Horst-
For your information
S.

MIGRATION TO MIRAM

1.0

80/11/05 8:21:34

WEDNESDAY NOVEMBER 5 1980

DOCUMENT:TUUA-F80

MARVIN NIPPER
LA QUINTA MOTOR INNS, INC.
P.O. BOX 32064
SAN ANTONIO, TEXAS 78216

T A B L E   O F   C O N T E N T S

1. SCOPE AND INTENT.

THIS DOCUMENT IS NOT INTENDED TO DISCUSS THE REASONS FOR GOING TO "MIRAM". THAT CAN BE JUSTIFIED BY SIMPLY EXTRAPOLATING SPERRY UNIVAC'S LONG RANGE SYSTEM SOFTWARE PLANS, FROM THE OS/3 RELEASE 7.0 ANNOUNCEMENTS. ALL THE NEW OPERATING SYSTEM COMPONENTS REVOLVE TOTALLY AROUND A "CENTRALIZED DATA MANAGEMENT" INTERFACE WHICH IS TOTALLY MIRAM BASED. ALL THE NEW SYSTEM DATA MANAGEMENT ENHANCEMENTS, SUCH AS MOVING TOWARD AN OPERATING SYSTEM ENVIRONMENT THAT WILL SUPPORT SHARED UPDATE OF ONE FILE BY TWO INDEPENDANT JOBS, IS BASED ON MIRAM. THE ORIGINAL DATA MANAGEMENT ACCESS METHODS ARE SIMPLY "SUPPORTED" IN RELEASE 7.0, AS THEY EXISTED IN RELEASE 6.0. THEREFORE, THOSE USERS WHO INTEND TO PROGRESS WITH OS/3 IN THE FUTURE, MUST BEGIN TO MOVE TOWARD MIRAM. THIS DOCUMENT, THEN, IS INTENDED SOLELY TO POINT OUT SOME CONSIDERATIONS THAT ONE SHOULD TAKE INTO ACCOUNT WHILE ATTEMPTING TO MAKE THAT MOVE.

2. WHAT IS MIRAM.

2.1. GENERAL DESIGN.

"MULTI-INDEXED RANDOM ACCESS METHOD" (MIRAM) IS ACTUALLY A SLIGHT MISNOMER, AS THE MIRAM DATA MANAGEMENT STRUCTURE ENCOMPASSES MUCH MORE THAN JUST "INDEXED" FILE STRUCTURES. MIRAM FILES CAN EXIST IN TWO PRIMARY STATES, INDEXED OR NON-INDEXED (SOMETIMES REFERED TO AS CONSECUTIVE). I REFER TO THESE AS "PRIMARY" STATES DUE TO THE FACT THAT THERE ARE ACTUALLY A NUMBER OF "HYBRID" IMPLEMENTATIONS OF MIRAM, INVOLVING POSSIBLY COMPLEX COMBINATIONS OF BOTH INDEXED AND NON-INDEXED DATA RECORDS WITHIN THE SAME FILE. THESE SPECIAL STATES WILL NOT BE DISCUSSED IN THIS PAPER AS THEY ARE APPLICABLE ONLY TO A SMALL SEGMENT OF USERS AND, IN FACT, CAN NOT EVEN BE CREATED EXCEPT AT THE ASSEMBLER "MACRO" LEVEL.

2.2. NON-INDEXED (CONSECUTIVE) MIRAM.

NON-INDEXED MIRAM FILES MAY MOST SIMPLY BE COMPARED TO CONVENTIONAL "SAM" FILES. HOWEVER, THEY ALSO PROVIDE THE BASIS FOR THE "DIRECT" FUNCTIONS THAT WERE TRADITIONALLY PERFORMED BY "DAM". THERE IS NO DISTINCTION MADE WITHIN FILE DESCRIPTOR INFORMATION IN A DISK PACK'S VTOC, BETWEEN MIRAM "SEQUENTIAL" AND "DIRECT" FILES. THEY ARE FULLY INTERCHANGABLE WITH THE METHOD OF PROCESSING BEING DETERMINED SOLELY BY THE FILE DEFINITION IN A USING PROGRAM.

2.3. INDEXED MIRAM.

THE INDEXED MIRAM FILES ARE, OF COURSE, INTENDED TO PROVIDE THOSE FUNCTIONS FORMERLY PROVIDED BY "ISAM" FILES. HOWEVER, NUMEROUS CHARACTERISTICS DISTINGUISH THEM FROM THEIR ISAM PREDECESSORS.

INDEXED MIRAM FILES ARE "TWO PARTITION" FILES. UNLIKE ISAM, THE FIRST PARTITION CONTAINS THE DATA RECORDS, AND THE SECOND ONE CONTAINS THE INDEX STRUCTURE. THERE CAN BE FROM ONE TO FIVE SEPARATE INDEXES INTO A GIVEN RECORD. EACH OF THE FIVE INDEXES IS DEFINED WITH FOUR "INDEX CHARACTERISTITCS". TWO OF THESE ARE OBVIOUSLY "KEY LENGTH" AND "KEY LOCATION". THE OTHER TWO DEAL WITH HOW A PARTICULAR INDEX MAY BE PROCESSED. IT MAY BE DEFINED AS ALLOWING OR NOT ALLOWING DUPLICATE KEYS. IT MAY ALSO BE DEFINED AS ALLOWING OR NOT ALLOWING A KEY TO BE CHANGED. WHEN A KEY IS CHANGED, MIRAM DATA MANAGEMENT AUTOMATICALLY CHANGES THE INDEX STRUCTURE SO AS TO PLACE THE RECORD IN ITS NEW SEQUENCE WITHIN THAT KEY.

