X'0F' INDICA SOLO CLOSE (RIDONDANTE)
B PPMEXEC VA A ESEGUIRE LA DIAGNOSI
*
PPMSPCL1 CLC 1(6,R1),=CL6'LOCAL ' SE E'
BE PPMCLSE $LOCAL
CLC 1(6,R1),=CL6'LOCAL,' OPPURE
BE PPMCLSE $$$
CLC 1(3,R1),=CL3'SS ' VA
BE PPMCLSE ESEGUITA
CLC 1(3,R1),=CL3'SS,' SOLO
BE PPMCLSE LA CLOSIF
*
MVI PPMFLAG,X'04' ALTRIMENTI E' SPOOL
*

```

PPMTRANS DS	OH	
MVC	PPMNAME(8),1(R1)	MUOVI IL CAMPO LUNGO 8
*		
LA	R1,PPMNAME	R1 PUNTA AL CAMPO
LA	R15,8	LUNGHEZZA MASSIMA
*		
PPMLOOP2 CLI	C(R1),C','	SE IL CAMPO FINISCE CON ','
BE	PPMCM1	VA A FORZARE UN BLANK
CLI	C(R1),C' '	SE IL CAMPO FINISCE CON ' '
BE	PPMBL1	VA A FINIRE LA STRINGA
LA	R1,1(R1)	INCREMENTA R1
BCT	R15,PPMLOOP2	TORNA NEL LOOP
B	PPMEXEC	CAMPO O.K. VA AD ESEGUIRE
*		
PPMCM1 MVI	C(R1),C' '	FORZA ' ' SULLA ','
PPMBL1 LA	R15,PPMNAME+7	FINE DEL CAMPO
CR	R1,R15	SE R1 E' MINORE DI R15
BNM	PPMEXEC	CONTINUA IL LOOP
LA	R1,1(R1)	INCREMENTA R1
B	PPMCM1	TORNA A METTERE ' '. NON OTTIMIZZ.
*		

PMEXEC	MVC	PPMDEV(4),=CL4'	' NETTI BLANK NEL CAMPO DEVICE
	TM	PCEID,PCEPRSID	SE E' UNA STAMPANTE
	BO	PPMPUTE	VA A METTERE 'E'
	MVI	PPMDEV,C'D'	ALTRIMENTI METTI 'D'
	MVI	PPMUFLAG,C'D'	NEL COMANDO E NELLA SAVEAREA
	B	PPMTAGTG	VA AD ESAMINARE IL FLAG
PPMUTE	MVI	PPMDEV,C'E'	METTI IL CARATTERE 'E'
	MVI	PPMUFLAG,C'E'	NEL COMANDO E NELLA SAVEAREA

```
PMTAGTG  CLI  PPMFLAG,X'08'      SE NON E' UN TAG
          BNE  PPMSPISP          VA A VEDERE SE E' SPOOL

          MVC  PPMCOMM(8),=CL8'TAG DEV '  INSERISCI TAG
          BAL  R15,PPMSVC          VA A ESEGUIRE IL TAG
          MVC  PPMCOMM(8),=CL8'SPOOL'     INSERISCI SPOOL
          MVC  PPMNAME(8),=CL8'RSCS1'     INSERISCI RSCS1
          BAL  R15,PPMSVC          VA A ESEGUIRE LO SPOOL
          B    PPMEND1            VA AD ESEGUIRE LA CLOSE

PMSPLSP  CLI  PPMFLAG,X'04'      SE NON E' UNA RICHIESTA DI SPOOL
          BNE  PPMCLASS          VA A VEDERE SE E' UN 'CLASS'
          MVC  PPMCOMM(8),=CL8'SPOOL'     INSERISCI SPOOL
          BAL  R15,PPMSVC          VA AD ESEGUIRE LO SPOOL
          B    PPMEND1            VA AD ESEGUIRE LA CLOSE

PMCLASS  CLI  PPMFLAG,X'02'      SE NON E' UNA CLASS
          BNE  PPMEND1          VA AD ESEGUIRE LA CLOSE

          MVC  PPMNAME+7(1),PPMNAME      INSERISCI IL CARATTERE
          MVC  PPMNAME(7),=CL7'CLASS'    INSERISCI CLASS
          MVC  PPMCOMM(8),=CL8'SPOOL'    INSERISCI SPOOL
          BAL  R15,PPMSVC          ESEGUI SPOOL CLASS
PMEND1   DS    OH                FINE DELLA PRIMA PART
```

```

*
      CLI   PPMFLAG,X'01'      SE ERA UN REMOTO
      BE    PPMEND2            SALTA ALLA FINE

*
PPMCLOSE MVC   PPMNAME(4),=CL4'      ' METTI BLANK IN CAMPO 'NONE'
          MVC   PPMNAME+4(4),=CL4'    ' LUNGO OTTO
          MVC   PPMDEV(4),=CL4'      ' METTI BLANK IN CAMPO UNIT
          MVC   PPMDEV(1),PPMDEVFLAG  INSERISCI L'INDIRIZZO DEI DEVICE
          MVC   PPMCOMM(8),=CL8'CLOSE' INSERISCI CLOSE
          BAL   R15,PPMSVC          ESEGUI CLOSE

