Retrieve calls

The IMS programming interface

COBOL: The ENTRY statement

At execution time, MVS gives control to DL/1. The DL/1 control program then passes control to the application program through the Entry Point and specifies all the PCB names used by the program. The PCB (Program Communication Block) addresses are passed to the program as parameters on entry.

The Entry Point is set up in one of two ways:

```
      PROCEDURE DIVISION USING PCB-name1, PCB-NAME2,....
or
      ENTRY  'DLITCBL'  USING  PCB-name1, PCB-NAME2,....
```

The Entry statement is the first entry in the PROCEDURE DIVISION and is the old method of specifying the entry point. The sequence of PCB names on the entry statement must be the **same** as their sequence in the PSB (Program Specification Block).
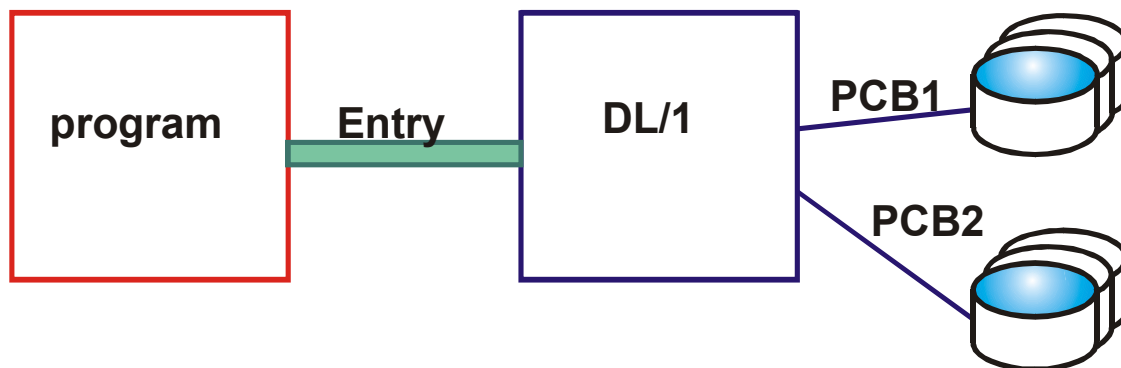
Programs use PCBs to access databases

IIB-10

PL/1: Entry to a PL/1 Program

The first procedure of a program is as follows:

```
      ANYPROG: PROCEDURE(PCB-ptr1,...,PCB-ptrn)OPTIONS(MAIN);
```

As with COBOL, the sequence of PCB pointers in the PROCEDURE statement must be the same as the sequence of the PCB names in the PSB.

IMS calls

When a program accesses an ordinary file, it will use statements such as READ and WRITE. When a program needs to access a segment in a DL/1 database, it will use a CALL statement to ask IMS to perform the required action:

CALL 'CBLTDLI' USING   *function*   *what do you want to do?*
                            PCB   *which database?*
                            IOAREA   *where will the data be placed?*
                            SSA1   *SSAs indicate ...*
                            :   *... which segments ...*
                            SSAn.   *... we want*

FUNCTION   4 byte DL/1 function

PCB   Name of PCB mask

IOAREA   A work area long enough to hold the segments being retrieved

SSA1 to SSAn   Segment Search Arguments – values for keys of segments

The functions are defined as 4-byte WORKING-STORAGE areas and are normally stored in a copy book. The contents would include the following:

```
01   CLAIMS-WORKING-CONSTANTS.
     02 IMS-ACCESS-CONSTANTS.
        03 DB-ACCESS-CONSTANTS.
           04 GU-FUNC                    PIC X(4)
                                         VALUE 'GU  '.
           04 GN-FUNC                    PIC X(4)
                                         VALUE 'GN  '.
           04 GNP-FUNC                   PIC X(4)
                                         VALUE 'GNP '.
           04 GHU-FUNC                   PIC X(4)
                                         VALUE 'GHU '.
           04 GHN-FUNC                   PIC X(4)
                                         VALUE 'GHN '.
           04 GHNP-FUNC                  PIC X(4)
                                         VALUE 'GHNP'.
           04 ISRT-FUNC                  PIC X(4)
                                         VALUE 'ISRT'.
           04 REPL-FUNC                  PIC X(4)
                                         VALUE 'REPL'.
           04 DLET-FUNC                  PIC X(4)
                                         VALUE 'DLET'.
```