ONE OF THE MOST IMPORTANT DIFFERENCES BETWEEN "ISAM" AND "MIRAM" IS THE INDEX DESIGN. SAM FILES ONLY INDEXED A SMALL PORTION, OF ONLY THE ORIGINALLY LOADED RECORDS. IN AN INDEXED MIRAM FILE, ALL RECORDS HAVE AN INDEX ENTRY WHICH POINTS TO THEM, REGARDLESS OF

WHETHER THEY ARE PART OF THE ORIGINALLY LOADED FILE, OR WHETHER THEY ARE ADDED LATER. THIS ONE VARIATION REQUIRES A USER TO THINK OF MIRAM IN A TOTALLY DIFFERENT FASHION, IMPACTING EVERYTHING FROM RECOVERY, TO PERFORMANCE. THESE TOPICS WILL ALL BE COVERED.

DURING AN INDEXED MIRAM FILE LOAD, A MULTI-LEVEL, TREE-STRUCTURED INDEX IS CREATED. THE "TOP-LEVEL" INDEX IS ALWAYS CREATED SO AS TO ALLOW FOR "HARDWARE SEARCHES". ALL OTHER LEVELS ARE SEARCHED VIA A SOFTWARE ALGORITHM. DURING THE LOAD PROCESS, THE INDEX BLOCKS ARE NEVER FILLED COMPLETELY, BUT INSTEAD, ARE ONLY FILLED TO THE "THREE-QUARTER" POINT. THE ACTUAL DATA RECORDS OF THE FILE ARE CONTIGUOUSLY PLACED INTO THE DATA PARTITION SEQUENTIALLY, IN THE SAME ORDER IN WHICH THEY ARE PRESENTED TO MIRAM.

THE DATA AREA CONTAINS NO "OVERFLOW" AREA. INSTEAD, NEWLY ADDED RECORDS ARE PLACED INTO THE DATA AREA, JUST AS RECORDS WERE DURING THE FILE LOAD. THEY ARE SIMPLY PLACED SEQUENTIALLY INTO THE FILE IMMEDIATELY BEHIND THE ORIGINALLY LOADED RECORDS. IF THE DATA AREA THAT WAS ORIGINALLY ALLOCATED IS EVER FILLED UP, THEN THE FILE WILL AUTOMATICALLY EXTEND AND CONTINUE TO BE FILLED IN THIS WAY.

THE INDEX IS CONTINUALLY MODIFIED DURING "ADD" FUNCTIONS TO REFLECT THE CURRENT RECORD CONTENTS. INDEX INTRIES ARE INSERTED IN THEIR PROPER PLACE WITHIN INDEX BLOCKS, UNTIL A GIVEN BLOCK HAS BEEN COMPLETELY FILLED. WHEN THIS OCCURS, MIRAM DATA MANAGEMENT MAKES A SYSTEM TRANSIENT CALL WHICH "SPLITS" THAT INDEX BLOCK IN HALF, THEREBY MAKING ROOM FOR MORE ENTRIES. THE INDEX MANAGEMENT ROUTINE, LIKE THOSE THAT MANAGE THE ACTUAL DATA RECORDS, PLACE THE NEW INDEX BLOCKS AT THE BACK OF THE INDEX. IT CAN ALSO CALL ON OS/3 EXTEND PROCESSING TO AQUIRE MORE PHYSICAL SPACE IN WHICH TO PLACE AN INDEX BLOCK.

2.4. RECORD CONTROL BYTES (RCB'S).

ASIDE FROM MULTIPLE INDEXES, ANOTHER MAJOR INOVATION IN MIRAM IS THE RECORD CONTROL BYTE (RCB). THIS IS AN OPTIONAL ONE BYTE AREA WHICH, IF PRESENT, WILL IMMEDIATELY PRECEED A DATA RECORD. IT IS NEVER MADE AVAILABLE TO A USER'S PROGRAM, BUT INSTEAD, IS COMPLETELY MAINTAINED BY DATA MANAGEMENT. ITS PRIMARY PURPOSE IS TO PROVIDE FOR THE "DELETE" FACILITY WITHIN MIRAM.

MIRAM DELETE PROCESSING DOES NOT CAUSE A RECORD TO BE PHYSICALLY PURGED. INSTEAD, THE RECORD IS FLAGGED SUCH THAT ALL FUTURE ACCESSES TO THE FILE WILL AUTOMATICALLY "BYPASS" OR "IGNORE" THE RECORD. THE RECORD IS NEVER TRULY REMOVED UNTIL SUCH TIME AS THE FILE UNDERGOES SOME TYPE OF "USER-INITIATED" REORGANIZATION.

IT IS IMPORTANT TO RECOGNIZE THAT THIS FORM OF DELETE IS MUCH DIFFERENT THAN THE HEX 'FF' DELETE CODE IMPLEMENTED WITHIN IMS/90. THIS "RCB" DELETE FACILITY INVOLVES NO USER CODE WHAT-SO-EVER. ONCE A RECORD IS DELETED, NO OTHER PROGRAM ACCESSING THE FILE WILL BE ABLE TO RETRIEVE IT.

RCB USE IS DETERMINED SOLELY AT FILE CREATION TIME. THIS MAY BE

WITH EITHER A "DTF" OPTION OR VIA JCL, BUT ONCE SET, ALL SUBSEQUENT USING PROGRAMS WILL PROCESS THE FILE ACCORDINGLY, REGARDLESS OF THEIR DTF OR JCL SETTINGS.

