COBOL: New Functions and Features

IBM COBOL for VSE/ESA Program Number 5686-068

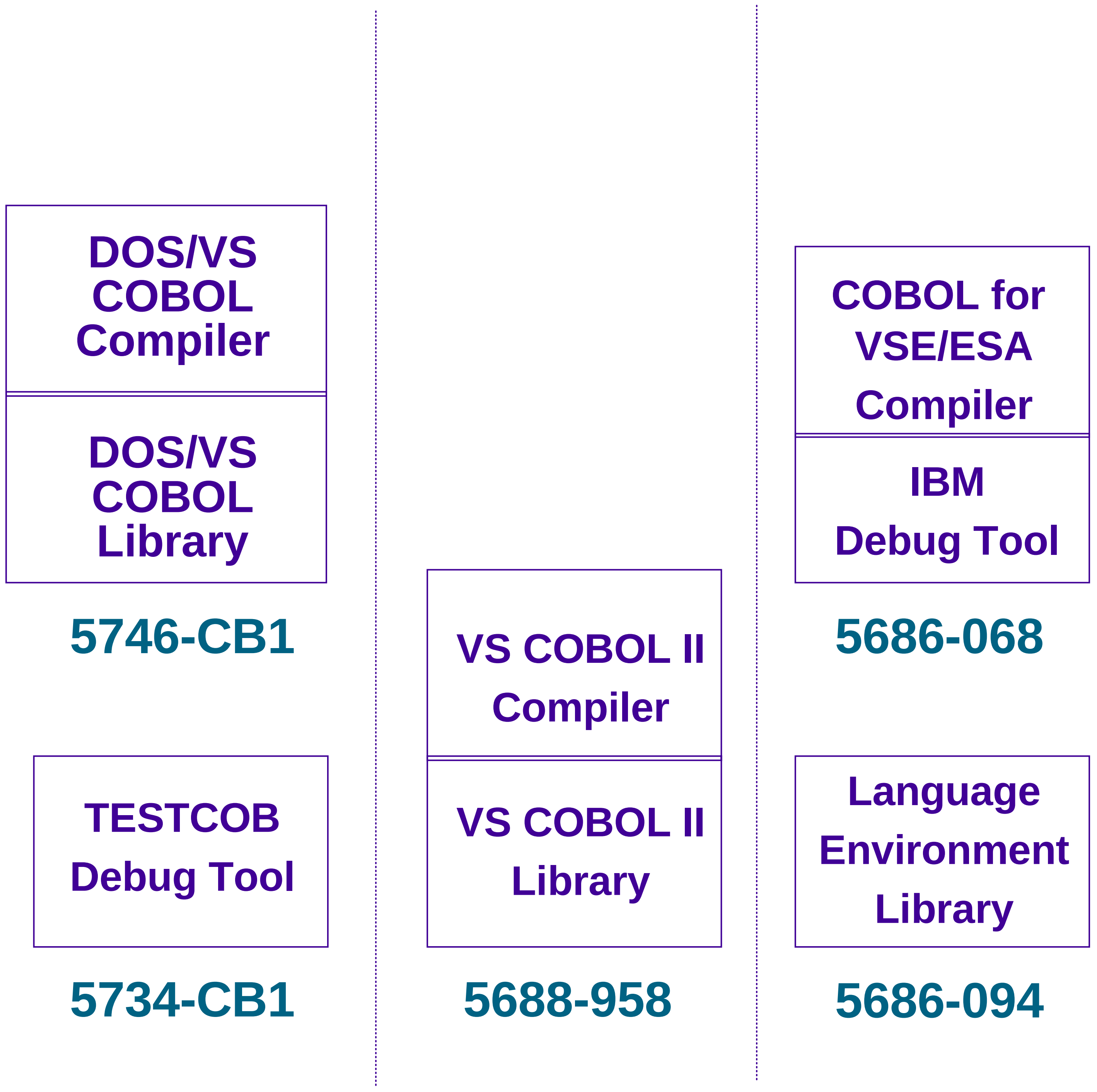
IBM COBOL for VSE/ESA Presentation Topics

- Position COBOL for VSE/ESA
  - Basically, it's VS COBOL II release 4
- Support of Features Introduced by VS COBOL II
- New Language Features with COBOL for VSE/ESA
- Language Environment Support

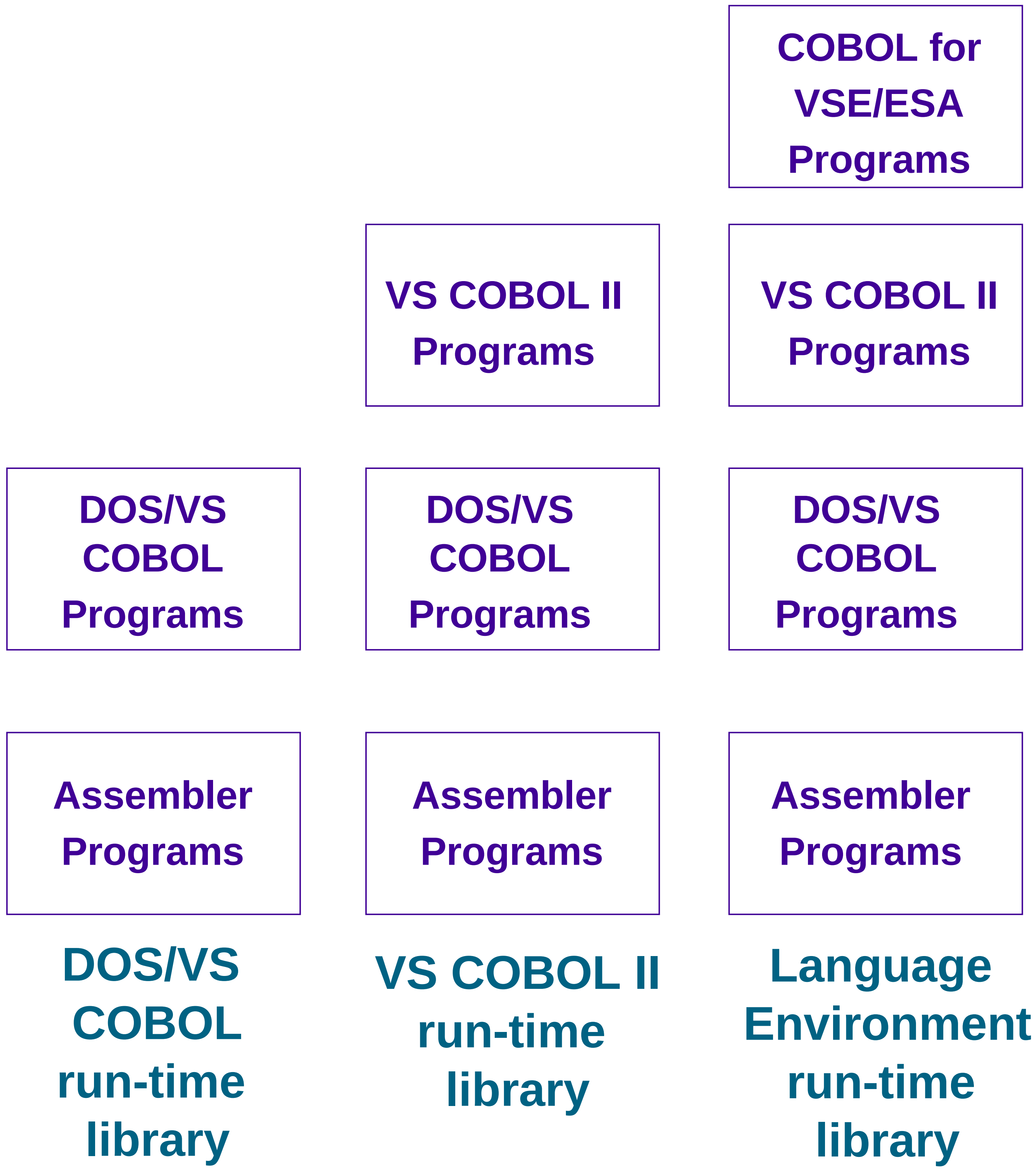
A History of COBOL

DOS/VS COBOL	VS COBOL II	COBOL for VSE/ESA
		Intrinsic Functions (addendum to '85 Std) Language Extensions Support for Language Environment Support for Debug Tool
	COBOL 85 Standard (no intrinsic functions) Structured Programming National Language DBCS Improved CICS Interface 31-Bit Addressing Reentrancy, Fast Sort Optimizer, SAA Flagging Interactive Debugging (full screen mode)	COBOL 85 Standard Structured Program ming National Language DBCS Improved CICS Interface 31-Bit Addressing Reentrancy, Fast Sort Optimizer, SAA Flagging
COBOL 74 Standard 74 STD FIPS Flagging Dynamic Debugging Batch Debugging Interactive Debugging (line mode)	COBOL 74 Compa tibility 85 STD FIPS Flagging Dynamic Debugging Batch Debugging Interactive Debugging (line mode)	COBOL 74 Compa tibility 85 STD FIPS Flagging Dynamic Debugging Batch Debugging

