

1.2
SORT2 SORT -- PHASE 2 MACRO V05.01b Monday 06-May-85 17:46
Table of contents

3-	1	PASS 2 INITIALIZATION
4-	1	DECIDE STRATEGY FOR NEXT MERGE PASS
5-	1	INITIALIZE FOR NEXT MERGE PASS
6-	1	DO MERGE PASS
14-	1	SUBROUTINES

```

1          .TITLE  SORT2  SORT -- PHASE 2
2 000000   .CSECT  SORT2
3          .ENABL  LC
4          000000' P2BASE =
5          ;
6          ; THIS SECTION OF SORT REPEATEDLY MERGES THE SORTED
7          ; STRINGS PRODUCED BY PASS 1 UNTIL ONLY A SINGLE SORTED
8          ; STRING REMAINS. THE FINAL MERGE PASS WRITES THIS STRING
9          ; TO THE OUTPUT FILE.
10         ;
11         ; Copyright (c) 1980, 1981.
12         ; S&H Computer Systems, Inc.
13         ; Nashville, Tennessee USA
14         ;
15         ; All rights reserved.
16         ; This software may not be sold, distributed or otherwise made
17         ; available for use except as licensed by S&H Computer Systems, Inc.
18         ;
19         ; COBOL-Plus, SORT-Plus, RTSORT, TSX and TSX-Plus are registered
20         ; trademarks of S&H Computer Systems, Inc.
21         ;
22         ;
23         ;
24         ; MACRO TO CAUSE ABORT
25         ;
26         .MACRO  XXX      COD
27 XXX'COD =
28         MOV     #COD',R0
29         CALL    ABORT
30         .ENDM    XXX
31         ;
32         ; GLOBAL DEFINITIONS.
33         ;
34         .GLOBL      SORTP2,P2TOP,P2SIZ
35         ;
36         ; GLOBAL REFERENCES.
37         ;
38         .GLOBL  RWDLEN,ABORT,FD3,CISRT2
39         .GLOBL  KTYPE,TOPMEM,RECLN,EOFBUF
40         .GLOBL  FILRN,FILST,MXFILS,MXFIL2
41         .GLOBL  FILNB,OBFEND,OBUFSZ
42         .GLOBL  NUMKY2,OBUF1,OBUF2,OBUFSZ
43         .GLOBL  NXTBLK,ORPOS,FLUSH,COMPAR
44         .GLOBL  DSKCVT,FLBLK,FILFL
45         .GLOBL  CUROFL,FT$T,KSTRT
46         .GLOBL  BLKFC,P2TERM,OWCNT
47         .GLOBL  ITYPE,RT$CR,RT$VLN
48         .GLOBL  RT$CBL,RT$DBL,RT$FTN,RT$TXT
49         .GLOBL  RT$OS,RT$OR,RT$OX,RT$F11
50         .GLOBL  RT$RA,RT$RB
51         .GLOBL  OTYPE,F$INPL
52         .GLOBL  UEOF,SFLAGS
53         ;
54         ;
55         ; FILE MANAGEMENT TABLES.
56         ;
57 000050   MXMRG = 40. ; ABSOLUTE MAX # OF FILES TO BE MERGED.

```

58	000120	MXMRG2 =	2#MXMRG	
59				
60	000000	MRGLST: .BLKW	MXMRG	;LIST OF FILES TO MERGE
61	000120	LOC: .BLKW	MXMRG	;ADDRESS OF RECORD(J).
62	000240	LOSER: .BLKW	MXMRG	;INDEX TO LOSER BELOW THIS NODE.
63	000360	FE: .BLKW	MXMRG	;EXTERNAL BLOCK FORWARD LINK.
64	000500	FI: .BLKW	MXMRG	;INTERNAL BLOCK FORWARD LINK.
65	000620	RN: .BLKW	MXMRG	;RUN # THIS RECORD WILL GO IN.
66	000740	IRPOS: .BLKW	MXMRG	;CUR POINTER INTO BUFFER
67	001060	MRCBA: .BLKW	MXMRG	;ADDRESS OF FILE BUFFER
68				
69		; GENERAL DATA.		
70	000015	CR =	15	;ASCII CARRIAGE RETURN
71	000012	LF =	12	;ASCII LINE FEED
72	000032	CTRLZ =	32	;CTRL-Z DIBOL EOF CHARACTER
73	001200	GRNTBL: .WORD	0	;ADDRESS OF START OF GRANULE TBL.
74	001202	GRNEND: .WORD	0	;END OF GRANULE TABLE.
75	001204	NFILE2: .WORD	0	;MAX # OF FILES WHICH CAN BE MERGED.
76	001206	NUMMRG: .WORD	0	;# OF FILES TO MERGE NEXT PASS.
77	001210	EMTBLK: .BLKW	10	;SPACE FOR EMT ARGS
78	001230	RC: .WORD	0	;RUN # OF CURRENT RUN.
79	001232	Q: .WORD	0	;INDEX TO CURRENT RECORD.
80	001234	RQ: .WORD	0	;RUN # CURRENT RECORD WILL GO IN.
81	001236	NXDBLK: .WORD	0	;ADDRESS OF NXT BLOCK TO OUTPUT.
82	001240	OFI INK: .WORD	0	;LINK TO PREV RUNS IN FILE
83				
84	001242	FINFLG: .BYTE	0	;1==>THIS IS FINAL MERGE.
85		.EVEN		