3. WHERE DOES IRAM FIT IN ?.

INDEXED RANDOM ACCESS METHOD (IRAM) IS SIMPLY THE PREDECESSOR TO MIRAM, BUT IT IS A DIFFERENT ACCESS METHOD AND IT DOES REQUIRE A TOTALLY DIFFERENT DATA MANAGEMENT MODULE AND DTF TO SUPPORT IT. BECAUSE CERTAIN OS/3 COMPONENTS WILL SUPPORT IRAM, BUT WILL NOT YET SUPPORT MIRAM, IT IS CRITICAL TO RECOGNIZE THE DIFFERENCES BETWEEN THE TWO.

IRAM INDEX AND DATA AREAS ARE MAINTAINED ESSENTIALLY JUST LIKE THOSE IN MIRAM, HOWEVER, IRAM FILES CAN ONLY HAVE ONE INDEX, WHERE MIRAM FILES COULD HAVE UP TO FIVE. SECONDLY, IRAM INDEXES WILL NOT SUPPORT CHANGES TO KEYS OR DUPLICATE KEYS. AND LASTLY, IRAM FILES CAN NEVER HAVE RCB'S.

ANY FILE CREATED BY IRAM DATA MANAGEMENT CAN BE PROCESSED BY PROGRAMS WHICH USE MIRAM DATA MANAGEMENT AND MIRAM "DTF"S. HOWEVER, IRAM DATA MANAGEMENT CAN NOT PROCESS ALL FILES CREATED BY MIRAM CODE. ONLY THOSE MIRAM FILES WHICH ARE SAID TO BE "IRAM COMPATIBLE" CAN BE PROCESSED BY IRAM DATA MANAGEMENT. TO BE IRAM COMPATIBLE, THE MIRAM FILE MUST:

- NOT HAVE MORE THAN ONE INDEX,
- NOT HAVE THE INDEX, IF ANY, DEFINED AS ALLOWING EITHER KEY CHANGES OR DUPLICATE KEYS,
- AND, CAN NOT HAVE THE RCB FEATURE INVOKED.

4. SUPPORT OF MIRAM.

4.1. LANGUAGES.

ALL THE TALK OF IRAM COMPATIBILITY IS SO IMPORTANT DUE, PRIMARILY, TO THE MIXED SUPPORT OF MIRAM AVAILABLE IN OS/3 6.X. FIRST, THE SUPPORT WILL BE SUMMARIZED, THEN THE CONSIDERATIONS RESULTING FROM THIS SUPPORT WILL BE DISCUSSED.

- RPG-II: THIS SUPPORTS ONLY IRAM, OR "IRAM COMPATIBLE" MIRAM FILES. IN ADDITION, RPG-II PROGRAMS CAN EITHER BE COMPILED WITH ALL CONVENTIONAL DATA MANAGEMENT (SAM, DAM, ISAM), OR ONLY IRAM, THERE CAN CURRENTLY BE NO MIXTURE OF THE TWO.
- IMS/90: IRAM OR "IRAM COMPATIBLE" ONLY, BUT ONLY FOR INDEXED OR DIRECT PROCESSING.
- TIP/30: FULL MIRAM SUPPORT.
- ANS 68 COBOL: NO SUPPORT.
- ANS 74 COBOL: FULL MIRAM SUPPORT (EXCEPT FOR ONE OVERSIGHT RELATED TO MIRAM "INDEX BUFFERS".)
- FORTRAN IV: FULL MIRAM SUPPORT.
- DATA UTILITY: FULL MIRAM SUPPORT.
- SORT/SORT3: IRAM OR "IRAM COMPATIBLE" ONLY.
- ASSEMBLER: FULL MIRAM SUPPORT.(HOWEVER, NONE OF THE I/O MACROS ARE DOCUMENTED.

THE ONE ASPECT OF THE CURRENT MIRAM IMPLEMENTATION WHICH CAUSES THE MOST CONFUSION IS THE FACT THAT NO IMS/90 SUPPORT YET EXISTS, EXCEPT IN IRAM COMPATABLE MODE. THIS MEANS THAT, FOR ONE, NO MULTIPLY INDEXED FILES CAN BE USED ON-LINE. THIS IS A HARD AND FAST LIMITATION. YOU CAN NOT CIRCUMVENT IT BY EXPECTING TO BE ABLE TO DEFINE TWO, SINGLE INDEXED IRAM FILES IN IMS/90 TO HANDLE ONE MULTIPLY INDEXED MIRAM FILE. IT JUST WILL NOT WORK. (SUPPORT FOR MULTIPLE INDEXES IS SCHEDULED TO COME IN SOME FUTURE RELEASE, BUT NOT IN RELEASE 7.0). AS NOTED ABOVE, HOWEVER, TIP/30 CAN CURRENTLY PROCESS MULTI-INDEXED FILES AS WELL AS MIRAM OUTPUT FILES, IN AN ON-LINE ENVIRONMENT.

4.2. FILE CREATION COMPATIBILITY CONSIDERATIONS.

4.2.1. RCB DELETION.

DUE TO THE VARIOUS MIXTURES OF MIRAM/IRAM SUPPORT ACROSS SYSTEM OMPONENTS, USERS MAY BE FORCED TO RUN THEIR MIRAM FILES IN TOTAL IRAM COMPATIBLTIY MODE. THIS MEANS, ASIDE FROM USING NO MORE THAN ONE INDEX, THAT ALL FILES MUST BE CREATED WITHOUT RCB'S. AS STATED EARLIER, THIS MUST BE DONE AT CREATION TIME. FOR ANS 74 COBOL, RCB CONTROL IS HANDLED SOLELY BY JCL. COBOL WILL AUTOMATICALLY PLACE RCB'S ON THE FILE UNLESS THEY ARE SUPPRESSED ON THE "// DD" STATEMENT. THIS IS DONE BY SPECIFYING "RCB=NO". REMEMBER, THIS HAS TO BE DONE WHEN THE FILE IS LOADED.

