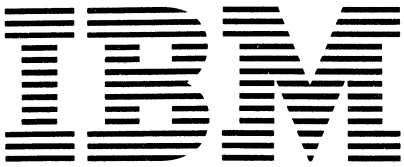

Application Program

GY20-0569-0

CALL/360-OS

PL/I System Manual – Volume III

Program Number 360A-CX-42X

The CALL/360-OS PL/I compiler (to be used with the CALL/360-OS System on an IBM System/360 Model 50 or higher) is described in the four volumes of this publication. The publication is addressed to system programmers and customer engineers who require a detailed knowledge of the compiler. It contains a general overview of the compiler and detailed information on the compiler and runtime routines and macros that perform required functions. Additional information required to understand CALL/360-OS PL/I compiler operations is provided in several appendices.

Volume III contains the remainder of the directory to runtime routines, consisting of the mathematical function package and aggregate manipulation package.

First Edition (August 1970)

This edition applies to Version 1, Modification Level 0, of CALL/360-OS (360A-CX-42X) and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters.

Changes are continually made to the information herein. Therefore, before using this publication, consult the latest System/360 SRL Newsletter (GN20-0360) for the editions that are applicable and current.

Copies of this and other IBM publications can be obtained through IBM branch offices.

A form has been provided at the back of this publication for readers' comments. If this form has been removed, address comments to: IBM Corporation, Technical Publications Department, 112 East Post Road, White Plains, New York 10601.

CONTENTS

MATHEMATICAL FUNCTION PACKAGE

The following routines constitute the Mathematical Function Package (MFUNC). They support CALL/360-OS built-in functions and can be grouped as shown below. Descriptions of these routines are given on succeeding pages of this manual, in the order indicated.

Note that, as far as possible, the general groupings are listed in alphabetic order according to the mnemonics of the routines. Within each grouping, discussion of short floating-point precedes discussion of long floating-point, and discussion of real precedes discussion of complex.

Arctangent and hyperbolic arctangent - ATAN, ATANH:

- Short Float Real Arctan (IHEATS)
- Long Float Real Arctan (IHEATL)
- Short Float Real Hyperbolic Arctan (IHEHTS)
- Long Float Real Hyperbolic Arctan (IHEHTL)
- Short Float Complex Arctan/Hyperbolic Arctan (IHEATW)
- Long Float Complex Arctan/Hyperbolic Arctan (IHEATZ)

Error function - ERF, ERFC:

- Short Float Real Error Function (IHEEFS)
- Long Float Real Error Function (IHEEFL)

Exponentiation - EXP:

- Short Float Real EXP (IHEEXS)
- Long Float Real EXP (IHEEXL)
- Short Float Complex EXP (IHEEXW)
- Long Float Complex EXP (IHEEXZ)

Logarithms - LOG, LOG2, LOG10:

- Short Float Real Log (IHELNS)
- Long Float Real Log (IHELNL)
- Short Float Complex Log (IHELNW)
- Long Float Complex Log (IHELNZ)

Sine and cosine, hyperbolic sine and hyperbolic cosine - SIN, COS, SINH, COSH:

- Short Float Real Sin/Cos (IHESNS)
- Long Float Real Sin/Cos (IHESNL)
- Short Float Real Hyperbolic Sin/Cos (IHESHS)
- Long Float Real Hyperbolic Sin/Cos (IHESHL)
- Short Float Complex Sin/Cos (IHESNW)
- Long Float Complex Sin/Cos (IHESNZ)

Square root - SQRT:

- Short Float Real SQRT (IHESQS)
- Long Float Real SQRT (IHESQL)
- Short Float Complex SQRT (IHESQW)
- Long Float Complex SQRT (IHESQZ)

Tangent and hyperbolic tangent - TAN, TANH:

- Short Float Real Tan (IHETNS)
- Long Float Real Tan (IHETNL)
- Short Float Real Hyperbolic Tan (IHETHS)
- Long Float Real Hyperbolic Tan (IHETHL)
- Short Float Complex Tan/Hyperbolic Tan (IHETNW)
- Long Float Complex Tan/Hyperbolic Tan (IHETNZ)

SPEED

The average execution times given in this subsection are based on information in IBM System/360 Instruction Timing Information (A22-6825) and include times for the modules called.

ACCURACY

In order to appreciate properly the meaning of the statistics for accuracy given with each module, some consideration of the limits and implications of these statistics is required. Because the size of a machine word is limited, small errors may be generated by mathematical routines. In an elaborate computation, slight inaccuracies can accumulate and become large errors. Thus, in interpreting final results, errors introduced during the various intermediate stages must be taken into account.

The accuracy of an answer produced by a routine is influenced by two factors:

1. The accuracy of the argument
2. The performance of the routine

Most arguments contain errors. An error in a given argument may have accumulated over several steps prior to the use of the routine. Even newly converted input data may contain slight errors. The effect of an argument error on the accuracy of an answer depends solely on the nature of the mathematical function involved and not on the particular coding by which that function is computed within a routine. In order to assist users in assessing the accumulation of errors, a guide on the propagational effect of argument errors is provided for each function described in this subsection. Wherever possible, this guide is expressed as a simple formula.

The performance statistics supplied in this document are based upon the assumption that the arguments are perfect (that is, without errors and, therefore, having no argument error propagation effect upon answers). Thus, the only errors in answers are those introduced by the routines themselves. For each routine, accuracy figures are given for the valid argument range of respective representative segments. In each case, the particular statistics are those most meaningful to the function and range under consideration.

For example, the maximum relative error and the root-mean-square of the relative error of a set of answers are generally useful and revealing statistics. However, they are useless for the range of a function where its value becomes 0, since the slightest error of the argument value can cause an unbounded fluctuation in the relative magnitude of the answer. Such is the case with sin(x) for values of x close to pi; in this range it is more appropriate to discuss absolute errors.

The statistics tabulated herein were derived from random distributions of 5000 arguments per segment, generated to be either uniform or exponential, as appropriate. It must be emphasized that each value quoted for the maximum error refers to a particular test using the method described above, and should be treated only as a guide to the true maximum error.

This explains, for example, why it is possible that the maximum error quoted for a segment may be greater than that found from a distribution of different arguments over a larger range which includes the test range.

HEXADECIMAL TRUNCATION ERRORS

While the use of hexadecimal numbers in System/360 has led to increased efficiency and flexibility, the effect of the variable number of significant digits carried by the floating-point registers must be noted in making allowance for truncation errors. In the production of the CALL/360-OS PL/I library, special care has been taken to minimize such errors, whenever this could be accomplished at minor cost. As a result, the relative errors produced by some of the library routines may be considerably smaller than the relative error produced in some instances by a single operation such as multiplication.

Representations of finite length entail truncation errors in any number system. With binary normalization, the effect of truncation is roughly uniform. With hexadecimal normalization, however, the effect varies by a factor of 16 depending on the size of the mantissa; in a chain of computations, the worst error committed in the chain usually prevails at the end.

In short-precision representation, a number has between 21 and 24 significant binary digits. Therefore, the truncation errors range from 2**-24 to 2**-20 (5.96*10**-8 to 9.5*10**-7). Assuming exact operands, a product or quotient is correct to the 24th binary digit of the mantissa. Hence truncation errors contributed by multiplication or division are no more than 2**-20. The same is true for the sum of two operands of the same sign. Subtraction, on the other hand, is the most common cause of loss of significant digits in any number system. For short-precision operations, therefore, a guard digit is provided which helps to reduce such loss.

In long-precision representation, a number has between 53 and 56 significant binary digits. Therefore, truncation errors range from 2**-56 to 2**-52 (1.39*10**-17 to 2.22*10**-16). Assuming exact operands, a quotient is correct to the 56th binary digit of the mantissa. Therefore, truncation errors resulting from division are no more than 2**-52. The accuracy of a product, on the other hand, depends on the necessity for post-normalization. If the mantissas of both operands are close to 1, the truncation error of a product is about 2**-56. If the product of the mantissas is about 1/16, the truncation error is about 2**-52. On the other hand, if the mantissas of both operands are close to 1/16, the intermediate product has seven leading zeros, and post-normalization introduces four trailing zeros. In this case, the truncation error can be close to 2**-48(3.5*10**-15). In particular, multiplication by 1 in the long-precision form has the effect of erasing the last hexadecimal digit of the multiplicand.

Normal care in numerical analysis should be exercised for addition and subtraction. In particular, when two algorithms are theoretically equivalent, it usually pays to choose the one which avoids subtraction between operands of similar size. There is no guard digit for long-precision additions and subtractions.

HEXADECIMAL CONSTANTS

Many of the modules described below discriminate between algorithms or test for errors by comparisons involving hexadecimal constants; where decimal fractions are used in the descriptions, the fractions are only quoted as convenient approximations to the hexadecimal values actually employed.

TERMINOLOGY

Maximum and root-mean-square values for the relative and (where necessary) the absolute errors are given for each module. These are:

Let f(x) = the correct value for a function
g(x) = the result obtained from the module in question

Then the absolute error of the result is

ABS(f(x) - g(x))

and the relative error of the result is

ABS((f(x) - g(x))/f(x))

Let the number of sample results obtained be N; then the root-mean-square of the absolute error is

$$\mathrm{SQRT}\left(\sum_{1}^{N} (\mathrm{ABS}(f(xi) - g(xi))**2)/N\right)$$

and the root-mean-square of the relative error is

$$\mathrm{SQRT}\left(\sum_{1}^{N} (\mathrm{ABS}((f(xi) - g(xi))/f(xi))**2)/N\right)$$

ARGUMENTS

Any restrictions on arguments are noted under two headings:

Range: This states any range of arguments for which a module is valid. Arguments outside the given ranges are assumed to have been excluded before the module is called.

Error and Exceptional Conditions: These cover any conditions which may result from the use of a routine:

P - Programmed conditions in the module concerned. Programmed tests are made where not too costly and, if an invalid argument is found, a branch is taken to the entry point IHEERRB of the Error Routine. (See "Handling of Interrupts Package" in Volume II. Error Routine is part of EXEP, which is a subpackage of HIP.) An appropriate message is printed and the ERROR condition is raised.

I - Interrupt conditions in the module concerned. For those routines where SIZE and FIXEDOVERFLOW are detected by programmed tests or where hardware interruptions may occur, the OVERFLOW and UNDERFLOW conditions pass to IHEERR and are treated in the normal way. The machine is assumed to be enabled for all interruptions except significance, which is masked.

O - Programmed conditions in modules called by the module concerned. These occur when invalid arguments are detected in module called.

H - As I, but the interrupt conditions occur in the modules called by the module concerned.

TITLE: SHORT FLOAT REAL ARCTAN (IHEATS)

Program Definition

Purpose and Usage

Short Float Real Arctan is used to calculate arctan(x) or arctan(y/x) where x is a short floating-point real expression in radians. The result range is:

arctan(x) (radians)	±pi/2
arctan(y/x) (radians)	±pi

Description

Method:

1. ATAN(y,x)

   If

   x = 0 or ABS(y/x) ≥ 2**24

   the answer, SIGN(y)**pi/2, is returned except for the error case, x = y = 0. Otherwise

   ATAN(y,x) = ATAN(y/x)if x > 0 or

   ATAN(y,x) = ATAN(y/x) + SIGN(y)*pi if x < 0

   Hence the computation is now reduced to the single argument case.

2. ATAN(x)

   The general case may be reduced to the range 0 ≤ x ≤ 1 since

   ATAN(-x) = - ATAN(x) and

   ATAN(1/ABS(x)) = pi/2 - ATAN(ABS(x))

   A further reduction to the range ABS(x) ≤ TAN(pi/12) is made by using

   ATAN(x) = pi/6 + ATAN((SQRT(3)*x-1)/(x + SQRT(3)))

   Care is taken to avoid the loss of significant digits in computing SQRT(3)*x - 1.

   For the basic range ABS(x) ≤ TAN(pi/12) use an approximation formula of the form

   ATAN(x)/x = a + b*x**2 + c/(d + x**2)

   with relative error less than 2**-27.1.

Effect of an Argument Error:

Let

t = x or y/x

Then the absolute error of the answer approaches the absolute error in t divided by (1 + t**2). Hence, for small values of t, the two

errors are approximately the same; however, as t becomes larger, the effect of the argument error on the answer error diminishes.

Accuracy:

Arguments		Relative Error *10⁶	
Range	Distribution	RMS	Maximum
IHEATS1			
Full Range	Tangents of numbers uniformly distributed between -pi/2 and pi/2	0.443	0.958
IHEATS2			
Full Range	y = sin and x = cos of numbers uniformly distributed between -pi/2 and pi/2	0.449	1.42

Implementation:

- Module size: 408 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the appropriate entry point in the tables:

  k = TAND(15)

	IBM System/360 Model Number				
ABS(x)	30	40	50	65	75

IHEATS1

≤k	3162	858	279	78.4	48.4
k < ABS(x) <1	4345	1136	347	97.8	58.0
1≤ ABS(x) <1/k	5051	1301	381	108	64.1
≥1/k	3868	1023	313	88.7	54.5

IHEATS2

≤k	4193	1138	363	106	67.3
k< ABS(x) <1	5376	1416	431	126	77.0
1≤ ABS(x) <1/k	6082	1581	465	136	83.1
≥1/k	4899	1303	496	117	73.4

Errors Detected

Error and Exceptional Conditions:

P: IHEATS2: x = y = 0 (206)

Local Variables

None

Program Interface

Entry Points

IHEATS1 - Entry for ATAN(x)

```
P7      = A(PLIST)
PLIST   = A(x)
          A(target)
```

IHEATS2 = Entry for ATAN(y,x)

```
P7      = A(PLIST)
PLIST   = A(y)
          A(x)
          A(target)
```

Exit Conditions

Normal. Returns to caller via the link register.
Abnormal. Branches to EXEP via the entry point IHEERRB.

Routines Called

EXEP Execution Error Package

Global Variables

None

Comments

IHEATS1 is called by compiled code. IHEATS2 is called by compiled code, IHELNW, and IHEATW.

TITLE: LONG FLOAT REAL ARCTAN (IHEATL)

Program Definition

Purpose and Usage

Long Float Real Arctan is used to calculate arctan(x) or arctan(y/x), where x is a long floating-point real expression in radians. The result range is:

arctan(x) (radians)	±pi/2
arctan(y/x) (radians)	±pi

Description

Method:

1. ATAN(y,x)

   If x = 0 or ABS(y/x) ≥ 2**56, the answer SIGN(y)*pi/2 is returned except for the error case x = y = 0. Otherwise

   ATAN(y,x) = ATAN(y/x) if x > 0 or

   ATAN(y,x) = ATAN(y/x) + SIGN(y)*pi if x < 0

   Hence the computation is now reduced to the single argument case.

2. ATAN(x)

   The general case may be reduced to the range 0 ≤ x ≤ 1 since

   ATAN(-x) = - ATAN(x) and

   (ATAN(1/ABS(x)) = pi/2 - ATAN(ABS(x))

   A further reduction to the range ABS(x) ≤ TAN(pi/12) is made by using

   ATAN(x) = pi/6 + ATAN((SQRT(3)*x-1)/(x + SQRT(3)))

   Care is taken to avoid the loss of significant digits in computing

   SQRT(3)*x - 1

   For the basic range ABS(x) ≤ TAN(pi/12), use a continued fraction of the form

   ATAN(x)/x = 1 + a1*x*x/(b1 + x*x + a2/(b2+x*x + a3/(b3 + x*x + a4/(b4 + x*x))))

   with relative error less than 2**(-57.9).

Effect of an Argument Error:

Let

t = x or y/x

Then the absolute error of the answer approaches the absolute error in t divided by (1 + t**2). Hence, for small values of t, the two errors are approximately the same; however, as t becomes larger, the effect of the argument error on the answer error diminishes.

