Consiglio Nazionale delle Ricerche

Un sistema per la gestione automatica dei dati di produzione sul sistema accoppiato IBM 360/67 - IBM 370/158 del CNUCE

R. Ferrini - O. Signore - E. Stefanelli

57

CNUCE

A cura di: R. Ferrini
O. Signore
E. Stefanelli

CONSIGLIO NAZIONALE DELLE RICERCHE

CNUCE

Un sistema per la gestione automatica dei dati di produzione sul sistema accoppiato IBM 360/67 - IBM 370/158 del CNUCE.

R. Ferrini

O. Signore

E. Stefanelli

Introduzione

Il CNUCE dispone di due sistemi di calcolo, IBM 360/67 con CP/CMS e IBM 370/158 con OS/VS2 Rel 1.6/HASP, connessi mediante linea BSC ad alta velocita'. Tale connessione e' realizzata per mezzo di una particolare macchina virtuale (OS370) che, gestita da un programma stand-alone, colloquia come fosse un 360/65 con HASP ed e' in grado di leggere, innescata da un interrupt di I/O, i files che vengono a trovarsi sul suo reader virtuale e di trasferire i files provenienti da HASP al READER della macchina virtuale interessata sul 360/67.
In tale ambiente il problema del rilevamento dei dati di produzione, necessari sia per conoscere l'effettivo risultato del servizio sia per ricavare gli addebiti relativi al lavoro dell'utente, e' risolto nella seguente maniera:

sul 360/67:

- il CP, per ogni sessione di macchina virtuale e per ogni device dedicato, perfora una scheda;

- il CMS-BATCH, per ogni job, perfora "virtualmente" una scheda;

sul 370/158:

- il VS e lo HASP tramite le SMF scrivono su SYS1.MANX/Y i records 4,5,6,26.

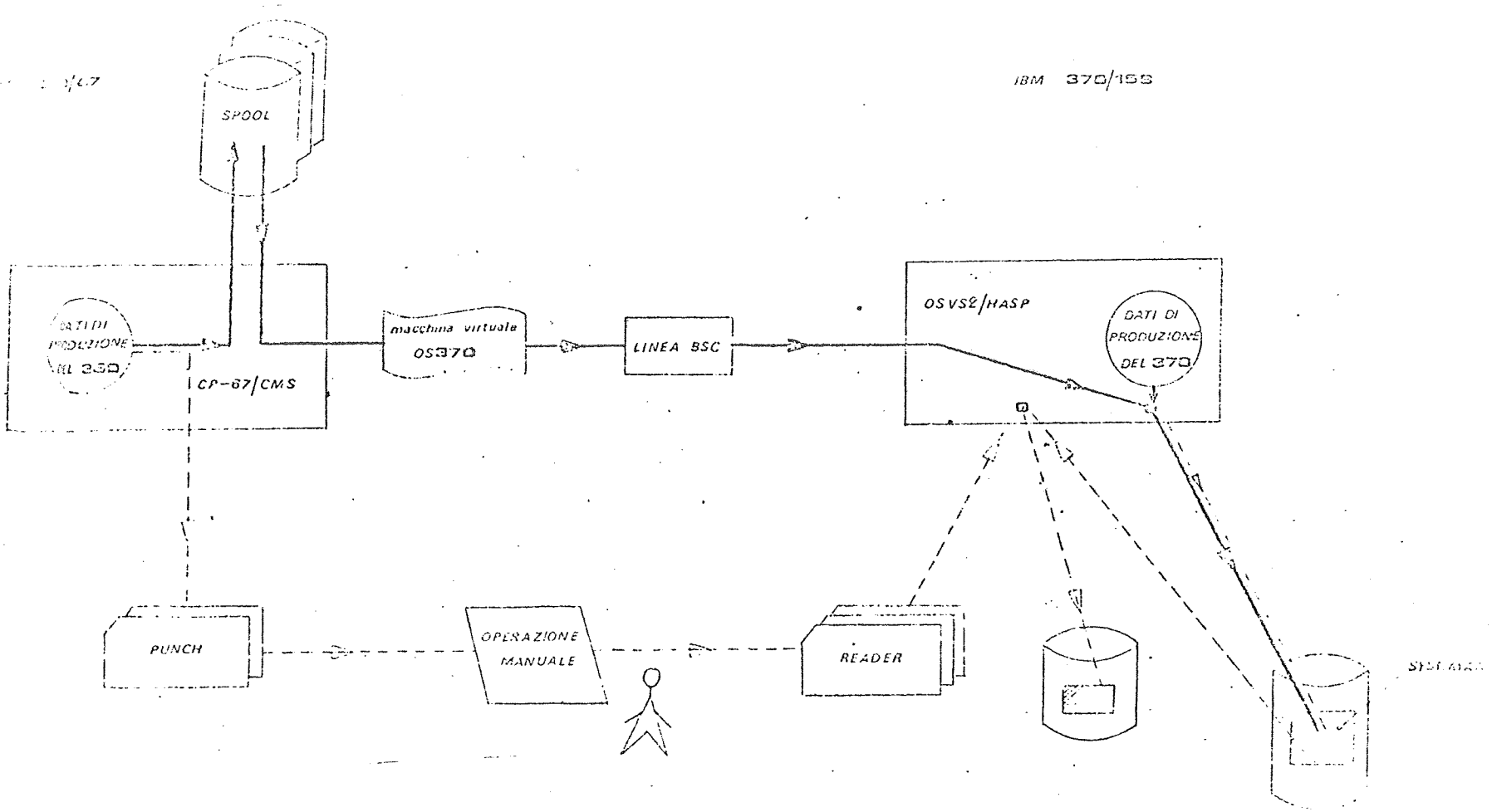
Questi metodi, sia in assoluto sia alla luce dell'accoppiamento tra i due sistemi, presentano alcuni inconvenienti di carattere tecnico ed altri di carattere organizzativo. In particolare, mentre si rimanda alla figura A (parte tratteggiata) per il flusso delle informazioni dal rilevamento in memoria alla riunione su di un supporto di I/O per l'elaborazione completa di questi dati, accade che

- essendo necessaria per ogni gruppo di 80 caratteri una SIO sul punch da parte del CP/67, il sistema e' soggetto ad un dispendio di CPU che deve considerarsi dannoso in un ambiente di tipo conversazionale;

- in caso di malfunzionamento del puncher del 360/67 o qualora esso si trovi in manutenzione, le informazioni in oggetto occupano la memoria centrale senza che sia possibile trasferirle su di un dispositivo ausiliario;

- la meccanica del puncher e la conseguente necessita' di usare il reader per l'elaborazione aumentano sensibilmente la possibilita' di errore e comportano un impegno operativo non indifferente;

- trovandosi, alla fine del rilevamento, le informazioni relative ai due sistemi su dispositivi non solo distinti ma anche di tipo diverso, la gestione di tali dati necessita, prima dell'elaborazione unitaria, di ulteriori procedure sia di programmazione che operative.



IBM 370/168

FLUSSO PRECEDENTE

FLUSSO AUTOMATICO

FIG. A

Il sistema automatico: i principi

I punti critici precedentemente individuati hanno imposto una revisione totale del processo di gestione che ha portato alla definizione ed alla realizzazione di un nuovo sistema. Questo sistema si basa sui seguenti principi:

- l'operatore non deve essere coinvolto nella gestione di tali informazioni;

- il risultato del processo deve portare in ogni momento tutte le informazioni (di entrambe le macchine) sullo stesso dispositivo veloce;

- l'affidabilita' del sistema, causa l'importanza di tali dati, deve essere completa;

- l'occupazione delle risorse per la realizzazione deve essere minima e sopra tutto deve influenzare il meno possibile l'utente;

Per realizzare questo sistema si e' modificato totalmente il flusso delle informazioni.