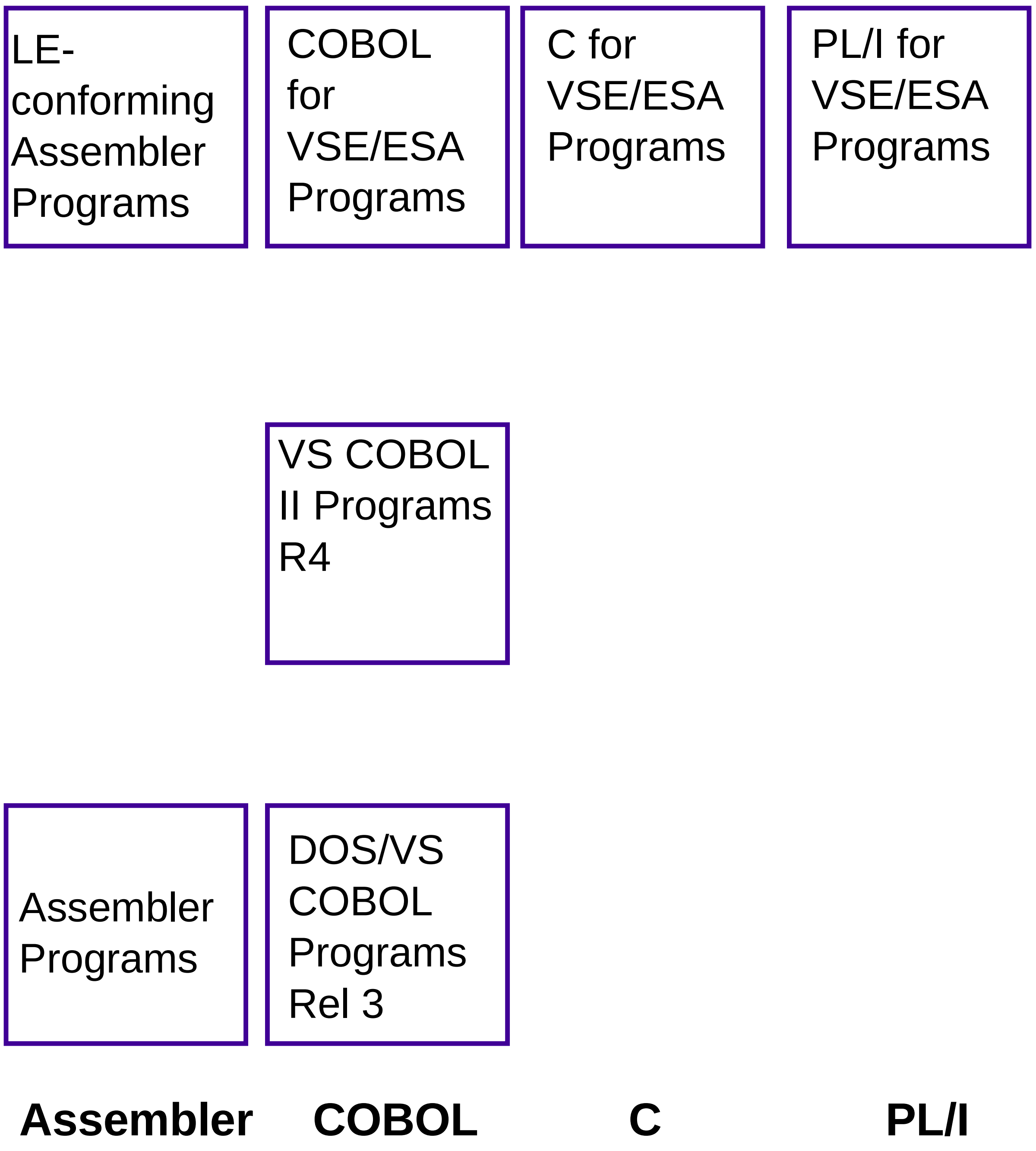
COBOL Product Relationships



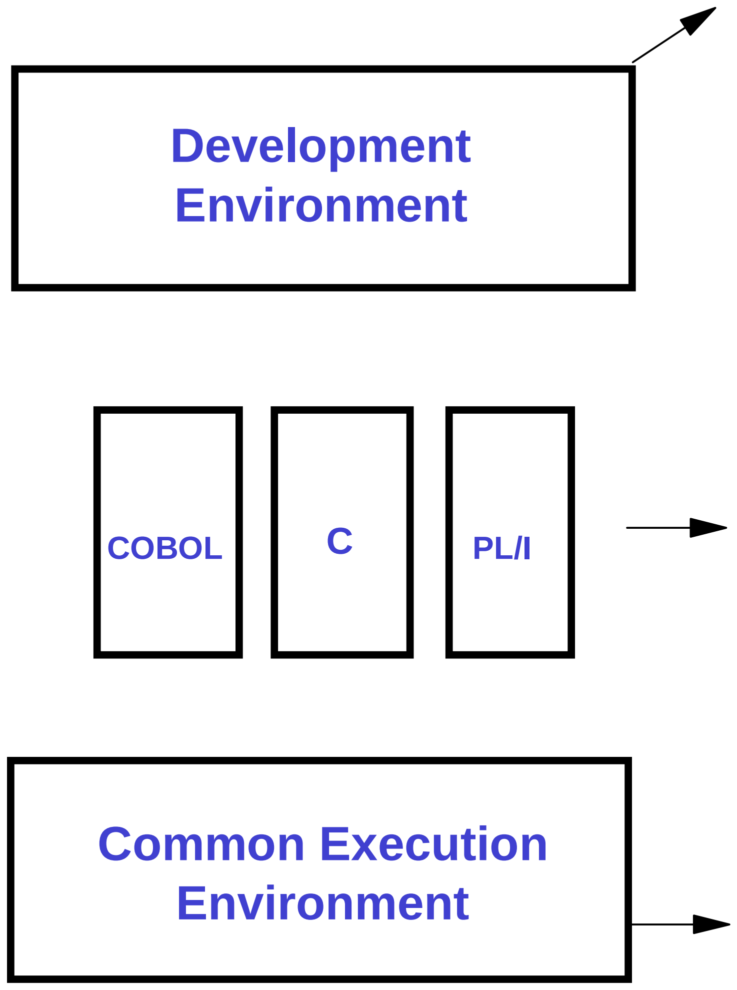
A History of COBOL Run-Times



The complete picture ...


LE-conforming Assembler Programs
COBOL for VSE/ESA Programs
C for VSE/ESA Programs
PL/I for VSE/ESA Programs
VS COBOL II Programs R4
Assembler Programs
DOS/VS COBOL Programs Rel 3
Assembler
COBOL
C
PL/I

Application Development Environment



Debug Tool

- Consistent across languages
- Consistent across platforms
- Improved programmer productivity

Compilers

- Functional language enhancements and implementation

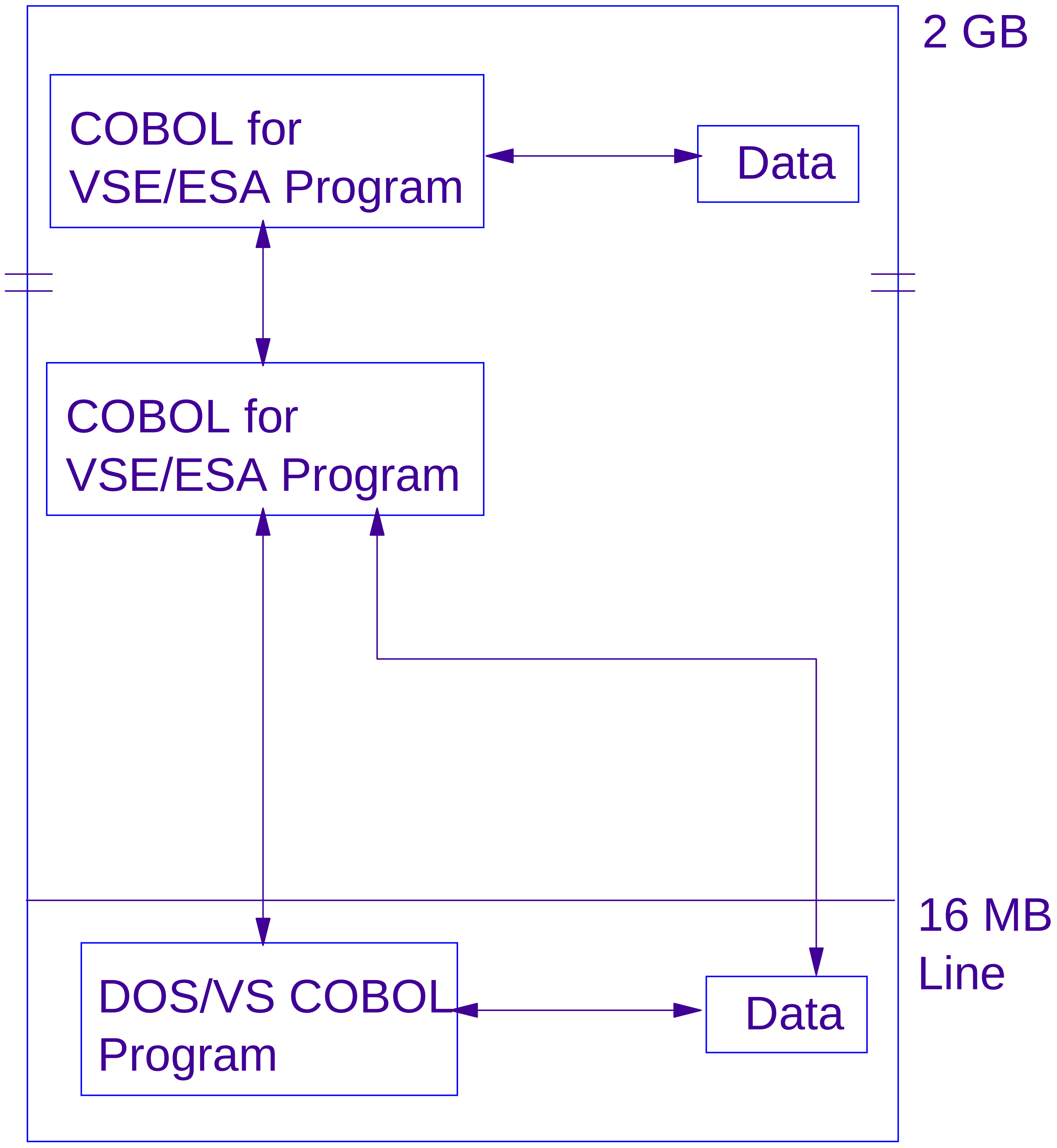
Language Environment

- Common run-time environment
- Language specific run-time services
- Enables assembly of applications
- Base for languages

Support of Features Introduced by VS COBOL II

- **ESA support, 31-bit addressing**
- **Reentrant object code**
- **Much improved CICS interface (no more BLL cell manipulation)**
- **Structured programming constructs**
  - **Review of 1985 Standard features**
- **And more!**