PCB Masks

The PSB used by a program will contain one or more PCBs. The PSB is stored externally to the application program, so the program needs to gain access to the PSB so that it can obtain certain information about the database which is being accessed. The program communicates with PCB details via a **PCB Mask**.
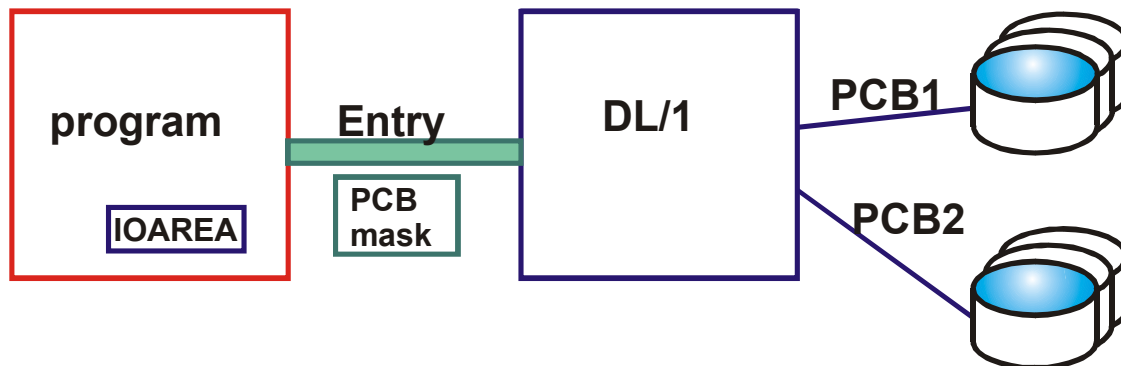
The PCB mask hold IMS interface details

IIB-25

The PCB mask contains:

|		
Database Name	Char	8
Hierarchical Level	Char	2
DLI Status Code	Char	2
Processing Options	Char	4
Reserved for DL/1	Binary	4
Segment Name Feedback	Char	8
Feedback Key Length	Binary	4
Number of Sensitive Segments	Binary	4
Key Feedback Area	Char	n

Sample PCB masks

COBOL

- You must code one mask area for each PCB in the Program's PSB.
- PCB masks are placed in the Linkage Section.

```
01   PCBNAME.
     02  DBD-NAME        PIC X(8).
     02  SEG-LEVEL       PIC XX.
     02  STATUS-CODE     PIC XX.
     02  PROC-OPTIONS    PIC XXXX.
     02  RESERVED        PIC S9(5) COMP.
     02  SEG-NAME        PIC X(8).
     02  LENGTH-FB       PIC S9(5) COMP.
     02  NO-SENSEGS      PIC S9(5) COMP.
     02  KEY-FBAREA      PIC X(18).
```
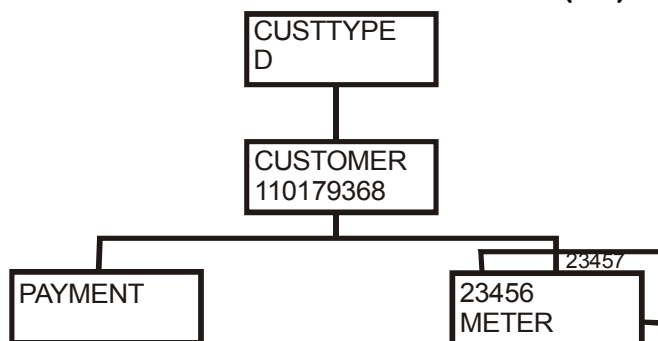
If the key of PAYMENT is 8 bytes long, the key feedback area must have a length of 18 bytes.
$$1 + 9 + 8$$

