

NRL Report 6106
NAREC Reference #28

NIP

A Floating-Point Interpretive Programming System for the NAREC Computer

E. A. STENNETT AND L. S. BEARCE

*Radio Techniques Branch
Radio Division*

AND

R. M. MASON

*Applied Mathematics Staff
Office of Director of Research*

December 18, 1964



U. S. NAVAL RESEARCH LABORATORY
Washington, D.C.

CONTENTS

Abstract.....	1
Problem Status	1
Authorization	1
 INTRODUCTION	 1
 PART 1 – REGULAR-PRECISION INTERPRETIVE ROUTINE (NIP I).....	 1
1. Name of Subroutine.....	1
2. Class	1
3. Purpose	2
4. Language Used	2
5. Authors	2
6. Formal Statement	2
a. Conventions and Terminology	2
b. Definitions of Operations	4
7. Entry and Return	9
8. Output.....	10
9. Tape Label	10
10. Address Information	10
11. External Working Memory	10
12. Versatility	10
13. Acknowledgements.....	10
 PART 2 – EXTENDED-PRECISION INTERPRETIVE ROUTINE (NIP II)	 10
1. Name of Subroutine	10
2. Class.....	10
3. Purpose	10
4. Language Used.....	11
5. Author.....	11
6. Formal Statement.....	11
a. Conventions and Terminology	11
b. Definitions of Operations	11
7. Entry and Return.....	11
8. Output.....	12
9. Tape Labels	12
10. Address Information	12
11. External Working Memory	13
12. Versatility	13
13. Acknowledgments	13

Appendix A – The Floating-Point System	13
Appendix B – An Example of a Program Written in Extended Precision	17
Appendix C – Block Diagrams of the Elementary Extended-Precision Operations: Addition, Complementation, Multiplication, and Division	28
Appendix D – Difference Plots for the Elementary Functions: Square Root, Cosine, Arc Sine, Logarithm, and Exponential	32
Appendix E – Timing of NIP Pseudo-Orders	35
Appendix F – NIP Error Stops	36
Appendix G – Summary of NAR Coding Symbols for NAREC Orders	37
Appendix H – Summary of Basic and NAR Coding Symbols for NIP Pseudo-Orders, Including Brief Descriptions of the Associated NIP Operations	38

NIP

A Floating-Point Interpretive Programming System for the NAREC Computer

E. A. STENNETT AND L. S. BEARCE

*Radio Techniques Branch
Radio Division*

and

R. M. MASON

*Applied Mathematics Staff
Office of Director of Research*

An interpretive programming system has been written for the U.S. Naval Research Laboratory's medium-speed electronic digital computer (NAREC). This system, known as NIP (acronym for NAREC Interpretive Programming system), provides for automatic floating-point computation over the range 10^{-1233} to 10^{+1233} in either of two modes. One mode, called "extended precision," allows for computational results containing as many as nineteen significant decimal digits. The other (and historically the first) mode, called "regular precision," allows for as many as eleven significant decimal digits. Using the same program, it is possible to compute in either mode, which provides a basis for determining the cumulative effect of computational precision on results. This report discusses conventions and defines instructions which are available for writing a program in the NIP language. A brief description of the floating-point system, an example of a program written in extended precision, and a table of instruction execution times are included in the appendixes.

INTRODUCTION

The purpose of the NAREC Interpretive Programming system (NIP) is to expedite the programming process by providing easy-to-use, interpreted, machine-like orders, called pseudo-orders, which reduce the time and effort required for programming and checking out routines. Since the NAREC (Naval Research Electronic Computer) is a fixed-point digital machine, the programming of problems in machine language often requires that much thought and effort be devoted to the task of scaling. Not only must suitable bounds be maintained on problem variables, but also considerable time and effort often must be expended in order to insure significant results. The programming system described in this report eliminates, for practical purposes, the problem of scaling through the use of a pseudo-order code. Various mathematical operations may be performed on numbers expressed in floating-point form, and various logical operations designed to facilitate the task of program-

ming may be employed. The NIP system will accept as input, decimal numbers which lie over the range 10^{-999} to 10^{+999} , and computation may be carried out in either of two modes—the "regular-precision" mode which allows for computational results containing as many as eleven significant decimal digits, or the "extended-precision" mode which permits as many as nineteen significant decimal digits.

This report is divided into two parts: Part I describes the system as it is applied to regular-precision computation; Part II describes the system as it is applied to extended-precision computation. Both parts follow the standard form for reporting NAREC routines.

PART I REGULAR-PRECISION INTERPRETIVE ROUTINE (NIP I)

1. Name of Subroutine

NIP I—Floating-Point Interpretive Routine.

2. Class

Y1, Interpretive Routine, Floating-Point.

NRL Problems B02-03 and R01-08; Projects RF 006-02-41-4351 and RR 003-09-41-5101. This is an interim report; work on this problem is continuing.

Manuscript submitted April 16, 1964.

3. Purpose

NIP I is an interpretive subroutine, written in machine language, which provides a programming language for automatic floating-point computations on the NAREC. Through the use of a pseudo-order code, floating-point programs may be written using interpretive language, in much the same way as fixed-point programs may be written using machine language. The NIP language, however, is much more powerful and convenient. It contains not only floating-point arithmetic and elementary function operations, but also a substantial group of operations which may be classified as programming aids. Included in this group are automatic counters, address modifiers, marker detectors, etc. Moreover, there are input-output operations which provide both automatic conversions and considerable format flexibility. Finally, in addition to the orders which are used directly for problem solving, a convenient tracking and monitoring order has been incorporated.

4. Language Used

Actual NAREC machine language.

5. Authors

Edwin A. Stennett and Loren S. Bearce, Code 5425, and Robert M. Mason, Code 4558.

6. Formal Statement

a. *Conventions and Terminology*—The material presented under this heading is intended to provide the reader with sufficient background to understand the definitions of the operations, which follow in Section 6, and to enhance his ability to apply these operations successfully.* Some material pertaining to the extended-precision program (NIP II) is included in this section in order to minimize redundancy in Part II of this report.

1. All ambiguous decimal numbers in this formal statement are underlined unless they appear as exponents or subscripts; all other numbers are expressed in hexadecimal notation.

*For further clarification concerning terminology, see R. M. Mason, "A Programming Glossary for NAREC Users," NRL Report 5779, August 6, 1962.

2. Pseudo-instruction words follow the same rule of formation as machine-instruction words. This rule is illustrated on page eleven of NRL Report 5779 and also in NAREC Reference #22. All pseudo-instructions which require two addresses, or an address and a parameter n , include parentheses in the basic coding symbol and occupy full computer words.

3. The range of the signed-magnitude integer n , used as a parameter in the address portion of certain pseudo-instructions, is in general given by the inequality $-fff \leq n \leq +fff$. When this parameter is interpreted by NIP, it must be expressed as a positive three-digit hexadecimal number preceded by a binary sign digit, *i.e.*, complemented number representation is unacceptable. If the NAREC Assembly Routine* is used, the number n must be expressed as a positive decimal integer; moreover, it also must be preceded by the minus sign character if n is negative, since $-4095 \leq n \leq 4095$.

4. The range of the address s is given by the inequality $0400 \leq s \leq 4ffe$, unless otherwise specified. However, if use of NIP II (extended precision) is anticipated, then the range of this address is given by $0900 \leq s \leq 4ffe$, unless otherwise specified. In general, if any part (or even all) of the address portion of a pseudo-instruction has not been specified, that part should be left zero in order to allow for possible future extensions of the system.

5. Each regular-precision NIP floating-point number consists of two quantities:

a. a signed 44-bit fractional part, say M , and
 b. a signed 12-bit exponent part, say q .

These two quantities, contained in two entire words termed a "word-pair," occupy successive storage locations, herein designated by s and $s+1$, in the computer memory and are transferred and treated as a unit. Reference to a word-pair, symbolized (M,q) , is required for and given to only the first address s , unless otherwise stated. Thus, the statement, "the contents of storage location s " is understood to be identical to "the word-pair contents of successive storage locations s and $s+1$ (sometimes referred to as s_0 and s_1)."

When (M,q) represents a floating-decimal number, it has the interpretation $M \times 10^q$ where M and q are binary-coded-decimal, signed-magnitude numbers. Moreover, M satisfies the

*NAREC Bulletin No. 31, December 19, 1961.

inequality $-1 < M < +1$ at all times, and q satisfies the inequality $-1000 < q < +1000$ during readin and $-100 < q < +100$ during printout (using the f0 pseudo-order). When (M,q) represents a floating-hexadecimal (or floating-binary) number, it is interpreted as $M \times 2^q$ where M and q are "binary 2's complement" numbers. Furthermore, M satisfies the inequality $-1 < M < +1$ and q satisfies the inequality $-fff \leq q \leq +fff$ at all times. A floating-decimal, nonzero number is said to be "adjusted" (normalized) or "in adjusted form" if it meets the additional condition: $|M| \geq 1/10$. Similarly, a floating-hexadecimal, nonzero number is said to be "adjusted" or "in adjusted form" provided that $|M| \geq 1/2$. The regular-precision portion of the quantity M is contained in and fills an arbitrary storage location. The quantity q is contained in the next storage location $s+1$ but fills only the left address portion. The remainder of $s+1$ must be all zeros, except when the extended-precision mode is employed — in which case the remainder of $s+1$ contains the extended-precision portion of M . Table 1 below is presented to show how typical floating-point numbers are represented.