1			.SBTTL PASS 2 INITIALIZATION	
2				
3			; ALLOCATE AVAILABLE CORE SPACE.	
4				
5	001244	004767	000000G	SORTP2: CALL CISRT2 ; CHECK IN FOR OVERLAY BREAKPOINT
6	001250	012701	006424'	MOV #P2TOP,R1 ; GET START OF FREE MEMORY SPACE.
7				
8				; SET UP GRANULE BIT TABLE.
9				; A 1 IN THIS TABLE INDICATES THE CORRESPONDING
10	001254	010167	177720	; DISK GRANULE IS FREE.
11	001260	016702	000000G	MOV R1,GRNTBL ; SET START OF TABLE ADDRESS.
12	001264	062702	000017	MOV NXTBLK,R2 ; GET # OF DISK BLKS TO ALLOCATE.
13	001270	006202		ADD #15,R2 ; CONVERT # OF BITS TO # OF WORDS.
14	001272	006202		ASR R2
15	001274	006202		ASR R2
16	001276	060102		ASR R2
17	001300	010267	177676	ADD R1,R2 ; GET ADDRESS OF END OF TABLE.
18	001304	005021		MOV R2,GRNEND
19	001306	020102		1\$: CLR (R1)+ ; MARK ALL GRANULES AS BUSY.
20	001310	103775		CMP R1,R2
21				BLO 1\$
22	001312	012702	000002	; THE LAST 2 GRANULES ARE ACTUALLY FREE
23	001316	016704	000000G	MOV #2,R2
24	001322	004767	004662	MOV NXTBLK,R4
25	001326	005304		6\$: CALL RELGRN ; FREE THOSE GRANULES.
26	001330	005302		DEC R4
27	001332	003373		DEC R2
28				BGT 6\$
29	001334	016703	000000G	; ALLOCATE FILE I/O BUFFERS.
30	001340	010167	000000G	MOV OBUFSZ,R3 ; GET SIZE OF TEMP FILE BUFFER.
31	001344	060301		3\$: MOV R1,OBUF1 ; ALLOCATE OUTPUT BUFFERS.
32	001346	010167	000000G	ADD R3,R1
33	001352	010167	000000G	MOV R1,OBFEND ; END OF BUFF1
34	001356	060301		MOV R1,OBUF2
35				ADD R3,R1
36				
37				; ALLOCATE FILE INPUT BUFFERS
38	001360	005002		
39	001362	010162	000120'	4\$: CLR R2 ; COUNT # OF FILES ALLOCATED
40	001366	066701	000000G	MOV R1,LOC(R2) ; ALLOCATE RECORD BUFFER
41	001372	010162	001060'	ADD RWDLEN,R1
42	001376	066701	000000G	MOV R1,MRCBA(R2) ; ALLOCATE FILE BUFFER
43	001402	020167	000000G	ADD OBUFSZ,R1
44	001406	101005		CMP R1,TOPMEM ; HIT TOP OF MEMORY?
45	001410	062702	000002	BHI 2\$; BR IF YES
46	001414	020227	000120	ADD #2,R2 ; COUNT ANOTHER FILE ALLOCATED
47	001420	103760		CMP R2,#MXMRG2 ; ALLOCATED MAX ALLOWED?
48	001422	006202		BLO 4\$; BR IF NOT
49	001424	010267	177554	2\$: ASR R2 ; GET # OF FILES
50	001430	005067	177600	MOV R2,NFILE2 ; SAVE # OF FILES ALLOCATED
51	001434	005067	177570	CLR RQ ; INIT MISC PARAMETERS
52	001440	005067	177566	CLR RC
53	001444	005067	177566	CLR Q
54	001450	105067	177566	CLR NXOBLK
				CLRB FINFLG

```

1          .SBTTL  DECIDE STRATEGY FOR NEXT MERGE PASS
2
3          ; THIS SECTION OF SORT DETERMINES THE STRATEGY
4          ; TO BE CARRIED OUT BY THE NEXT MERGE PASS.
5          ; ON ENTRY TO DOMRG THE CELL NUMMRG CONTAINS
6          ; THE NUMBER OF FILES TO MERGE.  FINFLG IS SET
7          ; TO 1 IF THIS IS THE FINAL MERGE.
8
9 001454 005002          STRAT:  CLR      R2          ; WILL GET MAX # OF RUNS IN ANY FILE
10 001456 005003          CLR      R3          ; WILL GET SUM OF RUNS IN ALL FILES.
11 001460 012704 0000000  MOV      #FILRN,R4      ; TABLE WITH # OF RUNS IN EACH FILE.
12 001464 010405          MOV      R4,R5
13 001466 062705 0000000  ADD      #MXFIL2,R5      ; GET ADDRESS OF END OF TABLE.
14 001472 012401          2$:  MOV      (R4)+,R1      ; GET # OF RUNS IN THIS FILE.
15 001474 020201          CMP      R2,R1          ; SAVE HIGHEST NUMBER.
16 001476 103001          BHS      1$
17 001500 010102          MOV      R1,R2
18 001502 060103          1$:  ADD      R1,R3          ; GET TOTAL NUMBER OF RUNS.
19 001504 020405          CMP      R4,R5          ; FINISHED YET?
20 001506 103771          BLO      2$
21 001510 020227 0000002  CMP      R2,#2          ; MAX # OF RUNS > 2?
22 001514 101030          BHI      MAJMRG          ; BRANCH IF YES.
23 001516 001413          BEQ      PENMRG
24 001520 020367 177460  CMP      R3,NFILE2      ; CAN WE DO FINAL MERGE?
25 001524 101010          BHI      PENMRG          ; BRANCH IF NOT.
26          ; THIS IS FINAL MERGE.
27 001526 105267 177510  INCB     FINFLG          ; SET FINAL MERGE FLAG.
28 001532 010367 177450  MOV      R3,NUMMRG        ; SET # OF FILES TO MERGE.
29 001536 012767 0000000 0000000  MOV      #MXFILS,CUROFL
30 001544 000442          BR       STRFIN          ; GO PERFORM MERGE.
31          ; SEE IF THIS IS NEXT TO LAST MERGE.
32 001546 016701 177432  PENMRG: MOV      NFILE2,R1
33 001552 006301          ASL      R1          ; GET 2 TIMES # OF FILES WE CAN MERGE.
34 001554 005301          DEC      R1          ; MINUS 1.
35 001556 020301          CMP      R3,R1          ; WILL NEXT PASS FINISH FILE?
36 001560 103006          BHS      MAJMRG          ; BRANCH IF NOT.
37          ; THIS IS THE NEXT TO LAST MERGE.
38          ; MERGE (SUM(RN)-NFILE2+1) FILES.
39 001562 166703 177416  SUB      NFILE2,R3
40 001566 005203          INC      R3
41 001570 010367 177412  MOV      R3,NUMMRG
42 001574 000426          BR       STRFIN          ; GO DO THE MERGE.
43          ; MERGE AS MANY FILES AS POSSIBLE ON THIS PASS.
44 001576 005001          MAJMRG: CLR      R1
45 001600 012704 0000000  MOV      #FILRN,R4      ; TABLE OF RUNS PER FILE.
46 001604 010405          MOV      R4,R5
47 001606 062705 0000000  ADD      #MXFIL2,R5      ; ADDRESS OF END OF TABLE.
48 001612 005724          2$:  TST      (R4)+      ; DOES THIS FILE HAVE SOME RUNS?
49 001614 001401          BEQ      1$          ; BRANCH IF IT'S EMPTY.
50 001616 005201          INC      R1          ; COUNT # OF FILES THAT AREN'T EMPTY.
51 001620 020405          1$:  CMP      R4,R5          ; FINISHED?
52 001622 103773          BLO      2$
53 001624 020127 0000000  CMP      R1,#MXFILS      ; MAX MERGE OF (MXFILS-1).
54 001630 103401          BLO      3$
55 001632 005301          DEC      R1
56 001634 020167 177344  3$:  CMP      R1,NFILE2      ; MAX MERGE OF (NFILE2).
57 001640 101402          BLOS     4$

```

58	001642	016701	177336		MOV	NFILE2,R1	
59	001646	010167	177334	4#:	MOV	R1,NUMMR0	;SET # OF FILES TO BE MERGED.
60	001652	006367	177330	STRFIN:	ASL	NUMMR0	;SET 2* # OF FILES TO MERGE

```

1          .SBTTL  INITIALIZE FOR NEXT MERGE PASS
2
3          ; THIS SECTION OF SORT USES THE INFORMATION SET UP BY
4          ; THE STRATEGY DECISION MAKER TO INITIALIZE TABLES WHICH
5          ; ARE USED TO DO THE ACTUAL MERGE.
6
7 001656 105767 177360      DOMRG:  TSTB      FINFLG      ; FINAL MERGE?
8 001662 001023           BNE      9$      ; BRANCH IF YES.
9
10         ; FIND WHICH FILE TO USE FOR OUTPUT.
11         ; USE THE FILE WITH THE SMALLEST NUMBER OF RUNS IN IT.
12 001664 012701 001747      MOV      #999, R1
13 001670 005002           CLR      R2
14 001672 020162 000000G    2$:     CMP      R1, FILRN(R2)
15 001676 101404           BLOS     1$
16 001700 010205           MOV      R2, R5      ; SAVE INDEX TO FILE
17 001702 016201 000000G    MOV      FILRN(R2), R1 ; GET # OF SORTED RUNS IN FILE
18 001706 001405           BEQ      10$      ; CAN'T BEAT ZERO
19 001710 062702 000002     1$:     ADD      #2, R2      ; MOVE ON TO NEXT FILE
20 001714 020227 000000G    CMP      R2, #MXFIL2 ; CHECKED ALL?
21 001720 103764           BLO      2$
22 001722 010567 000000G    10$:    MOV      R5, CURDFL ; SAVE INDEX TO OUTPUT FILE.
23 001726 005265 000000G    INC      FILRN(R5) ; ANOTHER RUN GOES TO THIS FILE.
24 001732 016701 000000G    9$:     MOV      OBUF1, R1 ; GET ADDRESS OF 1ST OUTPUT BUF.
25 001736 105767 177300     TSTB     FINFLG      ; IS THIS FINAL MERGE?
26 001742 001002           BNE      8$      ; BRANCH IF YES.
27 001744 005021           CLR      (R1)+      ; ZERO FLINK FIELD.
28 001746 005021           CLR      (R1)+      ; ZERO # OF BYTES IN BLOCK.
29 001750 010167 000000G    8$:     MOV      R1, ORPOS ; SAVE ADDRESS WHERE NXT REC GOES.
30         ; FIND THE NUMMRG/2 FILES WITH THE LARGEST NUMBER OF RUNS
31         ; TO USE AS INPUT.
32 001754 005001           CLR      R1      ; # OF FILES WE HAVE GOTTEN.
33 001756 005002           7$:     CLR      R2      ; # OF FILE TO CHECK NEXT.
34 001760 005003           CLR      R3
35 001762 012700 177777     MOV      #177777, R0 ; LOOK FOR SMALLEST # OF BLOCKS
36 001766 020362 000000G    6$:     CMP      R3, FILRN(R2) ; SAVE LARGEST NUMBER OF RUNS.
37 001772 101027           BHI      3$
38 001774 103405           BLO      11$      ; TAKE IT IF MORE RUNS
39 001776 005703           TST      R3      ; IS THIS FIRST TIME THROUGH?
40 002000 001424           BEQ      3$      ; BR IF YES
41 002002 020062 000000G    CMP      R0, FILNB(R2) ; SELECT FILE WITH SMALLEST # BLOCKS
42 002006 101421           BLOS     3$
43 002010 020267 000000G    11$:    CMP      R2, CURDFL ; IS THIS THE OUTPUT FILE?
44 002014 001416           BEQ      3$      ; IF YES THEN CAN'T USE FOR INPUT.
45 002016 005004           CLR      R4      ; SEE IF FILE IS ALREADY IN IN LIST.
46 002020 020401           4$:     CMP      R4, R1      ; AT END OF LIST?
47 002022 001406           BEQ      5$      ; BRANCH IF YES.
48 002024 020264 000000'    CMP      R2, MRGLST(R4) ; SEE IF IN LIST.
49 002030 001410           BEQ      3$      ; IF YES THEN DON'T GET IT AGAIN.
50 002032 062704 000002     ADD      #2, R4
51 002036 000770           BR      4$
52 002040 016203 000000G    5$:     MOV      FILRN(R2), R3 ; SAVE # OF RUNS IN THIS FILE.
53 002044 016200 000000G    MOV      FILNB(R2), R0 ; GET # OF BLOCKS IN FILE
54 002050 010205           MOV      R2, R5      ; SAVE INDEX TO FILE.
55 002052 062702 000002     3$:     ADD      #2, R2      ; MOVE ON TO NEXT FILE
56 002056 020227 000000G    CMP      R2, #MXFIL2 ; CHECKED ALL FILES?
57 002062 103741           BLO      6$      ; BRANCH IF NOT.
58 002064 010561 000000'    MOV      R5, MRGLST(R1) ; ADD FILE TO INPUT LIST.

