

IMS/VIS

DATA BASE

LOGIC & FLOW

Course Materials

Useful:

PLM

LY20-8041

Vol III of III

IMS/VS DATA BASE LOGIC AND FLOW

DATES OF COURSE:

8 Half-Days, August 22 Thru August 31,
1977

INSTRUCTOR:

Bill Lockhart J15/F14 Santa Teresa Lab
Phone - x4240

COURSE ABSTRACT:

This lecture/workshop course is intended to be an in-depth view of the internal workings of an IMS/VS Batch system. In the class you will be provided with several IMS/VS dumps where we will track the basic DL/I calls through the Control Blocks and Modules of IMS/VS.

COURSE OBJECTIVES:

Upon completion of this course, the student should be able to:

- * Load a given data base using any of the IMS/VS data organizations and access methods.
- * Utilize the DL/I Test Program (DFSDDLTO) and the DL/I calls necessary to minipulate logical and physical data bases.
- * Analyze the various types of IMS/VS storage dumps to determine both proper and improper operation of a batch IMS/VS system.
- * List the steps required by DL/I modules to process the basic DL/I calls.
- * Match the DL/I modules with their functions.
- * Locate a given root segment in the data base buffer pool and all its dependents for each of the DL/I data organizations.
- * Indicate the source and purpose of any DL/I control block.
- * Map out a batch IMS/VS storage dump showing the location and contents of the major DL/I control blocks.

COURSE TOPICS:

DL/I General Call Flow
DL/I Control Blocks
Retrieve Call Flow (GET type Calls)
Buffer Management
Load/Insert Call Flow
Delete/Replace Call Flow
Data Base Traces

CLASS EXERCISES: Several Storage dumps taken on 'STLMVS1'
(MVS Batch) will be used for analysis
during class.

MATERIALS: Student Guide (includes copies of all foils
used), Core Dumps, and Library copies of
IMS/VS Pubs.

Updated: 08/11/77

Prerequisite Quiz For Course: IMS/VS D.B. LOGIC & FLOW

Name: _____ Dept/Office: _____ Phone: _____

*
* INSTRUCTIONS: *
* 1. READ EACH QUESTION OVER AND PICK THE BEST ANSWER. EVEN IF YOU *
* FEEL THE 'BEST' ANSWER IS NOT THERE, PICK THE 'NEXT BEST' AMONG *
* THE ANSWERS LISTED. *
* 2. WHEN YOU HAVE FINISHED THIS QUIZ MAIL IT TO: *
* *

* NOTES: *
* THIS QUIZ IS PRIMARILY A TOOL FOR YOU TO ASSURE YOURSELF THAT *
* YOU ARE PREPARED FOR THE COURSE. THIS PREPARATION CAN COME *
* FROM ANY COURSES AND/OR EXPERIENCE YOU HAVE HAD ON THE SUBJECT. *
* ANY QUESTION YOU HAVE DIFFICULTY WITH SHOULD POINT OUT TO YOU *
* AN AREA OF THE SUBJECT THAT YOU SHOULD REVIEW BEFORE COMING TO *
* CLASS. *
* IF YOU HAVE DIFFICULTY WITH THE MAJORITY OF QUESTIONS ON THE *
* QUIZ YOU SHOULD CONSIDER ENROLLING IN PREREQUISITE CLASS(ES) *
* AND TAKING THIS COURSE AT A LATER DATE. *

Circle the correct answer(s) or fill in the blank area:

1. In IMS/VS terms, a Data Base is defined as a nonredundant collection of interrelated data items processable by one or more application program(s).

True/False True

2. When loading a HISAM (ISAM/OSAM) Data Base, each physical data base record starts within an:

- A. ISAM logical record
- B. OSAM logical record
- C. QSAM logical record
- D. BSAM logical record

Use FIGURE 1 (attached) to answer questions 3-7.

3. The circled number 1 indicates a physical:

- A. PARENT pointer
- B. CHILD pointer
- C. CHILD pointer, first & last
- D. TWIN pointer
- E. TWIN pointer, forward & backward

4. The circled number 2 indicates a physical:
- A. PARENT pointer
 - B. CHILD pointer
 - C. CHILD pointer, first & last
 - D. TWIN pointer
 - E. TWIN pointer, forward & backward
5. The circled number 3 indicates a physical:
- A. PARENT pointer
 - B. CHILD pointer
 - C. CHILD pointer, first & last
 - D. TWIN pointer
 - E. TWIN pointer, forward & backward
6. The circled number 4 indicates a physical:
- A. PARENT pointer
 - B. CHILD pointer
 - C. CHILD pointer, first & last
 - D. TWIN pointer
 - E. TWIN pointer, forward & backward
7. The circled number 5 indicates a physical:
- A. PARENT pointer
 - B. CHILD pointer
 - C. CHILD pointer, first & last
 - D. TWIN pointer
 - E. TWIN pointer, forward & backward
8. There must be at least B(=1) DATASET statement(s) for each physical DBD generation.
- A. ZERO
 - B. ONE
 - C. TWO
 - D. THREE
 - E. FOUR
9. The call interface between DL/I and an application program is dependent on a particular storage organization and access method.
- True/False False

10. GET calls may or may not have SSAs.

True/False True

11. DELETE and REPLACE calls may have qualified SSA's

True/False False

12. Any logical data structure may be composed of one or more physical data base records.

True/False True

13. Symbolic pointers may NOT be used when the data base "pointed-to" exists in HIDAM or HDAM.

True/False False

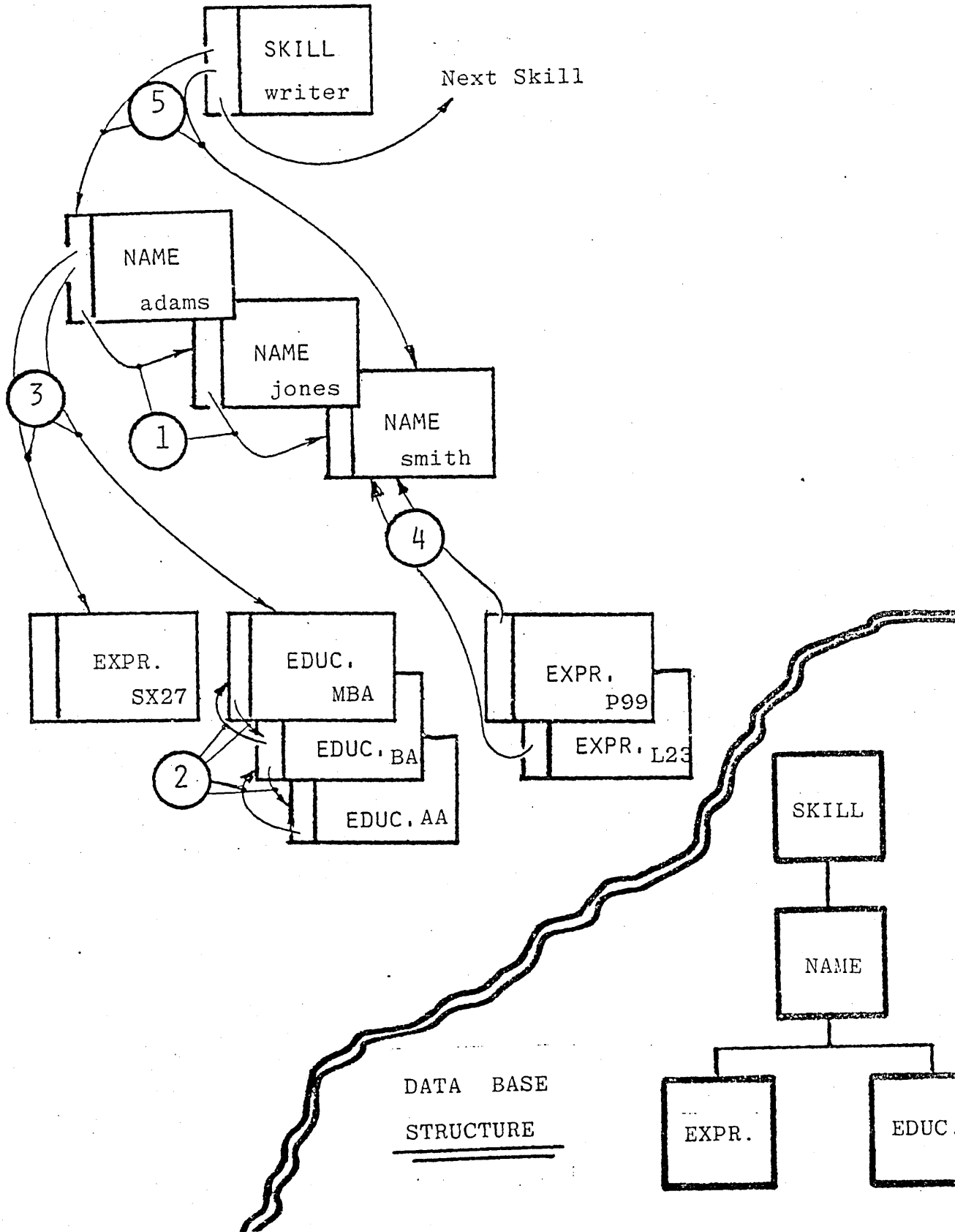
14. When direct address pointers are used to establish a logical relationship, the pointers exist as part of the user data.

True/False False

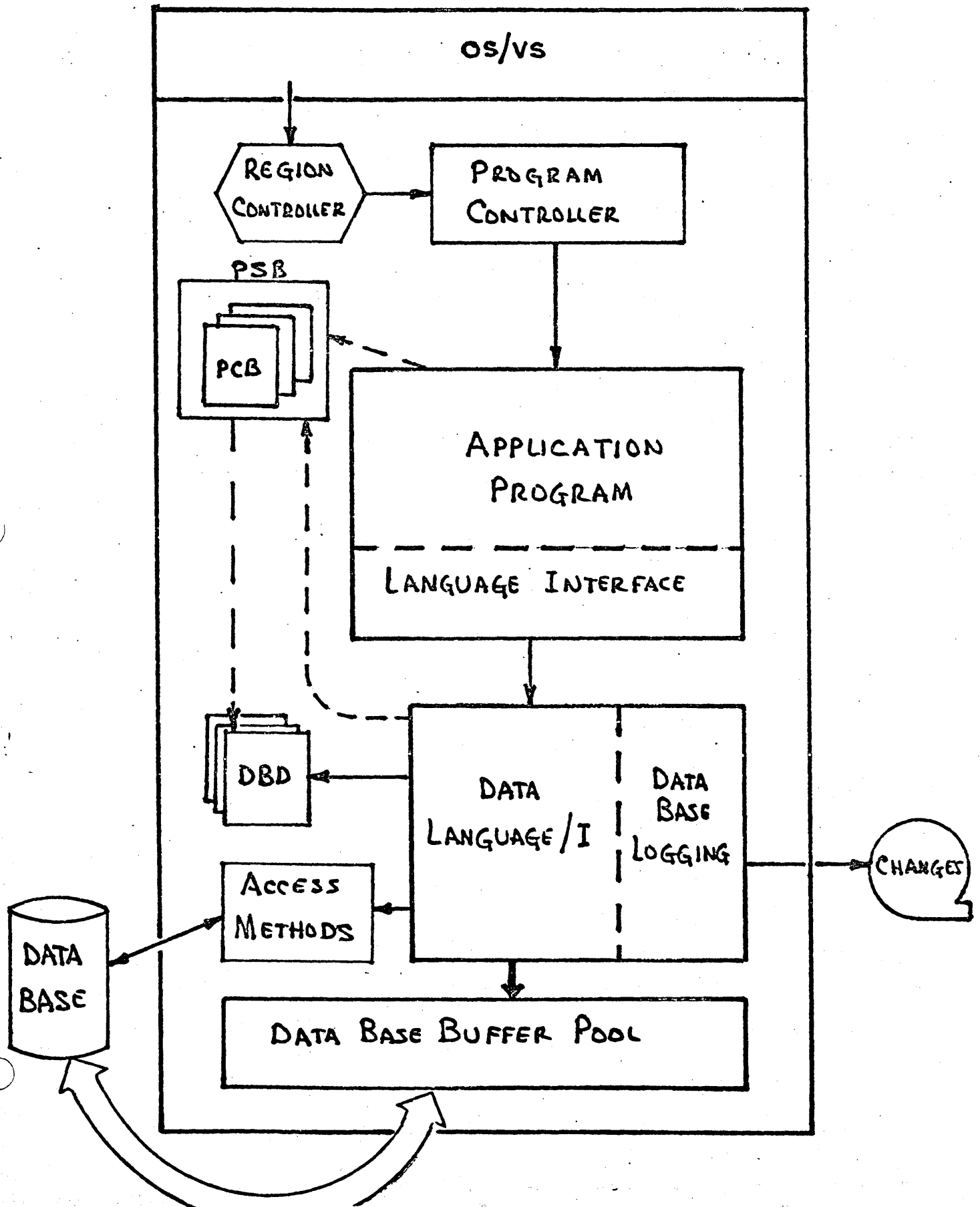
15. When symbolic pointers are used to establish a logical relationship, the pointers exist as part of user data.

True/False True

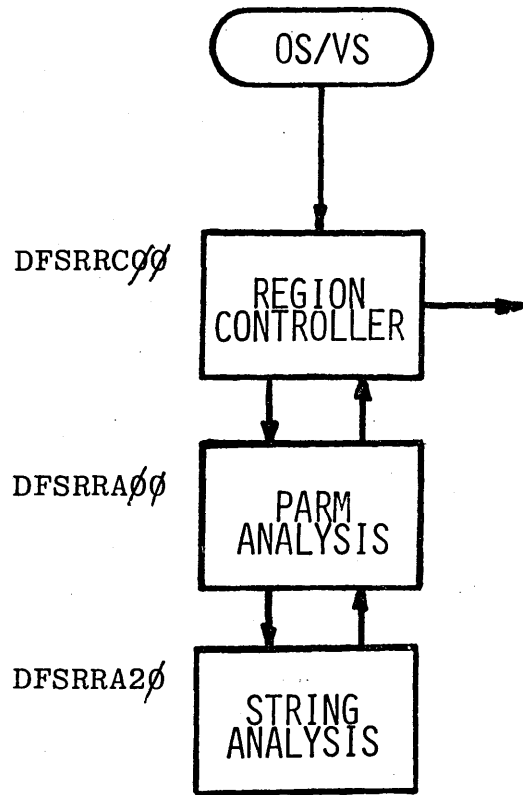
FIGURE 1



BATCH PROCESSING REGION



DLI REGION INITIALIZATION



//STEP EXEC PGM=DFSRRCC00, PARM=^{Card of Region}'DLI, USRPGM, PSB, BUF'^{Ways (SIZE)}

REG 1 Points To EXEC Card Parm List

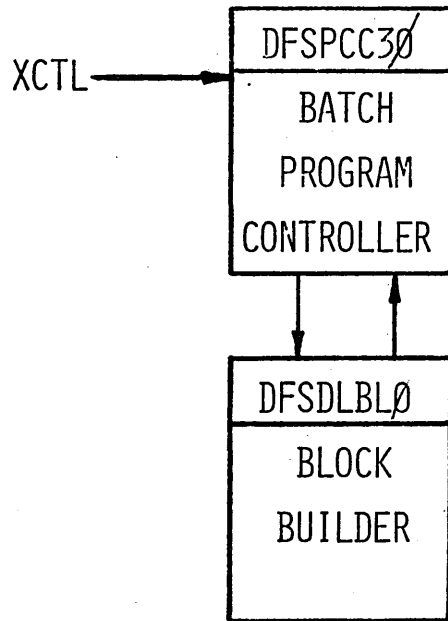
LOAD: DFSRPRX0 - Region Control Blocks
 SECONDARY SCD - Current SVC Numbers
System Control Directory

Breakout And Decode Parm's From EXEC card

AT COMPLETION, XCTL To DFSPCC30 -BATCH PROGRAM CONTROLLER-

*Transient
 Initialization
 Parameters*

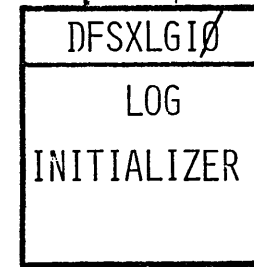
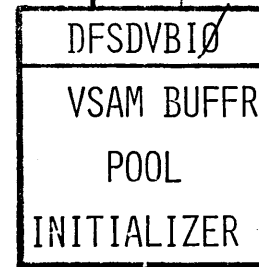
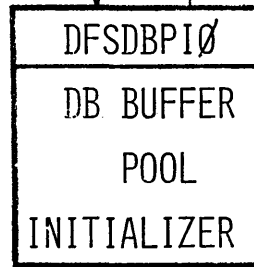
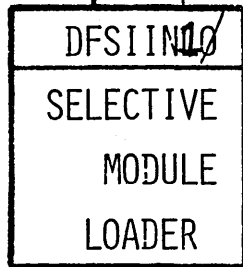
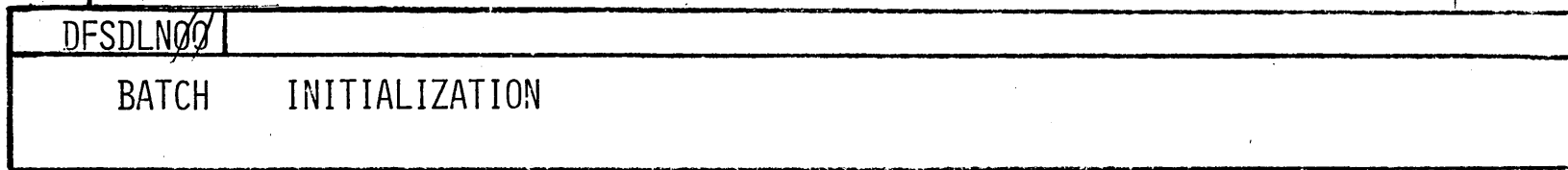
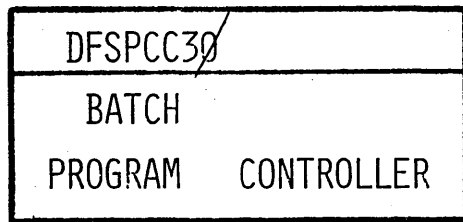
DLI REGION INITIALIZATION (CONT)



- LOAD BATCH NUCLEUS
- MOVE SVC NUMBERS INTO SCD
- OPEN DCBs FOR PSBLIB & DBDLIB
- CALL DFSDLBL0
- BUILD DL/I CONTROL BLOCKS BASED ON PSB NAME TAKEN FROM EXEC CARD PARM FIELD.
- LOAD REGISTER 1 WITH PST ADDRESS
- CLOSE DCBs FOR PSBLIB & DBDLIB

*Partition
Specification
Table*

DLI REGION INITIALIZATION (CONT)



- LOAD REQUIRED DL/I MODULES.
- LOAD SYSTEM REQUIRED & USER SPECIFIED MODULES.

- GETMAIN FOR DB BUFFER POOL.
- FORMAT POOL.

- GETMAIN FOR BH POOL.
- CALL VSAM TO BUILD SUBPOOLS BASED ON DFSVSAMP DATA SET.

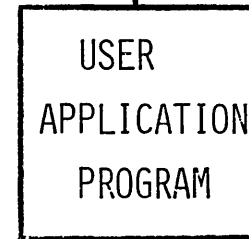
- IF LOGGING, LOG WORK AREA, LOG BUFFERS; OPEN IOBs; FORMAT LOG; PAGE FIX LOG; OPEN LOG.

*Buffer
Handles*

DLI REGION INITIALIZATION (CONT)

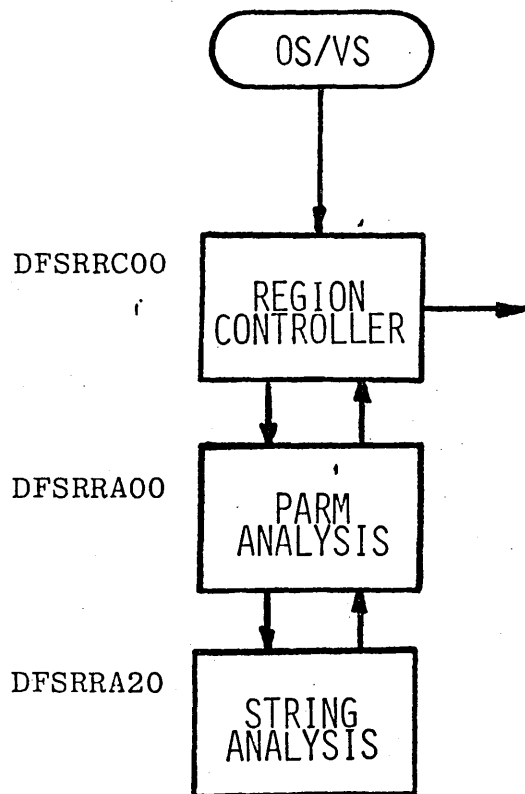


- LOAD STAE EXIT ROUTINE (DFSFL0S0)
 - LOAD REGISTER 1 WITH PCB LIST ADDRESS
- LINK TO USER APPLICATION PROGRAM



(END OF INITIALIZATION)

DBB REGION INITIALIZATION



//STEP EXEC PGM=DFSRRCOO, PARM='DBB' USRPGM, PSB, BUF'

REG 1 Points To EXEC Card Parm List

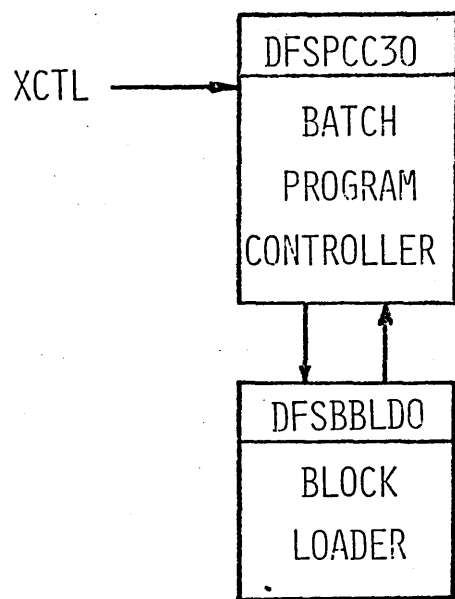
LOAD: DFSPRPX0 - Region Control Blocks

SECONDARY SCD - Current SVC Numbers

Breakout And Decode Parm's From EXEC card

AT COMPLETION, XCTL To DFSPCC30 -BATCH PROGRAM CONTROLLER-

DBB REGION INITIALIZATION (CONT)

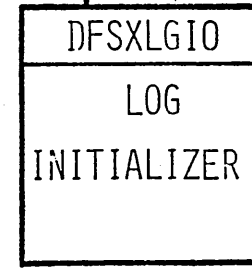
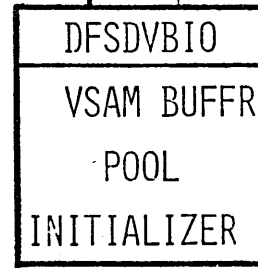
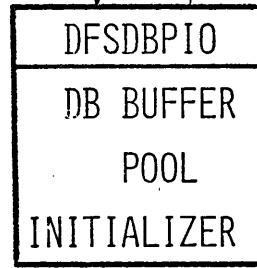
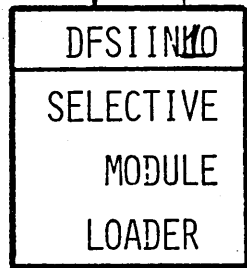
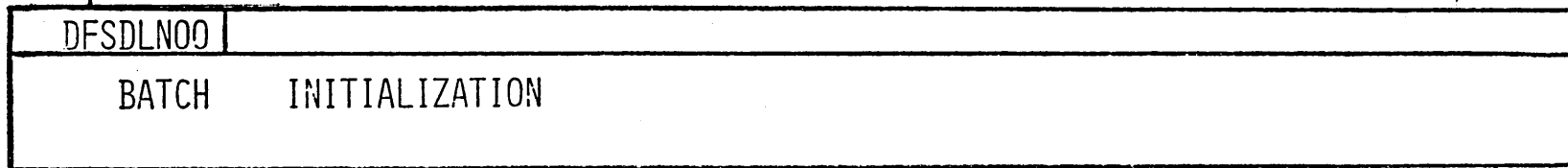
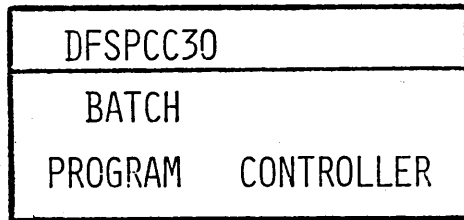


- LOAD BATCH NUCLEUS
- MOVE SVC NUMBERS FROM SECONDARY SCD TO "REAL" SCD
- CALL BLOCK LOADER (Blocks previously built have been stored in ~~ACB~~ ACB LIB)
- OPEN DCB FOR ACBLIB
- LOAD DL/I CONTROL BLOCKS (PREBUILT BY ACBGEN)
- LOAD REGISTER 1 WITH PST ADDRESS
- CLOSE DCB FOR ACBLIB
- RETURN TO DFSPCC30

(DFSDBLx0)

(x = D,P,I,R,M)

DBB REGION INITIALIZATION (CONT)



- LOAD REQUIRED DL/I MODULES.
- LOAD SYSTEM REQUIRED & USER SPECIFIED MODULES.