In particolare:

- e' stato modificato il CP-67 per inviare le informazioni sia di CP che di CMS-BATCH non sul puncher ma sul lettore virtuale di OS370;

- sono state modificate tutte le procedure di recovery del CP;

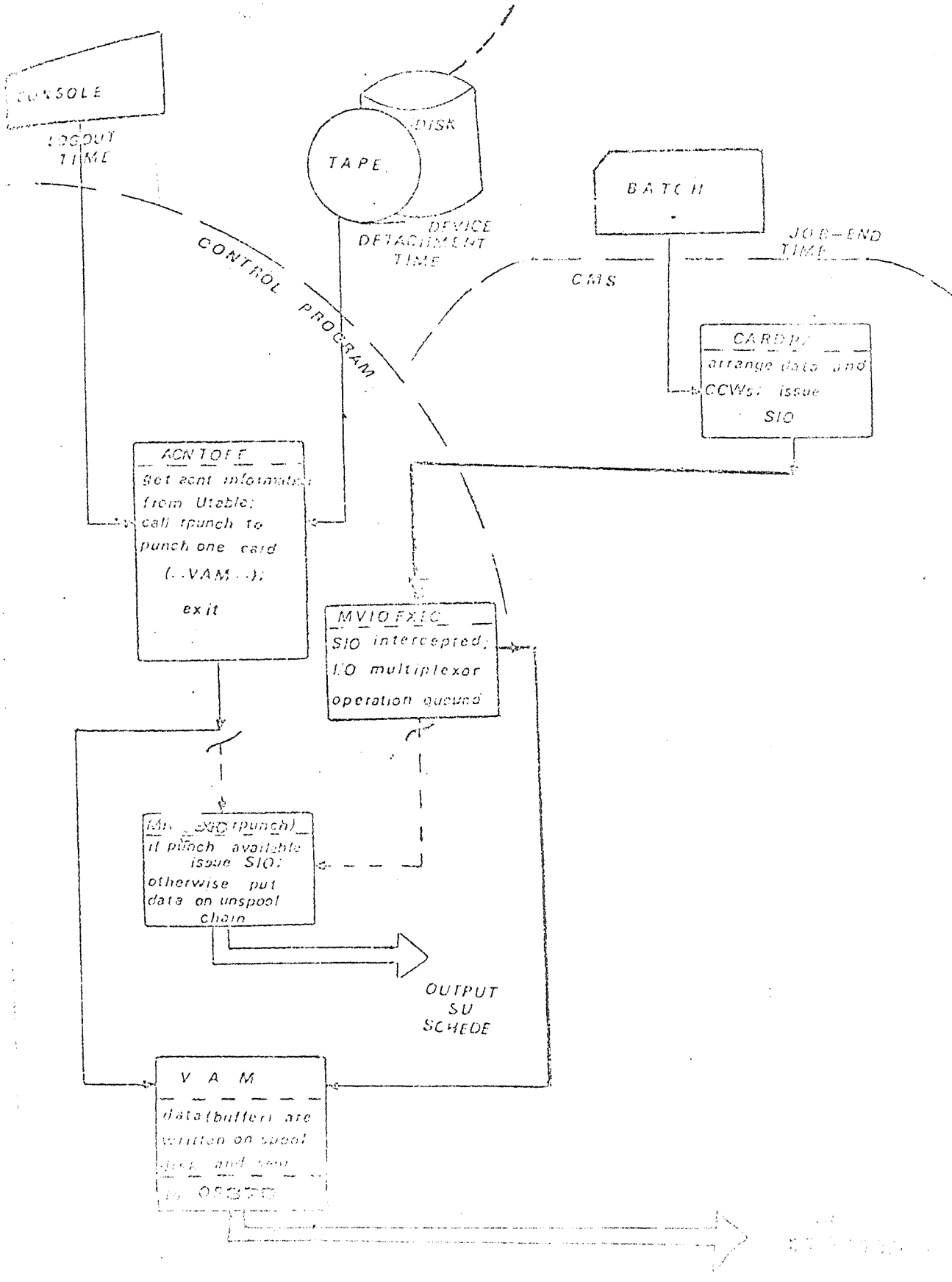
- e' stato modificato HASP perche' selezionasse queste informazioni e le scrivesse su SYS1.MANX/Y sotto forma di un nuovo record SMF.

Il nuovo flusso di informazioni e' visualizzato in figura A (parte piena). E' opportuno insistere sul problema dell'affidabilita' perche' esso ha condizionato ogni scelta ed ogni metodo realizzativo e' stato giudicato tanto valido quanto piu' esso incontrava la necessita' di provvedere al salvataggio in caso di crash.

Il sistema automatico: i metodi realizzativi

a) il prelievo dei dati sul 360/67 e l'invio al 370/158

Questa parte e' stata realizzata modificando l'aggancio tra ACNTOFF e MRIOEXEC e la scrittura, tramite MVIOEXEC, sul disco di SPOOL delle schede di accounting relative ai jobs CMS-BATCH, con salti incondizionati alla routine VAM.
Questa routine, scritta ex novo e inserita nel CP, provvede al trasferimento di tali informazioni al reader virtuale di OS370 (disco di SPOOL) che, come e' sua natura, li invia tramite linea BSC al 370/158. (Figura B)
Piu' in dettaglio ACNTOFF ad ogni logout o detach e MVIOEXEC ogni volta che individua una perforazione sullo stacker 3 (scheda di accounting di un job CMS-BATCH) trasferiscono le informazioni a VAM che provvede al riempimento di un buffer di 800 bytes (BVAM), che viene acquisito di volta in volta e il cui indirizzo viene salvato in EQU-67. Quando BVAM e' finito, VAM scrive le informazioni sul disco di SPOOL lasciando aperto il file e salvando la posizione su disco in EQU-67: questo processo continua fino a che sono stati scritti sullo SPOOL N insiemi di 800 bytes, momento in cui VAM concatena le informazioni in un unico file (FVAM) che viene indirizzato a OS370 e riflette l'interrupt di I/O relativo al reader a quest'ultima macchina perche' questo file venga subito letto e trasferito.
E' opportuno ora discutere il valore da assegnare ad N: infatti dando a N un valore basso e' necessario chiudere piu' spesso il file e si impone alla macchina virtuale OS370 una lettura e un trasferimento piu' frequente mentre dando a N un valore piu' grande e' necessario occupare piu' area di SPOOL e, in caso di ripartenza a freddo, le informazioni perdute sarebbero maggiori. Attualmente si e' scelto N=2 che si traduce, in considerazione dell'attuale carico macchina, in circa 25 trasferimenti giornalieri ma non si esclude che questo valore possa essere aumentato se sara' ritenuto opportuno.
Mentre si rimanda all'appendice I per queste modifiche e' opportuno affrontare ora il problema del recovery.
Il recovery standard che in alcuni casi si traduce nel richiamo da disco del modulo CHKPT ed in altri nel caricamento del programma stand-alone BUZZARD, non prevede il salvataggio dei files ancora aperti e naturalmente neanche del buffer BVAM. Inoltre dopo il Re-ipl il CP/67, tramite il modulo CPINIT, legge le informazioni relative alle macchine attive al momento del CRASH, salvate, e le invia al perforatore.