```

```

58 002070 062701 000002          ADD    #2,R1          ;COUNT UP 1 MORE FILE
59 002074 020167 177106          CMP     R1,NUMMRG        ;GOT ALL WE NEED?
60 002100 103726                  BLO     7#             ;BRANCH IF NOT.
61                                ;
62                                ; AT THIS POINT WE ARE ABOUT TO DO A MERGE FROM NUMMRG
63                                ; FILES WHOSE INDEX NUMBERS ARE STORED IN MRGLST TO
64                                ; THE OUTPUT FILE WHOSE INDEX NUMBER IS IN CUROFL.
65                                ; FINFLG IS NON-ZERO IF THIS IS THE FINAL MERGE.
66                                ;
67                                ; SET UP MERGE TABLES.
68 002102 016702 177100  MRGSET: MOV    NUMMRG,R2        ;GET # OF FILES TO BE MERGED.
69 002106 006202          ASR     R2                ;GET 1* # OF FILES TO BE MERGED
70 002110 010205          MOV     R2,R5
71 002112 005302          DEC     R2
72 002114 010503 3$:    MOV     R5,R3                ;GET P ( NUMBER OF FILES TO MERGE).
73 002116 060203          ADD     R2,R3                ;GET P+J (J=INDEX TO FILE).
74 002120 006203          ASR     R3                ;(P+J)/2
75 002122 006303          ASL     R3                ;CONVERT TO WORD TABLE INDEX.
76 002124 010204          MOV     R2,R4                ;GET J.
77 002126 006204          ASR     R4                ;GET J/2.
78 002130 006304          ASL     R4                ;CONVERT TO WORD TABLE INDEX.
79 002132 006302          ASL     R2                ;CONVERT J TO WORD TABLE INDEX.
80 002134 010362 000360'        MOV     R3,FE(R2)        ;SET UP EXTERNAL FLINK.
81 002140 010462 000500'        MOV     R4,FI(R2)        ;SET UP INTERNAL FLINK.
82 002144 005062 000620'        CLR     RN(R2)          ;ZERO RUN # THIS REC WILL GO IN.
83 002150 010262 000240'        MOV     R2,LOSER(R2)
84 002154 006202          ASR     R2
85 002156 005302          DEC     R2
86 002160 002355          BGE     3#
87                                ; INITIALIZE FILE BUFFERS.
88 002162 005001          CLR     R1
89 002164 016103 000000' 9$:    MOV     MRGLST(R1),R3    ;GET FILE NUMBER.
90 002170 016104 001060'        MOV     MRGBA(R1),R4    ;GET ADDRESS OF BUFFER FOR FILE.
91 002174 010461 000740'        MOV     R4,IRPOS(R1)    ;INITIALIZE RECORD POINTER
92 002200 016324 000000G        MOV     FILST(R3),(R4)+ ;SET UP STARTING DISK ADDRESS.
93 002204 005014          CLR     (R4)                ;ZERO # OF BYTES IN BLOCK.
94 002206 062701 000002          ADD     #2,R1
95 002212 020167 176770          CMP     R1,NUMMRG        ;INITIALIZED ALL FILES?
96 002216 002762          BLT     9#                 ;BRANCH IF NOT.
97 002220 005067 177004          CLR     RC
98 002224 005067 177002          CLR     Q
99 002230 005067 177000          CLR     RQ
100                               ;
101                               ; SET UP OUTPUT FILE
102                               ;
103 002234 105767 177002          TSTB    FINFLG          ;IS THIS FINAL MERGE?
104 002240 001013          BNE     SR?                ;BR IF YES
105 002242 016702 000000G        MOV     CUROFL,R2        ;GET FILE INDEX #
106 002246 016267 000000G 176764 MOV     FILST(R2),OFLINK ;SAVE POINTER TO OTHER RUNS IN FILE
107 002254 004767 004022          CALL    GETORN         ;GET A FREE CLUSTER
108 002260 010162 000000G        MOV     R1,FILST(R2)    ;SET AS 1ST BLOCK OF FILE
109 002264 010167 000000G        MOV     R1,FILEL

```

```

1          .SBTTL DO MERGE PASS
2
3          ; ENTER ACTUAL MERGE ROUTINE.
4
5 002270 016701 176734 SR2: MOV RC,R1 ; DOES CURRENT RECORD GO
6 002274 020167 176734 CMP R1,R0 ; IN THIS RUN?
7 002300 001413 BEQ SR3 ; BRANCH IF YES.
8 002302 026727 176726 000001 CMP R0,#1 ; HAVE WE HIT THE END OF FILE?
9 002310 101002 BHI 2$ ; BR IF YES
10 002312 005701 TST R1 ; IS THE INITIALIZATION RUN?
11 002314 001402 BEQ 1$ ; BRANCH IF YES
12 002316 000167 001006 2$: JMP SRFIN ; NO -- WE'RE FINISHED
13 002322 016767 176706 176700 1$: MOV R0,RC ; ADVANCE TO NEXT RUN.
14 002330 005767 176700 SR3: TST R0 ; INITIALIZATION RUN?
15 002334 001002 BNE 1$ ; BRANCH IF NOT
16 002336 000167 000144 JMP SR4
17          ; OUTPUT RECORD(Q).
18 002342 105767 176674 1$: TSTB FINFLG ; FINAL OUTPUT MERGE?
19 002346 001051 BNE FINMOV ; BRANCH IF YES.
20
21          ; WE ARE MERGING TO ANOTHER TEMP FILE
22          ; MOVE RECORD TO OUTPUT BUFFER.
23
24 002350 016702 176656 TMPMOV: MOV Q,R2 ; GET MERGE LIST FILE INDEX #
25 002354 016203 000120' MOV LOC(R2),R3 ; GET ADDR OF WINNER RECORD
26 002360 016702 000000G MOV ORPOS,R2 ; GET POINTER INTO OUTPUT BUFFER
27 002364 016705 000000G MOV OBFEND,R5 ; GET ADDRESS OF END OF BUFFER
28 002370 010304 MOV R3,R4 ; GET ADDRESS OF END OF RECORD
29 002372 066704 000000G ADD RECLN,R4
30 002376 020304 2$: CMP R3,R4 ; MOVED ALL?
31 002400 103031 BHS 4$ ; BR IF FINISHED MOVE
32 002402 020205 CMP R2,R5 ; HIT END OF BUFFER?
33 002404 103421 BLO 1$ ; BR IF NOT
34          ; OUTPUT BUFFER IS FULL -- WRITE IT OUT
35 002406 016700 000000G MOV OBUF1,R0 ; GET ADDRESS OF BUFFER
36 002412 160002 SUB R0,R2 ; CALC # OF CHARS IN BUFFER
37 002414 010260 000002 MOV R2,2(R0) ; STORE # BYTES IN BUFFER HEADER
38 002420 004767 003656 CALL GETGRN ; GET FREE DISK BLOCK
39 002424 010167 000000G MOV R1,FLBLK ; SET FILE FLINK
40 002430 004767 000000G CALL FLUSH ; WRITE OUT BUFFER
41 002434 016702 000000G MOV OBUF1,R2 ; POINT TO START OF NEW RECORD
42 002440 062702 000004 ADD #4,R2 ; SKIP OVER BUFFER HEADER WORDS
43 002444 016705 000000G MOV OBFEND,R5 ; GET ADDRESS OF END OF BUFFER
44          ; CONTINUE MOVING RECORD
45 002450 112322 1$: MOVB (R3)+,(R2)+ ; MOVE RECORD TO FILE BUFFER
46 002452 001351 BNE 2$ ; BR IF NON-NULL CHAR
47 002454 032767 000000G 000000G BIT #RT$VLN, ITYPE ; ARE RECORDS VARIABLE LENGTH?
48 002462 001745 BEQ 2$ ; BR IF NOT
49 002464 010267 000000G 4$: MOV R2,ORPOS ; SAVE OUTPUT BUFFER POINTER
50 002470 000406 BR SR4
51
52          ; MOVE RECORD TO FINAL OUTPUT BUFFER
53
54 002472 016702 176534 FINMOV: MOV Q,R2 ; GET MERGE LIST INDEX #
55 002476 016201 000120' MOV LOC(R2),R1 ; GET ADDRESS OF WINNER RECORD
56 002502 004767 002260 CALL OUTREC ; MOVE RECORD TO OUTPUT BUFFER
57