6. Each floating-hexadecimal, nonzero number that enters a computation must be adjusted prior to use. It is sufficient to do this one time with the JP pseudo-order because all numbers which result from succeeding pseudo-order operations are in adjusted form. Also, all floating-decimal input that is converted by the CDDB or CDB pseudo-orders is automatically adjusted.

7. Floating-point zeros are defined to be those numbers (M,q) having the arithmetic portion M equal to zero for any q . The particular value of q is immaterial in any zero number; thus the

decimal and hexadecimal representations of zero are identical, which is as it should be. Each zero is considered exact, *i.e.*, expressed with infinite precision.

8. Floating-point markers are defined to be those word-pairs $(s,s+1)$ that are not numbers, *i.e.*, that have the sign digit of s equal to one and the remainder of the arithmetic portion (the numeric part) of s equal to zero for any $s+1$.

9. A "stopper" is defined as a floating-point marker in which both words have the sign digit equal to one and the numeric part equal to zero. The state of the overflow digits is immaterial in both markers and stoppers.

10. The main working register is a programmed construction called the P register (where P denotes the adjective "pseudo"). It automatically receives the result of most operations, and also often serves as an operand source; hence the answer from a previous operation is often kept in proper position to be operated on by the next pseudo-instruction. This "register," which occupies locations 03b4 and 03b5 (sometimes referred to as P_0 and P_1), then takes a prominent role in most pseudo-orders.

11. In addition to the P register, two other programmed registers are of importance—the σ and γ registers. The σ register, sometimes referred to as σ_0 and σ_1 , occupies locations 0202 and 0203 and performs much the same function as the E counter on the NAREC main frame. The right-hand side of σ_0 (location 0202) always contains the address of the pseudo-order currently being interpreted and is recorded automatically before each error stop. The γ register occupies location 03b3 and is analogous to the C register of the NAREC main frame, *i.e.*, the

TABLE I
Examples of Floating-Point Numbers

Number in Decimal	Regular-Precision Hexadecimal	Extended-Precision Hexadecimal
0.7	s 0.b33 33 3333 33	0.b33 33 3333 33
	s+1 0.000 00 0000 00	0.000 33 3333 33
6.4	s 0.ccc cc cccc cd	0.ccc cc cccc cc
	s+1 0.003 .00 0000 00	0.003 cc cccc cd
0.10712	s 0.db6 db 6db6 db	0.db6 db 6db6 db
	s+1 f.ffd 00 0000 00	f.ffd 6d b6db 6e
-0.66666	s f.555 55 5555 55	f.555 55 5555 55
	s+1 0.000 00 0000 00	0.000 55 5555 55

pseudo-order currently being interpreted is stored at location 03b3.

b. *Definitions of Operations*—The definitions of the operations which follow include statements of all changes in the contents of the P register and all storage locations. It should be noted specifically that the transfer of a word-pair between storage and the P register leaves the initial word-pair unchanged. The immediate contents of the A and U machine registers are destroyed after entering the interpretive mode. However, the contents of the P register are invariant under all operations in machine language, unless the contents of its storage locations are disturbed directly.

Pseudo-Order
Number

Basic Coding
Symbol

d0

ADL (n,s)
-fff ≤ n ≤ fff

Add the integer n to the left address of the word at storage location s, and place the result in location s.

d1

ADR (n,s)
-fff ≤ n ≤ fff

Add the integer n to the right address of the word at storage location s, and place the result in location s.

d2

MCL s

If the P register contains a marker, transfer interpretive control to the left pseudo-order of storage location s; otherwise, proceed to the next pseudo-order in sequence.

d3

MCR s

If the P register contains a marker, transfer interpretive control to the right pseudo-order of storage location s; otherwise, proceed to the next pseudo-order in sequence.

d4

NCL s

If the sign digit of the fractional part of the P register is zero, exit the interpretive mode of control, enter the machine-language mode, and

transfer control to the left order of storage location s; otherwise, proceed to the next pseudo-order in sequence.

d5

NCR s

If the sign digit of the fractional part of the P register is zero, exit the interpretive mode of control, enter the machine-language mode, and transfer control to the right order of storage location s; otherwise, proceed to the next pseudo-order in sequence.

d6

CDDB (s₁,s₂)

Starting at s₁, convert in sequence each nonzero decimal-point, or fixed-point *number*, located in the first of a pair of successive storage locations (allotted for standard floating-point storage), from possibly short-word binary-coded-decimal form to adjusted floating binary, and store the result in that pair of locations. Proceed until either a stopper is encountered or all of the numbers between s₁ and s₂, inclusive, have been converted. Leave those word-pairs (including markers) unchanged whose bits to the right of the sign character in the first word are zero if the first word is not in shortened form; otherwise, make both words equal to zero. If no decimal-point indicator is included, a number will be treated as a fixed-point fraction with the decimal point understood to be immediately preceding the first digit to the right of the sign character. All numbers to be converted must have either a plus (+) or minus (−) sign as a first character, with twelve characters being the maximum allowed for a given number. Note that if a plus or minus character (which is indistinguishable from an "a" or "b" character after standard input) is not included in any first word, that word-pair will remain unchanged. If a decimal-point is to be included, the symbol "d" must be used in its place. (Thus, the maximum number of digits allowed for a binary-coded-decimal single-precision number which includes a decimal-point indicator is ten.) The prior contents of the second word of each word-pair allocated to a number are not critical.

d7

RSL (s₁,s₂)

Copy the contents of the right address portion of s₁ in the left address portion of s₂.

d8	LSL (s_1, s_2)	de	REP n $0 \leq n \leq fff$
Copy the contents of the left address portion of s_1 in the left address portion of s_2 .		Repeat (perform n times) the sequence of instructions starting with a pseudo-order at location $s+2(L)$ and terminating with the instruction FR $s+1$; then transfer interpretive control to the right-hand side of location s , <i>i.e.</i> , $s(R)$. The repeat pseudo-instruction must appear only in the left-hand part of location s , <i>i.e.</i> , $s(L)$; furthermore, the entire next word $s+1$ must be reserved for the interpretive routine to use as a working location (the prior contents of $s+1$ are not critical).	
d9	LSR (s_1, s_2)	df	SRNL s
Copy the contents of the left address portion of s_1 in the right address portion of s_2 .		Exit the interpretive mode of control, place the right half of the instruction word, followed by six hexadecimal zeros, in the U register and transfer machine control to $s(L)$, the first instruction of the subroutine. After the subroutine exit, re-enter the interpretive mode of control and transfer control to the next <i>left-hand</i> pseudo-order in sequence. In accordance with established convention, the subroutine's exit instruction must be a transfer of machine control to the left order at that address which immediately follows the address given (by the interpretive routine) in the left address portion of the A register at the time of entry. The df pseudo-instruction must appear only in the left half of the instruction word, the right half being reserved for subroutine parameter specification.*	
da	Sin P	dc	CMP s
Compute the sine of the contents of the P register, expressed in radians, and place the adjusted result in the P register. (In the extended-precision mode, an error stop will occur if the magnitude of the argument is greater than 2^{76} .)		Let α and β denote the contents of the P register and of storage location s , respectively. If either α or β is a marker, or if $\alpha - \beta$ exceeds the allowed range, then this operation is undefined; otherwise, (a) if $\alpha < \beta$, interpretation proceeds to the next pseudo-order in sequence, (b) if $\alpha = \beta$, interpretation proceeds to the pseudo-order after next in sequence, and (3) if $\alpha > \beta$, interpretation proceeds to the second pseudo-order after the next in sequence.	
db	Arc Sin P	dd	BYP n $0 \leq n \leq fff$
If the magnitude of the contents of the P register is less than or equal to one, calculate the principal value (<i>i.e.</i> , the angle in the range $-\pi/2$ to $\pi/2$), in radians, of the angle whose sine is equal to the contents of the P register, and place the adjusted result in the P register; otherwise, halt on an error stop.		Bypass the next pseudo-order in sequence for the first n interpretations of this pseudo-order; then on each of the following interpretations do nothing but proceed directly to the next pseudo-order in sequence. During each successive interpretation, the integer in the address portion of the instruction is changed to take on successive values from the sequence $n-1, n-2, \dots, 1, 0, 0, 0, \dots$, unless changed by the program.	
dc	CMP s	e0	SKIP
Let α and β denote the contents of the P register and of storage location s , respectively. If either α or β is a marker, or if $\alpha - \beta$ exceeds the allowed range, then this operation is undefined; otherwise, (a) if $\alpha < \beta$, interpretation proceeds to the next pseudo-order in sequence, (b) if $\alpha = \beta$, interpretation proceeds to the pseudo-order after next in sequence, and (3) if $\alpha > \beta$, interpretation proceeds to the second pseudo-order after the next in sequence.		Proceed to the next pseudo-order in sequence.	
dd	BYP n $0 \leq n \leq fff$	e1	JP
Bypass the next pseudo-order in sequence for the first n interpretations of this pseudo-order; then on each of the following interpretations do nothing but proceed directly to the next pseudo-order in sequence. During each successive interpretation, the integer in the address portion of the instruction is changed to take on successive values from the sequence $n-1, n-2, \dots, 1, 0, 0, 0, \dots$, unless changed by the program.		*NOTE: No monitor printout occurs in connection with this order; however, a monitor-type printout can be obtained any time the monitor is on by using the particular pseudo-instruction NI 0390 which, oddly enough, will also immediately return interpretive control to the next pseudo-order in sequence.	