ESA Support, 31-Bit Addressing



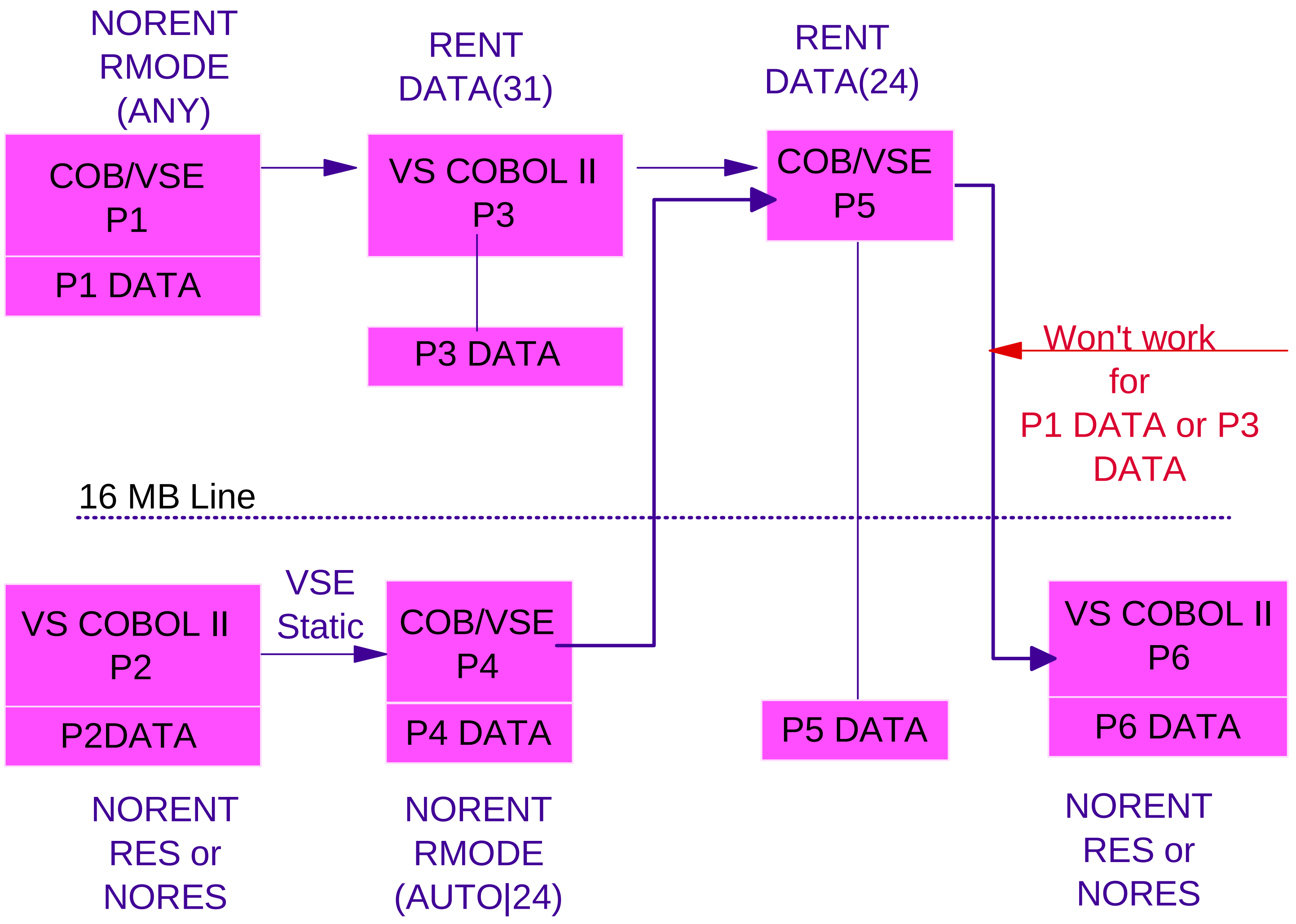
Options for Application Growth

COBOL for VSE/ESA compiler options for 31-bit addressing

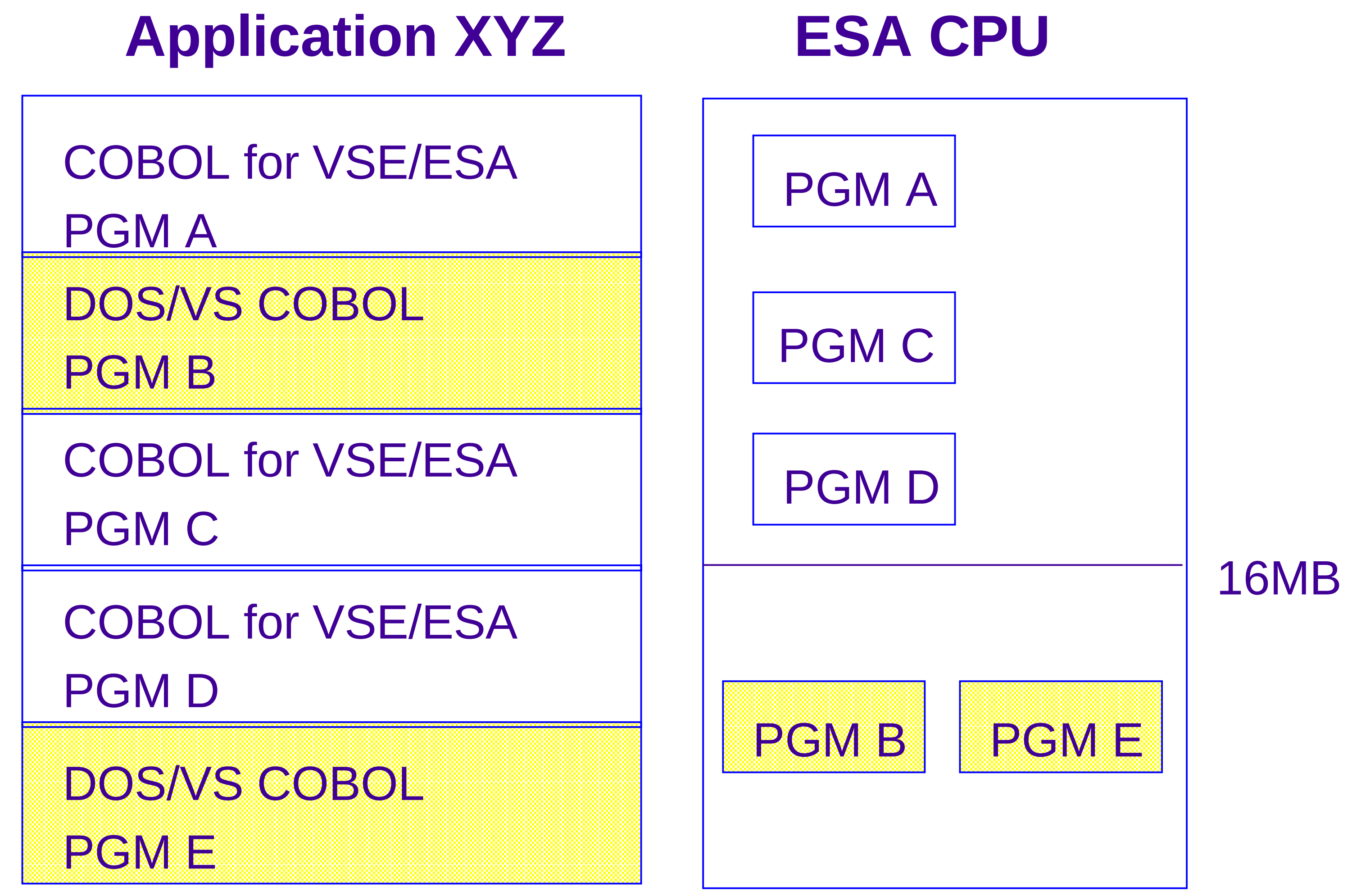
To get this:	**Code this using VS COBOL II**	**Or this using COBOL for VSE/ESA**
WORKING-STORAGE below the line	NORES,NORENT or RENT,DATA(24)	RENT, DATA(24) or NORENT with RMODE(AUTO or 24)
WORKING-STORAGE above the line	RES,RENT, DATA(31)	RENT, DATA(31) or NORENT, RMODE(ANY)
AMODE(24)	NORES,NORENT or RES,NORENT	N/A
AMODE(ANY)	RES,RENT or RES, NORENT	always
RMODE(24)	NORENT	RMODE(24) or NORENT,RMODE(AUTO)
RMODE(ANY)	RES,RENT	RMODE(ANY) or RENT,RMODE(AUTO)

ESA Support, 31-Bit Addressing

Example: Mixing new and old (all calls DYNAMIC)



ESA Support, 31-Bit Addressing ...



REENTRANT code

- Controlled by RENT compiler option
- Load module is 'read-only'
- Programs can be preloaded into SVA (Share Virtual Area)
  - or other read-only areas
- WORKING-STORAGE section is not in the load module
  - Memory is dynamically acquired

CICS Support

- Designed interface
- Reentrant code
- Elimination of BLLs and SERVICE RELOADs
- Subprograms can contain CICS commands

Simplified COBOL-CICS Interface

COBOL-CICS Interface

DOS/VS COBOL

```
WORKING-STORAGE SECTION.
77    LRECL-REC1   PIC   S9(4) COMP.

LINKAGE SECTION.

01  BLLCELLS.
  02 FILLER              PIC S9(8) COMP.
  02 BLL-REC1A          PIC S9(8) COMP.
  02 BLL-REC1B          PIC S9(8) COMP.
  02 BLL-REC2           PIC S9(8) COMP.

01 REC-1.
  02 CTR                PIC S(4) COMP.

PROCEDURE DIVISION.

EXEC CICS READ UPDATE ...
  SET (BLL-REC1A)
  LENGTH (LRECL-REC1)
  END-EXEC.

SERVICE RELOAD REC-1.

IF LRECL-REC1 > 4096
  THEN ADD 4096 TO BLL-REC1A
GIVING BLL-REC1B.

EXEC CICS REWRITE ...
  FROM (REC-1)
  LENGTH (LRECL-REC1)
  END-EXEC.
```

COBOL-CICS Interface

COBOL for VSE/ESA

```
LINKAGE SECTION.

01 REC-1.
  02 CTR                PIC S(4) COMP.

PROCEDURE DIVISION.

EXEC CICS READ UPDATE ...
  SET (ADDRESS OF REC1)
  END-EXEC.

EXEC CICS REWRITE ...
  FROM (REC-1)
  END-EXEC.
```