```

```

58      ; GET NEXT RECORD FROM INPUT FILE.
59      ;
60 002506 016700 176520      SR4:  MOV  Q,R0      ;GET MERGE LIST FILE INDEX #
61 002512 016001 000120'    MOV  LOC(R0),R1    ;GET ADDRESS OF RECORD BUFFER
62 002516 016002 000740'    MOV  IRPOS(R0),R2   ;GET POINTER INTO INPUT BUFFER
63 002522 016003 001060'    MOV  MRQBA(R0),R3   ;GET POINTER TO INPUT BUFFER
64 002526 066303 000002      ADD  2(R3),R3      ;GET POINTER TO END OF BUFFER
65 002532 010105            MOV  R1,R5          ;GET ADDR OF END OF RECORD
66 002534 066705 000000G    ADD  RECLN,R5
67 002540 020105      3$:  CMP  R1,R5          ;MOVED ALL OF RECORD?
68 002542 103173          BHS  6$             ;BR IF YES
69 002544 020203      4$:  CMP  R2,R3          ;HIT END OF BUFFER?
70 002546 103557          BLO  2$             ;BR IF NOT
71      ; HIT END OF BUFFER -- SEE IF THERE ARE MORE RECORDS IN SORTED RUN
72 002550 016700 176456      MOV  Q,R0          ;GET MERGE LIST FILE INDEX #
73 002554 016002 001060'    MOV  MRQBA(R0),R2   ;POINT TO FILE BUFFER
74 002560 011203          MOV  @R2,R3          ;GET FILE FLINK FROM HEADER
75 002562 100570          BMI  ENDRUN          ;BR IF END OF SORTED RUN
76      ; READ IN NEXT BLOCK FROM FILE
77 002564 010304          MOV  R3,R4          ;RELEASE NEW GRANULE
78 002566 004767 003416      CALL  RELGRN
79 002572 016004 000000'    MOV  MRGLST(R0),R4  ;GET REAL FILE INDEX #
80 002576 005364 000000G    DEC  FILNB(R4)      ;SAY 1 LESS BLOCK IN FILE
81 002602 004767 000000G    CALL  DSKCVT       ;CVT CLUSTER # TO FILE & BLOCK #
82 002606          $READ  R4,R2,QWCNT,R3      ;START READ FOR NEXT BLOCK
83 002760          $WAIT  R4                  ;WAIT FOR COMPLETION
84 003072 010203          MOV  R2,R3          ;GET ADDRESS OF BUFFER
85 003074 066203 000002      ADD  2(R2),R3      ;GET ADDRESS OF END OF BUFFER
86 003100 062702 000004      ADD  #4,R2        ;GET ADDR OF START OF DATA
87 003104 000617          BR   4$
88      ; CONTINUE MOVING RECORD
89 003106 112221      2$:  MOVB  (R2)+,(R1)+
90 003110 001213          BNE  3$             ;BR IF NON-NULL RECORD
91 003112 032767 000000G 000000G    BIT  #RT$VLN,ITYPE ;ARE RECORDS VARIABLE LENGTH?
92 003120 001607          BEQ  3$             ;BR IF NOT
93      ; PAD END OF VARIABLE LENGTH RECORDS WITH NULLS
94 003122 020105      7$:  CMP  R1,R5          ;HIT END OF RECORD?
95 003124 103002          BHS  6$             ;BR IF YES
96 003126 105021          CLRB  (R1)+          ;FILL WITH NULLS
97 003130 000774          BR   7$
98      ; FINISHED RECORD MOVE
99 003132 016700 176074      6$:  MOV  Q,R0          ;GET MERGE LIST INDEX #
100 003136 010260 000740'    MOV  R2,IRPOS(R0)  ;SAVE POINTER INTO BUFFER
101 003142 000414          BR   SR4L
102      ;
103      ; HIT END OF SORTED RUN
104      ;
105 003144 012767 000002 176062  ENDRUN: MOV  #2,RQ      ;SAY NO MORE RECORDS FROM FILE
106 003152 016004 000000'    MOV  MRGLST(R0),R4  ;GET FILE INDEX NUMBER
107 003156 042703 100000      BIC  #100000,R3    ;CLEAR EOF FLAG
108 003162 010364 000000G    MOV  R3,FILST(R4)  ;SAVE NEW FILE START ADDRESS
109 003166 005364 000000G    DEC  FILRN(R4)      ;SAY ONE LESS RUN IN FILE
110 003172 000403          BR   SR5
111      ;
112      ; PUT THIS RECORD IN RUN #1
113 003174 012767 000001 176032  SR4L:  MOV  #1,RQ
114      ; COMPARE NEW RECORD WITH PREVIOUS ONES.

```

115	003202	016701	176024	SR5:	MOV	Q,R1	;GET INDEX TO NEW RECORD.
116	003203	016104	000360'		MOV	FE(R1),R4	;GET INDEX TO FATHER (T).
117	003212	005704		SR6:	TST	R4	;AT HEAD OF TREE?
118	003214	001002			BNE	2\$	;BRANCH IF NOT
119	003216	000167	177046		JMP	SR2	
120	003222	016703	176006	2\$:	MOV	RQ,R3	;IS THIS INITIALIZATION RUN?
121	003226	001435			BEG	SR7	;BRANCH IF YES.
122	003230	020364	000620'		CMP	R3,RN(R4)	;RQ:RN(T).
123	003234	101014			BHI	1\$	;BR IF RQ>RN(T).
124	003236	001031			BNE	SR7	
125	003240	016401	000240'		MOV	LOSER(R4),R1	;GET LOSER(T).
126	003244	016101	000120'		MOV	LOC(R1),R1	;GET LOC(LOSER(T)).
127	003250	016703	175756		MOV	Q,R3	
128	003254	016302	000120'		MOV	LOC(R3),R2	;GET LOC(Q).
129	003260	004777	0000006		CALL	@COMPAR	;LOSER(T):Q.
130	003264	002016			BGE	SR7	;BR IF LOSER(T)>=Q.
131							; RECORD(Q) BECOMES NEW LOSER AT THIS LEVEL.
132	003266	016701	175742	1\$:	MOV	RQ,R1	
133	003272	016467	000620' 175734		MOV	RN(R4),RQ	;RN(T)-->RQ.
134	003300	010164	000620'		MOV	R1,RN(R4)	;RQ-->RN(T).
135	003304	016703	175722		MOV	Q,R3	
136	003310	016467	000240' 175714		MOV	LOSER(R4),Q	;LOSER(T)-->Q.
137	003316	010364	000240'		MOV	R3,LOSER(R4)	;Q-->LOSER(T).
138							; MOVE UP TREE. (T IS ADVANCED TO NEXT HIGHER NODE)
139	003322	016404	000500'	SR7:	MOV	FI(R4),R4	;FI(T)-->T.
140	003326	000731			BR	SR6	

```

1
2
3
4 003330 105767 175706 SRFIN: TSTB FINFLG ; WAS THIS THE LAST MERGE?
5 003334 001020 BNE FINCLS ; BRANCH IF IT WAS.
6 ; CLOSE OUT TEMP FILE.
7 003336 016701 0000000 MOV OBUF1,R1 ; SET END OF RUN FLAG.
8 003342 016702 0000000 MOV ORPOS,R2 ; GET CURRENT BUFFER INSERT POINTER
9 003346 160102 SUB R1,R2 ; CALCULATE # OF BYTES IN BUFFER
10 003350 010261 0000002 MOV R2,2(R1) ; STORE BYTE COUNT INTO BUFFER HEADER
11 003354 052711 100000 BIS #C'D1000000,(R1)
12 003360 016767 175654 0000000 MOV OFLINK,FLBLK ; CHAIN TO PREVIOUS RUNS IN FILE
13 003366 004767 0000000 CALL FLUSH ; WRITE OUT LAST BLOCK.
14 003372 000167 176056 JMP STRAT ; DETERMINE FUTURE STRATEGY.
15
16 ; Close out the final output (result) file.
17
18 003376 016704 0000000 FINCLS: MOV ORPOS,R4 ; GET CURRENT OUTPUT BUFFER POINTER
19 003402 016705 0000000 MOV OBFEND,R5 ; ADDRESS OF END OF OUTPUT BUFFER
20
21 ; See if we need to insert line-feed after last record.
22
23 003406 016700 0000000 MOV OTYPE,R0 ; GET OUTPUT FILE TYPE FLAGS
24 003412 032700 0000000 BIT #RT$CBL+RT$FTN,R0 ; COBOL OR FORTRAN FILE?
25 003416 001412 BEQ 2$ ; BR IF NOT
26 003420 032700 0000000 BIT #RT$RA,R0 ; RECORDING MODE ASCII?
27 003424 001407 BEQ 2$ ; BR IF NOT
28 003426 032700 0000000 BIT #RT$OS,R0 ; ORGANIZATION SEQUENTIAL?
29 003432 001404 BEQ 2$ ; BR IF NOT
30 003434 112703 000012 MOVB #LF,R3 ; INSERT LF
31 003440 004767 002502 CALL OUTCHR
32
33 ; See if we should insert control-z at end of file.
34
35 003444 016700 0000000 2$: MOV OTYPE,R0 ; GET FLAGS
36 003450 032700 0000000 BIT #RT$DBL,R0 ; DIBOL FILE?
37 003454 001413 BEQ FILLST ; BR IF NOT
38 003456 032700 0000000 BIT #RT$OS,R0 ; ORGANIZATION SEQUENTIAL?
39 003462 001004 BNE 3$ ; BR IF YES
40 003464 032767 0000000 0000000 BIT #F$INPL,SFLAGS ; INPLACE RANDOM FILE?
41 003472 001004 BNE FILLST ; NO CONTROL-Z IF YES
42 003474 112703 000032 3$: MOVB #CTRLZ,R3 ; INSERT CONTROL-Z
43 003500 004767 002442 CALL OUTCHR