Accuracy:

IHEATL1

Arguments		Relative Error $*10^{15}$	
Range	Distribution	RMS	Maximum
-1<x<1	Uniform	0.0438	0.207

Implementation:

- Module size: 544 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the appropriate entry point in the table:

	IBM System/360 Model Number				
Entry Point	30	40	50	65	75
IHEATL1	20472	4389	826	181	100
IHEATL2	23523	5042	967	217	123

Errors Detected

Error and Exceptional Conditions:

P: IHEATL2: x = y = 0 (206)

Local Variables

None

Program Interface

Entry Points

IHEATL1 - Entry for ATAN(x)

```
P7      = A(PLIST)
PLIST   = A(x)
          A(target)
```

IHEATL2 - Entry for ATAN(y,x)

```
P7      = A(PLIST)
PLIST   = A(y)
          A(x)
          A(target)
```

Exit Conditions

Normal. Returns to caller via the link register.
Abnormal. Branches to EXEP via the entry point IHEERRB.

Routines Called

EXEP Execution Error Package

Global Variables

None

Comments

IHEATL1 is called by compiled code. IHEATL2 is called by compiled code, IHELNZ, and IHEATZ.

TITLE: SHORT FLOAT REAL HYPERBOLIC ARCTAN (IHEHTS)

Program Definition

Purpose and Usage

Short Float Real Hyperbolic Arctan is used to calculate hyperbolic arctan(x), where x is a short floating-point real expression in radians.

Description

Method:

1. ABS(x) ≤ 0.2

   Use a rational approximation of the form:

   ATANH(x) = x + x**3/(a + b*x**2)

2. 0.2 < ABS(x) < 1

   ATANH(x) = -SIGN(x)*0.5*LOG((0.5 - ABS(x/2))/(0.5 + ABS(x/2)))

Effect of an Argument Error:

The absolute error caused in the result is approximately equal to the absolute error in the argument divided by (1 - x**2). Thus as x approaches +1 or -1, relative error increases rapidly. Near x = 0, the relative error in the result is of the order of that in the argument.

Accuracy:

Arguments		Relative Error $*10^6$	
Range	Distribution	RMS	Maximum
-0.8≤x ≤0.8	Uniform	0.440	1.32
-0.9<x <0.9	Uniform	0.389	1.14

Implementation:

- Module size: 192 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the table:

	IBM System/360 Model Number				
ABS(x)	30	40	50	65	75
≤0.2	2520	667	208	52.0	31.3
0.2< ABS(x) <1	7091	1829	606	163	94.8

Errors Detected

Error and Exceptional Conditions:

P: ABS(x) > 1 (208)

Local Variables

None

Program Interface

Entry Points

IHEAHS0

```
P7    = A(PLIST)
PLIST = A(x)
        A(target)
```

Exit Conditions

Normal. Returns to caller via the link register.
Abnormal. Branches to EXEP via the entry point IHEERRB.

Routines Called

IHELNS	Short Float Real Log
EXEP	Execution Error Package

Global Variables

None

Comments

Called by compiled code and IHEATW.

TITLE: LONG FLOAT REAL HYPERBOLIC ARCTAN (IHEHTL)

Program Definition

Purpose and Usage

Long Float Real Hyperbolic Arctan is used to calculate hyperbolic arctan(x), where x is a long floating-point real expression in radians.

Description

Method:

1. ABS(x) ≤ 0.25

   Use a Chebyshev polynomial of degree 8 in x**2 to compute ATANH(x)/x.

2. 0.25 < ABS(x) < 1

   ATANH(x) = -SIGN(x)*0.5*LOG((0.5 - ABS(x/2))/(0.5 + ABS(x/2)))

Effect of an Argument Error:

The absolute error caused in the result is approximately equal to the absolute error in the argument divided by (1 - x**2). Thus as x approaches +1 or -1, relative error increases rapidly. Near x = 0, the relative error in the result is of the order of that in the argument.

Accuracy:

Arguments		Relative Error $*10^{15}$	
Range	Distribution	RMS	Maximum
ABS(x)≤ 0.25	Uniform	0.0650	0.223
ABS(x)≤ 0.95	Uniform	0.133	0.397

Implementation:

- Module size: 262 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the table:

	IBM System/360 Model Number				
ABS(x)	30	40	50	65	75
≤0.25	12252	3037	562	121	68
0.25< ABS(x) <1	20448	4900	1040	242	137

Errors Detected

Error and Exceptional Conditions:

P: ABS(x) > 1 (208)

Local Variables

None

Program Interface

Entry Points

IHEAHL0

```
P7    = A(PLIST)
PLIST = A(x)
        A(target)
```

Exit Conditions

Normal. Returns to caller via the link register.
Abnormal. Branches to EXEP via the entry point IHEERRB.

Routines Called

IHELNL	Long Float Real Log
EXEP	Execution Error Package

Global Variables

None

Comments

Called by compiled code and IHEATZ.

TITLE: SHORT FLOAT COMPLEX ARCTAN/HYPERBOLIC ARCTAN (IHEATW)

Program Definition

Purpose and Usage

Short Float Complex Arctan/Hyperbolic Arctan is used to calculate arctan(z) or hyperbolic arctan(z), where z is a short floating-point complex expression in radians.

Description

Method:

Let

```
z = x + yI
```

Then

```
REAL(ATANH(z)) = (ATANH(2*x/(1 + x*x + y*y)))/2

IMAG(ATANH(z)) = (ATAN(2*y, (1 - x*x - y*y)))/2 and

ATAN(z) = -(ATANH(zI))I
```

Effect of an Argument Error:

The absolute error in the result is approximately equal to the absolute error in the argument divided by (1 + z**2) in the case of IHEATT0, or by (1 - z**2) in the case of IHEAHT0. Thus the effect may be considerable in the vicinity of z = ±1I(IHEATT0) or z = ±1(IHEAHT0).

Accuracy:

Arguments		Relative Error $*10^6$	
Range	Distribution	RMS	Maximum
IHEATT0			
Full Range	Exponential	0.216	2.88
IHEAHT0			
Full Range	Exponential	0.208	1.18

Implementation:

- Module size: 304 bytes
- Execution times:

Approximate execution times in microseconds for the System/360 models given below are obtained from the appropriate entry point in the table, where

a = ABS(2*u/(1+x*x+y*y))
u = y for IHEATT0
  = x for IHEAHT0

	IBM System/360 Model Number				
	30	40	50	65	75

IHEATT0

a≤0.2	12235	3306	1033	279	173
0.2<a<1	16056	4454	1408	462	276

IHEAHT0

a≤0.2	12100	3267	1017	275	171
0.2<a<1	15921	4415	1392	458	274

Errors Detected

Error and Exceptional Conditions:

P: IHEATT0 z = ±1I (211)
   IHEAHT0 z = ±1 (211)

Local Variables

None

Program Interface

Entry Points

IHEATT0 - Entry for ATAN(z)

P7 = A(PLIST)
PLIST = A(z)
        A(target)

IHEAHT0 - Entry for ATANH(z)

Linkage as for IHEATT0

Exit Conditions

Normal. Returns to caller via link register.
Abnormal. Branches to EXEP via the entry point IHEERRB.

Routines Called

IHEATS	Short Float Real Arctan
IHEHTS	Short Float Real Hyperbolic Arctan
EXEP	Execution Error Package

Global Variables

None

Comments

Called by compiled code.

TITLE: LONG FLOAT COMPLEX ARCTAN/HYPERBOLIC ARCTAN (IHEATZ)

Program Definition

Purpose and Usage

Long Float Complex Arctan/Hyperbolic Arctan is used to calculate arctan(z) or hyperbolic arctan(z), where z is a long floating-point complex expression in radians.

Description

Method:

Let

z = x + yI

Then

REAL(ATANH(z)) = (ATANH(2*x/(1 + x*x + y*y)))/2

IMAG(ATANH(z)) = (ATAN(2*y, (1 - x*x - y*y)))/2 and

ATAN(z) = -(ATANH(zI))I

Effect of an Argument Error:

The absolute error in the result is approximately equal to the absolute error in the argument divided by (1 + z**2) in the case of IHEATM0, or by (1 - z**2) in the case of IHEAHM0. Thus the effect may be considerable in the vicinity of z = ±1I(IHEATM0) or z = ±1(IHEAHM0).

Accuracy:

Arguments		Relative Error $*10^{15}$	
Range	Distribution	RMS	Maximum

IHEATM0

Full Range	Exponential	0.141	7.93

IHEAHM0

Full Range	Exponential	0.0826	1.20

Implementation:

- Module size: 296 bytes
- Execution times:

Approximate execution times in microseconds for the System/360 models given below are obtained from the appropriate entry point in the table, where

a = ABS(2*u/(1 + x*x + y*y))

u = y for IHEATM0
  = x for IHEAHM0

	IBM System/360 Model Number				
	30	40	50	65	75

IHEATM0

a≤0.25	43477	9977	2006	455	260
0.25<a<1	51673	11840	2406	576	329

IHEAHM0

a≤0.25	43293	9923	1987	450	258
0.25<a<1	51489	11786	2466	571	327

Errors Detected

Error and Exceptional Conditions:

P: IHEATM0 z = ±1I (211)
   IHEAHM0 z = ±1 (211)

Local Variables

None

Program Interface

Entry Points

IHEATM0 - Entry for ATAN(z)

P7 = A(PLIST)
PLIST = A(z)
        A(target)

IHEAHM0 - Entry for ATANH(z)

Linkage as shown for IHEATM0

Exit Conditions

Normal. Returns to caller via the link register.
Abnormal. Branches to EXEP via the entry point IHEERRB.

Routines Called

IHEATL	Long Float Real Arctan
IHEHTL	Long Float Real Hyperbolic Arctan
EXEP	Execution Error Package

Global Variables

None

Comments

Called by compiled code.

TITLE: SHORT FLOAT REAL ERROR FUNCTION (IHEEFS)

Program Definition

Purpose and Usage

Short Float Real Error Function is used to calculate the error function of x or the complement of this function, where x is a short floating-point real expression.

Description

Method:

1. $0 \leq x \leq 1.317$

   Compute ERF(x)/x by using a Chebyshev interpolation polynomial of degree 6 in x**2, with relative error less than 2**-24.

   ERFC(x) = 1 - ERF(x)
   (ERFC(x) > 1/16 in this range.)

2. $1.317 < x \leq k$, where k = 2.04000092

   Compute ERFC(x) by using a Chebyshev interpolation polynomial of degree 7 in (x-k), with absolute error less than 1.3*2**-30.

   ERF(x) = 1 - ERFC(x)
   (ERFC(x) > 1/256 in this range.)

3. $k < x < 13.306$

   ERFC(x)*xEXP(x**2) is computed by using a Chebyshev interpolation polynomial of degree 6 in x**-2, with relative error less than 1.2*2**-23:

   If x < 3.9192, ERF(x) = 1 - ERFC(x)

   If x ≥ 3.9192, ERF(x) = 1

4. $x \geq 13.306$

   Results 1 and 0 are returned for ERF(x) and ERFC(x) respectively.

5. $x < 0$

   ERF(x) = -ERF(-x)

Effect of an Argument Error:

The absolute error caused in the result is approximately equal to the absolute error in the argument multiplied by EXP(-x**2).

ERF(x): As the magnitude of the argument increases from 1, the effect of an argument error diminishes rapidly. For small x, the relative error of the result is of the order of the relative error of the argument.

Accuracy:

Arguments		Relative Error *10⁶	
Range	Distribution	RMS	Maximum
ABS(x) ≤1.3	Uniform	0.139	0.934
1.3< ABS(x) ≤2	Uniform	0.0372	0.263
2< ABS(x) ≤3.9	Uniform	0.0347	0.0605

Implementation:

- Module size: 376 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the appropriate entry point in the table:

	IBM System/360 Model Number				
ABS(x)	30	40	50	65	75

IHEEFS0

≤1.32	4354	1191	392	100	57.2
1.32< ABS(x) ≤2.04	4613	1266	418	110	62.0
2.04< ABS(x) <3.92	10013	2843	868	228	140
≥3.92	1530	473	183	50.7	32.0

Errors Detected

None

Local Variables

None

Program Interface

Entry Points

IHEEFS0 - Entry for ERF(x)

```
P7      = A(PLIST)
PLIST   = A(x)
          A(Target)
```

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEEXS	Short Float Real EXP

Global Variables

None

Comments

Called by compiled code.

TITLE: LONG FLOAT REAL ERROR FUNCTION (IHEEFL)

Program Definition

Purpose and Usage

Long Float Real Error Function is used to calculate the error function of x or the complement of this function, where x is a long floating-point real expression.

Description

Method:

1. $0 \leq x \leq 1$

   Compute ERF(x)/x by using a Chebyshev interpolation polynomial of degree 11 in x**2, with relative error less than 1.07*2**-57.

   ERFC(x) = 1 - ERF(x)
   (ERFC(x) > 1/16 in this range)

2. $1 < x \leq 2.04000092$

   Compute ERFC(x) by using a Chebyshev interpolation polynomial of degree 18 in (x - 1.999999), with absolute error less than 1.5*2**-61.

   ERF(x) = 1 - ERFC(x)
   (ERFC(x) > 1/256 in this range)

3. $2.04000092 < x < 13.306$

   ERFC(x) is computed by using a Chebyshev interpolation polynomial of degree 20 in x**2 for ERFC(x)*xEXP(x**2), with relative error ranging from 2**-53 at 2.04000092 to 2**-51 at 13.306:

   If $x < 6.092$, ERF(x) = 1 - ERFC(x)

   If $x \geq 6.092$, ERF(x) = 1

4. $x \geq 13.306$

   Results 1 and 0 are returned for ERF(x) and ERFC(x), respectively.

5. $x < 0$

   ERF(x) = -ERF(-x)

Effect of an Argument Error:

The absolute error caused in the result is approximately equal to the absolute error in the argument multiplied by EXP(-x**2).

ERF(x): As the magnitude of the argument increases from 1, the effect of an argument error diminishes rapidly. For small x, the relative error of the result is of the order of the relative error of the argument.

Accuracy:

Arguments		Relative Error *10^{15}	
Range	Distribution	RMS	Maximum
ABS(x)≤1.317	Uniform	0.0280	0.202
1.317<ABS(x)≤2.04	Uniform	0.0107	0.0291
2.04<ABS(x)<6.092	Uniform	0.00803	0.0170

Implementation:

- Module size: 744 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the appropriate entry point in the table:

	IBM System/360 Model Number				
ABS(x)	30	40	50	65	75
≤1.00	16567	4154	805	180	103
1.00<ABS(x)≤2.04	24413	6095	1175	263	147
2.04<ABS(x)<6.09	45574	1105	2142	477	269
≥6.09	2493	707	222	58.0	36.3

Errors Detected

None

Local Variables

None

Program Interface

Entry Points

IHEEFL0 - Entry for ERF(x)

```
P7      = A(PLIST)
PLIST   = A(x)
          A(target)
```

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEEXL	**Long Float Real EXP**

Global Variables

None

Comments

Called by compiled code.

TITLE: SHORT FLOAT REAL EXP (IHEEXS)

Program Definition

Purpose and Usage

Short Float Real EXP is used to compute e to the power x, where x is a short floating-point real expression.

Description

Method:

If x < -180.2, a zero result is returned immediately. Otherwise, EXP(x) is calculated as 2**(x*LOG2(e)).

The calculation is performed as follows:

$$x*LOG2(e) = f + N$$

where N = 4h + g, h is an integer such that g = 0, 1, 2, or 3, and $0 \leq f < 1.0$.

Then, by subtracting 0.5, this is reduced to the range $-0.5 < f < 0.5$. Next, 2**f is calculated as (a + b/(c + x*x) + x)/(a + b/(c + x*x) - x).

This is multiplied by 2**0.5 and then shifted in the appropriate direction to give the effect of multiplication by 2**g. Finally, the exponent of the result is obtained from h.