CONSOLE
LOGOUT TIME
DISK
TAPE
DEVICE DETACHMENT TIME
BATCH
JOB-END TIME
CMS
CONTROL PROGRAM
CARDP/
arrange data and CCWs; issue SIO
ACNTOFF
get acnt information from Utable; call rpunch to punch one card (..VAM..); exit
MVIOFXEC
SIO intercepted; I/O multiplexor operation queued
(M... ...XIO rpunch)
if punch available issue SIO; otherwise put data on unspool chain
OUTPUT SU SCHEDE
V A M
data(buffer) are written on spool

Questa situazione ha imposto modifiche a CHKPT e BUZZARD perche', prima di effettuare il previsto salvataggio, provvedessero a scrivere il buffer non ancora completato sul disco di SPOOL, concatenassero queste informazioni all'eventuale file ancora aperto sullo SPOOL e infine chiudessero il file (FVAM).
Inoltre e' stato modificato CPINIT perche', invece di inviare le informazioni relative alle macchine attive al momento del crash sul perforatore, le trasferisca a VAM che, come e' sua natura, provvede ad inviarle ad OS370.
Mentre si rimanda all'appendice II per queste modifiche e' opportuno specificare che, in considerazione delle particolari condizioni del CP/67 in queste situazioni, e' stato impossibile usare i moduli standard per adempiere alle funzioni descritte e quindi e' stato necessario, per esempio, lanciare direttamente l'operazione di canale.
Si conclude osservando che le indicazioni per recuperare le informazioni suddette si trovano in particolari posizioni di EQU-67 preventivamente impostate da VAM.

b) la raccolta dei dati sul 370/158 e la scrittura su disco

Questa seconda parte e' stata realizzata modificando HASP perche' invece di effettuare uno "SKIP" di queste informazioni, provvedesse a scriverle sul data set SYS1.MANX/Y come un nuovo tipo di record (128).
Cio' e' stato possibile imponendo che la prima "scheda" di ogni FVAM fosse una /* ed in particolare una /*PRIORITY con un carattere speciale a colonna 16 (FF).
Quando HASP rileva questa situazione in RPRICARD chiude normalmente il job precedente, se esiste, e imposta uno switch che segnala la presenza di un file FVAM in arrivo; quando vengono rilevate le schede seguenti, esse vengono trasferite alla routine RPUTACCT (scritta ex novo ed aggiunta ad HASP), che le accumula in un buffer di Nx80 caratteri e, quando questo buffer e' pieno, esegue una SMFWTM su SYS1 MANX/Y. In caso di "end of file" o di rilevamento di una //job o /* PRIORITY il sistema conclude le operazioni in corso e il processo continua normalmente.
Mentre si rimanda all'appendice III per queste modifiche e' opportuno discutere il valore dato ad N. Esso e' stato scelto uguale a 4, non per particolari ragioni di ottimizzazione ma solo perche' attualmente lo HASP INPUT PROCESSOR, per essere contenuto in 4096 bytes, non permette un valore maggiore.
Si vuole concludere questa parte aggiungendo che la logica di tali modifiche e' stata scelta proprio per non alterare la filosofia di HASP.

Conclusioni

Alla luce di quanto detto nell'illustrazione dei metodi realizzativi si puo' concludere che questo sistema risolve tutti gli inconvenienti descritti nell'introduzione ed adempie ai principi indicati nel disegno pur senza caricare apprezzabilmente le risorse del sistema.
Infatti per cio' che riguarda la memoria, le modifiche relative al CP/67 impongono una spesa ulteriore di circa 1500 bytes mentre quelle relative allo HASP di circa 500 bytes ma nello stesso tempo e' eliminata la necessita' in caso di crash del perforatore che queste informazioni rimangano in memoria.
L'occupazione della linea e del disco di SPOOL e' trascurabile e viene compensata dal "bypass" del perforatore e delle schede.
L'utilizzo della CPU e' minimo ed anzi relativamente alle SIO e' diminuito sul 360/67 mentre e' aumentato sul 370/158 dove pero' questa incidenza nell'economia generale del sistema non e' ne' pesante ne' paragonabile a quella che si avrebbe in un sistema di tipo conversazionale come il CP/CMS.
L'insieme di questo carico e' da considerarsi in ogni caso insignificante se si nota che questo sistema "sorpassa" completamente il supporto schede e l'operatore e riunisce tutte le informazioni su un disco permanentemente in linea in modo automatico e affidabile.
Per concludere e' opportuno dire che tutto il sistema anche se necessita di un certo periodo di controllo in normale produzione, e'da considerarsi "funzionante".

