This paper reports the results of a study to select a high order programming language for the development of computer programs for the digital communications terminal. All languages suitable for use with the NSC800 microprocessor were considered. The nine final candidates were evaluated by a methodology including benchmarking and determination of a figure of merit. During the conduct of the study it became clear that the program support environment must include both a minicomputer software engineering host, and a microcomputer development
20. ABSTRACT

The language selected is Interactive Systems (C). The system includes a cross-compiler running on a PDP-11 and generating code for the NSC800.
DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.
DIGITAL COMMUNICATIONS TERMINAL
HIGH ORDER PROGRAMMING
LANGUAGE STUDY

(DCT HOL STUDY)
VOLUME TWO: APPENDICES
26 NOVEMBER 1980

MARINE
CORPS
TACTICAL
SYSTEMS
SUPPORT
ACTIVITY

CAMP PENDLETON
CALIFORNIA 92055

Approved for public release; distribution unlimited.
### TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>REFERENCES</td>
<td>A-1</td>
</tr>
<tr>
<td></td>
<td>Tab 1</td>
<td>Primary References</td>
</tr>
<tr>
<td></td>
<td>Tab 2</td>
<td>Periodicals</td>
</tr>
<tr>
<td></td>
<td>Tab 3</td>
<td>General Bibliography</td>
</tr>
<tr>
<td></td>
<td>Tab 4</td>
<td>Language Documentation</td>
</tr>
<tr>
<td>B</td>
<td>BENCHMARK PROGRAMS</td>
<td>B-1</td>
</tr>
<tr>
<td></td>
<td>Tab 1</td>
<td>Program DCT BENCHMARK (Commented)</td>
</tr>
<tr>
<td></td>
<td>Tab 2</td>
<td>Program DCT BENCHMARK (Code Only)</td>
</tr>
<tr>
<td></td>
<td>Tab 3</td>
<td>Error-Seeded DCT BENCHMARK</td>
</tr>
<tr>
<td></td>
<td>Tab 4</td>
<td>Interactive &quot;C&quot; Source</td>
</tr>
<tr>
<td></td>
<td>Tab 5</td>
<td>Whitsmith &quot;C&quot; Source</td>
</tr>
<tr>
<td></td>
<td>Tab 6</td>
<td>Microsoft FORTRAN-80 Source</td>
</tr>
<tr>
<td></td>
<td>Tab 7</td>
<td>Cromemco RATFOR Source</td>
</tr>
<tr>
<td></td>
<td>Tab 8</td>
<td>PASCAL/WT Source</td>
</tr>
<tr>
<td></td>
<td>Tab 9</td>
<td>PASCAL/Z Source</td>
</tr>
<tr>
<td></td>
<td>Tab 10</td>
<td>PLI-80 Source (Version 1)</td>
</tr>
<tr>
<td></td>
<td>Tab 11</td>
<td>PLI-80 Source (Version 2)</td>
</tr>
<tr>
<td></td>
<td>Tab 12</td>
<td>PLMX Source</td>
</tr>
<tr>
<td></td>
<td>Tab 13</td>
<td>PLZ Source</td>
</tr>
<tr>
<td>C</td>
<td>BENCHMARK RESULTS</td>
<td>C-1</td>
</tr>
<tr>
<td>D</td>
<td>DELPHI PHASE 1: WEIGHTS</td>
<td>D-1</td>
</tr>
<tr>
<td>E</td>
<td>DELPHI PHASE 2: SCORES AND FIGURES OF MERIT</td>
<td>E-1</td>
</tr>
</tbody>
</table>

### LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-1</td>
<td>Benchmark Results</td>
<td>C-3</td>
</tr>
<tr>
<td>D-1</td>
<td>Ranking of Features</td>
<td>D-3</td>
</tr>
<tr>
<td>E-1</td>
<td>Figures of Merit without Benchmark Values</td>
<td>E-6</td>
</tr>
<tr>
<td>E-2</td>
<td>Final Figures of Merit</td>
<td>E-7</td>
</tr>
</tbody>
</table>
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-1</td>
<td>DCT HOL Summary Benchmark Results</td>
<td>C-2</td>
</tr>
<tr>
<td>D-1</td>
<td>Phase 1 Delphi Study Statistical Summary</td>
<td>D-2</td>
</tr>
<tr>
<td>D-2</td>
<td>First Iteration DCT HOL Language Feature Response Summary</td>
<td>D-4</td>
</tr>
<tr>
<td>D-3</td>
<td>Second Iteration DCT HOL Language Feature Response Summary</td>
<td>D-5</td>
</tr>
<tr>
<td>D-4</td>
<td>Third Iteration DCT HOL Language Feature Response Summary</td>
<td>D-6</td>
</tr>
<tr>
<td>E-1</td>
<td>Delphi Study -- Phase 2 Summary (1st Iteration)</td>
<td>E-2</td>
</tr>
<tr>
<td>E-2</td>
<td>Delphi Study -- Phase 2 Summary (2nd Iteration)</td>
<td>E-3</td>
</tr>
<tr>
<td>E-3</td>
<td>Delphi Study -- Phase 2 Summary (3rd Iteration -- MCTSSA)</td>
<td>E-4</td>
</tr>
<tr>
<td></td>
<td>Delphi Study -- Phase 2 Summary (3rd Iteration -- NOSC Included)</td>
<td>E-5</td>
</tr>
</tbody>
</table>

Accession For

<table>
<thead>
<tr>
<th>NTIS GRAAI</th>
<th>DTIC TAB</th>
<th>Unannounced</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

By Distribution/ Availability Codes

<table>
<thead>
<tr>
<th>Dist</th>
<th>Avail and/or Special</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
</tr>
</tbody>
</table>
APPENDIX A

BIBLIOGRAPHIC INFORMATION

This appendix contains four tabs. Tab 1 presents the documents and articles which were of particular value in the preparation of the report. Tab 2 lists periodicals which were reviewed for advertising and general market information regarding available HOLs. Tab 3 is a detailed bibliography of the trade journal literature on high order languages. Tab 4 is a list of the publications obtained from language vendors which are references for the languages which were evaluated.
Tab 1
Primary References


(b) Considerations for a Microprocessor Policy, with appendixes, NCTSSA, 18 June 1980.


(g) Pratt, T. W., Programming Languages: Design and Implementation, Prentice-Hall, 1975, p. 207.


(k) Schindler, Max, "UNIX with Workbench will Serve Superminis,
Electronics, 8 November 1979, pp. 30-31.

Systems, August 1980, pp. 68-76.
Tab 2
Periodicals


The information to prepare this bibliography of high order languages was obtained by researching the journals listed below, representing the period March through September 1980:

**Byte**

*Communications of the ACM*

*Computer Design*

*Datamation*

*EDN*

*Electronics*

*Electronics Design*

*Mini-Micro Systems*

High Order Languages -- General Information: Journal articles that contain surveys, comparisons, discussion and general information about the various high order Languages.

High Order Languages -- Specific: Journal articles that contain information about specific language characteristics and their use in applications programming.

Microcomputer Development Systems: Journal articles that contain information about the hardware and software characteristics of the various state-of-the-art MDSs that are currently available.

In Circuit Emulation: Journal articles that discuss the uses and types of in-circuit emulator as a product-development and interactive debugging facility.

High Order Language Philosophy: A collection of material by several authors discussing the relative merits of using HOLs for applications development.
High Order Language Study Bibliography

High Order Languages; General Information


Product Advertisement

High Order Language Study Bibliography

High Order Languages: Specific

ADA


APL


FORTH


MDL - Modular Development Language


MPL - Developed by Motorola


PASCAL


PL/1


PLZ


PILOT/P


ROSETTA


TINY "C"


Product Advertisements


High Order Language Study Bibliography

Microcomputer Development Systems


Product Advertisements


High Order Language Study Bibliography

In Circuit Emulation


Product Advertisements


High Order Language Study Bibliography

High Order Language Philosophy


High Order Language Study Bibliography

Miscellaneous Material


Product Advertisements etc.


8. Texas Instruments Inc., "TI To Add PASCAL to its PS990 Development Unit," Electronics, Mar 26, 1979, p.36.


The following documents have been ordered to support this study effort and have been received. They are listed in alphabetical order according to title.


Application Note Index, Hewlett-Packard, nd.


"DC -- An Interactive Desk Calculator," Bell Laboratories, md.


Getting Started with PASCAL/64000, Hewlett-Packard, nd.


An Introduction to STOIC, Jonathan M. Sachs, Biomedical Engineering Center for Clinical Instrumentation, March 1978.


"The Portable C Library (on UNIX)," N. E. Lesk, Bell Laboratories, nd.


RATFOR-A Preprocessor for a Rational FORTRAN, Brian W. Kernighan, Bell Laboratories, 1975.


"Tiny PASCAL," SuperSoft, nd.


Tutorial Introduction to the Unix Text Editor, A. B. W. Kernighan, Bell Laboratories, 1975.


"YACC--Yet Another Compiler-Compiler," Stephen C. Johnson, Bell Laboratories, nd.


Product brochures have been received from Dynabyte, Tektronix, Whitesmith Ltd., GNAT computers, Zilog, Avocet Corp., Intel Corp., FORTH, Inc., Heurikon Corp., Infosoft, Hewlett-Packard, Cromemco, MicroSoft, Ontel, Emulogic, Hughes, SuperSoft, and Lifeboat Associates.
APPENDIX B
DCT/HOL STUDY BENCHMARK
PROGRAM LISTINGS

This appendix consists of the DCT high order language benchmark program listings for the surviving candidate languages. The listings are divided into two sections. Tabs 1 through 3 contain the pseudo-code specification for the benchmark program. There are three versions of the specification: a code-only version, a heavily commented version, and an error-seeded version.

Tabs 4 through 13 consist of the actual source code listings for the benchmark program written in the target languages. Page numbers are found on each page in the lower right-hand corner because many of the listings continue beyond margin boundaries.
<table>
<thead>
<tr>
<th>Tab 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program DCT Benchmark</td>
</tr>
<tr>
<td>(Commented)</td>
</tr>
</tbody>
</table>
**OCT BENCHMARK**

The purpose of this OCT benchmark program is to evaluate different microcomputer high order languages (HOLs) for the OCT application.

The benchmark is part of a larger effort to pick the best microcomputer HOL for the OCT. The primary purpose of the benchmark program itself is to obtain data on compile time, execution time, object program size, and to illustrate advantages and disadvantages of implementing the benchmark in various languages.

The benchmark is presented below in algorithmic form which closely resembles Pascal or Algol. Some readers may consider the benchmark to be presented in a programming design language, at any rate, the benchmark is presented here with extensive comments to assist the programmer in implementing the benchmark in different languages such as Pascal, PL/1, FORTRAN, PL/I, and NATIVE.

The programmer should read the following commented benchmark carefully before implementing the benchmark in a particular language, because extensive guidance is given. Also, a separate document, entitled "Rules for Implementing the OCT Benchmark" should be studied prior to coding the benchmark in a particular language.

This benchmark was developed at the Marine Corps Tactical Systems Support Activity, Camp Pendleton, California, 92055.

---

**CONSTANTS, TYPES, AND VARIABLE DECLARATIONS FOR DATA ITEMS REQUIRED TO BE**

**GLOBAL IN SCOPE, FOR LANGUAGES WHICH DO NOT SUPPORT 'TYPE', THIS FUNCTION SHOULD BE ACHIEVED BY USING 'LITERALLY', 'REPLACE', 'MEANS', OR THE LOGICAL EQUIVALENT.**

**PROGRAM OCTBENCHMARK**

**CONST**

ARRAYSIZE = 125;  (* used to control the size of ARMA and AMAT, as well as the number of loops in the array accessing loop. *)

LULOOPS = 575;  (* used to control the number of loop iterations *)

UPR=ACTION=LOOPS = 100;  (* used to control the number of loop iterations in the * )

---
TIMINGCONTROL = 2771
  (* USED TO CONTROL THE NUMBER OF TIMES THAT *)
  (* KERNEL, THE ACTUAL BENCHMARK EXECUTION *)
  (* PROCEDURE IS CALLED. THIS WILL BE ADJUSTED *)
  (* SO THAT RUN TIMES ARE EASY TO MEASURE. *)

LUPON = 2771
  (* THIS PARAMETER IS INSTALLATION-DEPENDENT. *)
  (* IT WILL BE CHosen SO THAT OUTPUT OCCURS *)
  (* ON AN I/O PORT THAT IS NOT CONNECTED TO *)
  (* ANYTHING, *)

NUMBERMSG = 871
  (* CONTROLS THE NUMBER OF TIMES THE MESSAGE *)
  (* PROCESSING LOOP IS EXECUTED. *)

BYTE7 = 001
  (* AN 8-BIT QUANTITY OF ALTERNATING O/P ANYTW.*)
  (* USES AND ZEROS TO TEST OUTPUT, SHIINTING, *)
  (* AND NOTATING CAPABILITIES. *)

INTEGER1 = 3901
  (* A 16-BIT INTEGER USED TO TEST INTEGER *)
  (* ARITHMETIC CAPABILITIES. *)

INTEGER2 = -1541
  (* A 16-BIT INTEGER USED TO TEST INTEGER *)
  (* ARITHMETIC CAPABILITIES. *)

IMASK = 11
  (* USED TO MASK THE HIGHEST BIT IN A BYTE *)

MSBLEN = 801
  (* INCURING MESSAGES ARE EXPECTED TO BE 00 *)
  (* CHARACTERS IN LENGTH. *)

BUFFERN = 151
  (* ALL BUFFERS ARE 16 CHARACTERS IN LENGTH *)
  (* RANGING FROM 0 TO 15, INCLUSIVE. *)

STARTCHAR = '1'
  (* THE ASCII 1 DENOTES THE START OF A MESSAGE *)

TYPE

BUFFERTYPE = ARRAYBUFFERN
  (* ALL CHARACTER BUFFERS RANGE FROM *)
  (* 0 TO BUFFERN, I.E., FROM 0 TO 15. *)

VAR

TIMINGLOOPER,
LOOPINDEX,
*INDEX,
*COUNT,
*ACTION,
INBUFFINDEX,
OUTBUFFINDEX
  (* USED TO CONTROL THE CASE STATEMENT, *)
  (* MINING IN VALUE FROM 0 TO 0, *)
  (* INDEX IN THE INPUT BUFFER, INBUFF.*)
  (* ALL OF THESE INTEGER SHOULD BE IMPE *)
  (* MENTED AS 16-BIT INTEGERS. *)

ARRAY1, ARRAY2, ARRAY3
  (* OUT ARRAY1 AND ARRAY2 ARE 16-BIT INTEGER *)
  (* ARRAYS OF SIZE ARRAYSIZE (128) AND ARE *)
  (* USED TO TEST 1-DIMENSIONAL ARRAY ACCESSING *)

INBUFF, BUFFERTYPE
  (* A CHARACTER BUFFER USED FOR INPUT *)

CHARRANGE
  (* A CHARACTER VARIABLE USED FOR TEMPORARY *)
  (* STORAGE AND PARAMETER PASSING. *)

*TYPE, RECORD
  (* A COMPLEX RECORD OR TABLE), CONSISTING *)
FUNCTION GETBYTE 1 CHANS

**PROLOGUE:**

Invokes GETBYTE returns a character value when

- Invoked, it obtains the character from the input
  buffer (inbuffer), and increments the buffer
  index (inbufferpt), checking to see if it exceeds
  buffermax in size. If it does, inbufferptr is
  reset to zero.