Adjust the contents of the P register if $P \neq 0$; otherwise, proceed to the next pseudo-instruction in sequence.	e5	NR s
e2 Transfer interpretive control to the left pseudo-order of storage location s.	FL s	Enter the NAREC machine-language mode of control and transfer control to the right order of storage location s. The re-entry options applicable after using this pseudo-instruction are the same as those for NL s, described above.
e3 Transfer interpretive control to the right pseudo-order of storage location s.	FR s	e6 s P Transfer the word-pair at storage location s to the P register.
e4 Enter the NAREC machine-language mode of control and transfer control to the left order of storage location s. Upon the occurrence of a special re-entry instruction LO 0390, interpretative control will be transferred to the next pseudo-instruction in the original sequence, provided that the contents of the interpretive sequence register σ are the same as they were when the e4 pseudo-order was last interpreted. Table 2 describes and summarizes the alternative re-entry orders which may be used and their corresponding functions. If it is temporarily necessary to enter the interpretive mode of control within the routine designated by the e4 instruction, the contents of σ (locations 0202 and 0203) must first be saved before such an entry. Moreover, this information must be restored to σ after leaving the interpretive mode of control and before executing the re-entry order.	NL s	e7 -s P Transfer the negative of the word-pair at location s to the P register.
		e8 P s Transfer the word-pair contained in the P register to storage location s.
		e9 s Pd Add the floating-point number at storage location s to the contents of the P register and place the adjusted result in the P register.
		ea -s Pd Subtract the floating-point number at storage location s from the contents of the P register and place the adjusted result in the P register.

TABLE 2
Re-Entry Instruction Table

Instruction	Description
LO 0390	Transfer interpretive control to the next pseudo-order in the original sequence.
RO 0203	Same as LO 0390 except that a monitor printout cannot occur.
RO 0258	Same as RO 0203 except that interpretive control is transferred to the pseudo-order after next in the original sequence.
RO 0244	Same as RO 0203 except that interpretive control is transferred to the second pseudo-order after next in the original sequence.

- eb MP s If the sign digit of n is a zero, leave the P register unchanged and record (via 33 orders) the sign and first $|n|$ digits of the converted fractional part followed by the sign and a two-digit exponent. (Note that truncation is accomplished via the 91 order, and at the completion of each print a 91 order resets the machine for full twelve-character (33 order) records.)
- Multiply the contents of the P register by the floating-point number at storage location s and place the adjusted, rounded, high-order product in the P register. (In the extended-precision mode, if the magnitude of the resulting product is greater than 2^{+4095} , an error stop will occur.)
- ec P/s If the sign digit of n is a one, simply place the converted number in the P register without recording. (If $|n| < 11$, the last binary-coded-decimal digit of the fractional part in P_0 will be followed by at least one hexadecimal b.)
- If the number at s is not equal to zero, divide the contents of the P register by the floating-point number at storage location s and place the adjusted, rounded, high-order quotient in the P register; otherwise, halt on an error stop. (In the extended-precision mode, if the magnitude of the resulting quotient is greater than 2^{+4095} , an error stop will occur.)
- ed Exp P If the floating-binary number in the P register is suitably scaled, round and convert it to decimal; then, using 90 orders, print the decimal number as a single word (not followed by a tab) with a maximum of thirteen characters, including the sign and decimal point. Substituting a space character for each leading zero, allow "i" characters to the left of the decimal point, in addition to the sign for the integer part of the number, and "j" characters to the right of the decimal point for the fractional part. Place the sign digit just prior to the first nonspace character.
- If the contents of P are less than 2^{11} , form the exponential of the floating-point number in the P register and place the adjusted result in the P register; otherwise, halt on an error stop.
- ee Ln P If the number is not suitably scaled, round and convert it to adjusted floating decimal: Then, using 90 orders, print the sign, first digit, decimal point, remaining ten-digit fractional part, and a five-character exponent (not followed by a tab). This exponent may contain up to four digits; space characters are substituted for leading zeros, and the sign character is placed just prior to the first nonspace character.
- If the contents of P are greater than zero, form the natural logarithm of the floating-point number in the P register and place the adjusted result in the P register; otherwise, halt on an error stop.
- ef POW n
$$0 \leq |n| \leq \left\lfloor \frac{\text{fff}}{\text{Log}_2|P|} \right\rfloor \leq \text{fff}$$
- Raise the floating-point number in the P register to the positive or negative n th power, where n is an integer satisfying the above inequality, and place the adjusted result in the P register.
- f0 PT n
$$0 \leq |n| \leq b$$

$$n = \pm |n|$$
- Round to $|n|$ decimal places (treating $n = 0$ as a special case meaning $n = b$) and form a conversion of the number in the P register from floating binary to adjusted floating decimal.
- If the floating-binary number in the P register is suitably scaled, round and convert it to decimal; then, using 90 orders, print the decimal number as a single word (not followed by a tab) with a maximum of thirteen characters, including the sign and decimal point. Substituting a space character for each leading zero, allow "i" characters to the left of the decimal point, in addition to the sign for the integer part of the number, and "j" characters to the right of the decimal point for the fractional part. Place the sign digit just prior to the first nonspace character.
- If the number is not suitably scaled, round and convert it to adjusted floating decimal: Then, using 90 orders, print the sign, first digit, decimal point, remaining ten-digit fractional part, and a five-character exponent (not followed by a tab). This exponent may contain up to four digits; space characters are substituted for leading zeros, and the sign character is placed just prior to the first nonspace character.
- In order to be suitably scaled, the number must not be so large that more than i characters are necessary. Moreover, it must not be so small that more than j characters are needed to show at least one nonzero digit—that is, of course, unless the number is a zero (in which case it always is considered suitably scaled).
- The f1 pseudo-order is contained in a single-address instruction with (i,j) in the (2nd, 4th) or (8th, 10th) hexadecimal digits depending, respectively, on whether the instruction occurs in the left or right address portion of a word;

i.e., $\overline{0i0j|f1}$. If i and j should be specified such that $(i+j) > b$, the program will halt on an error stop. Note that if the hexadecimal digit preceding i is made to equal 8, virtual zero printout is possible. In other words, if more than j characters are needed to show at least one non-zero digit, the number will still be considered suitably scaled, and a sign, decimal point, and j zero characters will be recorded.

f2

 \sqrt{P}

If the contents of the P register are greater than or equal to zero, form the square root of the floating-point number in the P register and place the adjusted result in the P register; otherwise, halt on an error stop.

f3

PC n
 $0 \leq n \leq 3f$

Punch on the output tape, or place in the line printer memory, the last six binary digits of n to represent a single character. Note that n is specified in the address portion of the pseudo-instruction and that the P register is unaffected by it.

f4

FCL s

If the sign bit (digit 0) of the fractional part of the P register is 0, transfer interpretive control to the left pseudo-order of storage location s ; otherwise, proceed to the next pseudo-order in sequence.

f5

FCR s

If the sign bit (digit 0) of the fractional part of the P register is 0, transfer interpretive control to the right pseudo-order of storage location s ; otherwise, proceed to the next pseudo-order in sequence.

f6

CDB (s_1, s_2)

Starting at s_1 , convert in sequence the nonzero floating-point *numbers* from floating decimal to floating binary, adjust, and replace them. Proceed until either a stopper is encountered or all of the numbers between storage locations s_1 and s_2 , inclusive, have been converted. Leave those

word-pairs (including markers) unchanged whose bits to the right of the sign place (numerical portion) in the first word are zero.

f7

s/P

If the number in the P register is not zero, divide the floating-point number at storage location s by the contents of the P register and place the adjusted, rounded, high-order quotient in the P register; otherwise, halt on an error stop. (In the extended-precision mode, if the magnitude of the resulting quotient is greater than 2^{+4095} , a different error stop will occur.)

f8

-P

Place the negative of the contents of P in the P register.

f9

/P/

Place the absolute value of the contents of P in the P register.

fa

 $\sqrt{|P|}$

Form the square root of the absolute value of the floating-point number in the P register and place the adjusted result in the P register.

fb

XP

Clear the entire contents of the P register.

fc

Cos P

Compute the cosine of the contents of the P register, expressed in radians, and place the adjusted result in the P register. (In the extended-precision mode an error stop will occur if the magnitude of the argument is greater than 2^{76} .)

fd

/s/ Pd

Add the absolute value of the floating-point number at storage location s to the contents of the P register and place the adjusted result in the P register.

fe Arc Cos P
 If the magnitude of the contents of the P register is less than or equal to one, calculate the principal value (*i.e.*, the angle in the range 0 to π), in radians, of the angle whose cosine is equal to the contents of the P register, and place the adjusted result in the P register; otherwise, halt on an error stop.
 ff MON s

Enter the monitoring mode of operation; track and/or monitor according to the plan of the directory at s.

Compare the location of each pseudo-instruction encountered with every one of the "begin-monitoring print" locations specified in the left address portions of the directory (which begins at s and ends with the first word which is entirely zero). When an agreement occurs, cause an automatic printout to occur after each pseudo-order is interpreted (including the first) that could change the contents of the P register (with the exception of pseudo-order f0), and also compare the location of each pseudo-instruction encountered with the "end monitoring print" location specified in the right address portion of that directory word. When agreement with the latter occurs, discontinue the automatic printout *before* "monitoring" the current pseudo-instruction and proceed as before to test the location of each pseudo-instruction encountered against the table of "begin monitoring print" locations unless binary digit 40 of the directory word is a 1 — in which case, exit the monitoring mode of operation and proceed with the next pseudo-instruction in sequence. Note that if an "end monitoring print" location, specified in the right-address portion of a directory word, is the same as the "begin monitoring print" location given in the left of the directory word, then automatic printout will continue until the pseudo-order with that address is encountered for a second time (if ever). The interpretive mode may be exited and re-entered at will without disturbing the action of the monitor.