Effect of an Argument Error:

The relative error caused in the result is approximately equal to the absolute error in the argument, that is, to the argument relative error multiplied by x. Thus, for large values of x, even the roundoff error of the argument causes a substantial relative error in the answer.

Accuracy:

Arguments		Relative Error *10⁶	
Range	Distribution	RMS	Maximum
-1<x<1	Uniform	0.132	0.490
Full Range	Uniform	1.29	2.61

Implementation:

- Module size: 232 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the table:

	IBM System/360 Model Number				
	30	40	50	65	75
	3847	1172	356	90.0	58.0

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW if x > 174.673 (201)

Local Variables

None

Program Interface

Entry Points

IHEEXS0

```
P7    = A(PLIST)
PLIST = A(x)
        A(target)
```

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

None

Global Variables

None

Comments

Called by compiled code, IHEXXS, IHESHS, IHETHS, IHEEFS, IHEEXW, and IHESNW.

TITLE: LONG FLOAT REAL EXP (IHEEXL)

Program Definition

Purpose and Usage

Long Float Real EXP is used to compute e to the power x, where x is a long floating-point real expression.

Description

Method:

If x < -130.2183, return zero result. Otherwise, let

$$y = x/LOG(2) = 4*a - b - c/16 - d$$

where a, b, and c are integers, $0 \le b \le 3$, $0 \le c \le 15$, and $0 \le d \le 1/16$.

Then

$$EXP(x) = 2**y = 16**a*2**(-b)*2**(-c/16)*2**(-d) .$$

Compute 2**(-d) by using the Chebyshev interpolation polynomial of degree 6 over the range $0 \le d \le 1/16$, with maximum relative error 2**(-57). If c > 0, multiply 2**(-d) by 2**(-c/16). The constants 2**(-c/16), $1 \le c \le 15$, are included in the subroutine. If b > 0, halve the result b times. Finally, multiply by 16**a by adding a to the characteristic of the result.

Effect of an Argument Error:

The relative error caused in the result is approximately equal to the absolute error in the argument, that is, to the argument relative error multiplied by x. Thus, for large values of x, even the roundoff error of the argument causes a substantial relative error in the answer.

Accuracy:

Arguments		Relative Error $*10^{15}$	
Range	Distribution	RMS	Maximum
-1<x<1	Uniform	0.0674	0.216
Full Range	Uniform	0.867	2.30

Implementation:

- Module size: 448 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the table:

	IBM System/360 Model Number				
	30	40	50	65	75
	12131	2901	616	343	194

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW if x > 174.673 (201)

Local Variables

None

Program Interface

Entry Points

IHEEXL0

```
P7    = A(PLIST)
PLIST = A(x)
      = A(target)
```

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

None

Global Variables

None

Comments

Called by compiled code, IHEXXL, IHESHL, IHETHL, IHEEFL, IHEEXZ, and IHESNZ.

TITLE: SHORT FLOAT COMPLEX EXP (IHEEXW)

Program Definition

Purpose and Usage

Short Float Complex EXP is used to calculate e to the power z, where z is a short floating-point complex expression.

Description

Method:

Let

z = x + yI

Then

REAL(EXP(z)) = EXP(x)*COS(y) and

IMAG(EXP(z)) = EXP(x)*SIN(y)

Effect of an Argument Error:

Let

EXP(x + yI) = s*EXP(kI)

Then k = y, and the relative error in s is approximately equal to the absolute error in x.

Accuracy:

Arguments		Relative Error *10⁶	
Range	Distribution	RMS	Maximum
ABS(x) ≤170 ABS(y) ≤pi/2	Uniform	1.32	3.14
ABS(x) ≤170 pi/2< ABS(y) ≤20	Uniform	1.31	3.34

Implementation:

- Module size: 136 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the table:

	IBM System/360 Model Number				
	30	40	50	65	75
	14240	4058	1244	326	200

Errors Detected

Error and Exceptional Conditions:

O: ABS(y) ≥ 2**18*pi (212) in Short Float Real Sin/Cos (IHESNS)
H: OVERFLOW (300) in Short Float Real EXP (IHEEXS)

Local Variables

None

Program Interface

Entry Points

IHEEXT0

```
P7    = A(PLIST)
PLIST = A(z)
        A(target)
```

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHESNS	Short Float Real Sin/Cos
IHEEXS	Short Float Real EXP

Global Variables

None

Comments

Called by compiled code and IHEXXW.

TITLE: LONG FLOAT COMPLEX EXP (IHEEXZ)

Program Definition

Purpose and Usage

Long Float Complex EXP is used to calculate e to the power z, where z is a long floating-point complex expression.

Description

Method:

Let

z = x + yI

Then

REAL(EXP(z)) = EXP(x)*COS(y) and

IMAG(EXP(z)) = EXP(x)*SIN(y)

Effect of an Argument Error:

Let

EXP(x + yI) = s*EXP(kI)

Then k = y, and the relative error in s is approximately equal to the absolute error in x.

Accuracy:

Arguments		Relative Error *10^{15}	
Range	Distribution	RMS	Maximum
ABS(x) <1 ABS(y) <pi/2	Uniform	0.136	0.478
ABS(x) <20 ABS(y) <20	Uniform	1.28	2.29

Implementation:

- Module size: 136 bytes
- Execution times:

  Approximate execution times in microseconds for System/360 models given below are obtained from the table:

	IBM System/360 Model Number				
	30	40	50	65	75
	42838	10560	2174	505	287

Errors Detected

Error and Exceptional Conditions:

O: ABS(y) > 2**50*pi (203) in Long Float Real Sin/Cos (IHESNL)
H: OVERFLOW (300) in Long Float Real EXP (IHEEXL)

Local Variables

None

Program Interface

Entry Points

IHEEXM0

```
P7    = A(PLIST)
PLIST = A(z)
        A(target)
```

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHESNL	Long Float Real Sin/Cos
IHEEXL	Long Float Real EXP

Global Variables

None

Comments

Called by compiled code and IHEEXL.

TITLE: SHORT FLOAT REAL LOG (IHELNS)

Program Definition

Purpose and Usage

Short Float Real Log is used to compute:

Log x to the base e	LOG(x)
Log x to the base 2	LOG2(x)
Log x to the base 10	LOG10(x)

where x is a short floating-point real expression.

Description

Method:

Let

x = m*16**p

where 1/16 ≤ m < 1 and p is an integer.

Two constants, a (= base point) and b (= -LOG2(a)), are defined as follows:

1/16 ≤ m < 1/8:	a = 1/16	b = 4
1/8 ≤ m < 1/2:	a = 1/4	b = 2
1/2 ≤ m < 1:	a = 1	b = 0

Let

y = (m - a)/(m + a)

Then

m = a*(1+y)/(1-y) and ABS(y) ≤ 1/3

Now

x = 2**(4*p-b)*(1+y)/(1-y)

Therefore

LOG(x) = (4*p-b)*LOG(2) + LOG((1+y)/(1-y))

LOG((1+y)/(1-y)) is computed by using the Chebyshev interpolation polynominal of degree 4 in y**2 for the range 0 ≤ y**2 ≤ 1/9, with maximum relative error 2**-27.8. LOG2(x) or LOG10(x) is calculated by multiplying LOG(x) by LOG2(e) or LOG10(e) respectively.

Effect of an Argument Error:

The absolute error caused in the result is approximately equal to the relative error in the argument. Thus if the argument is close to 1, even the roundoff error of the argument causes a substantial relative error in the answer, since the function value there is very small.

Accuracy:

Arguments		Relative Error $*10^6$	
Range	Distribution	RMS	Maximum

IHELNS0

Excluding 0.5<x<2.0	Exponential	0.032	0.577

IHEL2S0

Excluding 0.5<x<2.0	Exponential	0.342	0.754

IHELGS0

Excluding 0.5<x<2.0	Exponential	0.170	0.963

Arguments		Absolute Error $*10^6$	
Range	Distribution	RMS	Maximum

IHELNS0

0.5<x<2.0	Uniform	0.0960	0.394

IHEL2S0

0.5<x<2.0	Uniform	0.177	0.842

IHELGS0

0.5<x<2.0	Uniform	0.0526	0.164

Implementation:

- Module size: 256 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the appropriate entry point in the table:

Entry Point	IBM System/360 Model Number				
	30	40	50	65	75
IHELNS0	4669	1238	385	173	95.7
IHEL2S0	5041	1342	417	180	101.3
IHELGS0	5054	1366	417	180	101.3

Errors Detected

Error and Exceptional Conditions:

P: $x \leq 0$ (202)

Local Variables

None

Program Interface

Entry Points

IHELNS0 - Entry for LOG(x)

```
P7      =   A(PLIST)
PLIST   =   A(x)
            A(Target)
```

IHELGS0 - Entry for LOG10(x)

Linkage as for IHELNS0

IHEL2S0 - Entry for LOG2(x)

Linkage as for IHELNS0

Exit Conditions

Normal. Returns to caller via the link register.
Abnormal. Branches to EXEP via the entry point IHEERRB.

Routines Called

EXEP Execution Error Package

Global Variables: None

Comments

Called by compiled code, IHEXXS, IHEXXW, IHEHTS, and IHELNW.

TITLE: LONG FLOAT REAL LOG (IHELNL)

Program Definition

Purpose and Usage

Long Float Real Log is used to compute:

Log x to the base e	LOG(x)
Log x to the base 2	LOG2(x)
Log x to the base 10	LOG10(x)

where x is a long floating-point real expression.

Description

Method:

Let

$$x = 16^{**}p*2^{**}(-q)*m$$

where p is the exponent, q is an integer such that $0 \le q \le 3$, and $1/2 \le m < 1$.

Two constants, a (= -base point) and b (= LOG2(a)), are defined as follows:

$1/2 \le m < 1/SQRT(2)$: a = 1/2, b = 1

$1/SQRT(2) \le m < 1$: a = 1, b = 0

Let

$$y = (m - a)/(m + a)$$

Then

$$m = a*(1 + y)/(1 - y)$$

and

$$ABS(y) < 0.1716$$

Now

$$x = 2^{**}(4*p - q - b)*(1 + y)/(1 - y)$$

Therefore

$$LOG(x) = (4*p - q - b)*LOG(2) + LOG((1 + y)/(1 - y))$$

LOG((1 + y)/(1 - y)) is computed by using the Chebyshev interpolation polynominal of degree 7 in y**2 for the range $0 \le y^{**}2 \le 0.02944$, with maximum relative error 2**(-59.6).

LOG2(x) or LOG10(x) is calculated by multiplying the result by LOG2(e) or LOG10(e) respectively.

Effect of an Argument Error:

The absolute error caused in the result is approximately equal to the relative error in the argument. Thus if the argument is close to 1, even the roundoff error of the argument causes a substantial relative error in the answer, since the function value there is very small.

Accuracy:

Arguments		Relative Error $*10^{15}$	
Range	Distribution	RMS	Maximum
IHELNL0			
Excluding 0.5<x<2.0	Exponential	0.0530	0.329
IHEL2L0			
Excluding 0.5<x<2.0	Exponential	0.443	2.60
IHELGL0			
Excluding 0.5<x<2.0	Exponential	0.155	0.402

Arguments		Absolute Error $*10^{15}$	
Range	Distribution	RMS	Maximum
IHELNL0			
0.5<x<2.0	Uniform	0.192	0.507
IHEL2L0			
0.5<x<2.0	Uniform	0.245	0.466
IHELGL0			
0.5<x<2.0	Uniform	0.0318	0.0625

Implementation:

- Module size: 360 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the appropriate entry point in the table:

	IBM System/360 Model Number				
Entry Point	30	40	50	65	75
IHELNL0	16216	3926	788	178	98.5
IHEL2L0	17315	4196	834	190	107
IHELGL0	17284	4192	828	188	105

Errors Detected

Error and Exceptional Conditions:

P: x ≤ 0 (202)

Local Variables

None

Program Interface

Entry Points

IHELNL0 - Entry for LOG(x)

```
P7      =   A(PLIST)
PLIST   =   A(x)
            A(Target)
```

IHELGL0 - Entry for LOG10(x)

Linkage as for IHELNL0

IHEL2L0 - Entry for LOG2(x)

Linkage as for IHELNL0

Exit Conditions

Normal. Returns to caller via the link register.
Abnormal. Branches to EXEP via the entry point IHEERRB.

Routines Called

EXEP Execution Error Package

Global Variables: None

Comments

Called by compiled code and IHEATL.

TITLE: SHORT FLOAT COMPLEX LOG (IHELNW)

Program Definition

Purpose and Usage

Short Float Complex Log is used to calculate the principal value of the natural log of z (that is, -pi < imaginary part of result ≤ pi), where z is a short floating-point complex expression.

Description

Method:

Let

```
LOG(x + yI) = u + vI
```

Then

```
u = LOG(ABS(x + yI))
  = LOG(SQRT(x**2 + y**2))
  = LOG(x**2 + y**2)/2

v = ATAN(y,x)
```

In computing u, the exponents of x and y are modified if necessary to avoid OVERFLOW or UNDERFLOW, with the appropriate correction being applied after the logarithm has been taken.

Effect of an Argument Error:

Let

```
z = r*EXP(hI)
```

and

```
LOG(z) = u + vI
```

Then the absolute error in u is approximately equal to the relative error in r. For the absolute error in v (h = ATAN(y,x)), see corresponding paragraph for Short Float Real Arctan (IHEATS).

Accuracy:

Arguments		Relative Error $*10^6$	
Range	Distribution	RMS	Maximum
Full Range except within 10^{-6} of 1+0I	Expanded radially, uniform round origin	0.150	2.27

Implementation:

- Module size: 272 bytes
- Execution times:

Approximate execution times in microseconds for the System/360 models given below are obtained from the table:

	IBM System/360 Model Number				
	30	40	50	65	75
(i)	11511	3414	1078	308	183
(ii)	11688	3489	1104	318	190
(iii)	11800	3520	1117	321	193

(i) ABS(x) and ABS(y) < SQRT(8)*16**31 and either ABS(x) or ABS(y) ≥ 16**-30.

(ii) Either ABS(x) or ABS(y) ≥ SQRT(8)*16**31.

(iii) ABS(x) and ABS(y) < 16**-30.

Errors Detected

Error and Exceptional Conditions:

0: x = y = 0 (202) in Short Float Real Log (IHELNS)

Local Variables

None

Program Interface

Entry Points

IHELNTO

```
P7    = A(PLIST)
PLIST = A(z)
        A(Target)
```

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHELNS	Short Float Real Log
IHEATS	Short Float Real Arctan

Global Variables

None

Comments

Called by compiled code and IHEXXW.

TITLE: LONG FLOAT COMPLEX LOG (IHELNZ)

Program Definition

Purpose and Usage

Long Float Complex Log is used to calculate the principal value of the natural log of z (that is, -pi < imaginary part of result ≤ pi), where z is a long floating-point complex expression.

Description

Method:

Let

```
LOG(x + yI) = u + vI
```

Then

```
u = LOG(ABS(x + yI))
  = LOG(SQRT(x**2 + y**2))
  = LOG(x**2 + y**2)/2

v = ATAN(y,x)
```

In computing u, the exponents of x and y are modified if necessary to avoid OVERFLOW or UNDERFLOW, with the appropriate correction being applied after the logarithm has been taken.

Effect of an Argument Error:

Let

```
z = r*EXP(hI)
```

and

```
LOG(z) = u + vI
```

Then the absolute error in u is approximately equal to the relative error in r. For the absolute error in v (h = ATAN(y,x)), see the corresponding paragraph for Long Float Real Arctan (IHEATL).

Accuracy:

Arguments		Relative Error $*10^{15}$	
Range	Distribution	RMS	Maximum
Full Range except within 10^{-6} of 1+0I	Exponential radially, uniform round origin	0.0558	1.46

Implementation:

- Module size: 288 bytes
- Execution times:

Approximate execution times in microseconds for System/360 models given below are obtained from the table:

	IBM System/360 Model Number				
	30	40	50	65	75
(i)	44101	10166	2086	480	274
(ii)	44398	10316	2155	507	290
(iii)	44438	10325	2156	507	290

(i) ABS(x) and ABS(y) < SQRT(8)*16**31 and either ABS(x) or ABS(y) ≥ 16**-26.