- Inputs: None.
- Outputs: GETBYTE, a character value.
- Called by: Kernel.
- Calls: No procedures or functions.

BEGIN

GETBYTE = INBUFFER(INBUFFERPTH)

INBUFFERPTH = INBUFFERPTH + 1

IF INBUFFERPTH > BUFFERMAX THEN INBUFFERPTH = 0

END GETBYTE.

PROCEDURE PUTBYTE (INCHAN 1 CHANS; POUTBUFFER 1 BUFFERTYPE)

**PROLOGUE:**

Procedure putbyte puts one character into the

- output buffer (putbuffer), and then increments
  the buffer index (outbufferpt), checking to see
  if it exceeds buffermax in size. If it does,
  outbufferptr is reset to zero.

- Inputs: INCHAN, a character value, and OUTBUFFER, a
  character buffer of size BUFFERMAX, which is
  passed by reference as an array to test compiler
  efficiency in passing whole arrays.

- Outputs: PUTBUFFER (modified).
- Called by: Kernel.
- Calls: No procedures or functions.

BEGIN

PUTBUFFER(OUTBUFFERPTH) = INCHAN;

OUTBUFFERPTH = OUTBUFFERPTH + 1

IF OUTBUFFERPTH > BUFFERMAX THEN OUTBUFFERPTH = 0

END PUTBYTE.

PROCEDURE KERNEL:
**PROCEDURE MAIN**

The main program performs the following actions:

1. **INPUTS:**
   - **NONE**

2. **CALLS:**
   - **FUNCTION GETBYTE**
   - **PROCEDURE PUTBYTE**

3. **OUTPUTS:**
   - **NONE**

4. **LOCAL VARIABLE DECLARATIONS**
   - **NONE**

5. **FUNCTIONS DECLARAT**
   - **NONE**

6. **PROCEDURES DECLARAT**
   - **NONE**

**BEGIN**

- **PROCEDURE MAIN**

**INPUT**

- **NONE**

**PROCESSING**

- **BEGIN**
  - **FOR** LOOP
  - **PROCEDURE GETBYTE**
  - **DECLARATION**
  - **INTERRUPT DECLARATION**
  - **FUNCTION DECLARATION**
  - **PROCEDURE DECLARATION**

**EXECUTION**

- **BEGIN**
  - **FOR** LOOP
  - **BEGIN**
    - **OUTPUT**
    - **BEGIN**
      - **PROCEDURE PUTBYTE**
    - **END**
    - **END**
    - **FOR** LOOP
    - **BEGIN**
      - **PROCEDURE PUTBYTE**
    - **END**

**END**

- **END**
WHILE MILECOUNTER < MILELENGTH DO

(*) THE PURPOSE OF THIS MILE LOOP IS TO TEST A LANGUAGE'S *
(*) MILE CAPABILITY — IN THOSE LANGUAGES WHICH DO NOT *
(*) SUPPORT A MILE CONSTRUCT THE MILE LOOP MUST BE *
(*) IMPLEMENTED BY THE STRUCTURED USE OF A GOTO OR OTHER *
(*) CONSTRUCT. ITERATIVE LOOPING IS NOT PERMITTED. *

BEGIN (* INPUT MESSAGE CHARACTERS *)
MENCEHAN & GETBSE; (* GET NEXT Character IN MSG *)
(* NOW SHIFT THE CHARACTER RIGHT ONE BIT AND SEE IF *)
(* THE RIGHTMOST BIT IS A ZERO. *)

IF (RIGHTSHIFT(MENCEHAN, 1) AND BITMASK) = 1 THEN
PUTBYTE(MENCEHAN, OUTBUFFER);
ELSE (* DO THE SAME THING = THIS IF/ELSE IS T * *)
(* CHECK COMPILE IF ELSE CAPABILITY ONLY. *)
PUTBYTE(MENCEHAN, OUTBUFFER);
MILECOUNTER = MILECOUNTER + 1;
END WHILE;
END (* FOR LOOPCOUNTER = 0 TO NUMBERMGSS LOOP *)

UPACTION = 0; (* INITIALIZE UPACTION TO ZERO SO THAT IT CAN *)
(* RANGE IN VALUE FROM 0 TO 9. IN IMOS *)
(* LANGUAGES WHICH SUPPORT A MODULUS FUNCTION *)
(* UPACTION SHOULD BE IMPLEMENTED AS A MODULUS *)
(* OTHERWISE IT SHOULD BE INCREMENTED BY ONE *)
(* EACH LOOP, CHECK AGAINST 9, AND RESET TO *)
(* ZERO IF GREATER THAN 9 AS DONE BELOW AT THE *)
(* BOTTOM OF THE CASE STATEMENT. *)

FUN LOOPSUCEPTER = 4 TO UPACATIONLOOPS UD
BEGIN (* CASE UPACTION WHERE UPACTION RANGES FROM 3 TO 9 *)
CASE UPACTION OF

(*) SHIFT TESTBYTE TO THE RIGHT BY THREE BITS *)
(*) THEN $U$E IN MIXETYPE.*)
0,3,9 : MIXETYPE.CHARBUFFER(UPACTION) =
RIGHTSHIFT(TESTBYTE, 3))

1,4,7 : (* CIRCULAR RIGHTTESTBYTE TO THE LEFT BY 2 *)
(* BITS AND CHECK TO SEE IF THE RIGHTMOST *)
(* BIT IS A ZERO. BITKE IN MIXETYPE *)
(* MIXETYPE.CHARBUFFER(UPACTION) =
LEFTSHIFT(TESTBYTE, 2) AND BITMASK *)

2,0 : (* MOVE ALL 16 CHARACTERS IN INBUFFER TO *)
(* THE CHARACTER BUFFER PORTION OF MIXETYPE *)
(* MIXETYPE.CHARBUFFER IS INBUFFER *)

OTHERWISE: (* WILL BE EXECUTED WHEN UPACTION = 9 UN 8, *)
(* TEST LANGUAGE CAPABILITY FOR 16-BIT *)
(* ARITHMETIC *)
(* MIXETYPE,INTERMEDIATE = (((INTEGER1/INTEGER2) + *)
(INTEGER1)/INTEGER2) + INTEGER1 *)

END CASE)

UPACTION = UPACTION + 1; (* INCREMENT UPACTION AND *)
IF UPACTION = 9 THEN
UPACTION = 1;
ENDIF (* FOR LOOPCOUNTER = 4 TO UPACTIONLOOPS *)
END (* PROCEDURE KERNEL *)
BEGIN (**************************************************************************)
(* MAIN PROGRAM = EXECUTION BEGINS HERE *)
(**************************************************************************)

**HINT** (BEGIN BENCHMARK EXECUTION): (*) WHEN THIS MESSAGE APPEARS *)
(*) ON THE CRT: BEGIN TIMING *)
(*) THE BENCHMARK EXECUTION *)

**FUN** LOOPCOUNTER = 0 TO ARRAYSIZE UD

**ARR**[2](LOOPCOUNTER) = LOOPCOUNTER;
(*) FILL **ARR**[2] WITH ASCENDING *)
(*) INTEGERS, 0 = 19, *)

**FUN** INBUFFERPTH = 0 TO BUFFERMAX UD

**INBUFFER**(INBUFFERPTH) = 'A';
(*) FILL THE INPUT BUFFER WITH *)
(*) PHONY TEXT, ALL 'A'S, *)

**INBUFFER**(10) = STARTCODE;
(*) PUT ONE STARTCODE IN BUFFER *)
(*) SO KERNEL WILL FIND IT, *)

**INBUFFERPTH**, OUTBUFFERPTH = 31
(*) POINT BOTH BUFFER INDICES *)
(*) TO FIRST CHARACTER IN BUFFER *)

**FUN** TIMINGLOOPEN = J TO TIMINGCOUNTER UD

**KERNEL**: (*) THIS LOOP CONTROLS HOW *)
(*) MANY TIMES **KERNEL** IS *)
(*) CALLED; THIS CONTROLLING *)
(*) BENCHMARK EXECUTION TIME *)

**HINT** (MIXEDTYPE, END EXECUTION): (*) WHEN THIS MESSAGE APPEARS ON *)
(*) THE CRT, STOP TIMING THE *)
(*) BENCHMARK EXECUTION TIME *)

END.

---

B-8
Tab 2
Program DCT Benchmark (Code Only)
PROGRAM UCTBENCHMARK

CONST

 AMMAYSIZE = 125
 UMMACTILUQQPS = 104
 TESTBYTE = 65
 INTEGER2 = 80
 INTEGER3 = 151

 STARTCOUNT = 151

 TYPE

BUFFERTYPE = ARRAY[0..BUFFENMAX] OF CHAR

VAR

 TIMINGLUOPEN, LOOPCOUNT, *MILECOUNTER, OPCODE, INBUFFERPTR,
 OUTBUFFERPTR, INTEGER;
 AMMAY, AMMAYS, AMMAYSIZE, INTEGER;
 INBUFFER, BUFFERTYPE;
 MACHMAN, CHA;

 MIXEDTYPE, RECORD;
 CHARBUFFER, BUFFERTYPE;
 INNUMBER, INTEGER;
 END RECORD MIXEDTYPE;

 FUNCTION GETBYTE(CCHAR);
 BEGIN
 GETBYTE = INBUFFER(INBUFFERPTR);
 INBUFFERPTR = INBUFFERPTR + 1;
 IF INBUFFERPTR > BUFFENMAX THEN INBUFFERPTR = 0; END GETBYTE;

 FUNCTION PUTBYTE(INCHAR, CHA, OUTBUFFER, BUFFERTYPE);
 BEGIN
 OUTBUFFER(OUTBUFFERPTR) = INCHAR;
 OUTBUFFERPTR = OUTBUFFERPTR + 1;
 IF OUTBUFFERPTR > BUFFENMAX THEN OUTBUFFERPTR = 0; END PUTBYTE;

 FUNCTION KERENLI;

VAR OUTBUFFER, BUFFERTYPE;

 BEGIN
 FOR LOOPCOUNTER = 0 TO ILLQQPS DO;
 OUTPUT((NOT(TESTBYTE)) , TEST); END LOOPCOUNTER;

 FOR LOOPCOUNTER = 0 TO AMMAYSIZE DO;
 AMMAY(LOOPCOUNTER) = AMMAY2(AMMAYSIZE - LOOPCOUNTER);
 END LOOPCOUNTER;

 FOR LOOPCOUNTER = 0 TO NUMELEMENTS DO;
 BEGIN
 REPEAT
 MACHMAN = GETBYTE;
 UNTIL MACHMAN = STARTCOUNT;
 MILECOUNTER = 21;
while *MILECOUNTER < MSGLENGTH DO
BEGIN
  NEXTCHAR := GETBYTE;
  IF (RIGHTSHIFT(NEXTCHAR, 1) AND BITMASK) = 1 THEN
    PUTBYTE(NEXTCHAR, OUTBUFFER);
  ELSE
    PUTBYTE(NEXTCHAR, OUTBUFFER);
  ENDWHILE;
END FOR
END

UPACTION := 31

FOR LOOPCOUNTER := 1 TO OPCODELOOPS DO
BEGIN
  CASE OPCODE OF
  0, 3, 9: MIXEDTYPE, CHARBUFFER := RIGHTSHIFT(TRANSFER, 3);
  1, 4, 7: MIXEDTYPE, CHARBUFFER := LEFTROTATE(TRANSFER, 2) AND BITMASK;
  2, 0: MIXEDTYPE, CHARBUFFER := INBUFFER;
  OTHERWISE: MIXEDTYPE, INNUMBER := ((INTEGER1 INTEGER2) * INTEGER1) INTEGER2 INTEGER2 INTEGER1;
END CASE;

IF OPCODE = 1 THEN OPCODE := 0
END FOR
END

BEGIN (* PROGRAM EXECUTION *)

**WHILE (*BEGIN BENCHMARK EXECUTION*)

FUN LOOPCOUNTER := 0 TO AHAYSIZE DO
  ANNATZ(LOOPCOUNTER) := LOOPCOUNTER;
END FOR

FUN INBUFFERPTH := 0 TO BUFFERMAX DO
  INBUFFER(INBUFFERPTH) := 'A';
END FOR

INBUFFER(16) := STARTCUE;
INBUFFERPTH, OUTBUFFERPTH := 0;

FUN TIMINGLOOPER := 0 TO TIMINGCOUNTHDL DO
END FOR

**WHILE (MIXEDTYPE, 'END EXECUTION')

END.

B-11
<table>
<thead>
<tr>
<th>Tab 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error-Seeded DCT BENCHMARK</td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PROGRAM UCTBENCHMARK

******************************************************************************

******************************************************************************

*** ERROR SEEDED UCT BENCHMARK ***

*** THIS VERSION OF THE UCT BENCHMARK CONTAINS FIVE ERRORS DESIGNED ***
*** TO CAUSE COMPILE-TIME ERRORS DUE TO INCORRECT SYNTAX. HOWEVER, ***
*** NOT ALL COMPILERS WILL RECOGNIZE ALL OF THE ERRORS AT COMPILE TIME. ***
*** THE FIVE ERRORS ARE CLEARLY MARKED IN THE LISTING BELOW WITH ***
*** A SERIES OF ASTERISKS ALONG WITH AN EXPLANATION OF THE ERROR. ***

******************************************************************************

******************************************************************************

CONST

ANNAYSIZE = 1291
UPNACTJUNOUPS = 1661
TIMINGCUNTNUL = 771
LOPCNT = 771
TESTBYTE = 051
INTEGER = -1571
MSLENGTH = 681
STARTCODE = '81

TYPE

BUFFERTYPE = ARRAY(BUFFERMAX) OF CHAR

VAR

(***** ERROR 1: BY PLACING COMMENT BRACKETS AROUND LOOPCOUNT IN THE **) 
(***** LINE BELOW, THE DECLARATION OF "LUOPCOUNT" IS ELIMINATED **) 
(***** THE PURPOSE IS TO SEE HOW DIFFERENT COMPILERS TREAT **) 
(***** VARIABLE DECLARATIONS ON THE LACK THEREOF **) 

TIMINGLUOPER, (* LOPCOUNT, **) WHILECOUNT, OPNACTJN, INBUFFEPTR,
UOUTBUFFEPTR; INTEGER
ANNAYS, ANNAY: INTEGER; ANNAYS, ANNAYSIZE, OF INTEGER
INBUFFEP : BUFFERVTYPE
MIXTYPE, RECORD
CHARHUPFP : BUFFERVTYPE
MIXTYPE, CHARNUMBER, INTEGER
EMU RECORD MIXTYPE

FUNCTION GETBYTE : CHAR

BEGIN
GETBYTE = INBUFFEP( INBUFFEPTR )
INBUFFEPTR = INBUFFEPTR + 1
IF INBUFFEPTR > BUFFERMAX THEN INBUFFEPTR = 0
EMU GETBYTE
END