4.2.2. INDEX BUFFER SIZES.

ANOTHER COMPATIBILITY PROBLEM IN MIRAM REVOLVES AROUND "INDEX BUFFER SIZE". THIS ONLY OCCURS WHEN DEALING WITH INDEXED FILES. WHEN A USER CREATES AN INDEXED MIRAM/IRAM FILE, HE MUST SPECIFY AT LEAST ONE, 256 BYTE AREA, BUT POSSIBLY MORE, TO BE USED BY MIRAM INDEX PROCESSING. THE CATCH IS THAT ALL FUTURE PROGRAMS WHICH REFERENCE THE FILE MUST SPECIFY THE EXACT SAME BUFFER SIZE. FAILURE TO ADHERE TO THIS RESULTS IN A "DM17" ERROR AT OPEN TIME. OF COURSE, THE LARGER THE BUFFER, THE MORE EFFICIENTLY MIRAM CAN OPERATE. BUT THERE IS A SECOND CATCH. WHEN THEY IMPLEMENTED ANS 74 COBOL, THEY FORGOT TO PROVIDE FOR USER SELECTABLE INDEX SIZES (THIS IS COMING IN RELEASE 7.0). COBOL ALWAYS COMPILES WITH ROOM FOR ONLY ONE 256 BYTE AREA. THUS, THOSE FILES WHICH MUST BE PROCESSED BY ANS 74 COBOL MUST BE CREATED WITH THIS IN MIND. (I.E., ONLY ONE INDEX BUFFER.)

4.2.3. MULTI-VOLUME MOUNT INDICATOR.

ANY NON-INDEXED (CONSECUTIVE) MIRAM FILE WHICH WILL BE USED FOR DIRECT PROCESSING MUST ALWAYS BE CREATED WITH THE "MULTI-VOLUME MOUNT INDICATOR" SET. THE ONLY PLACE THAT A USER REALLY HAS TO WORRY ABOUT THIS IS AT THE ASSEMBLER LEVEL, OR WITH "DATA UTILITIES". IN THE CASE OF DATA UTILITIES, THERE IS A "V" POSITIONAL PARAMETER WHICH SHOULD ALWAYS BE CODED. FAILURE TO CODE IT WILL RESULT IN A "DM" ERROR AT OPEN TIME.

4.3. INDEX PARTITION SIZING.

AS WAS STATED EARLIER, MIRAM INDEXING IS IMPLEMENTED VIA A TWO PARTITION FILE. DUE TO THE FACT THAT ALL RECORDS ARE INDEXED, THE INDEX AREA OCCUPIES SIGNIFICANTLY MORE SPACE OUT OF THE TOTAL FILE, THAN IT DID IN AN ISAM ENVIRONMENT. IN ADDITION, THIS ARRANGEMENT RESULTS IN LARGE VARIATIONS, FROM ONE FILE TO ANOTHER, AS TO WHAT PERCENTAGE OF TOTAL FILE SPACE A MIRAM INDEX SHOULD OCCUPY. WHEN

CREATING A NEW, INDEXED MIRAM FILE, MIRAM WILL AUTOMATICALLY TAKE INTO ACCOUNT INDEX LENGTH AND RECORD SIZE TO COME UP WITH AN OPTIMUM PERCENTAGE TO BE ALLOCATED TO THE INDEX. THE CATCH HERE IS THAT, ALTHOUGH YOU WOULD EXPECT OTHERWISE, THIS OPTIMIZATION FUNCTION IS NOT DONE BY DEFAULT. THE DEFAULT IS FOR MIRAM TO ALWAYS ASSIGN ONLY ONE PERCENT OF THE FILE SPACE TO THE INDEX. YOU MUST INCLUDE THE "// DD" PARAMETER "SIZE=AUTO" IN ORDER TO GET THE OPTIMUM ALLOCATION.

4.4. RECOVERY.

4.4.1. WHAT IS RECOVERY ?

RECOVERY CAPABILITY IS ONE OF THE MOST IMPORTANT OF THE OPTIONAL FEATURES IMPLEMENTED IN MIRAM.. HOWEVER, DUE TO ITS POTENTIAL FOR INCREASING I/O PROCESSING OVERHEAD, IT IS CRITICAL THAT IT ONLY BE USED WHEN NEEDED.

THE RECOVERY OPTION IS GEARED TOWARD ONLY ONE THING, TO ENSURE FILE INTEGRITY AT TIMES WHEN RECORDS HAVE BEEN ADDED TO THE FILE, AND THE FILE DOES NOT GET PROPERLY CLOSED. THIS COULD HAPPEN DUE TO SUCH THINGS AS POWER FAILURES OR HPR STOPS.

RECOVERY CAN BE APPLIED TO BOTH SEQUENTIAL/RANDOM FILES (CONSECUTIVE) AND INDEXED FILES. IN THE CASE OF CONSECUTIVE FILES, RECOVERY WILL SIMPLY ENSURE THAT ONCE THE END-OF-FILE LOCATION BEGINS TO CHANGE, AFTER A FILE IS OPENED, THAT THE NEW END-OF-FILE LOCATION WILL NOT BE LOST, SHOULD A SYSTEM FAILURE OCCUR. FAILURE TO HAVE RECOVERY IN EFFECT WOULD ONLY CAUSE YOU TO LOSE THOSE RECORDS ADDED SINCE THE "OPEN", BUT WOULD NOT CAUSE THE FILE TO BE RENDERED UNUSABLE. THIS SAME PROTECTION IS PROVIDED FOR INDEXED MIRAM FILES, AS WELL AS PROTECTION AGAINST COMPOMISED INDEXES (MORE ON THIS LATER).