(ii) Either ABS(x) or ABS(y) ≥ SQRT(8)*16**31.

(iii) ABS(x) and ABS(y) < 16**-26.

Errors Detected

Error and Exceptional Conditions:

O: x = y = 0 (202) in Long Float Real Log (IHELNL)

Local Variables

None

Program Interface

Entry Points

IHELNM0

```
P7    = A(PLIST)
PLIST = A(z)
        A(Target)
```

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHELNL	Long Float Real Log
IHEATL	Long Float Real Arctan

Global Variables

None

Comments

Called by compiled code and IHEXXZ.

TITLE: SHORT FLOAT REAL SIN/COS (IHESNS)

Program Definition

Purpose and Usage

Short Float Real Sin/Cos is used to compute sin(x) and cos(x), where x is a short floating-point real expression in radians.

Description

Method:

Let

k = pi/4

Evaluate

p = ABS(x)*(1/k)

where x is in radians using long-precision multiplication to safeguard accuracy. Separate p into integer part q and fractional part r, that is, p = q + r where $0 \leq r < 1$.

Define

q1 = q (if SIN is required and $x \geq 0$)

q1 = q + 2 (if COS is required)

q1 = q + 4 (if SIN is required and $x < 0$).

Then for all values of x, each case has been reduced to the computation of SIN(k*(q1 + r)) = SIN(t), where $t \geq 0$.

Let

q2 = MOD(q1, 8)

If q2 = 0, then SIN(t) = SIN(k*r)

If q2 = 1, then SIN(t) = COS(k*(1-r))

If q2 = 2, then SIN(t) = COS(k*r)

If q2 = 3, then SIN(t) = SIN(k*(1-r))

If q2 = 4, then SIN(t) = -SIN(k*r)

If q2 = 5, then SIN(t) = -COS(k*(1-r))

If q2 = 6, then SIN(t) = -COS(k*r)

If q2 = 7, then SIN(t) = -SIN(k*(1-r))

Thus it is necessary to compute only SIN(k*r1) or COS(k*r1) where r1 = r or 1 - r and $0 \leq r1 \leq 1$.

This is performed by using the Chebyshev interpolation polynomials of degree 3 in r1**2, with maximum relative error of 2**-28.1 in the sine polynomial and 2**-24.6 in the cosine polynomial.

Effect of an Argument Error:

The absolute error of the answer is approximately equal to the absolute error in the argument. Hence, the larger the argument, the larger its absolute error and the larger the absolute error of the result. Since the function diminishes periodically for both sine and cosine, no consistent control of the relative error can be maintained outside the range -pi/2 to pi/2 radians.

Accuracy:

Arguments		Absolute Error *10⁶	
Range	Distribution	RMS	Maximum

IHESNS0

ABS(x)≤pi/2	Uniform	0.0557	0.126
pi/2<ABS(x)≤10	Uniform	0.0553	0.148
10<ABS(x)≤100	Uniform	0.0560	0.143

IHECSS0

0≤x≤pi	Uniform	0.0553	0.149
-10≤x<0, pi<x≤10	Uniform	0.0571	0.154
10<ABS(x)≤100	Uniform	0.0553	0.142

Arguments		Relative Error *10⁶	
Range	Distribution	RMS	Maximum

IHESNS0

ABS(x)≤pi/2	Uniform	0.198	1.40

Implementation:

- Module size: 320 bytes

- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the appropriate entry point in the table:

	IBM System/360 Model Number				
ABS(x)	30	40	50	65	75

IHESNS0

<pi/4	4091	1120	333	85.0	50.6
≥pi/4	4386	1190	362	92.5	53.8

IHECSS0

<pi/4	4078	1115	329	83.6	49.9
≥pi/4	4373	1184	357	91.0	53.1

Errors Detected

Error and Exceptional Conditions:

P: IHESNS0, IHECSS0: ABS(x) ≥ 2**18*pi (212)

Local Variables

None

Program Interface

Entry Points

IHESNS0 - Entry for SIN(x)

```
P7      =   A(PLIST)
PLIST   =   A(x)
            A(Target)
```

IHECSS0 - Entry for COS(x)

Linkage as for IHESNS0

Exit Conditions

Normal. Returns to caller via the link register.
Abnormal. Branches to the EXEP via the entry point IHEERRB.

Routines Called

EXEP Execution Error Package

Global Variables: None

Comments

Called by compiled code, IHESNW, and IHEEXW.

TITLE: LONG FLOAT REAL SIN/COS (IHESNL)

Program Definition

Purpose and Usage

Long Float Real Sin/Cos is used to compute sin(x) and cos(x), where x is a long floating-point real expression in radians.

Description

Method:

Let

y = ABS(x)/(pi/4) for x in radians and

y = q + r, q integral, $0 \leq r < 1$

Take

q1 = q for SIN with positive or zero argument

q1 = q + 2 for COS

q1 = q + 4 for SIN with negative argument and

q2 = MOD(q1, 8)

Since

COS(x) = SIN(ABS(x) + pi/2) and

SIN(-x) = SIN(ABS(x) + pi)

it is only necessary to find

SIN(pi/4*(q2 + r)), for $0 \leq q2 \leq 7$

Therefore compute:

SIN(pi/4*r)	if q2 = 0 or 4
COS(pi/4*(1-r))	if q2 = 1 or 5
COS(pi/4*r)	if q2 = 2 or 6
SIN(pi/4*(1-r))	if q2 = 3 or 7

SIN(pi/4*r1) where r1 is r or (1 - r) is computed by using the Chebyshev interpolation polynomial of degree 6 in r1**2, in the range $0 \leq$ r1**2 ≤ 1, with maximum relative error 2**(-58).

COS(pi/4*r1) is computed by using the Chebyshev interpolation polynomial of degree 7 in r1**2, in the range $0 \leq$ r1**2 ≤ 1, with maximum relative error 2**(-64.3).

Finally, if $q2 \geq 4$, a negative sign is given to the result.

Effect of an Argument Error:

The absolute error of the answer is approximately equal to the absolute error in the argument. Hence, the larger the argument, the larger its absolute error and the larger the absolute error of the result. Since the function diminishes periodically for both sine and cosine,

no consistent control of the relative error can be maintained outside the range -pi/2 to pi/2 radians.

Accuracy:

IHESNL0

Arguments		Relative Error *10^15	
Range	Distribution	RMS	Maximum
-pi/2<x<pi/2	Uniform	0.0542	0.381

IHECSL0

Arguments		Relative Error *10^15	
Range	Distribution	RMS	Maximum
-pi/2<x<pi/2	Uniform	0.0604	0.168

Implementation:

- Module size: 416 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the appropriate entry point in the table:

	IBM System/360 Model Number				
Entry Point	30	40	50	65	75
IHESNL0	13654	3290	661	155	85.3
IHECSL0	13641	3288	654	153	84.3

Errors Detected

Error and Exceptional Conditions:

P: IHESNL0, IHECSL0: ABS(x) ≥ 2**50*pi (203)

Local Variables

None

Program Interface

Entry Points

IHESNL0 - Entry for SIN(x)

```
P7      =   A(PLIST)
PLIST   =   A(x)
            A(Target)
```

IHECSL0 - Entry for COS(x)

Linkage as for IHESNL0

Exit Conditions

Normal. Returns to caller via the link register.
Abnormal. Branches to the EXEP via the entry point IHEERRC.

Routines Called

EXEP Execution Error Package

Global Variables

None

Comments

Called by compiled code, IHESNZ, and IHEEXZ.

TITLE: SHORT FLOAT REAL HYPERBOLIC SIN/COS (IHESHS)

Program Definition

Purpose and Usage

Short Float Real Hyperbolic Sin/Cos is used to calculate hyperbolic sin(x) or hyperbolic cos(x), where x is a short floating-point real expression in radians.

Description

Method:

For IHESHS, if ABS(x) ≤ 1, use a polynomial approximation of the seventh degree. Otherwise

SINH(x) = EXP(x)/2 - 0.5/EXP(x)

COSH(x) = EXP(x)/2 + 0.5/EXP(x)

These two versions of EXP(x)/2±0.5/EXP(x) are preferable to the equivalent versions of (EXP(x) - 1/EXP(x))/2 because, in floating-point, 0.5 has three more significant bits than 1.0.

Effect of an Argument Error:

The relative error caused in the result is approximately as follows:

SINH: The absolute error in the argument divided by TANH(x), that is, of the order of the absolute error in the argument for large x, or of the relative error in the argument for small x.

COSH: The absolute error in the argument multiplied by TANH(x), that is, of the order of the absolute error in the argument.

Thus, for large values of x, even the roundoff error of the argument causes a substantial relative error in the answer.

Accuracy:

Arguments		Relative Error *10^6	
Range	Distribution	RMS	Maximum
IHESHS0			
0< ABS(x)≤ 1	Uniform	0.200	0.932
1< ABS(x)< 2	Uniform	0.221	0.950

IHECHS0

ABS(x)≤1	Uniform	0.367	0.908
1<ABS(x)<2	Uniform	0.192	0.700

Implementation:

- Module size: 216 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the appropriate entry point in the table:

Entry Point	IBM System/360 Model Number				
	30	40	50	65	75
IHESHS0					
ABS(x)≤1	2544	692	228	56.3	33.0
ABS(x)>1	5647	1693	526	144	91.8
IHECHS0	5500	1648	509	139	88.4

Errors Detected

Error and Exceptional Conditions:

H: OVERFLOW (300) in Short Float Real EXP (IHEEXS)

Local Variables: None

Program Interface

Entry Points

IHESHS0 - Entry for SINH(x)

```
P7      =   A(PLIST)
PLIST   =   A(x)
            A(Target)
```

IHECHS0 - Entry for COSH(x)

Linkage as for IHESHS0

Exit Conditions: Normal. Returns to caller via link register.

Routines Called

IHEEXS Short Float Real EXP

Global Variables: None

Comments: Called by compiled code.

TITLE: LONG FLOAT REAL HYPERBOLIC SIN/COS (IHESHL)

Program Definition

Purpose and Usage

Long Float Real Hyperbolic Sin/Cos is used to calculate hyperbolic sin(x) or hyperbolic cos(x), where x is a long floating-point real expression in radians.

Description

Method:

For IHESHL, if ABS(x) < 0.3465736, compute SINH(x)/x using polynomial approximation of degree 5 in x**2, with relative error less than 2**-61.9. Otherwise, compute

```
s = EXP(ABS(x))
```

then

```
COSH(x) = (s + 1/s)/2

SINH(x) = SIGN(x)*(s - 1/s)/2
```

Effect of an Argument Error:

The relative error caused in the result is approximately as follows:

SINH: The absolute error in the argument divided by TANH(x), that is, of the order of the absolute error in the argument for large x, or of the relative error in the argument for small x.

COSH: The absolute error in the argument multiplied by TANH(x), that is, of the order of the absolute error in the argument.

Thus, for large values of x, even the roundoff error of the argument causes a substantial relative error in the answer.

Accuracy:

Arguments		Relative Error $*10^{15}$	
Range	Distribution	RMS	Maximum
IHESHL0			
ABS(x)< 0.34657	Uniform	0.0530	0.217
0.34657 < ABS(x)≤ 5	Uniform	0.0870	0.359

IHECHL0

ABS(x)≤5	Uniform	0.123	0.429

Implementation:

- Module size: 264 bytes
- Execution times:

  Approximate execution times in microseconds for System/360 models given below are obtained from the appropriate entry point in the table:

	IBM System/360 Model Number				
ABS(x)	30	40	50	65	75

IHESHL0

<0.347	9024	2279	450	101	59.0
0.347≤ ABS(x) ≤174.6	18634	4338	938	215	125

IHECHL0

≤174.6	18493	4300	924	211	123

Errors Detected

Error and Exceptional Conditions:

H: OVERFLOW (300) in Long Float Real EXP (IHEEXL)

Local Variables

None

Program Interface

Entry Points

IHESHL0 - Entry for SINH(x)

```
P7      =   A(PLIST)
PLIST   =   A(x)
            A(Target)
```

IHECHL0 - Entry for COSH(x)

Linkage as for IHESHL0

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEEXL Long Float Real EXP

Global Variables

None

Comments

Called by compiled code.

TITLE: SHORT FLOAT COMPLEX SIN/COS (IHESNW)

Program Definition

Purpose and Usage

Short Float Complex Sin/Cos is used to calculate sin(z), hyperbolic sin(z), cos(z), and hyperbolic cos(z), where z is a short floating-point complex expression in radians.

Description

Method:

Let

z = x + yI

Then

REAL(SIN(z)) = SIN(x)*COSH(y) and

IMAG(SIN(z)) = COS(x)*SINH(y)

REAL(COS(z)) = COS(x)*COSH(y) and

IMAG(COS(z)) = -SIN(x)*SINH(y)

REAL(SINH(z)) = COS(y)*SINH(x) and

IMAG(SINH(z)) = SIN(y)*COSH(x)

REAL(COSH(z)) = COS(y)*COSH(x) and

IMAG(COSH(z)) = SIN(y)*SINH(x)

To avoid making calls to evaluate SINH and COSH separately, and thus frequently having to evaluate EXP twice for the same argument, SINH(u) is computed as follows:

1. u > 0.3465736

   SINH(u) = (EXP(u) - 1/EXP(u))/2

2. 0 ≤ u ≤ 0.3465736

   SINH(u)/u is approximated by a polynomial of the form a + a1*u**2 + a2*u**4 (which has a relative error of less than 2**-26.4).

3. u ≤ 0

   SINH(u) = -SINH(-u)

Then

COSH(u) = SINH(ABS(u)) + 1/EXP(ABS(u))

Effect of an Argument Error:

Combine the effects on SIN, COS, SINH, and COSH according to the method of evaluation described in the above paragraph.

Accuracy:

Arguments		Relative Error *10⁶	
Range	Distribution	RMS	Maximum
IHESNT0			
ABS(x) ≤10, ABS(y)≤1	Uniform	0.721	2.07
IHESHT0			
ABS(x) ≤10, ABS(y)≤1	Uniform	0.561	1.86
IHECST0			
ABS(x) ≤10, ABS(y) ≤20	Uniform	0.546	2.00
IHECHT0			
ABS(x) ≤10, ABS(y) ≤20	Uniform	0.558	2.35

Implementation:

- Module size: 320 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the appropriate entry point in the tables:

SIN,COS : ABS(y)>0.3465736
SINH,COSH : ABS(x)>0.3465736

	IBM System/360 Model Number				
Entry Point	30	40	50	65	75
IHESNT0	15826	1508	648	363	75
IHECST0	15898	1518	653	366	225
IHESHT0	15930	1520	653	366	226
IHECHT0	15900	1519	655	367	227

SIN,COS : ABS(y)≤0.3465736
SINH,COSH : ABS(x)≤0.3465736

IHESNT0	16896	1585	674	381	232
IHECST0	16968	1595	679	384	234
IHESHT0	17000	1596	679	384	234
IHECHT0	16970	1595	681	384	235

Errors Detected

Error and Exceptional Conditions:

0: IHESNWS, IHECST0: ABS(x) ≥ 2**18*pi (212) in Short Float Real Sin/Cos (IHESNS)
H: OVERFLOW (300) in Short Float Real EXP (IHEEXS)

Local Variables: None

Program Interface

Entry Points

IHESNT0 - Entry for SIN(z)

```
   P7  =   A(PLIST)
PLIST  =   A(Target)
```

IHECSTO - Entry for COS(z)

Linkage as for IHESNTO

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHESNS	Short Float Real Sin/Cos
IHEEXS	Short Float Real EXP

Global Variables: None

Comments: Called by compiled code.