FUNCTION PUTBYTE : CHAR

BEGIN
PUTBYTE = INBUFFEP( INBUFFEPTR )
INBUFFEPTR = INBUFFEPTR + 1
IF INBUFFEPTR > BUFFERMAX THEN INBUFFEPTR = 0
EMU PUTBYTE
END
PROCEDURE PUTBYTE (LNCHAN, INCHAN, OUTBUFFER, OUTBUFFLEN)
BEGIN
  PUTBUFFER (OUTBUFFEPTR) = INCHAN
  OUTBUFFEPTR = OUTBUFFEPTR + 1
END PUTBYTE

PROCEDURE KERNEL
VAR OUTBUFFER, INCHAN, OUTBUFFLENGTH
BEGIN
FOR LOOPCOUNTER = 0 TO 10 LOOPS DO
  OUTPUT (NOT (TESTBYTE), LOOPCOUNTER)
END FOR LOOPCOUNTER
FOR LOOPCOUNTER = 0 TO ARRAYSIZE DO
  ARRAY1 (LOOPCOUNTER) = ARRAY2 (ARRAYSIZE - LOOPCOUNTER)
END FOR LOOPCOUNTER
FOR LOOPCOUNTER = 0 TO NUMMBERS DO
  REPEAT
    NECHAR = GETBYTE;
    UNTIL NECHAR = STARTCUE
  WHILE LOOPCOUNTER = 0
  WHILE LOOPCOUNTER = NUMLENGTH DO
    NECHAR = GETBYTE
    BEGIN
      (***** ERROR: IN THE LINE BELOW, OUTBUFFER HAS BEEN REPLACED *)
      (***** REIFIED: WITH ARRAY1 THE PURPOSE *)
      (***** IS TO EVALUATE TYPE-CHECKING OF PAKA *)
      (***** METEAS BEING PASSED SINGLE ARRAY1 IS *)
      (***** AN INTEGER ARRAY AND OUTBUFFER IS A *)
      (***** CHARACTER ARRAY *)
      IF (RIGHTSHIFT (NECHAR, 1) AND BITMASK = 1) THEN
        PUTBYTE (NECHAR, ARRAY1)
      ELSE
        PUTBYTE (NECHAR, OUTBUFFER)
      END IF
      LOOPCOUNTER = LOOPCOUNTER + 1
    END REPEAT
    END WHILE LOOPCOUNTER = 0
END FOR LOOPCOUNTER
END FOR LOOPCOUNTER
END FOR LOOPCOUNTER
END FOR LOOPCOUNTER
RIGHTSHIFT(RESULTE, 3))

1/4/7  1 MIXEDTYPE, CHANBUFFEP(OMACTION) #
       LEFTROTATE(RESULTE, 2) AND SLIMASK;

2/0  1 MIXEDTYPE, CHANBUFFEP [# INBUFFEP;]

***** ERROR 5; IN THE NEXT LINE, A LEFT PARENTHESIS #
***** MAY HAVE BEEN REMOVED TO CHECK FOR UNBALANCED #
***** PARENTHESES ERROR DETECTION AND REPORTING. #

THEN IF MIXEDTYPE, INTNUMBER # (((INTEGRI/INTEGR2) #
       (INTEGRI)/INTEGR2) * INTEGR2) * INTEGR1)

END CASE;

OMACTION # OMACTION + 1;
IF OMACTION > 9 THEN OMACTION = 0;
END FUR;
END KERNEL;

BEGIN (* PROGRAM EXECUTION *)

*LINE (*BEGIN BENCHMARK EXECUTION*)

FUR LOOPCOUNT = 1 TO ARRAYSIZE UU
       ARRAY2[LOOPCOUNT] = LOOPCOUNT;

FUR INBUFFENTH = 1 TO BUFFERMAX UU
       INBUFFEP([INBUFFENTH] = 'A!';

INBUFFEP(16) = 'STARTCURL'

INBUFFENTH+ UUTBUFFENTH = 0;

FUR TIMINGOPER = 1 TO TIMINGCONTROL UU
       KERNEL;

*LINE (MIXEDTYPE, 'END EXECUTION')

END.
Tab 4
Interactive "C" Source
/* PROGRAM DCTBENCHMARK */

/* Final version, with direct code. At Interactive 27OCT80 */

#define ARRAYSIZE 125
#define OPRATIONLOOPS 100
#define IOPORT 32
#define TESTBYTE 85
#define INT2 -150
#define MSGLENGTH 80
#define STARTCODE 'S'
#define ILOOPS 575
#define TIMINGCONTROL 12
#define NUMBERSGS 200
#define INT1 300
#define BITMASK 1
#define BUFFERMAX 15

/* Global data */

int timloop, whilcntr, opaction;
int inbufptr, outbfptr;
int array1[ARRAYSIZE+1],array2[ARRAYSIZE+1];
int loopcntr,index;

char inbuffer[BUFFERMAX+1];
char newchar;

char notbyte; /* Global for reference in asm .... */

struct record1 {
   char charbuf[r BUFFERMAX+1];
   int intnumbr;
};
struct record1 mixtype;

/* End of data declarations */

/****************************
* * * 
* Main Program
* *
*******************************************/

main()
{
   printf("Begin benchmark execution\n");
   for(loopcntr = 0; loopcntr <= ARRAYSIZE; loopcntr++)
      array2[loopcntr] = loopcntr;
   for(inbufptr = 0; inbufptr <= BUFFERMAX; inbufptr++)
      inbuffer[inbufptr] = 'A';
   inbuffer[10] = STARTCODE;
   inbufptr = outbfptr = 0;
   printf("End benchmark execution\n");
}
for(timloop = 0; timloop <= TIMINGCONTROL; timloop++)
    kernel();

for(index = 0; index <= BUFFERMAX; index++)
    printf("%s", mixtype.charbufr[index]);
    printf("%d\n", mixtype.intnumbr);
    printf("End execution\n");
} /* End main program *********************************************/

/*Function definitions */

/* KERNEL: Exercises the support functions. */

kernel()
{
    char outbufr[BUFFERMAX + 1];
    notbyte = ~TESTBYTE;
    for(loopcntr = 0; loopcntr <= IOLOOPS; loopcntr++)
    {
        asm lda.nn notbyte; /* a <-- byte to be output */
        asm ldrr c, IOPORT; /* c <-- output port */
        asm out.cr a; /* do the output */
    }

    for(loopcntr = 0; loopcntr <= ARRAYSIZE; loopcntr++)
        array1[loopcntr] = array2[ARRAYSIZE - loopcntr];

    for(loopcntr = 0; loopcntr <= NUMBERMSGS; loopcntr++)
    {
        do {
            newchar = getbyte();
        } while(newchar != STARTCODE);

        whilecntr = 0;
        while(whilecntr++ < MSGLENGTH)
        {
            if (((newchar = getbyte()) >> 1) & BITMASK)
                putbyte(newchar, outbufr);
            else
                putbyte(newchar, outbufr);
        } /* End while */
    } /* End for */

    opaction = 0;
    for(loopcntr = 0; loopcntr <= OPRATIONLOOPS; loopcntr++)
    {
        switch (opaction)
        {
            case 0: break;
            case 3: break;
            case 9: mixtype.charbufr[opaction] = (TESTBYTE >> 3);
                    break;
            case 1:
            case 4:
            case 7: mixtype.charbufr[opaction] =
Iftrot(TEStBYTE, 2) & BITMASK;
break;
case 2:
case 6: for(index = 0; index <= BUFFERMAX; index++)
mixtype.charbufr[index] = inbuffer[index];
break;
default: mixtype.intnumbr = (((INT1/INT2)\nINT1)/INT2)*INT2) * INT1;
break;
} /* End switch */
if (**opaction > _2)opaction = 0;
} /* End for */
} /* End kernel */

/* GETBYTE: Get a character from inbuffer. Maintain inbufptr. */
char getbyte()
{
    char rtnbyte;
    rtnbyte = inbuffer[inbufptr];
    if (**inbufptr > BUFFERMAX) inbufptr = 0;
    return(rtnbyte);
} /* End getbyte */

/* PUTBYTE: Put a character in putbuffer. Maintain outbfptr. */
putbyte(inchar, putbuffr)
{
    char inchar;
    char putbuffr[];
    putbuffr[outbfptr] = inchar;
    if (**outbfptr > BUFFERMAX) outbfptr = 0;
} /* End putbyte */

/* IFTROT: Rotate a byte left, by number of bits. */
char Iftrot(rotbyte, number)
char rotbyte, number;
{
    asm ldr.ixd b,ix*8; /* b <-- number of shifts */
    asm ldr.ixd a,ix*6; /* a <-- byte to be shifted */
    asm lrot: rcr a; /* rotate the byte, while */
    asm djnza lrot; /* decrementing b, jump on not zero */
    asm ld.ixdr ix*6,a; /* put rotated byte back */

    /* the return will put (ix*6) into the hl register pair */
    return (rotbyte);
} /*End Iftrot */

/* END DCT BENCHMARK PROGRAM *****************************************/
Tab 5
Whitesmith "C" Source
/*
 DCT BENCHMARK PROGRAM IN WHITESMITH'S C
 */

#define ARRAYSIZE 125
#define BITMASK 001
#define BUFFERMAX 15
#define INT1 300
#define INT2 -150
#define IDOOPS 575
#define IOPORT 0xff /* adjust if conflict */
#define MSGLENGTH 80
#define NUMBERMSGS 200
#define OPRATIONLOOPS 100
#define STARTCODE 's'
#define TESTBYTE 95
#define TIMINGCONTROL 12

/*
 EXTERNAL VARIABLES:
 Due to an anomaly in the Whitesmith C compiler, all
 external variables must be given initial values.
 Therefore, external variables which require no initialization
 must be initialized to satisfy the compiler.
 */

char inbuf[BUFFERMAX+1]
int array1[ARRAYSIZE+1] (0);
int array2[ARRAYSIZE+1] (0); int inbufp (0);
int outbufp (0);

struct 
{
    char charbuf[BUFFERMAX+1];
    int intnumber1;
} mixedtype ("", 0);

/**************************************************************************
 * *
 * Main Program *
 * *
 */

main()
{
    register loopctr; /* for speed */
    static int timinelooper;
    printf("Begin benchmark execution\n");
    for (loopctr = 0; loopctr <= ARRAYSIZE; ++loopctr)
        array2[loopctr] = loopctr;
    for (timinelooper = 0; timinelooper <= TIMINGCONTROL; ++timinelooper)
for (loopctr = 0; loopctr <= BUFFERMAX; ++loopctr)
    printf("%a", mixedtype.charbuf[loopctr]);
    printf("\n\%s\nEnd execution\n", mixedtype.intnumber);

/*
 * Support Functions
 */

set a character from inbuf, increment inbufp to next character.

char setbyte()
{
    static char c;
    c = inbuf[inbufp];
    if (++inbufp > BUFFERMAX)
        inbuf = 0;
    return (c);
}

kernal is the main function in the benchmark program.
It is declared VOID to indicate that no return value
is expected. VOID is defined in the file STD.H as
being equivalent to int (16 bit integer).

VOID kernal()
{
    static char outbuf[BUFFERMAX+1];
    static char newchar;
    static int opaction;
    register loopctr, whilectr; if /* for speed */
    for (loopctr = 0; loopctr <= IOOOPS; ++loopctr)
        out(IOPORT, ~TESTBYTE); /* standard library output routine */
    for (loopctr = 0; loopctr <= ARAYSIZE; ++loopctr)
        array[loopctr] = array2[ARAYSIZE-loopctr];
    for (loopctr = 0; loopctr <= NUMBERMSGS; ++loopctr) {
        while (((newchar = setbyte()) != STARTCODE)
    }
    /*
 * oaction = 0;
 */
    for (loopctr = 0; loopctr <= OPRATIONLOOPS; ++loopctr) {
        switch(opaction) {
            case 0:
            case 3:
            case 9:
            case 91:
B-22
mixedtype.charbuf[opaction] = ++TESTBYTE;
break;

case 1:
case 4:
case 7:
mixedtype.charbuf[opaction] = 1rot(TESTBYTE, 2);
break;

case 2:
case 6:
for (i = 0; i <= BUFFERMAX; ++i)
    mixedtype.charbuf[i] = inbuf[i];
break;

default:
mixedtype.intnumber = (((INT1/INT2) * INT1)
/ INT2) * INT2) + INT1!

if (++opaction > 9)
    opaction = 0;

} /* end kernel */

g/* put a byte into the next position in putbuf */

VOID putbyte(inchar, putbuf)
char inchar;
char putbuf[];
{
putbuf[outbufp] = inchar;
if (++outbufp > BUFFERMAX)
    outbufp = 0;

/* rotate byte n left b bit positions */

char 1rot(n, b)
char n;
char b;
{
b &= 007;    /* 0..7 */
return((n << b) | (n >> 8-b));
}
Tab 6

Microsoft FORTRAN-80 Source
PROGRAM OCT HOL BENCHMARK IN MICROSOFT FORTRAN-80 FOR CP/M
WRITTEN BY: CAPT B.F. BRADY, U.S.M.C

C***** VARIABLE DECLARATIONS FOR MAIN PROGRAM ****
IMPLICIT LOGICAL(L-R)
INTEGER OTBFPT, INBFPT, TIMLPR, INTNUM, ARRAY2
LOGICAL*1 CHRBUF, INBUFF, BITMSK, TSTBYT
C***** DEFINE GLOBALS IN COMMON :
COMMON //OTBFPT, INBUFF(16), INBFPT, CHRBUF(16), ARRAY2(126),
* INTNUM, BITMSK, TSTBYT
C***** PRESET DATA :
DATA INBUFF/10*1HA, 1HS, 5*11IA/, BITMSK/1/, OTBFPT/0/, TSTBYT/11U/

C***** BENCHMARK MAIN PROGRAM
C***** WRITE OUT THE START MESSAGE
WRITE(5, 1)
C***** INITIALIZE ARRAY2
DO 102 TIMLPR=1,126
  ARRAY2(TIMLPR)=TIMLPR
C***** END INITIALIZE LOOP
102 CONTINUE
DO 105 TIMLPR=0,12
C***** BEGIN MAIN TIMING LOOP
CALL KERNEL
C***** END MAIN TIMING LOOP
105 CONTINUE
C***** WRITE OUT THE END MESSAGE
WRITE(5, 2)
C***** END OF MAIN PROGRAM
C***** FORMAT STATEMENTS :
1 FORMAT(' Begin BENCHMARK Execution.')
2 FORMAT(' End BENCHMARK Execution.')
STOP
END
C
C***** SUBROUTINES AND FUNCTIONS :
FUNCTION GETBYT(DUMARG)
LOGICAL*1 INBUFF, DUMARG
C***** DUMARG IS USED ONLY BECAUSE A PARAMETER IS
C***** REQUIRED FOR A FUNCTION CALL
INTEGER OTBFPT, INBFPT, LOCPNT
COMMON //OTBFPT, INBUFF(16), INBFPT
LOCPNT=INBFPT
INBFPT = INBFPT + 1
IF (INBFPT.GT.16) INBFPT = 1
GETBYT = INBUFF (LOCPNT)
RETURN
C***** END GETBYT
END