IIB-50

PL/1

- The PCBs are passed as pointers, so only one mask is needed.
- Set PTR to the correct PCB address before issuing the call to DL/1.

```
DCL   1    PCBNAME          BASED(PTR),
      2    DBD_NAME         CHAR(8),
      2    SEG_LEVEL        CHAR(2),
      2    STATUS_CODE      CHAR(2),
      2    PROC_OPTIONS     CHAR(4),
      2    RESERVED         FIXED BIN(31),
      2    SEG_NAME         CHAR(8),
      2    LENGTH_FB        FIXED BIN(31),
      2    NO_SENSEGS       FIXED BIN(31),
      2    KEY_FBAREA       CHAR(18);
```
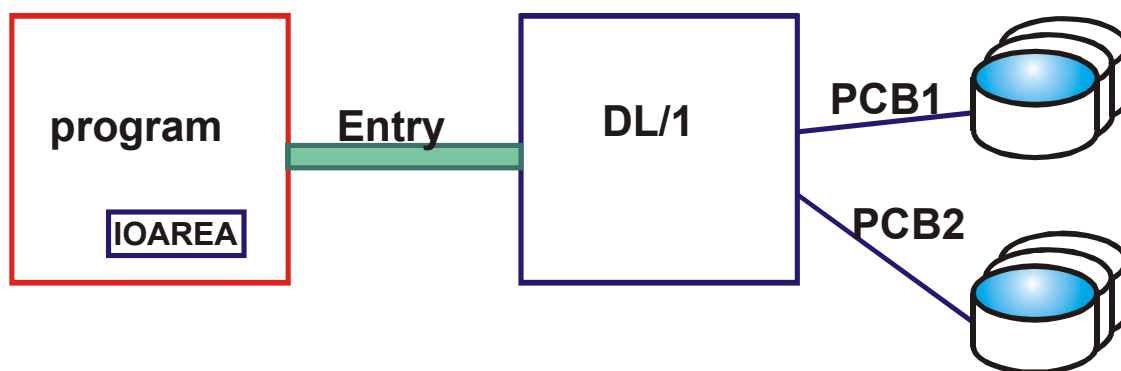
Input/Output Areas

Each program must provide an area into which DL/1 can place the data. The size and format of the **I/O Areas** is often supplied by the DBA.

Usually, each segment has its own I/O Area. The structure is the same as for normal record I/O Areas.

IMS data is placed in an IOAREA

IIB-20

In the following example, we want to retrieve a segment from the database referenced by **PCB1**. The call is a **Get Unique** which is similar to a random read. Details of the segment we want are placed in a **Segment Search Argument**, part of which contains a value of P012001234. When IMS locates the required segment it copies it to our specified I/O area (IO-POL in this case). We will examine the structure of the SSAs in the next few pages.

```
MOVE 'P012001234'  TO SSA-POL-VALUE
CALL  'CBLTDLI'  USING  GU-FUNC
                       PCB1
                       IO-POL
                       SSA-POL
```

At this point of the program, we should check the **status code** telling us if the call was successful or not. The status code is one of the fields of the PCB.

Segment Search Arguments

The segment search argument tells IMS which segment is required. The simplest SSA is said to be **unqualified:** it contains the 8 character segment name followed by one blank. The result of using an unqualified SSA call will return the *next segment* of the specified type from the database.
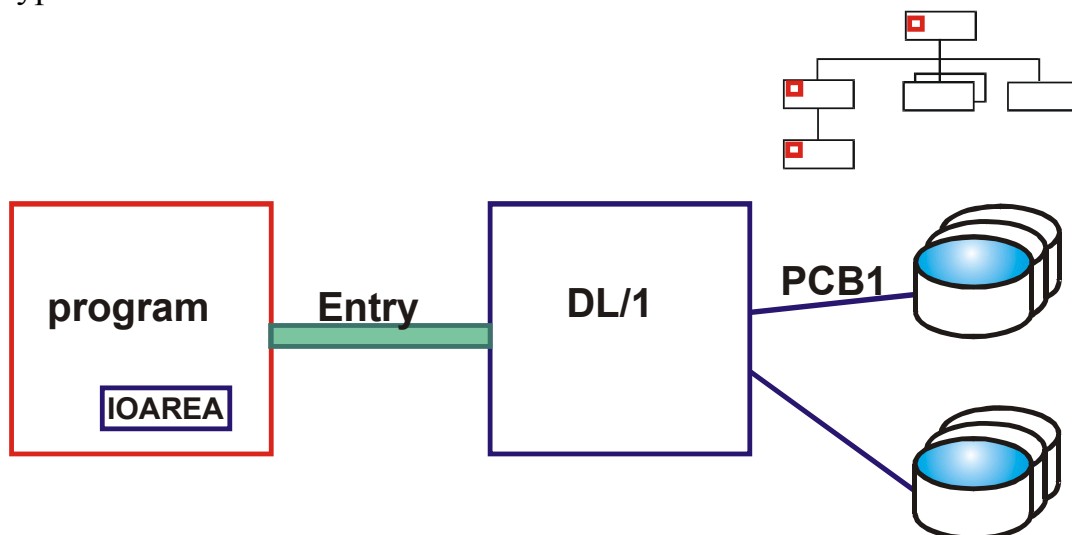
SSAs specify the target segments