TITLE: LONG FLOAT COMPLEX SIN/COS (IHESNZ)

Program Definition

Purpose and Usage

Long Float Complex Sin/Cos is used to calculate sin(z), hyperbolic sin(z), cos(z), or hyperbolic cos(z), where z is a long floating-point complex expression in radians.

Description

Method:

Let

z = x + yI

Then

REAL(SIN(z)) = SIN(x)*COSH(y) and

IMAG(SIN(z)) = COS(x)*SINH(y)

REAL(COS(z)) = COS(x)*COSH(y) and

IMAG(COS(z)) = -SIN(x)*SINH(y)

REAL(SINH(z)) = COS(y)*SINH(x) and

IMAG(SINH(z)) = SIN(y)*COSH(x)

REAL(COSH(z)) = COS(y)*COSH(x) and

IMAG(COSH(z)) = SIN(y)*SINH(x)

To avoid making calls to evaluate SINH and COSH separately, and thus frequently having to evaluate EXP twice for the same argument, SINH(u) is computed as follows:

1. u > 0.3465736

   SINH(u) = (EXP(u) - 1/EXP(u))/2

2. 0 ≤ u ≤ 0.3465736

   SINH(u)/u is approximated by a polynomial of the fifth degree in u**2 which has a relative error of less than 2**-61.8.

3. u < 0

   SINH(u) = -SINH(-u)

Then

COSH(u) = SINH(ABS(u)) + 1/EXP(ABS(u))

Effect of an Argument Error:

Combine the effects on SIN, COS, SINH, and COSH according to the method of evaluation described in the above paragraph.

Accuracy:

Arguments		Relative Error $*10^{15}$	
Range	Distribution	RMS	Maximum
IHESNM0			
ABS(x) ≤10, ABS(y)≤1	Uniform	2.11	82.4
IHESHM0			
ABS(x) ≤10, ABS(y)≤1	Uniform	0.180	2.31
IHECSM0			
ABS(x) ≤10, ABS(y)≤1	Uniform	0.389	6.33
IHECHM0			
ABS(x) ≤10, ABS(y) ≤20	Uniform	1.11	19.4

Implementation:

- Module size: 368 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the appropriate entry point in the tables:

SIN,COS : ABS(y)>0.3465736
SINH,COSH : ABS(x)>0.3465736

	IBM System/360 Model Number				
Entry Point	30	40	50	65	75
IHESNM0	46584	11267	2363	552	313
IHECSM0	46656	11294	2373	555	315
IHESHM0	46726	11317	2378	557	316
IHECHM0	46640	11304	2374	556	316

SIN,COS : ABS(y)≤0.3465736
SINH,COSH : ABS(x)≤0.3465736

IHESNM0	54173	13141	2656	612	345
IHECSM0	54245	13168	2666	615	347
IHESHM0	54325	13191	2671	617	347
IHECHM0	54247	13177	2667	616	348

Errors Detected

Error and Exceptional Conditions:

0: IHESNZS, IHECSM0: ABS(x) ≥ 2**50*pi (203) in Long Float Real Sin/Cos (IHESNL)
H: OVERFLOW (300) in Long Float Real EXP (IHEEXL)

Local Variables: None

Program Interface

Entry Points

IHESNM0 - Entry for SIN(z)

```
P7      =   A(PLIST)
PLIST   =   A(z)
            A(Target)
```

IHECSM0 - Entry for COS(z)

Linkage as for IHESNM0

Exit Conditions: Normal. Returns to caller via the link register.

Routines Called

IHESNL Long Float Real Sin/Cos
IHEEXL Long Float Real EXP

Global Variables: None

Comments: Called by compiled code.

TITLE: SHORT FLOAT REAL SQRT (IHESQS)

Program Definition

Purpose and Usage

Short Float Real SQRT is used to compute the square root of x, where x is a short floating-point real expression.

Description

Method:

If

x = 0, SQRT(x) = 0

Otherwise, let

x = 16**(2*p + q)*f

where p is an integer, q = 0 or 1, and 1/16 ≤ f < 1. Then

SQRT(x) = 16**(p + q)*z

where

z = SQRT(f) if q = 0

z = SQRT(f)/4 if q = 1

An initial approximation, y0, is taken in the hyperbolic form a + b/(c + f) with different sets of constants for the two cases:

1. q = 0 a = 1.80713

   b = 1.57727

   c = 0.954182

   The maximum relative error in this range is less than 2**(-5.44), with an exact fit at f = 1 to guard as far as possible against loss of the last hexadecimal digit when f is nearly 1.

2. q = 1 a = 0.428795

   b = -0.3430368

   c = 0.877552

   The maximum relative error in this range is less than 2**(-6)*f**(-1/8).

Then

y1 = 16**(p + q)*y0

Two Newton-Raphson iterations then yield:

y2 = (y1 + x/y1)/2

SQRT(x) = y2 + (x/y2 - y2)/2

For case q = 0, the final relative error from this algorithm is less than 2**(-24.7), and, for case q = 1, less than 2**(-29).

Effect of an Argument Error:

The relative error caused in the result is approximately half the relative error in the argument.

Accuracy:

Arguments		Relative Error *10^6	
Range	Distribution	RMS	Maximum
Full Range	Exponential	0.230	0.924

Implementation:

- Module size: 168 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the table:

	IBM System/360 Model Number				
	30	40	50	65	75
	3140	793	227	68.4	40.7

Errors Detected

Error and Exceptional Conditions:

P: x < 0 (200)

Local Variables

None

Program Interface

Entry Points

IHESQS0

```
P7    = A(PLIST)
PLIST = A(x)
        A(Target)
```

Exit Conditions

Normal. Returns to caller via the link register.
Abnormal. Branches to the EXEP via IHEERRB.

Routines Called

EXEP Execution Error Package

Global Variables

None

Comments

Called by compiled code, IHEABW, and IHESQW.

TITLE: LONG FLOAT REAL SQRT (IHESQL)

Program Definition

Purpose and Usage

Long Float Real SQRT is used to compute the square root of x, where x is a long floating-point real expression.

Description

Method:

If

x = 0, SQRT(x) = 0

Otherwise, let

x = 16**(2*p - q)*f

where p is an integer,

q = 0 or 1, and 1/16 ≤ f < 1

Then

SQRT(x) = 16**p*2**(-2*q)*SQRT(f)

An initial approximation, y0, is taken by using (2/9 + 8/9*f) for SQRT(f). Multiplication by 2**(-2) when q = 1 is accomplished by using the HALVE instruction twice. The maximum relative error of this approximation is 1/9.

Four Newton-Raphson iterations of the form y(n+1) = (yn + x/yn)/2 are then applied, two in short precision and two in long precision, the last being computed as

SQRT(x) = y3 + (x/y3 - y3)/2

to minimize the truncation error.

The maximum relative error in the result from this algorithm is 2**(-65.7).

Effect of an Argument Error:

The relative error caused in the result is approximately half of the relative error in the argument.

Accuracy:

Arguments		Relative Error $*10^{15}$	
Range	Distribution	RMS	Maximum
$x>10^{-52}$	Exponential	0.0276	0.124

Implementation:

- Module size: 160 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the table:

	IBM System/360 Model Number				
	30	40	50	65	75
	8282	1733	376	97.7	57.2

Errors Detected

Error and Exceptional Conditions:

P: x < 0 (200)

Local Variables

None

Program Interface

Entry Points

IHESQL0

```
P7    = A(PLIST)
PLIST = A(x)
        A(Target)
```

Exit Conditions

Normal. Returns to caller via the link register.
Abnormal. Branches to EXEP via IHEERRB.

Routines Called

EXEP Execution Error Package

Global Variables

None

Comments

Called by compiled code, IHEABZ, and IHESQZ.

TITLE: SHORT FLOAT COMPLEX SQRT (IHESQW)

Program Definition

Purpose and Usage

Short Float Complex SQRT is used to calculate the principal value of the square root of z (that is, $-pi/2 < \text{argument of result} \leq pi/2$), where z is a short floating-point complex expression.

Description

Method:

Let

$z = x + yI$ and

$SQRT(z) = u + vI$

If

1. $x = y = 0$

   Then

   $u = v = 0$

2. $x \geq 0$

   Then

   $u = SQRT((ABS(x) + ABS(x + yI))/2)$ and

   $v = y/(2*u)$

3. $x < 0$

   Then

   $u = y/(2*v)$ and

   $v = S(y)*SQRT((ABS(x) + ABS(x + yI))/2)$

   where

   $S(y) = 1$ if $y \geq 0$
   $= -1$ if $y < 0$

Effect of an Argument Error:

Let

$z = r*EXP(hI)$ and

$SQRT(z) = s*EXP(kI)$

Then the relative error in s is approximately half the relative error in r, and the relative error in k is approximately equal to the relative error in h.

Accuracy:

Arguments		Relative Error *10⁶	
Range	Distribution	RMS	Maximum
Full Range	Exponential radially, uniform round origin	0.513	1.51

Implementation:

- Module size: 152 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the table:

	IBM System/360 Model Number				
	30	40	50	65	75
	11130	3023	1006	265	164

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
H: OVERFLOW (300) in Short Float Complex ABS (IHEABW)

Local Variables: None

Program Interface

Entry Points

IHESQT0

```
P7    = A(PLIST)
PLIST = A(z)
        A(Target)
```

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEABW	Short Float Complex ABS
IHESQS	Short Float Real SQRT

Global Variables: None

Comments

Called by compiled code.

TITLE: LONG FLOAT COMPLEX SQRT (IHESQZ)

Program Definition

Purpose and Usage

Long Float Complex SQRT is used to calculate the principal value of the square root of z (that is, $-pi/2 < \text{argument of result} \leq pi/2$), where z is a long floating-point complex expression.

Description

Method:

Let

$z = x + yI$ and

$SQRT(z) = u + vI$

If

1. $x = y = 0$

   Then

   $u = v = 0$

2. $x \geq 0$

   Then

   $u = SQRT((ABS(x) + ABS(x + yI))/2)$ and

   $v = y/(2*u)$

3. $x < 0$

   Then

   $u = y/(2*v)$ and

   $v = S(y)*SQRT((ABS(x) + ABS(x + yI))/2)$

   where

   $S(y) = 1$ if $y \geq 0$
   $= -1$ if $y < 0$

Effect of an Argument Error:

Let

$z = r*EXP(hI)$ and

$SQRT(z) = s*EXP(kI)$

Then the relative error in s is approximately half the relative error in r, and the relative error in k is approximately equal to the relative error in h.

Accuracy:

Arguments		Relative Error $*10^{15}$	
Range	Distribution	RMS	Maximum
Full Range	Exponential radially, uniform round origin	0.263	1.54

Implementation:

- Module size: 144 bytes
- Execution times:

  Approximate execution times in microseconds for System/360 models given below are obtained from the table:

	IBM System/360 Model Number				
	30	40	50	65	75
	26996	5957	1352	341	203

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
H: OVERFLOW (300) in Long Float Complex ABS (IHEABZ)

Local Variables: None

Program Interface

Entry Points

IHESQM0

```
P7    = A(PLIST)
PLIST = A(z)
        A(Target)
```

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEABZ	Long Float Complex ABS
IHESQL	Long Float Real SQRT

Global Variables: None

Comments

Called by compiled code.

TITLE: SHORT FLOAT REAL TAN (IHETNS)

Program Definition

Purpose and Usage

Short Float Real Tan is used to compute tan(x), where x is a short floating-point real expression in radians.

Description

Method:

Evaluate

p = (4/pi)*ABS(x)

Let q and r be respectively the integral and fractional parts of p.

If q is even, then s = r

If q is odd, then s = 1 - r

Let

q1 = MOD(q,4)

Then

If q1 = 0, then TAN(ABS(x)) = TAN(pi*s/4)

If q1 = 1, then TAN(ABS(x)) = COT(pi*s/4)

If q1 = 2, then TAN(ABS(x)) = -COT(pi*s/4)

If q1 = 3, then TAN(ABS(x)) = -TAN(pi*s/4)

Compute TAN(pi*s/4) and COT(pi*s/4) as the ratio of two polynomials:

TAN(pi*s/4) = s*p(s**2)/q(s**2)

COT(pi*s/4) = q(s**2)/(s*p(s**2))

where

p(s**2) = 212.58037 - 12.559912*s**2

q(s**2) = 270.665736 - 71.645273*s**2 + s**4

Finally, if x < 0,

TAN(x) = -TAN(ABS(x))

Effect of an Argument Error:

The absolute error of the answer is approximately equal to the absolute error of the argument multiplied by (1 + TAN(x)**2). Hence if x is near an odd multiple of pi/2, an argument error will produce a large absolute error in the answer.

The relative error in the result is approximately equal to twice the absolute error in the argument divided by SIN(2*x). Hence, if x is near a multiple of pi/2, an argument error will produce a large relative error in the result.

Accuracy:

Arguments		Relative Error *10⁶	
Range	Distribution	RMS	Maximum
ABS(x)≤ pi/4	Uniform	0.319	1.92
pi/4< ABS(x) <1.5	Uniform	0.465	1.24
pi/4< ABS(x) <pi/2	Uniform	3.14	170*
pi/2< ABS(x) ≤10	Uniform	1.25	70.6*
10< ABS(x) ≤100	Uniform	3.57	205*

*These maximum errors are those encountered in a sample of 5000 points; each figure depends very much on the particular points encountered near the singularities of the function.

Implementation:

- Module size: 280 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the table:

	IBM System/360 Model Number				
ABS(x)	30	40	50	65	75
<pi/4	4429	1172	336	85.8	51.0
≥pi/4	4788	1262	368	95.1	55.0

Errors Detected

Error and Exceptional Conditions:

P: ABS(x) ≥ 2**18*pi (213)
I: OVERFLOW (300)

Local Variables

None

Program Interface

Entry Points

IHETNS0

```
P7      =   A(PLIST)
PLIST   =   A(x)
            A(Target)
```

Exit Conditions

Normal. Returns to caller via the link register.
Abnormal. Branches to EXEP via the entry point IHEERRB.

Routines Called

EXEP Execution Error Package

Global Variables

None

Comments

Called by compiled code and IHETNW.

TITLE: LONG FLOAT REAL TAN (IHETNL)

Program Definition

Purpose and Usage

Long Float Real Tan is used to compute tan(x), where x is a long floating-point real expression in radians.

Description

Method:

Evaluate

p = (4/pi)*ABS(x)

Let q and r be respectively the integral and fractional parts of p.

If q is even, then s = r.

If q is odd, then s = 1 - r.

Let

q2 = MOD(q,4)

Then

If q2 = 0, then TAN(ABS(x)) = TAN(pi*s/4)

If q2 = 1, then TAN(ABS(x)) = COT(pi*s/4)

If q2 = 2, then TAN(ABS(x)) = -COT(pi*s/4)

If q2 = 3, then TAN(ABS(x)) = -TAN(pi*s/4)

Compute TAN(pi*s/4) and COT(pi*s/4) as the ratio of two polynomials:

TAN(pi*s/4) = s*p(s**2)/q(s**2)

COT(pi*s/4) = q(s**2)/(s*p(s**2))

where p(s**2) is of degree 3 and q(s**2) is of degree 4 in s**2, and maximum relative error is 3.4*10**-19.

Finally, if x < 0,

TAN(x) = -TAN(ABS(x))

Effect of an Argument Error:

The absolute error in the result is approximately equal to the absolute error in the argument multiplied by (1 + TAN(x)**2). Hence, if x is near an odd multiple of pi/2, an argument error will produce a large absolute error in the result.

The relative error in the result is approximately equal to twice the absolute error in the argument divided by SIN(2*x). Hence, if x is near a multiple of pi/2, an argument error will produce a large relative error in the result.