C SUBROUTINE PUTBYT(INCHAR, PUTBUF)
C
LOGICAL*1 INCHAR, PUTBUF (16)
C
INTEGER OTBFPT
C
COMMON //OTBFPT
C
PUTBUF (OTBFPT) = INCHAR
OTBFPT = OTBFPT + 1
IF (OTBFPT.GT.16) CTBFPT = 1
RETURN
C***** END PUTBYT
END

C SUBROUTINE KERNEL
IMPLICIT LOGICAL (L-R)
C
LOGICAL*1 NEWCHR, INBUFF, TSTBYT, BITMSK, CHRBUF, CTBUFF (16), START, BYTE
C
INTEGER OTBFPT, INBFPT, INTNUM, LPCNTR, TIMLPR, WHLCNT, CPRACT, ARG2, ARRAY2, ARRAY1 (126), INTOR1, INTOR2, TEMP
C
COMMON //OTBFPT, INBUFF (16), INBFPT, CHRBUF (16), ARRAY2 (126),
* INTNUM, BITMSK, TSTBYT
C
DATA START/1HS/, INTGR1/300/, INTGR2/-150/
C
DO 112 LPCNTR = 0, 575
BYTE = .NOT. TSTBYT
CALL OUT (BYTE, 200)
112 CONTINUE
C
DO 115 LPCNTR = 1, 126
ARG2 = 127 - LPCNTR
ARRAY1 (LPCNTR) = ARRAY2 (ARG2)
115 CONTINUE
C
DO 127 LPCNTR = 0, 200
C***** BEGIN MESSAGE LOOP
119 CONTINUE
IF (GETBYT (BYTE).NE.START) GOTO 119
WHLCNT = 0
122 IF (WHLCNT.GT.80) GOTO 126
C***** BEGIN WHILE LOOP
NEWCHR = GETBYT (BYTE)
TEMP = NEWCHR
IF(RTSHFT(TEMP,1).AND.BITMSK)GOTO 124
CALL PUTBYT(NEWCHR,OTBUFF)
GOTO 125
124 CONTINUE
   CALL PUTBYT(NEWCHR,OTBUFF)
125 CONTINUE
   WHLCNT=WHLCNT+1
C***** END WHILE LOOP
   GOTO 122
126 CONTINUE
C***** END MESSAGE LOOP
127 CONTINUE
   OPRACT=0
   DO 138 LPCNTR=0,100
C***** BEGIN OPERATOR ACTION LOOP
   INDEX=OPRACT+1
   GOTO(129,131,128,129,134,131,129,134,128),OPRACT
C
C***** CASE OF 0,3 OR 9
128   CHRBUF(INDEX)=RTSHFT(TSTBYT,3)
C***** END 0,3 OR 9
     GOTO 135
C
C***** CASE OF 1,4 OR 7
129   CHRBUF(INDEX)=LFTROL(TSTBYT,2).AND.BITMSK
C***** END 1,4 OR 7
     GOTO 135
C
C***** CASE OF 2 OR 6
131   DO 133 ARG2=1,16
       CHRBUF(ARG2)=INBUFF(ARG2)
133   CONTINUE
C***** END 2 OR 6
     GOTO 135
C
C***** OTHERWISE:
134   INTNUM=(((INTGR1/INTGR2)*INTGR1)/INTGR2)*INTGR2)+INTGR1
C***** END OTHERWISE
C
C***** END OF CASE SIMULATION
135   CONTINUE
       OPRACT=OPRACT+1
       IF(OPRACT.GT.9) OPRACT=0
138 CONTINUE
C***** END OPERATOR ACTION LOOP
C***** END KERNEL
RETURN
END
Tab 7
Cromemco RATFOR Source
#PROGRAM DCT HOL BENCHMARK in RATFOR for CP/M
#Written by: Capt B.F. BRADY, U.S.M.C
# 
#include RFGLABEL.RAT # need this for TSW RatFor only #
#define Constants:
#define(arsize, 126)
#define(ioloop, 575)
#define(opacif, 100)
#define(timctl, 12)
#define(ioport, 200)
#define(nmsgs, 200)
#define(tstbyt, 85)
#define(intsr1, 300)
#define(intsr2, (-150))
#define(msglen, 80)
#define(strtcid, 'S')
#define(bufmax, 16)
#
#define Variable declarations for main program
#
INTEGER otbfpt, inbfpt, timlpr, intnum, array2
#
LOGICAL*1 chrbuf, inbuff, bitmsk, temp
#
#define globals in COMMON:
COMMON //otbfpt, inbuff(bufmax), inbfpt, chrbuf(bufmax),
         array2(ar size), temp, intnum, bitmsk
#
#define DATA:
#
DATA inbuff/10*'A', 'S', 5*'A', .5*'A/, bitmsk/1/, otbfpt/0/
#
#define Benchmark Main Program
#
WRITE(5,1) #put start message on console
  # initialize array2 elements to equal their index
FOR(inbfpt=1; inbfpt<arsize; inbfpt=inbfpt+1)
         array2(inbfpt) = inbfpt
# end initialize loop
inbfpt = 0
FOR(timlpr=1; timlpr<timctl; timlpr=timlpr+1)
  (# BEGIN main timing loop
   CALL kernel
  )# END main timing loop
WRITE(5,2)chrbuf, intnum
  # put out record values and end message on console
#FORMAT Statements:
  1 format(' begin benchmark execution')
  2 format(1x,16A1,I4, ' end execution')
STOP
END
#
#define end of Main Program
#
# Subroutines and Functions:

FUNCTION setbyt(dumars)
LOGICAL inbuff, dumars  # dumars is used only because a parameter is required for a FUNCTION call
INTEGER otbf#, inbf#, loc#, t

COMMON //otbf#, inbuff(bufmax), inbf#

loc# = inbf#
inbf# = inbf# + 1
IF(inbf# .GT. bufmax)
inbf# = 1
setbyt = inbuff(loc#)

RETURN
END

SUBROUTINE putbyt(inchar, putbuf)
LOGICAL inchar, putbuf(bufmax)
INTEGER otbf#
COMMON //otbf#

putbuf(otbf#) = inchar
otbf# = otbf# + 1
IF(otbf# .GT. bufmax)
otbf# = 1

RETURN
END

SUBROUTINE kernel
LOGICAL newchr, inbuff, temp, bitmask, chrbuf, otbuf(bufmax), start
INTEGER otbf#, inbf#, intnum, tim#, pr, l#cnt#, whlcnt, opract, ars2, array2, array1(arsize)

COMMON //otbf#, inbuff(bufmax), inbf#, chrbuf(bufmax), array2(arsize), temp, intnum, bitmask

DATA start/stdcd/

FOR(1#cnt# = 1; 1#cnt# <= arsize; 1#cnt# = 1#cnt# + 1)
   #BEGIN I/O Loop
   temp = (.NOT. tstbyt)
   CALL out(temp, iomr)
   #END I/O Loop

FOR(1#cnt# = 1; 1#cnt# <= arsize; 1#cnt# = 1#cnt# + 1)
   #BEGIN Array Loop
   ars2 = (arsize + 1) - 1#cnt#
   array1(1#cnt#) = array2(ars2)
FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)

FOR(I=cnt=1; I<=nmsgs; I=cnt+1)
Tab 8
PASCAL/NT Source
(* DCT Benchmark Program in Pascal/MT for CP/M *)
(* written by : Capt B.F. BRADY, U.S.M.C. *)

PROGRAM DCTBENCHMARK;

CONST
  arraysiz* = 1251;
  numbermsv* = 2001;
  opracloops = 1001;
  import = 2001;
  ioloop* = 5751;
  timinscntrl = 121;
  testbyte = 851;
  integer1 = 3001;
  integer2 = -1501;
  bitmask = 11;
  msvlength = 801;
  startcode = 'S';
  startbyte = 101;
  buffermax = 151;

TYPE
  buffertype = PACKED ARRAY[0..buffermax] of CHAR;
  mixedtype = RECORD
    charbuffer : buffertype;
    intnumber : INTEGER
  END (*RECORD mixedtype*)

VAR
  timinsclooper, inbufptr, outbufptr : INTEGER;
  array1, array2 : ARRAY[0..arraysiz*] of INTEGER;
  inbuffer : buffertype;
  mixrec : mixedtype;

FUNCTION setbyte : CHAR;
BEGIN
  setbyte := inbuffer[inbufptr];
  inbufptr := inbufptr + 1;
  IF inbufptr > buffermax THEN inbufptr := 0
END (*setbyte*);

PROCEDURE putbyte (VAR inchar : CHAR; VAR putbuffertype);
BEGIN
  putbuffertype[outbufptr] := inchar;
  outbufptr := outbufptr + 1;
  IF outbufptr > buffermax THEN outbufptr := 0
END (*putbyte*);

PROCEDURE LFTROL(VAR temp : INTEGER; bits : INTEGER);
BEGIN
  INLINE("LDA / bits/", "MOV B,A /", "LDA / temp/", "LROT", "RLC /");
END (*LFTROL*);
PROCEDURE kernel;

VAR outbuffer : buffertype;
loopcounter, whilecounter, opaction, temp : INTEGER;
newchar : CHAR;

BEGIN
FOR loopcounter := 0 TO 1000 DO
BEGIN
    temp := testbyte;
    (* use in-line direct code to complement temp and put *)
    (* it out the specified port *)
    INLINE( "LDA /temp/
    "CMA /
    "OUT /ioport);)
END (*FOR#);

FOR loopcounter := 0 TO arraysize DO
    array[loopcounter] := array2[arraysize - loopcounter];

FOR loopcounter := 0 TO numbermsgs DO
BEGIN
    REPEAT
        newchar := setbyte
        UNTIL newchar = startcode;
    whilecounter := 0;

    WHILE whilecounter < msglength DO
    BEGIN
        newchar := setbyte;
        temp := SHR(newchar,1);
        IF (ODD(temp) AND ODD(bitmask)) THEN
            putbyte(newchar, outbuffer)
        ELSE
            putbyte(newchar, outbuffer);
        whilecounter := whilecounter + 1;
    END (* WHILE#);
END (* FOR#);

opaction := 0;
FOR loopcounter := 0 TO opactionloops DO
BEGIN
    temp := testbyte;
    CASE opaction OF
        0, 3, 9 : mixrec.charbuffer[opaction] := CHR(SHR(temp, 3));
        1, 4, 7 : BEGIN
            LFTROL(temp, 2);
    END (* FOR#);
mixrec.charbuffer[opraction] :=
CHR(ODD(temp) AND ODD(bitmask))
END;

2.6 : mixrec.charbuffer := inbuffer;

ELSE
mixrec.intnumber := (((integer1 DIV integer2)*integer1)
DIV integer2)*integer2) + integer1
END (*CASE*);

opraction := opraction + 1;
IF opraction > 9 then opraction := 0
END (*FOR*);
END (*kernel*);

BEGIN (* MAIN PROGRAM EXECUTION *)
WRITELN(' Begin Benchmark Execution');

FOR timinglooper := 0 TO arraysize DO
array2[timingslooper] := timingslooper;

FOR timingslooper := 0 TO buffermax DO
inbuffer[timingslooper] := 'A';
inbuffer[startbyte] := startcode;
inbufptr := 0;
outbufptr := 0;
FOR timingslooper := 0 TO timingscnt1 DO
  kernel1;
  writeln(mixrec.charbuffer,mixrec.intnumber);
END.
WRITELN(' End Benchmark Execution');
| Tab 8 |
PASCAL/MT Source |
(* Intersperse pascal source in asml source *)

(* DCT HOL BENCHMARK in Pascal/Z from Ithaca Intersystems *)
(* Written by: Capt B.F. Brady, U.S.M.C. *)

PROGRAM DCTBENCHMARK:

CONST
  arraysize = 125;
  numbermss = 200;
  operacloops = 100;
  iolops = 200;
  timincntr = 2;
  testbyte = 85;
  integer1 = 300;
  integer2 = -150;
  bitmask = 1;
  msslength = 80;
  startbyte = 10;
  buffermax = 151;

TYPE
  buffertype = ARRAY[0..buffermax] of CHAR;
  mixedtype = RECORD
    charbuffer : buffertype;
    intnumber : INTEGER
  END (*RECORD mixedtype*)

VAR
  timingslooper : INTEGER;
  inbufptr, outbufptr : INTEGER;
  array1, array2 : ARRAY[0..arraysize] of INTEGER;
  inbuffer : buffertype;
  mixrec : mixedtype;

FUNCTION setbyte : CHAR;

VAR
  localpnt : INTEGER;

BEGIN
  localpnt := inbufptr;
  inbufptr := inbufptr + 11;
  IF inbufptr > buffermax THEN inbufptr := 0;
  setbyte := inbuffer[localpnt];
END (*setbyte*);

PROCEDURE putbyte (VAR inchar : CHAR; VAR putbuffer : buffertype);

BEGIN
  putbuffer[outbufptr] := inchar;
  outbufptr := outbufptr + 11;
  IF outbufptr > buffermax THEN outbufptr := 0;
END (*putbyte*);
PROCEDURE ANDCHR(VAR temp : INTEGER; mask : INTEGER); EXTERNAL;
PROCEDURE OUTPUT(VAR temp : INTEGER; port : INTEGER); EXTERNAL;
PROCEDURE LFTROL(VAR temp : INTEGER; bits : INTEGER); EXTERNAL;
PROCEDURE RTSHFT(VAR temp : INTEGER; bits : INTEGER); EXTERNAL;

PROCEDURE kernel;

VAR
outbuffer : buffer_type;
loopcounter, whilecounter : INTEGER;
opraction, temp : INTEGER;
newchar : CHAR;

BEGIN
writeln(‘kernel’);
writeln(‘ioloops’):
FOR loopcounter := 0 TO ioloops DO
BEGIN
   temp := testbyte;
   OUTPUT(temp, iomport)
END (*FOR*);

writeln(‘arrayloops’):
FOR loopcounter := 0 TO arraysize DO
array1[loopcounter] := array2[arraysize - loopcounter];

writeln(‘mssloops’):
FOR loopcounter := 0 TO numbermss DO
BEGIN
   REPEAT
      newchar := setbyte
   UNTIL newchar = ‘$’;

   whilecounter := 0;
   WHILE whilecounter < msslensth DO
      BEGIN
         newchar := setbyte;
         temp := ord(newchar);
         RTSHFT(temp, 1);
         ANDCHR(temp, bitmask);
         IF temp = 1 THEN
            putbyte(newchar, outbuffer)
         ELSE
            putbyte(newchar, outbuffer);
         whilecounter := whilecounter + 1
      END (*WHILE*)
   END (*FOR*)

opraction := 01;
writeln(‘opacloops’):
FOR loopcounter := 0 TO operacloops DO
BEGIN
  temp := testbyte;

  (*$J9 Compiler option to create CASE Jump Table for 0..9 *)
  CASE opraction OF
    0, 3, 9 : BEGIN
      RTSHFT(temp, 3);
      mixrec.charbuffer[opraction] := chr(temp)
    END;
    1, 4, 7 : BEGIN
      LFTROL(temp, 2);
      ANDCHR(temp, bitmask);
      mixrec.charbuffer[opraction] := chr(temp)
    END;
    2, 6 : mixrec.charbuffer := inbuffer;
    ELSE:
      mixrec.intnumber := (((intser1 DIV integer2)*intser1) DIV integer2)*intser1
    END
END (*CASE*);

  opraction := opraction + 1;
  IF opraction > 9 then opraction := 0
END (*FOR*);