IIB-30

```
WORKING-STORAGE SECTION.

01   GET-UNIQUE              PIC X(4) VALUE 'GU '.

01   CUSTOMER-INFO-IO-AREA.
     02  CUSTOMER-NO         PIC X(5).
     02  CUSTOMER-NAME       PIC X(20).
     02  CUSTOMER-ADDR       PIC X(15).
     02  CUSTOMER-TOWN       PIC X(15).
     02  CUSTOMER-POSTCODE   PIC X(8).

01   CUSTOMER-SSA-UNQ
     02  FILLER             PIC X(8) VALUE 'CUSTINFO'
     02  FILLER             PIC X(1) VALUE SPACE.

PROCEDURE DIVISION.
ENTRY-POINT SECTION.
     ENTRY 'DLITCBL' USING PCB-MASK

     CALL 'CBLTDLI' USING GET-UNIQUE
                          PCB-MASK
                          CUSTOMER-INFO-IO-AREA
                          CUSTOMER-SSA-UNQ
```

```
PCB-MASK contents following a successful call:
          DBD-NAME          CUSTDB
          SEG-LEVEL         1
          STATUS-CODE       bb
          PROC-OPTION       GO
          RESERVE-DLI       -
          SEG-NAME-FB       CUSTINFO
          LENGTH-KEY-FB     5
          NUMB-OF-SEN-SEG   4
          KEY-FB-AREA       01234
```

If we want to retrieve a segment with a specific key value then we supply a fully qualified SSA.

Qualified SSA format

segment name	command codes	(	segment name	op	field value	)
8	4(var)	1	8	2	n	1

Unqualified SSA format

segment name	command codes	฿
8	4(var)	1

IPC-10

Segment Name   The name of the segment as defined in the PSB.

Command Codes   Additional go-faster options; see later; optional

Begin - Op '('   If present, it indicates that the SSA is qualified.

Field Name   This is the name of a field which has been defined within this segment in the DBD. The name used in the SSA must exactly match the name in the DBD. It can be a Key Field or a Search Field.

OP   This is a relational operator which defines how the contents of the named field will be compared with the given value. The operators available are:

|		
=b b=	EQ	equal to b indicates a blank
>= =>	GE	greater than
<= =<	LE	less than or equal to
b> >b	GT	greater than
b< <b	LT	less than
¬=	NE	not equal to

Field Value   The value which will be compared with the contents of the named field. The value must have the same format as the named field.

End-Op ')'   This signifies the end of the SSA. SSAs are treated as character strings, so do not omit the 'end-op'.

```
COBOL      01    CUST-SSA.
                 05     SEGNAME    PIC X(8)    VALUE 'CUSTOMER'.
                 05     COMMAND-CD PIC X(4)    VALUE '*---'.
                 05     BEGIN-OP   PIC X       VALUE '('.
                 05     KEYNAME    PIC X(8)    VALUE 'CACNO'.
                 05     COMP-OP    PIC XX      VALUE 'b='.
                 05     KEYVALUE   PIC 9(8).
                 05     END-OP     PIC X       VALUE ')'.
```
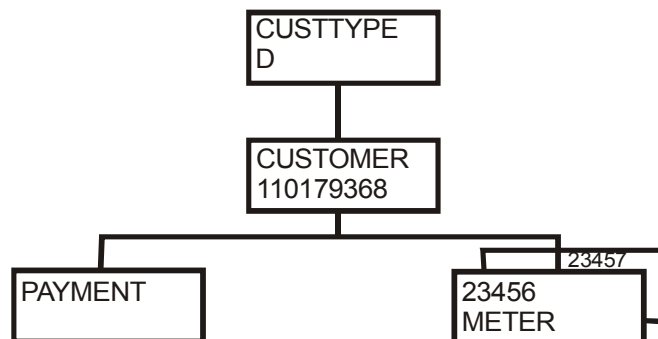

IIB-50

Examples of SSAs

1. Qualified at each level:

    CUSTTYPE*---(TYPEbbbbb=D)
    CUSTOMER*---(CACNObbbb=110179368)
    METERbbb*---(MTRNObbbb=71628)

2. Missing level:

    CUSTTYPE(TYPEbbbbb=D)
    METERbbb(MTRNObbbb=12249)