```

```

1      ;
2      ; Determine how to fill out end of current buffer.
3      ;
4 003504 016703 0000000  FILLST: MOV      OTYPE,R3      ;GET OUTPUT FILE TYPE FLAGS
5 003510 032703 0000000      BIT      #RT#OS,R3      ;ORGANIZATION = SEQUENTIAL?
6 003514 001560      BEQ      2$      ;BR IF NOT
7      ;
8      ; Output organization is sequential.
9      ;
10 003516 004767 002016  4$:      CALL      OUTBUF      ;WRITE CURRENT BUFFER TO FILE
11      ; If this is an inplace sort, fill rest of file with nulls.
12      ;
13 003522 032767 0000000 0000000      BIT      #F#INPL,SFLAGS ;INPLACE SORT?
14 003530 001002      BNE      1$      ;BRANCH AROUND IF INPLACE
15 003532 000167 001110      JMP      CLSOUT      ;JUMP IF NOT INPLACE SORT
16      ; Write nulls to remainder of file.
17 003536 005024  1$:      CLR      (R4)+      ;FILL BUFFER WITH NULLS
18 003540 020405      CMP      R4,R5      ;FINISHED?
19 003542 103775      BLO      1$      ;BR IF NOT
20 003544      3$:      $WRITE  #FD3, OBUF1, OWCNT, NXOBLK
21 003724      $WAIT  #FD3
22 004040 103002      BCC      31$
23 004042 000167 000600      JMP      CLSOUT      ;JUMP IF END OF FILE REACHED
24 004046      31$:
25 004046 066767 0000000 175162      ADD      BLKFC, NXOBLK      ;ADVANCE BLOCK #
26 004054 000633      BR      3$      ;CONTINUE WRITING NULLS
27      ;
28      ; Output organization is Relative.
29      ;
30 004056 016703 0000000  2$:      MOV      OTYPE,R3      ;GET OUTPUT FILE TYPE FLAGS
31 004062 032703 0000000      BIT      #RT#DBL,R3      ;DIBOL FILE?
32 004066 001033      BNE      10$      ;BR IF YES
33 004070 032703 0000000      BIT      #RT$RA,R3      ;RECORDING MODE = ASCII?
34 004074 001210      BNE      4$      ;IF YES THEN FILL REST OF FILE WITH NULLS
35 004076 032703 0000000      BIT      #RT#CBL,R3      ;COBOL BINARY RELATIVE FILE?
36 004102 001605      BEQ      4$      ;BR IF NOT
37      ;
38      ; Fill COBOL relative organization binary file with deleted records.
39      ;
40 004104 016703 0000000  5$:      MOV      RECLEN,R3      ;GET RECORD LENGTH
41 004110 020405      CMP      R4,R5      ;IS BUFFER FULL?
42 004112 103405      BLO      6$      ;BR IF NOT
43 004114 004767 001440      CALL      RLOUT      ;DUMP BUFFER
44 004120 103002      BCC      6$
45 004122 000167 000520      JMP      CLSOUT      ;JUMP IF END OF FILE REACHED
46 004126 010324      6$:      MOV      R3, (R4)+      ;STORE DELETED-RECORD RECORD CONTROL WORD
47 004130 020405      7$:      CMP      R4,R5      ;IS BUFFER FULL?
48 004132 103405      BLO      8$      ;BR IF NOT
49 004134 004767 001420      CALL      RLOUT      ;DUMP BUFFER
50 004140 103002      BCC      8$
51 004142 000167 000500      JMP      CLSOUT      ;JUMP IF END OF FILE HIT
52 004146 005024      8$:      CLR      (R4)+      ;FILL RECORD WITH NULLS
53 004150 005303      DEC      R3
54 004152 003366      BGT      7$
55 004154 000753      BR      5$      ;GO DO NEXT RECORD
56      ;
57      ; This is a DIBOL random access file.

```

```

58          ; If this is an inplace sort, preserve the contents of the end of the
59          ; file beyond the sort output.
60          ;
61 004156 032767 0000000 0000000 10%:  BIT    #F$INPL,SFLAGS ; INPLACE SORT?
62 004164 001002          BNE    14%          ;
63 004166 000167 000450          JMP    13%          ; JUMP IF NOT
64 004172          14%:
65          ; Read in block from file and pad with original data.
66 004172 010403          MOV    R4,R3          ; GET OFFSET WITHIN CURRENT BUFFER
67 004174 166703 0000000  SUB    OBUF1,R3
68 004200 032703 000777          BIT    #777,R3          ; ARE WE ON A BLOCK BOUNDARY NOW?
69 004204 001002          BNE    15%
70 004206 000167 000430          JMP    13%          ; JUMP IF YES (DON'T NEED TO FILL THIS BLOCK)
71 004212          15%:  $WAIT    #FD3
72 004226          $READ    #FD3,OBUF2,DWCNT,NXOBLK
73 004306          $WAIT    #FD3
74 004322 010302          MOV    R3,R2          ; GET POINTER INTO NEW BUFFER CORRESPONDING TO
75 004324 066702 0000000  ADD    OBUF2,R2          ; OFFSET IN OLD BUFFER
76 004330 112224          12%:  MOVB   (R2)+,(R4)+      ; FILL WITH ORIGINAL FILE DATA
77 004332 005203          INC     R3
78 004334 032703 000777          BIT    #777,R3          ; REACHED BLOCK BOUNDARY YET?
79 004340 001373          BNE    12%          ; BR IF NOT
80          ; Now write out the last buffer load.
81 004342 004767 000672  13%:  CALL    OUTBUF
82          ;
83          ; Close output file and terminate sort.
84          ;
85 004346          CLSOUT: $WAIT    #FD3          ; WAIT FOR LAST WRITE TO FINISH
86 004762 000167 0000000  JMP     P2TERM        ; TERMINATE THE SORT