DB2 and SQL/DS

- Dynamic SQL statements
- Reentrant object code
- Example

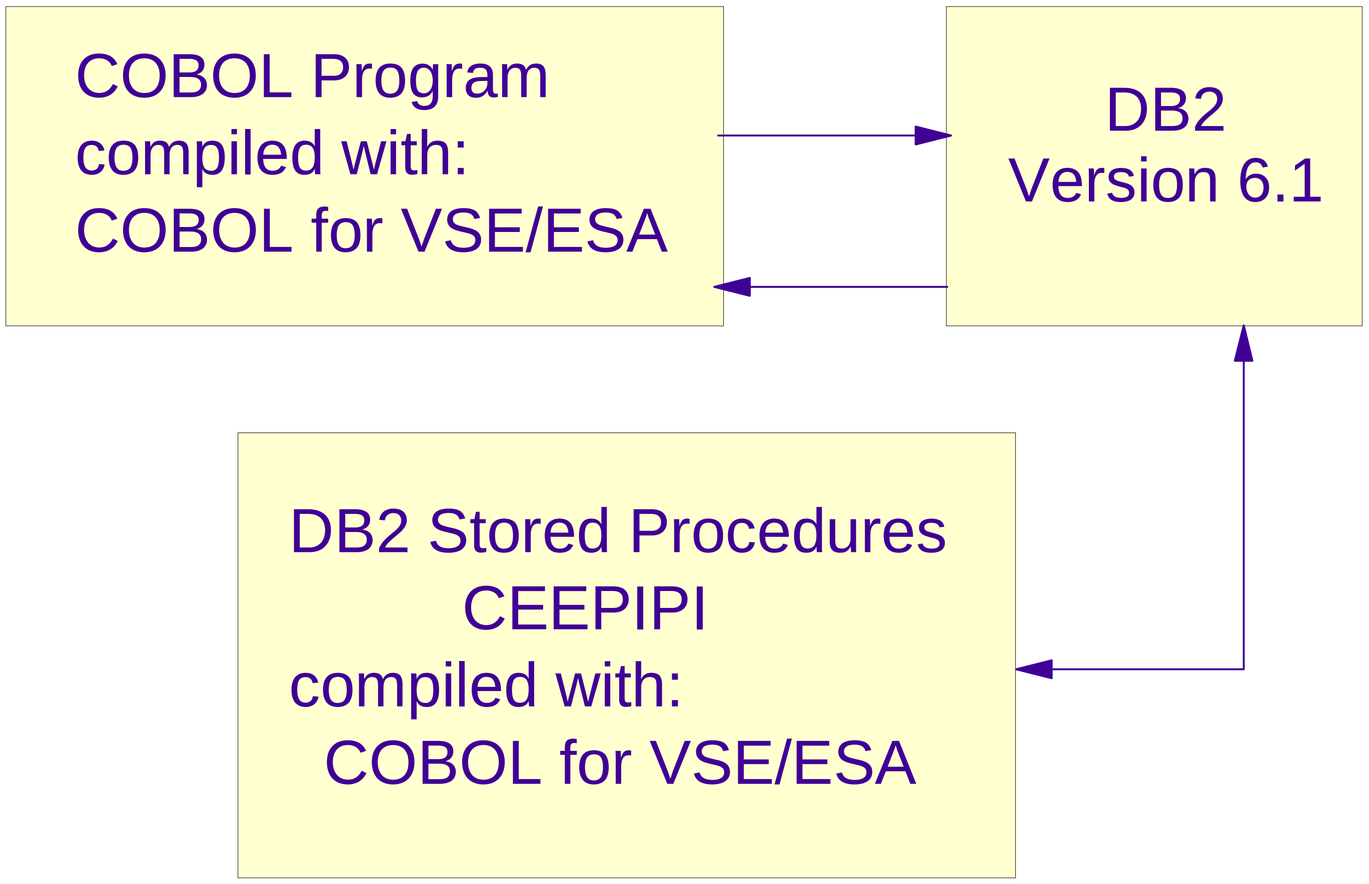
```
EXEC SQL INCLUDE SQLCA END-EXEC.
```

- DB2 Stored Procedures Support

DB2 Stored Procedure Support

DB2 Stored Procedures must be compiled with COBOL for VSE/ESA, PL/I for VSE/ESA, or C for VSE/ESA

DB2 Data calls that invoke DB2 Stored Procedures can be called from a COBOL for VSE/ESA program


COBOL Program compiled with: COBOL for VSE/ESA
DB2 Version 6.1
DB2 Stored Procedures CEEPIPI compiled with: COBOL for VSE/ESA

SC09-2662-00: DB2 Server for VSE Application Programming Version 6 Release 1, Appendix C. Using Stored Procedures

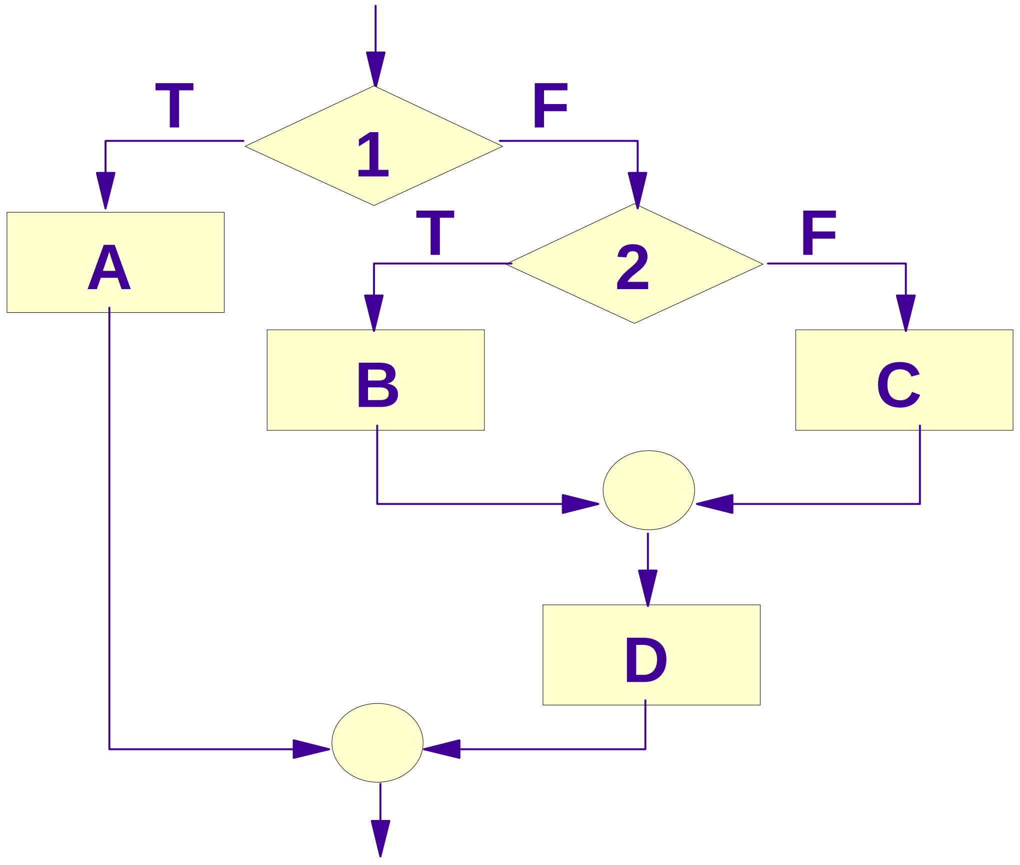
Review of 1985 Standard features

- **Prime control structures**
  - **Sequence**
  - **Selection**
  - **Iteration**
- **Explicit scope terminators**
  - **Example: END-IF**
- **Conditional Statements**
  - **AT END**
  - **NOT AT END**
- **EVALUATE and CONTINUE statements**
- **In-line PERFORMs**
- **Nested programs**

Prime Control Structures

- Sequence
- Selection
  - IF THEN ELSE
  - EVALUATE
- Iteration
  - DO WHILE
  - DO UNTIL

IF Control Structure

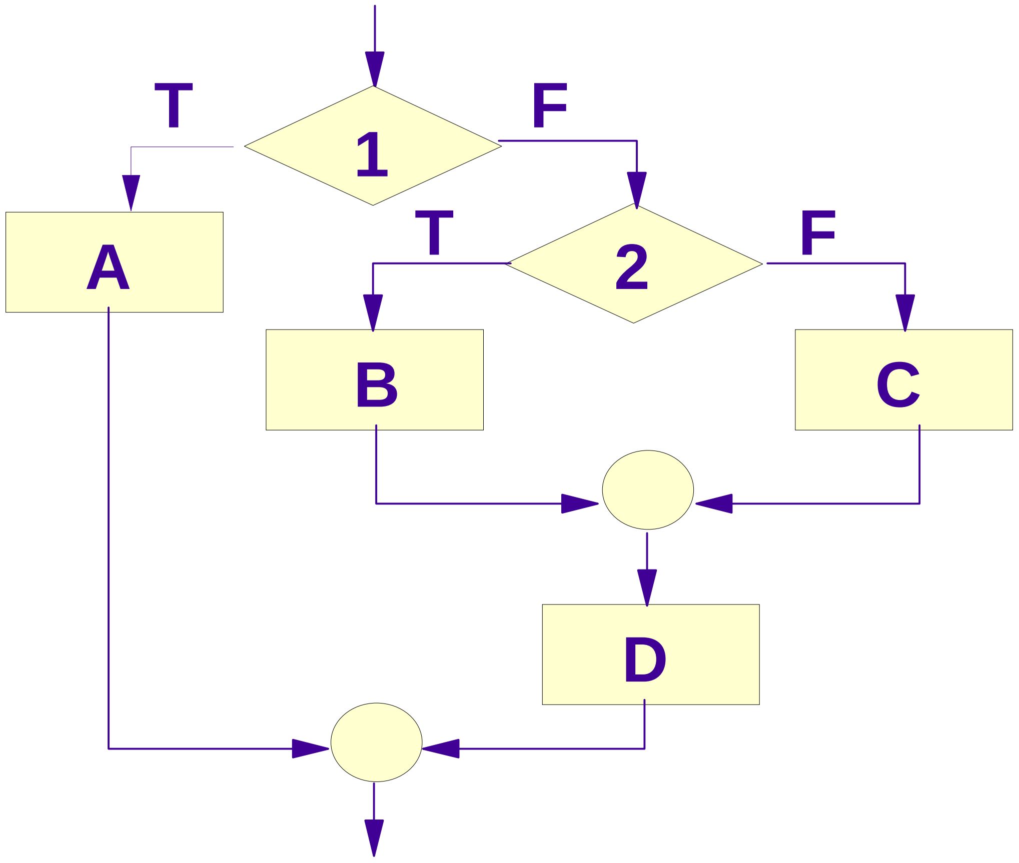


How to implement in 68/74 Std COBOL:

```
IF condition-1 THEN
    action-A
ELSE
    IF condition-2 THEN
        action-B
    ELSE
        action-C
... Now what?
```

IF Control Structure ...

Try again ...

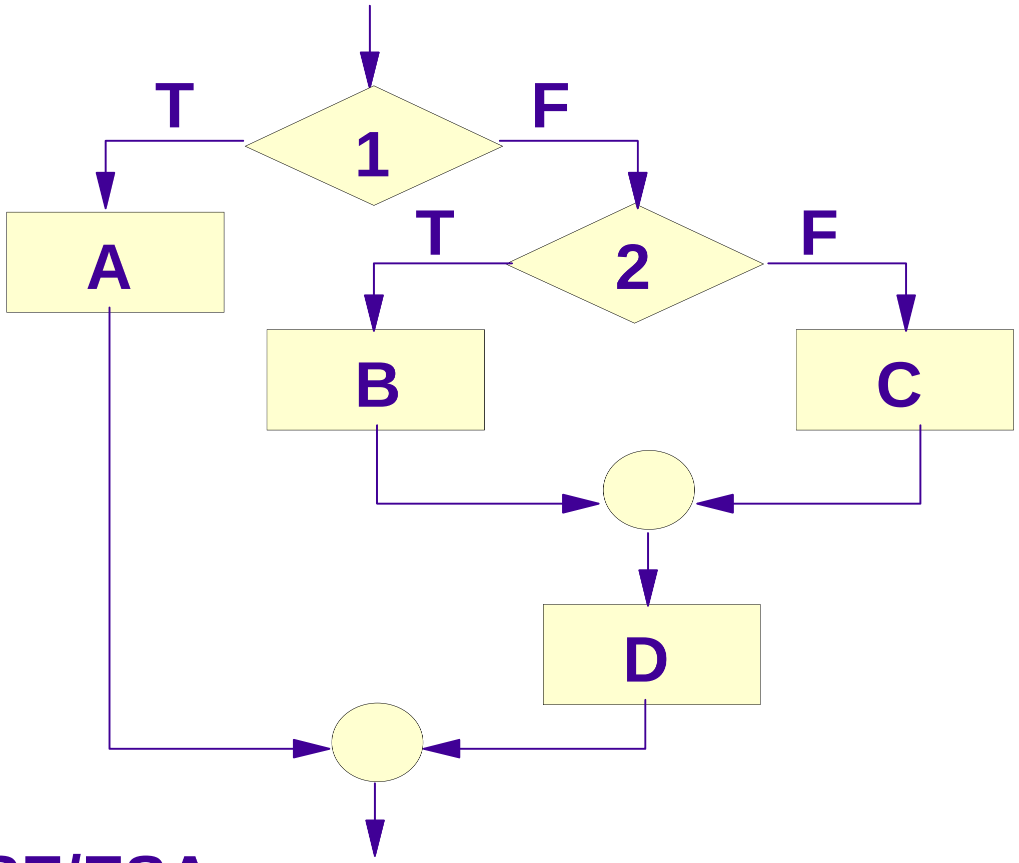


In 68/74 Std COBOL:

```
IF condition-1 THEN
    action-A
ELSE
    PERFORM ANOTHER-TEST.
GO TO AROUND-THE-TEST.
ANOTHER-TEST.
     IF condition-2 THEN
        action-B
    ELSE
        action-C.
     action-D.
AROUND-THE-TEST.
```

IF Control Structure ...