3. Qualified and unqualified:

    CUSTTYPE*---(TYPEbbbbb=I)
    CUSTOMERb

    CUSTTYPEb
    CUSTOMER*---(CACNObbbb=368179110)

```
WORKING-STORAGE SECTION.
01   GET-UNIQUE              PIC X(4) VALUE 'GU '.

01   SHIP-ADDRESS-I-O-AREA.
     02   SHIP-NUMBER        PIC X(5).
     02   SHIP-ADDR          PIC X(15).
     02   SHIP-TOWN          PIC X(15).
     02   SHIP-POSTCODE      PIC X(8).
01   CUST-SSA-Q.
     02   FILLER             PIC X(8) VALUE 'CUSTINFO'
     02   FILLER             PIC X(4) VALUE '*---'.
     02   FILLER             PIC X(1) VALUE '('.
     02   FILLER             PIC X(8) VALUE 'CUSTNO'.
     02   FILLER             PIC X(2) VALUE '= '.
     02   CUST-KEY           PIC X(5).
     02   FILLER             PIC X(1) VALUE ')'.
01   SHIP-SSA-Q.
     02   FILLER             PIC X(8) VALUE 'SHIPADDR'
     02   FILLER             PIC X(4) VALUE '*---'.
     02   FILLER             PIC X(1) VALUE '('.
     02   FILLER             PIC X(8) VALUE 'CSTSHPNO'.
     02   FILLER             PIC X(2) VALUE '= '.
     02   SHIP-TO-KEY        PIC X(5).
     02   FILLER             PIC X(1) VALUE ')'.


LINKAGE SECTION.
01   PCB-MASK.
     02   DBD-NAME           PIC X(8).
     02   SEG-LEVEL          PIC X(2).
     02   STATUS-CODE        PIC X(2).
     02   PROC-OPTION        PIC X(4).
     02   RESERVE-DLI        PIC S9(5) COMP.
     02   SEG-NAME-FB        PIC X(8).
     02   LENGTH-KEY-FB      PIC S9(5) COMP.
     02   NUMB-OF-SEN-SEG    PIC S9(5) COMP.
     02   KEY-FB-AREA        PIC X(10).


PROCEDURE DIVISION.
ENTRY-POINT SECTION.
     ENTRY 'DLITCBL' USING PCB-MASK

     MOVE IN-CUST-NO TO CUST-KEY
     MOVE IN-SHIP-NO TO SHIP-TO-KEY

     CALL 'CBLTDLI' USING  GET-UNIQUE
                           PCB-MASK
                           SHIP-ADDRESS-I-O-AREA
                           CUST-SSA-Q
                           SHIP-SSA-Q

     IF STATUS-CODE EQUAL 'GE'
          PERFORM SHIP-TO-ADDR-NOT-ON-FILE
                .
                .
SHIP-TO-ADDR-NOT-ON-FILE SECTION.
                .
```

> This sample coding is for a qualified Get Unique using two SSAs.

```
PCB-MASK contents following a successful call.
          DBD-NAME          CUSTDB
          SEG-LEVEL         2
          STATUS-CODE       bb
          PROC-OPTION       GO
          RESERVE-DLI       -
          SEG-NAME-FB       SHIPADDR
          LENGTH-KEY-FB     10
          NUMB-OF-SEN-SEG   4
          KEY-FB-AREA       0123454321
```

```
PCB-MASK contents following a failed
call.
          DBD-NAME          CUSTDB
          SEG-LEVEL         1
          STATUS-CODE       GE
          PROC-OPTION       GO
          RESERVE-DLI       -
          SEG-NAME-FB       CUSTINFO
          LENGTH-KEY-FB     5
          NUMB-OF-SEN-SEG   4
          KEY-FB-AREA       01234
```
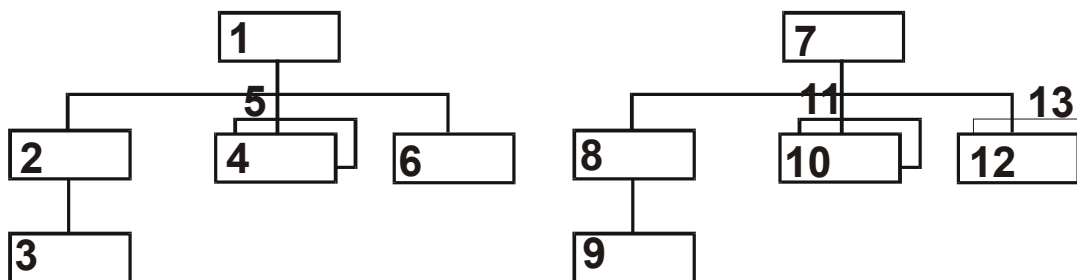
Retrieve Functions

IMS has several functions which will retrieve a segment:

GU   Get Unique, similar to a random read. Any *unqualified* GU will start at the *beginning* of the data base whether or not a previous position has been established.