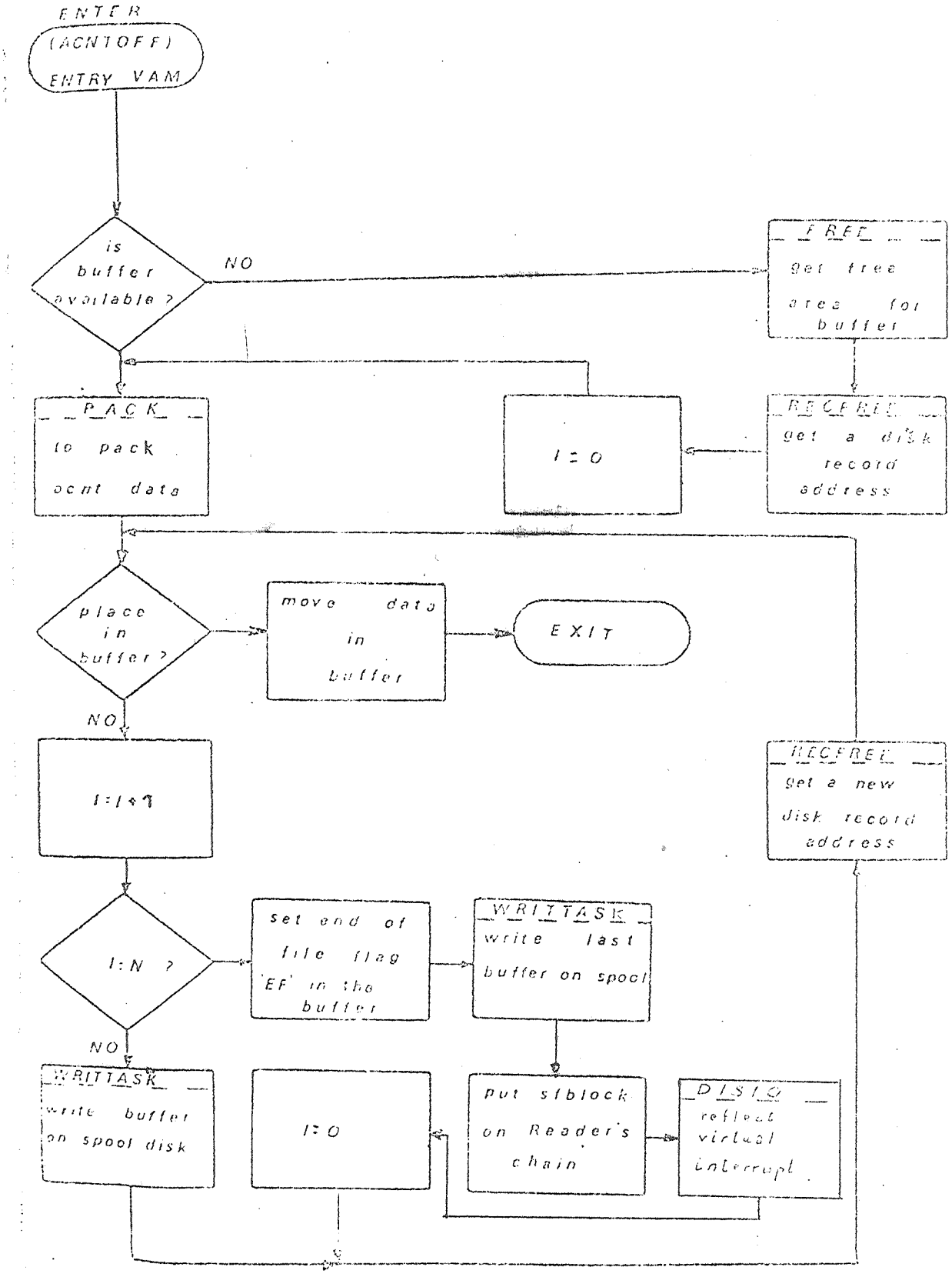
Appendice I

```
VAM

*****************************************************************
*                                                               *
*         ROUTINE TO WRITE ACCOUNTING CARD ON SPOOL DISK        *
*                                                               *
*         ENTRY POINT    R4 ADDRESS OF ACCOUNTING DATA          *
*                                                               *
*****************************************************************

VAM       DS    OH
          USING *,R12
          ENTER (R0,R11)
          CLI   VAMSW2,X'01'              IS WORK AREA AVAILABLE?
          BE    VAM1                           YES, GO
          LA    R0,BVAMSZ
          CALL  FREE
          LR    R6,R1
          USING BUFVAM,R6
          ST    R1,ADDR                   SAVE ADDR. OF THE BUFFER
          ST    R1,ABUFVAM                SAVE FOR CHKPT
          MVI   VAMSW2,X'01'              NOW IT IS AVAILABLE
          B     VAM2
VAM1      L     R6,ADDR                   R6 BASE REGISTER
          USING BUFVAM,R6                      OF THE WORK AREA
VAM2      ST    R4,VAMADDM                SAVE DATA ADDRESS
          CLI   VAMSW1,X'01'              IS BUFFER AVAILABLE?
          BE    VAMIO1                    YES,GO
          BAL   R5,PRBUF                  NO,GET BUFFER
VAMIO1    MVI   VAMDAT,X'40'              BLANKS IN CURRENT AREA
          MVC   VAMDAT+1(83),VAMDAT
          L     R9,VAMADDR                R9 BASE REGISTER OF THE BUFFER
          USING BUFFREE,R9
          LA    R2,80                     DATA SIZE
          STC   R2,VAMDAT                 IN CURRENT AREA
          CLI   VAMSW3,X'01'              IN THE FIRST CARD?
          BNE   VAMEX                     NO
          MVC   VAMDAT+1(10),=C'/*PRIORITY'    PUT CONTROL CARD IN BUFFER
          MVC   VAMDAT+16(1),=X'FF'
          B     VAMEX2
VAMEX     MVC   VAMDAT+1(80),24(R4)       MOVE DATA IN BUFFER
VAMEX2    LA    R1,VAMDAT                 FOR PACK
          LA    R2,VAMDATP                FOR PACK
          CALL  PACK
VAMEX1    AH    R9,VAMCOUNT               UPDATE BUFFER ADDRESS
          SR    R7,R7                     CLEAR
          IC    R7,0(R2)                  DATA PACKED SIZE
```

```
         LA     R5,827                  BUFFER SIZE
         SH     R5,VAMCOUNT             AREA STILL AVAILABLE?
         CR     R5,R7                   ENOUGH SPACE?
         BH     VAMENGH                 YES, PUT DATA IN IT
         BAL    R7,VAMRF                NO,GO TO WRITE
         B      VAMEX1                  PUT DATA IN BUFFER
ADDR     DS     OF
VAMSW2   DS     F
PRBUF    DC     X'00'
         LA     R0,105                  BUFFER SIZE
         CALL   FREE
         LR     R9,R1                   R9 BASE REGISTER
         USING  BUFFREE,R9
         ST     R1,VAMADDR              SAVE
PRBUF1   XC     0(256,R9),0(R9)         SAVE
         XC     256(256,R9),256(R9)     CLEAR BUFFER
         XC     512(256,R9),512(R9)
         XC     768(72,R9),768(R9)
         CALL   RECFREE                 GET RECORD ADDRESS
         MVC    VAMBBCC1(8),0(R1)       SAVE FIRST RECORD ADDRESS
         MVI    VAMSW1,X'01'            NOW BUFFER IS AVAILABLE
         MVI    VAMSW3,X'01'            FOR CONTROL CARD
         MVC    VAMCOUNT(2),=X'0005'    INITIAL COUNT
         MVI    VAMSKCNT,X'C0'          INITIAL NUMBER OF BUFFERS
         MVC    VAMXID(8),=C'OS370'     USER TO XFER ACCOUNTING DATA
         MVC    VAMBBCC(8),0(R1)        SAVE DISK ADDRESS FOR CURR-BUFF
         CALL   FRET                     FREE DOUBLE WORD
         BR     R5                      RETURN
VAMENGH  EX     R7,VAMMI                MOVE DATA IN BUFFER
         LA     R7,1(R7)                ADD 1
         AH     R7,VAMCOUNT             UPDATE COUNT
         STH    R7,VAMCOUNT             SAVE
         CLI    VAMSW3,X'01'            WAS IT THE FIRST CARD?
         BNE    VAMEXIT                 NO,EXIT
         MVI    VAMSW3,X'00'            CLEAR SWITCH
         B      VAMIO1                  THEN PUT ACCT DATA IN BUFFER
VAMEXIT  EXIT   (R0,R11)
VAMMI    MVC    0(0,R9),0(R2)           EXECUTE
VAMRF    IC     R5,VAMSKCNT             BUFFERS' NUMBER IN R5
         LA     R5,1(R5)                UPDATE
         STC    R5,VAMSKCNT             SAVE
         CLI    VAMSKCNT,X'02'          IS IT THE LAST BUFFER?
         BE     VAMCLOSE                YES,CLOSE FILE
         MVI    0(R9),X'FF'             SET 'FF' FLAG AT THE END
         CALL   RECFREE                 GET NEXT RECORD ADDRESS
         MVC    VAMBC(8),0(R1)          SAVE
         L      R9,VAMADDR              GET BUFFER ADDRESS
         MVC    0(5,R9),3(R1)           PUT NEXT RECORD ADDRESS IN TOP
*                                       OF CURRENT BUFFER
         CALL   FRET                    FREE DOUBLE WORD
VAMRF1   LR     R1,R9                   FOR WRITTASK
```

```
         LA     R3,VAMBECC                FOR WRITTASK
         L      R15,=V(RDEVTABL)          GET DEVICE TABLE
         SR     R2,R2                     CLEAR
         IC     R2,VAMHHRD+3              GET DEVICE CODE
         L      R2,0(R2,R15)              GET RDEVBLOK
         CALL   WRITTASK                  WRITE
         MVC    VAMBBCC(8),VAMBC          SAVE DISK ADDR. FOR CURR. BUFF.
         MVC    VAMCOUNT(2),=X'0005'      UPDATE COUNT
         LA     R2,VAMDATP                RESTORE DATA PACKED ADDRESS
         BR     R7                        RETURN
VAMCLOSE L      R9,VAMADDR                GET BUFFER ADDRESS
         MVI    0(R9),X'00'
         MVC    1(4,R9),0(R9)             ZERO IN FIRST FIVE BYTES
         AH     R9,VAMCOUNT               ADD COUNT
         MVI    0(R9),X'EF'               END OF FILE
         L      R9,VAMADDR
         BAL    R7,VAMRF1                 GO TO WRITE THE BUFFER
         LA     R0,3                      THREE DOUBLE WORDS FOR SFBLOK
         CALL   FREE
         XC     0(24,R1),0(R1)            CLEAR SFBLOK
         MVC    4(8,R1),VAMBBCC1          MOVE FIRST RECORD ADDRESS IN IT
         MVC    16(8,R1),VAMXID           AND USERID
         MVI    15(R1),TYP2540R           AND DEVICE TYPE
         L      R15,=V(READERS)
VAMF2    L      R3,0(R15)
         LTR    R3,R3
         BZ     VAMF1
         LR     R15,R3
         B      VAMF2
VAMF1    ST     R1,0(R15)
         L      R11,RUNUSER
         L      R2,=A(NUMUSERS)
         LH     R14,0(R2)                 NUM USERS
VAMF3    CLC    USERID(8),16(R1)          RIGHT UTABLE?
          BE     VAMF9                     FOUND
         L      R11,NEXTUSER              NEXT
         BCT    R14,VAMF3                 LOOP
VAMF4    BAL    R5,PRBUF1                 NOT FOUND
         B      VAMIO1                    PUT DATA IN BUFFER
VAMF9    EQU    *
         USING  MUDEBLOK,R5
         L      R5,VMXSTART               MPX START
VAMF6    LTR    R5,R5                     END?
         BZ     VAMF4                     YES
         CLI    MUDEVTYP,TYP2540R         RIGHT DEVICE?
         BE     VAMF5                     YES
VAMF7    L      R5,MUDEVPNT               NEXT DEV
         B      VAMF6
VAMF5    L      R2,MVIOB
         LTR    R2,R2                     TEST FOR BUSY
         BNZ    VAMF7
```

```
          MVI     MUDESTAT,DE                DEVICE END INT
          LH      R2,MUDEVADD                DEV ADDRESS
          SRL     R2,8                       GET CHAN
          L       R3,=V(CHMASKS)             GET MASK BITS
          LA      R3,0(R2,R3)
          OC      PENDING (1),0(R3)
          CALL    DISIO
          B       VAMF4
          DROP    R5
***       THIS BUFFER IS USED BY VAM ROUTINE
          DS      0F
BUFFREE   DSECT
VAMBUF    DS      840C
BFRSIZE   EQU     (*-BUFFREE)/8
BUFVAM    DSECT
VAMADDM   DS      F
VAMADDR   DS      F
VAMBBCC1  DS      F
VAMHHRD1  DS      F
VAMBBCC   DS      F
VAMHHRD   DS      F
VAMBC     DS      F
VAMHD     DS      F
VAMSKCNT  DC      X'00'
VAMSWI    DC      X'00'
VAMCOUNT  DC      H'5'
VAMDAT    DS      CL84
VAMDATP   DS      CL84
VAMXID    DS      D
VAMSW3    DC      X'00'
BVAMSZ    EQU     (*-BUFVAM)/8
TYP2540R  EQU     60
```