New and improved ...



In COBOL for VSE/ESA:

```
IF condition-1 THEN
    action-A
ELSE
    IF condition-2 THEN
        action-B
    ELSE
        action-C
    END-IF          <----  terminates
    action-D               previous IF
END-IF
```

CONTINUE vs NEXT SENTENCE

68/74/85 Std:	NEXT SENTENCE
85 Std:	CONTINUE

```
If aField = "ABC"
   Search All aTable
      At End
          Continue
      When aElem (Ind) = 1
          Next Sentence
Else
    Perform Not-ABC
End-If
Display "In same sentence, "
               "but in next statement"
  .
Display "In next sentence"
  .
```

AT END/NOT AT END

Replaces IF statements

Implementation with IF statements:

```
READ INPUT-FILE
   AT END
      MOVE 'EOF' TO EOF-FLAG.
IF NOT EOF THEN
   PERFORM PROCESS-INPUT-DATA
ELSE
   PERFORM EXIT-PARA.
```

Implementation with NOT AT END:

```
READ INPUT-FILE
   AT END
      PERFORM EXIT-PARA
   NOT AT END
      PERFORM PROCESS-INPUT-DATA
END-READ
```

EVALUATE Statement

Replaces several IF statements

Implementation with IF statements:

```
IF CARPOOL-SIZE = 1 THEN
    MOVE "SINGLE" TO CARPOOL-STATUS
ELSE
    IF CARPOOL-SIZE = 2 THEN
      MOVE "COUPLE" TO CARPOOL-STATUS
    ELSE
         IF CARPOOL-SIZE > 2 AND CARPOOL-SIZE < 7 THEN
            MOVE "SMALL GROUP" TO CARPOOL-STATUS
         ELSE
            IF  CARPOOL-SIZE = 7 OR
                  CARPOOL-SIZE = 8 THEN
               MOVE "MEDIUM GROUP" TO CARPOOL-STATUS
            ELSE
               MOVE "LARGE GROUP" to CARPOOL-STATUS.
```

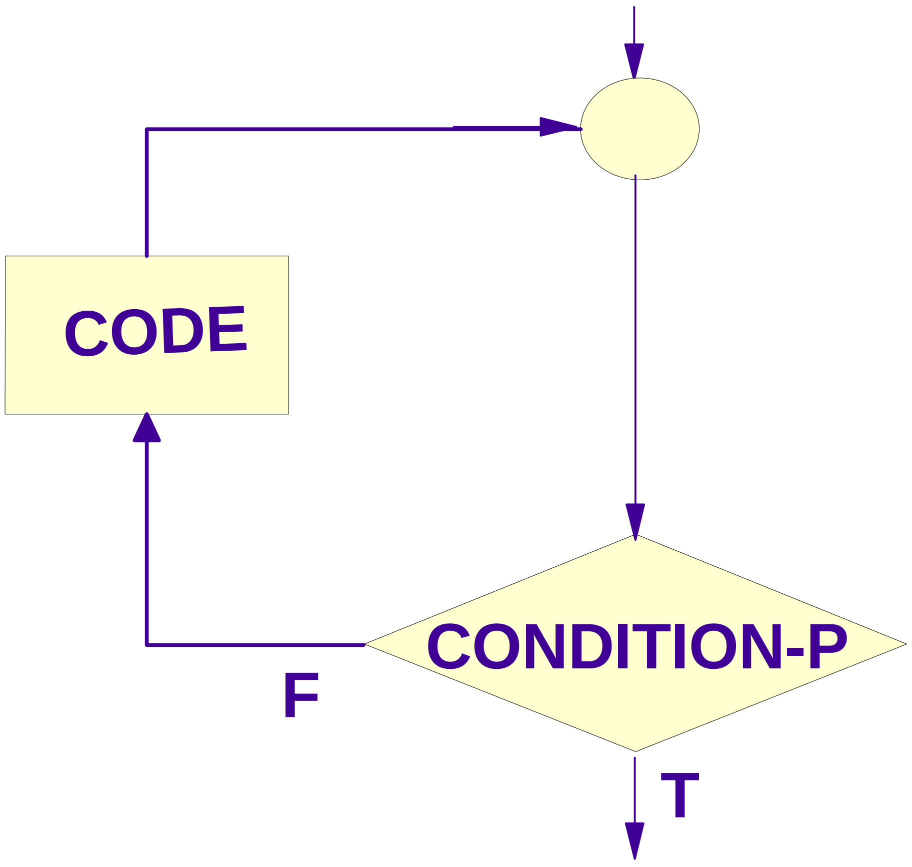
Corresponding Evaluate Statement:

```
EVALUATE CARPOOL-SIZE
     WHEN 1
           MOVE "SINGLE" TO CARPOOL-STATUS
     WHEN 2
           MOVE "COUPLE" TO CARPOOL-STATUS
     WHEN 3 THRU 6
           MOVE "SMALL GROUP" TO CARPOOL-STATUS
     WHEN 7 THRU 8
           MOVE "MEDIUM GROUP" TO CARPOOL-STATUS
     WHEN OTHER
           MOVE "LARGE GROUP" to CARPOOL-STATUS
END-EVALUATE.
```

EVALUATE Statement

```
EVALUATE TRUE ALSO FALSE ALSO TRUE
   WHEN X=1 ALSO Y=3 ALSO Z<4
      PERFORM
          ** stuff **
      END-PERFORM
   WHEN X=2 ALSO ANY ALSO ANY
    WHEN OTHER
       DISPLAY 'error!'
END-EVALUATE



EVALUATE  NUM
 WHEN 1, 2, 3
       PERFORM
          ** stuff **
      END-PERFORM

  WHEN 4 THRU 5 WHEN 8
      CALL 'SUB2'

  WHEN OTHER
      GOBACK
END-EVALUATE
```