The following printout will occur during the monitor output phase if binary digit 16 of the pertinent directory word is a 0.

If binary digit 16 is a 1, the pseudo-instruction and its address will not be recorded. The former type of monitor output behavior is referred to as "tracking monitor" printout, whereas the latter is "simple monitor" printout.

Binary digit 20 of the directory word indicates whether monitoring printout is to begin with a left-hand or right-hand pseudo-instruction. If it is a 0, the left pseudo-instruction will be monitored; if it is a 1, the right pseudo-order will be monitored. Binary digit 44 governs the "end monitoring printout" instruction location in a similar manner.

If interpretive control is already in the monitoring mode, any subsequent ff pseudo-orders are interpreted as skips, *i.e.*, control proceeds to the next pseudo-order in sequence.

7. Entry and Return

The interpretive mode of control is entered via the machine-language mode, and a full word is required for standard entry. This word may be either

- (a.) s: sA, RO 0200, or
- (b.) s: sA, LO 0200.

The former word is slightly more efficient in time and, therefore, is recommended. Once the entry is executed, interpretive control will be sent to the left side of location s+1. The left-hand instruction at s+1 must be a pseudo-instruction, and all subsequent operations prior to return to the machine-language mode also must be specified in pseudo-language.

Return to the machine-language mode of computation is accomplished by any one of the following pseudo-instructions: d4, d5, df, e4, and e5. (A formal definition of each of these has already been presented in Section 6.b.)

NOTE: Re-entry into the interpretive mode need not be restricted to standard entry instructions, but also may be accomplished by more convenient techniques described in the definition of pseudo-order e4 (see Section 6.b).

Recorded via 90 order		Recorded via 33 order
CR, tab, XXXX, 2 spaces, XXXXXX, 2 spaces, ±XXXXXXXXXXXX, tab, ±XX, tab		
Address of Pseudo-order (in hexadecimal)	Interpreted instruction (in hexadecimal)	Contents of P (in floating decimal)

8. Output

a. *Printout*—See definitions of operations for pseudo-orders f0, f1, and f3 in Section 6b. Note however that hexadecimal format is recommended for all interpreted printout (*i.e.*, the format mode switch on the NAREC console should be set for hexadecimal output).

b. *Location of Results*—As was shown in Section 6 the results of most NIP operations are placed in the P register. The regular-precision P register is formed of two consecutive working locations as follows:

(1.) The sign and 44 bit fractional part of the number are contained in storage location 03b4.

(2.) The signed exponent of the number is contained in binary digits 0 to 12 of storage location 03b5. Binary digits 13 through 44 remain fixed (as zeros) throughout all NIP I operations.

c. *Register Contents*—Not applicable, except in the case of pseudo-order df; see Section 6.

d. *Changes to External Locations*—See Section 6.

9. Tape Label

2700—0000

10. Address Information

- a. First Address: 0000
- b. Last Address: 03ff
- c. Number of Words: 1024 (decimal)
400 (hexadecimal)

11. External Working Memory

Although no external working memory is required, programmers are urged to avoid using memory space between locations 0400 and 08ff in order to facilitate possible use of the NIP II program which allows for extended-precision computation.

12. Versatility

There are no changes which can readily be made to this subroutine.

13. Acknowledgments

Tape 2700 (NIP I) is a revision and extension of Tape 2000 and is a member of the sequence of NAREC floating-point interpretive routines, which now consists of Tapes 501, 1000, 1000A, 1500, 1500A, 2000, 2000A, 2700, and 4700 (NIP II). For the sake of completeness, Part I of this report includes a detailed description of every operation included in Tape 2700, even though many of them are due to Mr. Leo Davis, who wrote the original Tape 501 program, or to Dr. Benjamin Lepson, Head of the Numerical Analysis Branch, Applied Mathematics Staff, under whose guidance previous revisions and extensions (Tapes 1000, 1500, and 2000) of the original program were made.

PART 2—EXTENDED-PRECISION INTERPRETIVE ROUTINE (NIP II)

1. Name of Subroutine

NIP II—Extended Precision for the NIP I Floating-Point Interpretive Routine.

2. Class

Y2, Interpretive Routine, Multiple Precision Floating-Point.

3. Purpose

NIP II, like NIP I, is an interpretive subroutine that enables NAREC to perform automatic floating-point computations. NIP II, unlike NIP I, can handle as many as nineteen significant decimal digits. Its pseudo-order code, with only one exception (the f0 print instruction), is identical to the NIP I code. Indeed, every effort has been made to make NIP II completely compatible with any program previously written in the NIP I language for regular-precision computation.

Since essentially the same set of pseudo-orders is employed in both NIP I and NIP II, three distinct modes of computation may be utilized with equal facility by the programmer. First, entirely new programs may be written to take full advantage of the extended-precision capability. Second, if necessary, extended- and regular-precision computation may be mixed

within the same program. And last, any program previously written in the NIP I language, and not occupying the same memory space as the Extended-Precision NIP II system itself, may be run without modification in the extended-precision mode.

4. Language Used

A combination of actual NAREC machine language and NIP interpretive language.

5. Author

Edwin A. Stennett, Code 5425.

6. Formal Statement

a. *Conventions and Terminology*—The conventions and terminology applicable to NIP II are identical to those presented in Part 1, Section 6a.

b. *Definitions of Operations*—The definitions of NIP II operations are, with but two exceptions, identical to those presented in the NIP I description. Of course, references to the “contents of s” or the “P register” are now to be understood as references to the extended-precision “contents of s” or the extended-precision “P register.”

The first exception concerns the pseudo-orders:

- d6 — CDDB (s_1, s_2)
- f1 — DPPT i, j
- ff — MON s.

Although executable in the extended-precision mode, these pseudo-orders continue to operate as though the regular-precision mode were being employed. In other words, the printout from pseudo-order f1 and pseudo-order ff will contain, at most, eleven and ten decimal digits, respectively, and pseudo-order d6 will convert, at most, ten decimal digits.

The second exception is the f0 print instruction which must be completely redefined as follows:

f0 Pt mn
 $0 \leq n \leq 13$
 $m \leq n, m \leq f$

If m is specified ($m \neq 0$), round the number in the P register to n decimal places ($0 \leq n \leq 13$), treating $n = 0$ as a special case meaning $n = 13$; then convert and print, via 90 orders, the number as an n-digit mantissa in m groups, followed by a signed exponent containing one, two, three, or four digits, as necessary. The $n - 1$ digits to the right of the decimal point are divided equally (with the possible exception of a short last group).

This pseudo-order is contained in a single-address instruction of the following form:

0 XXX f0.
} }
m n

Execution is subject to the double restriction that (a) if n should ever exceed 13 or (b) if m should ever become greater than n, the program will halt on an error stop. In the example shown in Table 3 below, a table of printouts of the same quantity is given to clarify the behavior of this print operation. (For purposes of the example, it is assumed that the number $\pi/10$ is in the P register.)

TABLE 3
 Examples of Printout for
 Different Forms of the Extended
 Precision Print Instruction

Instruction	Result
0113f0	-3.141592653589793238 -1
0313f0	-3.141592 653589 793238 -1
0210f0	-3.14159265 3589793 -1
030ff0	-3.14159 26535 8979 -1

If m is not specified ($m = 0$), however, then this pseudo-order behaves in every way identically to the f0 pseudo-order of NIP I.

7. Entry and Return

Programmed entry into the extended-precision mode of computation is always made from the regular-precision mode. Either of the instructions

0470 e4

or

05d3 e4

accomplishes this entry, provided, of course, that tape 4700 is in its proper operating position; interpretive control is then sent to the pseudo-order immediately following the entry instruction. All floating-point pseudo-orders subsequent to the extended-precision entry instruction and prior to the return instruction are executed in the extended-precision mode, regardless of the number of times which the interpretive mode itself may be exited and re-entered.

To return to the NIP I mode of computation, one of the instructions

0471 e4

or

0658 e4

must be executed. And, in a manner similar to the entry sequence, interpretive control will again be sent to the pseudo-order immediately following the return instruction, after the mode change has been accomplished.

In order to process, in the extended-precision mode, programs which were previously written for the regular precision mode *without modifying* such programs, it is necessary only to initiate the order

LO 0472

from the console prior to executing the usual starting order of the program. The NAREC will then stop immediately at location

0473

and the C register will contain

eeee 82 04e0 10.

The usual starting order may then be given from the console, and all subsequent floating-point computation will be performed in the extended-precision mode.

WARNING: Although extended-precision pseudo-orders may operate on numbers constructed in regular-precision format, the opposite is not true. Regular-precision pseudo-orders will usually yield erroneous results when forced to operate on numbers which are constructed in extended-precision format. Therefore, care must be taken to convert all numbers in ex-

tended-precision format to regular-precision format before operating on them in the regular-precision mode, *i.e.*, binary digits 13 through 44 of s+1 (see Table 1, in Part 1, Section 6.a) must be made zero.

8. Output

a. *Printout*—See the definition of operation for pseudo-orders f0 and f1 in Part 2, Section 6, and in Part 1, Section 6b for f3. Note however, that a hexadecimal setting of the format mode switch is recommended for all interpreted print-out.

b. *Location of Results*—While in the extended-precision mode, the P register is constructed as follows:

(1.) The sign and first 44 bits of the fractional part of the number are contained in the arithmetic portion of the NAREC word at location 04b4.