ENTER
(ACNTOFF)
ENTRY VAM
is buffer available ?
NO
FREE
get free area for buffer
PACK
to pack acnt data
I = 0
RECFREE
get a disk record address
place in buffer ?
move data in buffer
EXIT
NO
I = I + 1
RECFREE
get a new disk record address
I = N ?
set end of file flag 'EF' in the buffer
WRITTASK
write last buffer on spool
NO
WRITTASK
write buffer on spool disk
I = 0
put sfblock on Reader's chain
DIAG
reflect virtual interrupt

Modifiche a MVIOEXEC

```
*****************************************************************
*    MVIOEXEC UPDATES TO WRITE BATCH USER'S ACCOUNTING          *
*       RECORDS ON SPOOL DISK THROUGH VAM ROUTINE               *
*****************************************************************


          EXTRN  VAM
MVIPACK   CLI    MVICCW,X'81'              IS CCW A PUNCH TO STACKER NUM. 3?
          BNE    MVIGO                     NO,CONTINUE
          LR     R4,R1
          S      R4,=F'23'
          LA     R4,0(R4)                  R4 ADDRESS OF DATA
          CALL   VAM                       WRITE DATA IN THE BUFFER
MVIGO     STC    R0,0(,R1)
* LATER THE FILLED BUFFER WILL BE WRITTEN ON SPOOL DISK
                         -------
                         -------
```

APPENDICE II

Modifiche a CHKPT

```
*****************************************************************
*     CHKPT UPDATES TO SAVE IN CORE ACCT RECORDS AFTER          *
*              SYSTEM ABNORMAL FAILURE (CRASH)                  *
*****************************************************************

TAMSAVE   L       R4,ABUFVAM                READ ADDRESS IN EQU SECTION
          C       R4,=F'0'                  NO BUFFER?
          BNE     VAMCOUNT
          BR      R10                       YES,RETURN
VAMCOUNT  L       R9,4(R4)                  GET BUFFER ADDRESS
          MVI     0(R9),X'00'
          MVC     1(4,R9),0(R9)             ZEROES IN POINTER
          AH      R9,34(R4)                 ADD COUNT
          MVI     0(R9),X'EF'               EOF FLAG IN THE BUFFER
          L       R9,4(R4)
          MVI     0(R9),X'00'               PUT ACCT BUFFER IN CUTPUT BUFFER
          MVC     RECBUFF+256(256),256
          MVC     RECBUFF+512(256),512(R9)
          MVC     RECBUFF+768(62),768(R9)
          MVC     24(8,R4),SEEK             SAVE STANDARD SEEK
          MVC     SEEK(8),16(R4)
          MVC     SAVESIO(4),SIOINTP+4
          MVC     SIOINTP+4(4),SIOTRY
          L       R15,ADEVTABL              GET DEVTABL ADDR
          SR      R2,R2                     CLEAR
          IC      R2,REC+1                  DEVICE CODE
          L       R8,0(R2,R15)              R8 RDEVBLOK ADDR
          LH      R5,RDEVADD                REAL ADDR
          L       R7,RDEVCU
          L       R6,RACTCHAN               R6 RCHBLOK ADDR
          LH      R2,RCHANADD               CHANNEL ADDRESS
          OR      R2,R5                     COMPUTE REAL DEVICE ADDR
          LA      R0,DSKCCW                 CCW ADDRESS
          ST      R0,CAW                    CAW
          LA      R9,200                    RETRY COUNT
VAMSIO    SIO     0(R2)                     ISSUE SIO TO SPOOL-DISK
          BNZ     SIOFAIL                   RETRY IF NOT SUCCESSFULL
          MVC     IONPSW(8),IOINTPVM
          LPSW    IWAIT                     WAIT FOR I/O INTERRUPT
IOINTVAM   XC       TABREAD(24),TABREAD       THE   FOLLOWING  INSTRUCT.  ARE
REQUIRED
          MVC     TABREAD+4(8),8(R4)        TO UPDATE READERS CHAIN
          MVC     TABREAD+16(8),208(R4)
```

```
         MVI     TABREAD+15,TYP2540R
         L       R15,AREADERS                GET READERS CHAIN
VAMF2    L       R2,0(R15)                   FIRST (NEXT) SFBLOK
         LTR     R2,R2                       THE LAST ?
         BZ      VAMF1                       YES
         LR      R15,R2                      NEXT SFBLOK
         B       VAMF2                       CONTINUE
VAMF1    LA      R2,TABREAD
         ST      R2,0(R15)
         MVC     SEEK(8),24(R4)              RESTORE SEEK
         MVC     SIOINTP+4(4),SAVESIO        RESTORE
         BR       R10                           RETURN  TO   STANDARD  RECOVERY
PROGRAM
SAVESIO  DS      F
SIOTRY   DC      A(VAMSIO-CHKPT+X'800')
TABREAD  DS      D
         DS      D
         DS      D
IOINTPVM DC      X'00040000'
         DC      A(IOINTVAM-CHKPT+X'800')
                 -------
                 -------
```