DO WHILE Control Structure



68/74 Std COBOL

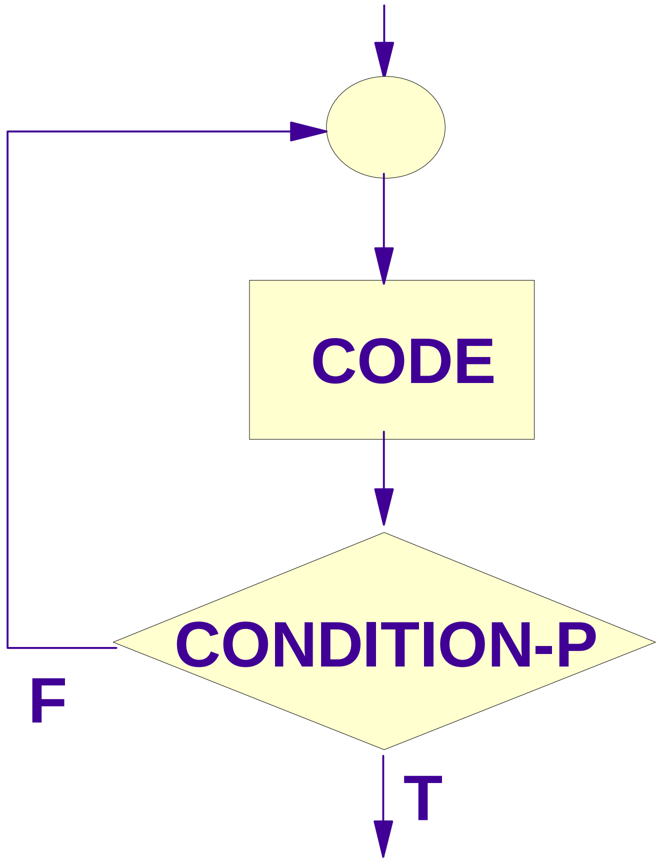
```
PERFORM PARA-NAME
    UNTIL P
```

```
PARA-NAME.
      CODE
```

COBOL for VSE/ESA

```
PERFORM WITH TEST
    BEFORE UNTIL P

          CODE

END-PERFORM.
```

DO UNTIL Control Structure



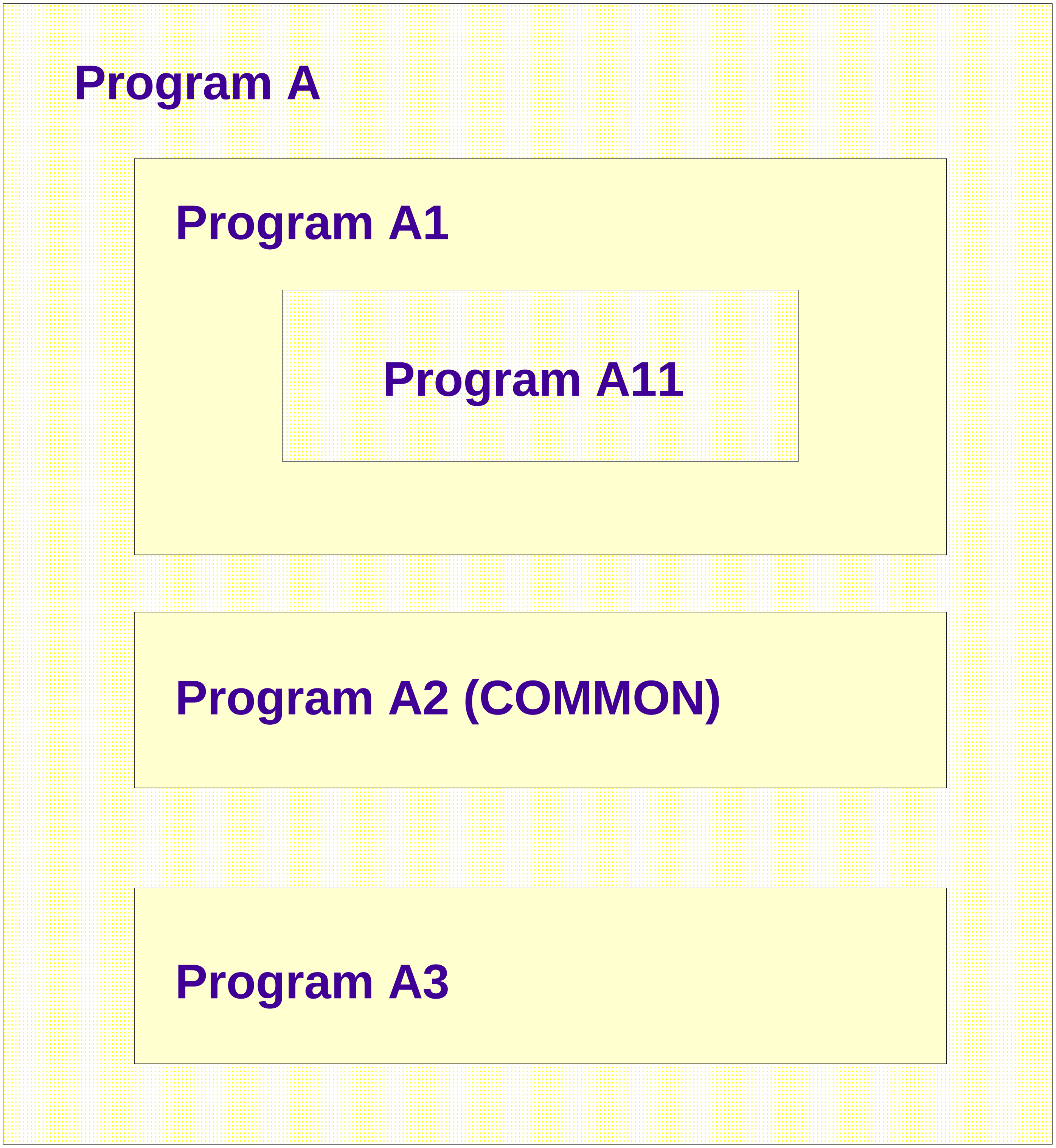
68/74 Std COBOL

```
PERFORM PARA-NAME
PERFORM PARA-NAME
   UNTIL P
.....................................

PARA-NAME.
     CODE
```

COBOL for VSE/ESA

```
PERFORM WITH TEST
    AFTER UNTIL P

          CODE

END-PERFORM.
```

Nested Programs



Other VS COBOL II Features (Review)

- INITIALIZE statement
- SET TO TRUE
- LENGTH OF special register
- POINTER data items
- Reference modification (substring handling)
- Hexadecimal Literals

INITIALIZE Example

Example:

```
01  CUSTOMER-RECORD.
    02  CUST-NUMBER          PIC 9(6).
    02  LAST-NAME            PIC X(15).
    02  FIRST-NAME           PIC X(15).
    02  DATABASE-ID          PIC S9(9) BINARY.
    02  INFO-PTR             POINTER.

*With INITIALIZE:

 INITIALIZE CUSTOMER-RECORD.

*Without INITIALIZE:

  MOVE ZEROES TO CUST-NUMBER.
  MOVE SPACES TO LAST-NAME FIRST-NAME.
  COMPUTE DATABASE-ID = 0.
  SET INFO-PTR TO NULL.
```

SET...TO TRUE Example

Example (How to replace READY TRACE!):

```
WORKING-STORAGE SECTION.
    01  TRACE-VAR PIC X.
       88 READY-TRACE  VALUE 'Y'.
       88 RESET-TRACE  VALUE 'N'.

PROCEDURE DIVISION.
  TRACE-IT SECTION USE FOR DEBUGGING
                  ON ALL PROCEDURES.
      IF READY-TRACE THEN
          DISPLAY 'Trace:' DEBUG-NAME DEBUG-LINE
      END-IF
  MAIN SECTION.

  MID SECTION.
      SET READY-TRACE TO TRUE.
  PARA1.
      ...
  PARA2.
      ...
      SET RESET-TRACE TO TRUE
  END1 SECTION.
```

LENGTH OF special register example

Example:

```
    01  CUST-REC.
      02  CUST-NUMBER          PIC 9(6).
      02  LAST-NAME            PIC X(15).
      02  FIRST-NAME           PIC X(15).
      02  DATABASE-ID          PIC S9(9) BINARY.
      02  INFO-PTR             POINTER.
    77  X                      PIC 9(4).

 PROCEDURE DIVISION.

       COMPUTE X = LENGTH OF CUST-REC.

       CALL 'SUB1' USING
                    BY VALUE LENGTH OF CUST-REC
```

POINTER data items example

Example:

```
WORKING-STORAGE SECTION.

 77  COND-DATA-PTR   USAGE POINTER.


 LINKAGE SECTION.
  01 COND-DATA.
     02 MSGNO     PIC X(4).
     02 ERRCOUNT  PIC 9(4).

PROCEDURE DIVISION USING COND-DATA.


    SET COND-DATA-PTR TO
          ADDRESS OF COND-DATA.

   CALL 'ASM-SUB' USING COND-DATA-PTR.
```

Reference Modification Examples

Example 1:

```
01  WHOLE-NAME              PIC X(25).
01  LAST-NAME               PIC X(25).

MOVE WHOLE-NAME(10:15)  to LAST-NAME
```

Example 2:   HHMMSSss ---->HH:MM:SS

```
01  TIME-ITEM               PIC X(8).

ACCEPT TIME-ITEM FROM TIME

DISPLAY "CURRENT TIME IS: "
    TIME-ITEM(1:2)
     ":"
    TIME-ITEM(3:2)
     ":"
    TIME-ITEM(5:2)
```

Hexadecimal Literal Example

Example:

```
WORKING-STORAGE SECTION.
  01 PD-GROUPA.
    2  S99PDA PIC S99 PACKED-DECIMAL.
  01 PD-GROUPB.
    2  S99PDB PIC S99 PACKED-DECIMAL.
  77 ALPHA  PIC X(30) VALUE X'0A0B0C0DEEFF' .

PROCEDURE DIVISION.

    MOVE X'015F' TO PD-GROUPA.

    IF S99PDA IS NUMERIC THEN
        DISPLAY 'S99PDA is numeric with hex F'
    ELSE
        DISPLAY 'S99PDA is NOT numeric with hex F'
    END-IF

    IF PD-GROUPA = X'015F' THEN
        DISPLAY  ' S99PDA still has hex F sign'
    ELSE
        DISPLAY  ' S99PDA does NOT have hex F sign'
    END-IF
```

New Language Features with COBOL for VSE/ESA

- **New CALL options**
- **Intrinsic Functions**
  - Mathematical
  - Statistical
  - Date/Time
  - Financial
  - Character Handling
  - General
- **Support for Language Environment callable services**
- **Support for Language Environment condition handling**
  - PROCEDURE-POINTER data type
- **New Date/Time features**
- **QUOTES and APOSTROPHES**
- **New compiler options**

New CALL Options

Example:

```
   01  RETURN-VALUE  PIC 9(4).
   01  X                          PIC 9(9) BINARY.
PROCEDURE DIVISION
             USING BY VALUE X
             RETURNING RETURN-VALUE.

      COMPUTE X = 1234567.

      CALL 'SUB1' USING BY VALUE X
                   RETURNING RETURN-VALUE.

      IF RETURN-VALUE = 0 THEN
          DISPLAY 'SUB1 Was successful'
      ELSE
           DISPLAY 'SUB1 had a problem'
      END-IF
```

Intrinsic Functions

- Amendment to 1985 ANSI COBOL Standard
- Minimize program size
- Perform these tasks:
  - Mathematical
  - Statistical
  - Date/Time
  - Financial
  - Character Handling
  - General

Intrinsic Functions ...

Mathematical	Statistical	Date/Time
ACOS	MEAN	CURRENT-DATE
ASIN	MEDIAN	DATE-OF-INTEGER
ATAN	MIDRANGE	DATE-TO-YYYYMMDD
COS	RANDOM	DATEVAL
FACTORIAL	RANGE	DAY-OF-INTEGER
INTEGER	STANDARD-DEVIATION	DAY-TO-YYYYDDD
INTEGER-PART	VARIANCE	INTEGER-OF-DATE
LOG		INTEGER-OF-DAY
LOG10		WHEN-COMPILED
MOD		YEAR-TO-YYYY
REM		YEARWINDOW
SIN		
SQRT		
SUM		
TAN		

Financial	Character Handling	General
ANNUITY	CHAR	LENGTH
PRESENT-VALUE	LOWER-CASE	MAX
	NUMVAL	MIN
	NUMVAL-C	ORD-MAX
	ORD	ORD-MIN
	REVERSE	
	UPPER-CASE	

Intrinsic Function Rules

1. A numeric function is a numeric expression
2. Numeric functions cannot be used in MOVE statements
3. Functions cannot be used as subscripts
4. The type of some functions is determined by the arguments

```
COMPUTE  PIC9 =  FUNCTION MAX ( 1 2 3 ).

MOVE FUNCTION MAX ( 'A' "B" 'C') TO PICX.
```

Intrinsic Function Examples

Example 1:

```
01  X                    PIC 99.
01  Y                    PIC 99.
01  Z                    PIC 99.

COMPUTE X = FUNCTION MAX(X Y Z)

IF  X = FUNCTION MAX(X Y Z) THEN
  ...
```

Example 2:

```
COMPUTE  Z =
  FUNCTION LOG(FUNCTION FACTORIAL(2 * X + 1))
```

Date/Time Intrinsic Functions

Facilitate date/time arithmetic

```
* Calculate due date 90 days from today

01  YYYYMMDD           PIC 9(8) DATE FORMAT YYYYXXXX..
01   I                 PIC S9(9) BINARY.
     .
     .
MOVE FUNCTION CURRENT-DATE(1:8) TO YYYYMMDD
COMPUTE  I = FUNCTION INTEGER-OF-DATE(YYYYMMDD)
ADD 90 to I
COMPUTE YYYYMMDD = FUNCTION DATE-OF-INTEGER(I)
DISPLAY "DUE DATE: "  YYYYMMDD

* Can also nest them!
COMPUTE YYYYMMDD = FUNCTION DATE-OF-INTEGER
       ( 90 +  FUNCTION INTEGER-OF-DATE
            (FUNCTION INTEGER
                 (FUNCTION NUMVAL
                        (FUNCTION CURRENT-DATE(1:8))
                 )
            )
       )
 DISPLAY "DUE DATE: "  YYYYMMDD
```