GN   Get Next, similar to a sequential read. If no position has been established and an unqualified call has been issued, processing will begin from the start of the data base. Successive Get Next calls will retrieve segments in the hierarchical order.

```
        ┌─────┐                           ┌─────┐
        │  1  │                           │  7  │
        └──┬──┘                           └──┬──┘
     ┌─────┤  5  ┌─────┐          ┌──────┤  11  ┌──────┐ 13
  ┌──┴──┐  ├──┴──┐ ┌──┴──┐     ┌──┴──┐  ├──┴──┐  ┌──┴──┐
  │  2  │  │  4  │ │  6  │     │  8  │  │ 10  │  │ 12  │
  └──┬──┘  └─────┘ └─────┘     └──┬──┘  └─────┘  └─────┘
  ┌──┴──┐                     ┌──┴──┐
  │  3  │                     │  9  │
  └─────┘                     └─────┘
```
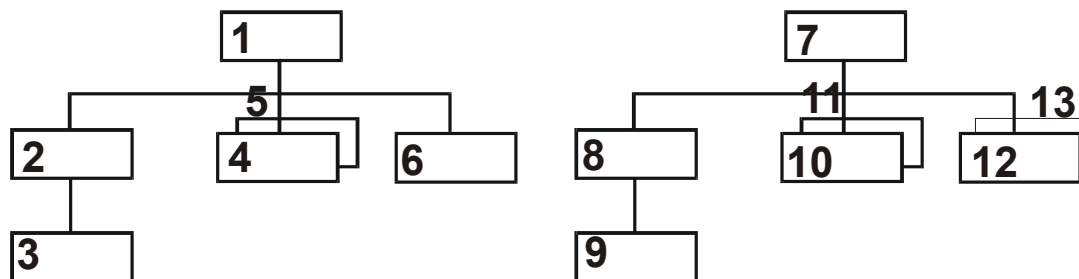
Segment hierarchical order IIB-40

GNP   Get Next within Parent; sequential read of dependent segment. A GNP call is similar to a GN except that only lower level dependent segments of an established parent are returned. Parentage is automatically set when a GNP is issued **following a successful GN or GU**.

GH calls Get Hold: as above, but IMS will lock the segment.
          These are needed before you can delete or update a segment
                  GHU
                  GHN
                  GHNP

Segment Search Arguments for GET Calls

- The calls may or may not have SSAs.

- SSAs may be qualified or unqualified.

- The search field must be known to DL/1.

- If an unqualified SSA is used, any occurrence of the segment type will satisfy the call.

- SSAs must be in hierarchical order.

- Unqualified SSAs are assumed at missing levels. As a general rule, you should not leave out any levels when using GU calls.

- Avoid relational operators other than = to qualify an SSA for a root segment.

Segment hierarchical order IIB-40

© Oakwood Computing Ltd
A Computer Education Techniques, Inc. Instructor-led course.

Status Codes

When a program issues a database call, IMS returns a value which indicates whether the call was successful or not. This value is called a status code. The program should check the value of the status code and take appropriate action if an error has occurred.
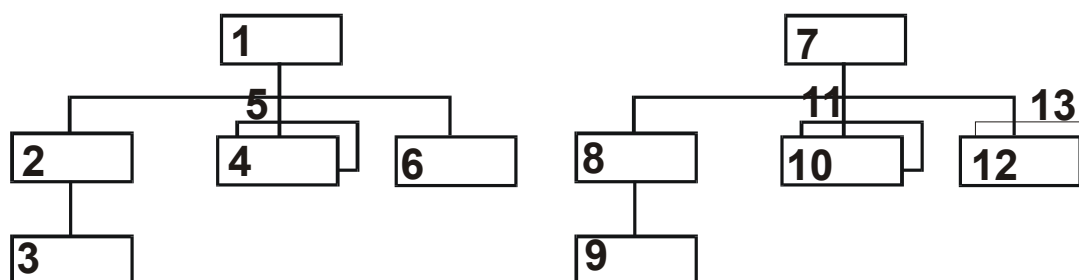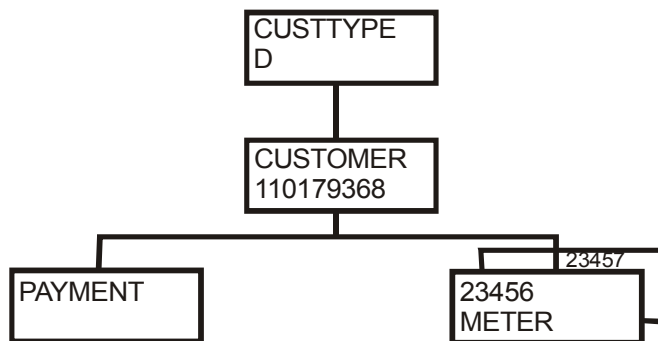