```

```

1
2      ; OUTREC is called to move the record pointed to by R1 to the
3      ; final output file buffer.
4
5 004766 010146      OUTREC: MOV      R1, -(SP)
6 004770 010246      MOV      R2, -(SP)
7 004772 010346      MOV      R3, -(SP)
8 004774 010446      MOV      R4, -(SP)
9 004776 010546      MOV      R5, -(SP)
10 005000 016704 000000G      MOV      ORPOS, R4      ; GET CURRENT POINTER INTO OUTPUT BUFFER
11 005004 016705 000000G      MOV      OBFEND, R5      ; GET POINTER TO END OF BUFFER
12 005010 016702 000000G      MOV      RECLEN, R2      ; GET SORT RECORD LENGTH
13
14      ; Determine record type.
15
16 005014 016700 000000G      MOV      OTYPE, R0      ; GET OUTPUT FILE TYPE FLAGS
17 005020 032700 000000G      BIT      #RT$F11, R0      ; FILES-11??
18 005024 001404      BEQ      3$      ; BRANCH IF NOT
19 005026 032700 000000G      BIT      #RT$OS, R0      ; SEQUENTIAL?
20 005032 001133      BNE      ORRS      ; BRANCH TO FILES-11 SEQUENTIAL
21 005034 000422      BR      ORFL      ; BRANCH TO FILES-11 RELATIVE
22 005036 032700 000000G      3$: BIT      #RT$CBL, R0      ; COBOL FILE?
23 005042 001404      BEQ      1$      ; BR IF NOT
24 005044 032700 000000G      BIT      #RT$RA, R0      ; COBOL ASCII FILE?
25 005050 001066      BNE      ORCA      ; BR IF YES
26 005052 000564      BR      ORCB      ; COBOL BINARY RECORD
27 005054 032700 000000G      1$: BIT      #RT$FTN, R0      ; FILE TYPE = FORTRAN?
28 005060 001404      BEQ      2$      ; BR IF NOT
29 005062 032700 000000G      BIT      #RT$OR+RT$RB, R0      ; RELATIVE OR BINARY?
30 005066 001001      BNE      2$      ; BR IF EITHER
31 005070 000456      BR      ORCA      ; FORTRAN, SEQUENTIAL, ASCII
32 005072 032767 000000G 000000G 2$: BIT      #RT$VLN, ITYPE      ; VARIABLE LENGTH INPUT RECORDS?
33 005100 001016      BNE      ORVL      ; BR IF YES
34
35      ; Fixed length record.
36
37 005102 020405      ORFL:  CMP      R4, R5      ; IS OUTPUT BUFFER FULL?
38 005104 103402      BLO      1$      ; BR IF NOT
39 005106 004767 000426      CALL      OUTBUF      ; FLUSH THE BUFFER
40 005112 112124      1$:  MOVB      (R1)+, (R4)+      ; MOVE RECORD TO OUTPUT BUFFER
41 005114 005302      DEC      R2      ; MOVE ALL OF RECORD
42 005116 003371      BGT      ORFL
43
44 005120 032767 000000G 000000G      ; See if we should add CR-LF.
45 005126 001574      BIT      #RT$CR, OTYPE      ; ADD CR-LF?
46 005130 004767 001026      BEQ      OREND      ; BR IF NOT
47 005134 000571      CALL      ADDCR      ; ADD CR-LF
48
49      ; Variable length record.
50
51 005136 112103      ORVL:  MOVB      (R1)+, R3      ; GET NEXT CHAR FROM RECORD
52 005140 001407      BEQ      2$      ; BR IF END OF RECORD HIT
53 005142 020405      CMP      R4, R5      ; OUTPUT BUFFER FULL?
54 005144 103402      BLO      1$      ; BR IF NOT
55 005146 004767 000366      CALL      OUTBUF      ; FLUSH BUFFER
56 005152 110324      1$:  MOVB      R3, (R4)+      ; MOVE CHAR TO OUTPUT BUFFER
57 005154 005302      DEC      R2      ; CONTINUE MOVING RECORD

```

```
58 005156 003357          BGT      ORVL
59                      ; If output organization = relative, pad with blanks to full record length.
60 005160 032767 000000G 000000G 2$: BIT      #RT$VLN,DTYPE ; ALLOW SHORT OUTPUT RECORDS?
61 005166 001010          BNE      3$      ; BR IF YES
62 005170 005702          TST      R2      ; NEED PADDING?
63 005172 001406          BEQ      3$      ; BR IF NOT
64 005174 112703 000040     MOVB     #' ,R3      ; GET PADDING SPACE
65 005200 004767 000742     4$: CALL     OUTCHR   ; STORE BLANKS INTO END OF RECORD
66 005204 005302          DEC      R2      ; PAD TO FULL RECORD LENGTH
67 005206 003374          BGT      4$
68                      ; Insert carriage-return line-feed.
69 005210 032767 000000G 000000G 3$: BIT      #RT$CR,DTYPE ; NEED TO INSERT CR-LF?
70 005216 001540          BEQ      OREND      ; BR IF NOT
71 005220 004767 000736     CALL     ADDCR      ; INSERT CR-LF
72 005224 000535          BR       OREND
```

1					
2					; COBOL ascii record. (And FORTRAN sequential, ascii record)
3					
4	005226	112703	000012	ORCA:	MOVB #LF,R3 ; INSERT LEADING LINE-FEED
5	005232	004767	000710		CALL OUTCHR
6	005236	112103		1\$:	MOVB (R1)+,R3 ; GET NEXT CHAR FROM RECORD
7	005240	001407			BEQ 3\$; BR IF END OF RECORD REACHED
8	005242	020405			CMP R4,R5 ; OUTPUT BUFFER FULL?
9	005244	103402			BLO 2\$; BR IF NOT
10	005246	004767	000266		CALL OUTBUF ; FLUSH BUFFER
11	005252	110324		2\$:	MOVB R3,(R4)+ ; MOVE CHAR TO OUTPUT BUFFER
12	005254	005302			DEC R2 ; MORE TO MOVE?
13	005256	003367			BGT 1\$; BR IF YES
14					; If file organization is Relative, pad with blanks to full record length.
15	005260	032767	0000006 0000006	3\$:	BIT #RT\$VLN,OTYPE ; ALLOW SHORT OUTPUT RECORDS?
16	005266	001010			BNE 4\$; BR IF YES
17	005270	005702			TST R2 ; DO WE NEED PADDING?
18	005272	001406			BEQ 4\$; BR IF NOT
19	005274	112703	000040		MOVB #' ,R3 ; GET PADDING BLANK
20	005300	004767	000642	5\$:	CALL OUTCHR ; INSERT BLANKS
21	005304	005302			DEC R2 ; FILL RECORD FILLED
22	005306	003374			BGT 5\$
23	005310	112703	000015	4\$:	MOVB #CR,R3 ; NOW INSERT CR
24	005314	004767	000626		CALL OUTCHR
25	005320	000477			BR OREND