- GETMAIN FOR DB BUFFER POOL.
- FORMAT POOL.

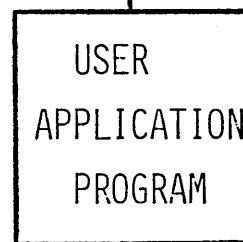
- GETMAIN FOR BH POOL.
- CALL VSAM TO BUILD SUBPOOLS BASED ON DFSVSAMP DATA SET.

- IF LOGGING, LOG WORK AREA, LOG BUFFERS; OPEN IOBs; FORMAT LOG; PAGE FIX LOG; OPEN LOG.

DBB REGION INITIALIZATION (CONT)



- LOAD STAE EXIT ROUTINE (DFSFL0S0)
 - LOAD REGISTER 1 WITH PCB LIST ADDRESS
- LINK TO USER APPLICATION PROGRAM



(END OF INITIALIZATION)

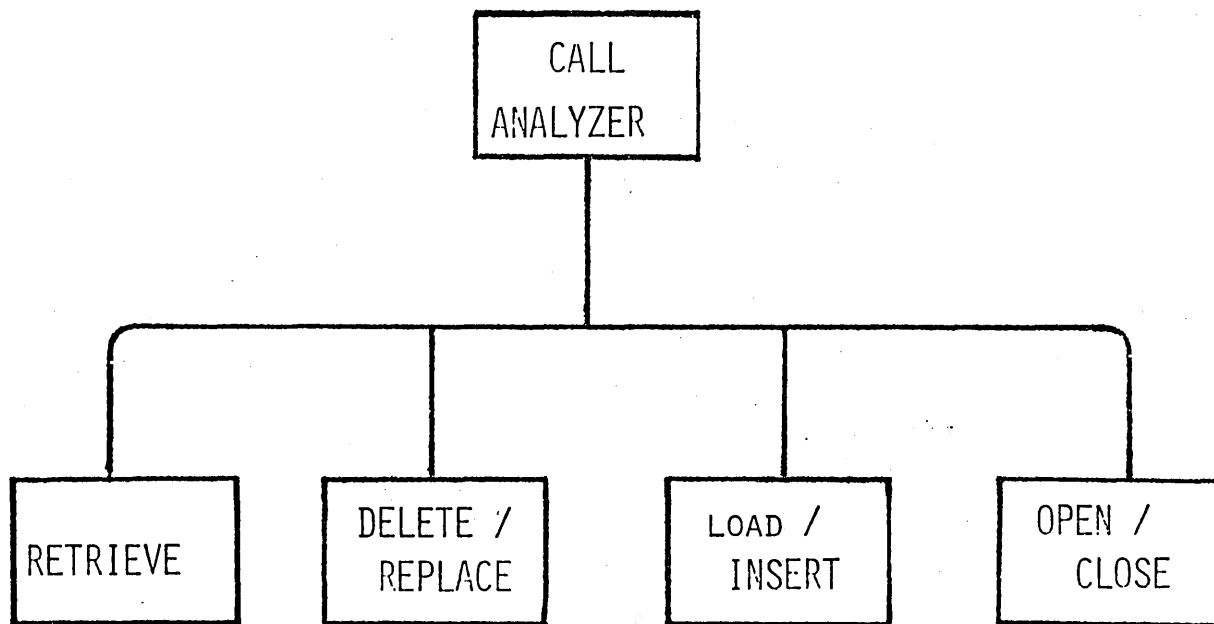
APPLICATION PROGRAM ISSUES A DL/I CALL

- USER PASSES A PARM LIST TO DL/I :
FUNCTION, PCB, I/O AREA, SSA, (SSA), ... (SSA)
- DL/I CALL ANALYZER (DFSDLA00) GOES TO WORK
- REGISTER 1 HAS THE ADDRESS OF THE PST
FIELD PSTIQPRM HAS THE ADDRESS OF THE CALL PARM LIST
- CALL ANALYZER DOES THE FOLLOWING:
 - VALIDATE THE PCB ADDRESS & STORE IN PSTDBPCB
 - STORE USERS I/O AREA ADDRESS IN PSTUSER
 - VERIFY & ENCODE FUNCTION & STORE IN JCBPRESF
 - VALIDATE FIELD NAME (IN SSA)
 - TEST FOR INVALID CALL (WITH OR WITHOUT SSA)
 - TEST FOR PROCOPT VIOLATION
 - DATA BASES OPEN? (CHECK JCBOPEN BIT)
 - No - CALL DL/I OPEN/CLOSE (DFSDLOC0)
 - Yes - CALL DL/I ACTION MODULE BASED ON FUNCTION
 - RETRIEVE - DFSDLR00
 - LOAD/ISRT - DFSDDL00
 - DLET/REPL - DFSDLD00

Job Control Block



DFSDLAOO



Data Base Open/Close

DFSDLOCQ

- OPENS A SINGLE DCB/ACB, A DATA SET GROUP, ALL DCB/ACBs FOR A PCB, FOR A DMB, FOR ALL DMBs

DBD →
(External)

Data
Management
Block (internal)

- USE IMS ENQ/DEQ TO SERIALIZE OPENS FOR DCB/ACB

- GET STORAGE FOR BISM 'IN STORAGE' INDEXES

- BUILD OPEN AND CLOSE LOG RECORDS

- LOAD/DELETE HDAM RANDOMIZING MODULES

- LOAD/DELETE SEGMENT EDIT COMPRESSION MODULES

- INITIALIZE AMPs, DSGs AND JCBs

Access
Method
Prefix (tell whether
we have ACB or
DCB)

- IF HSAM USING BSAM, DO GETMAIN FOR BSAM BUFFER

- SET JCBORGN TO INDICATE OPEN/CLOSE STATUS

RETRIEVE

DFSDLR00

FUNCTIONS -

- ✘ RECEIVES CONTROL FROM CALL ANALYZER
- ✘ CALLS BUFFER HANDLER FOR PHYSICAL I/O
- ✘ DEBLOCKS SEGMENTS FOR HS ORGANIZATION
- ✘ ESTABLISHES 'POSITION' IN THE DATA BASE
- ✘ IF POSITION AVAILABLE FOR REQUESTED SEGMENT,
NO BUFFER HANDLER CALL
- ✘ UPDATES POSITION AND LENGTH IN DB CONTROL BLOCKS
- ✘ PERFORMS POSITIONING FOR INSERTS
- ✘ FOR 'PI' CALL ENQ-DEQ FOR POSITION AND ENQ TESTS
- ✘ POSITION TO PHYSICAL ROOT FOR HISAM, HIDAM, HDAM
- ✘ FOR 'PI' WILL IWAIT TASK IF NECESSARY

*Program
Isolation*

RETRIEVE POSITION

SDBPOSP	SDBPOSC	SDBPOSN
PREVIOUS	CURRENT	NEXT

HSAM: NO POSITIONING USED

HISAM:
PREV - NOT USED
CURR - RELATIVE RECORD NUMBER (RRN) OF LRECL
NEXT - OFFSET INTO LRECL FOR SEGMENT

HDAM/HIDAM:

PREV - RELATIVE BYTE NUMBER (RbN) OF PRIOR SEGMENT ON CHAIN *(Depends on sequence)*
CURR - RbN OF CURRENTLY ESTABLISHED POSITION
NEXT - RbN OF NEXT SEGMENT ON CHAIN

LOAD/INSERT

DFSDDLEØ

FUNCTION

- DOES LOADS AND INSERTS FOR ALL ORGANIZATIONS
- RETRIEVE POSITIONS FOR ALL INSERTS EXCEPT ROOTS IN HISAM

FLOW - SEPARATE LOGIC WITH COMMON SUBROUTINES FOR

- HSAM LOAD
- HISAM LOAD
- HISAM DEPENDENT INSERT
- HISAM ROOT INSERT
- HDAM OR HIDAM LOAD OR INSERT
- HISAM REPLACE FOR VARIABLE LENGTH SEGMENTS
- HD REPLACE FOR VARIABLE LENGTH SEGMENTS

PERTINENT INFORMATION PASSED TO INSERT

- PARTITION SPECIFICATION TABLE (PST)
- PCB (FROM THE DL/I CALL)
- FIRST LEVEL TO BE INSERTED
- LOGICAL SEGMENT CONTROL BLOCK (SDB)
- PHYSICAL SEGMENT CONTROL BLOCK (PSDB)
- POINTERS TO ALL OTHER DB CONTROL BLOCKS

Segment
Descriptor
Block

CURRENT POSITION

LEVEL TABLE

LEVLEV

LEVSDB

SDB

SDBSYM

SDBKEYFD

SDBPOSP
SDBPOSC
SDBPOSN

DBPCB

DBPCBDBD

DBPCBKFD

-- CONCATENATED KEY -----

Delete/Replace

DFSDL00

REPLACE FUNCTIONS

- ENSURE KEY FIELD NOT ALTERED
- CHECKS FOR REPLACE RULE VIOLATION
- INTERFACE WITH INDEX MAINTENANCE IF INDEXED SEGMENT
- INTERFACE WITH LOAD/INSERT IF VARIABLE LENGTH
- INTERFACE WITH ENQ/DEQ IF PROGRAM ISOLATION IN EFFECT

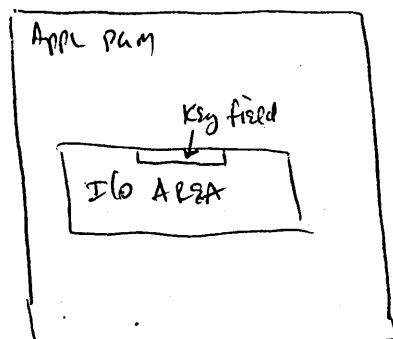
DELETE FUNCTIONS

- ENSURE KEY FIELD NOT ALTERED
- CHECKS FOR DELETE RULE VIOLATION
- SCANS DATA BASE RECORD

-HISAM- ONLY IF LOGICAL RELATIONSHIP OR INDEXED SEGMENTS IN HIERARCHY

-HD- ALWAYS

- ▲ ● INTERFACE WITH INDEX MAINTENANCE IF INDEX SOURCE SEGMENT
- ▲ ● INTERFACE WITH SPACE MANAGEMENT IF HD ORGANIZATION
- ▲ ● INTERFACE WITH ENQ/DEQ IF PROGRAM ISOLATION IN EFFECT



Work Areas Used By Delete/Replace (DLTWA)

- OBTAINED BY GETMAIN FROM SUBPOOL ZERO
- MAINTAINS CONCATENATED KEYS
- MAINTAINS POSITION OF SEGMENTS IN THE DATA BASE RECORD
- USED DURING DELETE SCAN
- USED DURING INDEX MAINTENANCE FOR REPLACE
- FIRST WORK AREA POINTED TO BY PSTDLTWA
- WORK AREAS CHAINED TOGETHER

VSAM Work Area Manager

DFSDLBO

- RESOLVE ALL NEW RBA'S IN DELETE/REPLACE WORK AREA WHEN A
CI OR CA SPLIT OCCURS

INDEX MAINTENANCE

DFSDXMT0

MAINTAINS PRIMARY INDICES FOR HIDAM DATA BASES

MAINTAINS SECONDARY INDEXES

MAKES INTERNAL DL/I CALLS

ISRT/DLET/REPL INDEX SEGMENTS

ALL UPDATES ARE LOGGED

VSAM Simple HISAM


Delete/Replace

DFSULDVO

REPLACE FUNCTIONS

- ENSURE KEY FIELD NOT ALTERED
- COMPARE OLD SEGMENT TO NEW
- IF UNEQUAL :
 - LOG OLD AND NEW
 - REPLACE OLD WITH NEW

DELETE FUNCTIONS

- ENSURE KEY FIELD NOT ALTERED
- LOG OLD SEGMENT
- ISSUE VSAM ERASE 

HD SPACE MANAGEMENT BIT MAP

- ALLOCATED EVERY 'N' BLOCKS

$$N=8(\text{BLOCKSIZE}-4-4(\#\text{RAP}'\text{S}))$$

- FILLS DATA STORAGE AREA OF FIRST BLOCK OF EACH SET OF 'N' BLOCKS
- ONE BIT REPRESENTS ONE BLOCK
- BIT 'ON' - AT LEAST ONE SPACE LARGE ENOUGH TO HOLD LARGEST SEGMENT OF DATA SET GROUP

HIERARCHICAL-DIRECT SPACE MANAGEMENT

BLOCK = 104 BYTES (DECIMAL)

- EMPTY

FSEAP	RAP	FSE	88 BYTES OF			
08 0	0	0 60 ID	FREE SPACE			

- FILLED - 2 SEGMENTS PLUS 2 FREE AREAS (SMALL ONES)

FSEAP	RAP	SKILL	FSE	FREE	NAME	FSE	FREE
30 0	2044	SEGMENT	52 0E ID	SPACE	SEGMENT	00 16 ID	SPACE

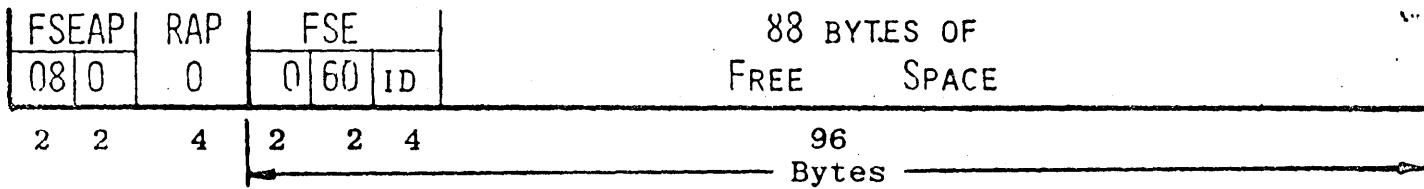
- DELETE NAME SEGMENT

FSEAP	RAP	SKILL	FSE	48 BYTES OF			
30 0	2044	SEGMENT	00 38 ID	FREE SPACE			

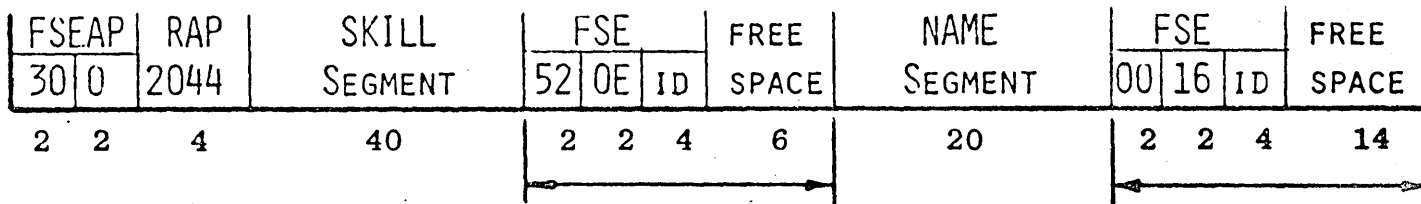
HIERARCHICAL-DIRECT SPACE MANAGEMENT

BLOCK = 104 BYTES (DECIMAL)

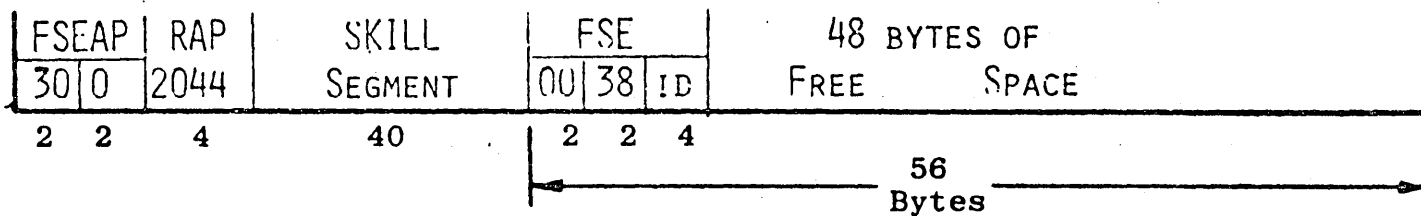
• EMPTY



• FILLED - 2 SEGMENTS PLUS 2 FREE AREAS (SMALL ONES)



• DELETE NAME SEGMENT



HD SEARCH CRITERIA

1. SAME BLOCK WITH NO CHANGE IN ITS BIT MAP
CLOSEST AVAILABLE SPACE:
 - (1) EQUAL TO SEGMENT LENGTH
 - (2) 2 TIMES ITS SEGMENT LENGTH
 - (3) SEGMENT LENGTH + MINIMUM SEGMENT SIZE
 - (4) GREATER THAN SEGMENT LENGTH
2. SAME BLOCK WHERE BIT MAP WILL CHANGE
CLOSEST AVAILABLE SPACE:
 - (1) 2 TIMES ITS SEGMENT LENGTH
 - (2) SEGMENT LENGTH + MINIMUM SEGMENT SIZE
 - (3) SEGMENT LENGTH
3. ANY BLOCK CURRENTLY IN BUFFER POOL ON SAME TRACK
4. ANY BLOCK CURRENTLY IN BUFFER POOL ON SAME CYLINDER
5. ANY BLOCK IN BIT MAP ON SAME TRACK WHERE MAXIMUM SEGMENT LENGTH EXISTS
6. ANY BLOCK IN BIT MAP ON SAME CYLINDER WHERE MAXIMUM SEGMENT LENGTH EXISTS
7. ANY BLOCK CURRENTLY IN BUFFER POOL WITHIN + N CYLINDERS
8. ANY BLOCK IN BIT MAP ON + N CYLINDERS WHERE MAXIMUM SEGMENT LENGTH EXISTS
9. ANY BLOCK IN BUFFER POOL WITHIN DATA SET
10. ANY BLOCK IN BIT MAP WITHIN DATA SET.