Segment hierarchical order   IIB-40

bb	Successful call	GA	Crossed hierarchical boundary
AB	No segment I/O area specified	**GB**	**End of data base**
AC	SSAs not in hierarchical sequence	**GE**	**Segment not found**
AD	Invalid function parameter	**GK**	**Different segment type at same level returned**
AH	At least one SSA is required	**GP**	**No parentage established**
AI	Data management open error	II	Segment already exists (update)
AJ	SSA qualification format invalid	LB	Segment already exists (load)
AK	Invalid field name in call	LC	Segment keys are out of sequence
AM	Call function conflicts with processing option	LD	Segment's parent has not been loaded
AO	I/O error	LE	Different sequence of sibling segments from the DBD.
DA	Segment key field has been changed	DX	Violated delete rule
DJ	No preceding successful GH call	IX	Violated insert rule
		RX	Violated replace rule

Nx N as the first character indicates that an error has occurred during index maintenance, the 2nd character indicates the 2nd character of a normal status code

Examples of Call Statements and the corresponding status codes


```
┌──────────────┐
│ CUSTTYPE     │
│ D            │
└──────┬───────┘
       │
┌──────┴───────┐
│ CUSTOMER     │
│ 110179368    │
└──────┬───────┘
   ┌───┴──────────────┐
┌────────────┐   ┌──────────23457─┐
│ PAYMENT    │   │ 23456          │
│            │   │ METER          │
└────────────┘   └────────────────┘
```

IIB-50

Call No.	Function	Segment Search Argument Returned	Status Code
1	GU	CUSTTYPE(TYPEbbbbb=D)	bb
2	GN	CUSTOMER(CACNObbbb=110179368)	bb
3	GU	CUSTTYPE(TYPEbbbbb=D) CUSTOMER(CACNObbbb=110179368) METERbbb(MTRNObbbb=56789)	GE
4	UG	CUSTTYPE(TYPEbbbbb=D)	AD
5	GU	CUSTTYPE(TYPEbbbbb=D) METERbbb(MTRNObbbb=45678) CUSTOMER(CACNObbbb=345678912)	AC
6	GU	CUSTOMER(NUMBERbbb=123456789)	AK
7	GN	CUSTOMER Sets parentage	bb
	GNP	METERbbb(MTRNObbbb=23456)	bb
	GNP	METER	bb
	GNP	METER	GE
8	GNP	CUSTOMER	GP
9	GU	CUSTTYPE(TYPEbbbb=D)	AJ

```
WORKING-STORAGE SECTION.
01    GET-NEXT                      PIC X(4) VALUE 'GN '.
01    EOJ-SW                        PIC X(3) VALUE SPACES.
01    SEG-I-O-AREA                  PIC X(63).
01    CUST-INFO-I-O-AREA REDEFINES SEG-I-O-AREA.
      02    CUSTOMER-NO             PIC X(5).
      02    CUSTOMER-NAME           PIC X(20).
      02    CUSTOMER-ADDR           PIC X(15).
      02    CUSTOMER-TOWN           PIC X(15).
      02    CUSTOMER-POSTCODE       PIC X(8).
01    SHIP-ADDRESS-I-O-AREA REDEFINES SEG-I-O-AREA.
      02    SHIP-NUMBER             PIC X(5).
      02    SHIP-ADDR               PIC X(15).
      02    SHIP-TOWN               PIC X(15).
      02    SHIP-POSTCODE           PIC X(8).
01    BILLING-I-O-AREA REFEFINES SEG-I-O-AREA
      02    BILLING-DATE            PIC X(6).
      02    BILLING-INVOICE-NUMBER  PIC X(8).
      02    BILLING-AMOUNT          PIC S9(6)V99.
01    CASH-I-O-AREA REDEFINES SEG-I-O-AREA.
      02    CASH-DATE               PIC X(6).
      02    CASH-INVOICE-NUMBER  PIC X(8).
      02    CASH-AMOUNT             PIC S9(6)V99.
LINKAGE SECTION.
01    PCB-MASK.
      02    DBD-NAME                PIC X(8).
      02    SEG-LEVEL               PIC X(2).
      02    STATUS-CODE             PIC X(2).
      02    PROC-OPTION             PIC X(4).
      02    RESERVE-DLI             PIC S9(5) COMP.
      02    SEG-NAME-FB             PIC X(8).
      02    LENGTH-KEY-FB           PIC S9(5) COMP.
      02    NUMB-OF-SEN-SEG         PIC S9(5) COMP.
      02    KEY-FB-AREA             PIC X(10).
PROCEDURE DIVISION.
ENTRY-POINT SECTION.
      ENTRY 'DLITCBL' USING PCB-MASK.
      PERFORM GET-NEXT-ROUTINE
            UNTIL EOJ-SW = 'EOJ'
      PERFORM END-OF-JOB-PROCESSING
      GOBACK.
GET-NEXT-ROUTINE SECTION.
      CALL 'CBLTDLI' USING    GET-NEXT
                              PCB-MASK
                              SEG-I-O-AREA.
      EVALUATE STATUS-CODE
      WHEN 'GB' MOVE 'EOJ' TO EOJ-SW
      WHEN ' '
      WHEN 'GA'
      WHEN 'GK' EVALUATE SEG-NAME-FB
                  WHEN 'CUSTINFO'      PERFORM PROCESS-CUST-INFO
                  WHEN 'SHIPADDR'      PERFORM PROCESS-SHIP-TO-ADDRESS
                  WHEN 'BILLING'  PERFORM PROCESS-BILLING
                  WHEN OTHER           PERFORM PROCESS-CASH
                END-EVALUATE
      WHEN OTHER PERFORM ERROR-END-OF-JOB
      END-EVALUATE
      EXIT.
```