EACH TIME THE "END-OF-FILE" SETTING FOR A FILE CHANGES, MIRAM NOTES THE NEW END-OF-FILE SETTING INTO THE FIRST SECTOR OF THE FILE (IT IS RESERVED SPECIFICALLY FOR THIS AT FILE CREATION TIME). IF SYSTEM FAILURE OCCURS, THE DESCREPENCY BETWEEN THIS "RECOVERY SECTOR" AND THE VTOC IS RECOGNIZED THE NEXT TIME THE FILE IS OPENED. THE END-OF-FILE NOTED IN THE SECTOR IS THEN USED TO OVERRIDE THE ONE SHOWN IN THE VTOC.

AS YOU MIGHT IMAGINE, THIS IMPLIES ONE ADDITIONAL WRITE, FOR EACH USER RECORD WRITTEN. BECAUSE THE RECOVERY FUNCTION IS HANDLED BY A "TRANSIENT CALL", THERE IS ACTUALLY MORE OVERHEAD INVOLVED THAN JUST THE I/O TO THE RECOVERY SECTOR. IT IS FOR THIS REASON THAT RECOVERY SHOULD NOT BE APPLIED LOOSELY TO ALL USER FILES. THOSE FILES USED IN SUCH A MANNER THAT SYSTEM CRASHES CAN BE CIRCUMVENTED BY COMPLETE RE-RUNS, FROM AN ORIGINAL VERSION OF THE FILE, SHOULD NOT BE RECOVERY PROTECTED. MOST RECOVERY PROTECTION WILL BE APPLIED TO ON-LINE FILES. CURRENTLY IMS/90 SUPPORTS NO

OUTPUT IRAM FILES AND RELATIVE IRAM FILES CAN NOT BE EXTENDED (AS COULD DAM FILES), HENCE, INDEXED IRAM ON-LINE FILES ARE THE ONLY ONES THAT REALLY NEED THE RECOVERY OPTION SET. NOTE AGAIN, HOWEVER, THAT TIP/30 DOES SUPPORT IRAM/MIRAM OUTPUT FILES, AND THAT RECOVERY CAN BE USED ON THEM TO ENSURE THAT THE END-OF-FILE POINTER IS NEVER LOST, AS IS CURRENTLY THE CASE WHEN USING "SAM" OUTPUT FILES.

EARLIER IT WAS STATED THAT MIRAM RECOVERY PROTECTS AGAINST A COMPROMISED INDEX AREA. ITS INTENT IS REALLY NOT TO PREVENT THEM FROM BEING COMPROMISED, INSTEAD, IT ENSURES THAT ONCE COMPROMISED, THE USER IS MADE AWARE OF THAT FACT SO THAT FURTHER MEASURES CAN BE TAKEN TO REPAIR THE FILE. WHAT MIRAM DOES IS TURN A FLAG IN THE RECOVERY SECTOR, ON JUST BEFORE INDEX MODIFICATION BEGINS, AND THEN OFF AGAIN WHEN INDEX MODIFICATION IS COMPLETE (MORE I/O OVERHEAD TO BE AWARE OF). THIS WOULD TAKE PLACE EACH TIME A NEW RECORD IS ADDED TO THE FILE. THUS WHEN AN INDEXED FILE IS OPENED, IF THE FLAG IS ON, MIRAM KNOWS THAT THE SYSTEM MUST HAVE CRASHED WHILE IN THE PROCESS OF MODIFYING THE INDEX. IN SUCH A CASE, A "DM66" ERROR IS ISSUED BY THE OPEN FUNCTION, AND THE FILE WILL NOT BE OPENED FOR THE REQUESTING PROGRAM.

ANYTIME A "DM66" OCCURS, A USER MUST IMMEDIATELY READ THE FILE "CONSECUTIVELY", IGNORING THE INDEXES. THIS WILL RETRIEVE ALL THE DATA RECORDS IN THE FILE, INCLUDING THOSE ADDED JUST PRIOR TO THE CRASH. THOSE MUST THEN BE SORTED ON THE KEY FIELD AND THEN RELOADED, SO THAT THE INDEX STRUCTURE WILL BE REBUILT. THE EASIEST WAY TO DO THIS IS TO FEED THE FILE DIRECTLY INTO THE SORT, AS IT ALWAYS READS IRAM FILES CONSECUTIVELY, AND DOES NOT USE THE INDEX.

IT IS ALSO IMPORTANT TO CONSIDER HERE, WHAT WILL HAPPEN TO A SYSTEM THAT CRASHES, WITH IRAM FILES WHICH HAVE HAD RECORDS ADDED TO THEM, BUT DO NOT HAVE RECOVERY SET. FIRSTLY, THE INDEX STRUCTURE WOULD ALWAYS BE COMPROMISED. THIS WOULD BE DUE TO THE FACT THAT THE NEW INDEX ENTRIES ADDED SINCE "OPEN" TIME, WOULD BE POINTING TO RECORDS WHICH NOW RESIDE PAST THE END OF FILE LOCATION NOTED IN THE VTOC. ATTEMPTING TO PROCESS THE FILE VIA ITS INDEXES WOULD RESULT IN THE "INVALID ID, EXCEEDS FILE LIMITS" DATA MANAGEMENT ERROR. THUS, THE USER IS FORCED TO IMMEDIATELY READ THE FILE CONSECUTIVELY, AND RELOAD IT, LOSING ALL RECORDS ADDED DURING THIS LAST SESSION. BUT, IF HE FAILS TO DO THIS, AND ACCIDENTLY BRINGS THE SYSTEM BACK UP, IRAM WOULD BEGIN ADDING NEW RECORDS TO THE FILE AND WOULD EVENTUALLY "PUSH" THE END OF FILE POINTER PAST THE POINT WHERE THE "LOST" RECORDS HAD BEEN ADDED. THE "FILE LIMITS" ERROR WOULD GO AWAY, BUT NOW ALL THOSE INDEX ENTRIES ADDED PRIOR TO THE CRASH WOULD BE POINTING TO RECORDS, WHICH WERE ADDED AFTER THE SYSTEM WAS BROUGHT BACK UP. THE INDEX AND DATA RECORD KEYS WOULD OBVIOUSLY NOT MATCH, RESULTING IN MORE FUN AND EXCITEMENT DURING SOME LATER BATCH RUN!