*Upside: keep segment as close to
subunits as possible*

DL / I CONTROL BLOCKS

SOURCES

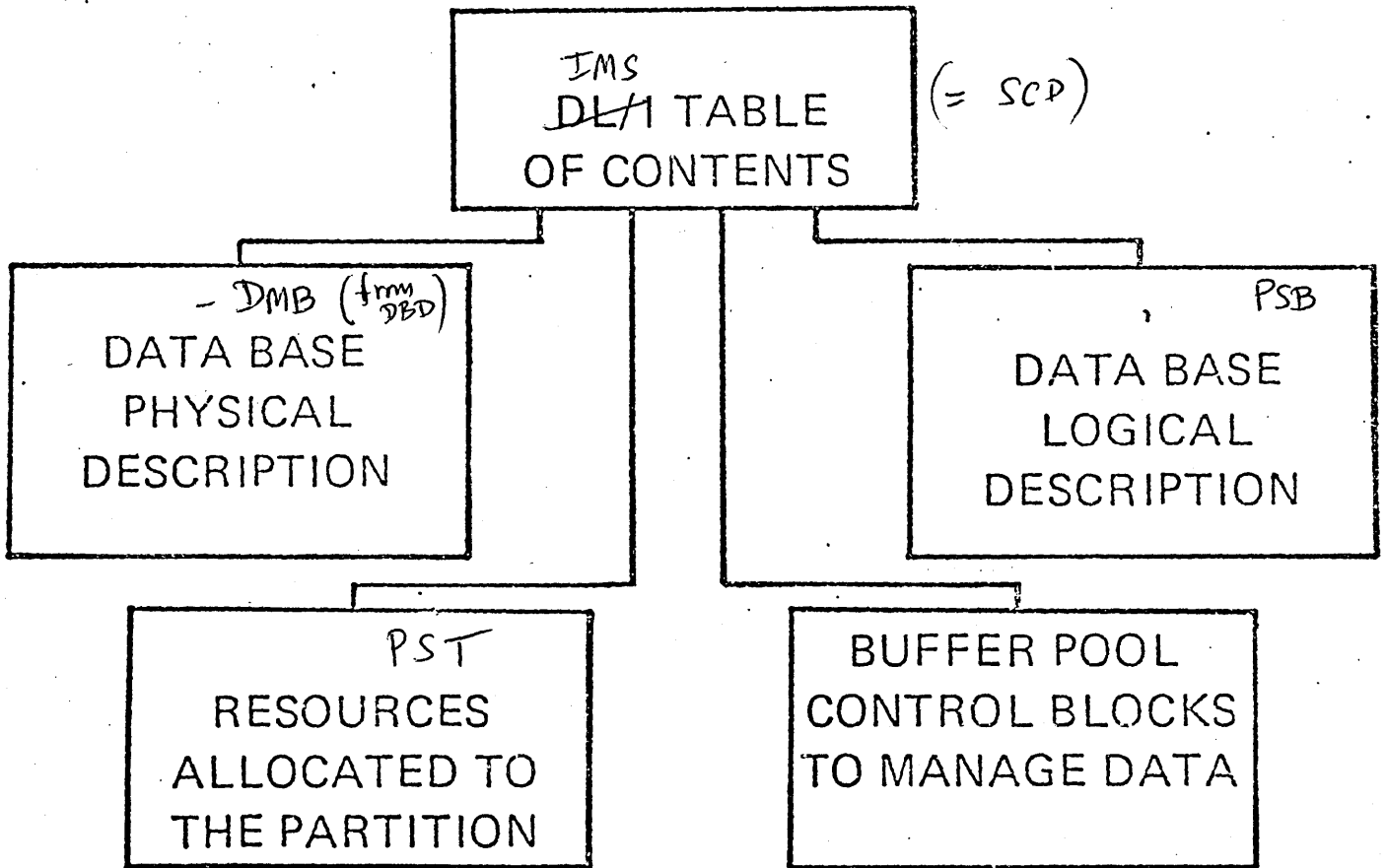
P S B

PCB
SENSEG

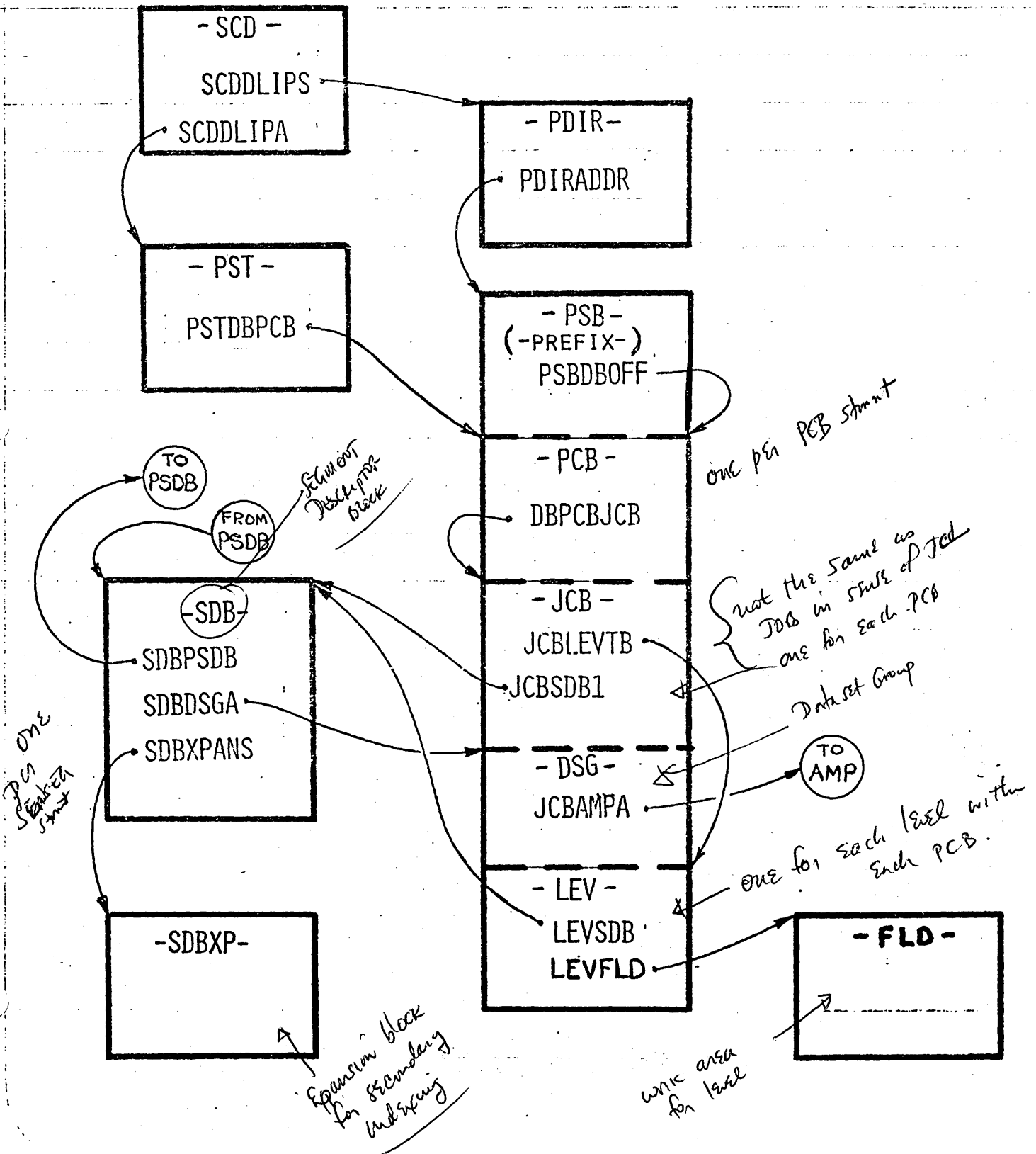
D B D

DBD
DATASET
SEGM
FIELD
LCHILD
XDFLD

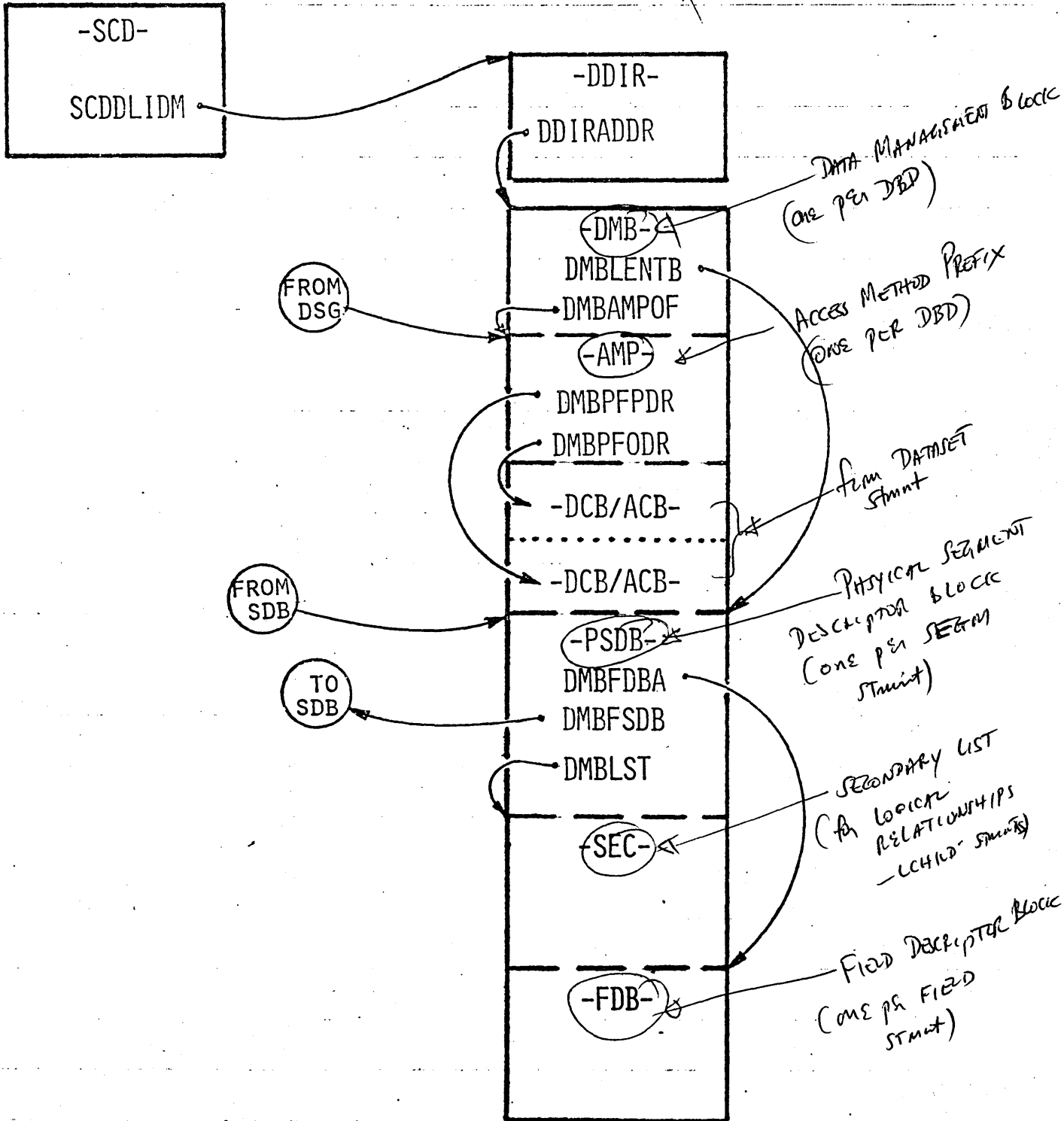
DL/I MAIN CONTROL BLOCKS



PSB GENERATED BLOCKS



DBD GENERATED BLOCKS



JCB CALL TRACE TABLE

Current
↓

0000|0000|0140|0440|04C5| 1|1|1

* FIVE TWO-BYTE ENTRIES

BYTE ONE

01-GU/GHU

03-GN/GHN

04-GNP/GHNP

21-REPL

22-DLET

41-ISRT

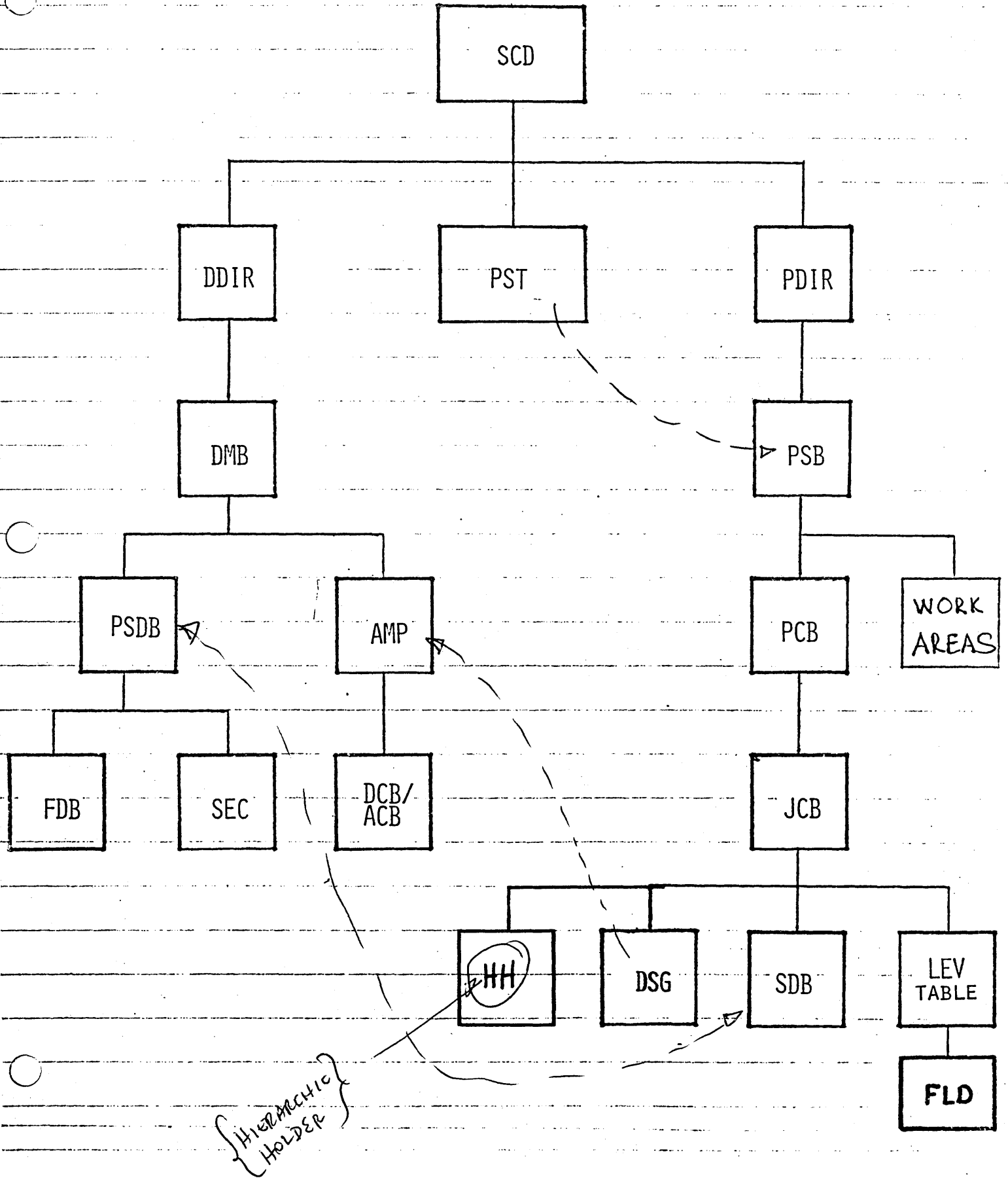
BYTE TWO

RIGHT BYTE OF THE
STATUS CODE

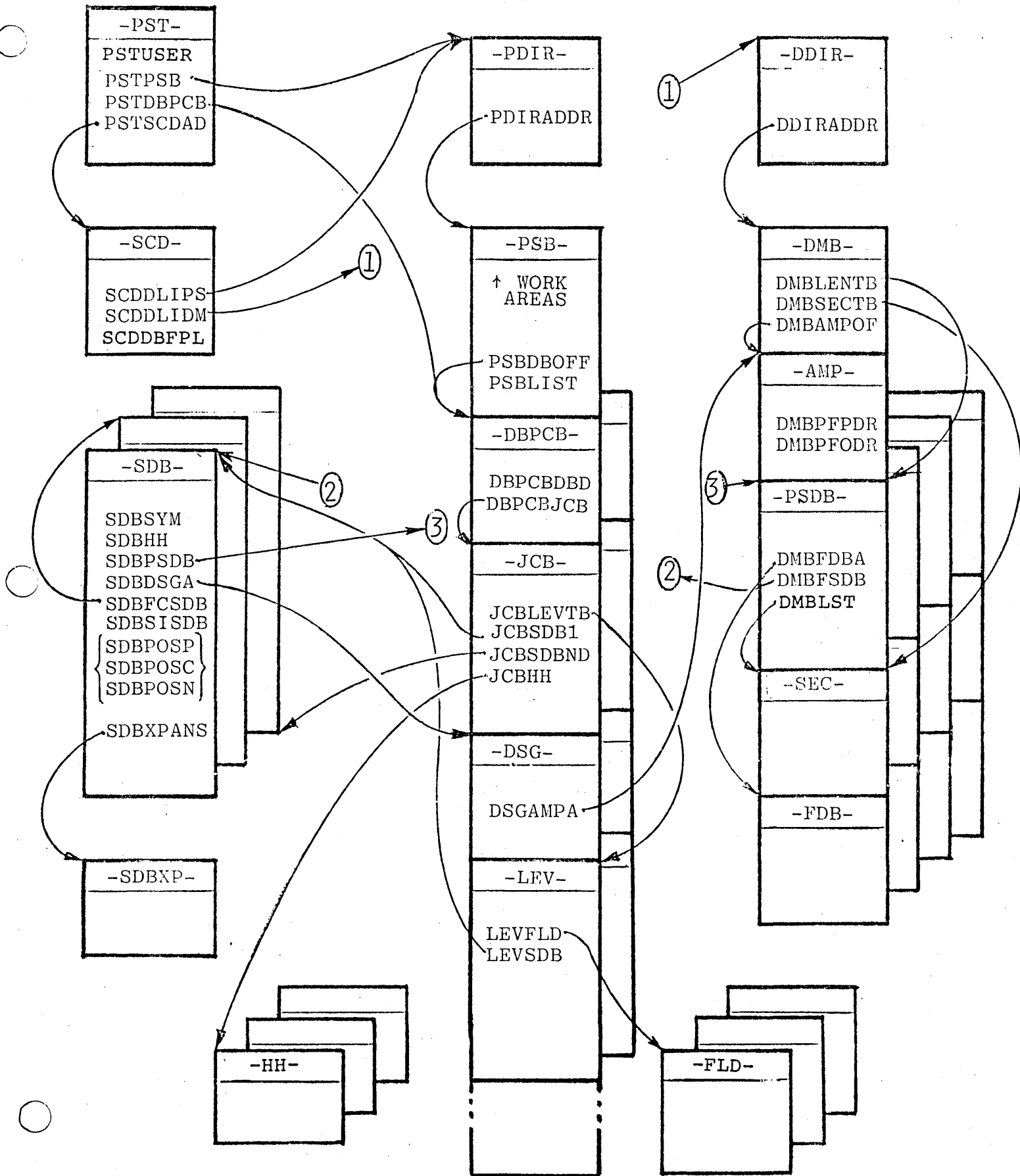
* MOST RECENT ON RIGHT

CODE NO. 1111

DL / I B L O C K S



1.1.3 DB Ctl BIKs



C P A C

(DMBCPAC)

"COMPRESSSIONAL SEGMENT BLOCK"

- A CSECT FOR EACH COMPRESSABLE SEGMENT
- EACH IS EIGHT (8) WORDS LONG
- PART OF THE DMB
- MAPS OUT COMPRESSION ROUTINE INFORMATION
- ONE CPAC PER PSDB - IE; ONE PER PHYSICAL SEGM.

D A C S

20 bytes

(DMBDACS)

Data Entry Data Base

- USED IN SUPPORT OF FAST PATH DEDBs

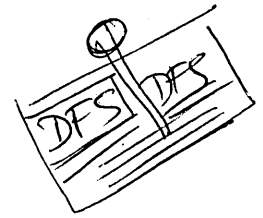


X M P R M

(DMBXMPRM)

"INDEX MAINTENANCE PARAMETER LIST"

- USED BY INDEX MAINTENANCE WITH SECONDARY INDEXING ANYTIME A SOURCE SEGMENT IS UPDATED.
- CONTAINS XDFLD AND OTHER INFORMATION ON THE SOURCE SEGMENT.



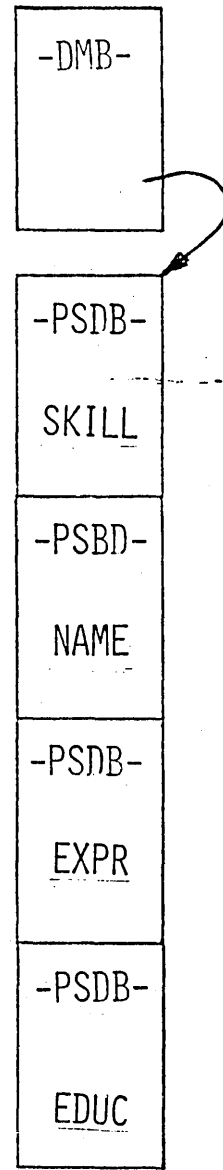
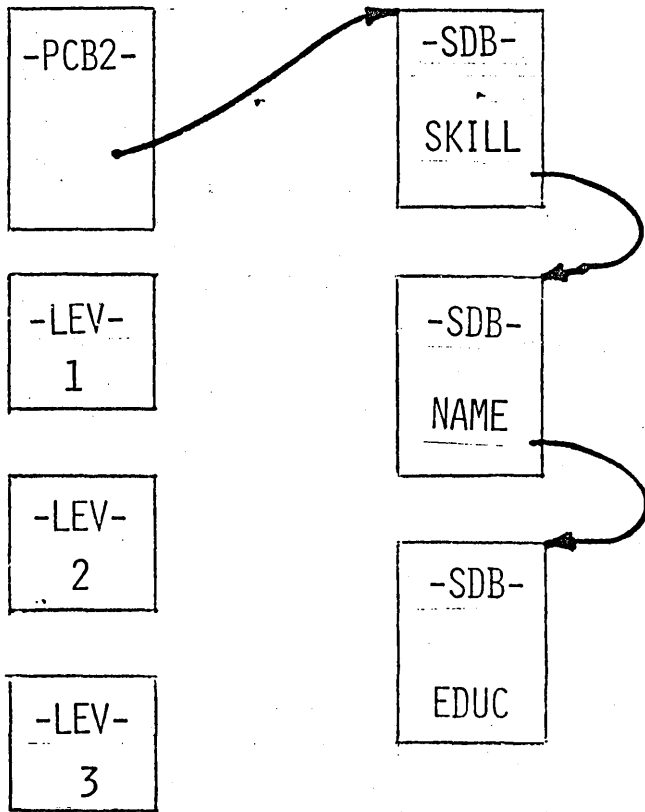
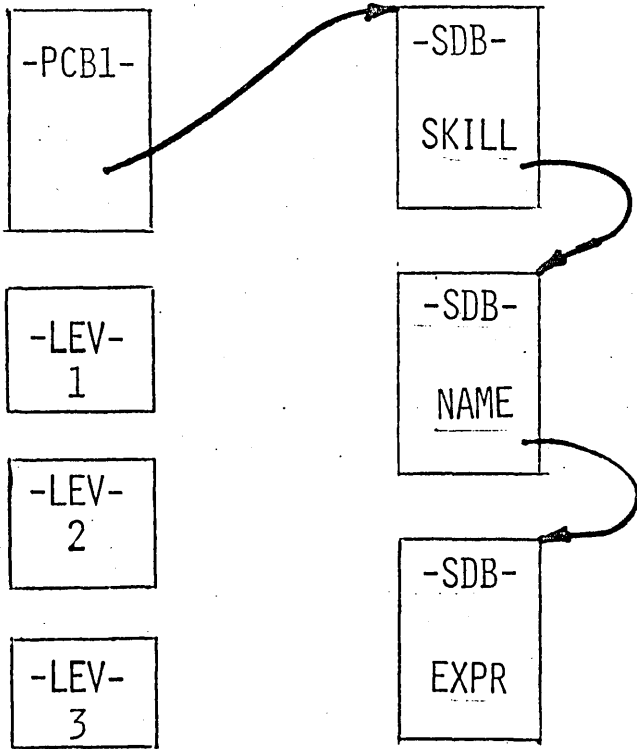
HH

(HHSTRT)

"HIERARCHIC HOLDER"

- BUILT BY RETRIEVE MODULE (DFSDLR00)
- USED TO HOLD POSITION INFO. WITHIN A DATA BASE
- FIRST ONE POINTED TO IN JCB (JCBHH +48)
- ALL HHs ARE FOUR WORDS LONG & CHAINED TOGETHER
- EACH SDB CONTAINS A RELATIVE NUMBER OF THE HHSTRT
FOR THAT SEGMENT (SDBHH + \emptyset E)
- HHSTRT OF CURRENT SEGMENT POSITION HELD IN THE
JCB (JCBACHH + 2D)