This program sample uses an unqualified GET NEXT call to read segments into a general area. The program must determine the type of segment read in and take appropriate action.

Program Execution

A batch program which uses IMS databases must be invoked from **DFSRRC00**, which is the batch initialisation module for MVS.

When the program is link-edited, the language interface is linkage edited with the application program. This contains the entry points

   PLITDLI, CBLTDLI, ASMTDLI.

At execution time, DFSRRC00 must be told the target program name and the name of the PSB which will be used.

Job control for execution

//STEP   EXEC PGM=DFSRRC00,PARM='DLI,progname,psb-name'

Batch Address Space

DFSRRC00

PSB
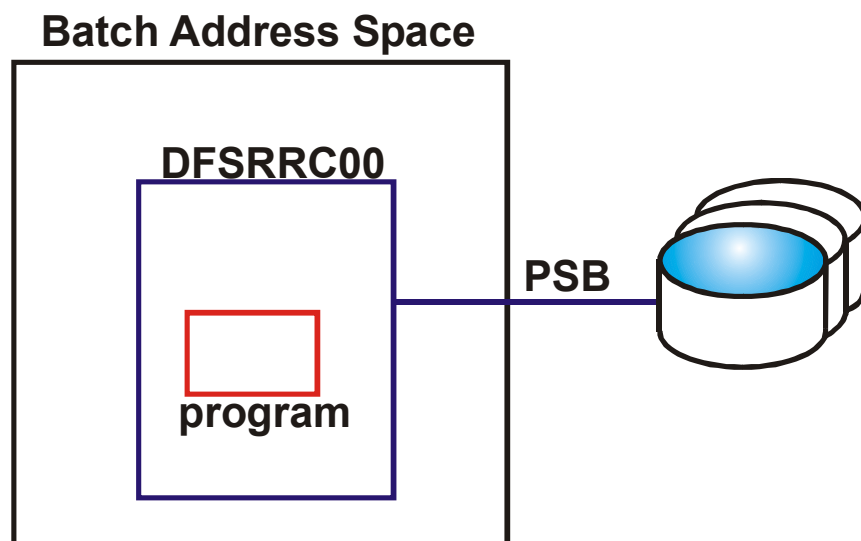
program

IIB-60

Batch IMS initialisation

Program Structure

So far, we have examined all of the basic components of an IMS (DL/1) program. We can now bring them all together and see how they are used in an actual program.

COBOL

```
IDENTIFICATION DIVISION.
PROGRAM-ID. XXXXXXXX.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
    data descriptions for call functions
    I/O area(s)
    SSAs
LINKAGE SECTION.
    PCB mask(s)
PROCEDURE DIVISION.
    ENTRY 'DLITCBL' USING PCB mask(s).
    .
    .
    CALL 'CBLTDLI' USING ..............
    check status code
    .
    .
    GOBACK.
```

PL/1

```
EXPROG:PROCEDURE(PCB-pointer(S)OPTIONS(MAIN);
DCL PLITDLI ENTRY;
declare PCB mask
declare call functions
declare I/O area(s)
declare SSAs
.
.
set PCB pointers to appropriate address
CALL PLITDLI (.................);
check status codes
.
.
RETURN;
```