Accuracy:

Arguments		Relative Error $*10^{15}$	
Range	Distribution	RMS	Maximum
ABS(x)≤ pi/4	Uniform	0.091	0.530
pi/4< ABS(x) <1.5	Uniform	0.437	2.31
pi/4< ABS(x) <pi/2	Uniform	7.75	416*
pi/2< ABS(x) ≤10	Uniform	18.3	1140*
10< ABS(x) ≤100	Uniform	271	13400*

*These maximum errors are those encountered in a sample of 5000 points; each figure depends very much on the particular points encountered near the singularities of the function.

Implementation:

- Module size: 352 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the table:

	IBM System/360 Model Number				
ABS(x)	30	40	50	65	75
<pi/4	15440	3622	687	154	87.2
≥pi/4	16130	3817	747	169	93.9

Errors Detected

Error and Exceptional Conditions:

P: ABS(x) ≥ 2**50*pi (204)
I: OVERFLOW (300)

Local Variables

None

Program Interface

Entry Points

IHETNL0

```
P7      =   A(PLIST)
PLIST   =   A(x)
            A(Target)
```

Exit Conditions

Normal. Returns to caller via the link register.
Abnormal. Branches to EXEP via the entry point IHEERRB.

Routines Called

EXEP Execution Error Package

Global Variables

None

Comments

Called by compiled code and IHETNZ.

TITLE: SHORT FLOAT REAL HYPERBOLIC TAN (IHETHS)

Program Description

Purpose and Usage

Short Float Real Hyperbolic Tan is used to calculate hyperbolic tan(x), where x is a short floating-point real expression in radians.

Description

Method:

1. ABS(x) ≤ 2**-12

   Return x as result.

2. 2**-12 < ABS(x) < 0.54931

   Use a transformed continued fraction of the form:

   TANH(x)/x = 1 - ((x**2 + a)/(x**2 + b + c/x**2))

   with relative error less than 2**-27.

3. 0.54931 ≤ x < 9.011

   TANH(x) = 1 - 2/(EXP(2*x) + 1)

4. x ≥ 9.011

   Return result 1.

5. x ≤ -0.54931

   TANH(x) = -TANH(-x)

Effect of an Argument Error:

The relative error caused in the result is approximately twice the absolute error in the argument divided by SINH(2*x). Thus for small values of x, it is of the order of the relative error in the argument, and as x increases, the effect of the argument error is diminished.

Accuracy:

Arguments		Relative Error *10⁶	
Range	Distribution	RMS	Maximum
-0.5<x<0.5	Uniform	0.174	0.867
-9<x<9	Uniform	0.0720	0.782

Implementation:

- Module size: 200 bytes
- Execution times:

Approximate execution times in microseconds for the System/360 models given below are obtained from the table:

	IBM System/360 Model Number				
	30	40	50	65	75
ABS(x)≤ 2^{-12}	791	263	102	28.7	21.7
2^{-12}< ABS(x)<0.5	3033	785	231	64.1	43.9
0.5≤x <9	5934	1805	562	152	117
x≥9	1095	363	139	40.5	35.2

Errors Detected

None

Local Variables

None

Program Interface

Entry Points

IHETHS0

```
P7      =   A(PLIST)
PLIST   =   A(x)
            A(Target)
```

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEEXS Short Float Real EXP

Global Variables

None

Comments

Called by compiled code and IHETNW.

TITLE: LONG FLOAT REAL HYPERBOLIC TAN (IHETHL)

Program Definition

Purpose and Usage

Long Float Real Hyperbolic Tan is used to calculate hyperbolic tan(x), where x is a long floating-point real expression in radians.

Description

Method:

1. ABS(x) < 0.54931

   Compute TANH(x)/x using a rational approximation, with relative error less than 2**-64.5.

2. 0.54931 ≤ x < 20.101

   TANH(x) = 1 - 2/(EXP(2*x) + 1)

3. x ≥ 20.101

   Return result 1.

4. x ≤ -0.54931

   TANH(x) = -TANH(-x)

Effect of an Argument Error:

The relative error caused in the result is approximately twice the absolute error in the argument divided by SINH(2*x). Thus for small values of x, it is of the order of the relative error in the argument, and as x increases, the effect of the argument error is diminished.

Accuracy:

Arguments		Relative Error *10^{15}	
Range	Distribution	RMS	Maximum
ABS(x)≤ 0.54931	Uniform	0.0440	0.211
0.54931 <ABS(x) ≤5	Uniform	0.0250	0.199

Implementation:

- Module size: 280 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the table:

	IBM System/360 Model Number				
ABS(x)	30	40	50	65	75
<0.549	12745	3030	564	123	67.9
0.549≤ ABS(x) <20.1	16400	3918	878	205	119
≥20.1	1239	372	135	39.3	25.5

Errors Detected

None

Local Variables

None

Program Interface

Entry Points

IHETHL0

```
P7    = A(PLIST)
PLIST = A(x)
        A(Target)
```

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEEXL Long Float Real EXP

Global Variables

None

Comments

Called by compiled code and IHETNZ.

TITLE: SHORT FLOAT COMPLEX TAN/HYPERBOLIC TAN (IHETNW)

Program Definition

Purpose and Usage

Short Float Complex Tan/Hyperbolic Tan is used to calculate tan(z) or hyperbolic tan(z), where z is a short floating-point complex expression in radians.

Description

Method:

Let

z = x + yI

Then

REAL(TAN(z)) = TAN(x)*(1 - TANH(y)**2)/(1 + (TAN(x)*TANH(y))**2)

IMAG(TAN(z)) = TANH(y)*(1 + TAN(x)**2)/(1 + (TAN(x)*TANH(y))**2)

TANH(z) = -(TAN(zI))I

Effect of an Argument Error:

The absolute error caused in the result is approximately equal to the absolute error in the argument divided by ABS(COS(z)**2) for IHETNT0 or divided by ABS(COSH(z)**2) for IHETHT0. The relative error caused in the result is approximately twice the absolute error in the argument divided by ABS(SIN(2*z)) for IHETNT0, or divided by ABS(SINH(2*z)) for IHETHT0.

Accuracy:

Arguments		Relative Error *10⁶	
Range	Distribution	RMS	Maximum
IHETNT0			
ABS(x)<1 ABS(y)<9	Uniform	0.430	1.67
IHETHT0			
ABS(x)<9 ABS(y)<1	Uniform	0.430	1.45

Implementation:

- Module size: 184 bytes

- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the formulas:

  IHETNT0: a + time for execution of IHETHS with argument y
  IHETHT0: b + time for execution of IHETHS with argument x

	IBM System/360 Model Number				
	30	40	50	65	75
a	9094	2310	696	186	111
b	9197	2454	716	191	115

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
O: ABS(u) ≥ 2**18*pi (213)

where u = x for IHETNT0
u = y for IHETHT0

H: OVERFLOW (300) in Short Float Real Tan (IHETNS)

Local Variables

None

Program Interface

Entry Points

IHETNT0 - Entry for TAN(z)

```
P7      =   A(PLIST)
PLIST   =   A(z)
            A(Target)
```

IHETHT0 - Entry for TANH(z)

Linkage as for IHETNT0

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHETHS	Short Float Real Hyperbolic Tan
IHETNS	Short Float Real Tan

Global Variables

None

Comments

Called by compiled code.

TITLE: LONG FLOAT COMPLEX TAN/HYPERBOLIC TAN (IHETNZ)

Program Definition

Purpose and Usage

Long Float Complex Tan/Hyperbolic Tan is used to calculate tan(z) or hyperbolic tan(z), where z is a long floating-point complex expression in radians.

Description

Method:

Let

$$z = x + yI$$

Then

$$REAL(TAN(z)) = TAN(x)*(1 - TANH(y)**2)/(1 + (TAN(x)*TANH(y))**2)$$

$$IMAG(TAN(z)) = TANH(y)*(1 + TAN(x)**2)/(1 + (TAN(x)*TANH(y))**2)$$

$$TANH(z) = -(TAN(zI))I$$

Accuracy:

Arguments		Relative Error $*10^{15}$	
Range	Distribution	RMS	Maximum
IHETNM0			
ABS(x)<1 ABS(y)<9	Uniform	0.139	1.11
IHETHM0			
ABS(x)<9 ABS(y)<1	Uniform	0.137	0.980

Implementation:

- Module size: 184 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the appropriate entry point in the tables:

	IBM System/360 Model Number				
ABS(y)	30	40	50	65	75

IHETNM0

<0.549	40843	9625	1844	411	233
0.549≤ ABS(y) <20.1	44498	10513	10365	493	284
≥20.1	29337	6967	1415	327	190

	IBM System/360 Model Number				
ABS(x)	30	40	50	65	75

IHETHM0

<0.549	41122	9709	1871	419	236
0.549≤ ABS(x) <20.1	44777	10597	2185	501	287
≥20.1	29616	7051	1442	334	193

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
O: ABS(u) ≥ 2**50*pi (204)

where u = x for IHETNM0
u = y for IHETHM0

H: OVERFLOW (300) in Long Float Real Tan (IHETNL)

Local Variables

None

Program Interface

Entry Points

IHETNM0 - Entry for TAN(z)

P7 = A(PLIST)
PLIST = A(z)
A(Target)

IHETHM0 - Entry for TANH(z)

Linkage as for IHETNM0

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHETHL	Long Float Real Hyperbolic Tan
IHETNL	Long Float Real Tan

Global Variables

None

Comments

Called by compiled code.

AGGREGATE MANIPULATION PACKAGE

The following routines constitute the Aggregate Manipulation Package (AMP). They support the CALL/360-OS built-in functions PROD, SUM, and POLY. Descriptions of these routines are given on succeeding pages of this manual, in the order indicated.

Note that the four general groupings appear in alphabetic order according to the mnemonics of the routines. Within each grouping, discussion of short floating-point precedes discussion of long floating-point, and discussion of real precedes discussion of complex.

Indexing:

Interleaved Array Indexer (IHEJXI)

PROD:

PROD-Interleaved Real Fixed Array (IHEPDF)
PROD-Interleaved Real Short Float Array (IHEPDS)
PROD-Interleaved Real Long Float Array (IHEPDL)
PROD-Interleaved Complex Fixed Array (IHEPDX)
PROD-Interleaved Complex Short Float Array (IHEPDW)
PROD-Interleaved Complex Long Float Array (IHEPDZ)

SUM:

SUM-Interleaved Real Fixed Array (IHESMF)
SUM-Interleaved Real/Complex Short Float Array (IHESMG)
SUM-Interleaved Real/Complex Long Float Array (IHESMH)
SUM-Interleaved Complex Fixed Array (IHESMX)

POLY:

POLY (A,X) (A and X Real Fixed) (IHEYGF)
POLY (A,X) (A and X Real Short Float) (IHEYGS)
POLY (A,X) (A and X Real Long Float) (IHEYGL)
POLY (A,X) (A and X Complex Fixed) (IHEYGX)
POLY (A,X) (A and X Complex Short Float) (IHEYGW)
POLY (A,X) (A and X Complex Long Float) (IHEYGZ)

SPEED

The average execution times given in this subsection are based on information in IBM System/360 Instruction Timing Information (A22-6825).

EFFECT OF HEXADECIMAL TRUNCATION

Allowance must be made for a certain amount of truncation error in System/360 handling of hexadecimal numbers. The accuracy of values returned by the built-in functions SUM, PROD, and POLY is governed accordingly. (See "Hexadecimal Truncation Errors" under "Mathematical Function Package.")

ARGUMENTS

Any restrictions on arguments are noted under two headings:

Range: This states any range of arguments for which a module is valid. Arguments outside the given ranges are assumed to have been excluded before the module is called.

Error and Exceptional Conditions: These cover conditions which may result from the use of a routine; they are listed in four categories:

P - Programmed conditions in the module concerned. Programmed tests are made where not too costly and, if an invalid argument is found, a branch is taken to the entry point IHEERRB of the Error Routine. (See "Handling of Interrupts Package" in Volume II. Error Routine is part of EXEP, which is a subpackage of HIP.) An appropriate message is printed and the ERROR condition is raised.

I - Interrupt conditions in the module concerned. For those routines where FIXEDOVERFLOW is detected by programmed tests, or where hardware interruptions may occur, the OVERFLOW, UNDERFLOW, and (when the conversion package is called) ERROR conditions pass to IHEERR and are treated in the normal way. The machine is assumed to be enabled for all interruptions except significance, which is masked.

O - Programmed conditions in modules called by the module concerned. These occur when invalid arguments are detected in the module called.

H - As I, but the interrupt conditions occur in the modules called by the module concerned.

TITLE: INTERLEAVED ARRAY INDEXER (IHEJXI)

Program Definition

Purpose and Usage

Interleaved Array Indexer is used to find the next element of an array and to return its byte address in register P7.

Entry point IHEJXIY is used to initialize the routine for byte addresses and to provide the address of the first element in the array. Entry point IHEJXIA is used thereafter to obtain the addresses of subsequent elements of the array; one address is returned for each entry into the routine.

Description

Method:

Arrays are stored in row-major order. Let Li be the lower bound and Ui the upper bound of the ith dimension, and n the number of dimensions. Starting with the element A(L1,L2,...,Ln), the routine varies the subscripts through their ranges to A(U1,U2,...,Un), changing the nth subscript most rapidly; in this way the elements are referenced in the order in which they are stored.

In the initializer part of the routine, it is determined how many of the last dimensions in the array are stored contiguously. This is done by comparing Mj for equality with U(j+1) - L(j+1) + 1 for j = n-1, n-2,...,1. The contiguous storage of elements is ended at the first point where this equality does not hold. This dimension is saved in the storage location called S. The count of the number of contiguous elements is calculated during this process and saved. An array INDEX is set to the lower bound values for all dimensions.

Initially the base address is equal to the address of A(L1,L2,...,Ln). Each subsequent contiguous element address is generated from the previous one by adding the multiplier Mn from the array dope vector (ADV) and reducing the contiguous element count by 1.

When the count reaches 0, the next subscript to the left is increased by 1, the next element address is recalculated using the values in INDEX, and the contiguous element count is restored. If increasing the INDEX value for a dimension causes it to pass the upper bound value, then the next dimension to the left is increased by 1, and so forth. In any case, when increasing a dimension by 1, all dimensions to the right are reset to their lower bounds.

Range:

$0 < \text{number of dimensions} \leq 255$

Implementation:

- Module size: 1196 bytes

Errors Detected

None

Local Variables

NDIMS	Number of dimensions
ELE	Number of contiguous elements
COUNT	Number of contiguous elements left
Q	Current dimension processing
S	Dimension ruining continuity
MN	Last multiplier
BNDPTR	Pointer to bound slot for dimension Q
R	Leftmost dimension being worked on
UBNDPTR	Pointer to upper bound for dimension Q
INDEX	Current subscript value array

Note: Storage for these local variables is kept in the routine itself; thus this routine is not reentrant. However, it can easily be made reentrant by moving its storage to the LCA area, since the storage needs to be preserved between calls to this routine.

Program Interface

Entry Points

IHEJXIY Entry to initialize for byte addressing, and to locate the first element of the array.

P7 = address of parameter list
parameter list: A(Source ADV)

IHEJXIA Entry to locate the next element of the array

Linkage: No explicit arguments.
Implicit arguments: LCA

Exit Conditions

Normal. Returns to caller via the link register with:

1. P7 = the byte address of the first or current element of the array.
2. G0 = 0, if the last address of the last element of the array provided was not the last element.
3. G0 = 4, if the last address of the last element of the array provided was the last element.

Routines Called

None

Global Variables

WJXIDVA	Dope Vector Address
WJXILADD	Last Element Address

Comments

Called by:

IHEPDF	IHESMF
IHEPDX	IHESMX
IHEPDS	IHESMG
IHEPDW	IHESMH
IHEPDL	
IHEPDZ	

TITLE: PROD-INTERLEAVED REAL FIXED ARRAY (IHEPDF)

Program Definition

Purpose and Usage

PROD-Interleaved Real Fixed Array is used to equate a long or short floating-point real target to the product of all the elements of an interleaved array of fixed-point reals.