(2.) The remaining 32 bits of the fractional part are contained in binary digits 13 to 44 of the NAREC word at location 04b5.

(3.) The signed exponent of the number is contained in binary digits 0 to 12 of the NAREC word at location 04b6.

c. *Register Contents*—Not applicable, except in the case of the df pseudo-order; see Part 1, Section 6.

d. *Changes to External Locations*—See Part 1, Section 6.6. Note also that NIP II may alter the contents of the regular-precision P register in an indeterminate manner. Therefore when switching from the regular- to the extended-precision mode of computation, care should be taken not to use the P register of NIP I as a storage location for intermediate results.

9. Tape Labels

4700 - 0400 (NIP II), and 2700 - 0000 (NIP I).

10. Address Information

a. Tape Number	2700	4700
b. First address	0000	0400
c. Last address	03 ff	08 ff
d. Number of Words	400	500
	(hexi)	(hexi)
	1024	1280
	(decimal)	(decimal)

11. External Working Memory

None.

12. Versatility

There are no changes which can readily be made to this subroutine.

13. Acknowledgments

The author of this program is indebted to Gerald A. Chayt and Elliod Dent for their helpful suggestions and willing counsel.

Appendix A

The Floating-Point System

INTRODUCTION

Much of the material contained in the following paragraphs may already be known to the reader—in fact, the wording may be familiar to him. The reason for this is that these paragraphs depend heavily upon the contents of an earlier publication (now out of print).^{*} The present authors have updated the applicable portions of that earlier report and included them here. With permission of the author, they have relied extensively on direct quotations and close paraphrases in an attempt to retain the high readability of the original document.

AUTOMATIC SCALING

The fixed-binary point design of the NAREC requires that all numbers used by the computer be less than 1 in absolute value. The computations which the computer is called upon to perform generally are not restricted to such numbers, so some modification of the equations used will have to be made, in order to provide that no numbers fall outside the range -1 to 1 . The conversion of a given set of equations to an equivalent set which the computer can accept is called scaling. Problem variables must be multiplied by appropriate scaling factors (usually integral powers of two or ten) before they are entered into the machine. For example, values of x and y satisfying

$$y = ax + b \quad (1)$$

will also satisfy

$$y \cdot 10^{-v} = (a \cdot 10^{-u})(x \cdot 10^{u-v}) + b \cdot 10^{-v}. \quad (2)$$

It is usually possible to select the u 's and v 's in such a way that the resulting equation is machineable. But the use of constant scaling factors throughout a computation sometimes leads to an unacceptable loss of significance in the results. The floating-point system embodied in the NAREC Interpretive Program (NIP) represents one technique for handling such situations. It makes use of a variable scaling factor which is associated with each number entering the computation, and thus permits a variable to go through a very wide range of values while retaining a maximum number of significant digits.

One way in which a variable scaling factor may be introduced is demonstrated by a convention sometimes used in hand computation. Given some decimal number, the decimal point is moved to a standard position and the resulting number is corrected by a factor 10^n , where n is the number of places the point has been moved and is taken to be positive if the point was moved to the left and negative if the point was moved to the right. For example, if the standard decimal point is to precede the most significant digit, the numbers given originally as

93257.164392 and 0.000052731944586

will appear as

.93257164392 $\cdot 10^5$ and .52731944586 $\cdot 10^{-4}$,

respectively.

^{*}L. E. Davis, "Floating-Point Coding for the NAREC," NRL Report 4579, Aug. 1955.

The same convention may be adopted when binary numbers are used, and this is essentially what is done in the NAREC floating-point system because the NAREC, as a binary machine, allows some simplifications in programming due to the agreement in number bases. The standard binary-point location has been selected to correspond with that of the NAREC word, or immediately to the right of the sign digit. The number of digits by which the point must be displaced to reach this position will be, with proper sign attached, the power of 2 by which the derived number must be multiplied. To state this in the form of an equation, the number x may be said to be expressed in floating-point form if it is written as

$$x = p \cdot 2^m, \quad (3)$$

where

$$\frac{1}{2} \leq |p| < 1 \quad (4)$$

and m is an integer. This representation exists uniquely for all real numbers, excepting zero. The case $x = 0$ is handled by setting $p = 0$, in violation of inequality (4). Thus, true zero differs from all other floating-point numbers in that it has no adjusted form and its exponent may take any integer value within the usual range; *i.e.*,

$$|m| < 2^{12}. \quad (5)$$

Such a floating-point number is called a "floating-point zero."

In the first floating-point routine written for the NAREC, the floating-point zero occasionally led to confusion in the addition operation, since adding this zero to a nonzero number sometimes produced a sum equal to zero instead of the nonzero number. The situation in which this occurred was the appearance of a much larger exponent with the zero than with the other number. The exponent of the zero was considered as an indication of its significance and the floating-point sum of zero and another number was given as much significance as permitted by the two numbers. Currently, in the NAREC Interpretive Program, for example, floating-

point zeros are distinguished and treated as special cases in those operations where they might cause trouble. As a result, this source of confusion, and perhaps others, is removed.

A possible source of error still remaining is in overreaching the magnitude bounds imposed upon floating-point numbers by the exponent representation. The largest positive number expressible in floating-point notation is $(1 - 2^{-44}) 2^{4095}$, or nearly 10^{1233} . The smallest nonzero positive number which can be represented is approximately 10^{-1233} . Operations which would produce numbers outside these limits will not be performed correctly because of register spills in exponent handling. However, the limits given above are more than adequate for most practical problems.

In order to request one floating-point operation, the programmer must write one NIP instruction. In order to carry out one floating-point instruction, the interpretive system must execute several NAREC orders. The steps that have been built into the NAREC Interpretive Program to perform the elementary operations of adjustment, multiplication, division, and addition in regular precision will be outlined in the following sections of this appendix, although the actual orders which would be used to carry them out will not be given here. The operations are discussed in the order of their increasing difficulty. Readers requiring more specialized information concerning elementary operations in extended precision will find this material in Appendix C.

ADJUSTMENT

The principal reason for keeping floating-point numbers in adjusted form is to prevent loss of significance in multiplication. For this reason, and for convenience in some of the other operations, the convention has been adopted in the floating-point system of assuming all numbers involved to be in adjusted form at the beginning of each operation and of leaving the results in this form at the completion of the operation. Since some operations would produce unadjusted numbers and since it is often convenient to introduce numbers in unadjusted form, it is necessary to have a routine which will convert

any unadjusted floating-point number, excepting zero, to standard floating-point form.

The problem here is as follows: Given a number p_1 such that $0 < |p_1| < 1$ and an integer m_1 which together represent a number y through the equation

$$y = p_1 \cdot 2^{m_1}, \quad (6)$$

find the numbers p and m such that

$$y = p \cdot 2^m,$$

where $1/2 \leq |p| < 1$ and m is an integer. If i is any number, it follows from Eq. (6) that

$$y = (p_1 \cdot 2^i) \cdot 2^{m_1-i}. \quad (7)$$

It can be shown that there is just one integer k for which

$$\frac{1}{2} \leq |p_1 \cdot 2^k| < 1,$$

and furthermore, $k \geq 0$ since $|p_1| < 1$.

If i is set equal to this value of k in Eq. (7), then clearly

$$p = p_1 \cdot 2^k$$

and

$$m = m_1 - k$$

are the values sought.

To find k , the absolute value of p_1 is placed in the A register and shifted left just as many times as needed to make the A register read greater than or equal to $1/2$, or, equivalently, until the first nonzero digit of $|p_1|$ has been shifted into the digit-1 position. These shifts are counted, and the count is the number k since a left shift is the same as multiplication by two. After the shifting process, the A register contains $|p| = |p_1| \cdot 2^k$, and the count number may be subtracted from m_1 to get the value of m . From this point it is only a matter of placing the values p , $m \cdot 2^{-12}$ in the desired result locations in storage.

MULTIPLICATION

Multiplication is the least complicated arithmetic operation in floating point. Given two

floating-point numbers $y_1 = p_1 \cdot 2^{m_1}$ and $y_2 = p_2 \cdot 2^{m_2}$, the product y_3 is

$$y_3 = y_1 y_2 = p_1 p_2 \cdot 2^{m_1+m_2}.$$

The product $p_3' = p_1 p_2$ and the sum $m_3' \cdot 2^{-12} = (m_1 + m_2) \cdot 2^{-12}$ are formed. Since p_3' may be less than $1/2$, the adjustment routine is used on p_3', m_3' to get p_3, m_3 , the standard floating-point representation of y_3 .