SDB - PSDB RELATIONSHIP

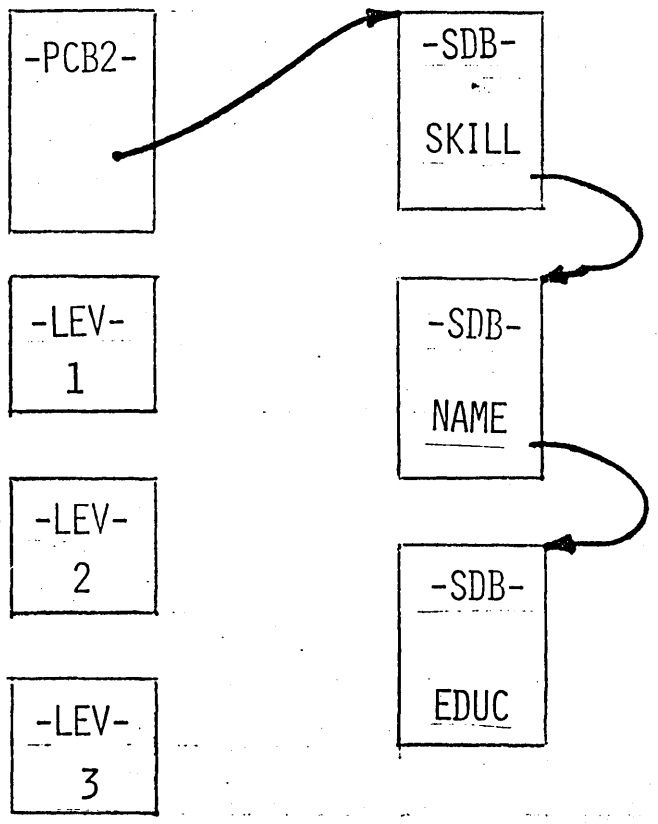
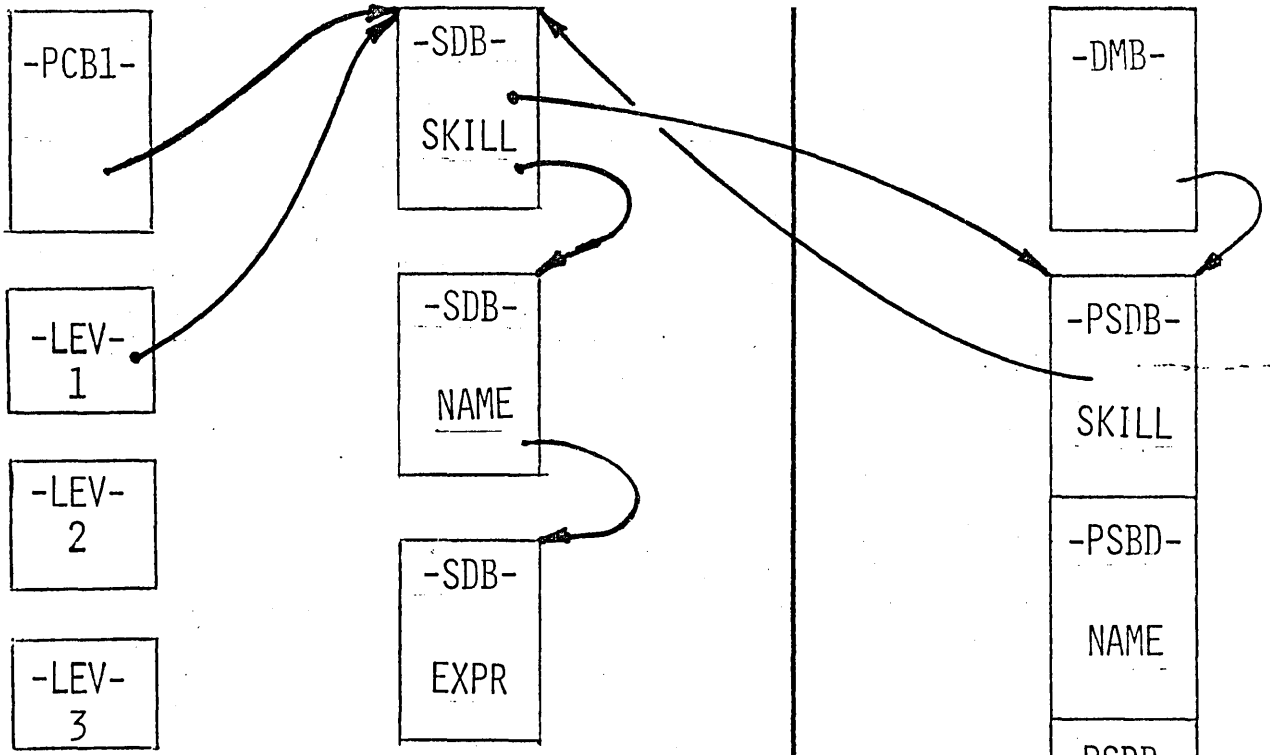


CALL:

I/O:

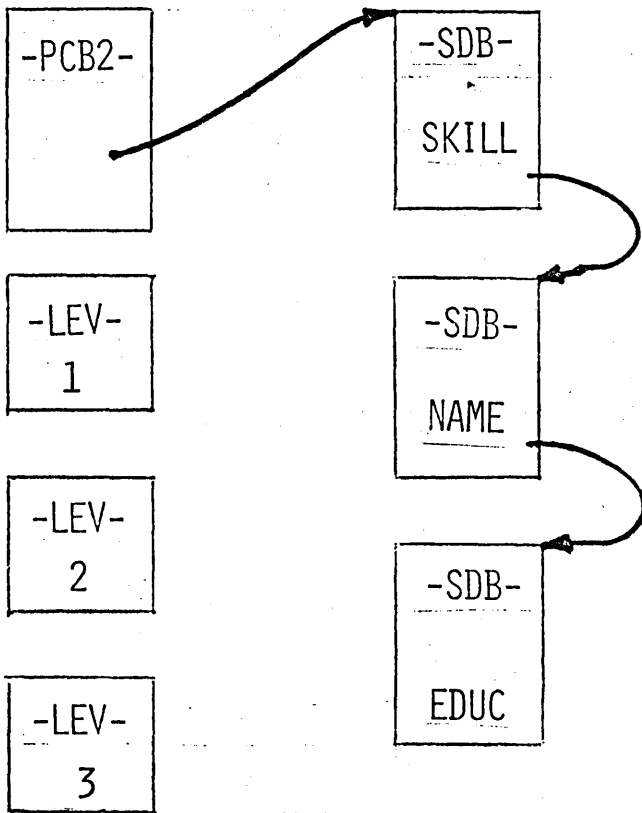
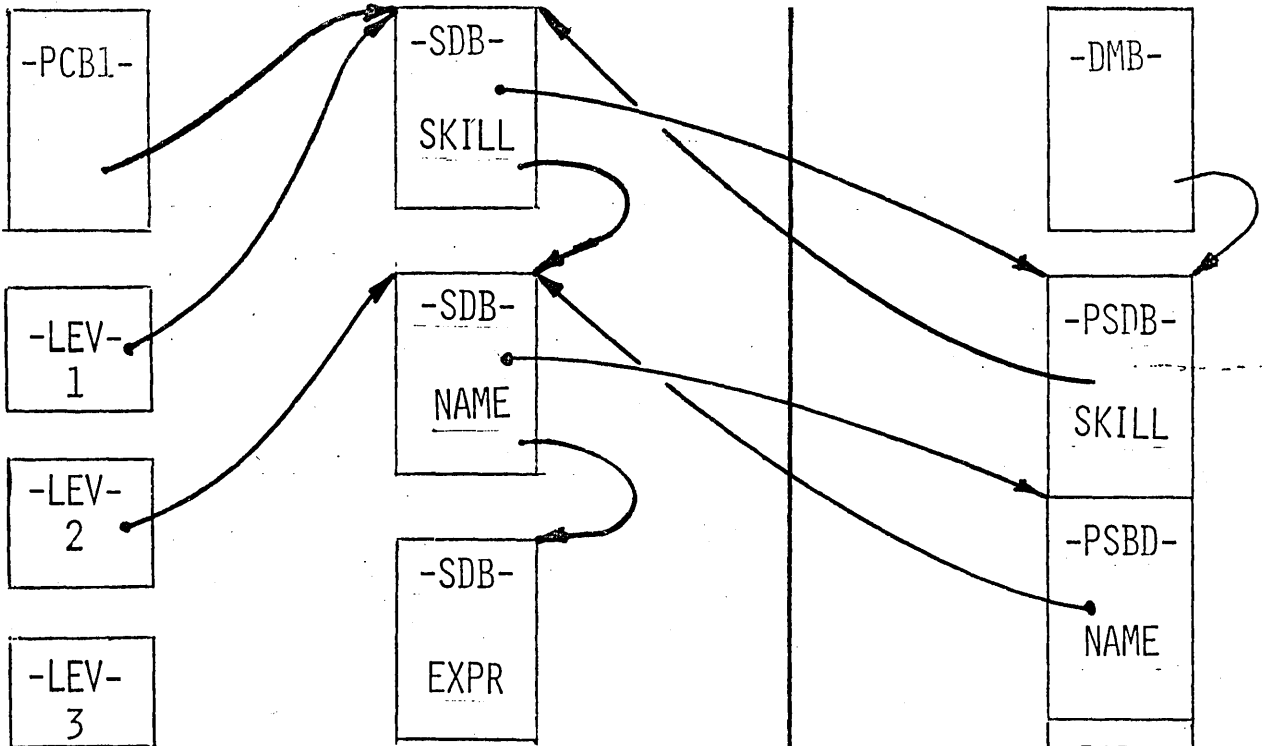
KEYFDBK:

SDB - PSDB RELATIONSHIP



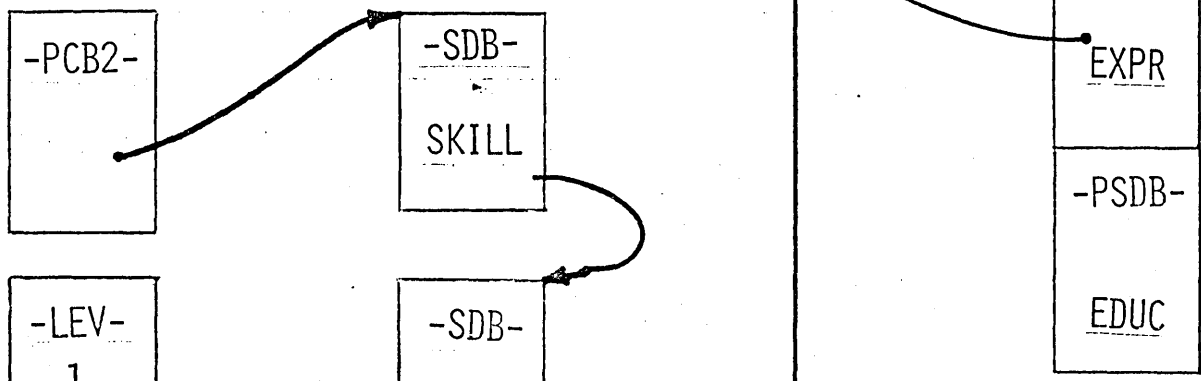
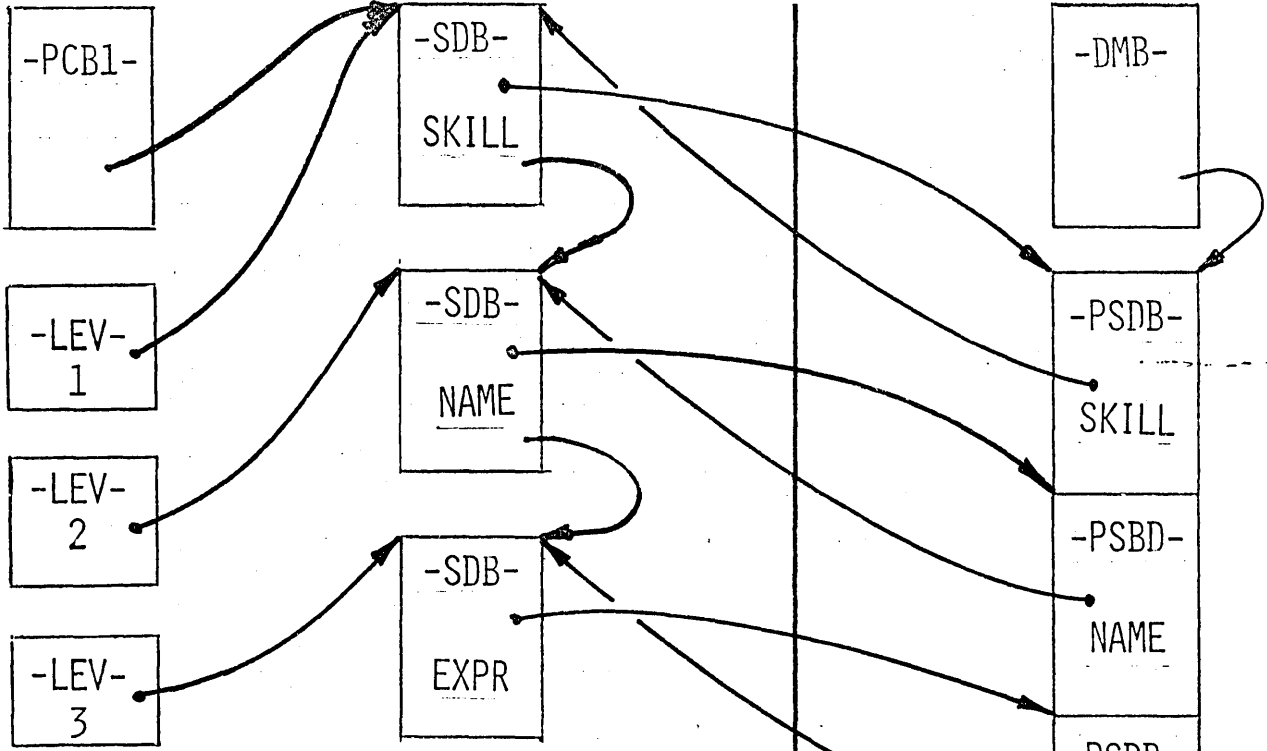
CALL: GU SKILL (SKCLASS= PROG)
 PCB1
 I/O: PROG STAFF PROGRAMMER
 KeyFDBK: PROG