Description

Method:

The elements of the array are used in row-major order to multiply the current product. For fixed-point arguments, each element is converted to floating-point by using the Total Conversion Package. The precision specified in the source DED determines the precision of the target.

Implementation:

- Module size: 144 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the following formulas. 'Short' or 'long' refers to the floating-point result.

Target	
short	a + T1 + R*(e + T3)
long	c + T1 + R*(f + T3)

  Constants used in the formulas are:

  R = number of elements in the array

  T1 = sum of times required to execute IHEJXI using IHEJXIY and IHEJXIA

  T3 = time for the appropriate conversion using IHEDMA

	IBM System/360 Model Number				
	30	40	50	65	75
a	1075	365	141	37.8	25
c	1124	382	147	39	25.9
e	645	201	69	18.2	12.1
f	1363	371	81.5	20.2	13

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
UNDERFLOW (340)

Local Variables

None

Program Interface

Entry Points

IHEPDF0

P7 = A(Parameter List) where Parameter List:
A(ADV)
A(Array DED)
A(Target)

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEDMA	Arithmetic Conversion Director
IHEJXI	Interleaved Array Indexer

Global Variables

None

Comments

Called by compiled code.

TITLE: PROD-INTERLEAVED REAL SHORT FLOAT ARRAY (IHEPDS)

Program Definition

Purpose and Usage

PROD-Interleaved Real Short Float Array is used to equate a short floating-point real target to the product of all the elements of an interleaved array of short floating-point reals.

Description

Method:

The elements of the array are used in row-major order to multiply the current product.

Implementation:

- Module size: 88 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the following formula.

  a + T1 + R*b

  Constants used in the formula are:

  R = number of elements in the array

  T1 = sum of times required to execute IHEJXI using IHEJXIY and IHEJXIA

	IBM System/360 Model Number				
	30	40	50	65	75
a	418	178	80	23.7	17.5
b	492	143	43.3	11.4	7.2

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
UNDERFLOW (340)

Local Variables

None

Program Interface

Entry Points

IHEPDS0

P7 = A(Parameter List) where Parameter List:
A(ADV)
A(Target)

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEJXI Interleaved Array Indexer

Global Variables

None

Comments

Called by compiled code.

TITLE: PROD-INTERLEAVED REAL LONG FLOAT ARRAY (IHEPDL)

Program Definition

Purpose and Usage

PROD-Interleaved Real Long Float Array is used to equate a long floating-point real target to the product of all the elements of an interleaved array of long floating-point reals.

Description

Method:

The elements of the array are used in row-major order to multiply the current product.

Implementation:

- Module size: 88 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the following formula.

  a + T1 + R*b

  Constants used in the formula are:

  R = number of elements in the array

  T1 = sum of times required to execute IHEJXI using IHEJXIY and IHEJXIA

	IBM System/360 Model Number				
	30	40	50	65	75
a	-290	8.8	67.5	20.7	15.5
b	1264	331	63.8	14.8	9.2

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
UNDERFLOW (340)

Local Variables

None

Program Interface

Entry Points

IHEPDL0

P7 = A(Parameter List) where Parameter List:
A(ADV)
A(Target)

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEJXI Interleaved Array Indexer

Global Variables

None

Comments

Called by compiled code.

TITLE: PROD-INTERLEAVED COMPLEX FIXED ARRAY (IHEPDX)

Program Definition

Purpose and Usage

PROD-Interleaved Complex Fixed Array is used to equate a long or short floating-point complex target to the product of all the elements of an interleaved array of fixed-point complexes.

Description

Method:

The elements of the array are used in row-major order to multiply the current product. For fixed-point arguments, each element is converted to floating-point by using the Total Conversion Package. The precision specified in the source DED determines the precision of the target.

Implementation:

- Module size: 272 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the following formula.

Source	Target	
binary	short	a + T1 + R*(e + 2*T3)

  Constants used in the formula are:

  R = number of elements in the array

  T1 = sum of times required to execute IHEJXI using IHEJXIY and IHEJXIA

  T3 = time for the appropriate conversion using IHEDMA

	IBM System/360 Model Number				
	30	40	50	65	75
a	1382	497	208	56.8	41.2
c	1374	501	205	56.7	40.8
e	2047	574	182	44.4	26.3
f	5097	1306	250	56.9	33.2

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
UNDERFLOW (340)

Local Variables

None

Program Interface

Entry Points

IHEPDX0

P7 = A(Parameter List) where Parameter List:
A(ADV)
A(Array DED)
A(Target)

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEDMA	Arithmetic Conversion Director
IHEJXI	Interleaved Array Indexer

Global Variables

None

Comments

Called by compiled code.

TITLE: PROD-INTERLEAVED COMPLEX SHORT FLOAT ARRAY (IHEPDW)

Program Definition

Purpose and Usage

PROD-Interleaved Complex Short Float Array is used to equate a short floating-point complex target to the product of all the elements of an interleaved array of short floating-point complexes.

Description

Method:

The elements of the array are used in row-major order to multiply the current product.

Implementation:

- Module size: 120 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the following formula.

  a + T1 + R*b

  Constants used in the formula are:

  R = number of elements in the array

  T1 = sum of times required to execute IHEJXI using IHEJXIY and IHEJXIA

	IBM System/360 Model Number				
	30	40	50	65	75
a	-814	-96	12.1	8.9	12.5
b	1694	452	132	32	17.7

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
UNDERFLOW (340)

Local Variables

None

Program Interface

Entry Points

IHEPDW0

P7 = A(Parameter List) where Parameter List:
A(ADV)
A(Target)

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEJXI Interleaved Array Indexer

Global Variables

None

Comments

Called by compiled code.

TITLE: PROD-INTERLEAVED COMPLEX LONG FLOAT ARRAY (IHEPDZ)

Program Definition

Purpose and Usage

PROD-Interleaved Complex Long Float Array is used to equate a long floating-point complex target to the product of all the elements of an interleaved array of long floating-point complexes.

Description

Method:

The elements of the array are used in row-major order to multiply the current product.

Implementation:

- Module size: 120 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the following formula.

  a + T1 + R*b

  Constants used in the formula are:

  R = number of elements in the array

  T1 = sum of times required to execute IHEJXI using IHEJXIY and IHEJXIA

	IBM System/360 Model Number				
	30	40	50	65	75
a	-3841	-852	-61.5	-4.7	4.2
b	4830	1214	211	46.1	25.7

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
UNDERFLOW (340)

Local Variables

None

Program Interface

Entry Points

IHEPDZ0

P7 = A(Parameter List) where Parameter List:
A(ADV)
A(Target)

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEJXI Interleaved Array Indexer

Global Variables

None

Comments

Called by compiled code.

TITLE: SUM-INTERLEAVED REAL FIXED ARRAY (IHESMF)

Program Definition

Purpose and Usage

SUM-Interleaved Real Fixed Array is used to equate a long or short floating-point real target to the sum of all the elements of an interleaved array of fixed-point reals.

Description

Method:

The elements of the array are added to the current sum in row-major order. For fixed-point arguments, each element is converted to floating-point by using routines of the Total Conversion Package. The precision specified in the source DED determines the precision of the target.

Implementation:

- Module size: 136 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the following formulas. 'Short' or 'long' refers to the floating-point result.

  Target

  short a + T1 + R*(e + T3)

  long c + T1 + R*(f + T3)

  Constants used in the formulas are:

  R = number of elements in the array

  T1 = sum of times required to execute IHEJXI using IHEJXIY and IHEJXIA

  T3 = time for the appropriate conversion using IHEDMA

	IBM System/360 Model Number				
	30	40	50	65	75
a	1074	363	141	37.3	24.6
c	1123	381	147	38.5	25.6
e	400	139	54.4	16.2	10.9
f	418	139	53.2	15.1	9.8

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
UNDERFLOW (340)

Local Variables

None

Program Interface

Entry Points

IHESMF0

P7 = A(Parameter List) where Parameter List:
A(ADV)
A(Array DED)
A(Target)

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEDMA	Arithmetic Conversion Director
IHEJXI	Interleaved Array Indexer

Global Variables

None

Comments

Called by compiled code.

TITLE: SUM-INTERLEAVED REAL/COMPLEX SHORT FLOAT ARRAY (IHESMG)

Program Definition

Purpose and Usage

SUM-Interleaved Real/Complex Short Float Array is used to equate a short floating-point real or complex target to the sum of all the elements of an interleaved array of short floating-point reals or complexes, respectively.

Description

Method:

The elements of the array are added to the current sum in row-major order. For a complex argument, the summations of the real and imaginary parts are developed concurrently.

Implementation:

- Module size: 128 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the following formulas.

  Real a + T1 + R*b

  Complex c + T1 + R*d

  Constants used in the formulas are:

  R = number of elements in the array

  T1 = sum of times required to execute IHEJXI using the IHEJXIY and IHEJXIA

	IBM System/360 Model Number				
	30	40	50	65	75
a	935	314	121	32	21.4
b	167	43.9	15.2	4.2	2.1
c	1129	372	142	36.8	24.6
d	334	87.8	30.4	8.3	4.2

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
UNDERFLOW (340)

Local Variables

None

Program Interface

Entry Points

IHESMGR Entry for real arrays.

P7 = A(Parameter List) where Parameter List:
A(ADV)
A(Target)

IHESMGC Entry for complex arrays.

Linkage as for IHESMGR

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEJXI Interleaved Array Indexer

Global Variables

None

Comments

Called by compiled code.

TITLE: SUM-INTERLEAVED REAL/COMPLEX LONG FLOAT ARRAY (IHESMH)

Program Definition

Purpose and Usage

SUM-Interleaved Real/Complex Long Float Array is used to equate a long floating-point real or complex target to the sum of all the elements of an interleaved array of long floating-point reals or complexes, respectively.

Description

Method:

The elements of the array are added to the current sum in row-major order. For a complex argument, the summations of the real and imaginary parts are developed concurrently.

Implementation:

- Module size: 128 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the following formulas.

  Real a + T1 + R*b

  Complex c + T1 + R*d

  Constants used in these formulas are:

  R = number of elements in the array

  T1 = sum of times required to execute IHEJXI using IHEJXIY and IHEJXIA

	IBM System/360 Model Number				
	30	40	50	65	75
a	887	298	111	30.9	20.4
b	366	116	42.4	12.3	7.9
c	993	337	125	34.4	23.5
d	514	157	55.9	16	9.8

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
UNDERFLOW (340)

Local Variables

None

Program Interface

Entry Points

IHESMHR Entry for real arrays.

P7 = A(Parameter List) where Parameter List:
A(ADV)
A(Target)

IHESMHC Entry for complex arrays.

Linkage as for IHESMHR

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEJXI Interleaved Array Indexer

Global Variables

None

Comments

Called by compiled code.

TITLE: SUM-INTERLEAVED COMPLEX FIXED ARRAY (IHESMX)

Program Definition

Purpose and Usage

SUM-Interleaved Complex Fixed Array is used to equate a long or short floating-point complex target to the sum of all the elements of an interleaved array of fixed-point complexes.

Description

Method:

The elements of the array are added to the current sum in row-major order. For fixed-point arguments, each element is converted to floating-point by using routines of the Total Conversion Package. For a complex argument, the summations of the real and imaginary parts are developed concurrently. The precision specified in the source DED determines the precision of the target.

Implementation:

- Module size: 224 bytes
- Execution times:

  Approximate execution times in microseconds for the System/360 models given below are obtained from the following formula.

Source	Target	
binary	short	a + T1 + R*(e + 2*T3)

  Constants used in the formula are:

  R = number of elements in the array

  T1 = sum of times required to execute IHEJXI using IHEJXIY and IHEJXIA

  T3 = time for the appropriate conversion using IHEDMA

	IBM System/360 Model Number				
	30	40	50	65	75
a	1327	511	229	63.8	47.8
c	1319	515	227	63.7	47.4
e	712	232	87	25.1	16.4
f	770	240	89	24	15.2

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
UNDERFLOW (340)

Local Variables

None

Program Interface

Entry Points

IHESMX0

P7 = A(Parameter List) where Parameter List:
A(ADV)
A(DED for Array)
A(Target)

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEDMA	Arithmetic Conversion Director
IHEJXI	Interleaved Array Indexer

Global Variables

None

Comments

Called by compiled code.

TITLE: POLY (A,X) (A AND X REAL FIXED) (IHEYGF)

Program Definition

Purpose and Usage

POLY (A,X) for real fixed A and X has two options:

1. Vector X:

   Let the arguments be arrays declared as A(m:n) and X(p:q). Then the function computed is:

$$A(m) + \sum_{j=1}^{n-m} A(m+j) * \prod_{i=0}^{j-1} X(p+i)$$

   unless n = m, when result is A(m).

   If

   $q - p < n - m - 1$,

   then, for

   $p + i > q$, $X(p+i) = X(q)$.

2. Scalar X:

   This may be interpreted as a special case of vector X, that is, a vector with one element, X(1), which is equal to X. Then the function computed is:

$$\sum_{j=0}^{n-m} A(m+j) * X^{j}$$

   A floating-point result is obtained in both cases.

Description

Method:

1. Vector X, ($q - p \geq n - m - 1$):

   POLY (A,X) is evaluated by nested multiplication and addition, that is, (...(A(n)*X(k)+A(n-1))*X(k-1)+A(n-2))*...+A(m+1))*X(p)+A(m) where k = p + n - m - 1.

2. Vector X, ($q - p < n - m - 1$):

   In the expression above, the terms in X with subscript ranging from k down to q are all made equal to X(q). The evaluation is treated as for scalar X until sufficient terms in X have been made equal to X(q), when the computation continues as in 1.

3. Scalar X:

   Terms in X with subscript ranging from k to p are equal to X.

   For fixed-point arguments, each element is converted to floating-point, by using the Total Conversion Package.

4. The target precision is determined according to the highest precision specified in the source DED's.

Implementation:

- Module size: 432 bytes
- Execution times:

  Let the arguments be declared as A(m:n) and X(p:q), or X, and T be the time for one conversion using the Arithmetic Conversion Director (IHEDMA). Then the approximate execution times in microseconds for the System/360 models given are obtained from the appropriate formula. 'Short' or 'long' refers to the floating-point result.

  Scalar X:

  ```
  short   a + 2*T + (n-m) * (b+T)
  long    c + 2*T + (n-m) * (d+T)
  ```

  Vector X, (q - p ≥ n - m - 1):

  ```
  short   e + T + (n-m) * (f + 2*T)
  long    g + T + (n-m) * (h + 2*T)
  ```

  Vector X, (q - p < n - m - 1):

  ```
  short   i + 2*T + (n-m) * (b+T) + (q-p+1) * (j+T)
  long    k + 2*T + (n-m) * (d+T) + (q-p+1) * (l+T)
  ```

	IBM System/360 Model Number				
	30	40	50	65	75
a	2297	834	338	99	67.1
b	904	387	100	26	17.1
c	2408	870	351	102	69.2
d	1706	484	124	30.4	20.2
e	2629	910	370	97.1	64.6
f	1480	459	155	41.2	26.3
g	2740	946	383	99.4	66.7
h	3044	844	200	50	32.6
i	3370	1197	491	140	94
j	258	91	35	9.5	6
k	3581	1233	504	143	96.1
l	280	101	39	10.7	7.1

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
UNDERFLOW (340)

Local Variables

None

Program Interface

Entry Points

IHEYGFV Entry for vector A and vector X.

P7 = A(Parameter List) where Parameter List:
A(A ADV)
A(A DED)
A(X ADV)
A(X DED)
A(Target)

IHEYGFS Entry for vector A and scalar X.

P7 = A(PLIST)
PLIST = A(A ADV)
A(A DED)
A(X)
A(X DED)
A(Target)

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEDMA Arithmetic Conversion Director

Global Variables

None

Comments

Called by compiled code.