DIVISION

Finding the quotient of two numbers is simplified by the adjustment convention. If y_1 and y_2 are given as above, their quotient is

$$y_3 = p_3 \cdot 2^{m_3} = \frac{y_1}{y_2} = \frac{p_1}{p_2} \cdot 2^{m_1-m_2}.$$

Since y_1 and y_2 are assumed to be floating-point numbers,

$$\frac{1}{2} \leq |p_1| < 1$$

and

$$\frac{1}{2} \leq |p_2| < 1.$$

Taking the two extreme possibilities, these relations give

$$\frac{1}{2} < \frac{|p_1|}{|p_2|} < 2.$$

Then either

$$\frac{1}{2} < \frac{|p_1|}{|p_2|} < 1$$

or

$$\frac{1}{2} \leq \frac{|p_1|}{|2p_2|} < 1.$$

Division thus breaks down into two cases: first, if $|p_1| < |p_2|$, then

$$p_3 = \frac{p_1}{p_2}, \text{ and } m_3 = m_1 - m_2;$$

second, if $|p_1| \geq |p_2|$, then

$$p_3 = \frac{p_1 \cdot 2^{-1}}{p_2}, \text{ and } m_3 = m_1 - m_2 + 1.$$

ADDITION

The addition operation is somewhat more complicated than the ones previously discussed. In the first place, the handling of y_1 and y_2 depends upon whether $m_1 > m_2$ or $m_1 < m_2$, while $m_1 = m_2$ may be considered with either one of these cases. In the second place, the adjustment of the sum will depend on whether y_1 and y_2 had the same or different signs. If they had the same signs it will depend upon whether an overflow occurred in the addition, that is, whether the absolute value of the numbers summed in the A register exceeded unity.

Assume y_1 and y_2 are of the same form as given previously and that $m_1 \neq m_2$ so that p_1 and p_2 may not be added directly. Now y_3 , the sum of y_1 and y_2 , may be written in either of the following two ways:

$$y_3 = (p_1 + p_2 \cdot 2^{m_2 - m_1}) \cdot 2^{m_1} \quad (8)$$

or

$$y_3 = (p_1 \cdot 2^{m_1 - m_2} + p_2) \cdot 2^{m_2}. \quad (9)$$

Both p_1 and p_2 are in absolute value not less than 2^{-1} , so they cannot be multiplied by positive powers of 2 without the results overflowing. However, either $m_2 - m_1$ or $m_1 - m_2$ will be nonpositive, and the equation in which this is true is the equation which will be employed. In other words, if $m_1 > m_2$, Eq. (8) will be used; if $m_1 < m_2$, Eq. (9) will be used; and if $m_1 = m_2$, either equation may be used.

Assume that $m_1 > m_2$ and, in fact, $m_1 - m_2 = k > 0$. Then Eq. (8) becomes

$$y_3 = (p_1 + p_2 \cdot 2^{-k}) \cdot 2^{m_1} = p'_3 \cdot 2^{m'_3},$$

where

$$p'_3 = p_1 + p_2 \cdot 2^{-k}$$

and

$$m'_3 = m_1.$$

Here two cases are possible. In the first case, y_1 and y_2 , and hence p_1 and p_2 , are of opposite signs so the absolute value of $p_1 + p_2 \cdot 2^{-k}$ will not be greater than the absolute value of p_1 and no spillover can occur. This sum may be smaller than 2^{-1} , so p'_3, m'_3 is sent to the adjustment routine to get the floating-point pair p_3, m_3 . In the second case, y_1 and y_2 are of the same sign. If $|p_1 + p_2 \cdot 2^{-k}|$ is less than unity, then

$$p_3 = p_1 + p_2 \cdot 2^{-k},$$

and

$$m_3 = m_1.$$

If $|p_1 + p_2 \cdot 2^{-k}|$ is not less than unity, then

$$p_3 = \frac{p_1 + p_2 \cdot 2^{-k}}{2}$$

and

$$m_3 = m_1 + 1$$

provide the floating-point sum. No further adjustment is needed for either type of result in this second case.

If $m_2 > m_1$, the situation is the same as above with the subscripts interchanged.

Appendix B

An Example of a Program Written in Extended Precision

The figures contained in this Appendix are reproductions of the programming sheets and other materials instrumental in a recent NAREC computation by Irene G. Fishman and William E. Conrad using NIP II. The purpose of this undertaking was to tabulate accurately, as nineteen-decimal-digit numbers, the seven quantities

$$n, p, \frac{1}{x} - \frac{1}{x(x+n)}, \frac{-1}{x(x+n)}, x, x + n,$$

$$\text{and } (x+n)x^{n-1},$$

which all pertain to a test matrix $A = xI + J$ which is useful in checking matrix inversion programs, where I is the identity matrix and J is a matrix of order n whose elements are all unity.* These quantities are, respectively, the order of A , a parameter, any of the equal diagonal

elements of A inverse, any of the equal off-diagonal elements of A inverse, the smallest eigenvalue of A , the largest eigenvalue of A , and the determinant of A .

The following parameter ranges were specified:

$n = 10, 20, 30, 40, 50, 51, 52, \dots, 99, 100, 200, 300, \dots, 3000$, and $p = 1, 2, \dots, w$, where w is the largest integer to satisfy the inequality $w \leq 100 \ln 2 / \ln n$, and x is given by the relation: $x = 1/n^p$.

NOTE: It should be observed that the value of x^{n-1} falls outside of the floating-point exponent range for certain combinations of the parameters; for example, this occurs in the case $n = 60$ and $p = 12$. Therefore, to avoid unwanted error stops, a detour around the calculation of $(x+n)x^{n-1}$ was made, and printout of the last column entry was suppressed whenever

$$w > \frac{4095 \ln 2}{(n-1) \ln n}$$

*M. Newman, and J. Todd, "The Evaluation of Matrix Inversion Programs," *J. Soc. Indust. Appl. Math.* 6(No. 4):469 (1958).

NAR OPERATOR INSTRUCTION SHEET (12/6/61)

Date

3-2-64

RCC Problem Number

407

NRL Account Number _____

Problem Title

Matrix Constants

Programmer

Fishman

Pass _____

Telephone

585

1. Version of NAR to be used: NAR I NAR IA NAR IB
2. "Back of Memory": ffff First Address in Object Program: 3000
3. "Pre-assign" Tapes (LO 0f77) (1) _____ (2) _____ (3) _____
4. Source Program Tapes:
- | | | | |
|-------------------|---------------|------------|---------|
| (1) <u>55073a</u> | CONTINUE P.B. | (6) _____ | LO 0fd2 |
| (2) <u>53963</u> | LO 0fd2 | (7) _____ | LO 0fd2 |
| (3) _____ | LO 0fd2 | (8) _____ | LO 0fd2 |
| (4) _____ | LO 0fd2 | (9) _____ | LO 0fd2 |
| (5) _____ | LO 0fd2 | (10) _____ | LO 0fd2 |
5. Push CONTINUE P.B. Stops on STOP #3. Any printout indicates an error.
6. Printouts Desired if any assembly errors: ALL NONE SOME
- If SOME: Object Prog. (first time) LO 03a5 Obj. Prog. (2nd time) LO 03ab
 List of Numb. Instrs.etc. LO 03b1 Checking Output LO 18e0
7. Printouts Desired if no assembly errors: ALL NONE SOME
- If SOME: Object Prog. (first time) LO 03a5 Obj. Prog. (2nd time) LO 03ab
 List of Numb. Instrs.etc. LO 03b1 Checking Output LO 18e0
8. Object Program to be placed in working position?
- YES YES if no errors NO
- Push CONTINUE P.B. if in STOP #8; Otherwise LO 3fd9
9. Special Instructions:

Fig. B1 - NAR Operator instruction sheet used in assembling the Matrix Constants program

MACHINE LANGUAGE OPERATOR INSTRUCTION SHEET (2/4/64)

Date 3-2-64

RCC Problem Number 407

Estimated Computation Time 3 1/2 hrs

Problem Title Matrix Constants

Actual Computation Time _____

Run _____

Programmer Fishman Tel.Ext. 585

OUTPUT FORMAT: A: Hexi, Deci; _____ col., Inf.Col. B: Hexi, Deci; _____ col., Inf.Col.
 PRINTER, PUNCH, BOTH PRINTER, PUNCH, BOTH

INPUT TAPE INFORMATION		Locations		Check Sum (if known)			
Tape Label		From	Thru				
<u>2700-0000</u>		<u>0000</u>	<u>03ff</u>	<u>d155</u>	<u>02</u>	<u>7d.0c</u>	<u>3B</u>
<u>4700-0400</u>		<u>0400</u>	<u>08ff</u>	<u>88B7</u>	<u>73</u>	<u>1c.33</u>	<u>77</u>
<u>3939B-0B00</u>		<u>0B00</u>	<u>0cft</u>	<u>d70f</u>	<u>28</u>	<u>f783</u>	<u>9d</u>

SPECIAL INPUT INSTRUCTIONS, MANUAL CHANGES, AND STARTING ORDER(S)

L0 3000

PRINT-OUTS

Estimated Time to First Print-out _____ Total Number of Output Lines _____

Time Per Line _____ Location of Final Stop Order 3051

RESTART ORDERS

Read in Tapes _____

Manual Changes _____

Starting Order L0 RO _____

DUMP: Output Format A B Other _____

Read in Tapes _____

Manual Changes _____

Starting Order L0 RO _____

SPECIAL INSTRUCTIONS, ADDITIONAL INFORMATION, ETC. None See Reverse Side

Fig. B2 - Machine-language operator instruction sheet used in running the Matrix Constants program