SDB - PSDB RELATIONSHIP



CALL: GN NAME
 PCB1
 I/O: 014430 JONES, JOHN PAUL
 KEYFDBK: PROG JONES

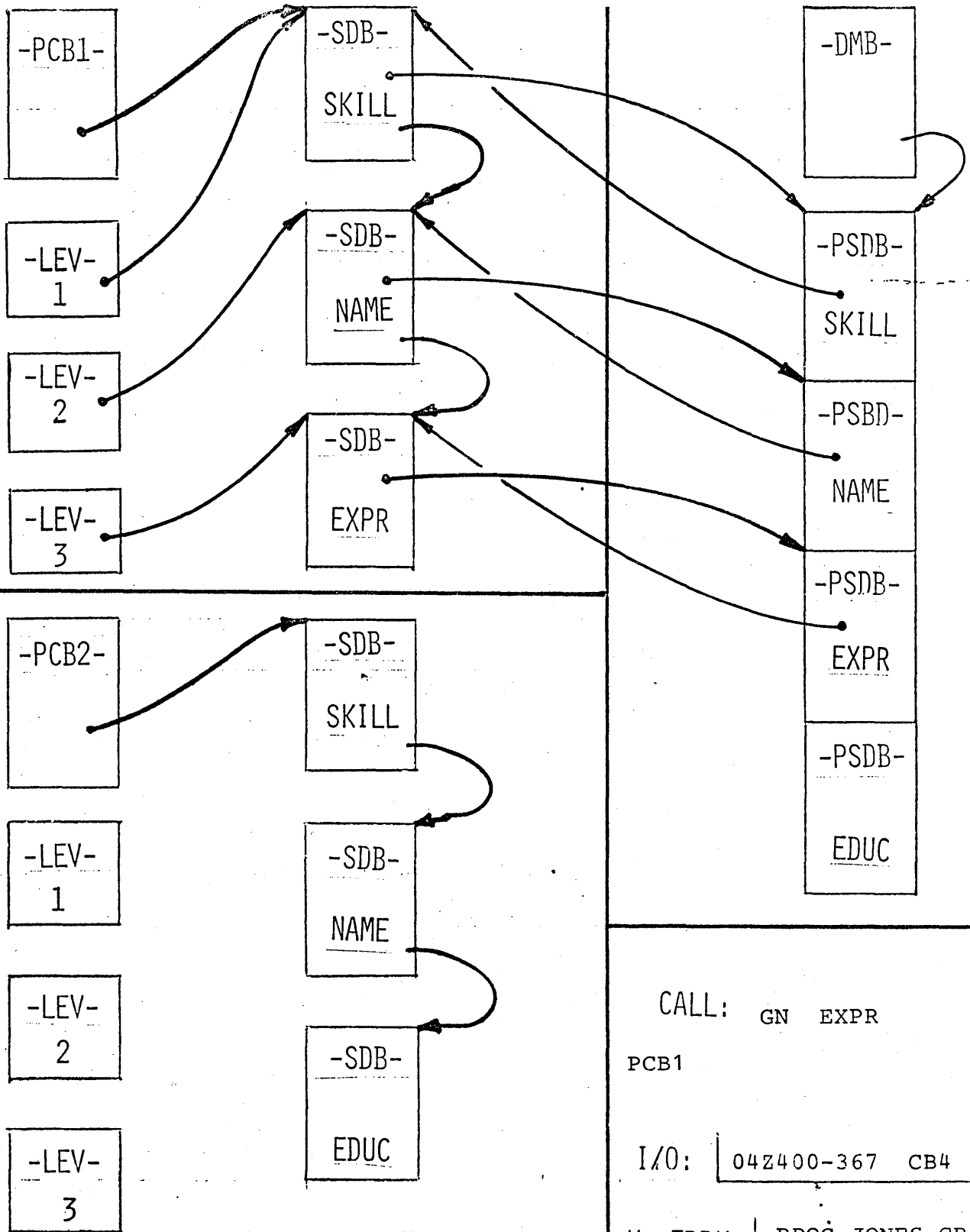
SDB - PSDB RELATIONSHIP



-LEV- 1
-LEV- 2
-LEV- 3

CALL: GN EXPR
PCB1
I/O: 08C400-262 AL5
KEYFDBK: PROG JONES AL5

SDB - PSDB RELATIONSHIP



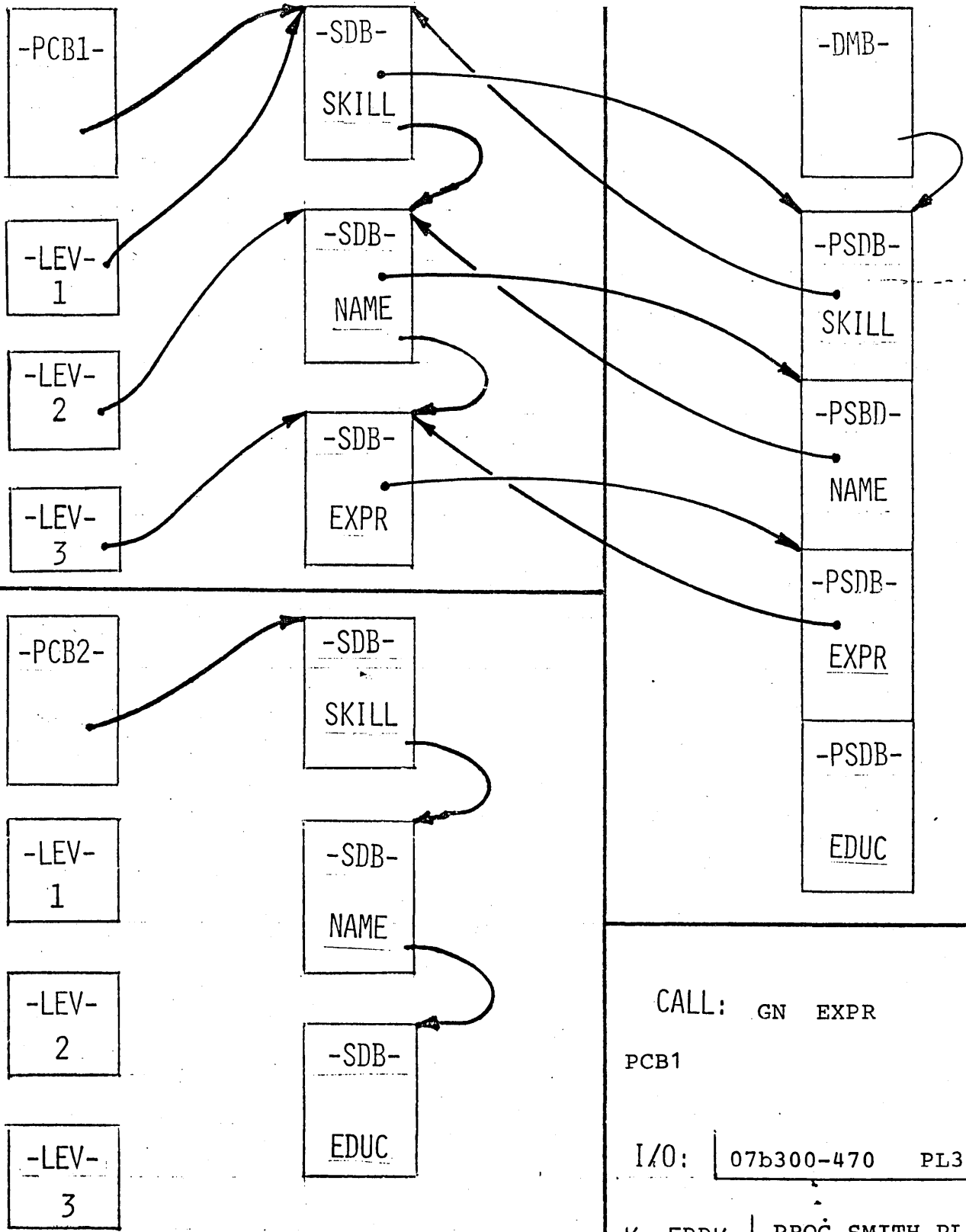
CALL: GN EXPR

PCB1

I/O: 04Z400-367 CB4

KEYFDBK: PROG JONES CB4

SDB - PSDB RELATIONSHIP



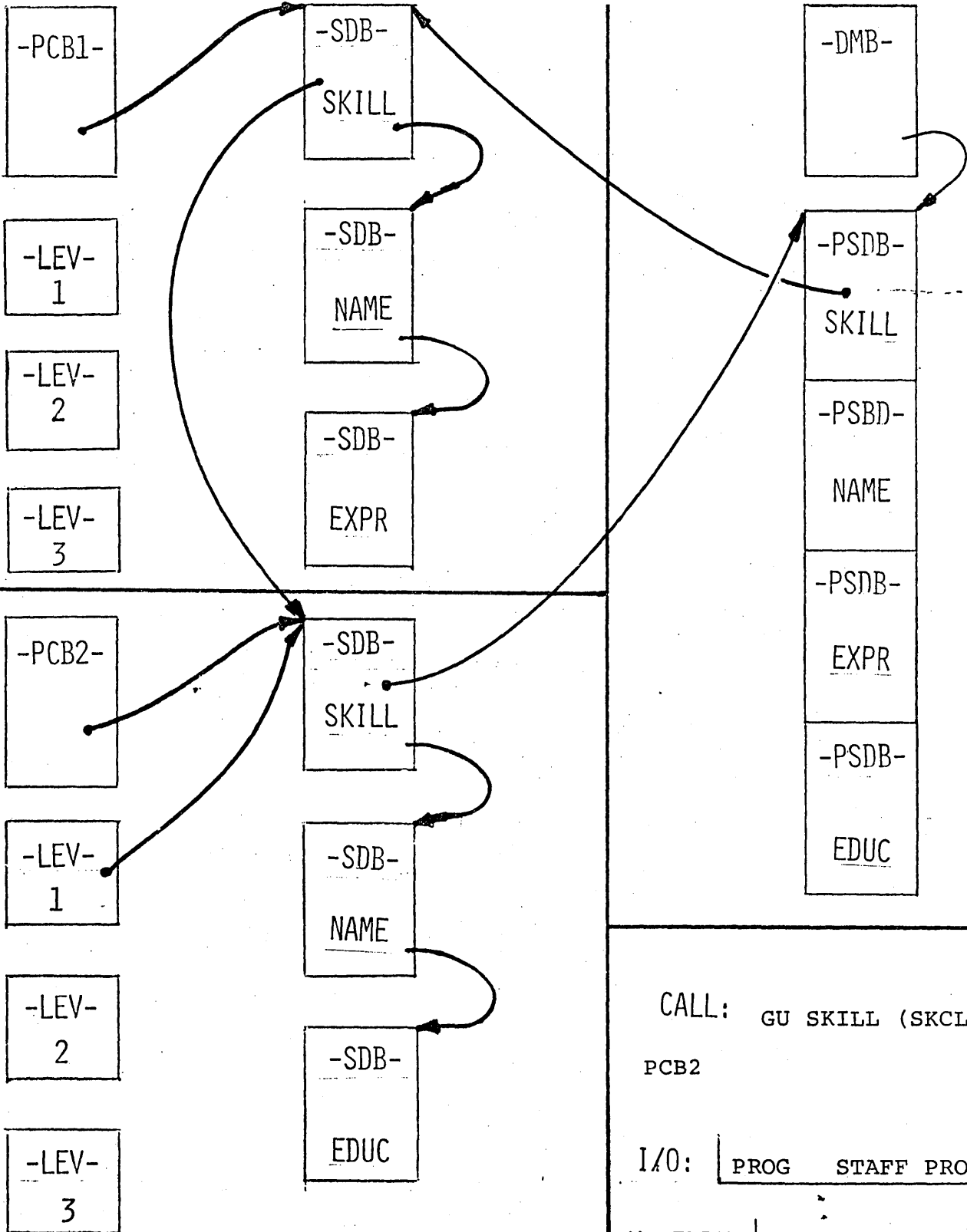
CALL: GN EXPR

PCB1

I/O: 07b300-470 PL3

KEYFDBK: PROG SMITH PL3

SDB - PSDB RELATIONSHIP



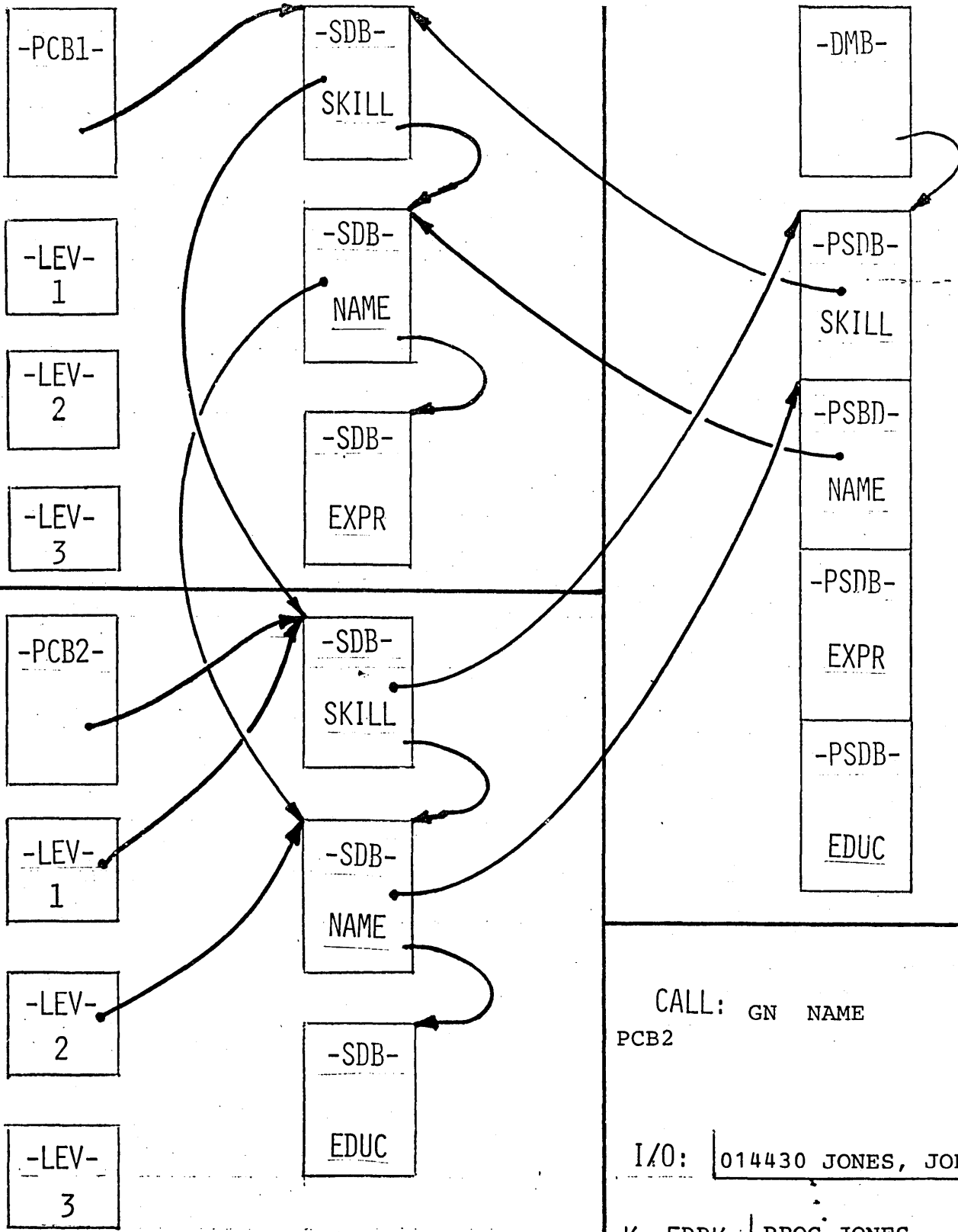
CALL: GU SKILL (SKCLASS = PROG)

PCB2

I/O: PROG STAFF PROGRAMMER

KEYFDBK: PROG

SDB - PSDB RELATIONSHIP

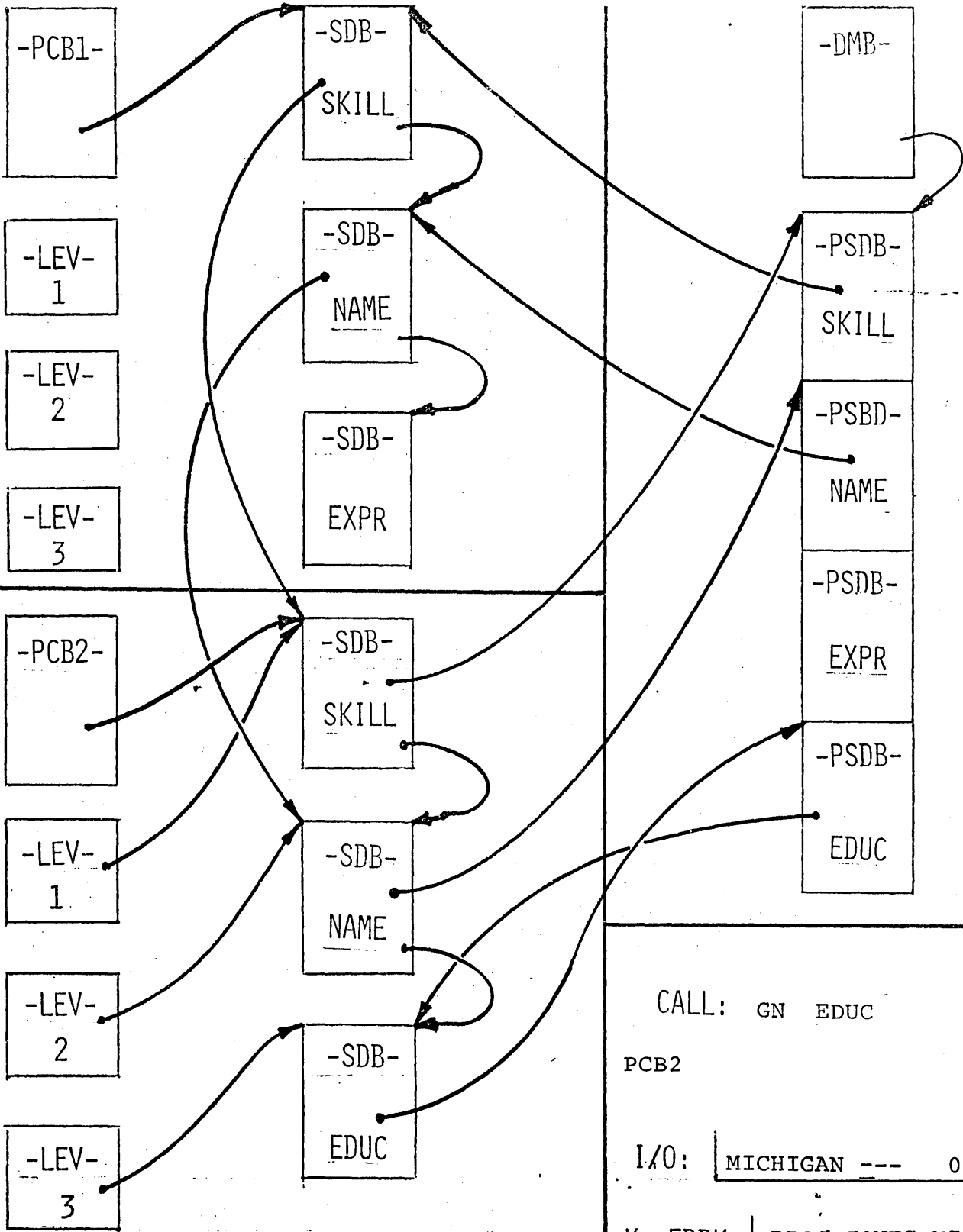


CALL: GN NAME
 PCB2

I/O: 014430 JONES, JOHN PAUL

KEYFDBK: PROG JONES

SDB - PSDB RELATIONSHIP



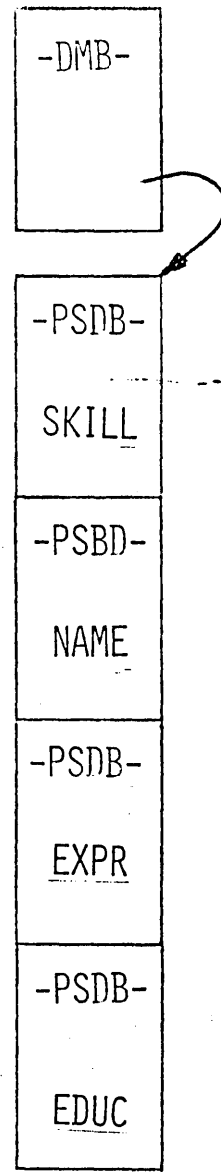
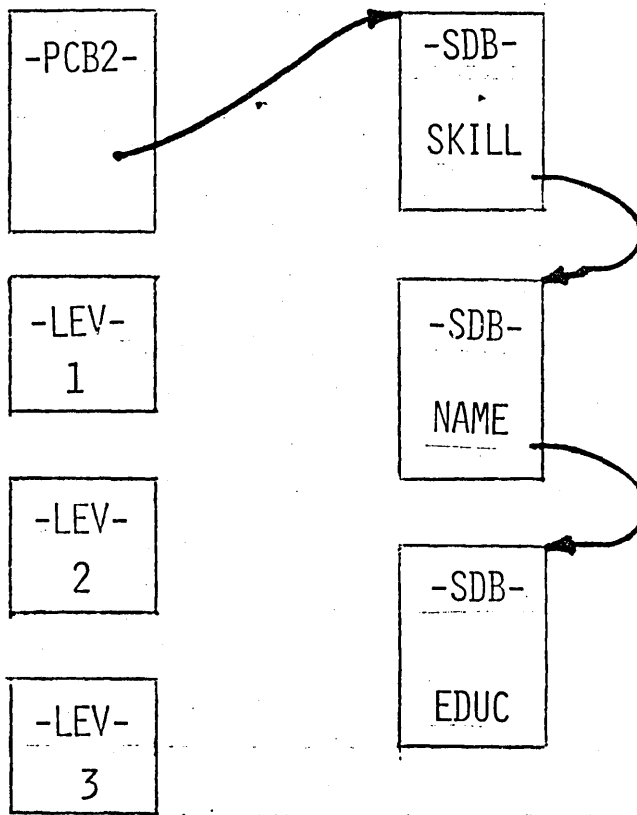
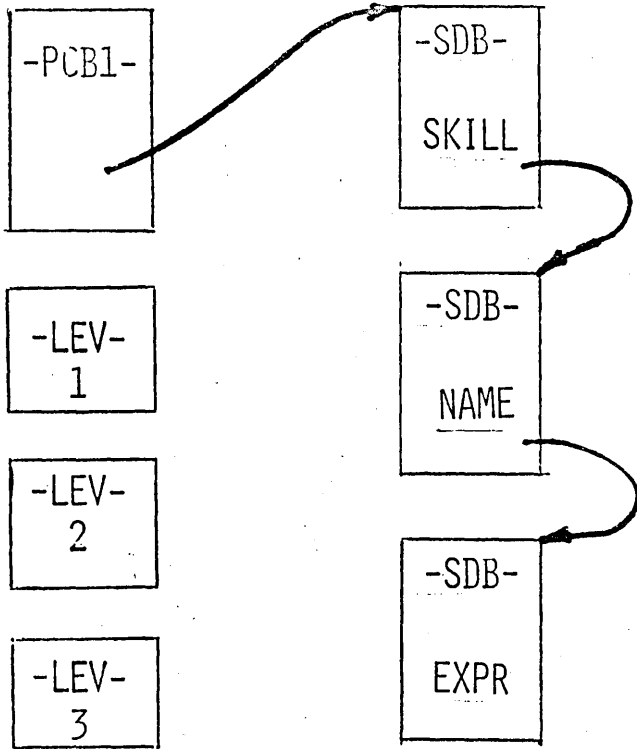
CALL: GN EDUC

PCB2

I/O: MICHIGAN --- 0664MBA

KEYFDBK: PRG JONES MICHIGAN

SDB - PSDB RELATIONSHIP



CALL: GN SKILL

PCB1

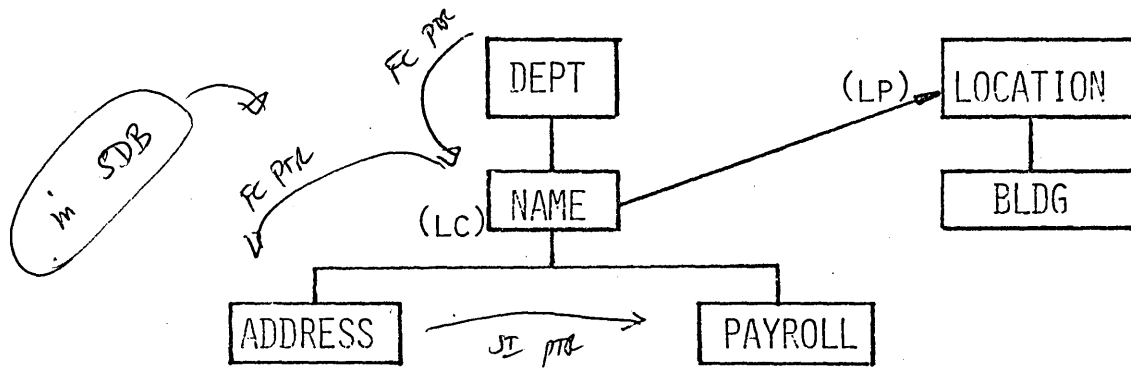
STATUS CODE = 'GE'

I/O:

KEYFDBK:

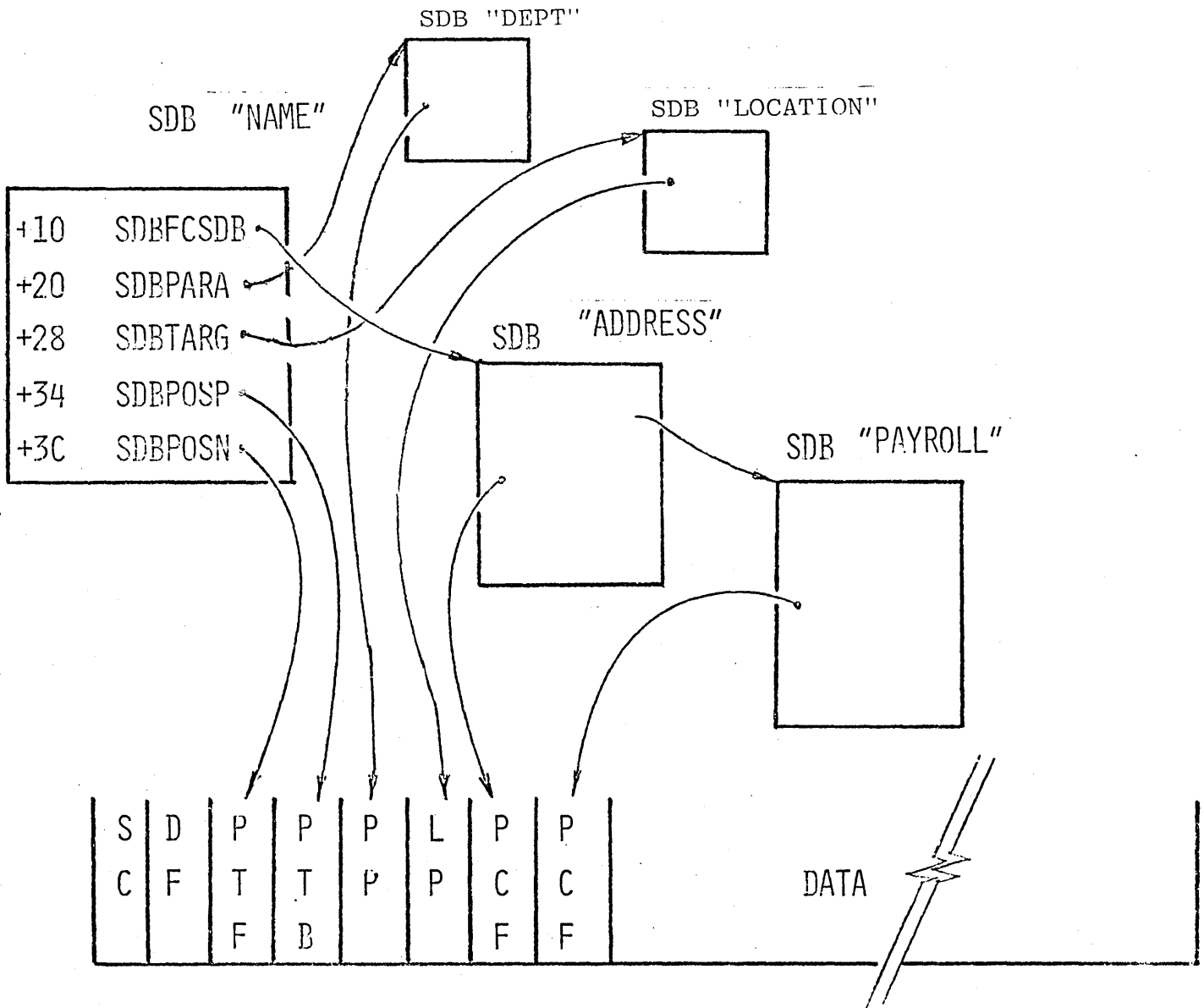
P R E F I X P O I N T E R S

● SAMPLE DATA BASE



SEGMENT PREFIX -

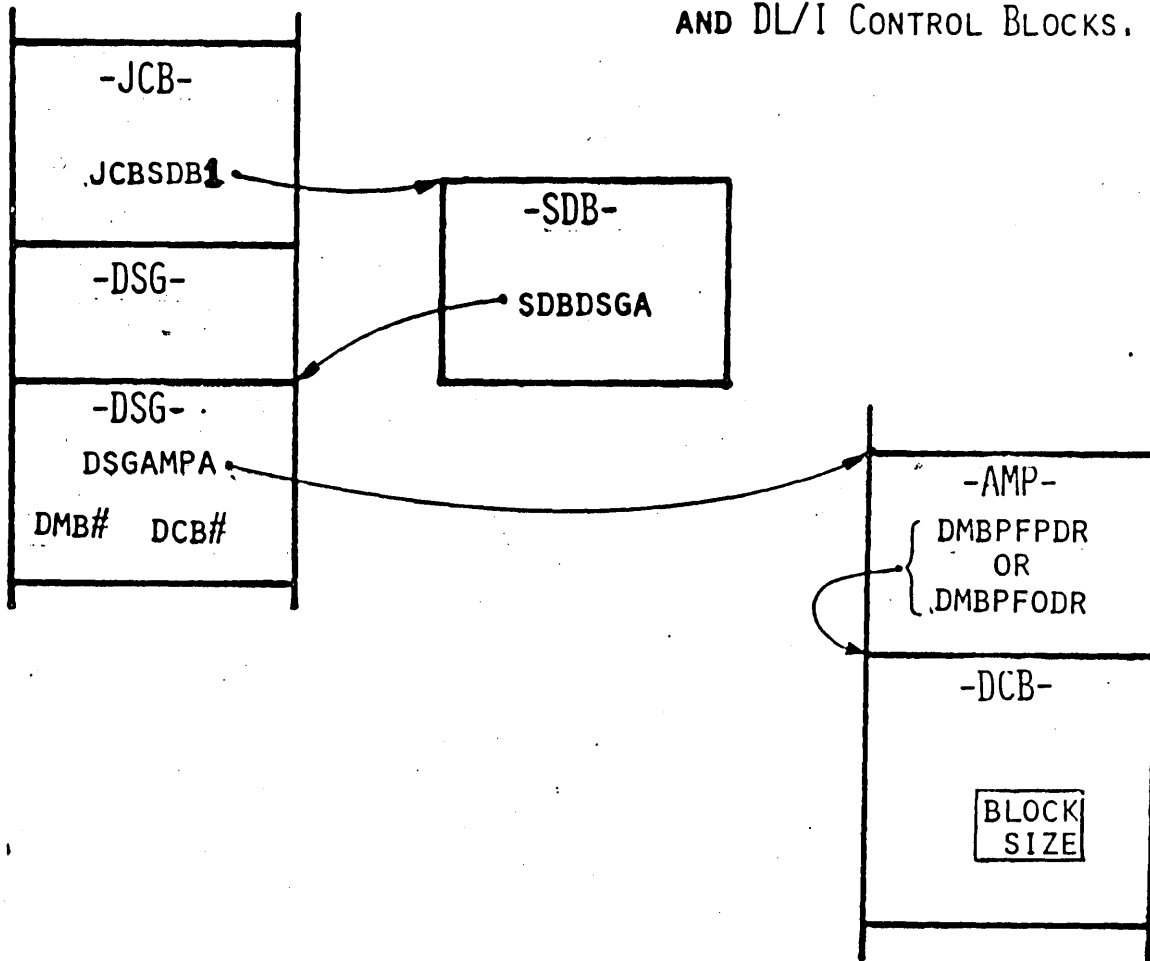
SEGMENT IN HD



LOCATING DATA IN BUFFERS

(ISAM/OSAM)

GIVEN: RELATIVE BYTE NUMBER
AND DL/I CONTROL BLOCKS.