Modifiche a CPINIT

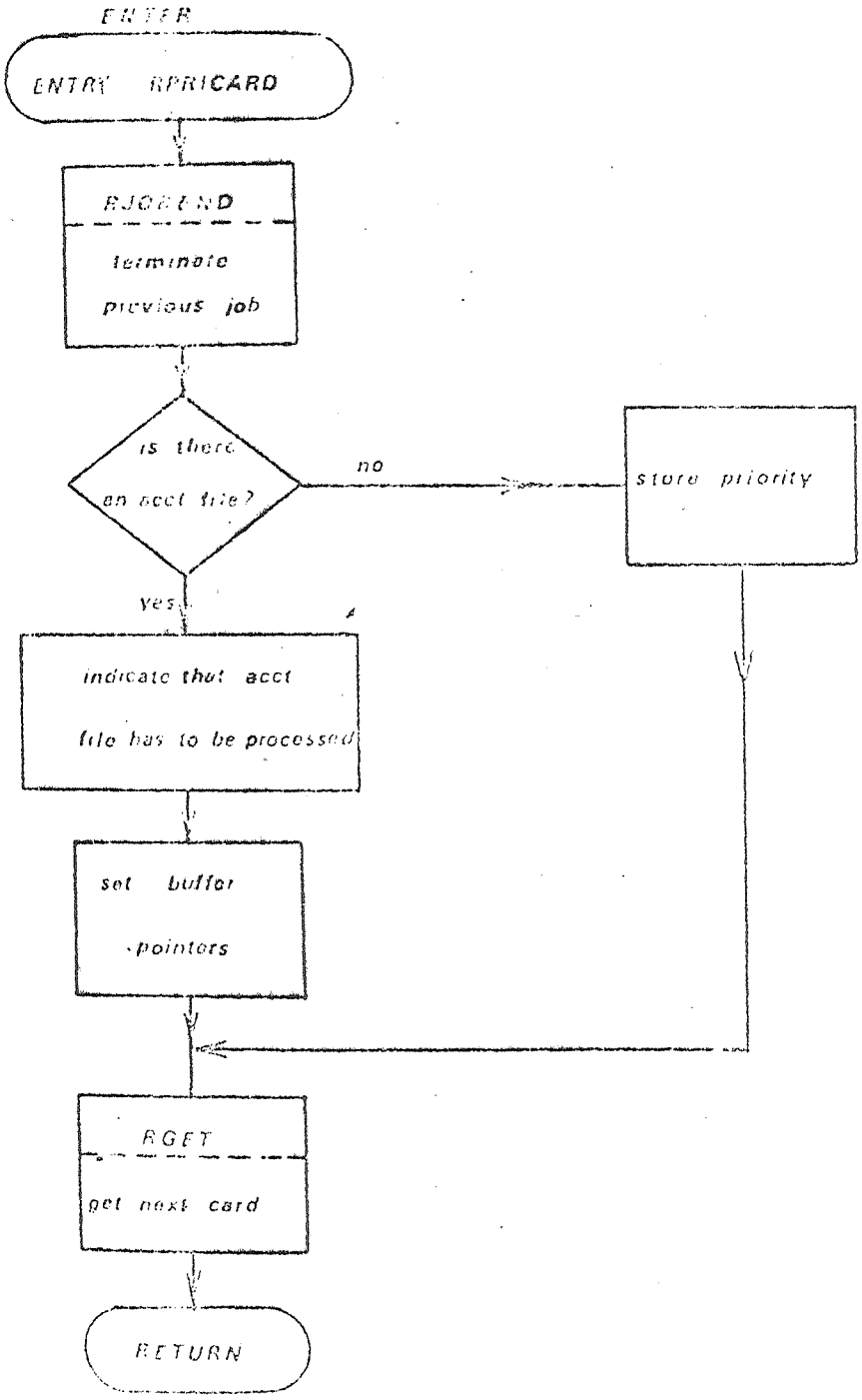
```
*****************************************************************
*  CPINIT UPDATES TO SAVE ACTIVE USER'S ACCOUNTING              *
*                  RECORDS AT SYSTEM REIPL                      *
*****************************************************************


***  CALL VAM TO PUT THE RESTORED ACCOUNTING CARDS IN THE BUFFER
          EXTRN  VAM
          L      R1,=V(MREALIO)            (FOR VAM)
AFTER     L      R4,0(R1)                  GET FIRST (NEXT) BLOK (FOR VAM)
          LTR    R4,R4                     IS IT THE END ?  (FOR VAM)
          BZ     WARMCONT                  YES,EXIT
          CALL   VAM                       GO TO VAM
          LR     R1,R4                     CONTINUE
          B      AFTER
WARMCONT  L      L6,=A(RMXSTART)
                                  -------
                                  -------
```