NAR PROGRAMMING SHEET		PRNG-NRL-36-544 (R-42)	DATE	TITLE	PROB. NO.	TAPE LABEL	PAGE NO.
CODER		Fickman	2/14/64	Matrix Constants		55073a	1
LINE NO.	SYMB. ADDR.	OPERATION	REMARKS	LINE NO.	SYMB. ADDR.	OPERATION	REMARKS
1	strt	xa		6		a	nfx
2		a	nfx	7	m2	m2	a
3		a	pfz	8		ro	512
4		hl	a	9		n	p
5		a	repi	30		find	pd
6		tenx	a	1		p	n
7		a	fxnd	2		in	p
8	enfi	enfi	a	3		p	logn
9	ro	ro	512	4		cont	p
10	05d3e4			5		p	n
1	ten	p		6		n	p
2	p	flnd		7		-one	pd
3	xp	xp		8		p	umil
4	p	p	pflo	9		mp	logn
5	p	p	n	0		cord	p
6	pc	pc	37	41		p	wcal
7	pc	pc	37	2		n1	m8
8	pc	pc	37	3	m8	pfz	a
9	repi	rep	5	4		onex	ad
20	n1	n1	fxl	5		a	pfz
1	bK	skip		6	m9	m9	a
2	skip	skip		7		ro	512
3	m1	nr	m1	8		pflo	p
4	nfx	nfx	a	9		one	pd
5	fxnd	fxnd	ad	50		p	pflo

Fig. B3 - Handwritten source program

NAR PROGRAMMING SHEET		PRNC - NRL-36-544 (8-42)	DATE	TITLE	PROB. NO.	TAPE LABEL	PAGE NO.
CODER		Fishman	2/14/64	Matrix Constants		55073a	2
LINE NO.	SYMB. ADDR.	OPERATION	REMARKS	LINE NO.	SYMB. ADDR.	OPERATION	REMARKS
1		comp	w	6		p	p13
2		n1	comp	7	co5	n1	co6
3		n1	comp	8	co6	nfx	a
4		n1	back	9		000441	
5	comp	ptx	a	80	co7	co7	a
6		co328		1		10	bdic
7		nfx	a	2		spac	w
8		-over	ad	3		pcu	
9		co428		4		ptx	a
60	co2	co2	a	5		000241	
1		ro	512	6	co9	co9	a
2		one	p	7		10	bdic
3		p /	n	8	co11	co11	a
4	co3	pow	o	9		ro	512
5		p	x	90		pc	4
6		n	pd	1		pc	4
7		p	show	2		p13	p
8		mp	x	3		0113fo	
9		p	denm	4		pc	4
70		-one	p	5		pc	4
1		p /	denm	6		pc	4
2		p	pr4	7		pr4	p
3		one	p	8		0113fo	
4		p /	x	9		pc	4
5		pr4	pd	100		pc	4

Fig. B3 (Continued) - Handwritten source program

NAR PROGRAMMING SHEET		PRIC - NRL-36-54 (B-42)	DATE	TITLE	PROB. NO.	TAPE LABEL	PAGE NO.
CODER		Fishman	2/14/64	Matrix Constants		55073a	3
LINE NO.	SYMB. ADDR.	OPERATION	REMARKS	LINE NO.	SYMB. ADDR.	OPERATION	REMARKS
1		pc		6	bkl	bkl	a
2		r		7		ro	512
3		0113fo		8		2p	
4		pc		9		p	pf10
5		pc		13		pc	37
6		pc		1		pc	37
7		show		2		pc	37
8		0113fo		3		fr	bk
9		pc		4	fxl	h2	a
110		pc		5		a	repi
1		pc		6		one x	a
2		pf10		7		a	fxnd
3		cm p		8	tl	tl	a
4		f1		9		ro	512
5		f1		14		one	
6		f1		1		p	fxnd
7	c444	r		2		f1	repi
8	c04	pow		3	fx2	h3	a
9		mp		4		a	repi
12		p		5		hex x	a
1		0113fo		6		a	fxnd
2	c0ut	pc		7	tz	tz	a
3		NI		8		ro	512
4	back	xa		9		hex x	p
5		a		15		p	fxnd

Fig. B3 (Continued) - Handwritten source program

NAR PROGRAMMING SHEET		PRIC - NRL - 56-54 (8-52)		DATE		TITLE		PROB. NO.		TAPE LABEL		PAGE NO.	
CODER		Fishman		2/14/64		Matrix Constants				55073a		4	
LINE NO.	SYMB. ADDR.	OPERATION	REMARKS	LINE NO.	SYMB. ADDR.	OPERATION	REMARKS	LINE NO.	SYMB. ADDR.	OPERATION	REMARKS	LINE NO.	SYMB. ADDR.
1				6				6					
2	fx3	fx1	repi	7	con1	0		7	08aa12				
3	tenx	stop		8				8	2B99c6				
4		000000		9				9	000706				
5	onex	000000		18°				18°	31e2B0				
6		000001		1	one	0		1	080000				
7	hurf	0		2				2	0				
8		000064		3				3	000100				
9	urf	0		4				4	0				
16°		0		5	con2	0		5	0B1670				
1	pf	0		6				6	006525				
2		0		7				7	000e25				
3	hl	rep	5	8				8	0B4286				
4		ur	fx1	9	hurf	0		9	0c8000				
5	fxnd	0		19°				19°	0				
6		0		1				1	000700				
7	h2	rep	50	2				2	0				
8		ur	fx2	3	find	0		3	0				
9	h3	rep	19	4				4	0				
17°		ur	fx3	5				5	0				
1	spac	0		6				6	0				
2		000004		7	pflo	0		7	0				
3	ten	0a.0000		8				8	0				
4		0		9				9	0				
5		000400		20°				20°	0				

Fig. B3 (Continued) - Handwritten source program

NAR PROGRAMMING SHEET - PRINC - NPL - 36 - 541 (8-42)		DATE	TITLE	PROB. NO.	TAPE LABEL	PAGE NO.	
CODER <i>Fickman</i>		<i>2/14/64</i>	<i>Matrix Constants</i>		<i>55073a</i>	<i>5</i>	
LINE NO.	SYMB. ADDR.	OPERATION	REMARKS	LINE NO.	SYMB. ADDR.	OPERATION	REMARKS
<i>20</i>	<i>n</i>	<i>0</i>		<i>6</i>		<i>0</i>	
<i>2</i>		<i>0</i>		<i>7</i>		<i>0</i>	
<i>3</i>		<i>0</i>		<i>8</i>		<i>0</i>	
<i>4</i>		<i>0</i>		<i>9</i>	<i>pt 4</i>	<i>0</i>	
<i>5</i>	<i>legn</i>	<i>0</i>		<i>230</i>		<i>0</i>	
<i>6</i>		<i>0</i>		<i>1</i>		<i>0</i>	
<i>7</i>		<i>0</i>		<i>2</i>		<i>0</i>	
<i>8</i>		<i>0</i>		<i>3</i>	<i>pt 3</i>	<i>0</i>	
<i>9</i>	<i>mil</i>	<i>0</i>		<i>4</i>		<i>0</i>	
<i>210</i>		<i>0</i>		<i>5</i>		<i>0</i>	
<i>1</i>		<i>0</i>		<i>6</i>		<i>0</i>	
<i>2</i>		<i>0</i>		<i>7</i>	<i>pt 7</i>	<i>0</i>	
<i>3</i>	<i>wea 1</i>	<i>0</i>		<i>8</i>		<i>0</i>	
<i>4</i>		<i>0</i>		<i>9</i>		<i>0</i>	
<i>5</i>		<i>0</i>		<i>240</i>		<i>0</i>	
<i>6</i>		<i>0</i>		<i>1</i>	<i>w</i>	<i>0</i>	
<i>7</i>	<i>r</i>	<i>0</i>		<i>2</i>		<i>0</i>	
<i>8</i>		<i>0</i>		<i>3</i>		<i>0</i>	
<i>9</i>		<i>0</i>		<i>4</i>		<i>0</i>	
<i>220</i>		<i>0</i>		<i>5</i>		<i>0</i>	
<i>1</i>	<i>show</i>	<i>0</i>		<i>6</i>			
<i>2</i>		<i>0</i>		<i>7</i>			
<i>3</i>		<i>0</i>		<i>8</i>			
<i>4</i>		<i>0</i>		<i>9</i>			
<i>5</i>	<i>del m</i>	<i>0</i>		<i>0</i>			

Fig. B3 (Continued) - Handwritten source program

173 ten 0a0000
174 0
175 000400
176 0
177 con1 q8aa12
178 02b59c6
179 900705
180 31e2b0
181 one 080000
182 0
183 000100
184 0
185 0b1670
186 00652a
187 000c22
188 ch4282
189 hunf 0c8000
190 0
191 000700
192 0
193 f1nd 0
194 0
195 0
196 0
197 pf'o 0
198 0
199 0
200 c
201 n 0
202 0
203 0
204 0
205 1ogn 0
206 0
207 0
208 0
209 nmi1 0
210 0
211 0
212 0
213 wca1 0
214 0
215 0
216 0
217 x 0
218 0
219 0
220 show 0
221 0
222 0
223 0
224 0
225 denn 0
226 0
227 0
228 0
229 pr4 0
230 0
231 0
232 pr3 0
233 0
234 0
235 0
236 pr7 0
237 0
238 0
239 0
240 v 0
241 0
242 0
243 0
244 0