ALL subscript specifies all elements of a table or table dimension:

```
01  Employee-table.
    05   Emp-count                          PIC S9(4).
    05   Emp-record OCCURS  1 to 500  TIMES.
         10  Emp-name                       PIC X(20).
         10  Emp-id                         PIC 9(9).
         10  Emp-salary                     PIC 9(7)V99.

COMPUTE max-salary = FUNCTION MAX(Emp-salary (ALL))
COMPUTE  I = FUNCTION ORD-MAX(Emp-salary  (ALL))
COMPUTE Avg-salary = FUNCTION MEAN(Emp-salary (ALL))
COMPUTE Salary-range = FUNCTION RANGE(Emp-salary(ALL))
COMPUTE Total-payroll = FUNCTION SUM(Emp-salary (ALL)

DISPLAY
   "Highest paid employee: " Emp-name(I)
   "Maximum salary          " Max-salary
   "Average salary          " Avg-salary
   "Salary range            "  Salary-range
   "Total Payroll           " Total-payroll.
```

LE-Callable Services Example

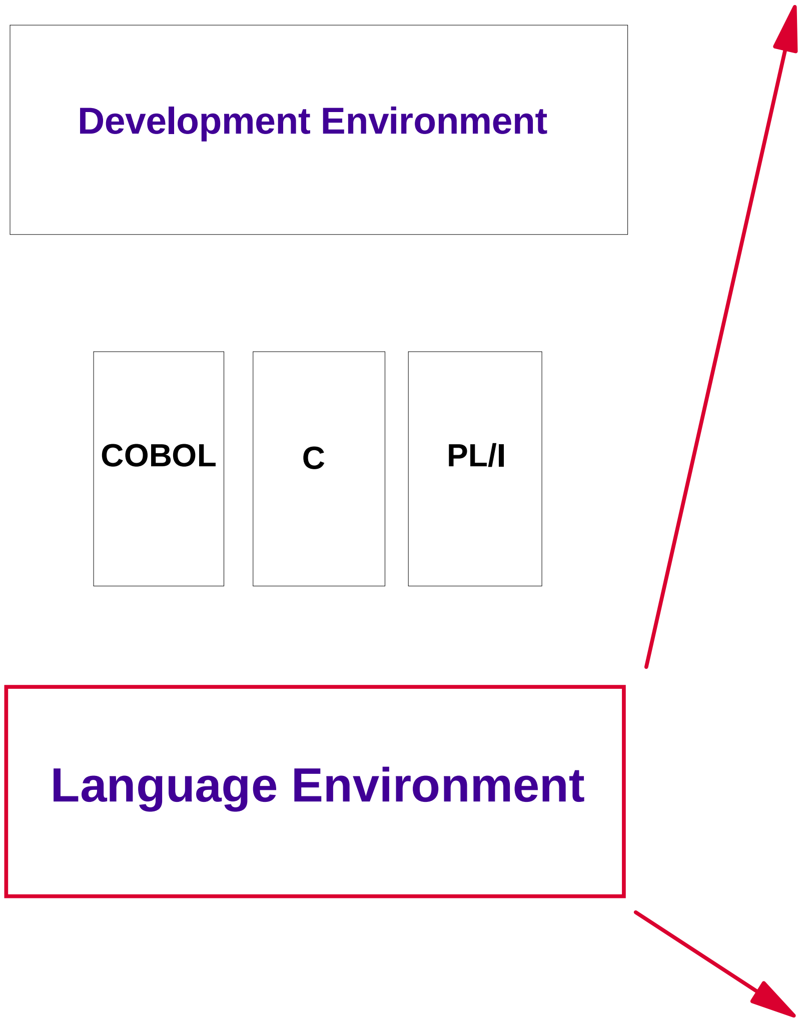
Example:

```
WORKING-STORAGE SECTION.
 01  PGMPTR   USAGE PROCEDURE-POINTER.
 77  FC              PIC X(12).
 77  ADDRSS   USAGE POINTER.
 77  HEAPID          PIC 9(9) BINARY.
 77  STGSIZE         PIC 9(9) BINARY.

 LINKAGE SECTION.
  01 COND-DATA.
     02 MSGNO     PIC X(4).
     02 ERRCOUNT  PIC 9(4).

PROCEDURE DIVISION.

    MOVE 0 TO HEAPID.
    MOVE LENGTH OF COND-DATA TO STGSIZE.
    CALL "CEEGTST" USING HEAPID, STGSIZE,
                                   ADDRSS, FC.

    SET ADDRESS OF COND-DATA TO ADDRSS.
```

PROCEDURE-POINTER Data Type

- COBOL for VSE/ESA extension to 1985 ANSI COBOL Standard
- USAGE IS PROCEDURE-POINTER
- Holds address of an entry point
- An 8-byte Language Environment entry variable
- Set with new format of SET statement
- SET PROC-PTR TO ENTRY 'SUB1'.

Functional Overview - Language Environment for VSE/ESA



- **Common Environment**
  - **Condition management**
  - **Memory management**
  - **Task management**
  - **Subsystem interface**
- **Common Protocols**
  - **Tasking**
  - **Linkage**
- **Common Services**
  - **Message**
  - **Dump**
- **Common Routines**
  - **Math**
  - **Callable services**
  - **Language runtimes**
- **Support for Debug Tool**
- **InterLanguage Communication**

A few benefits provided by a common run-time environment for COBOL for VSE/ESA:

- **Improved InterLanguage Communication (ILC)**
- **Callable services (via COBOL CALL statement)**
  - Storage management
  - Date and time calculations
  - Math calculations
  - Message handling
  - National language support
  - Other services such as formatted dumps
- **Common condition handling mechanism across languages**
- **Comprehensive run-time options**

Language Environment Condition Handling Objectives:

- **Predictable condition handling in applications**
  - single language or mixed language
- **Within a mixed application, honor each HLL's error handling semantics**
  - ie: ON SIZE ERROR, ON EXCEPTION, etc
- **Provide more capability for HLLs with limited built-in error handling**
  - Like COBOL!
- **Fault tolerant systems; crash protection**
  - only a truly catastrophic failure needs to disrupt your application environment
- **Enable new function, such as resumption after error occurs**
- **Allow error handlers to be written in COBOL**

Language Environment Condition Handling ...

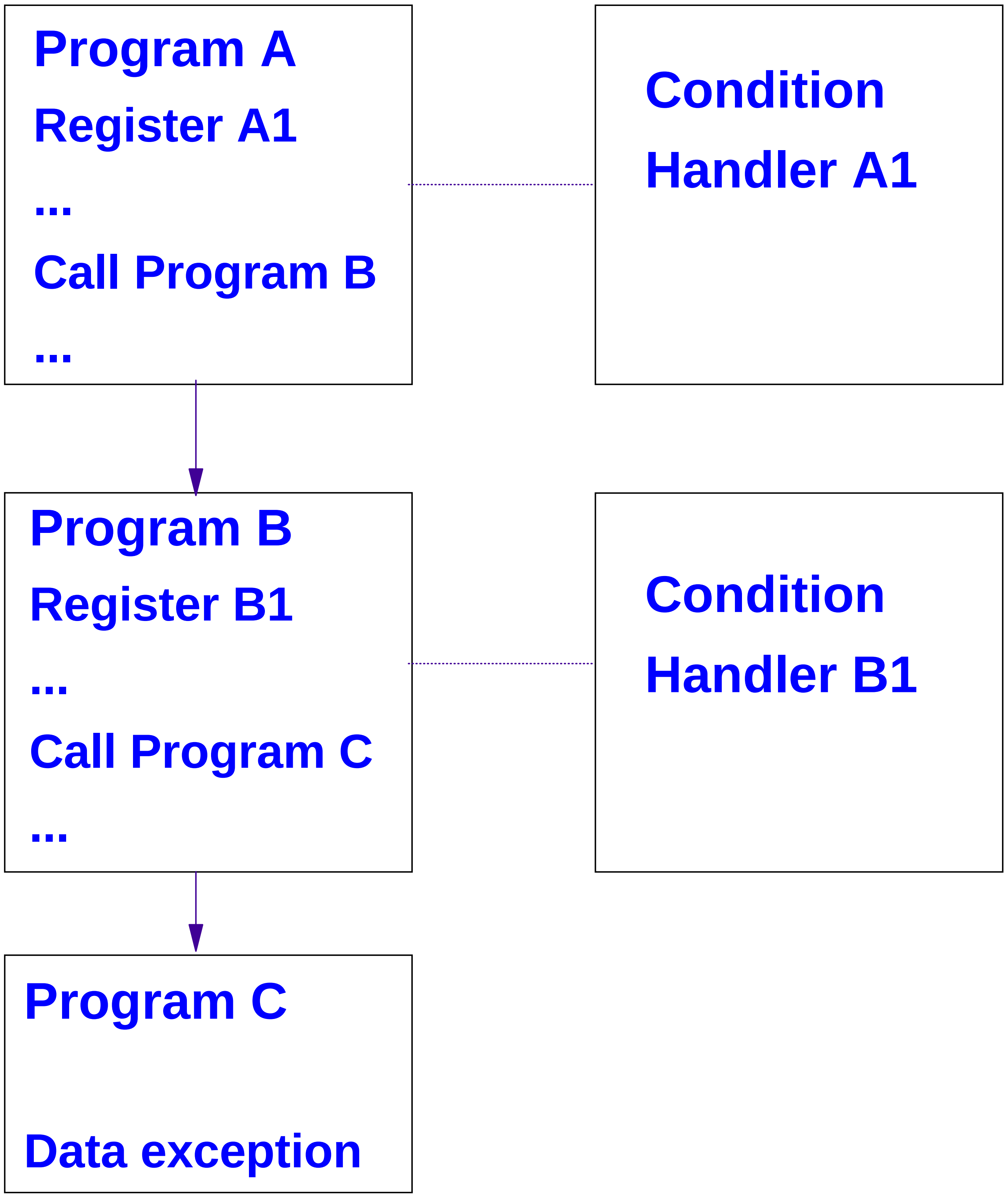
- **Some Concepts and Technology:**

- **Conditions include:**
  - Program interrupts
  - ABENDs
  - Software generated signals

- **Condition token is created**

- **User-written condition handler**
  - Is a separate program
  - Gets invoked when a condition occurs
  - Optional: Gathers information about the condition from the condition token
  - Causes these actions to be taken: resume, percolate, promote

Calling chain:



Language Environment Condition Handling ...

```
ID DIVISION.    PROGRAM-ID.   MAIN.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.      FILE-CONTROL.
    SELECT RUNDATA ASSIGN TO SYSIN-S-FILE1DD
      FILE STATUS IS RUNDATA-FS.
DATA DIVISION.
FILE SECTION.
FD  RUNDATA.
01  WORK-RECORD.
      03  CUST-NAME.
          05 FIRST-NAME PIC X(10).
          05 LAST-NAME  PIC X(15).
      03  BIRTH-DATE.
          05  MONTH    PIC 99.
          05           PIC X.
          05  DAYO     PIC 99.
          05           PIC X.
          05  YEAR     PIC 99.
      03  ACCOUNT-NUM   PIC 9(8).
      03  CURRENT-AGE   PIC 999.
      03  BENNY-FACTOR  PIC 9(3).
      03  ADJUSTMENT    PIC 9(3).
      03  PAYOUT        PIC X(6).
      03                PIC X(24).

WORKING-STORAGE SECTION.
01  RUNDATA-FS    PIC 99.
01  WORKING-DATA  PIC 99.
77  EOF-IND       PIC X.
 88  EOF          VALUE "Y".
 88  NOT-EOF      VALUE "N".
```