APPENDICE III

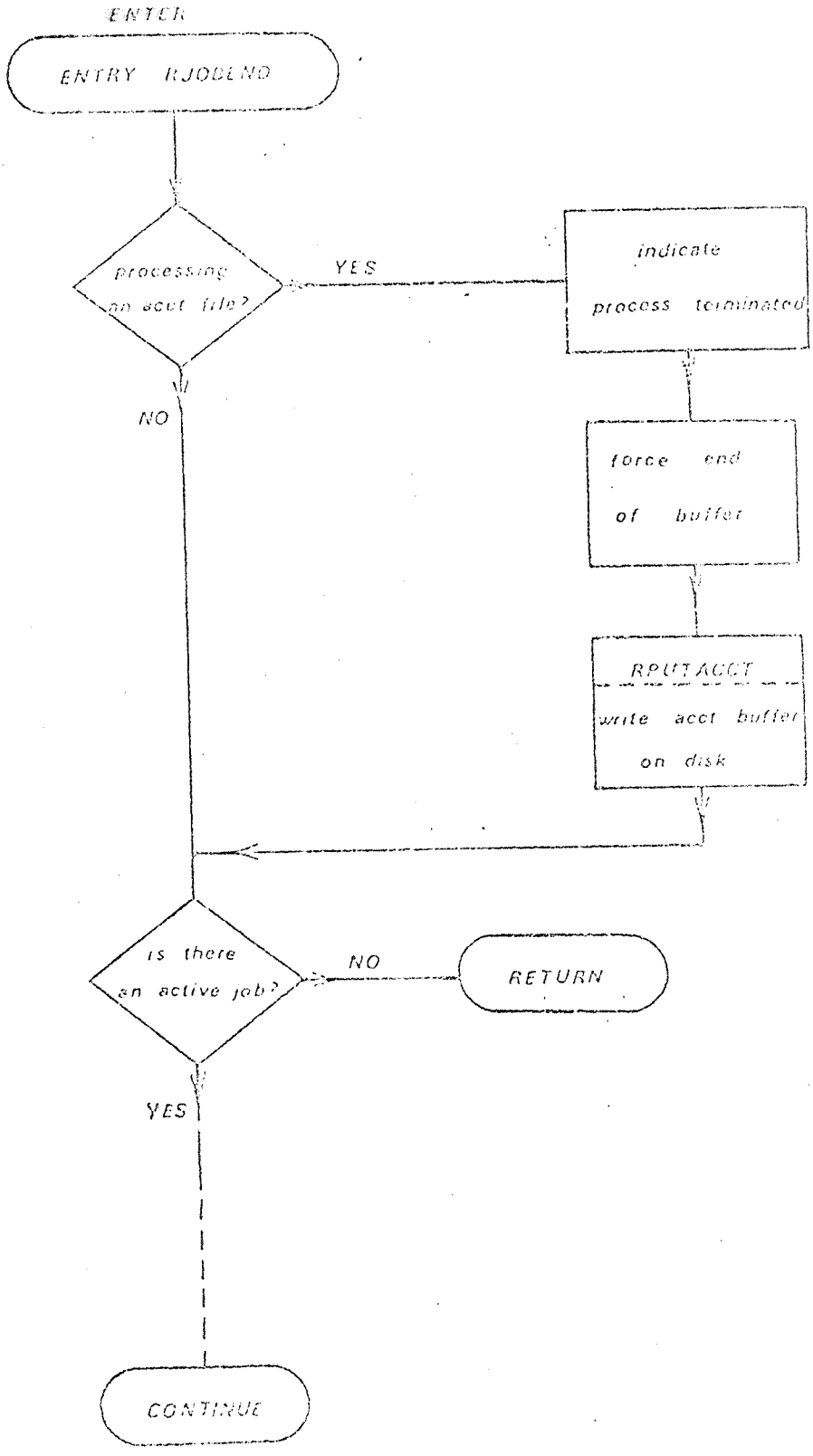
**** MAIN PROCESSOR ****

```
HASPRDR  CSECT                          DEFINE MAIN PROCESSOR RESIDENT
RCRDOUT  BAL     RL1,RPUT               PUT CARD IN OUTPUT BUFFER
         CLI     SWACC,X'01'            TEST FOR ACCOUNTING SWITCH      PERFCP
         BNE     *+8                    BRANCH IF OFF                   PERFCP
         BAL     RL1,RPUTACCT           PUT CARD IN ACCOUNTING BUFFER   PERFCP
RNXTCRD  BAL     RL1,RGET               GET NEXT CARD
         B       RDREND                 BRANCH IF EOF
         TM      RDRSW,RJCLSW+RXBJOBSW  TEST CURRENT MODE
                    -------
                    -------
         $WTO    RMESSAGE,L'RFLMSG+8 ,JOB=NO,    ISSUE FLUSHING MESSAGE
                 ROUTE=$LOG-$UR,CLASS=$NORMAL,PRI=$ST
         MVI     RPRIORTY,C'*'          RESET PRIORITY
         NI      RDRSW,255-RJFLUSH      TURN OFF FLUSH SWITCH
         CLI     SWACC,X'01'            TEST FOR ACCOUNTING SWITCH      PERFCP
         BNE     *+8                    BRANCH IF OFF                   PERFCP
         BAL     RL1,RPUTACCT           PUT CARD IN ACCOUNTING BUFFER   PERFCP
         B       RNXTCRD                GET NEXT CARD
```

```
*********************************************************************************
*                                                                               *
*                      HASP PRIORITY CARD PROCESSING ROUTINE                    *
*                                                                               *
*********************************************************************************


RPRICARD NULL
         BAL     RL1,RJOBEND        TERMINATE PREVIOUS JOB          PERFCP
         CLI     15(RPI),X'FF'      TEST FOR ACCOUNTING FILE        PERFCP
         BNE     PICKUP             BRANCH IF NO                    PERFCP
         MVI     SWACC,X'01'        SET ON THE SWITCH               PERFCP
         LA      R1,RBUFACT         LOAD ADDR OF ACCOUNTING BUFFER  PERFCP
         ST      R1,RBO             STORE THIS ADDRESS              PERFCP
         ST      R1,RBINEXT         SET POINTER TO THE BEGINNING    PERFCP
         LA      R1,259(R1)         SET ADDRESS OF LAST             PERFCP
         ST      R1,RBOEND          CARD TO BE WRITTEN              PERFCP
         B       SWITCH             BRANCH TO SET RDRSW             PERFCP
PIKUP    IC      RW,15(,RPI)        PICK UP COLUMN 16               PERFCP
         CLI     16(RPI),C' '       TEST COLUMN 17 FOR BLANK
         BE      RPRIBL             BRANCH IF COLUMN 17 IS BLANK
         IC      RW,16(,RPI)        NO, PICK UP COLUMN 17
         LA      RW,10(,RW)         ADD 10 TO CHARACTER
RPRIBL   STC     RW,RPRIORITY       SAVE PRIORITY CHARACTER
SWITCH   NI      RDRSW,255-ROSINSW  RESET O/S INPUT DATA SET SW.  PERFCP
         OI      RDRSW,RJFLUSH+RJCLSW     SET FLUSH AND JCL SWITCHES
         BAL     RL1,RGET           GET NEXT CARD
         B       RPRIEOF            BRANCH IF END OF FILE
         LA      R12,RSCANCHK       LOAD ADDRESS OF EXIT
       $RETURN                        AND RETURN
RPRIEOF  LA      RL2,RDREND         LOAD ADDRESS OF EOF EXIT
       $RETURN                              AND RETURN
```