BEGIN (* MAIN PROGRAM EXECUTION *)
  WRITELN(' Begin Benchmark Execution);

  FOR timinslooper := 0 TO arraysize DO
    array2[timinslooper] := timinslooper;

  FOR timinslooper := 0 TO buffermax DO
    inbuffer[timinslooper] := 'A';

  inbuffer[startbyte] := 'S';
inbuffptr := 0;
outbuffptr := 0;
  FOR timinslooper := 0 TO timinscntrl DO
    kernel1;
    WRITELN(mixrec.charbuffer, mixrec.intnumber);
    WRITELN(chr(7), ' End Benchmark Execution');

END.
Tab 10
PLI-80 Source (Version 1)
/* BENCHMARK PROGRAM IN PLI/80 FOR CP/M */
/* Written by: Capt B.F. BRADY, U.S.M.C */

BENCHMARK_PLI_80:
PROCEDURE OPTIONS(MAIN);
/*REPLACE /* DEFINE CONSTANTS */
opr_action_loops BY 100,
timn_control BY 1,
array_size BY 125,
io_loops BY 575,
io_port BY 200,
number_msgs BY 200,
test_byte BY '55'B4,
buffer_max BY 15,
integer1 BY 300,
integer2 BY 150,
bias_mask BY '1'B,
msg_length BY 80,
start_byte BY 10,
start_code BY 'S';

/*
DEFINE EXTERNAL PROCEDURE ENTRIES */
DCL
OUTPUT ENTRY(BIT(8), BIT(8)),
RTSHFT ENTRY(BIT(8), FIXED(7)),
LFTROT ENTRY(BIT(8), FIXED(7));

/*
DEFINE VARIABLES FOR MAIN PROGRAM:
*/
DCL
1 mixed_type STATIC,
2 character_buffer(0:buffer_max) CHAR(1),
2 integer_number FIXED BINARY;
DCL
input_buffer(0:buffer_max) CHAR(1) INITIAL((10)'A', 'S', (5)'A') STATIC,
out_buffer_pointer FIXED BINARY(7) STATIC,
in_buffer_pointer FIXED BINARY(7) STATIC,
timn_loop_counter FIXED BINARY STATIC,
i FIXED BINARY(7) STATIC,
array_2(array_size) FIXED BINARY STATIC;

/*
PROCEDURE AND FUNCTION DEFINITIONS:
*/
PUT_BYTE:
PROCEDURE(in_character, put_buffer);
DCL
in_character CHAR(1),
put_buffer(0:buffer_max) CHAR(1);
BEGIN:
put_buffer(out_buffer_pointer) = in_character;
out_buffer_pointer = out_buffer_pointer + 11
IF out_buffer_pointer > buffer_max THEN
  out_buffer_pointer = 0;
END;
END PUT_BYTE;

GET_BYTE:
PROCEDURE RETURNS(CHAR(1)); /* A FUNCTION DEFINITION */
DCL
  local_pointer FIXED BINARY(7);
BEGIN:
  /* SAVE INDEX IN local_pointer FOR THE RETURN */
  local_pointer = in_buffer_pointer;
  in_buffer_pointer = in_buffer_pointer + 11;
  IF in_buffer_pointer > buffer_max THEN
    in_buffer_pointer = 0;
  END;
  RETURN(input_buffer(local_pointer));
END GET_BYTE; /* END FUNCTION GET_BYTE */

KERNEL:
PROCEDURE:
DCL
  out_buffer(0:buffer_max) CHAR(1) STATIC,
  loop_counter FIXED BINARY STATIC,
  array_1(array_size) FIXED BINARY STATIC,
  new_character CHAR(1) STATIC,
  temp FIXED BINARY(7) STATIC,
  operator_action FIXED BINARY(7) STATIC,
  while_counter FIXED BINARY(7) STATIC;
BEGIN:
  DO loop_counter = 0 TO io_loops;
    CALL output(io_port,test_byte);
  END; /* I/O Loop */
  DO loop_counter = 0 TO array_size;
    array_1(loop_counter)=array_2(array_size - loop_counter);
  END; /* Array Loop */
  DO loop_counter = 0 TO number_mess;
    DO WHILE(get_byte() = start_code);
      END; /* DO WHILE */
    while_counter = 0;
  DO WHILE(while_counter < mess_length);
    new_character = get_byte();
    temp = UNSPEC(new_character);
    CALL rtshift(1,temp);
    IF (UNSPEC(temp) & bit_mask) THEN
      CALL put_byte(new_character,out_buffer);
    ELSE
      CALL put_byte(new_character,out_buffer);
    while_counter = while_counter + 1;
  END; /* WHILE Loop */
END; /* Message Loop */
operator_action = 0;
DO loop_counter = 0 TO opr_action_loops;
  put skip list(operator_action,mixed_type,character_buffer(operator_action,
GOTO CASE(operator_action)

CASE(0): 1
CASE(3): 1
CASE(9):
    temp = test_byte;
    CALL rtshift(3,temp);
    UNSPEC(mixed_type.character_buffer(operator_action)) = UNSPEC(temp);
    GOTO CASE(0)

CASE(1): 1
CASE(4): 1
CASE(7):
    temp = test_byte;
    CALL lftrot(2,temp);
    UNSPEC(mixed_type.character_buffer(operator_action)) =
        (UNSPEC(temp) & bit_mask);
    GOTO CASE(9)

CASE(2): 1
CASE(6):
    mixed_type.character_buffer = input_buffer;
    GOTO CASE(1)

CASE(5):
CASE(8):
    mixed_type.integer_number =
        (((integer1/(-integer2))*integer1)/(-integer2))*(integer2)+integer1;
END_CASE:

    put skip list(operator_action,mixed_type.character_buffer(operator_action)
        operator_action = operator_action + 1;
    IF operator_action > 9 THEN operator_action = 0;
    END /* Operator Action Loop */
END;
END KERNEL;

/*************** BEGIN main Program Execution ***********/

PUT SKIP LIST(' Begin Benchmark Execution');
input_buffer(start_byte) = start_code;
DO timings_loop_counter = 0 TO array_size;
    array_2(timings_loop_counter) = timings_loop_counter;
END; /* initialize array_2 */
DO timings_loop_counter = 0 TO timings_control;
    CALL kernel;
END; /* main timings loop */
PUT SKIP;
DO i=1 TO buffer_max;
    put list(mixed_type.character_buffer(i));
END;
PUT LIST(mixed_type.integer_number);
PUT SKIP LIST(End Execution);
END;
END BENCHMARK_PLI_901 /* Main Program */
Tab 11
PLI-80 Source (Version 2)
FILENAM Ex: BMPLIS.PLI 17-NOV-80

**MARK: PROCEDURE OPTIONS (MAIN);

/
*************** DOCUMENT BENCHMARK PROGRAM IN PL/I-80 **************/

* * WRITTEN BY: F. P. MACLACHLAN *
* * DATE: 14-OCT-80 *

* *************** DOCUMENT BENCHMARK PROGRAM IN PL/I-80 **************/

#REPLACE
ARRAY_SIZE BY 125,
BITMASK BY '1 9',
BUFFER_MAX BY 15,
INTEGER1 BY 300,
INTEGER2 BY (-150), WOw'T ACCEPT (-150)!!
/*
**
IO_LOOPS BY 575,
IO_PORT BY 200, /* ADJUST */
MSG_LENGTH BY 80,
NUMBER_MSGS BY 200,
CPM_ACTION_LOOPS BY 100,
START_CODE BY 'S',
TEST_BYTE BY '55'B4, /* 95 DECIMAL */
TIMING_CONTROL BY 12;

DCL
SHIFTR ENTRY (BIT(8), FIXED(7)) RETURNS (BIT(9)),
ROTATL ENTRY (BIT(8), FIXED(7)) RETURNS (BIT(9)),
OUTPUT ENTRY (FIXED(7), BIT(8));

DCL
NEW_CHAR CHAR,
OUT_BUFFER(0:15) CHAR STATIC, /* NOTE: COULD USE CHAR(16) */
ARRAY1(0:ARRAY_SIZE) FIXED BINARY,
ARRAY2(0:ARRAY_SIZE) FIXED BINARY,
IN_BUFFER(0:15) CHAR STATIC INITIAL
IN_BUFFER_PTR FIXED BINARY STATIC INITIAL (0),
LOOP_COUNTER FIXED BINARY,
OFR_ACTION FIXED BINARY,
OUT_BUFFER_PTR FIXED BINARY STATIC INITIAL (0),
TIMING_COUNTER FIXED BINARY,
WHILE_COUNTER FIXED BINARY,

1 MIXED TYPE STATIC,
2 CHAR_BUFFER(0:15) CHAR,
2 INT_NUMBER FIXED BINARY;

GET_BYTE:
PROCEDURE RETURNS (CHAR);
DCL C CHAR;
C = IN BUFFER(IN_BUFFER_PTR);
IN_BUFFER_PTR = IN_BUFFER_PTR + 1;
IF IN_BUFFER_PTR > BUFFER_MAX THEN
IN_BUFFER_PTR = 0;
RETURN (C);
END GET_BYTE;

PUT_BYTE:
PROCEDURE (IN_CHAR, PUT_BUFFER);
DCL
IN_CHAR CHAR,
PUT_BUFFER(2:BUFFER_MAX) CHAR;
PUT_BUFFER(OUT_BUFFER_PTR) = IN_CHAR;
OUT_BUFFER_PTR = OUT_BUFFER_PTR + 1;
IF OUT_BUFFER_PTR > BUFFER_MAX THEN
OUT_BUFFER_PTR = 0;
END PUT_BYTE;

KERNEL:
PROC;
DCL
OUT_BUFFER(2:BUFFER_MAX) CHAR;
DO LOOP_COUNTER = 0 TO 10 LOOPS;
CALL OUTPUT(IO_PORT, TEST_BYTE);
END;
DO LOOP_COUNTER = 0 TO ARRAY_SIZE;
ARRAY1(LOOP_COUNTER) = ARRAY2(ARRAY_SIZE-LOOP_COUNTER);
END;
DO LOOP_COUNTER = 0 TO NUMBER_MGS;
NEW_CHAR = GET_BYTE();
DO WHILE (NEW_CHAR = START_CODE);
NEW_CHAR = GET_BYTE();
END;
WHILE_COUNTER = 0;
DO WHILE (WHILE_COUNTER < MSG_LENGTH);
NEW_CHAR = GET_BYTE();
IF SHIFTR(UNSPEC(NEW_CHAR), 1) & '1' THEN
CALL PUT_BYTE(NEW_CHAR, OUT_BUFFER);
ELSE
CALL PUT_BYTE(NEW_CHAR, OUT_BUFFER);
WHILE_COUNTER = WHILE_COUNTER + 1;
END; /* DO WHILE */
END; /* DO LOOP_COUNTER */

OPR_ACTION = 0;
DO LOOP_COUNTER = 0 TO OPR_ACTION_LOOPS;
GOTO CASE(OPR_ACTION);
CASE(2):;
CASE(3):;
CASE(9):;
UNSPEC(MIXED_TYPE.CHAR_BUFFER(OPR_ACTION)) =
SHIFT3(TEST_BYTE, 3);
GOTO END_CASE;

CASE(1):;
CASE(4):;
CASE(7):;
UNSPEC(MIXED_TYPE.CHAR_BUFFER(OPR_ACTION)) =
ROAT3L(TEST_BYTE, 2) & BITMASK;
GOTO END_CASE;

CASE(2):;
CASE(6):;
MIXED_TYPE.CHAR_BUFFER = IN_BUFFER;
GOTO END_CASE;

CASE(3):;
CASE(9):;
MIXED_TYPE.INT_NUMBER =
(((INTEGER1 / (-150)) * INTEGER1) / (-150)) * (-150)) + INTEGER1;
END_CASE:

IF OPR_ACTION > 9 THEN
OPR_ACTION = 0;
END; /* DO LOCP_COUNTER = 0 TO OPR_ACTION_LOOPS */

END KERNEL;

******************************************************************************
* START OF MAIN PROGRAM *
******************************************************************************

DCL

I FIXED BINARY;

PUT SKIP LIST ('BEGIN BENCHMARK EXECUTION');
DO I = 3 TO BUFFER_MAX;
  ARRAYS(I) = I;
END;

DO TIMING_LUPPER = 0 TO TIMING_CONTROL;
  CALL KERNEL();
END;

DO I = 3 TO 15;
  PUT LIST (MIXED_TYPE.CHAR_BUFFER(I));
END;

PUT LIST
  (MIXED_TYPE.INT_NUMBER,
   'END EXECUTION');
END BMARK;
<table>
<thead>
<tr>
<th>Tab 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLMX Source</td>
</tr>
</tbody>
</table>
bmark: do;
declare
equ literally 'literally',
ARRAY$SIZE equ '126', /* 0..125 */
BITMASK equ '01h',
BUFFER$SIZE equ '16', /* 0..15 */
CR equ '0Dh',
INT1 equ '300',
INT2 equ '-150',
10%LOOPS equ '975',
10%PORT equ '200', /* adjust */
LP equ '0Ah',
MSQ$LENQTH equ '80',
NUMBER$MSGS equ '80',
OPIACTION$LOOPS equ '100',
START$CODE equ '053h', /* ascii big S */
TEST$BYTE equ '85',
TIMING$CONTROL equ '127'
declare
array1(ARRAY$SIZE) address,
array2(ARRAY$SIZE) address,
i address,
in$buffer(BUFFER$SIZE) byte initial ('AAAAAAAAAAAAAAAA'),
in$buffer$ptr address initial (0),
out$buffer$ptr address initial (0),
timing$Idofer address);
declare
mixed type structure (
char$buffer(BUFFER$SIZE) byte,
int$number address )
);
nmout: procedure (value, base, lc, buf, width) external;
declare
(value, buf) address,
(base, lc, width) byte;
end nmout;
/* signed 16 bit divide (d2 / d1) */
divid: procedure (dl, d2) address external;
declare
(d1, d2) address;
end divid;
/* print character string until $ encountered */
pr$buf: procedure (strp) external;
declare
strp address;
end pr$buf;

/* print character at console */
wr$con: procedure (ch) external;
declare
ch byte;
end wr$con;

/* print address variable at console with one leading space */
putdec: procedure (num);
declare
num address,
buf(7) byte; /* scratch buffer for conversion */
call nmout(num, 10, ' ', buf, 0);
buf(6) = '$';
call pr$buf(buf);
end putdec;

/* get the next byte from in$buffer, increment in$buffer$ptr */
get$byte: procedure byte;
declare
c byte;

c = in$buffer(in$buffer$ptr);
in$buffer$ptr = in$buffer$ptr + 1;
if in$buffer$ptr > last(in$buffer) then
  in$buffer$ptr = 0;
return c;
end get$byte;

/* put character into buffer, increment out$buffer$ptr */
put$byte: procedure (in$char, pb);
declare
in$char byte,
(pb, pc) address,
put$char based pc byte;

pc = pb + out$buffer$ptr; /* compute ptr to next cell in buffer */
put$char = in$char;
out$buffer$ptr = out$buffer$ptr + 1;
if out$buffer$ptr > BUFFER SIZE - 1 then
  out$buffer$ptr = 0;
end put$byte;