4.4.2. RECOVERY JCL.

IN ORDER TO HAVE A FILE CREATED WITH RECOVERY, "RECV=YES" MUST BE SPECIFIED ON THE "// DD" STATEMENT WHEN THE FILE IS CREATED. IT CAN NOT BE ADDED LATER WITHOUT RELOADING THE FILE.

DUE TO THE FACT THAT MIRAM WILL NOT OPEN A FILE WHOSE INDEX IS COMPROMISED, A VARIATION OF THE "RECV" KEYWORD EXISTS WHICH WILL OVERRIDE THE "DM66" ERROR WHICH IS NORMALLY ISSUED WHEN AN OPEN IS ATTEMPTED TO THE BAD FILE. THIS SHOULD ONLY BE USED IN THOSE JOB STREAMS INTENDED FOR SYSTEM CRASH RECOVERY, FOR EXAMPLE IN A DEVICE ASSIGNMENT SET FOR A SORT INPUT FILE. IN THESE CASES, YOU MUST CODE "RECV=FCE" ON THE "// DD" STATEMENT.

4.5. VARIABLE SECTOR SIZE (VSEC).

IN MIRAM'S FORERUNNER, IRAM, ONE OF ITS MOST ATTRACTIVE CHARACTERISTICS, FROM A PROGRAMMING STANDPOINT, WAS THE INTRODUCTION OF THE SECTORED DATA MANAGEMENT APPROACH. WHAT THIS MEANT WAS THAT ON SECTORIZED DISKS (8416'S AND 18'S) THE SECTOR, OF 256 BYTES, WAS THE ACTUAL PHYSICAL BLOCK SIZE. THUS, ALL IRAM FILES HAD THE EXACT SAME BLOCK SIZE. RECORDS WERE STORED CONTIGUOUSLY WITHIN THE FILE, WITH NO REGARD FOR WHERE THESE PHYSICAL BLOCKS STARTED OR ENDED. RECORDS SPANNED SECTORS, TRACKS, AND CYLINDERS. THE VALUE THAT YOU CODED IN USER PROGRAMS AS BLOCK SIZE WAS -REALLY- USED TO SET THE PROGRAM'S I/O BUFFER SIZE. IT WAS ALWAYS ROUNDED UP TO SOME MULTIPLE OF 256 BYTES. USER'S COULD SPECIFY LARGE OR SMALL BLOCK (BUFFER) SIZES FOR THE SAME FILE DEPENDING ON THE APPLICATION INVOLVED. IRAM WOULD SIMPLY READ AS MANY SECTORS, IN ONE INPUT OPERATION, AS WOULD FIT IN THAT PARTICULAR BUFFER. WHAT THIS MEANT WAS THAT THE OLD CONCEPT OF BLOCK SIZES BECAME MEANINGLESS. THE ONLY LIMITATION DEALT WITH RECORD SIZE. A USER'S BUFFER HAD TO BE LARGE ENOUGH TO HOLD THE NUMBER OF SECTORS WHICH ONE RECORD COULD POSSIBLY SPAN. FOR EXAMPLE, A RECORD OF 258 BYTES COULD, AT SOME POINT IN THE FILE, SPAN THREE SECTORS. THUS, THE MINIMUM BUFFER SIZE FOR FILES OF THIS RECORD SIZE WOULD ALWAYS BE 768 BYTES (3 X 256). ALL THE COMPILERS WERE CODED TO ACCOUNT FOR THESE MINIMUMS, REGARDLESS OF WHAT A USER MIGHT CODE. FOR EXAMPLE, AN RPG-II USER WHO CODED A BLOCK SIZE OF 258 FOR THE ABOVE RECORD WOULD HAVE REALLY GOTTEN A 768 BYTE BUFFER.

THIS WAS A VERY KEEN CONCEPT IN THEORY. IT HAD WORKED GREAT FOR IBM'S SYSTEM/3, WHY NOT UNDER OS/3 ? IT WAS NICE FOR THE SECTORIZED DISK USERS (8416 AND 18 USERS). THE PROBLEMS AROSE WITH THE NON-SECTORIZED SELECTOR DISKS, ESPECIALLY THE 8430/33 DISKS. TO CONFORM WITH DESIGN CONSTRAINTS, IRAM WROTE PHYSICAL BLOCK SIZES OF 256 BYTES TO ALL SELECTOR DISKS, JUST AS IT DID FOR THE SECTORIZED DISKS. THIS WORKED JUST FINE. HOWEVER, DUE TO THE FACT THAT SMALL BLOCK SIZES, LIKE 256 BYTES, RESULT IN LARGER NUMBERS OF "INTER-RECORD GAPS", IRAM FORMATTED DISK SPACE WASTED APPROXIMATELY 40 TO 45% OF THE TOTAL CAPACITY AVAILABLE ON THE 8430/33 DISK DRIVE. THIS MADE SELECTOR DISK OWNERS VERY UNHAPPY, TO SAY THE LEAST.