1					
2					; FILES-11 sequential access
3					
4	005322	005204		ORRS: INC R4	;ROUND BUFFER POINTER TO NEXT
5	005324	042704	000001	BIC #1,R4	;WORD
6					; Strip trailing nulls or blanks.
7	005330	010103		MOV R1,R3	;GET ADDRESS OF RECORD
8	005332	060203		ADD R2,R3	;POINT PAST LAST WORD OF RECORD
9	005334	032767	0000006 0000006	BIT #RT#RB,DTYPE	;BINARY MODE?
10	005342	001405		BEG 7#	;BRANCH IF NOT
11	005344	105743		2#: TSTB -(R3)	;SCAN OFF TRAILING NULL WORDS
12	005346	001010		BNE 1#	;BR IF REACHED NON-NULL
13	005350	005302		DEC R2	;KEEP TRACK OF RECORD LENGTH
14	005352	003374		BGT 2#	;BR IF MORE TO CHECK
15	005354	000405		BR 1#	
16					; ASCII recording mode
17	005356	122743	000040	7#: CMPB #' ,-(R3)	
18	005362	001002		BNE 1#	
19	005364	005302		DEC R2	
20	005366	003373		BGT 7#	
21					; Record size word.
22	005370	020405		1#: CMP R4,R5	;IS OUTPUT BUFFER FULL?
23	005372	103402		BLO 3#	;BR IF NOT
24	005374	004767	000140	CALL OUTBUF	;FLUSH BUFFER
25	005400	010224		3#: MOV R2,(R4)+	;STORE RECORD LENGTH
26	005402	001407		BEG 6#	;BR IF NONE
27	005404	020405		5#: CMP R4,R5	;OUTPUT BUFFER FULL?
28	005406	103402		BLO 4#	;BR IF NOT
29	005410	004767	000124	CALL OUTBUF	;FLUSH BUFFER
30	005414	112124		4#: MOVB (R1)+,(R4)+	;MOVE RECORD TO OUTPUT BUFFER
31	005416	005302		DEC R2	
32	005420	003371		BGT 5#	
33	005422	000436		6#: BR DREND	
34					
35					; COBOL Binary record.
36					
37	005424	042702	000001	ORCB: BIC #1,R2	;MAKE SURE RECORD LENGTH IS EVEN
38	005430	032767	0000006 0000006	BIT #RT#OR,DTYPE	;ORGANIZATION = RELATIVE?
39	005436	001007		BNE 1#	;BR IF YES
40					; Strip trailing nulls.
41	005440	010103		MOV R1,R3	;GET ADDRESS OF RECORD
42	005442	060203		ADD R2,R3	;POINT PAST LAST WORD OF RECORD
43	005444	005743		2#: TST -(R3)	;SCAN OFF TRAILING NULL WORDS
44	005446	001003		BNE 1#	;BR IF REACHED NON-NULL
45	005450	162702	000002	SUB #2,R2	;KEEP TRACK OF RECORD LENGTH
46	005454	003373		BGT 2#	;BR IF MORE TO CHECK
47					; Build record control word.
48	005456	020405		1#: CMP R4,R5	;IS OUTPUT BUFFER FULL?
49	005460	103402		BLO 3#	;BR IF NOT
50	005462	004767	000052	CALL OUTBUF	;FLUSH BUFFER
51	005466	010214		3#: MOV R2,(R4)	;STORE RECORD LENGTH
52	005470	052724	100000	BIS #100000,(R4)+	;SET RECORD-EXISTS FLAG
53	005474	006202		ASR R2	;GET # WORDS IN RECORD
54	005476	001407		BEG 6#	;BR IF NONE
55	005500	020405		5#: CMP R4,R5	;OUTPUT BUFFER FULL?
56	005502	103402		BLO 4#	;BR IF NOT
57	005504	004767	000030	CALL OUTBUF	;FLUSH BUFFER

```
58 005510 012124      4$:      MOV      (R1)+, (R4)+      ; MOVE RECORD TO OUTPUT BUFFER
59 005512 005302              DEC      R2
60 005514 003371              BGT      5$
61 005516 000400      6$:      BR       (OREND)
62                      ;
63                      ; Finished moving record to output buffer.
64                      ;
65 005520 010467 000000G  OREND:  MOV      R4, ORPOS      ; SAVE OUTPUT BUFFER POINTER
66 005524 012605              MOV      (SP)+, R5
67 005526 012604              MOV      (SP)+, R4
68 005530 012603              MOV      (SP)+, R3
69 005532 012602              MOV      (SP)+, R2
70 005534 012601              MOV      (SP)+, R1
71 005536 000207              RETURN
```

```

1      ; -----
2      ; OUTBUF is called to write the current output buffer to the final output file.
3      ; An abort occurs if the output file overflows.
4      ; Inputs:
5      ;   R4 = Pointer to end of data in buffer.
6      ;   OBUF1 = Pointer to start of buffer.
7      ; Outputs:
8      ;   R4 = Address of start of fresh buffer.
9      ;   R5 = Address of end of buffer.
10     ;
11 005540 004767 000014 OUTBUF: CALL    RLOUT          ;TRY TO WRITE BUFFER TO FILE
12 005544 103004        BCC     1$          ;BR IF NO OVERFLOW
13 005546        XXX     16          ;FILE OVERFLOW
14 005556 000207 1$:      RETURN
15
16     ; -----
17     ; RLOUT is called to write the current buffer full of data up to
18     ; (but not beyond) the end of the current output file.
19     ; The c-flag is set on return if the end of file is reached.
20     ; Inputs:
21     ;   R4 = Address of end of data in current output buffer.
22     ;   OBUF1 = Address of current output buffer.
23     ; Outputs:
24     ;   R4 = Address of start of fresh buffer.
25     ;   R5 = Address of end of buffer.
26     ;
27 005560 RLOUT: $WAIT    #FD3          ;WAIT FOR LAST WRITE TO FINISH
28 005574 103523        BCS     9$          ;BR IF ERROR
29 005576 032704 000001 1$:   BIT     #1,R4      ;IS R4 ON EVEN BYTE BOUNDARY?
30 005582 001401        BEQ     2$          ;BR IF YES
31 005584 105024        CLRB    (R4)+        ;PUT IN NULL
32 005586 166704 000000G 2$:   SUB     OBUF1,R4    ;GET # BYTES IN BUFFER
33 005592 001477        BEQ     3$          ;BR IF BUFFER EMPTY
34 005594 006204        ASR     R4          ;GET # WORDS IN BUFFER
35 005596        $WRITE  #FD3,OBUF1,R4,NXOBLK
36 005602 103424        BCS     9$          ;BR IF HIT END OF FILE
37 005604 062704 000377 ADD     #377,R4      ;BOUND # WORDS UP TO BLOCK BOUNDARY
38 005610 042704 000377 BIC     #377,R4
39 005614 000304        SWAB    R4          ;GET # BLOCKS WRITTEN
40 005616 060467 173124 ADD     R4,NXOBLK    ;ADVANCE FILE BLOCK COUNTER
41     ; Set up new buffer pointers.
42 005612 016704 000000G 3$:   MOV     OBUF2,R4    ;GET ADDRESS OF NEW BUFFER
43 005616 016767 000000G 000000G MOV     OBUF1,OBUF2    ;SWITCH BUFFERS
44 005624 010467 000000G        MOV     R4,OBUF1
45 005630 010405        MOV     R4,R5
46 005632 066705 000000G ADD     OBUFSZ,R5    ;GET ADDRESS OF END OF BUFFERR
47 005636 010567 000000G MOV     R5,OBFEND
48 005642 000241        CLC              ;SAY NOT AT END OF FILE
49 005644 000207 9$:      RETURN

```

1				
2				
3				
4				
5				
6				
7	006146	020405		
8	006150	103402		
9	006152	004767	177362	
10	006156	110324		
11	006160	000207		
12				
13				
14				
15				
16				
17				
18				
19				
20	006162	010346		
21	006164	012703	000015	
22	006170	004767	177752	
23	006174	012703	000012	
24	006200	004767	177742	
25	006204	012603		
26	006206	000207		

```

;-----
; OUTCHR is called to place the character in R3 in the output buffer.
; Inputs:
;   R4 = Pointer into output buffer.
;   R5 = Address of end of output buffer.
;
OUTCHR: CMP     R4,R5           ;IS OUTPUT BUFFER FULL?
        BLO     1$             ;BR IF NOT
        CALL    OUTBUF         ;FLUSH THE BUFFER
1$:      MOV     R3,(R4)+       ;MOVE CHAR TO OUTPUT BUFFER
        RETURN

```

```

;-----
; ADDCR is called to insert Carriage-return & Line-feed into the
; current output buffer.
; Inputs:
;   R4 = Pointer into output buffer.
;   R5 = Address of end of output buffer.
;
ADDCR:  MOV     R3,-(SP)
        MOV     #CR,R3         ;INSERT CR
        CALL    OUTCHR
        MOV     #LF,R3         ;INSERT LF
        CALL    OUTCHR
        MOV     (SP)+,R3
        RETURN

```

```

1          .SBTTL  SUBROUTINES
2
3          ; ROUTINE TO FREE THE DISK BLOCK WHOSE ADDRESS IS IN R4.
4          ; R4 IS NOT ALTERED BY THE SUBROUTINE.
5
6 006210 010146 RELGRN: MOV     R1, -(SP)      ; SAVE SOME REGISTERS.
7 006212 010246      MOV     R2, -(SP)
8 006214 010401      MOV     R4, R1          ; GET DISK GRANULE #.
9 006216 042701 177760 BIC     #<^0177760>, R1 ; GET # OF BIT IN WORD.
10 006222 012702 100000 MOV     #<^0100000>, R2
11 006226 000402      BR      1$
12 006230 000241 2$: CLC
13 006232 006002      ROR     R2             ; POSITION BIT TO PROPER POSITION.
14 006234 005301 1$: DEC     R1
15 006236 002374      BGE     2$
16 006240 010401      MOV     R4, R1
17 006242 006201      ASR     R1             ; GET INDEX TO WORD.
18 006244 006201      ASR     R1             ; (DIVIDE BY 16)
19 006246 006201      ASR     R1
20 006250 006201      ASR     R1
21 006252 006301      ASL     R1             ; CVT TO BYTE INDEX
22 006254 066701 172720 ADD     GRNTBL, R1      ; GET ADDRESS OF WORD.
23 006260 020167 172716 CMP     R1, GRNEND      ; DON'T GO PAST END OF TABLE
24 006264 002052      BGE     RELDVF
25 006266 030211      BIT     R2, (R1)        ; GRANULE SHOULD BE BUSY NOW.
26 006270 001050      BNE     RELDVF
27 006272 050211      BIS     R2, (R1)        ; MARK GRANULE AS FREE.
28 006274 012602      MOV     (SP)+, R2      ; RESTORE REGISTERS.
29 006276 012601      MOV     (SP)+, R1
30 006280 000207      RETURN
31
32          ; ROUTINE TO GET THE ADDRESS OF A FREE DISK GRANULE.
33          ; THE ADDRESS IS RETURNED IN R1.
34
35 006302 010246 GETGRN: MOV     R2, -(SP)      ; SAVE SOME REGISTERS.
36 006304 010346      MOV     R3, -(SP)
37 006306 010446      MOV     R4, -(SP)
38 006310 016701 172664 MOV     GRNTBL, R1      ; GET ADDRESS OF START OF BIT TABLE.
39 006314 005721 1$: TST     (R1)+          ; LOOK FOR WORD WITH FREE GRANULE.
40 006316 001776      BEQ     1$             ; BRANCH IF NONE IN THIS WORD.
41 006320 014102      MOV     -(R1), R2      ; GET FREE GRANULE BITS.
42 006322 005003      CLR     R3
43 006324 006102 3$: ROL     R2             ; SEARCH FOR FREE GRAN BIT.
44 006326 103402      BCS     2$             ; BRANCH WHEN FOUND.
45 006330 005203      INC     R3             ; KEEP TRACK OF ADDRESS.
46 006332 000774      BR      3$
47 006334 010304 2$: MOV     R3, R4          ; SAVE BIT INDEX TO GRANULE.
48 006336 000241 4$: CLC
49 006340 006002      ROR     R2             ; MARK GRANULE AS USED NOW.
50 006342 005303      DEC     R3             ; REPOSITION BIT.
51 006344 002374      BGE     4$
52 006346 010211      MOV     R2, (R1)        ; RESTORE WORD WITH GRANULE BUSY.
53 006350 166701 172624 SUB     GRNTBL, R1      ; GET WORD INDEX.
54 006354 006301      ASL     R1             ; CONVERT WORD TO BIT INDEX.
55 006356 006301      ASL     R1
56 006360 006301      ASL     R1
57 006362 060401      ADD     R4, R1          ; ADD IN BIT INDEX WITHIN WORD.