TITLE: POLY (A,X) (A AND X REAL SHORT FLOAT) (IHEYGS)

Program Definition

Purpose and Usage

POLY (A,X) for real short float A and X has two options:

1. Vector X:

   Let the arguments be arrays declared as A(m:n) and X(p:q). Then the function computed is:

$$A(m) + \sum_{j=1}^{n-m} A(m+j) * \prod_{i=0}^{j-1} X(p+i)$$

   unless n = m, when result is A(m).

   If

   q - p < n - m - 1,

   then, for

   p + i > q, X(p+i) = X(q).

2. Scalar X:

   This may be interpreted as a special case of vector X, that is, a vector with one element, X(1), which is equal to X. Then the function computed is:

$$\sum_{j=0}^{n-m} A(m+j)*X**j$$

   A floating-point result is obtained in both cases.

Description

Method:

1. Vector X, (q - p ≥ n - m - 1):

   POLY (A,X) is evaluated by nested multiplication and addition, that is, (...(A(n)*X(k)+A(n-1))*X(k-1)+A(n-2))*...+A(m+1))*X(p)+A(m) where k = p + n - m - 1.

2. Vector X, (q - p < n - m - 1):

   In the expression above, the terms in X with subscript ranging from k down to q are all made equal to X(q). The evaluation is treated as for scalar X until sufficient terms in X have been made equal to X(q), when the computation continues as in 1.

3. Scalar X:

   Terms in X with subscript ranging from k to p are equal to X.

4. The target is short precision.

Implementation:

- Module size: 240 bytes
- Execution times:

  Let the arguments be declared as A(m:n) and X(p:q), or X, and T be the time for one conversion using the Arithmetic Conversion Director (IHEDMA). Then the approximate execution times in microseconds for the System/360 models given are obtained from the appropriate formula.

  Scalar X:

  $a + (n - m)*b$

  Vector X:

  $(q - p \geq n - m - 1)$: $c + (n - m)*d$
  $(q - p < n - m - 1)$: $e + (n - m)*b + (q - p + 1)*f$

	IBM System/360 Model Number				
	30	40	50	65	75
a	1232	430	182	49.5	33.3
b	461	121	37.6	9.8	5.3
c	1871	623	259	69.6	45.4
d	490	128	40.9	10.5	5.7
e	2140	733	304	83.6	54.7
f	29	7.5	3.3	0.7	0.4

Errors Detected

Error and Exceptional Conditions:

```
I:     OVERFLOW (300)
       UNDERFLOW (340)
```

Local Variables

None

Program Interface

Entry Points

```
IHEYGSV    Entry for vector A and vector X

   P7      =   A(PLIST)
   PLIST   =   A(A ADV)
               A(X ADV)
               A(Target)

IHEYGSS    Entry for vector A and scalar X

   P7      =   A(Parameter List) where Parameter List:
               A(A ADV)
               A(X)
               A(Target)
```

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

None

Global Variables

None

Comments

Called by compiled code.

TITLE: POLY (A,X) (A AND X REAL LONG FLOAT) (IHEYGL)

Program Definition

Purpose and Usage

POLY (A,X) for real long float A and X has two options:

1. Vector X:

   Let the arguments be arrays declared as A(m:n) and X(p:q). Then the function computed is:

$$A(m) + \sum_{j=1}^{n-m} A(m+j) * \prod_{i=0}^{j-1} X(p+i)$$

   unless n = m, when result is A(m).

   If

   $q - p < n - m - 1$,

   then, for

   $p + i > q$, $X(p+i) = X(q)$.

2. Scalar X:

   This may be interpreted as a special case of vector X, that is, a vector with one element, X(1), which is equal to X. Then the function computed is:

$$\sum_{j=0}^{n-m} A(m+j) * X^{**}j$$

   A floating-point result is obtained in both cases.

Description

Method:

1. Vector X, ($q - p \geq n - m - 1$):

   POLY (A,X) is evaluated by nested multiplication and addition, that is, (...(A(n)*X(k)+A(n-1))*X(k-1)+A(n-2))*...+A(m+1))*X(p)+A(m) where k = p + n - m - 1.

2. Vector X, ($q - p < n - m - 1$):

   In the expression above, the terms in X with subscript ranging from k down to q are all made equal to X(q). The evaluation is treated as for scalar X until sufficient terms in X have been made equal to X(q), when the computation continues as in 1.

3. Scalar X:

   Terms in X with subscript ranging from k to p are equal to X.

4. The target is long precision.

Implementation:

- Module size: 240 bytes
- Execution times:

  Let the arguments be declared as A(m:n) and X(p:q), or X, and T be the time for one conversion using the Arithmetic Conversion Director (IHEDMA). Then the approximate execution times in microseconds for the System/360 models given are obtained from the appropriate formula.

  Scalar X:

  $a + (n - m)*b$

  Vector X:

  $(q - p \geq n - m - 1)$: $c + (n - m)*d$
  $(q - p < n - m - 1)$: $e + (n - m)*b + (q - p + 1)*f$

	IBM System/360 Model Number				
	30	40	50	65	75
a	1320	451	189	49.6	33.3
b	1241	308	56.9	13.1	7.3
c	1959	644	266	69.7	45.4
d	1270	316	60.2	13.7	7.7
e	2228	755	311	83.6	54.7
f	29	7.5	3.3	0.7	0.4

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
UNDERFLOW (340)

Local Variables

None

Program Interface

Entry Points

IHEYGLV Entry for vector A and vector X.

P7 = A(PLIST)
PLIST = A(A ADV)
A(X ADV)
A(Target)

IHEYGLS Entry for vector A and scalar X.

P7 = A(Parameter List) where Parameter List:
A(A ADV)
A(X)
A(Target)

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

None

Global Variables

None

Comments

Called by compiled code.

TITLE: POLY (A,X) (A AND X COMPLEX FIXED) (IHEYGX)

Program Definition

Purpose and Usage

POLY (A,X), for complex fixed A and X has two options:

1. Vector X:

   Let the arguments be arrays declared as A(m:n) and X(p:q). Then the function computed is:

$$A(m) + \sum_{j=1}^{n-m} A(m+j) * \prod_{i=0}^{j-1} X(p+i)$$

   unless n = m, when result is A(m).

   If

   $q - p < n - m - 1$,

   then, for

   $p + i > q$, $X(p+i) = X(q)$.

2. Scalar X:

   This may be interpreted as a special case of vector X, that is, a vector with one element, X(1), which is equal to X. Then the function computed is:

$$\sum_{j=0}^{n-m} A(m+j) * X^{**}j$$

   A floating-point result is obtained in both cases.

Description

Method:

1. Vector X, ($q - p \geq n - m - 1$):

   POLY (A,X) is evaluated by nested multiplication and addition, that is, (...(A(n)*X(k)+A(n-1))*X(k-1)+A(n-2))*...+A(m+1))*X(p)+A(m) where k = p + n - m - 1.

2. Vector X, ($q - p < n - m - 1$):

   In the expression above, the terms in X with subscript ranging from k down to q are all made equal to X(q). The evaluation is treated as for scalar X until sufficient terms in X have been made equal to X(q), when the computation continues as in 1.

3. Scalar X:

   Terms in X with subscript ranging from k to p are equal to X.

   For fixed-point arguments, each element is converted to floating-point, by using the Total Conversion Package.

4. The target precision is determined by the highest precision indicated in the array/scalar DED's.

Implementation:

- Module size: 688 bytes
- Execution times:

  Let the arguments be declared as A(m:n) and X(p:q), or X, and T be the time for one conversion using the Arithmetic Conversion Director (IHEDMA). Then the approximate execution times in microseconds for the System/360 models given are obtained from the appropriate formula. 'Short' or 'long' refers to the floating-point result.

  Scalar X:

short	a + 4*T + (n-m)*(b+2*T)
long	c + 4*T + (n-m)*(d+2*T)

  Vector X, (q - p ≥ n - m - 1):

short	e + 2*T + (n-m)*(f+4*T)
long	g + 2*T + (n-m)*(h+4*T)

  Vector X, (q - p < n - m - 1):

short	i + 4*T + (n-m)*(b+2*T)+(q-p+1)*(j+2*T)
long	k + 4*T + (n-m)*(d+2*T)+(q-p+1)*(l+2*T)

	IBM System/360 Model Number				
	30	40	50	65	75
a	3245	1174	478	142	95.4
b	2345	664	220	58.3	33.5
c	3471	1221	496	145	97.6
d	5519	1433	301	73.2	42.6
e	3187	1114	447	127	83.5
f	4368	1226	399	103	58.1
g	3533	1161	465	129	85.7
h	10636	2746	556	133	76.4
i	4087	1459	592	171	114
j	545	187	72.8	21.1	12.7
k	4243	1506	610	176	116
l	567	196	76.8	22.3	13.6

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
UNDERFLOW (340)

Local Variables

None

Program Interface

Entry Points

IHEYGXV Entry for vector A and vector X.

```
   P7      =   A(Parameter List) where Parameter List:
               A(A ADV)
               A(A DED)
               A(X ADV)
               A(X DED)
               A(Target)
```

IHEYGXS Entry for vector A and scalar X.

```
   P7      =   A(PLIST)
   PLIST   =   A(A ADV)
               A(A DED)
               A(X)
               A(X DED)
               A(Target)
```

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

IHEDMA Arithmetic Conversion Director

Global Variables

None

Comments

Called by compiled code.

TITLE: POLY (A,X) (A AND X COMPLEX SHORT FLOAT) (IHEYGW)

Program Definition

Purpose and Usage

POLY (A,X) for complex short float A and X has two options:

1. Vector X:

   Let the arguments be arrays declared as A(m:n) and X(p:q). Then the function computed is:

$$A(m) + \sum_{j=1}^{n-m} A(m+j) * \prod_{i=0}^{j-1} X(p+i)$$

   unless n = m, when result is A(m).

   If

   q - p < n - m - 1,

   then, for

   p + i > q, X(p+i) = X(q).

2. Scalar X:

   This may be interpreted as a special case of vector X, that is, a vector with one element, X(1), which is equal to X. Then the function computed is:

$$\sum_{j=0}^{n-m} A(m+j)*X**j$$

   A floating-point result is obtained in both cases.

Description

Method:

1. Vector X, (q - p ≥ n - m - 1):

   POLY (A,X) is evaluated by nested multiplication and addition, that is, (...(A(n)*X(k)+A(n-1))*X(k-1)+A(n-2))*...+A(m+1))*X(p)+A(m) where k = p + n - m - 1.

2. Vector X, (q - p < n - m - 1):

   In the expression above, the terms in X with subscript ranging from k down to q are all made equal to X(q). The evaluation is treated as for scalar X until sufficient terms in X have been made equal to X(q), when the computation continues as in 1.

3. Scalar X:

   Terms in X with subscript ranging from k to p are equal to X.

4. The target is short precision.

Implementation:

- Module size: 280 bytes
- Execution times:

  Let the arguments be declared as A(m:n) and X(p:q), or X, and T be the time for one conversion using the Arithmetic Conversion Director (IHEDMA). Then the approximate execution times in microseconds for the System/360 models given are obtained from the appropriate formula.

  Scalar X:

  $a + (n - m)*b$

  Vector X:

  $(q - p \geq n - m - 1)$: $c + (n - m)*d$
  $(q - p < n - m - 1)$: $e + (n - m)*b + (q - p + 1)*f$

	IBM System/360 Model Number				
	30	40	50	65	75
a	1396	475	198	54.9	36.1
b	1672	425	126	30.5	15
c	2035	667	275	75	48.1
d	1701	432	129	31.2	15.4
e	2304	775	320	88.9	57.4
f	29	7.5	3.3	0.7	0.4

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
UNDERFLOW (340)

Local Variables

None

Program Interface

Entry Points

IHEYGWV Entry for vector A and vector X

```
P7      =   A(PLIST)
PLIST   =   A(A ADV)
            A(X ADV)
            A(Target)
```

IHEYGWS Entry for vector A and scalar X

```
P7      =   A(Parameter List) where Parameter List:
            A(A ADV)
            A(X)
            A(Target)
```

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

None

Global Variables

None

Comments

Called by compiled code.

TITLE: POLY (A,X) (A AND X COMPLEX LONG FLOAT) (IHEYGZ)

Program Definition

Purpose and Usage

POLY (A,X), for complex long float A and X has two options:

1. Vector X:

   Let the arguments be arrays declared as A(m:n) and X(p:q). Then the function computed is:

$$A(m) + \sum_{j=1}^{n-m} A(m+j) * \prod_{i=0}^{j-1} X(p+i)$$

   unless n = m, when result is A(m).

   If

   $q - p < n - m - 1$,

   then, for

   $p + i > q$, $X(p+i) = X(q)$.

2. Scalar X:

   This may be interpreted as a special case of vector X, that is, a vector with one element, X(1), which is equal to X. Then the function computed is:

$$\sum_{j=0}^{n-m} A(m+j) * X^{**}j$$

   A floating-point result is obtained in both cases.

Description

Method:

1. Vector X, ($q - p \geq n - m - 1$):

   POLY (A,X) is evaluated by nested multiplication and addition, that is, (...*A(n)*X(k)+A(n-1))*X(k-1)+A(n-2))*...+A(m+1))*X(p)+A(m) where k = p + n - m - 1.

2. Vector X, ($q - p < n - m - 1$):

   In the expression above, the terms in X with subscript ranging from k down to q are all made equal to X(q). The evaluation is treated as for scalar X until sufficient terms in X have been made equal to X(q), when the computation continues as in 1.

3. Scalar X:

   Terms in X with subscript ranging from k to p are equal to X.

   For fixed-point arguments, each element is converted to floating-point, by using the Total Conversion Package.

4. The target is long precision.

Implementation:

- Module size: 280 bytes
- Execution times:

  Let the arguments be declared as A(m:n) and X(p:q), or X, and T be the time for one conversion using the Arithmetic Conversion Director (IHEDMA). Then the approximate execution times in microseconds for the System/360 models given are obtained from the appropriate formula.

  Scalar X:

  $a + (n - m)*b$

  Vector X:

  $(q - p \geq n - m - 1)$: $c + (n - m)*d$
  $(q - p < n - m - 1)$: $e + (n - m)*b + (q - p + 1)*f$

	IBM System/360 Model Number				
	30	40	50	65	75
a	1572	517	211	54.9	36.1
b	4824	1183	203	44.2	23
c	2211	710	288	75	48.1
d	4853	1192	207	44.9	23.4
e	2480	821	333	89	57.4
f	29	7.5	3.3	0.7	0.4

Errors Detected

Error and Exceptional Conditions:

I: OVERFLOW (300)
UNDERFLOW (340)

Local Variables

None

Program Interface

Entry Points

IHEYGZV Entry for vector A and vector X.

P7 = A(PLIST)
PLIST = A(A ADV)
A(X ADV)
A(Target)

IHEYGZS Entry for vector A and scalar X.

P7 = A(Parameter List) where Parameter List:
A(A ADV)
A(X)
A(Target)

Exit Conditions

Normal. Returns to caller via the link register.

Routines Called

None

Global Variables

None

Comments

Called by compiled code.

READER'S COMMENT FORM

CALL/360-OS

GY20-0569-0

PL/I SM - Vol. III

Please comment on the usefulness and readability of this publication, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you wish a reply, be sure to include your name and address.

COMMENTS

fold

fold

fold

fold

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A. FOLD ON TWO LINES, STAPLE AND MAIL.

YOUR COMMENTS PLEASE...

Your comments on the other side of this form will help us improve future editions of this publication. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material.

Please note that requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or the IBM branch office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY...

IBM Corporation
112 East Post Road
White Plains, N. Y. 10601

Attention: Technical Publications

fold fold

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

CALL/360-OS PL/I SM Volume 3 Printed in U.S.A. GY20-0569-0

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York 10601
(USA only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)