THE "SOLUTION" TO THIS, INTRODUCED WITH MIRAM AND RETROFITTED TO IRAM IN RELEASE 6.0, WAS "VARIABLE SECTOR" SUPPORT. THE IDEA, WHICH IS NOT APPLICABLE TO 8416 AND 18 DEVICES SEEMS SIMPLE IN THEORY, BUT AS YOU WILL SEE, IT TOO HAS SOME SHORTCOMINGS. THE IDEA IS TO ALLOW A USER TO "SET" WHATEVER SECTOR SIZE HE DESIRES FOR A GIVEN FILE. THE IDEA BEING THAT, BY USING LARGER SECTOR SIZES, MUCH BETTER TRACK UTILIZATION PERCENTAGES CAN BE OBTAINED, JUST AS THEY ARE WITH THE OLD CONVENTIONAL ACCESS METHODS.

THE USER SPECIFIES THE VARIABLE SECTOR SIZE BY MEANS OF ANOTHER "// DD" PARAMETER, "VSEC=NNNN". UNLIKE MANY OTHER MIRAM "DD" PARAMETERS, THIS ONE MUST EXIST IN ALL JOB STREAMS WHICH USE THE FILE. FAILURE TO CODE IT WILL RESULT IN A "DM17" OPEN ERROR. THIS IS WHERE THE PROBLEM BEGINS. (THEORETICALLY YOU CAN USE THE ",,ACCEPT" PARAMETER ON LFD STATEMENTS OF ALL SUBSEQUENT USERS OF THE FILE, INSTEAD OF CODING A "VSEC" ON THEIR "DD". HOWEVER, THERE ARE THOSE OS/3 SOFTWARE PERSONNEL WHO CONTEND THAT AN "ACCEPT" MAY DO SOME UNWANTED PROCESSING, AS WELL AS HANDLING THE VARIABLE SECTOR PROCESSING INDEPENDENT OF A "DD" PARAMETER. ALSO, THERE IS NO WAY TO BOTH "ACCEPT" AND "EXTEND" A FILE, AS THESE PARAMETERS ARE MUTUALLY EXCLUSIVE.) ALL OF THE VSEC SIZING TAKES PLACE EXTERNALLY TO MIRAM, AND IS COMMUNICATED TO OS/3 ONLY WHEN THE FILE IS ACCESSED. COMPILERS AND SYSTEM UTILITIES ARE STILL UNDER THE ASSUMPTION THAT THE REAL SECTOR SIZE IS 256 BYTES, AND THEY ENSURE THEIR MINIMUM BUFFER SIZES ACCORDINGLY.

SO, THE INITIAL PROBLEM WITH VSEC FILES IS THAT COMPILED CODE IS NO LONGER "BLOCK SIZE INDEPENDENT" AS IN THE ORIGINAL IRAM IMPLEMENTATION. YOU CAN NO LONGER VARY MIRAM BUFFER SIZES FOR THE SAME FILE, WITHOUT SOME CONCERN FOR PHYSICAL BLOCK SIZES. IT IS VERY EASY, INDEED, TO END UP WITH A 512 BYTE BUFFER IN AN RPG PROGRAM WHICH CAN NOT EVEN BEGIN TO HOLD THE 2048 BYTE SECTOR SIZES ASSIGNED BY A USER WITH A "VSEC" PARAMETER. THE SECOND PROBLEM RESULTS FROM THE FACT THAT ESSENTIALLY THE SAME RULES APPLY TO THE SECTOR, WHETHER IT IS 256 BYTES OR SOME LARGER SIZE, SUCH AS 2048 BYTES, IN LENGTH. REMEMBER THE 258 BYTE RECORD IN THE 256 BYTE SECTOR THAT YIELDED A 768 BYTE MINIMUM BUFFER ? IF I HAVE A 2050 BYTE (OR LARGER) RECORD AND A 2048 BYTE SECTOR, I NEED 3 X 2048, OR 6144 BYTES AS A MINIMUM BUFFER. YOU WOULD HAVE TO MANUALLY INSURE THAT ALL PROGRAMS AND UTILITIES HAD THEIR "BLOCK SIZES" OR "RECORD CONTAINS" SPECIFICATIONS CODED LARGE ENOUGH TO HOLD ONE OF THESE 6144 BYTE AREAS, NOT JUST 2048.

VSEC SIZING MUST BE DONE WITH GREAT CARE, AND ALL PROGRAMS USING THAT FILE MUST HAVE THEIR "BUFFER SIZES" (BLOCKING FACTORS) ASSIGNED TO BE COMPATIBLE WITH THE VSEC, ONCE CHOSEN. IN CALCULATING VSEC, YOU MUST FIRST DETERMINE YOUR "SLOT SIZE". THIS IS EITHER YOUR RECORD SIZE, OR, IF YOU ARE USING RCB'S, YOUR RECORD SIZE PLUS ONE. THE TRICK THEN, IS TO CHOSE A VSEC SIZE THAT IS AN EXACT MULTIPLE OF "SLOT SIZE". FOR EXAMPLE, IF WE HAVE 100 BYTE RECORDS, WITH RCB'S, WE WOULD HAVE A SLOT SIZE OF 101. LET'S SAY THAT WE DECIDE IN THIS CASE TO USE A VSEC SIZE OF 1010, AS THIS WOULD EXACTLY HOLD 10 SLOTS. (NOTE THAT VSEC NEED NOT BE A MULTIPLE OF 256!) THIS WOULD MEAN THAT:

- ALL USER PROGRAMS MUST HAVE BLOCKING FACTORS TO GUARANTEE AT LEAST A 1010 BYTE BUFFER.

- ALL DEVICE ASSIGNMENT SETS FOR THAT FILE MUST INCLUDE A "// DD VSEC=1010" PARAMETER SETTING.