Language Environment Condition Handling ...

```
PROCEDURE DIVISION.

STARTIT. OPEN INPUT RUNDATA.
         IF RUNDATA-FS NOT EQUAL TO 0
           DISPLAY "** ERROR ** NOT ABLE TO OPEN"
                   "  RUNDATA FILE **"
           GO TO STOPIT
         END-IF
         SET NOT-EOF TO TRUE.

LOOP.
        PERFORM UNTIL EOF
            READ RUNDATA
                 AT END
                     SET EOF TO TRUE
                 NOT AT END
                     CALL "SUBRTN" USING WORK-RECORD
            END-READ

         END-PERFORM.

STOPIT.
        CLOSE RUNDATA.
         STOP RUN.

END PROGRAM MAIN.
```

Language Environment Condition Handling ...

```
ID DIVISION.    PROGRAM-ID. SUBRTN.
DATA DIVISION.
WORKING-STORAGE SECTION.
  01  CURR-DATE.
      05  CURR-YEAR    PIC 9(4).
      05  CURR-MONTH   PIC 99.
      05  CURR-DAY     PIC 99.
  01  PGMPTR   USAGE PROCEDURE-POINTER.
  01  DATA-PTR USAGE POINTER.
*****************************************************************
* Parameters for CEECBLDY                                       *
*****************************************************************
  01 PICSTR.
    05 PICSTR-LENGTH   PIC 9(2) COMP VALUE 8.
    05 PICSTR-STRING   PIC X(50)     VALUE "MM/DD/YY".
  01 INPUT-DATE.
    05 INPUT-LENGTH    PIC 9(2) BINARY.
    05 INPUT-STRING    PIC X(50).
  77 LILIAN            PIC 9(9) COMP.
  77  FC               PIC X(12).
```

Language Environment Condition Handling ...

```
******************************************************************
* Working Storage for DATE-OF-INTEGER
******************************************************************
 01 4-DIGIT-DATE PIC 9(8).
 01 4-DIGIT-REDEFINED REDEFINES 4-DIGIT-DATE.
   05 YYYY PIC 9(4).
   05 MM   PIC 9(2).
   05 DD   PIC 9(2).

 LINKAGE SECTION.
 01  WORK-RECORD.
     03  CUST-NAME.
         05 FIRST-NAME PIC X(10).
         05 LAST-NAME  PIC X(15).
     03  BIRTH-DATE.
         05  MONTH     PIC 99.
         05            PIC X.
         05  DAYO      PIC 99.
         05            PIC X.
         05  YEAR      PIC 99.
     03  ACCOUNT-NUM   PIC 9(8).
     03  CURRENT-AGE   PIC 999.
     03  BENNY-FACTOR  PIC 9(3).
     03  ADJUSTMENT    PIC 9(3).
     03  PAYOUT        PIC 9(6).
```

Language Environment Condition Handling ...

```
 PROCEDURE DIVISION USING WORK-RECORD.

****************************************************************
*  put address of common data area in TOKEN passed to
*  LE/370 condition manager.
****************************************************************
     SET DATA-PTR TO ADDRESS OF WORK-RECORD.
****************************************************************
*  put name and address of user-written condition
* handler into PROCEDURE-POINTER data item to
*  pass to condition manager.
****************************************************************
     SET PGMPTR TO ENTRY "USERHDLR".
****************************************************************
*  register the user-written condition handler with the
*  Language Environment condition manager.
****************************************************************
     CALL "CEEHDLR" USING PGMPTR DATA-PTR FC.
```

Language Environment Condition Handling ...

```
*****************************************************************
*  get todays date with 4-digit year.
*****************************************************************
     MOVE FUNCTION CURRENT-DATE(1:8) TO CURR-DATE.

*****************************************************************
*  convert 2-digit year from file into integer using LE/370
*  service to get COBOL Lilian date.
*****************************************************************
     MOVE LENGTH OF BIRTH-DATE TO INPUT-LENGTH.
     MOVE BIRTH-DATE TO INPUT-STRING.
     CALL "CEECBLDY" USING INPUT-DATE PICSTR LILIAN FC.

*****************************************************************
*  convert COBOL Lilian date into YYYYMMDD format.
*****************************************************************
     COMPUTE 4-DIGIT-DATE = FUNCTION
                             DATE-OF-INTEGER(LILIAN).
     DISPLAY ">>>Birth Year = " YYYY.

     COMPUTE CURRENT-AGE = CURR-YEAR - YYYY.

*****************************************************************
*  this is the statement likely to fail with bad data.
*****************************************************************
     COMPUTE PAYOUT =
          (CURRENT-AGE * BENNY-FACTOR) / ADJUSTMENT.

     GOBACK.
 END PROGRAM SUBRTN.
```

Language Environment Condition Handling ...

```
ID DIVISION.   PROGRAM-ID. USERHDLR.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
   01  TYPE-0   PIC S9(9) USAGE BINARY VALUE ZERO.
   01  TYPE-1   PIC S9(9) USAGE BINARY VALUE 1.
   01  FC.
      05  SEVERITY PIC 9(4) USAGE BINARY.
      05  MSGNO    PIC 9(4) USAGE BINARY.
      05  FILLER   PIC X(8).
   01  FAILING-OFST PIC S9(9) USAGE BINARY.

 LINKAGE SECTION.
   01  WORK-RECORD.
       03  CUST-NAME.
           05 FIRST-NAME PIC X(10).
           05 LAST-NAME  PIC X(15).
       03  BIRTH-DATE.
           05  MONTH     PIC 99.
           05            PIC X.
           05  DAYO      PIC 99.
           05            PIC X.
           05  YEAR      PIC 99.
       03  ACCOUNT-NUM   PIC 9(8).
       03  CURRENT-AGE   PIC 999.
       03  BENNY-FACTOR  PIC 9(3).
       03  ADJUSTMENT    PIC 9(3).
       03  PAYOUT        PIC 9(6).
       03                PIC X(24).
   01  CURRENT-CONDITION   PIC X(12).
   01  DATA-PTR USAGE POINTER.
   01  RESULT-CODE  PIC S9(9) USAGE BINARY.
   01  NEW-CONDITION       PIC X(12).
```

```
PROCEDURE DIVISION USING CURRENT-CONDITION
                         DATA-PTR
                         RESULT-CODE
                         NEW-CONDITION.

*****************************************************************
*  get addressability to common data area.                      *
*****************************************************************
    SET ADDRESS OF WORK-RECORD TO DATA-PTR.

*    DISPLAY "***In Userhdlr, WORK-RECORD = "
          WORK-RECORD.

*****************************************************************
*  find out which field(s) had bad data.                        *
*****************************************************************
    IF YEAR NOT NUMERIC THEN
       DISPLAY "    Bad data in year field"
       COMPUTE YEAR = 0
    END-IF
    IF BENNY-FACTOR NOT NUMERIC THEN
       DISPLAY "    Bad data in benefits factor field"
       COMPUTE BENNY-FACTOR = 1
    END-IF
    IF ADJUSTMENT NOT NUMERIC THEN
       DISPLAY "    Bad data in adjustment field"
       COMPUTE ADJUSTMENT = 1
    END-IF
*    DISPLAY "***After error checking WORK-RECORD = "
            WORK-RECORD.
```

Language Environment Condition Handling ...

```
*****************************************************************
*  put out message indicating which record was bad.
*****************************************************************
     DISPLAY "  Bad data in record with account number:"
              ACCOUNT-NUM.
     DISPLAY "    and customer name: " CUST-NAME.
     COMPUTE PAYOUT = 0.
*****************************************************************
*    Retrieve the offset of the error
*****************************************************************
     CALL "CEE3GRO" USING FAILING-OFST, FC.
     IF SEVERITY > 0 THEN
        DISPLAY "CALL to CEE3GRO failed with Severity = "
                SEVERITY  '  and message number   = " MSGNO
        GOBACK
     END-IF.
     DISPLAY "Offset of error is " FAILING-OFST.
*****************************************************************
*  resume execution at MAIN, process next record.
*****************************************************************
     CALL "CEEMRCR" USING TYPE-1 FC.
     IF SEVERITY > 0 THEN
        DISPLAY "CALL to CEEMRCR failed with Severity = "
                SEVERITY  '  and message number   = " MSGNO
     END-IF

*****************************************************************
*  mark the condition as handled
*****************************************************************
     COMPUTE RESULT-CODE = 10.

     GOBACK.
 END PROGRAM USERHDLR.
```

New DATE/TIME Features

- MLE: Millennium Language Extensions

```
77 YYMMDD    PIC 9(6)
             DATE FORMAT  YYXXXX.
77 YYYYMMDD PIC 9(8)
             DATE FORMAT  YYYYXXXX.

IF YYMMDD > YYYYMMDD THEN
  MOVE YYMMDD TO YYYYMMDD
END-IF
```

- New formats of ACCEPT

```
ACCEPT 4-DIGIT-YEAR-GREGORIAN
          FROM DATE YYYYMMDD.

ACCEPT 4-DIGIT-YEAR-JULIAN
          FROM DATE YYYYDDD.
```

NOTE: These formats are ALLOWED under CICS while the 2-digit year formats are not allowed under CICS

APOST/QUOTE

- You can now use both apostrophes (sometimes called single quotes) and quotes in the same program:

- DISPLAY 'The compiler looks to match'
  "whichever it finds first".

- Now COPYBOOKs with QUOTES can be used in programs that use APOST

- And vice-versa

New Compiler Options

- **ADATA**
- **CURRENCY**
- **INTDATE**
- **OPT(FULL)**
- **RMODE**
- **DATEPROC/YEARWINDOW**

NOTE:
No RES/NORES option anymore; always 'RES'

IBM Debug Tool

- **Advanced debugging capabilities**
  - **Interactive debugging of CICS-COBOL applications**
  - **Multi-language applications**
  - **Subset of COBOL language statements for Debug commands**

SET, MOVE, COMPUTE, IF, EVALUATE, PERFORM, CALL, ...

Evaluate expressions without recompiling

Note: Available in Full Function Feature of
COBOL for VSE/ESA,
PL/I for VSE/ESA, or
C for VSE/ESA

COBOL Report Writer Precompiler

- Continued support for COBOL Report Writer macros via the COBOL Report Writer Precompiler (5798-DYR)
  - Convert Report Writer statements to non Report Writer
  - Allows Report Writer statements in COBOL for VSE/ESA applications

Summary - What Was Covered

- Position COBOL for VSE/ESA
- Support of Features Introduced by VS COBOL II
- New Language Features with COBOL for VSE/ESA
- Language Environment Support
- Debug Tool Support