IBM-STL-

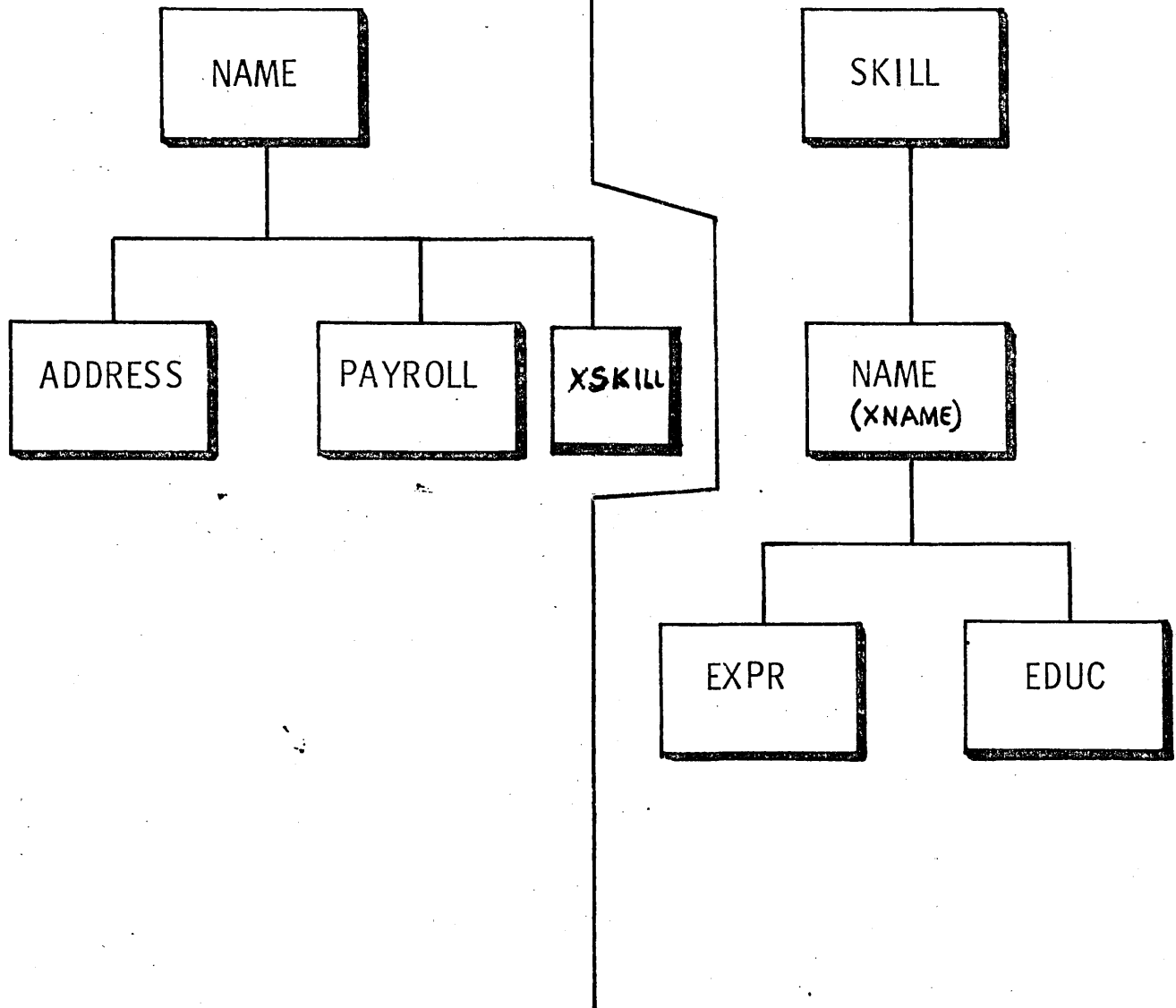
$$\frac{\text{RELATIVE BYTE NUMBER}}{\text{BLOCK SIZE}} = \text{NUMBER PLUS REMAINDER}$$

DMB#, DCB#, RELATIVE BLOCK#, OFFSET INTO BLOCK

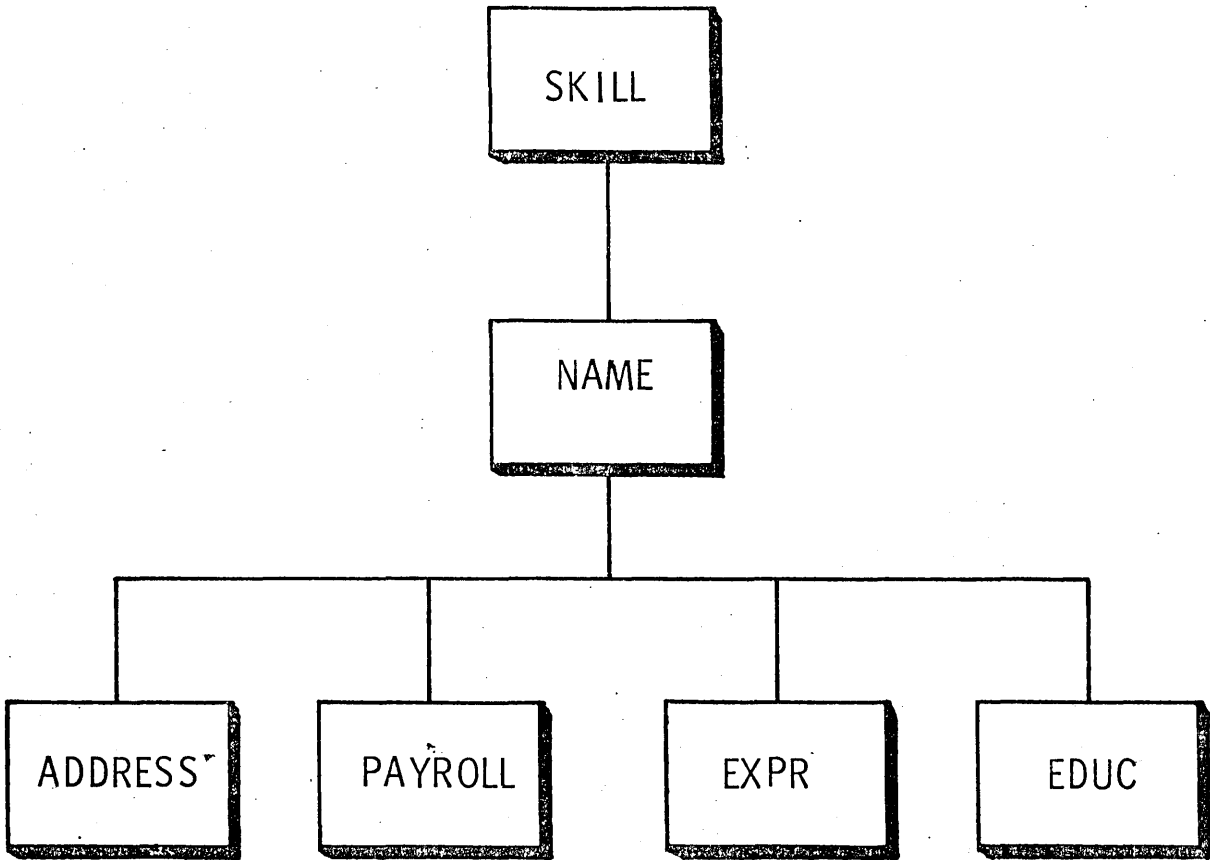
Two Physical Data Bases

PAYROLL DATA BASE

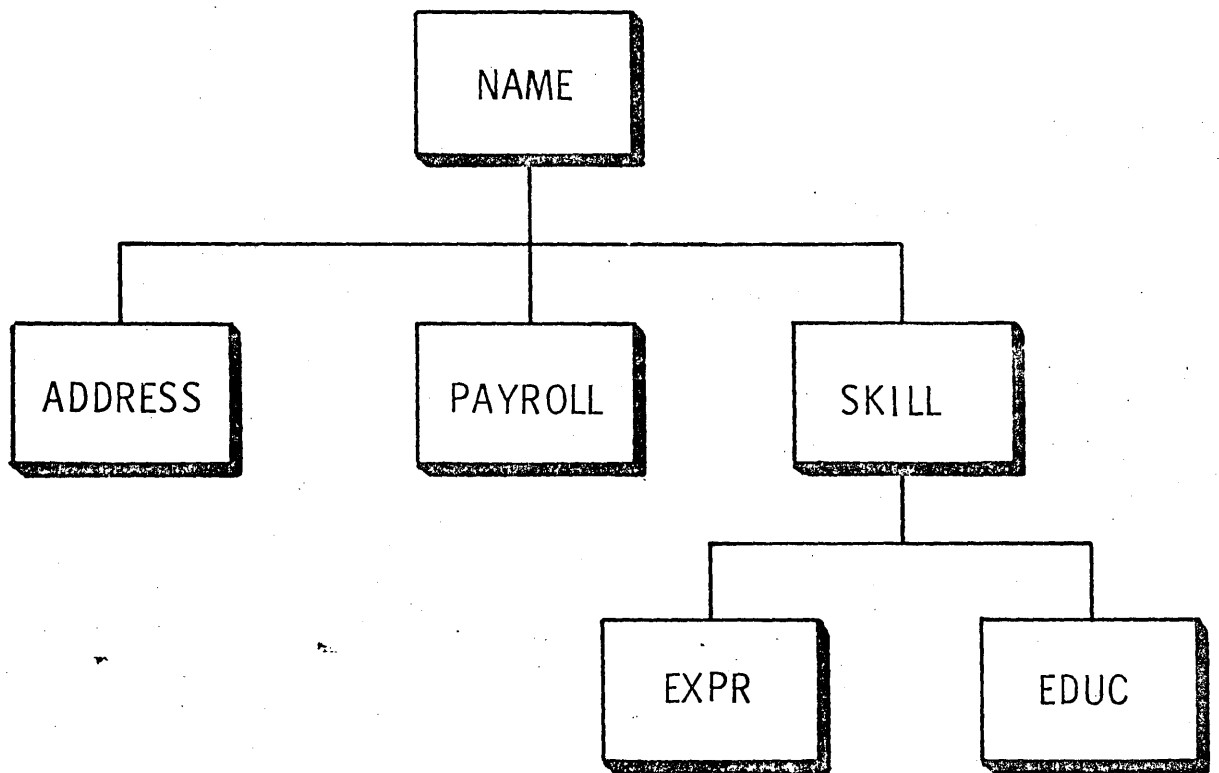
SKILLS INVENTORY DATA BASE



Logical Data Base 1



Logical Data Base 2



SCD EXERCISE

1. What is the address of the SCD? (PST+168) ' B9180 '
2. What are the user specified IMSGEN parameters for:
Type 1 SVC Number: E4 = 228
Type 2 SVC Number: E6 = 224
Type 4 SVC Number: D3 = 211
Channel End Appendage Suffix: E9F1 (= 21)
3. What is the IMS/VS Release & Modification Level number for this system? 1.1.3
4. What is the entry point address of the following modules:
DL/I Call Analyzer: ~~D5028~~ D1550
ISAM Simulator: ~~D5028~~ C6110
VSAM Interface: B9250
DL/I Retrieve: ~~D5028~~ DEF68
Load / Insert: ~~D5028~~ B9258
Dlet / Replace: D5028
ISAM/OSAM Buffer Handler: C6E60
Get/Free Space: CEB50
Index Maintenance: C3A08
5. What is the address of the PSB Directory? (SCD+E8)
B7000
6. What is the address of PXPARGS? (SCD+148) B5060
7. What is the address of the ISAM/OSAM Buffer Pool?
(SCD+2BC) E6000

ISAM-OSAM DUMP PROJECT NUMBER ONE

1. What is the current or last used DB PCB address?
 ' C5924 '

2. How many data base calls have been issued for this schedule of the program? 5

3. What are the contents of the current DBPCB?

DBDNAME: LOGNAME
 LEVEL: 0
 PROCOPT: AP
 STATUS CODE: GP
 NUMBER OF SENSEGS: 6
 KEY FEEDBACK: ADAMS. JOHN QUINCY (42 BYTES)

4. What is the location of the JCB associated with this PCB?
 ' C5944 '

5. What were the prior 5 calls and ^{thier} ~~thier~~ return codes?
0-0, GU-B, GN-B, GN-B, ~~GN-B~~ GN-B GN-9

6. Where do the SDB's and Level Tables start for this PCB?
 LEVEL TABLE START = C5BCC
 1st SDB ENTRY = B488C

7. What is the lowest level segment which can be accessed by this PCB? 3

8. What is the data base organization of the Root Segment?
HDAM, HIDAM, HSAM, HISAM, SHISAM

9. What sensitivity (ISRT, DLET, REPL, GET, etc.), does the user have for the Root segment? GIRDP (=AP)

10. What is the location of the DCB to be used for the Root segment? ' B4570 '

11. What is the RbN (if HD), or RRN & Offset (if HISAM), of the location of the Root segment? 27A4 Which is it, RbN or RRN? RbN

12. What is the location of the PSDB of the Root segment?
 ' B4418 '

13. What is the total segment length (Prefix and Data), of the root segment? (Hex or Decimal) 8E = 142

14. What pointers are present in the prefix of the Root segment?

CTR
 Phys Tw in Fwd
 => Phys Child Forward also

S	D	C	P	P	P	P
C	F	R	T	C	C	C
1	1	4	4	4	4	4

15. What are the ISRT, DLET & REPL Rules for the Root segment? ISRT: logical, not used, DLET: logical, REPL: logical.

16. What is the name of the sequence field (key) of the Root segment? FULNAM

17. Is this field unique or non-unique? Unique

18. Why does the instructor want to know all these things? Can't he find them himself?

ISAM/OSAM BUFFER POOL EXERCISE

-Use Dump Number 1-

Fill in the buffer pool map below for the buffers labeled A,B,C,... from Low to High storage. For each buffer fill in the Address, the "use chain" pointer addresses, the Buffer "ID" (Block#, DMB#, DCB#), and the buffer size in Hex. Mark the End of the Buffer Pool by drawing a line under the last buffer. Also mark each buffer as to being either Full or Empty.

STORAGE ADDRESS

	<u>E6000</u>	*****
		* 'BFPL'
		* Most: <u>EA208</u> Least: <u>EA8C8</u>

'A'	<u>E6070</u>	* Next Lower: <u>EA8C8</u> Higher: <u>E8708</u>
		* Size: <u>DB0</u> Block: <u>2</u> DMB: <u>3</u> DCB: <u>2</u>
		* Full or Empty? <u>F</u>

'B'	<u>E6E20</u>	* Next Lower: <u>E9B48</u> Higher: <u>EA208</u>
		* Size: <u>18E8</u> Block: <u>1</u> DMB: <u>4</u> DCB: <u>1</u>
		* Full or Empty? <u>F</u>

'C'	<u>E8708</u>	* Next Lower: <u>E6070</u> Higher: <u>E8DC8</u>
		* Size: <u>12C4</u> Block: <u>2</u> DMB: <u>2</u> DCB: <u>1</u>
		* Full or Empty? <u>E</u>

'D'	<u>E8DC8</u>	* Next Lower: <u>E8708</u> Higher: <u>E9488</u>
		* Size: <u>6C0</u> Block: <u>3</u> DMB: <u>2</u> DCB: <u>1</u>
		* Full or Empty? <u>F</u>

'E'	<u>E9488</u>	* Next Lower: <u>E8DC8</u> Higher: <u>E9B48</u>
		* Size: <u>6C0</u> Block: <u>4</u> DMB: <u>2</u> DCB: <u>1</u>
		* Full or Empty? <u>F</u>

'F'	<u>E9B48</u>	* Next Lower: <u>E9488</u> Higher: <u>E6E20</u>
		* Size: <u>6C0</u> Block: <u>5</u> DMB: <u>2</u> DCB: <u>1</u>
		* Full or Empty? <u>F</u>

'G'	<u>EA208</u>	* Next Lower: <u>E6E20</u> Higher: <u>φ</u>
		* Size: <u>6C0</u> Block: <u>6</u> DMB: <u>2</u> DCB: <u>1</u>
		* Full or Empty? <u>F</u>



GET (RETRIEVE) CALL FLOW

Formatted Dump (Dump Number 2)

Using the DL/I Test Program and a formatted SNAP after a GET call we will trace the flow for this call. The call was a GU to Logical Data Base 1 for NAME under SKILL. with unqualified SSA's.

NAME is a LC/LP concatenation of XANAME and NAME in the physical data bases. We begin when Call Analyzer (DLA0) get control.

I. The Users CALL LIST is moved to LIPARMS and converted to the Implicit format.

Function
PCB
I/O Area
(SSA's)

Validate and store the PCB address 'C57A4' in PSTDBPCB (+160) at 'B6168'. If the PCB address is invalid, a U476 abend is issued.

Encode the function to X'01' and store in JCBPRESF (+7F) at 'C58A3'. The address of JCB is in the DBPCB at +10. 'AD' status code is issued if this is an invalid function. Move user's I/O area address to the PSTUSER (+BC) at 'B60C4'.

II. Find the SDB for the name in the SSA. The first SDB is at JCBSDB1 (+8) at 'C582C'. Since this is for the SKILL segment we found the correct SDB at 'B488C'. an 'AC' status code is issued if there is no SDB for the segment name in the SSA.

Find the Level Table entry for this segment. Segment level is in the SDBLEV (+8), and the start and end of the Level Table is in the JCB at JCBLEVTB (+0) and JCBLEVND (+4). Level One entry is at 'C5A4C'. Store the address of the SDB at LEVNUSDB (+20) for level one.

III. If there is an SSA (in our case there is), is the segment name followed by a blank, "(" or *. If it is a "(" store the address of this left paren at LEVSSA (+1C). LEVSSA is at 'C5A68'.

In our case there is no "field level qualification" in the SSA, but if there was the FIELD name must be validated.

SDBPSDB (+1C) points to the Physical Segment Description in the DMB. Within the DMB there is a PSDB for each physical segment. DMBFDBA (+10) points to the FDB's

within this segment. The first FDB for the SKILL segment is at '64388'. Turn to the Field Description Block in the DSECTS and look at the FDBDCENF (+A) flag. If a qualified SSA attempted to search on a field and no FDB was found an 'AK' status code would be returned at this time.

Looking at the DL/I call format, this is where we are:
 FUNCT SEG (FIELD OP VALUE)

The next thing to check is the OPERATOR. It must be encoded and checked against the valid ones. An invalid OPERATOR gives us an AK status code.

The method used to check for the field value length is to take the length from the FDBFLENG (FDB+'B'), add that to the starting location of the value and compare for either a right parenth or boolean operator. The field SKCLASS in segm SKILL is how long? The FDBFLENG is at '64302' and contains '7', so at plus '8' into the value there would be either a ')' or a boolean operator.

- IV. If there is another SSA we repeat step III. When all SSA's are validated and decoded, CALL ANALYZER next checks the Processing Options for any "violation". This check takes on greater importance on an ISRT, REPL or DLET! The test for these calls is made against field SDBF3 (SDB+'A') which is at '84896' and is 'F4'. Checking in your DSECTS this PROCOPT is:

AP

Call Analyzer is finished and depending on the function goes to the proper action module. (ie: Retrieve, Load/Isrt, Dlet/Repl)

- V. Analyzer calls RETRIEVE (DFSDLR00) since our function was 'GU'. Retrieve initializes base regs for DBPCB, JCB, SDB (root), LEV TAB and its Work Areas. (R1 was set to the PST address by DLA0) The first check is for a Qualified or Unqualified call (is there a Seg name?), ours is Qualified.

Next check is whether or not existing position can be used. Check the LEVF1 (LEV +'1') for "LEVEEMPTY"(Bit 1) on each level in the call. At this point in the processing the LEVF1s were: LEV 01 = '40'; LEV 02 = '40'; LEV 03= '41'. Therefore the 2 level(s) in our call do NOT have existing (Previous) position.

Since LEV 01 position was not "usable", DLRO clears the top half of all level tables below the root. This means

the ones at 'CSA74' and 'CSA9C' in our current DEPCB.

The organization of this DB must be established. SDBORGN (SDB+'9') for the root is at 'B4895' and is '04'. This means it is in 'HISAM'.

Follow the SDBFCSDB (SDB+'10') pointer to the NAME SDB and since this is the "end" of the call, set on SDBEOC (Bit '0' in SDBF6 @ SDB+'D'). This "Retrieve Switch" is going to be set again, so at the end of the call SDBF6 for NAME at 'B48E2' is '60'. Prior to the value you see in the dump it was '80' for the SDBEOC flag.

VI. Retrieve is going to the buffer handler next so it puts an '08' in Reg 1, an '04' in the DSGINDA (DSG+'7'), and calls the I/O Interface (part of DLR00 called SETL). SETL uses this info to call the Buffer Handler with a value of 'F0' in PSTFNCTN (PST+'16C'), and the address of the PST in Reg 1. A value of 'F0' is called PST STLBG. This means "get the first LRECL by KEY in HISAM", in our case, the first SKILL. (PST p12)

VII. Buffer Handler Router (DFSDVBH0), calls ISAM/OSAM Buffer Handler (DFSDBH00), and decodes the PSTSTLEG to mean "locate the LRECL with the lowest key in the data base." DBH00 calls ISAM SIMULATOR (DFSISM00), which uses SETL to locate the first ISAM LRECL in either BISAM or OISAM. This is based on the DCBMACRF field in the "prime" DCB for this DSG. The SDB (for SKILL) +'24' points to the DSG at 'C59D4' and the DSG +'0' points to the AMP at 'B42D4'. The AMP +'18' points to the ISAM DCB at 'B4474'.