NOW LETS DECIDE THAT YOU SUDDENLY DECIDE TO PERMANENTLY REMOVE THE RCB WITH A DATA UTILITY, BECAUSE YOU WANT TO RUN THE FILE UNDER IMS/90. RCB'S WERE SUPPOSED TO BE "TRANSPARENT" TO USER CODE, THERE PRESENCE WAS SUPPOSED TO BE ELECTABLE AT FILE CREATION, WITH NO OTHER CONSIDERATIONS. NOT SO WITH VSEC! BECAUSE, TAKING OFF THE RCB CHANGES YOUR SLOT SIZE, AND NOW, THE ORIGINAL VSEC RESULTS IN A SITUATION WHERE THE RECORDS WILL SPAN SECTORS. THIS, PER ORIGINAL SECTOR RULES, MEANS THAT YOU MUST HAVE BUFFER SPACE FOR TWO SECTORS OR 2 X 1010, OR IN OTHER WORDS, 2020 BYTES. BUT MOST OF YOUR PROGRAMS WERE ORIGINALLY COMPILED TO ENSURE ONLY ROOM FOR A 1010 BYTE BUFFER. THE RESULT IS THAT YOU MUST EITHER RECOMPILE THEM ALL WITH DIFFERENT BLOCK SIZE SELECTIONS, OR SELECT A NEW VSEC SIZE AND CHANGE ALL YOUR JCL TO ADHERE TO IT.

A FINAL ASPECT OF VSEC WHICH IS INTENDED TO HELP RECTIFY THIS PROBLEM IS A VARIATION OF THE "VSEC" STATEMENT, IN WHICH ALL JCL FOR A PARTICULAR FILE IS CODED WITH "VSEC=YES" INSTEAD OF "VSEC=NNNN". IN THIS CASE, MIRAM USES THE SLOT SIZE AND THE BUFFER SIZE OF THE PROGRAM THAT IS USING THE FILE, AND CALCULATES THE LARGEST VSEC WHICH WILL FIT INTO THAT BUFFER. THIS KEEPS FUTURE JCL CHANGES TO A MINIMUM, BUT MEANS THAT:

- YOUR LOAD PROGRAM (POSSIBLY A DATA UTILITY) MUST HAVE A CAREFULLY CONSIDERED BLOCKSIZE, SO AS TO YIELD THE VSEC SIZE THAT YOU DETERMINE TO BE THE MOST EFFICIENT, FOR A PARTICULAR SLOT SIZE,

- ALL PROGRAMS WHICH ACCESS THE FILE MUST HAVE AT LEAST "VSEC=YES" (BUT COULD BE CODED WITH THE ABSOLUTE VSEC).

- ALL USER PROGRAMS MUST THEN BE CODED WITH BLOCK SIZES LARGE ENOUGH TO HOLD THE VSEC CALCULATED BY THE LOAD PROGRAM,

- THOSE BLOCK SIZES MUST BE CODED SO THAT THEY DO NOT RESULT IN A LARGER BUFFER SIZE THAN THAT OF THE LOAD PROGRAM, OTHERWISE, THE CALCULATED VSEC SIZE OF THAT USING PROGRAM WOULD BE DIFFERENT THAN THE ONE CALCULATED FOR THE LOAD PROGRAM, RESULTING IN A "DM17" ERROR. (THERE IS SOME QUESTION AS TO WHETHER THIS LAST POINT IS VALID !!!)

A USER SHOP WILL HAVE TO DETERMINE WHETHER "VSEC=NNNN" OR "VSEC=YES" IS THE BETTER METHOD OF IMPLEMENTING VARIABLE SECTOR SUPPORT ON THEIR SELECTOR DISKS. UNFORTUNATELY, NEITHER COMES CLOSE TO PROVIDING THE TRUE BLOCKSIZE INDEPENDENT FLEXIBILITIES ORIGINALLY AFFORDED TO A USER OF 256 BYTE SECTORS.

5. MISCELLANOUS CONSIDERATIONS.

5.1. ANS 74 COBOL EXTEND.

IN ALL PRIOR OS3 RELEASES, THE DESIGN WAS SUCH THAT JCL SPECIFIED FILE PARAMETERS TOOK PRECEDENCE OF THOSE CODED IN THE PROGRAM. FOR EXAMPLE, IF A USER CODED ",,EXTEND" ON AN LFD FOR A SEQUENTIAL OUTPUT FILE, THEN IT EXTENDED THE FILE. THIS IS NOT TRUE OF SEQUENTIAL MIRAM OUTPUT. THERE IS A COBOL OPEN OPTION CALLED "OPEN EXTEND". ONE WOULD THINK THAT THIS WOULD ALLOW A USER TO REQUEST AN EXTEND, IN CASE IT WAS NOT REQUESTED IN JCL. ONE WOULD ALSO THINK THAT IF YOU CODED "OPEN OUTPUT" AND USED A ",,EXTEND" ON THE LFD THAT YOU WOULD ALSO GET EXTENDING. THE LATTER IS NOT SO. FOR MIRAM OUTPUT FILES, THE COBOL OPEN VERB HAS PRESEDENCE. THIS WOULD BE UNDERSTANDABLE EXCEPT THAT IN THE SAME 74 COBOL, WHEN USING CONVENTIONAL "SAM" FILES, THE JCL OPTION STILL TAKES PRECEDENCE. THIS POINT OF CONFLICT HAS BEEN TAKEN UP WITH SPERRY UNIVAC VIA A "REQUEST FOR CHANGE" SUBMITTED AT THE FALL 80 AUUA CONFERENCE.

I N D E X

- E N D O F D O C U M E N T -