/* kernel is the main procedure in the benchmark program. */
/* kernel procedure. */
(temp1, temp2) address,
i address,
loop$counter address,
new$char byte,
opr$action address,
out$buffer(BUFFER$SIZE) byte,
while$counter address;

if loop$counter = 0 to 100*LOOPS;
output(IO$PORT) = not TEST$BYTE;
end;

if loop$counter = 0 to last(array1);
array1(loop$counter) = array2(ARRAY$SIZE-loop$counter);
end;

if loop$counter = 0 to NUMBER$MSGS;
do while ((new$char := get$byte) <>' START$CODE);
end;

while$counter = 0;
do while (while$counter < MSG$LENGTH);
if shr((new$char := get$byte), 1) then /* tests only bit 0 */
call put$byte(new$char, .out$buffer);
else
call put$byte(new$char, .out$buffer);
while$counter = while$counter + 1;
end; /* do while */
end; /* do loop$counter */
opr$action = 0;
do loop$counter = 0 to OPR$ACTION$LOOPS;
if opr$action = 0 or opr$action = 3 or opr$action = 9 then
mixed$type.char$buffer(opr$action) =
shr(TEST$BYTE, 3);
else if opr$action = 1 or opr$action = 4 or opr$action = 7 then
mixed$type.char$buffer(opr$action) =
rol(TEST$BYTE, 2) and BITMASK;
else if opr$action = 2 or opr$action = 6 then
do i = 0 to last(mixed$type.char$buffer);
mixed$type.char$buffer(i) = in$buffer(i);
end;
else do; /* default */

end kernel;

end; /* do loop$counter = 0 to OPR$ACTION$LOOPS */

start of main program *

call pr$buf('.('Begin benchmark execution', CR, LF, 'S'));
do i = 0 to last(array2);
array2(i) = i;
end;
do TIMING$loop$er = 0 to TIMING$CONTROL;
call kernel;
end;
do i = 0 to BUFFER$SIZE - 1;
call wr$con(mixed$type.char$buffer(i));
end;
call prbuf( ('End Execution', CR, LF, '\$'));

end bmark;
<table>
<thead>
<tr>
<th>Tab 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLZ Source</td>
</tr>
</tbody>
</table>
BEGIN GLOBAL (TO THIS MODULE) DATA DECLARATIONS:

INTERNAL

TIMINGLOOPER, LOOPCOUNTER, WHILECOUNTER, OPRACION, OUTBUFFERPTR, TEMPI, TEMPII : INTEGER;
ARRAY1, ARRAY2 : ARRAY[ARRAYSIZE + 1 : INTEGER];
INBUFFER : SUFFERTYPE;
NEWCHAR : CHAR;

MIXEDTYPE : RECORD(
CHARBUFFER : SUFFERTYPE;
INNUMBER : INTEGER;
) ; END RECORD MIXEDTYPE !

END DATA DECLARATIONS !
BEGIN INTERNAL PROCEDURE DECLARATIONS/DEFINITIONS !

GETBYTE PROCEDURE RETURNS(ReturnBYTE : CHAR);
ENTRY
RETURNBYTE := INBUFFER[INBUFFERPTR];
INBUFFERPTR := INBUFFERPTR + 1;
END GETBYTE;

PUTBYTE PROCEDURE (INCHAR : CHAR, CHAPTR : CHARBPTR) ;
ENTRY
CHAPTR := INCHAPTR + 1;
END PUTBYTE;

KERNEL PROCEDURE;
ENTRY
RETURNBYTE := INBUFFER[INBUFFERPTR];
RETURNBYTE := OUTSUFFERTYPE;
END KERNEL PROCEDURE;
LOCAL
INTEGER ! LOCAL COUNTER !
OUTBUFFER : SUFFERTYPE;
AND OUTBUFFER ARE AUTOMATIC/DYNAMICALLY ALLOCATED !
ENTRY
LOOPCOUNTER = 0;
DO
IF LOOPCOUNTER > 10000 THEN EXIT FI;
OUTPUT(NOT(278564), 10000);
LOOPCOUNTER = 1;
END
LOOPCOUNTER = 0;
DO
IF LOOPCOUNTER > ARRAYSIZE THEN EXIT FI;
ARRAY[LOOPCOUNTER] = ARRAY[ARRAYSIZE - LOOPCOUNTER];
END
LOOPCOUNTER = 0;
DO
IF LOOPCOUNTER > NUMBERMSGS THEN EXIT FI;
NEWCHAR := GETBYTE;
IF NEWCHAR = STARTCODE THEN EXIT FI;
WHILECOUNTER := 0;
DO
IF WHILECOUNTER > MSGLENGTH THEN EXIT FI;
NEWCHAR := GETBYTE;
IF (SHIFT(NEWCHAR, 1) AND BITMASK) = 1 THEN
PUTBYTE(NEWCHAR, #OUTBUFFER);
ELSE
PUTBYTE(NEWCHAR, #OUTBUFFER);
FI;
WHILECOUNTER += 1;
END
END OF FOR LOOPCOUNTER FROM 0 TO NUMBERMSGS LOOP
IF OPRACTION = 0;
LOOPCOUNTER := 0;
DO
IF LOOPCOUNTER > OPRACTIONLOOPS THEN EXIT FI;
IF OPRACTION
CASE 0, 3, 9 THEN MIXEDTYPE.CHARBUFFER[OPRACION] := SHIFT(123456, 3);
CASE 1, 4, 7 THEN MIXEDTYPE.CHARBUFFER[OPRACION] := ROTATL(123456, 2) AND BITMASK;
CASE 2, 6 THEN
I := 0;
DO IF I > BUFFERMAX THEN EXIT FI;
MIXEDTYPE.CHARBUFFER[I] := INBUFFER[I];
I += 1;
END ELSE
TEMP1 := DIVID(INTEGER2, INTEGER1) * INTEGER1;
TEMP2 := DIVID(INTEGER2, TEMP1);
MIXEDTYPE.INTNUMBER := TEMP2 + INTEGER2 + INTEGER1;
END
NOTE: ALTHOUGH NOT CLEAR FROM THE DOCUMENTATION, THE DIVIDE SHOWN BELOW WILL
 NOT WORK SINCE PLZ HANDLES NEGATIVE NUMBERS INTERNALLY AS THOUGH THEY WERE
 LARGE POSITIVE NUMBERS. FOR EXAMPLE, -(-2) IS EQUAL TO ZERO.
MIXEDTYPE.INTNUMBER := \(((\text{INTEGER1} * \text{INTEGER2}) + \text{INTEGER1}) * \text{INTEGER2}) * \text{INTEGER1};
END RESULT
END OPRATION := 1;
LOOPCOUNTER += 1;
IF OPRATION > 3 THEN OPRATION := 0 FI;

END KERNEL;

GLOBAL: *************** GLOBAL DEFINITION OF MAIN PROGRAM ***************

MAIN PROCEDURE

ENTRY

PUTSTRING (CONOUT, '#BEGIN BENCHMARK EXECUTION,%R/');

LOOPCOUNTER := 0;

IF LOOPCOUNTER > ARRAYSIZE THEN EXIT FI;

ARRAY[LOOPCOUNTER] := LOOPCOUNTER;

LOOPCOUNTER := 1;

INBUFFERPTR := 0;

IF INBUFFERPTR > BUFFERMAX THEN EXIT FI;

INBUFFER[INBUFFERPTR] := 'A';

INBUFFERPTR := 1;

INBUFFER[10] := STARTCODE;

INBUFFERPTR := 0;

OUTBUFFERPTR := 0;

TIMINGLOOPER := 0;

IF TIMINGLOOPER > TIMINGCONTROL THEN EXIT FI;

KERNEL;

TIMINGLOOPER := 1;

PUTCHARS(CONOUT, '#MIXEDTYPE,CHARBUFFER[0], BUFFERMAX + 1); PUTINTEGER(CONOUT, MIXEDTYPE, INTNUMBER);

PUTSTRING (CONOUT, '#%REND EXECUTION,%R/');

END MAIN;

END DCTBENCHMARK; ! END MODULE!
APPENDIX C

DCT/HOL STUDY BENCHMARK

PROGRAM RESULTS

This appendix contains information generated by running benchmark programs using the candidate languages.
<table>
<thead>
<tr>
<th>Language</th>
<th>Executive Time (Min: Sec)</th>
<th># Bytes Absolute Object Code (No I/O)</th>
<th>Program Support Environment</th>
<th>Compile Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactive C</td>
<td>58.5</td>
<td>1286</td>
<td>PDP-11/70 UNIX</td>
<td>45</td>
</tr>
<tr>
<td>Whitesmith C</td>
<td>1:00</td>
<td>2538</td>
<td>CP/M</td>
<td>5:17</td>
</tr>
<tr>
<td>FORTRAN-80</td>
<td>4.03</td>
<td>3570</td>
<td>CP/M</td>
<td>1:47</td>
</tr>
<tr>
<td>RATFOR</td>
<td>3:43</td>
<td>3925</td>
<td>CP/M</td>
<td>3:01</td>
</tr>
<tr>
<td>Pascal/MT</td>
<td>1:36</td>
<td>3298</td>
<td>CP/M</td>
<td>5:52</td>
</tr>
<tr>
<td>Pascal/Z</td>
<td>2:18</td>
<td>2304</td>
<td>CP/M</td>
<td>3:01</td>
</tr>
<tr>
<td>PLI-80</td>
<td>2:30</td>
<td>4514</td>
<td>CP/M</td>
<td>2:17</td>
</tr>
<tr>
<td>PLIX</td>
<td>59</td>
<td>1759</td>
<td>CP/M</td>
<td>7:00</td>
</tr>
<tr>
<td>PLZ</td>
<td>2:48 (1)</td>
<td>2165</td>
<td>ZILOG</td>
<td>4:00</td>
</tr>
</tbody>
</table>

(1) Corrected for 2.5 MHZ Z80
<table>
<thead>
<tr>
<th>TIME</th>
<th>LANGUAGE</th>
<th>SIZE</th>
<th>LANGUAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>4:15</td>
<td></td>
<td>4514</td>
<td>PLI-80</td>
</tr>
<tr>
<td>4:03</td>
<td>FORTRAN-80</td>
<td>3925</td>
<td>RATFOR</td>
</tr>
<tr>
<td>3:43</td>
<td>RATFOR</td>
<td>3570</td>
<td>FORTRAN-80</td>
</tr>
<tr>
<td>3:29</td>
<td>RATFOR</td>
<td>3298</td>
<td>PASCAL/MT</td>
</tr>
<tr>
<td>2:48</td>
<td>PLZ</td>
<td>2538</td>
<td>WHITESMITH C</td>
</tr>
<tr>
<td>2:30</td>
<td>PLI-80</td>
<td>2304</td>
<td>PASCAL/Z</td>
</tr>
<tr>
<td>2:18</td>
<td>PASCAL/Z</td>
<td>2165</td>
<td>PLZ</td>
</tr>
<tr>
<td>1:36</td>
<td>PASCAL/MT</td>
<td>1759</td>
<td>PLMX</td>
</tr>
<tr>
<td>1:00</td>
<td>WHITESMITH C</td>
<td>1286</td>
<td>INTERACTIVE C</td>
</tr>
<tr>
<td>0</td>
<td>INTERACTIVE C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure C-1. Benchmark Results**

EXECUTION TIME

CODE SIZE (#BYTES ABSOLUTE OBJECT CODE [NO I/O])
This appendix presents the statistical results of Phase 1 of the Delphi study. The object of Phase 1 was to assign weights to the various language features.
Table D-1. Phase I Delphi Study Statistical Summary

<table>
<thead>
<tr>
<th>LANGUAGE FEATURE</th>
<th>1st Iteration</th>
<th>2nd Iteration</th>
<th>3rd Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>Std. Dev.</td>
<td>Average</td>
</tr>
<tr>
<td>DATA REPRESENTATION</td>
<td>121.3333</td>
<td>35.1034</td>
<td>2895</td>
</tr>
<tr>
<td>SYSTEMS PROGRAMMING</td>
<td>120.4444</td>
<td>81.6295</td>
<td>.3741</td>
</tr>
<tr>
<td>CONTROL STRUCTURES</td>
<td>115.0000</td>
<td>27.6514</td>
<td>.2401</td>
</tr>
<tr>
<td>PROGRAM EFF. EFF.</td>
<td>108.6667</td>
<td>57.6132</td>
<td>.3461</td>
</tr>
<tr>
<td>DOCUMENTATION</td>
<td>65.4444</td>
<td>28.2091</td>
<td>.4446</td>
</tr>
<tr>
<td>TESTABILITY</td>
<td>52.6667</td>
<td>20.8417</td>
<td>.3950</td>
</tr>
<tr>
<td>TIME EFFICIENCY</td>
<td>43.4444</td>
<td>19.7934</td>
<td>.4596</td>
</tr>
<tr>
<td>SPACE EFFICIENCY</td>
<td>40.2222</td>
<td>16.6713</td>
<td>.4145</td>
</tr>
<tr>
<td>EXITURE OF USE</td>
<td>51.0000</td>
<td>29.6564</td>
<td>.5815</td>
</tr>
<tr>
<td>ASSEMBLY LANGUAGE</td>
<td>50.3333</td>
<td>29.0831</td>
<td>.5937</td>
</tr>
<tr>
<td>TARGET CPU TRANSPORTABILITY</td>
<td>47.1111</td>
<td>32.6743</td>
<td>.6936</td>
</tr>
<tr>
<td>LEARNABILITY</td>
<td>34.7778</td>
<td>18.1437</td>
<td>.8217</td>
</tr>
<tr>
<td>HSC RED (2ND) INTR. SYN USE</td>
<td>32.7778</td>
<td>20.6599</td>
<td>.6241</td>
</tr>
<tr>
<td>MULTIVARIATE</td>
<td>24.3333</td>
<td>16.2635</td>
<td>.6604</td>
</tr>
<tr>
<td>RELIABILITY &amp; SECURITY</td>
<td>32.0000</td>
<td>12.5091</td>
<td>.3934</td>
</tr>
<tr>
<td>BORABLE OBJECT CODE</td>
<td>17.0000</td>
<td>5.8737</td>
<td>.3455</td>
</tr>
<tr>
<td>COMPIL-TIME EFFICIENCY</td>
<td>20.0000</td>
<td>9.4116</td>
<td>.4171</td>
</tr>
<tr>
<td>FOR SOFTWARE TRANSPORTABILITY</td>
<td>17.4444</td>
<td>8.9450</td>
<td>.5129</td>
</tr>
</tbody>
</table>
Figure D-1. Ranking of Features
### Table D-2. First Iteration DCT HOL Language Feature Response Summary