If the ISAM LRECL has an overflow pointer (P1), get this LRECL via OSAM. Wherever the lowest key is found (ISAM or OSAM), the block is read into the Buffer Pool. The fields filled in now are:

PSTRTCDE (+ '16D')
 PSTOFFST (+ '16E')
 PSTDMBNN (+ '178')
 ESTDCBNN (+ '17A')
 PSTBYTNN (+ '17C')
 PSTDATA (+ '180')

All of this is traced by DFSDVBH0 into the DSECT 'BFSPTRAC', and control returns to Retrieve.

VIII. Retrieve moves PSTBYTNN (+ '17C') into SDBPOSC (+ '38') and PSTOFFST (+ '16E') into SDBPOSN (+ '3C') for "current position" and turns on Bit 1 in SDBF6 (+ 'D') showing the

SDB "posted" for SKILL. Retrieve next sets Bit 4 on in all dependent SDB's to show "not posted". The level table for this level (01), is plugged into JCBLEV1C (+20). LEVTTR (+4) and LEVSEGOF (+8) are plugged with the SDBPOSC and SSBPOSN data to "save" position info.

The key of this segment (in this case "ANALYST"), is plugged into the PCB Key Feedback area (DBPCB+24) at 'CS7C8'.

- IX. The next SSA in our call is for "any" NAME under the SKILL just located. This boils down to the "first" NAME under the established SKILL of "ANALYST". The SDB for NAME is located by following the SDBFCSD (SDB+10) in the SKILL segm. The NAME SDB is at 'B48D4'.

Back in step V. retrieve reset (cleared) the top half of the level tables of all dependents of SKILL. Therefore level 02 cannot have "position" to use.

The subroutine that is used here is called "Forward this Level" and its purpose is to get a segment at this level by a "forward" search in the data base.

Our SSA for level 02 (NAME segm), is unqualified and any NAME segment at this level can be searched. This is true because the LEVF2 flag (LEV+2), at 'CSA76' is '08', which is called LEVCONT. (Search may continue at this level)

Now if retrieve gets the next NAME under the SKILL of "ANALYST" we are in fat-city! Once this is done SDBF6 (+D) has SDBCHEOC set on (Bit 3) to show this segm is the "child" end-of-chain. This segm is in HISAM and is involved with logical relationships so we must check the Delete Flag to see if this segm is available on this "path". The DF is '00' so all is well.

The one we got was "JONES, JOHN PAUL" and it's not deleted so retrieve now posts its position in the SDB at SDBPOSC (+38) 'B490C' (RRN) and SDBPOSN (+3C) 'B4910' (Offset). The RRN is '1' and Offset is '3' in the OSAM (overflow) data set.

SDBF3 flag (+A) for this LChild has SDBSENK on (Bit 4), so this segm will not be returned to the user even though this is the lowest level in the call.

- X. Check the SDBTFLG (+28) for NAME segm at 'B48FC'. This value ('01'), means the segment we have retrieved points to a LParent and the LParent must be retrieved. Does this segm (the LChild), have a Direct Pointer to

the LP? Check the DMBPTR (PSDB+'7') at 'B4953' which is '01'. This means there is no LP pointer in the segment prefix. In fact this means the entire prefix is 2 bytes long so the Symbolic Pointer must be used.

The SDB for our LChild segm is connected to its LParent via the SDBTARG ('+28') which points to a Generated (or "out-of-line") SDB at 'B463C'. It is easy to check this SDB against an "in-line" one since the first 8 bytes are zeroes. Check the SDBTFLG ('+28') for this SDB at 'B4664'. (It tells 4 things about this SDB.) This SDB points to the DSG (SDBDSGA '+24') at 'C5A24' and the PSDB (SDBPSDB '+1C') at 'C5DE8'.

Checking DSGINDA ('+7'), for the LParent at 'C5A2B' we learn that this DSG is for a HIDAM data base.

This out-of-line SDB points to another SDB via the SDBTARG ('+28') and this SDB at 'B46CC' has an SDBORGN ('+9') of '44' meaning it is for the Index data base.

Retrieve now has the SDB, DSG and the KEY of the "NAME" segm (LParent) we want, so it sets R1 = '16' and goes to its I/O Interface (SETL).

The call to Buffer Handler is prepared by plugging a 'F2' in PSTFNCTN ('+16C'), putting the address of the "Key" in PSTBYTMM ('+17C'), and going after the HIDAM Index.

- XI. Buffer Handler searches the ISAM index data base to locate our Key and returns with the RRN of the INDEX segment in PSTBYTMM ('+17C') which retrieve plugs into SDBPOSC ('+38') at 'B4704'. This RRN is 'A' and SDBPOSN has the OFFSET from PSTOFFST which is '3'. In the prefix portion of this index segment is the RbN of the LParent segment we want.

This RbN is next used by retrieve by moving it to PSTBYTMM ('+17C') and making another call to Buffer Handler with a PSTFNCTN of 'E2' (Byte Locate). This time we are going after the HIDAM DATA data base.

To locate a segment in the HIDAM data base we must "back up" the SDB's from the INDEX one to the HIDAM one by taking the SDBPARA ('+20) in the INDEX SDB's. at 'B46EC' which contains 'B463C'. Both of these are out-of-line SDB's.

XII. Buffer Handler must do a convert of the RbN to Rel BLOCK Number and Offset this time, but if all is well it finds the segment. Bufr. Handler returns to retrieve with PSTDATA (+180) pointing to the segment in the buffer pool. OSAM was used to read in the block that contained this segment if the block was not in the buffer pool.

XIII. Retrieve now has the LParent segment and does a post of the position. The SDBPOSC and SDBPOSN values at 'B4674' and 'B4678' are '~~1D76~~' and '~~1C4E~~'. The LParent segment just retrieved starts at 'B9EC6'. Check these values against the PSTBYTMM and PSTDATA fields in the dump.

The key of this segment is found by taking the SDBPSDB (+1C) pointer to the PSDB at 'C5DE8'. At '10' into the PSDB is the pointer to the first FDB which is at 'C5E78'. Check each FDB at FDBCENF (+A) for Bit 1 on. The FDB named FULNAM is the sequence field for the NAME segment and it is '420' bytes long and starts in col. '7'. Check this length against SDBKEYLN (+20) in the dump.

This key field must be extracted from the segment just retrieved and plugged into the DBPCBKFD, but where? At this time Retrieve is working on the "out-of-line" SDB for the LParent at 'B463c'. In this SDB SDBPARA (+20) at 'B465c' has 'B48D4' which is the address of the SDB for the LChild. (NOTE: The "in-line" SDB's point to "out-of-line" SDB's via thier SDBTARG field, and the first "out-of-line" SDB points "back" via its SDBPARA field.) The SDBPARA (+20) for this segment at 'B48F4' has 'B488c' which points to its parent. By following these two SDBPARA's we end up at the root segment. Taking the SDBKEYLN (+20) at 'B48Ac' for SKILL which is '7' we have the offset within DBPCBKFD for this KEY we have retrieved.

Taking the SDBFCSDB (+10) in the SKILL SDB we come back to the SDB for NAME at 'B48D4'. In this SDB at SDBKEYFD (+30) at 'B4904' is where retrieve plugs the address of the key offset. This address is now used to move the Key of the segment just retrieved into the DBPCBKFD.

XIV. This segment is the lowest level in the call and the "call sensitivity" at SDBF3 (+A) is 'F4', so this segment will be returned to the user. The length of the prefix in DMBPRSZ (PSDB+8) at 'C5DF0' is '220' and the data length in DMBDL (PSDB+A) at 'C5DFV' is '120'. This data length is plugged into PSTSEGL

(+'B8') and the prefix length is added to PSTDATA (+'180') and placed in PSTSEG (+'B4') all by retrieve.

The address of the Level Table for this segm. is "posted" in the JCBLEV1C (+'20') at ' CS844 ' and Retrieve returns control to Call Analyzer.

- XV. Analyzer plugs the Key Feedback length into DBPCBLKY (+'1C') and returns control to Program Request Handler.

Program Request Handler moves the segment based on the address in PSTSEG (+'B4') to the address in PSTUSER (+'BC') using the length in PSTSEGL (+'B8') and returns control to the application program.

UPDATED: 12/20/76

INSERT CALL FLOW

- I. Our task is to ISRT a new 'SKILL' segment (in Logical D.B. 2), which is a Logical Child 'XSKILL' in the NAME data base (ISRT via the Physical Path) and a Logical Parent 'SKILL' in the SKILL data base (ISRT via the Logical Path).

The first call in the dump is a qualified 'GU' to the NAME segment of BRUIN, FRED J. This call establishes "position" in the data base so that the ISRT call that follows places the SKILL of ARTIST under that particular NAME. The GU call is snapped so that we can look at some of the control blocks & buffers "before" the ISRT call.

At this time turn forward to the ISRT call (about 10 pages in the dump), and you can see the exact call as issued by DL/I Test.

- II. Flow is the same as for a GET Call up through the Call Analyzer. The call is decoded to be an Insert, and the PROCOPT is checked to be 'I' by checking the SDB for the segment being inserted. In this instance we must check the SENSITIVITY of the "SKILL" SDB. Scanning the interpreted side of the dump on page 129 we find the value "SKILL". This is the the SDB which starts at 'B4964'. The SDBF3 ('A') at 'B496E' has '7C'. No matter what other bits are on, if Bit 1 ('40') is ON this segment is OK to insert with this PCB. Once we have passed this test, Analyzer gives control to DFSDLR00 - DL/I RETRIEVE.

- III. Retrieve establishes position in the HIDAM data base for the ISRT of the XSKILL segment (the Logical Child). 'Position' in this case refers to the Hierarchical location within this data base record where this particular segment must go. In order to do this (and follow the Retrieve flow), locate the PST at 'B6008', go +160 to '~~B6008~~' (the DBPCB address) which is 'C5924', DBPCB + 10 to the pointer to the JCB which is at 'C59A4' and plus 8 in the JCB to 'C59AC'. This location contains 'B488C' which is the address of the first SDB in this PCB. The first 8 bytes of that SDB are 'NAME B666' which is the name of the Root segment. This segment has position which was established by the first 'GU' call and by looking at SDBPOSC (+38) at 'B48C4' it is '12C4'.

At SDBFCSDB (+10), at 'B489C' there is '48' (Half-Word) which is the "Offset" to the first Child SDB for this segment. At that offset is segment name: ADDRESS. At SDBSISDB (+12) in this SDB

' 648E ' is ' 48 '. That points to the first Sibling SDB for this segment which is named: PAF04. The SDBSISDB in this segment at ' 64920 ' takes us to the "SKILL" SDB at ' 64964 '.

Retrieve uses the "current" position of the "NAME" segment and by reading the prefix pointer in that segment for the Physical Child "SKILL" gets the first segment to check for position. The key field of this "SKILL" segment is read and compared to the Key of the segment in the user's I/O area. In our example the first "SKILL" in the data base record for BRUIN, FRED J. is "BW173" (Photomurals). The Key fields are compared by retrieve and ARTIST comes before BW173. The "position" (RbN) of BW173 is placed in SDBPOSN of the "SKILL" SDB at ' 649A0 '. This RbN is ' 138c '. Since there was no SKILL lower than the one being inserted Retrieve plugs zeroes into SDBPOSP at ' 64998 '. The SDBPOSC is not filled in at this time. (The value you see there is your dump was placed there later in this call flow.) Remember that this is not the actual "Physical" location on DASD where the new 'XSKILL' segment will be inserted. Once this 'position' is found control passes back to Call Analyzer which next passes control to Load/Insert (DFSDDLE0).

IV. Our segment to be inserted (SKILL), is a LC/LP concatenation. The SDBTFLG field (SDB +28) ' 6498c ' has a value of ' 01 ' which means that this segment 'Points To A Logical Parent'. Therefore Load/Insert must check for the Lparent. In our case the LP is in the data base, (SKILL of "ARTIST "). The Insert Rules are found by starting at the SDB and following the SDBPSDB (+1C) pointers to the PSDB for each segment. First is XSKILL and the field (DMBISRT) at PSDB +1C is ' 64990 ' which is ' 23 '. To get the LParent PSDB go back to the SDB (LChild) and take the SDBTARG (+28' actually +29') to the out-of-line SDB at ' 6463c '. The SDB points to its PSDB which is at ' 650F8 '. Now go plus 'C' (DMBISRT) to ' 65D04 ' for the ISRT rules field which is ' 23 '. The Insert rule of Logical means that if the LParent has been inserted in the data base previously only the LChild portion of the concatenated segment will be inserted. However if the LParent did not exist both the LChild and LParent will be inserted into their physical data bases.

Next check the file organization of the LChild at SDBORGN (+9 ' 6496D '). Ours is ' 10 ' and that means ~~###~~ . H(9)AM

Is this segment an Index SOURCE segment? (Secondary Indexing) Check PSDB +20 (DMBFLAG) ' 644A4 ' which has:

'48'. A value of '10' means this segment is an INDEX SOURCE; '04' means it is INDEX TARGET. This means there is no call to Index Maintenance for this ISRT.

Check this segment for Fixed or Variable Length at DNEVLDFG (+18 'B449c') in the PSDB. Ours is: '00' which means FIXED LENGTH.

- V. Based on the 'position' given to Load/Insert by Retrieve now locates space for the "new" segment, "XNAME" by placing an '01' in PSTFNCTN (+'16C'), (this is called "get space"), and also putting the RbN of the parent segment "NAME" in PSTBYTNM (+'17C') and calling HD Space Management. This RbN of the parent segment came from Retrieve when it established position for Load/Insert.

This is the criteria for a "get space" as used by HD Space Mgmt:

Space in the same BLOCK; In a Block on the same DASD TRACK; In a Block on the same CYLINDER; In a Block within the SCAN limits (Delta Cylinders); and if all else fails, the end of the Data Set. HD Space Management makes calls to the Buffer Handler to find the "Most Desirable Block" in the data set for this "new" segment. In order to follow the activity of HD Space Mgmt. a Trace Table is provided. In your PLM Vol 3 of 3 on Page 6.109 is a diagram showing the location of the HDTR. In the dump the PST+'168' is at 'B6170' and the SCD is at 'B9180' and the HDTR (SCD+'160') is 'Bc340'. '+'160' is correct for Rel 1.1.3 even if your PLM says '164'. That value was correct for Rel 1.1.2!! (The DSECT description of HDTRX is also in your PLM Vol 3 of 3 on Page 5.150) The HDTR+'4' points to the "current" entry in the HD Sp Mgmt trace at 'Bc350'. Turn to Page 6.110 in the PLM (3 of 3), for a "map" of the dsects HDTRACE & HDTRX. The "current" entry has a function code at 'Bc350' of '01' which is a "Get Space" request. At HDTRX+'10' at 'Bc360' is '12c4' which is the RbN of the "parent" segment which is the block where Load/Insert wants the "get space" search to start. In this case the parent is the NAME segment for Fred Bruin and the RbN was supplied to Load-Isrt by Retrieve. This RbN is passed to the Buffer Handler by HD Sp Mgmt in order to see if space is available in that block for the "new" segment. If there is no space the criteria mentioned above is used to find space in the data set.

In the HDTRX at '+'14' is the RbN that HD Sp Mgmt passed back to Load-Isrt for the XSKILL segment to be inserted. This is the value that Space Mgmt placed in PSTBYTNM(+ '17C') when it went back to

Load-Isrt. At HDTRX+'C' : BC35C ' is ' EA4E8 ' which is the address (in the buffer pool), where this RbN actually exists at this time. This address is returned to Load-Insert by Space Mgmt. in PSTDATA ('+180'). At HDTRX+'8' : BC358 ' is ' 00 ' which is the return code passed back to Load-Insert in PSTRTCODE ('+16D').

Remember that this dump was taken after the entire call completed and the PST fields just mentioned have been changed several times!!

VI. Space Management Returns to Load/Insert -

A. With the following info:

- PSTBYTNM (+17C ' B6184 ') - RbN location of where to put the new segment being added. (Dual purpose field coming-and-going!)
- PSTDATA (+180 ' B6188 ') - Address where segment data for the New LChild is located.
- PSTRTCODE (+16D ' B6175 ') - Return code (if '03' the bit map must be updated).

- B. The next operation is to build the complete segment, both Prefix and Data portion and move it to the buffer pool. The Data portion of the segment comes from the Users I/O Area PSTUSER (+BC ' B6004 '), which at "this" time is 'C2F10' and not the address you see in the dump. That address has a pointer which has been updated to the LParent which comes later in the call flow. The Prefix is "carved" out with a segment code taken from the PSDB for the SKILL segment. That PSDB is at ' B4484 ' and the code is ' 04 '. The delete flag is plugged with a 00 byte and then the number of blank prefix-pointers is established by reading the DMBPTR (PSDB+'7') at ' B448B ' which is ' 50 ' and the DMBPRSZ (PSDB+'8') at ' B448E ' which is ' A '. The DMBPRSZ tells the size of the prefix, and the DMBPTR tells what pointers are present. The values there mean that we have Physical min physical parent pointers and the prefix is 10 (= A₁₆) bytes long. The values moved into these prefix pointers come from two locations. The Physical Twin Forward pointer comes from the SDBPOSN field at ' B49A0 ' which is ' 13BC '. The Physical Parent pointer comes from the parent and we follow the SDBPARA ('+20') at ' B4984 ' to the "NAME" SDB and go to SDBPOSC at ' B488C ' and get ' 12C4 ' and plug that into the PP pointer. This completes the building of the segment. Load-Insert updates the SDB current position holder located at ' B499C ' which has ' 2A1C '. SDBPOSP and SDBPOSN were filled in by Retrieve. Load-Insert now moves the new segment to the address in the buffer pool. Map out the segment here:

04	0	13BC	12C4	ARTISTKS
(Sc)	DF	PT pr	PP pr	(Dnm)
1	1	4	4	8

If logging is being used (in this case I used a DD DUMMY for the log), the NEW segment inserted in the data base will be logged out now.

If the return code from Space Management, PSTRTCODE (+16D '06175') was an '03', a second call to DFSDHDSO (Space Management) is made to update the Bitmap. The value in the dump at HDTRX+'8' at '6C35B' is '00' so no call to HD Space Mgmt. with a Function Code of '03' is made in this dump, but if an update was to be made this is the time!

Since the Bitmap is itself a block in the data base, this change must be logged out.

The Hier-Holder (HH) is updated next. This shows any subsequent user that this new segment is really there!

VII. Our XSKILL segment is physically paired so DL/I must ISRT the paired segment (XNAME) over in the SKILL data base. We discovered this Physical Pairing when we looked at the PSDB+'20' (DMBFLAG) for the LChild segm.

Re-positioning ourselves at the "SKILL" SDB at '84964' we find SDBTFLG (+28) is '01' and SDBTARG is '8463C'. The Gen'd SDB pointed to at this address has at SDBORGN (+9) at '84645' an '04' meaning it is in HISAM. SDBTFLG (+28) at '84664' is '3A' which means this SDB is Generated and points to a LChild (ie: XSKILL). SDBTARG is '846CC' which points to another Gen'd SDB. In this SDB at SDBTFLG '846F4' is '10' which means this SDB for Physical Pairing. The SDBPSDB (+1C) at '846FB' has 'C5DIC' which is the PSDB for the "Paired" LChild. Now that we have located the SDB and PSDB for the "Paired" LChild we can continue with the ISRT!

Segment XNAME, the paired segment, is a HISAM Dependent so a different insert path is taken:

A. DFSDDLE0 first locates the point in the HISAM LRECL containing the "previous" and "next" segments in sequence so that we may insert the new segment. Next it scans the LRECL to see if the new segment will fit. If not it must get a new LRECL to hold the new segment. The change to the old LRECL is to place a (P2) Pointer to the new LRECL and move the 'old' segm. in the added LRECL after the 'new' segment. If there is room in the existing LRECL all that is needed is to

'shift' the old segments and write the new one in. (All this is being done in the Buffer!) If there is no room in the 'new' LRECL for the segment(s) pushed off another new LRECL is allocated and the 'pushed-off' segments are moved there (from the original LRECL).

In order to accomplish all this "location", Load-Isrt must read the LParent segment (in ISAM), and follow the data-base-record looking for: the first "XNAME" (and do a Key comparison), or the end of Data-Base-Record. The segments are identified by segment codes and we find the SC for XNAME by starting back at the Gen'd SDB for this Physical Pairing at 'B46CC'. The SDBPSDB (+1C) at 'B46E8' has 'C5D1C' and that PSDB has a SC of '02'. So, the "scan" will start at the root and look for a SC of '02', '>02', or end with a P4 pointer (4 bytes of zeroes).

In this example there are no dependents of the SKILL "ARTIST" so the P4 pointer is reached first. There is no room in the LRECL in ISAM for any dependents so the "position" for our new "Paired" LChild is in a new LRECL in OSAM.

B. This is a good time to map out the SKILL segment in the Buffer Pool, so here we go! To locate a HISAM root in the buffer pool we must know the DMB#, DCB#, and BLOCK#. The first two are found in the DSG, so it's back to the SDB for the LParent at 'B463C' again!! The SDBDSG (+24) at 'B4660' has 'C5BA4'. The DSGDMBNO (+4 Half Word) is '3' and the DSGDCBNO (+6 One Byte) is '01', and while we're here the DSGAMPA (+0 Full Word) is 'C5CA0'. When a HISAM root is retrieved the "KEY" is given to ISAM and the BLOCK is returned to us in the buffer pool. I will play ISAM and tell you the BLOCK is #1. Also in the DSG at DSGBOFF (+C) '79' is the offset within the Block for the segment. Now you have the BLOCK#, DMB# and DCB# and we are ready to jump into the Buffer Pool. (Ouch!)

DMB# = 3
DCB# = 1
BLOCK# = 1
OFFSET = 79

C. Locate the SCD at 'B9180' and look at SCDDBFPL (+2BC) at 'B943C' which is 'E6000' and go to that address. Now we are at the 'BFPL', or the Buffer Pool Prefix as is says in the DSECTS. Searching the pool for a segment is done via the "Most recently used buffer" at BFPLFWDT (+8) at 'E6008' which is 'E8708'. Now look at the dsect "BFFRDS" for an individual buffer and compare: The BLOCK at +C 'E87042' with what we are looking for ('00000001'); The DMB at +10 '2' with our DMB#; The DCB at +12 with our

Block# 2

DCB. This is NOT the buffer we want, so take the BFFRFD (+'18') at 'E8720' which is 'E8DC8' and get the next buffer. Do the same comparison again for BLOCK# ('C') at 'E8DD4', DMB# ('10') at 'E8008', and DCB# ('12') at 'E9DDA'. This time we have an equal compare on all items and this is the buffer we want. According to the (sects) the Buffer Prefix is '20' bytes long, so if we add the offset (from the DSG) to the start of data we arrive at the ISAM LRECL at 'E8E61'. The first thing here is a P1 Pointer of '000000' showing there are no roots chained off this one. Next at 'E8E04' is the SC of '01', delete flag of '00', a pointer of some kind at 'E8E06' with a value of '00000001' and the start of data at 'E8E6A'. To determine what kind of pointer that is we must go back to the PSDB for the SKILL segment at 'C5CF8'. DMBPTR ('7') at 'C5CFE' has '81' which means this segment has a Counter. That counter in the prefix has a value of '1' meaning it has that many Logical Children without direct pointers to them. Before leaving the PSDB go to DMBDL ('A') at 'C5D07' which is '6A' and get the data length.