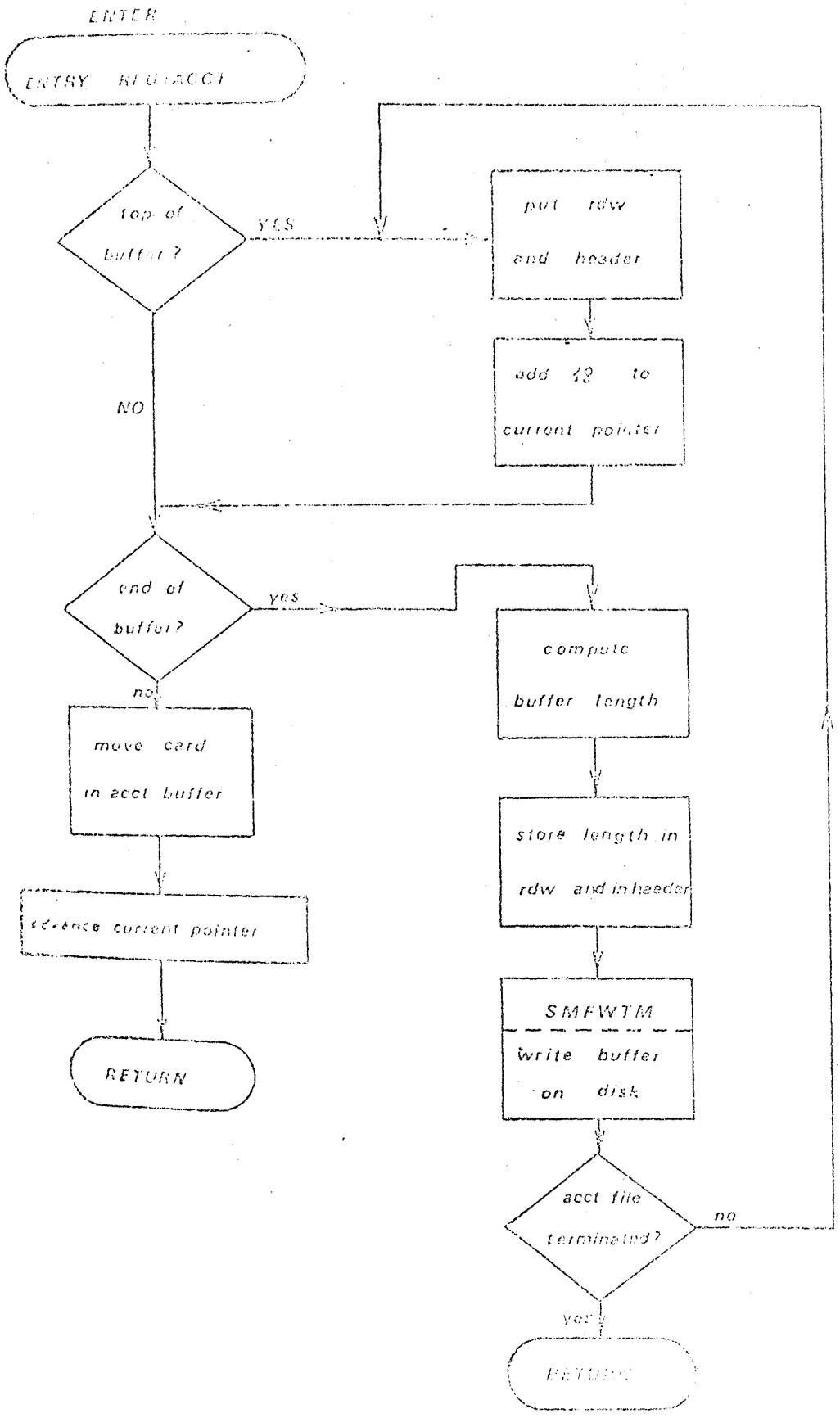

ENTER
ENTRY RPRICARD
RJOBEND
terminate
previous job
is there
an acct file?
no
store priority
yes
indicate that acct
file has to be processed
set buffer
pointers
RGET
get next card
RETURN

```
*********************************************************************************
*                                                                               *
*         RJOBEND -- SUBROUTINE TO COMPLETE JOB INPUT PROCESSING                *
*                                                                               *
*               FUNCTIONS -- 1) TEST FOR ACTIVE JOB                             *
*                            2) TERMINATE JOB                                   *
*                            3) PLACE JOB IN EXECUTION QUEUE                    *
*                                                                               *
*         LINK REGISTER -- RL1                                                  *
*                                                                               *
*         EXTERNAL ROUTINE -- RJOBTERM                                          *
*                                                                               *
*********************************************************************************


RJOBEND  CLI     SWACC,X'01'     TEST FOR ACCOUNTING SWITCH          PERFCP
         BNE     JCTTEST         BRANCH IF OFF                       PERFCP
         MVI     SWACC,X'00'     SET OFF ACCOUNTING SWITCH           PERFCP
         MVC     RBOEND,RBONEXT  FORCE END OF BUFFER                 PERFCP
         ST      RL1,RLSAVE1     SAVE LINK REGISTER                  PERFCP
         BAL     RL1,RPUTACCT    WRITE SMF RECORD                    PERFCP
         L       RL1,RLSAVE1     RESTORE LINK REGISTER               PERFCP
         LTR     JCT,JCT         TEST FOR ACTIVE JOB                 PERFCP
         BCR     Z,RL1           RETURN IF NO JCT
                 --------
                 --------
```


ENTER
ENTRY RJOBEND
processing an acct file?
YES
indicate process terminated
NO
force end of buffer
RPUTACCT
write acct buffer on disk
is there an active job?
NO
RETURN
YES
CONTINUE

RPUTACCT

```
*********************************************************************** PERFCP
*                                                                     * PERFCP
*         RPUTACCT  --  SUBROUTINE TO WRITE SMF TYPE 128 RECORD       * PERFCP
*                                                                     * PERFCP
*               FUNCTIONS --  1) IF ACCT BUFFER IS FULL,              * PERFCP
*                                SET UP RDW,WRITE BUF. ON DISK,       * PERFCP
*                                INITIALIZE ACCT BUFFER               * PERFCP
*                             2) ADD CARD TO ACCT BUFFER              * PERFCP
*                                                                     * PERFCP
*               LINK REGISTER  --  RL1                                * PERFCP
*                                                                     * PERFCP
*********************************************************************** PERFCP


RPUTACCT L       R1,RBONEXT            GET OUTPUT POINTER            PERFCP
         CLR     RBO,R1               TEST FOR ACCT BUFFER BEGINN    PERFCP
         BNE     EOBTEST            BRANCH IF NOT                    PERFCP
         MVC     RBUFACT(4),=X'000E0000' PUT HEADER RDW              PERFCP
         MVC     RBUFACT+4(14),=X'048000000000099367CC1C3C3E3'       PERFCP
*           RECORD HEADER IS        TY( RECL )( DATE )A C C T        PERFCP
         LA      R1,18(R1)            ADVANCE OUTPUT POINTER        PERFCP
         C       R1,RBOEND            TEST FOR END OF BUFFER        PERFCP
         BNL     WRTMAN               BRANCH IF END OF BUFFER       PERFCP
         MVC     0(80,R1),0(RPI)      MOVE CP ACCOUNTING CARD       PERFCP
         LA      R1,80(R1)            ADVANCE                       PERFCP
         ST      R1,RBONEXT             OUTPUT POINTER              PERFCP
         BR      RL1                  RETURN                        PERFCP
         SR      R1,RBO               GET FILLED BUFFER LENGTH      PERFCP
         STH     R1,RBUFACT           SET UP RDW                    PERFCP
         SH      R1,=X'0004'          SUBTRACT RDW LENGTH           PERFCP
         STH     R1,RBUFACT+8         MOVE RECORD LENGTH            PERFCP
         L       1,RBO                LOAD BUFFER ADDRESS           PERFCP
         SMFWTM (1)                   WRITE SMF RECORD              PERFCP
         BNE     DROPRBO              BRANCH IF OFF                 PERFCP
         B       INBUFF               BRANCH TO INITIALIZE BUFFER   PERFCP
         BR      RL1                  RETURN                        PERFCP
```


ENTER
ENTRY REGACCT
top of buffer?
YES
NO
put rdw and header
add 18 to current pointer
end of buffer?
yes
no
compute buffer length
store length in rdw and in header
SMFWTM
write buffer on disk
acct file terminated?
no
yes
RETURN
move card in acct buffer
advance current pointer
RETURN