<table>
<thead>
<tr>
<th>LANGUAGE FEATURE</th>
<th>RESPONSES - ORDERED LOW TO HIGH</th>
<th>TOTAL</th>
<th>AVERAGE</th>
<th>STD. DEV.</th>
<th>ST. DEV./ AVERAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEMS PROGRAMMING</td>
<td>05 90 95 110 120 130 150 190 200</td>
<td>1156</td>
<td>128.4444</td>
<td>41.6297</td>
<td>.3241</td>
</tr>
<tr>
<td>DATA REPRESENTATION</td>
<td>75 85 90 100 120 150 152 170</td>
<td>1092</td>
<td>121.3333</td>
<td>35.1034</td>
<td>.2893</td>
</tr>
<tr>
<td>CONTROL STRUCTURES</td>
<td>80 85 100 100 110 120 150 150 160</td>
<td>1035</td>
<td>115.0000</td>
<td>27.6134</td>
<td>.2401</td>
</tr>
<tr>
<td>PROGRAM SUPP. ENV.</td>
<td>70 80 80 88 95 100 160 175</td>
<td>978</td>
<td>108.6667</td>
<td>37.6132</td>
<td>.3461</td>
</tr>
<tr>
<td>DOCUMENTATION</td>
<td>30 35 46 50 60 65 70 105 110</td>
<td>571</td>
<td>63.4444</td>
<td>28.2051</td>
<td>.4446</td>
</tr>
<tr>
<td>READABILITY</td>
<td>15 35 40 50 55 60 70 75 83</td>
<td>474</td>
<td>52.6667</td>
<td>20.8447</td>
<td>.3958</td>
</tr>
<tr>
<td>INTENT OF USE</td>
<td>1 10 35 50 63 70 75 75 75</td>
<td>459</td>
<td>51.0000</td>
<td>29.6564</td>
<td>.5815</td>
</tr>
<tr>
<td>ASSEMBLY LANG. LINKAGE</td>
<td>20 21 25 40 45 52 55 90 105</td>
<td>453</td>
<td>50.3333</td>
<td>29.8831</td>
<td>.5937</td>
</tr>
<tr>
<td>TARGET CPU TRANSPORTABILITY</td>
<td>5 20 20 29 45 60 65 70 110</td>
<td>424</td>
<td>47.1111</td>
<td>32.6743</td>
<td>.6936</td>
</tr>
<tr>
<td>TIME EFFICIENCY</td>
<td>15 20 25 35 55 55 55 60 70</td>
<td>391</td>
<td>43.4444</td>
<td>19.7934</td>
<td>.4556</td>
</tr>
<tr>
<td>SPACE EFFICIENCY</td>
<td>15 25 30 30 40 47 50 60 65</td>
<td>362</td>
<td>40.2222</td>
<td>16.6717</td>
<td>.4145</td>
</tr>
<tr>
<td>LEARNABILITY</td>
<td>13 15 15 35 35 35 50 50 65</td>
<td>313</td>
<td>34.7778</td>
<td>18.1437</td>
<td>.5217</td>
</tr>
<tr>
<td>HSC 800 (280) INST. SET USE</td>
<td>6 15 25 30 34 35 35 35 35 80</td>
<td>295</td>
<td>32.7778</td>
<td>20.4559</td>
<td>.6241</td>
</tr>
<tr>
<td>RETHRANCY &amp; RECURSION</td>
<td>15 20 28 30 30 35 35 35 60</td>
<td>288</td>
<td>32.0000</td>
<td>12.5897</td>
<td>.3934</td>
</tr>
<tr>
<td>COMPIL-TIME EFFICIENCY</td>
<td>10 12 15 15 18 20 20 30 40</td>
<td>180</td>
<td>20.0000</td>
<td>9.4736</td>
<td>.4737</td>
</tr>
<tr>
<td>PSE SOFTWARE TRANSPORTABILITY</td>
<td>2 10 15 15 20 20 20 20 20 35</td>
<td>157</td>
<td>17.4444</td>
<td>8.9458</td>
<td>.5128</td>
</tr>
<tr>
<td>ROMable OBJECT CODE</td>
<td>9 10 14 15 15 20 20 25 25</td>
<td>153</td>
<td>17.0000</td>
<td>5.8737</td>
<td>.3455</td>
</tr>
</tbody>
</table>
### Table D-3. Second Iteration DOT HOL Language Feature Response Summary

<table>
<thead>
<tr>
<th>LANGUAGE FEATURE</th>
<th>RESPONSES - ORDERED LOW TO HIGH</th>
<th>TOTAL</th>
<th>AVERAGE</th>
<th>STD.DEV.</th>
<th>STD.DEV./AVERAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA REPRESENTATION</td>
<td>100 105 110 120 120 120 168 170 180</td>
<td>1193</td>
<td>132.5556</td>
<td>31.0447</td>
<td>.2342</td>
</tr>
<tr>
<td>SYSTEMS PROGRAMMING</td>
<td>92 100 100 110 120 130 135 150 190</td>
<td>1127</td>
<td>125.2222</td>
<td>30.7481</td>
<td>.2455</td>
</tr>
<tr>
<td>CONTROL STRUCTURES</td>
<td>80 86 100 100 105 120 125 130 150 996</td>
<td>110.6667</td>
<td>22.4221</td>
<td>.2026</td>
<td></td>
</tr>
<tr>
<td>PROGRAM SUPP. ENV.</td>
<td>80 80 81 85 90 100 110 130 160 916</td>
<td>101.7778</td>
<td>27.5172</td>
<td>.2704</td>
<td></td>
</tr>
<tr>
<td>DOCUMENTATION</td>
<td>40 50 50 55 58 60 70 90 110 583</td>
<td>64.7778</td>
<td>22.1460</td>
<td>.3419</td>
<td></td>
</tr>
<tr>
<td>READABILITY</td>
<td>35 38 40 55 70 80 80 80 83 561</td>
<td>62.3333</td>
<td>20.3531</td>
<td>.3265</td>
<td></td>
</tr>
<tr>
<td>SPACE EFFICIENCY</td>
<td>30 30 35 40 50 50 75 75 80 465</td>
<td>51.6667</td>
<td>20.1556</td>
<td>.3901</td>
<td></td>
</tr>
<tr>
<td>TIME EFFICIENCY</td>
<td>20 35 35 50 52 54 55 75 80 456</td>
<td>50.6667</td>
<td>19.0919</td>
<td>.3768</td>
<td></td>
</tr>
<tr>
<td>ASSEMBLY LANG. LINKAGE</td>
<td>21 30 31 40 43 45 60 65 95 430</td>
<td>47.7778</td>
<td>22.5875</td>
<td>.4728</td>
<td></td>
</tr>
<tr>
<td>EXTENT OF USE</td>
<td>1 5 30 40 50 63 76 77 85 427</td>
<td>47.4444</td>
<td>30.9561</td>
<td>.6525</td>
<td></td>
</tr>
<tr>
<td>TARGET CPU TRANSPORTABILITY</td>
<td>5 15 30 30 40 45 55 59 65 344</td>
<td>38.2222</td>
<td>20.1791</td>
<td>.5279</td>
<td></td>
</tr>
<tr>
<td>LEARNABILITY</td>
<td>13 20 20 22 30 35 45 65 70 320</td>
<td>35.5556</td>
<td>20.4641</td>
<td>.5755</td>
<td></td>
</tr>
<tr>
<td>MCC 600 (280) INST. SET USE</td>
<td>6 25 30 30 30 30 33 34 50 80</td>
<td>35.3333</td>
<td>20.2176</td>
<td>.5722</td>
<td></td>
</tr>
<tr>
<td>BREADANCY &amp; RECURSION</td>
<td>13 15 25 28 30 30 35 35 35 246</td>
<td>27.3333</td>
<td>8.3217</td>
<td>.3045</td>
<td></td>
</tr>
<tr>
<td>MULTITASKING</td>
<td>10 12 15 15 20 23 25 30 48 198</td>
<td>22.0000</td>
<td>11.7047</td>
<td>.5320</td>
<td></td>
</tr>
<tr>
<td>COMPILE-TIME EFFICIENCY</td>
<td>10 10 10 12 17 18 20 25 30 152</td>
<td>16.8889</td>
<td>7.2015</td>
<td>.4264</td>
<td></td>
</tr>
<tr>
<td>COMRAble OBJECT CODE</td>
<td>4 5 10 10 14 15 25 25 30 138</td>
<td>15.3333</td>
<td>9.3274</td>
<td>.6083</td>
<td></td>
</tr>
<tr>
<td>PSE SOFTWARE TRANSPORTABILITY</td>
<td>2 6 10 10 12 15 20 20 30 130</td>
<td>14.4444</td>
<td>9.7225</td>
<td>.6731</td>
<td></td>
</tr>
<tr>
<td>RESPONSES</td>
<td>TOTAL AVERAGE</td>
<td>STDEV.</td>
<td>AVERAGE</td>
<td>STDEV.</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>---------------</td>
<td>--------</td>
<td>---------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>DATA REPRESENTATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>105</td>
<td>110</td>
<td>115</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>90</td>
<td>95</td>
<td>100</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>80</td>
<td>85</td>
<td>90</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>60</td>
<td>65</td>
<td>70</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>45</td>
<td>50</td>
<td>55</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>35</td>
<td>40</td>
<td>45</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>12</td>
<td>15</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>SYSTEMS PROGRAMMING</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>225</td>
<td>235</td>
<td>245</td>
<td>250</td>
<td>255</td>
<td></td>
</tr>
<tr>
<td>180</td>
<td>190</td>
<td>200</td>
<td>210</td>
<td>220</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>80</td>
<td>85</td>
<td>90</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>60</td>
<td>65</td>
<td>70</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>83.04</td>
<td>84.15</td>
<td>85.26</td>
<td>86.37</td>
<td>87.48</td>
<td></td>
</tr>
<tr>
<td>16.49</td>
<td>16.49</td>
<td>16.49</td>
<td>16.49</td>
<td>16.49</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3.17</td>
<td>3.17</td>
<td>3.17</td>
<td>3.17</td>
<td>3.17</td>
<td></td>
</tr>
<tr>
<td>7.82</td>
<td>7.82</td>
<td>7.82</td>
<td>7.82</td>
<td>7.82</td>
<td></td>
</tr>
</tbody>
</table>

Table D-4: Third Iteration DCT HOL Language Feature Response Summary
This appendix presents the statistical results of Phase 2 of the Delphi study. The object of Phase 2 was to assign gross scores and figures of merit to each language based upon the weights of the various language features assigned in Phase 1.
<table>
<thead>
<tr>
<th>Language Feature</th>
<th>APL</th>
<th>BASIC</th>
<th>C</th>
<th>COBOL</th>
<th>FORTRAN-66</th>
<th>FORTRAN-77</th>
<th>RATFOR</th>
<th>FORTRAN</th>
<th>PL/I-80</th>
<th>PLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Representation</td>
<td>59</td>
<td>33.60%</td>
<td>93.37%</td>
<td>79.06%</td>
<td>60.02%</td>
<td>66.75%</td>
<td>62.5%</td>
<td>114.46%</td>
<td>59.57%</td>
<td>110</td>
</tr>
<tr>
<td>System Programming</td>
<td>24.0</td>
<td>15.95%</td>
<td>71.03%</td>
<td>27.20%</td>
<td>47.82%</td>
<td>46.5%</td>
<td>55.8%</td>
<td>80.60%</td>
<td>94.26%</td>
<td>83.6%</td>
</tr>
<tr>
<td>Control Structures</td>
<td>62.0</td>
<td>34.10%</td>
<td>31.80%</td>
<td>57.5%</td>
<td>41.29%</td>
<td>72.76%</td>
<td>74.9%</td>
<td>86.93%</td>
<td>66.34%</td>
<td>89.80%</td>
</tr>
<tr>
<td>Program Supp. Rev.</td>
<td>30</td>
<td>72.06%</td>
<td>36.39%</td>
<td>53.2%</td>
<td>50.9%</td>
<td>57%</td>
<td>66.5%</td>
<td>74.41%</td>
<td>47.50%</td>
<td>76</td>
</tr>
<tr>
<td>Documentation</td>
<td>29.6</td>
<td>54.11%</td>
<td>56.11%</td>
<td>61.42%</td>
<td>52.64%</td>
<td>60.3%</td>
<td>53.65%</td>
<td>63.36%</td>
<td>30.21%</td>
<td>57.16%</td>
</tr>
<tr>
<td>Readability</td>
<td>6.44</td>
<td>37.40%</td>
<td>46.0544</td>
<td>53.32%</td>
<td>32.25%</td>
<td>37.74%</td>
<td>38.75%</td>
<td>56.39%</td>
<td>19.09%</td>
<td>55.26%</td>
</tr>
<tr>
<td>Time Efficiency</td>
<td>11.44</td>
<td>5.43%</td>
<td>36.600%</td>
<td>17.16%</td>
<td>33.0452%</td>
<td>26.455%</td>
<td>36.60%</td>
<td>35.1322%</td>
<td>43.170%</td>
<td>42.9%</td>
</tr>
<tr>
<td>Space Efficiency</td>
<td>22.2</td>
<td>15.06%</td>
<td>30.85%</td>
<td>19.425%</td>
<td>36.075%</td>
<td>31.9125%</td>
<td>40.2375%</td>
<td>35.6941%</td>
<td>44.0865%</td>
<td>36.85%</td>
</tr>
<tr>
<td>Reliability of Use</td>
<td>4.9</td>
<td>44.1%</td>
<td>28.5017%</td>
<td>39.2%</td>
<td>37.79%</td>
<td>35.32%</td>
<td>30.625%</td>
<td>35.525%</td>
<td>9.8000%</td>
<td>29.4%</td>
</tr>
<tr>
<td>Assembly Language Linkage</td>
<td>45.2</td>
<td>10.65%</td>
<td>29.70%</td>
<td>17.17%</td>
<td>34.252%</td>
<td>32.7%</td>
<td>32.77%</td>
<td>29.600%</td>
<td>31.6400%</td>
<td>32.77%</td>
</tr>
<tr>
<td>Target CPU Transport-ability</td>
<td>12.6</td>
<td>23.94%</td>
<td>22.5729%</td>
<td>16.557%</td>
<td>24.005%</td>
<td>20.679%</td>
<td>20.475%</td>
<td>15.5263%</td>
<td>22.0500%</td>
<td>17.325%</td>
</tr>
<tr>
<td>Locomobility</td>
<td>3.15</td>
<td>26.56%</td>
<td>24.6739%</td>
<td>21.00%</td>
<td>20.0245%</td>
<td>19.045%</td>
<td>25.2%</td>
<td>24.8052%</td>
<td>12.6000%</td>
<td>21.4062%</td>
</tr>
<tr>
<td>LOC SSD (SPL) Inst. Net Rev.</td>
<td>12.44</td>
<td>6.89%</td>
<td>14.5112%</td>
<td>6.92%</td>
<td>15.55%</td>
<td>15.95%</td>
<td>15.95%</td>
<td>16.9094%</td>
<td>10.3656%</td>
<td>11.4625%</td>
</tr>
<tr>
<td>Multitasking</td>
<td>2.73</td>
<td>13.79%</td>
<td>9.82%</td>
<td>5.5%</td>
<td>6.187%</td>
<td>8.25%</td>
<td>8.25%</td>
<td>7.5625%</td>
<td>15.5485%</td>
<td>9.1057%</td>
</tr>
<tr>
<td>Reactivity &amp; Recursion</td>
<td>0</td>
<td>3.32%</td>
<td>21.25%</td>
<td>0%</td>
<td>2.75%</td>
<td>1.6675%</td>
<td>1.6675%</td>
<td>10.5000%</td>
<td>16.2500%</td>
<td>16.25%</td>
</tr>
<tr>
<td>Dynamic Object Code</td>
<td>0</td>
<td>6.216%</td>
<td>11.1%</td>
<td>9.92%</td>
<td>10.04%</td>
<td>8.8%</td>
<td>9.25%</td>
<td>10.0460%</td>
<td>8.5250%</td>
<td>3.33%</td>
</tr>
<tr>
<td>Compile-Time Efficiency</td>
<td>4.25</td>
<td>3.35%</td>
<td>11.2375%</td>
<td>3.065%</td>
<td>9.82%</td>
<td>8.518%</td>
<td>8.921%</td>
<td>8.9900%</td>
<td>9.3700%</td>
<td>7.0007%</td>
</tr>
<tr>
<td>PMS Software Transportability</td>
<td>2.46</td>
<td>9.79%</td>
<td>9.94%</td>
<td>7.3996%</td>
<td>10.479%</td>
<td>8.917%</td>
<td>8.81%</td>
<td>10.3320%</td>
<td>7.9950%</td>
<td>9.225%</td>
</tr>
<tr>
<td>TOTAL</td>
<td>277.750%</td>
<td>414.5157%</td>
<td>428.3219%</td>
<td>421.1773%</td>
<td>554.3217%</td>
<td>599.8952%</td>
<td>610.2931%</td>
<td>663.5647%</td>
<td>59.1795%</td>
<td>708.5161%</td>
</tr>
</tbody>
</table>
Table E-2. Delphi Study—Phase 2 Summary (2nd Iteration)