```

```
58 006364 020167 0000009      CMP      R1,NXTBLK      ;MAKE SURE WE DON'T OVERFLOW.
59 006370 003004      BGT      OVTMP      ;TEMP FILE OVERFLOW
60 006372 012604      MOV      (SP)+,R4
61 006374 012603      MOV      (SP)+,R3
62 006376 012602      MOV      (SP)+,R2
63 006400 000207      RETURN
64 006402      OVTMP:  XXX      J7
65 006412      RELOVF: XXX      J8
66      ;
67      006424'      P2TOP    =      .+2
68      006424      P2SIZ    =      P2TOP-P2BASE
69      000001      .END
```

Symbol table

ABORT = ***** G	FILST = ***** G	MRCSET 002102R	002 DWCNT = ***** G	RT\$VLN= ***** G
ADDR 000162R	002 FINCLS 003376R	002 MXFILS= ***** G	PENMRG 001546R	002 RT11 = 000000
BELL = 000007	FINFLG 001242R	002 MXFIL2= ***** G	PFSPC = 000002	RWDLEN= ***** G
BIGEST= 177777	FINMOV 002472R	002 MXMRG = 000050	P2BASE= 000000R	002 SEVERR= 000010
BLKFC = ***** G	FLBLK = ***** G	MXMRG2= 000120	P2SIZ = 006424 G	SFLAGS= ***** G
BUFSIZ= 001000	FLUSH = ***** G	NFILE2 001204R	002 P2TERM= ***** G	SORTP2 001244R 002
CHNB11= 000400	FT\$T = ***** G	NOERR = 000001	P2TOP = 006424R 002	SRFIN 003330R 002
CHND = 000000	F\$INPL= ***** G	NUMKY2= ***** G	Q 001232R	002 SR2 002270R 002
CISR12= ***** G	GETGRN 006302R	002 NUMMRG 001206R	002 QLEN = 000003	SR3 002330R 002
CLSOUT 004646R	002 GRNEND 001202R	002 NXDBLK 001236R	002 RC 001230R	002 SR4 002506R 002
COMPAR= ***** G	GRNTBL 001200R	002 NXTBLK= ***** G	RECLEN= ***** G	SR4L 003174R 002
CR = 000015	HB\$ARS= 000006	DBFEND= ***** G	RELGRN 006210R	002 SR5 003202R 002
CSPCDF= 000376	HB\$DFS= 000002	DBUFSZ= ***** G	RELOVF 006412R	002 SR6 003212R 002
CTRLZ = 000032	HB\$DIP= 000030	DBUF1 = ***** G	RLOUT 005560R	002 SR7 003322R 002
CUROFL= ***** G	HB\$DUP= 000001	DBUF2 = ***** G	RMON = 000054	STRAT 001454R 002
DBLKFC= 000005	HB\$RIU= 000032	DFLINK 001240R	002 RN 000620R	002 STRFIN 001652R 002
DOMRG 001656R	002 HB\$RSZ= 000004	ORCA 005226R	002 RQ 001234R	002 TMPMOV 002350R 002
DSKCVT= ***** G	HD\$DEV= 000000	ORCB 005424R	002 RSTSJB= 000404	TOPMEM= ***** G
EMTBLK 001210R	002 HD\$SZ= 000004	OREND 005520R	002 RT\$CBL= ***** G	UEOF = ***** G
ENDRUN 003144R	002 IRPOS 000740R	002 ORFL 005102R	002 RT\$CR = ***** G	USERRB= 000053
EOFBUF= ***** G	ITYPE = ***** G	ORPOS = ***** G	RT\$DBL= ***** G	USRLOC= 000046
ERR = 000006	JSW = 000044	ORRS 005322R	002 RT\$FTN= ***** G	VERSNO= 000276
ERRLOC= 000052	KSTRT = ***** G	ORVL 005136R	002 RT\$F11= ***** G	WRDS = 000004
FD3 = ***** G	KTYPE = ***** G	OTYPE = ***** G	RT\$OR = ***** G	XEMT = ***** G
FE 000360R	002 LF = 000012	OUTBUF 005540R	002 RT\$OS = ***** G	XXX16 = 005546R 002
FI 000500R	002 LOC 000120R	002 OUTCHR 006146R	002 RT\$OX = ***** G	XXX17 = 006402R 002
FILFL = ***** G	LOSER 000240R	002 OUTREC 004766R	002 RT\$RA = ***** G	XXX18 = 006412R 002
FILLST 003504R	002 MAJMRG 001576R	002 OVLBIT= 001000	RT\$RB = ***** G	... V1 = 000003
FILNB = ***** G	MRCBA 001060R	002 OVTMP 006402R	002 RT\$TXT= ***** G	... V2 = 000061
FILRN = ***** G	MRELST 000000R	002		

ABS. 000000 000 (RW, 1, GBL, ABS, DVR)
 000000 001 (RW, 1, LCL, REL, CON)
 SORT2 006422 002 (RW, 1, GBL, REL, DVR)

Errors detected: 0

*** Assembler statistics

Work file reads: 0
 Work file writes: 0
 Size of work file: 15783 Words (62 Pages)
 Size of core pool: 18432 Words (72 Pages)
 Operating system: RT-11

Elapsed time: 00:00:26.36
 ,DB4: CBSRT2, DK: CBSRTH, CRF=DK: CBSRTH, CBSRT2/C

...V1	6-52	6-82	6-83	6-83	8-20	8-20	8-21	8-21	8-71	8-71	8-72	8-72	8-73	8-73
	8-55	8-85	12-27	12-27	12-35	12-35								
...V2	6-82	6-82	6-82#	6-82#	6-83	6-83#	8-20	8-20	8-20#	8-20#	8-21	8-21#	8-71	8-71#
	8-72	8-72	8-72#	8-72#	8-73	8-73#	8-85	8-85#	12-27	12-27#	12-35	12-35	12-35#	12-35#
ABORT	2-38	12-13	14-64	14-65										
ADDR	2-46	7-71	13-20#											
BELL	1-40#													
BIGEST	1-36#													
BLKFC	2-46	8-25												
BUFSIZ	1-34#													
CHNBIT	1-53#													
CHNO	1-87#	6-82	6-82	6-83	6-83	8-20	8-20	8-21	8-21	8-71	8-71	8-72	8-72	8-73
	8-73	8-85	8-85	12-27	12-27	12-35	12-35							
CISRT2	2-38	3-5												
CLSOUT	2-15	8-23	8-45	8-51	8-85#									
COMPAR	2-43	6-129												
CR	1-38#	2-70#	10-23	13-21										
CSPCOF	1-55#													
CTRLZ	1-41#	2-72#	7-42											
CURDFL	2-45	4-29*	5-21*	5-42	5-105									
DBLKFC	1-35#													
DOMRG	5-7#													
DSKCVT	2-44	6-81												
EMTBLK	2-77#													
ENDRUN	6-75	6-105#												
EOFBUF	2-39													
ERR	1-90#	6-82*	6-82*	6-82*	6-83	6-83*	6-83*	6-83*	8-20*	8-20*	8-20*	8-21	8-21*	8-21*
	8-21*	8-71	8-71*	8-71*	8-71*	8-72*	8-72*	8-72*	8-73	8-73*	8-73*	8-73*	8-85	8-85*
	8-85*	8-85*	12-27	12-27*	12-27*	12-27*	12-35*	12-35*	12-35*					
ERRLOC	1-49#	6-82	6-83	8-20	8-21	8-71	8-72	8-73	8-85	12-27	12-35			
F\$INPL	2-51	7-40	8-13	8-61										
FD3	2-38	8-20	8-21	8-71	8-72	8-73	8-85	12-27	12-35					
FE	2-63#	5-80*	6-116											
FI	2-64#	5-81*	6-139											
FILFL	2-44	5-109*												
FILLST	7-37	7-41	8-4#											
FILNB	2-41	5-40	5-52	6-80*										
FILRN	2-40	4-11	4-45	5-13	5-16	5-22*	5-35	5-51	6-109*					
FILST	2-40	5-92	5-106	5-108*	6-108*									
FINCLS	7-5	7-18#												
FINFLG	2-84#	3-54*	4-27*	5-7	5-24	5-103	6-18	7-4						
FINMOV	6-19	6-54#												
FLBLK	2-44	6-39*	7-12*											

[illegible]

[illegible]

[illegible]

[illegible]