*
PPMRESET CLI   PPMFLAG,X'08'      ERA UN TAG?
          BNE   PPMRSSPL          VA A VEDERE SE ERA SPOOL
          MVC   PPMCOMM(8),=CL8'TAG  DEV ' INSERISCI TAG DEV
          BAL   R15,PPMSVC          ESEGUI TAG OFF

*
PPMRSTSP MVC   PPMCOMM(8),=CL8'SPOOL'  INSERISCI SPOOL
          MVC   PPMNAME(8),=CL8'OFF'    INSERISCI 'OFF'
          BAL   R15,PPMSVC          ESEGUI SPOOL OFF
          B     PPMEND2            FINE

*
PPMRSSPL CLI   PPMFLAG,X'04'      SE ERA SPOOL
          BE    PPMRSTSP          VA A FARE SPOOL OFF

*
          CLI   PPMFLAG,X'02'      SE NON ERA CLASS
          BNE   PPMEND2            VA A FINE
          MVC   PPMNAME(8),=CL8'CLASS A' INSERISCI CLASS A
          MVC   PPMCOMM(8),=CL8'SPOOL'  INSERISCI SPOOL
          BAL   R15,PPMSVC          ESEGUI RESET DELLA CLASSE
          B     PPMEND2            FINE

```

PPMSVC DS CH INGRESSO
 LA R1,PPMCOMX PREPARA
 ST R1,PPMP1 I DUE
 LA R1,PPMRETCO PUNTATORI
 ST R1,PPMP2 PER LA SVC
 MVC PPMRETCO(4), PASSWORD IMPOSTA LA PASSWORD
 LA R1,PPMP1 IMPOSTA IL REG 1

SVC 237
 BR R15 RITORNA

*

* COSTANTI USATE

PPMRETCO EQU PMESSAGE
PPMCOMM EQU PMESSAGE+8
PPMP1 EQU PMESSAGE+32
PPMP2 EQU PMESSAGE+36
PPMFLAG EQU PDDBSKIP+2
PPMUFLAG EQU PDDBSKIP+3
PPMDEV EQU PPMCOMM+8
PPMNAME EQU PPMCOMM+12
PASSWORD DC '0',X'01234567'

*

PPMEND2 DS OH FINE DELLA SECONDA PARTE

*

L RO, PASSWORD
LM PC1, PC2, PRCCWEJ
BAL PL, PPPUT

INDICA SKIP FORZATO
CARICA CCW DI SKIP
AGGIUNGI ALLA CATENA DI CCW

MOGOL56 P5489970
MOGOL56 P5489980
MOGOL56 P5489990

CL RO, PASSWORD
BE ECHAN1

SE E' RICHIESTO SKIP FORZATO
NON ESEGUIRE IL CONTROLLO

MOGOL56 P5489991
MOGOL56 P5939900
MOGOL56 P5939950

IGC237 CSECT

BALR	9,0		
USING	*,9		
*			
LM	4,5,0(1)	CARICA I REGISTRI 4 E 5 CON PUNTATORI	
CLC	0(4,5),PASSWORD	CONTROLLA LA PASSWORD	
BNE	RETURN8	SE NON UGUALE RITORNA (C.C.=8)	
*			
MVC	COMANDO(20),0(4)	PRENDI LA STRINGA DEL COMANDO	
LA	4,COMANDO	REG. 4 CONTIENE L'INDIRIZZO	
LA	6,20	REG.6 = DEC.20 LUNGHEZZA FISSA	
*			
DC	X'83460008'	DIAGNOSE TIPO 8	REG. 4 P 6
*			
LTR	6,6	CONTROLLA IL CODICE DI RITORNO	
BZ	RETURN	SE O.K. RITORNA	
MVC	0(4,5),=F'12'	IMPOSTA CODICE DI RITORNO 12	
BR	14	RITORNA . COMANDO RIFIUTATO	
*			
RETURN8	MVC	0(4,5),=F'8'	IMPOSTA CODICE DI RITORNO A 8.
	BR	14	RITORNA. SENZA AUTORIZZAZIONE
*			
RETURN	XC	0(4,5),0(5)	IMPOSTA CODICE DI RITORNO A ZERO
	BR	14	RITORNO TUTTO O.K.
*			
PASSWORD	DC	X'01234567'	
COMANDO	DS	0D,CL20	
*			
	END	IGC237	
*			

Una parte di questa ricerca è stata presentata al SEAS ANNEVERSARY
MEETING in Dublino l'8 Settembre 1975.