<table>
<thead>
<tr>
<th>Language Feature</th>
<th>Interactive C</th>
<th>Flitmark's C</th>
<th>Microsoft FORTRAN</th>
<th>Zilog FORTRAN</th>
<th>UCSD FORTRAN</th>
<th>Cromemco FORTRAN</th>
<th>Pascal/MT</th>
<th>Pascal/2</th>
<th>FL1-80</th>
<th>FLAX</th>
<th>FL2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Representation</td>
<td>97.2362</td>
<td>92.0125</td>
<td>62.9062</td>
<td>52.25</td>
<td>59.125</td>
<td>60.75</td>
<td>110.8412</td>
<td>104.0437</td>
<td>110.55</td>
<td>69.4375</td>
<td>56.25</td>
</tr>
<tr>
<td>System Programming</td>
<td>112.6202</td>
<td>111.6</td>
<td>74.09</td>
<td>71.3</td>
<td>68.2</td>
<td>72.3250</td>
<td>110.5708</td>
<td>79.5600</td>
<td>76.80</td>
<td>10.05</td>
<td>77.75</td>
</tr>
<tr>
<td>Control Structures</td>
<td>90.0812</td>
<td>84.7119</td>
<td>53.5</td>
<td>51.36</td>
<td>57.76</td>
<td>75.6597</td>
<td>95.8497</td>
<td>86.0933</td>
<td>75.7801</td>
<td>86.9375</td>
<td>69.6</td>
</tr>
<tr>
<td>Program Support Rev.</td>
<td>71.9625</td>
<td>69.35</td>
<td>68.875</td>
<td>42.75</td>
<td>54.625</td>
<td>69.5115</td>
<td>76</td>
<td>74.1</td>
<td>76.95</td>
<td>68.875</td>
<td>52.25</td>
</tr>
<tr>
<td>Documentation</td>
<td>50.32</td>
<td>56.24</td>
<td>54.76</td>
<td>46.25</td>
<td>51.8</td>
<td>51.8</td>
<td>59.2</td>
<td>55.5</td>
<td>55.65</td>
<td>51.8</td>
<td>51.8</td>
</tr>
<tr>
<td>Readability</td>
<td>48.1438</td>
<td>47.18</td>
<td>56.2275</td>
<td>39.092</td>
<td>41.708</td>
<td>42.3676</td>
<td>59.2178</td>
<td>57.29</td>
<td>51.6754</td>
<td>49.7075</td>
<td>49.528</td>
</tr>
<tr>
<td>Extent of Use</td>
<td>26.46</td>
<td>25.725</td>
<td>36.75</td>
<td>18.0075</td>
<td>18.375</td>
<td>22.05</td>
<td>27.6017</td>
<td>22.05</td>
<td>23.52</td>
<td>40.0183</td>
<td>26.1317</td>
</tr>
<tr>
<td>Assembly Language Listings</td>
<td>42.94</td>
<td>38.42</td>
<td>36.16</td>
<td>21.47</td>
<td>21.47</td>
<td>25.312</td>
<td>25.312</td>
<td>32.77</td>
<td>43.6903</td>
<td>41.4503</td>
<td></td>
</tr>
<tr>
<td>Target CPU Transportability</td>
<td>18.1125</td>
<td>25.625</td>
<td>25.2</td>
<td>22.05</td>
<td>12.6</td>
<td>21.2625</td>
<td>22.48</td>
<td>20.79</td>
<td>24.1511</td>
<td>31.5</td>
<td>25.2</td>
</tr>
<tr>
<td>Multitasking</td>
<td>0</td>
<td>.6075</td>
<td>.9157</td>
<td>.9157</td>
<td>0</td>
<td>0</td>
<td>.6075</td>
<td>.9156</td>
<td>.9157</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Recomposability &amp; Recomposition</td>
<td>21.6675</td>
<td>18.875</td>
<td>.8325</td>
<td>1.25</td>
<td>0</td>
<td>0</td>
<td>20.625</td>
<td>23.325</td>
<td>18.325</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>DRI/able Object Code</td>
<td>0</td>
<td>0</td>
<td>13.32</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8.928</td>
<td>.928</td>
<td>7.955</td>
<td>14.8</td>
<td>14.8</td>
</tr>
<tr>
<td>TOTALS*</td>
<td>697.6675</td>
<td>627.5913</td>
<td>525.7007</td>
<td>419.3477</td>
<td>413.978</td>
<td>401.6906</td>
<td>695.9895</td>
<td>552.0749</td>
<td>608.7323</td>
<td>556.4605</td>
<td>600.2725</td>
</tr>
</tbody>
</table>

*These totals do not reflect the inclusion of space efficiency, time efficiency, or compile-time efficiency.
<table>
<thead>
<tr>
<th>Language</th>
<th>Interactive C</th>
<th>White-Smith's</th>
<th>Microsoft FORTRAN-DO</th>
<th>Crummoch ESROT</th>
<th>Pascal/RF</th>
<th>Pascal/5</th>
<th>PL/I-DO</th>
<th>PL/M</th>
<th>PL/I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Representation</td>
<td>115.895</td>
<td>110</td>
<td>62.9512</td>
<td>96.25</td>
<td>123.75</td>
<td>120.635</td>
<td>111.9525</td>
<td>79.5437</td>
<td>91.3275</td>
</tr>
<tr>
<td>System Programming</td>
<td>109.529</td>
<td>110.5994</td>
<td>51.6994</td>
<td>64.0504</td>
<td>108.6984</td>
<td>103.7304</td>
<td>94.3972</td>
<td>103.5352</td>
<td>98.1708</td>
</tr>
<tr>
<td>Control Structure</td>
<td>96.13</td>
<td>96.3</td>
<td>48.9097</td>
<td>81.7694</td>
<td>77.4904</td>
<td>76.1297</td>
<td>87.1194</td>
<td>87.1194</td>
<td>86.3397</td>
</tr>
<tr>
<td>Program Support Environment</td>
<td>63.703</td>
<td>70.033</td>
<td>70.033</td>
<td>70.033</td>
<td>70.033</td>
<td>70.033</td>
<td>70.033</td>
<td>70.033</td>
<td>70.033</td>
</tr>
<tr>
<td>Documentation</td>
<td>62.9</td>
<td>54.8704</td>
<td>57.72</td>
<td>59.94</td>
<td>61.42</td>
<td>62.9</td>
<td>64.02</td>
<td>51.8</td>
<td>59.9404</td>
</tr>
<tr>
<td>Readability</td>
<td>53.32</td>
<td>54.877</td>
<td>42.9609</td>
<td>49.9651</td>
<td>60.66</td>
<td>60.7543</td>
<td>55.8476</td>
<td>55.4567</td>
<td>56.8114</td>
</tr>
<tr>
<td>Space Efficiency</td>
<td>55.5</td>
<td>29.128</td>
<td>19.9911</td>
<td>18.1618</td>
<td>21.6394</td>
<td>30.9745</td>
<td>15.8119</td>
<td>40.376</td>
<td>32.9607</td>
</tr>
<tr>
<td>Extent of Use</td>
<td>20.5417</td>
<td>33.0904</td>
<td>41.2304</td>
<td>38.3017</td>
<td>33.0904</td>
<td>38.3017</td>
<td>33.4617</td>
<td>26.1317</td>
<td>35.525</td>
</tr>
<tr>
<td>Assembly Language Linkage</td>
<td>30.0945</td>
<td>29.7069</td>
<td>35.1802</td>
<td>27.4409</td>
<td>30.3472</td>
<td>37.1273</td>
<td>34.5418</td>
<td>34.8672</td>
<td>36.16</td>
</tr>
<tr>
<td>Target CPU</td>
<td>5.2478</td>
<td>12.6</td>
<td>23.0999</td>
<td>19.9409</td>
<td>21.109</td>
<td>11.8125</td>
<td>18.9</td>
<td>25.7229</td>
<td>6.615</td>
</tr>
<tr>
<td>Inst. Set Use</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Multithreading</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Containment &amp; Accrual</td>
<td>24.5825</td>
<td>24.5825</td>
<td>0</td>
<td>0</td>
<td>22.5</td>
<td>20.415</td>
<td>2.5</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>Compile-time Efficiency</td>
<td>14.5</td>
<td>2.0575</td>
<td>6.0989</td>
<td>3.6047</td>
<td>12.4963</td>
<td>4.763</td>
<td>1.5529</td>
<td>2.7167</td>
<td></td>
</tr>
<tr>
<td>TOTAL*</td>
<td>643.6798</td>
<td>508.6008</td>
<td>580.9999</td>
<td>570.7995</td>
<td>717.8094</td>
<td>718.6094</td>
<td>616.0994</td>
<td>619.1994</td>
<td>617.2994</td>
</tr>
<tr>
<td>FOR</td>
<td>750.6795</td>
<td>704.8101</td>
<td>587.9995</td>
<td>595.5995</td>
<td>708.6995</td>
<td>774.3333</td>
<td>659.3333</td>
<td>710.6995</td>
<td>672.0995</td>
</tr>
</tbody>
</table>

*Total without space, time, compile-time efficiency.
Table E-4. Delphi Study—Phase 2 Summary (3rd Iteration—NOSC Included)

<table>
<thead>
<tr>
<th>Language Feature</th>
<th>Interactive 0</th>
<th>Whitesmith's 0</th>
<th>Microsoft PERT-60</th>
<th>Greness RAPTOR</th>
<th>Pansel/HP</th>
<th>Pansel/S</th>
<th>PLA-60</th>
<th>PLA6</th>
<th>PLA6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Representation</td>
<td>115.0375</td>
<td>111.48</td>
<td>64.4775</td>
<td>95.1037</td>
<td>123.75</td>
<td>127.5725</td>
<td>110.0653</td>
<td>79.5437</td>
<td>91.3275</td>
</tr>
<tr>
<td>System Programming</td>
<td>110.5956</td>
<td>112.282</td>
<td>59.0788</td>
<td>90.2</td>
<td>103.9968</td>
<td>94.86</td>
<td>92.1195</td>
<td>102.3282</td>
<td>98.1708</td>
</tr>
<tr>
<td>Control Structures</td>
<td>95.977</td>
<td>93.9246</td>
<td>49.0669</td>
<td>82.0875</td>
<td>95.6499</td>
<td>94.7992</td>
<td>86.9375</td>
<td>87.1196</td>
<td>86.3597</td>
</tr>
<tr>
<td>Program Support Environment</td>
<td>47.7435</td>
<td>50.1639</td>
<td>47.4848</td>
<td>75.6015</td>
<td>75.4867</td>
<td>75.09</td>
<td>78.6475</td>
<td>78.7075</td>
<td>49.533</td>
</tr>
<tr>
<td>Documentation</td>
<td>60.8038</td>
<td>55.0738</td>
<td>50.5898</td>
<td>50.5898</td>
<td>57.6184</td>
<td>56.0912</td>
<td>56.1142</td>
<td>51.0</td>
<td>56.3404</td>
</tr>
<tr>
<td>Readability</td>
<td>52.372</td>
<td>55.1719</td>
<td>58.3438</td>
<td>48.2869</td>
<td>59.5344</td>
<td>59.5008</td>
<td>53.92</td>
<td>55.4367</td>
<td>56.8114</td>
</tr>
<tr>
<td>Time Efficiency</td>
<td>97.2</td>
<td>95.07</td>
<td>13.768</td>
<td>18.0095</td>
<td>34.8577</td>
<td>24.2871</td>
<td>19.11</td>
<td>96.7134</td>
<td>19.917</td>
</tr>
<tr>
<td>Space Efficiency</td>
<td>95.4</td>
<td>90.1218</td>
<td>19.499</td>
<td>16.1814</td>
<td>21.6178</td>
<td>20.9745</td>
<td>15.8119</td>
<td>40.576</td>
<td>32.967</td>
</tr>
<tr>
<td>Rent of Use</td>
<td>28.3122</td>
<td>34.6038</td>
<td>39.2588</td>
<td>35.704</td>
<td>32.34</td>
<td>33.1975</td>
<td>29.1</td>
<td>26.1317</td>
<td>35.525</td>
</tr>
<tr>
<td>Assembly Language Linkage</td>
<td>30.0260</td>
<td>30.5060</td>
<td>39.3069</td>
<td>29.0952</td>
<td>30.6366</td>
<td>34.9354</td>
<td>35.3140</td>
<td>34.4872</td>
<td>36.16</td>
</tr>
<tr>
<td>Target CPU Transportability</td>
<td>0.0000</td>
<td>12.7935</td>
<td>22.09</td>
<td>19.7594</td>
<td>21.1491</td>
<td>17.596</td>
<td>18.6496</td>
<td>25.7229</td>
<td>6.615</td>
</tr>
<tr>
<td>Multithreading</td>
<td>1.375</td>
<td>1.375</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Memory &amp; Recursion</td>
<td>24.1675</td>
<td>24.0625</td>
<td>0</td>
<td>0</td>
<td>20.535</td>
<td>19.22</td>
<td>5.000</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>TOTAL*</td>
<td>669.0078</td>
<td>657.2052</td>
<td>452.5650</td>
<td>594.3706</td>
<td>694.453</td>
<td>670.3129</td>
<td>607.4003</td>
<td>619.1908</td>
<td>617.2983</td>
</tr>
<tr>
<td>FOR</td>
<td>797.0918</td>
<td>783.1599</td>
<td>522.6456</td>
<td>591.1688</td>
<td>763.4764</td>
<td>777.1724</td>
<td>647.0724</td>
<td>780.0569</td>
<td>672.093</td>
</tr>
</tbody>
</table>

*Total without space, time, compile-time efficiency.
Figure E-1. Figures of Merit without Benchmark Values

E-6
Figure E-2. Final Figures of Merit