Block# 1
DMB# = 3
JCB# = 1

flag e 13₁₆ = 00

D. Now go to the buffer pool and add the data length to the start of the data and there should be the segment code of the next sequential segm. in this data base record. This is at 'E8EC4' and it is one byte of '00'. A zero segment code means this is the start of a P2 pointer and the next three bytes contain the RRN of the LRECL in OSAM where the next segment is located. (If these three bytes were zeroes this would be a P4 pointer and the data base record is ended. In this example there WAS a P4 pointer there before the ISRT call, but now there is a three-byte P2 with '33' meaning the next sequential segment in this data-base-record is in OSAM at the '33'-rd LRECL from the start of the data set.

This segment is the "Paired" one Load-Insert had to put in the data base since the User put in the XSKILL. Let's go back to the Gen'd SDB for this "paired" segment at 'B46CC' and pick up the SDBPSDB ('1C') at 'A46E8' and the SDBDSGA ('24') at 'B46F0'. The DSG is the same one as for the Root (SKILL) but the DCB will be for "overflow" instead of "prime". The PSDB at 'C5D1C' is level '02' and has a prefix size DMBPRSZ ('8') of '7' and a data length DMBDL ('A') of '2A'. The DSG at 'C5B44' has the DMB# at '+'4' which is '3' and the DCB since this is OSAM is '02' this time. We know the RRN is '33' but in order to know which Block it's in we go to the AMP via the DSGAMPA (+0) at 'C5B44'. DMBPFODR ('1C') at 'C5C8C' points

DCB# = 02
DMB = 3
Block = 2

to the OSAM DCB at 'CSF44'. The OSAM Block Size is at '+18' in the DCB at 'CSF50' which is 'D89' and the LRECL size is '+52' at 'CSF96' which is '69'.

E. Doing some fancy math we find the RRN we want is in the second Block for the DMB and DCB numbers already established. Going back to the Buffer Pool, follow the "use chain" looking for our Block, DMB & DCB match. We left off at the buffer at 'E8DC8' so take the BFFRFD (+18) at 'E8DE0' and check the next buffer at 'E6070'. It's Block (+C) '✓' DMB (+10) '3' and DCB (+12) '✓'. Right you are, and now add the prefix size to the start of the buffer. That is the start of Data in this Block. The "XNAME" we are looking for is in the LRECL at 'B6789', where there is a P3 pointer with all zeroes. Three bytes into the LRECL is the segment code of '07', delete byte next is '00', and if you wrote it down the prefix length (in the PSDB), was '02' so that's all! The data is next for '2A' bytes and the next byte is supposed to be a segment code. The SC there at 'B67B8' is '00' meaning this is a Pointer. The next three bytes are all zeroes so this is a P4 pointer and the end of the data base record has been reached.

F. This is the segment inserted by DL/I when the user inserted the other "half" of the physical pair. All changes must be logged and this is done now.

The SDB Position Holders and the HH's are updated to show any changes.

Is this Segment a Secondary Index SOURCE segment? If so call Index Maintenance (DFSDXMT0) now to keep the secondary index in synch.

VIII. The Logical Parent segment (SKILL) will have to be updated since we added a new XSKILL segment which is its Logical Child. If this SKILL has a counter we must add one to it and if it has a LCF pointer we must fill it in with the RbN of this 'new' XSKILL, but only if the new segment is the FIRST logical child. If the SKILL segment has a LCL pointer and the new XSKILL is last on the Logical Twin Chain (Virtual Pairing), we must fill in the LCL pointer with the RbN of the new segment.

Go back to the Buffer Pool and see where the Counter for the SKILL segment of "ARTIST" has a value of '1'. This is the first and only XSKILL pointing to this LParent and since it was just

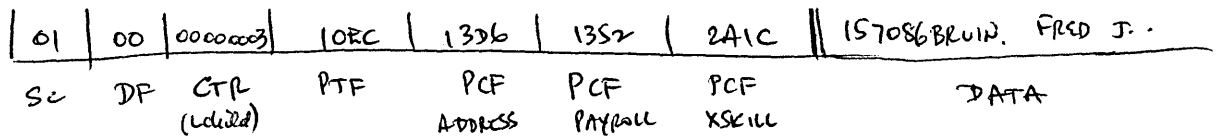
inserted that value was plugged in by Load-Insert at this time in the call flow.

IX. The Physical Parent Segment 'NAME' needs updating. Since its Lchild 'XNAME' is in HISAM all we can do is add 1 to its Counter. (No LC Pointer Possible into HISAM) In the Buffer Pool the NAME of "BRUIN, FRED J." is at 'E8C68' and the counter is at 'E8C6A'. The counter value is '3' in the dump which means it was '2' before this ISRT call.

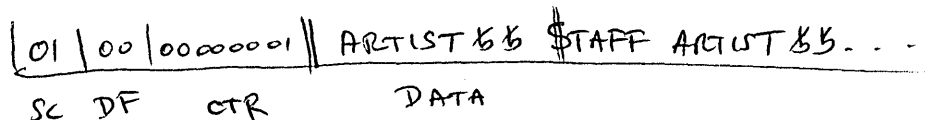
Log any of the segments changed as a result of this prefix updating and we can go back to the Call Analyzer (DFSDLA00).

X. Lets look in the buffer pool and map out some of the segments involved in this little ISRT. In case you forgot, the DL/I Buffer Pool starts at 'E6000'. The 'NAME' segment is located in the pool at 'E8C68'. The 'SKILL' segment is at 'E8E64'. The 'XSKILL' segment is at 'EA4E8'. The 'XNAME' segment is at 'E678C'.

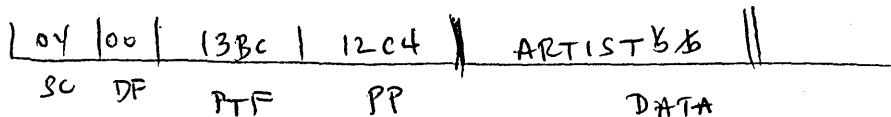
Map out NAME for FRED BRUIN here:



Map out SKILL for ARTIST here:



Map out the XSKILL for ARTIST here:



Map out the XNAME for FRED BRUIN here:

02	00	BRUIN, FRED J. . . .
SC	DF	DATA

Updated: 02/07/77

DELETE CALL FLOW

This call exercise consists of two steps. The first is a dlet of a root segment in the HISAM "Skill-Name" data base. In your handouts this is known as Logical Data Base 1. Step two is a dlet of a LC/LP concatenated segment "SKILL" via Logical Data Base 2. That is in terms of Physical data bases, a dlet of the segment "XSKILL" in the HIDAM data base. Snaps were taken after each call, so there is a formatted dump of the DL/I control blocks & Buffers after the GHU calls and a full region dump after each DLET call.

STEP ONE

- I. Step one is to Delete the 'SKILL' segment with a key value of PROG via the Logical Data Base 'SKILL-NAME'. This is a Delete via the physical path. The Delete RULE for this segment is Logical and SKILL is both a Physical and a Logical Parent, with its Lchild Physically Paired.

These facts (given to us by our friendly D.B.A.), mean that:

1. Either Physical Or Logical Deletion can occur first on the LP.
2. All Logical Children must be marked as Logically deleted.
3. All Physical Children must be marked as Physically deleted.

At the completion of the Delete call:

1. SKILL, XNAME, EXPR and EDUC will not be accessible on the Physical path.
2. XSKILL will not be accessible on the Logical Path.

- II. The actual CALL flow starts with the mandatory 'HOLD' call. In this case it is a GHU to SKILL qualified on SKCLASS of PROG. At the completion of this call the Level Table for SKILL (Level 1), has bit two on in the field LEVF1 (LEV +1) ' CSA4D ' indicating that this segment was returned to the user with a HOLD call: ' 28 '. If this was not done we would get a 'DJ' status code on the DLET call.

- III. Turn forward in the dump to the DLET call. First of all Dlete/Replace (DFSDL00), checks the key field in the users I/O area against the key stored away in the PCB. If these don't equal the call is not completed and a status code of 'DA' is issued.

To accomplish this we must locate the PST at DFSBNUCO

+ '8' on page 120 in the dump at ' B6008 '. The PSTDBPCB (+ '160') at ' B6168 ' has ' C57A4 '. Now go plus '24' to the DBPCBKFD field at ' C57C8 ' which has in it: PROG6666 . This is the key value as DL/I sees it from the GHU call. The field PSTUSER (+ 'BC') points to the Users I/O area at ' (B60C4) = ' and it contains the KEY in the first 8 bytes which is PROG6666 . The two are equal so we don't get a 'DA' status code. (F5710)

IV. The next question that Dlet/Replace must answer is do we have Logical Relationships, Secondary Indexing, or is this data base HD? To answer these questions we must start back at the "SKILL" SDB. To get there we take the "current" DBPCB + '10' at ' C5704 ' to the JCB at ' C5824 '. In the JCB at JCBSDB1 (+ '8') at ' C582C ' is the pointer to the root SDB. This is the "SKILL" SDB we want, so look there at SDBDDIR (+ '14') at ' B48A0 ' which is ' B4A74 '. This address is the DDIR entry for this SDB. Each DMB Directory entry (DDIR) is '28' bytes long and the first DDIR is pointed to by the SCDDLIM (+ 'F4') at ' B9274 ' which has ' B4A4C ' in the dump on page 123. The DDIR pointed to by the "SKILL" SDB is the 2nd entry, right? Looking at the DSECT for the DDIR's you see that the DEBNAME is at + '8' and the pointer to the DMB (DDIRADDR) is at + '10'. In the DDIR we just found at + '10' is ' B42AB ' or the DMB associated with "SKILL". Check the DMB for Logicals and/or Secondary Indexing and/or HD organization:

HISAM DMBORG (+A) at ' B42B2 ' checks for HD. It is ' 01 ' DMBFLAG (PSDB +20) at ' B4348 ' for Logicals & S.I. It is ' 20 '. Anything BUT '00' in the DMBFLAG gives us a "YES" answer to the question.

The answer to HD and S.I. is no, but the answer to Logicals is YES.

V. Based on that YES answer, DFSDLDSO builds a Dlete Work Area and enques the Root. The address of the work area is in the PST at offset '1FC'. DLDSO locates the first segment for XNAME EXPR & EDUC under this SKILL and saves the Key of each segment. Unfortunately this field in the PST (PSTDLTWA) is reset before the end of the call so in the dump the field is all zeroes!

VI. DFSDLDOO does a 'down' scan next. Since this is a DLET of an entire data base record in HISAM, this scan is done because of the Logical relationships involved. During this 'scan' DLD00 uses the work area to build

information needed to do any actual deleting in the next step.

VII. Now map out the D.B. Record for this deletion as this 'scan' would do. For a start, the SKILL segm. is in Block 1, DMB #2, DCB #1 in the buffer pool. That info came from the DSG for the SKILL data base. Also in the DSG at offset '0' is a pointer to the AMP. The DSG starts at 'C59D4'. This AMP in turn contains pointers to both DCB's for the ISAM/OSAM pair in use. The AMP is at 'B42D0'. Since you know what SKILL segm. we are deleting you can scan the dump to find it. The "EPPL" starts on Pg134 (That's not really cheating I'd say.) The root segm starts at 'E763F' with a segment code of '01', a delete byte of '24' and a counter of '3'. Following this is the segment data. The length of data is found in the PSDB, offset 'A' (DMBDL), which is at 'B4337' and is '5B'. If data starts at 'E7B45' (following the counter), adding the length should put us at the next segment in sequence. However the segment code at this address is '00' which is the 'flag-byte' portion of a HISAM P2 pointer. The next three bytes of the P2 pointer contain the RRN in the OSAM (Overflow) data set where our data base record is continued.

If our ISAM (Prime) data set was DMB #2, DCB #1 and we are in the OSAM portion now, it will be DMB #2 and DCB #2. The RRN value in the P2 pointer was '002E'. This means our next segment is in Block 2, DMB #2, DCB #2 at the start of the 13th LRECL right after the P3 pointer. (The P3 should have the high-order bit on to signify 'dependents only' in this LRECL.)

That's great but how long are the LRECLs? Go back to the AMP and look in offset '1C' for the pointer to the Overflow DCB. This OSAM DCB starts at 'B4574'. At '18' into the DCB is the BLOCK size and at '52' is the LRECL size. The LRECL size is '0069'.

The Buffer Pool starts at 'E6800' on page 134 in the dump, and the first buffer on the "forward" use chain is at '+8' at 'E6808' and is 'E7620'. This "most recently used" buffer is on page 136 and is for Block 1, DMB 2, DCB 1. The next buffer on the "forward" chain is at 'E6870'. It is for Block '2', DMB '2' and DCB '2', and since it is not a 'new' buffer and has no channel program, the data starts '+20' at 'E6890'. This is the correct one and I will help you locate the 13th (decimal) LRECL at 'E6D7C'. Look there and see if this is a P3 pointer with the high-order bit on. Right after the P3 is the segment code for our next segment in the D.B. Record. It should

be an XNAME for ADAMS, JOHN QUINCY (just to check your work up to this point!). The segment code is '02' and the delete byte is '24'. That is the complete prefix of XNAME so the data is next.

The length of data for XNAME (called NAME in this logical data base), is in the PSDB for level 2 under SKILL. This PSDB starts at 'B434C' and at + 'A' into this PSDB is the length of '2A'. While we are here, write down the lengths of the other segments: EXPR '14'; EDUC '46'.

Back to the buffer pool to add the length of data to XNAME and we find a segment code of '03' for the next segm. (Which is EXPR. with a value of CB3.) After the S.C. is the D.F. '24' and the data. At the end of the data this time we have a '00' S.C. which indicates this is a P2 pointer to the next LRECL. This P2 pointer at 'E6D2C1' contains '2F', and since we got to this LRECL via a P2 with a value one less than this, the P2 here is pointing to the next LRECL in this buffer. This "next" LRECL is at 'E6DE5'.

At the start of the next LRECL is another P3 pointer and then a S.C of '02' for the next XNAME segment. The name this time is: JONES. JOHN PAUL. This process continues on with pointers to LRECLs until the end of the D.B. Record is reached and a P4 pointer is found at 'E6F70'. The segments are as follows:

```
XNAME - 'ADAMS'
  EXPR - 'CB3'
XNAME - 'JONES'
  EXPR - 'A15'
  EXPR - 'CB4'
  EDUC - 'MICHIGAN.....MBA'
XNAME - 'SMITH'
  EXPR - 'PL3'
  EDUC - 'HOLY CROSS.....BBA'
```

This is a good time to look at the delete-bytes of all the segments in this D.B. Record and see that they are all '24', including the root. The value you see in the dump is after the entire DLET operation was completed. (On the 'down' scan made by Dlet/Repl they were all '00'.)

VIII. That completes the 'down' scan as DLD00 did it. Next DLD00 does the 'up' scan and sets bits 2 and 5 on in the delete bytes of each segment. That process marks each Physical Child (as well as the Root), as Physically Deleted, and these changes are logged out. That explains the values you see in the dump!

We return to DLDDO since XNAME is physically paired with XSKILL. The "rule" is that when we PD the XNAME we must LD its pair, XSKILL. This 'blocks' the logical path from NAME.

- IX. In order to locate the correct XSKILL segm. our path is to take the symbolic pointer in the first XNAME under this SKILL and retrieve NAME. In the buffer you can see that this is 'ADAMS, JOHN QUINCY ', and this NAME segment is in Block 2, DMB 3, DCB 1 at 'E9E25'. This is on page 137 in the dump. We now need to skip over the SC and DF and get to the Physical Child Pointer for XSKILL. That is the 5th word past the Delete Flag and is at 'E9D02' with an RbN of 'F02'. (This is HIDAM so the DMB & DCB are the same.) The RbN conversion gives us Block 2, Offset '10E'. Locate Block 5, DMB 3, DCB 1 (page 139 in the dump), and go into that buffer (starting past the header), to 'E9E66' for the XSKILL segm. The SC is '04', the PTF ptr. is '0', the PP ptr. is 'D3C' and the data is 'PROG'.

DLDDO turns on bits 2 and 6 in the DF to show 'precessing by Dlet.' and Logical Deletion. This change is logged out and we continue. Refer to the value of '22' at 'E9E37'.

The next step in the delete process is to check for additional XNAME segments in the 'deleted' D.B. Record. Besides ADAMS we found JONES and SMITH in the 'down' scan. The process is essentially the same for these two as it was for ADAMS. When the L.D. bit is turned on in the Delete Flag for XSKILL pointing to SKILL of PROG under NAME of SMITH this step of the process is complete.

- X. This change is logged and no more segments are to be deleted. DLDDO now returns control to Call Analyzer and this call is done.

STEP TWO

- I. Step two is to delete the same 'SKILL' segment via the Logical Data Base 'NAME-SKILL'. SKILL is a concatenation of the LC 'XSKILL' and the LP 'SKILL'. This deletion is made via the Logical Path and results in deleting the LC 'XSKILL' under 'NAME' of ADAMS. The

delete rules are Logical for both the LC and LP.

In step one this XSKILL segment was marked as Logically Deleted (LD bit on). It is physically paired with XNAME and has no physical children. At the completion of this DLET call:

XSKILL will not be accessible on the Physical Path.

II. Again the mandatory HOLD call is issued, this time against the 'NAME-SKILL' logical data base. The level table entry to check is Level 2 this time and LEVF1 at '_____ ' has '_____'. All is well! Check page 4 in this, the 3rd dump in this call flow example!)

III. Page forward to the DLET call where DLD00 checks the DBPCBKFD (+24) '_____ ' which has '_____ ' against the User's I/O Area (pointed to by PSTUSER, PST+BC). The User's I/O area (in DFSDDLTO), starts at '_____'. Where in the DBPCBKFD is the "key" of the segment to check this time? Go back to the SDB for "SKILL" at '84964' and look at SDBKEYFD ('+30') at '_____'. The address there is the position within the key feedback for Dlet/Repl to compare. The length of the comparison is in the SDB at SDBKEYLN ('+20' One Byte) at '84984' and is '_____'. If we compare these two values we find that we didn't change any KEYS so all is well this time.

IV. Logical relationships, secondary indexing or HD is the question again. This time the DMB fields at DMBORG (+A) '_____ ' has '_____ ' and DMBFLAG (PSDB +20 for XSKILL) at '_____ ' has '_____'. The answer is YES to both Logical and HD this time, so DLD00 needs a work area.

V. The scan 'down' is rather short this time because there are only two segments involved: NAME and XSKILL. NAME is in Block 2, DMB 2, DCB 1, at Offset '8': '_____'. Map out the segment using the prefix length from the PSDB +8 '_____ ' which is '_____ ' and the Prefix pointer options at PSDB +7 which is '_____'. That means there is a SC, DF, CTR, PTF, PC (Address), PC (Payroll), and a PC (XSKILL).

N O T E :

(In order to see the segments just as the "down" scan sees them we must turn back to the dump taken after the GHU call. That is the GHU call made with the PCB for Logical Data Base 2 with segments NAME

and SKILL with values of "ADAMS" and "PROG ". In that snap-dump on page 9 in the buffer pool we can find the "old" segment values just as Dlet/Repl saw them.)

The Counter value is '00000001' and the PC Ptr to XSKILL is '0E86'. Invoking the RbN conversion routine gives us: XSKILL at Block 2, DMB 2, DCB 1, offset '1CE' into the buffer. The buffer starts at 'B7B0' so at '+20' from there at 'B7BD0' is the start of "data". We must add our offset, which was '1CE' to the start of "data" which puts us at '_____ ' in the "old" dump on page 9. The SC is '04', DF is '22' (LD bit on), PT Ptr is '00000000' and the PP Ptr is '00000CC0'. With no Physical Children nor Twins the 'down' scan is complete.

VI. DLDDO does another 'up' scan and since this is HIDAM the action is a little more involved. By setting Bit 5 on in the DF of XSKILL this segm. now becomes both Logically and Physically deleted and the space can be freed-up.

VII. DLDAO calls DHDSO (HD Space Management), to free up the space, re-adjust the FSE's and if necessary, update the Bit Map. In order to see the results of this action better I suggest that you compare the 'XSKILL' segment in the SNAP (after the GHU call), with the same segment in the SNAP after the delete is completed. In the first dump the segm. is at 'E8FF6' and the first FSE (FSEAP) for this block points to an FSE at offset '~~0~~'. Turn forward to the 'after' situation and the same FSE (FSEAP) points to '1CE'. By adding this offset to the FSEAP you get 'E8FF6' which is the Seg Code location of our deleted segment. However, this location is now the start of a "freed-up" area and contains a new FSE. Check this out by comparing the pointer to the next FSE and the length of free space. The offset to the next FSE is '0' (what the old value was in the FSEAP), and the length is '17' (the free space including the FSE itself). FSE offset pointers are always relative to the first FSE, known as the FSEAP which is the start of the Block.

IX. If this free-space was large enough to change the "status" of the block from "not enough space for the largest segment in the Data Set Group", to "enough space ...", the Bit Map would be updated and the bit corresponding to this block would be turned on. This time the space is too small to make a difference so no call is made.

X. Delete is almost finished, but the "down" scan showed that XSKILL is physically paired with XNAME and whenever we physically delete one member of a pair we must logically delete the other. That means we must locate the XNAME pointing to "ADAMS, JOHN QUINCY" and logically delete it. Just to show you my heart is in the right place I'll tell you where the start of this segment is in the buffer pool! (Nice??) It is at '6657F'. The very next byte is the ~~segment~~ code and sure enough the Logical delete bit is on AND the HISAM Segment delete bit is also on! In fact the value there is 'A1' and this segment will be physically removed from the data base at reorganization time.

delete byte

XI. If you remember there was more than just this one XNAME under the SKILL of "PROG". In fact the names were JONES & SMITH, and these segments have NOT been deleted in the other data base. This points out the situation Dlet/Replace must check for before marking the SKILL segment as deleted (Bit 1 on in the D.F.). Dlet/Replace checks for more XNAME segments under the SKILL of "PROG" and finds two. The D.F. for each of these has a value of '24' which means they have only been marked as "not available" on the Physical Path. Therefore no action is taken on the D.F. for SKILL of "PROG".

That does NOT mean there is nothing to do on the SKILL segment because we have physically deleted one of its Logical Children. That means we must decrement the Counter by 1, and the segment is in the buffer pool at 'B874F'. The counter at 'E92E9' is '2' now that the DLET is finished.

E92E9

This same operation must be done for the Root segment NAME for "ADAMS, JOHN QUINCY" which starts at 'B7BD8' in the buffer pool. The counter value at 'E873X' is '0' which means that there are no more Logical Children in the data base for this NAME segment.

E8730

XII. After the changes have been logged out delete has "done its thing" and control returns to DLA00. Call Analyzer again returns control to Program Request Handler whic returns control to the application program. In this case the Return Code was 'bb' and the DLET went OK!

Updated: 02/09/77