XXXXXXXXXX

Fig. B4 - Typewritten source program

nnnn

1	strt	ks
2		a nfx
3		a pfx
4		n' a
5		tepi
6		a'xnd
7		enf)
8		enf) a
9		ro 512
10		0543e4
11		ten p
12		p f'nd
13		xp
14		p pf'o
15		p n
16		pc 37
17		pc 37
18		pc 37
19	repi	rep 5
20	bk	n' ix)
21		skip
22		skp
23	m1	skp
24		nfx a
25		ixnd ad
26		a nfx
27	m2	m2 a
28		ro 512
29		n p
30		f'nd pd
31		p n
32		n p
33		p'ogn
34		com'/ p
35		p w
36		n p
37		-one pd
38		p mml
39		mp'ogn
40		com2/ p
41		n' m8
42		n' m8
43	m8	nfx a
44		onex ad
45		a pfx
46	m9	m9 a
47		ro 512
48		pf'o p
49		one pd
50		p pf'o
51		cmp w
52		n' comp
53		n' comp
54		n' back
55	comp	comp
56		pfx a
57		co3,28
58		nfx a
59		co3,28
60	co2	co2 a
61		ro 512
62		one p
63	co3	p/n
64		pow 0
65		p x
66		n pd
67		p show
68		mp x
69		p denm
70		-one p
71		p/ denm
72		p pr4
73		one p
74		p/ x
75		pr4
76	co5	pr4
77	co6	n' fac6
78		nfx a
79	co7	000447
80		co7 a
81		fo bdic
82		spac u
83		pcu

Appendix C

Block Diagrams of the Elementary Extended-Precision Operations of Addition, Complementation, Multiplication, and Division

To some extent the techniques employed within NIP II for basic arithmetic are dictated by the NAREC's hardware characteristics. On the other hand, once action blocks for the elementary extended-precision operations are programmed, techniques for the extended-precision functions may be developed with little or no thought given to the computer for which they were intended. Only the basic arithmetic techniques are discussed in this appendix since they are peculiar to the extended-precision subroutine and all other NIP II arithmetic operations and functions are based on them.

In order to better understand the four block diagrams, Figs. C1 through C4, which show the steps required for each of the elementary operations, the following ground rules should be noted:

1. Each extended-precision, floating-point binary number $x = (M, q)$, where M is a signed 76-bit fractional part and q is a signed 12-bit exponent, *i.e.*, $x = m \cdot 2^q$, is normally stored in a word-pair $(s, s+1)$ (see Word Form A).

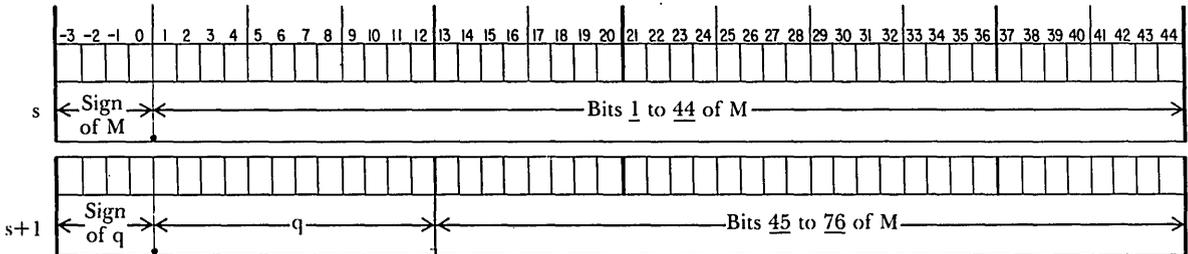
2. For convenience of handling within the NIP II subroutine, such word-pairs are separated and stored in sets of three consecutive working locations in memory.

a. For example, the extended-precision P register is constructed of working locations, referred to as ϕ_1 , ϕ_2 , and ϕ_3 (see Word Form B).

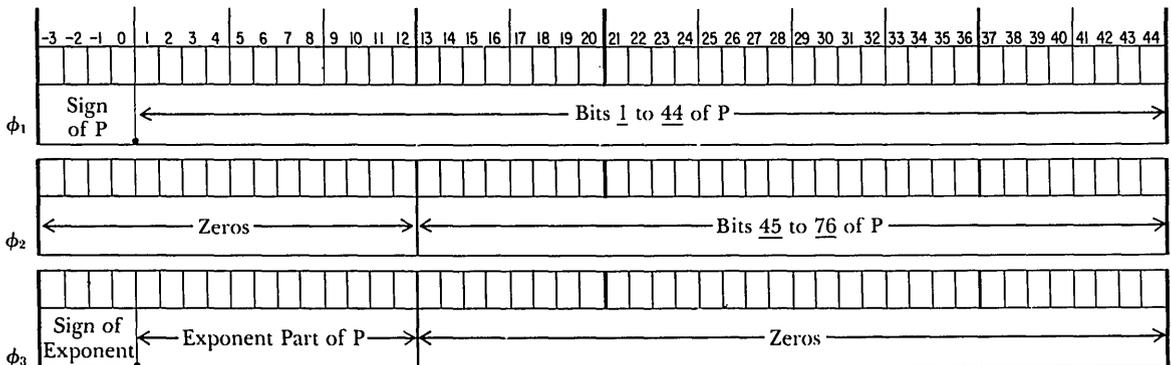
b. A similar set of working locations, referred to as μ_1 , μ_2 , and μ_3 , often contains the argument located at address s in three-part form.

Figure C1* shows the steps required to form a sum in extended precision. Note that the expression "justify ϕ with respect to μ " means that ϕ_1 and ϕ_2 must be shifted right (incrementing ϕ_3 by unity for each binary shift) until $\phi_3 = \mu_3$. The term "carry" in this block diagram (and in the succeeding diagrams) has its usual arithmetic meaning.

*It is emphasized that the block diagrams are intended as outlines only and are not to be considered complete.



Word Form A



Word Form B

e9

s Pd

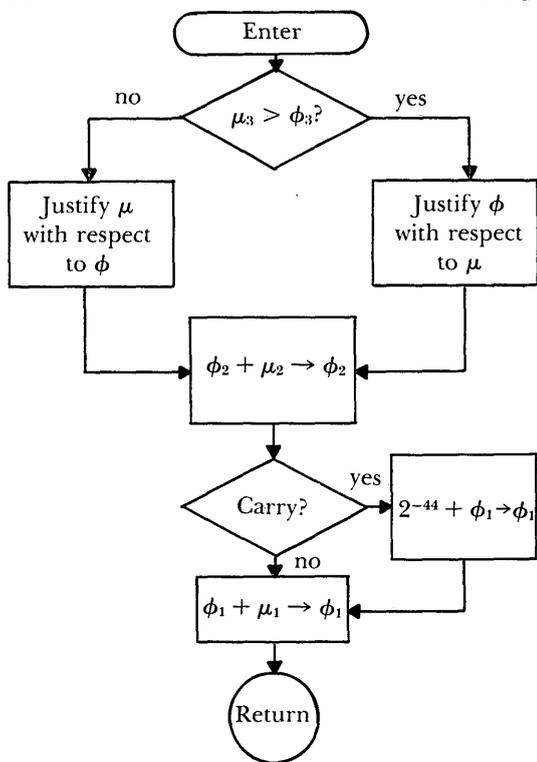


Fig. C1 - Steps required to form a sum in extended precision (NIP II subroutine)

f8

-P

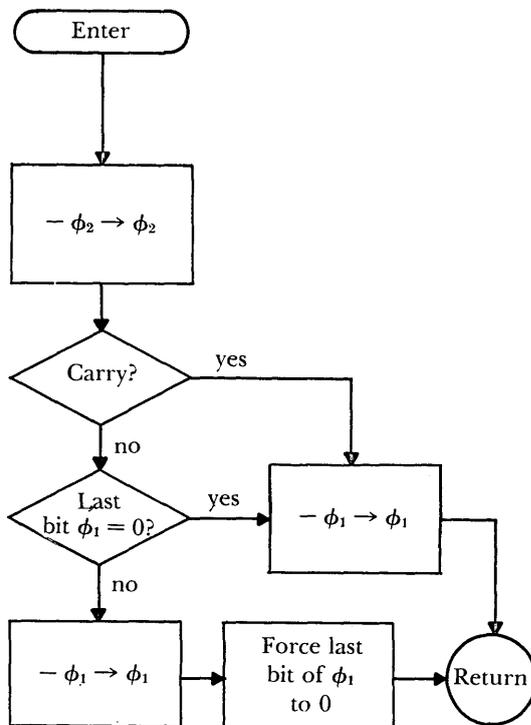


Fig. C2 - Steps required to form a two's complement in extended precision (NIP II subroutine)

Figure C2 shows the steps required to form a two's complement in extended precision. Whenever a NAREC word is negated (via the regular machine order $-s$ A), the two's complement is formed by adding a one to binary digit 44 of the inversion, which is obtained by replacing each binary 0 in the original number by a 1, and vice versa. Clearly, the two's complement of an extended-precision number arising in NIP II operation should be formed by adding a one to the 76th binary place of the inversion of the original number instead (this is accomplished in the block labeled $-\phi_2 \rightarrow \phi_2$). By combining a complementation (of the subtrahend) with an addition, the elementary operation of subtraction is obtained.

Figure C3 shows the steps required to form a product in extended precision. This is accomplished by means of the following variation of the general formula for the exact multiplication of two binomials: $(a+b)(c+d) \approx ac + (ad+bc)$.

Note also that use is made of the low-order product resulting from the regular NAREC multiply order m s.

Finally, Fig. C4 shows the steps required to form a quotient in extended precision. The block diagram depicts the Newton-Raphson method for approximating the reciprocal of a number N , in accordance with the iteration formula

$$X_{i+1} = X_i (2 - X_i N).$$

Starting initially with the value X_0 as a first approximation to $1/N$, the process yields a precision in the result known to double with each iteration. Because the first approximation to $1/s$ is determined to 44 binary places by means of a NIP I division, the iteration formula needs to be applied only once and the result multiplied immediately by the contents of the P register in order to arrive at the required quotient P/s .

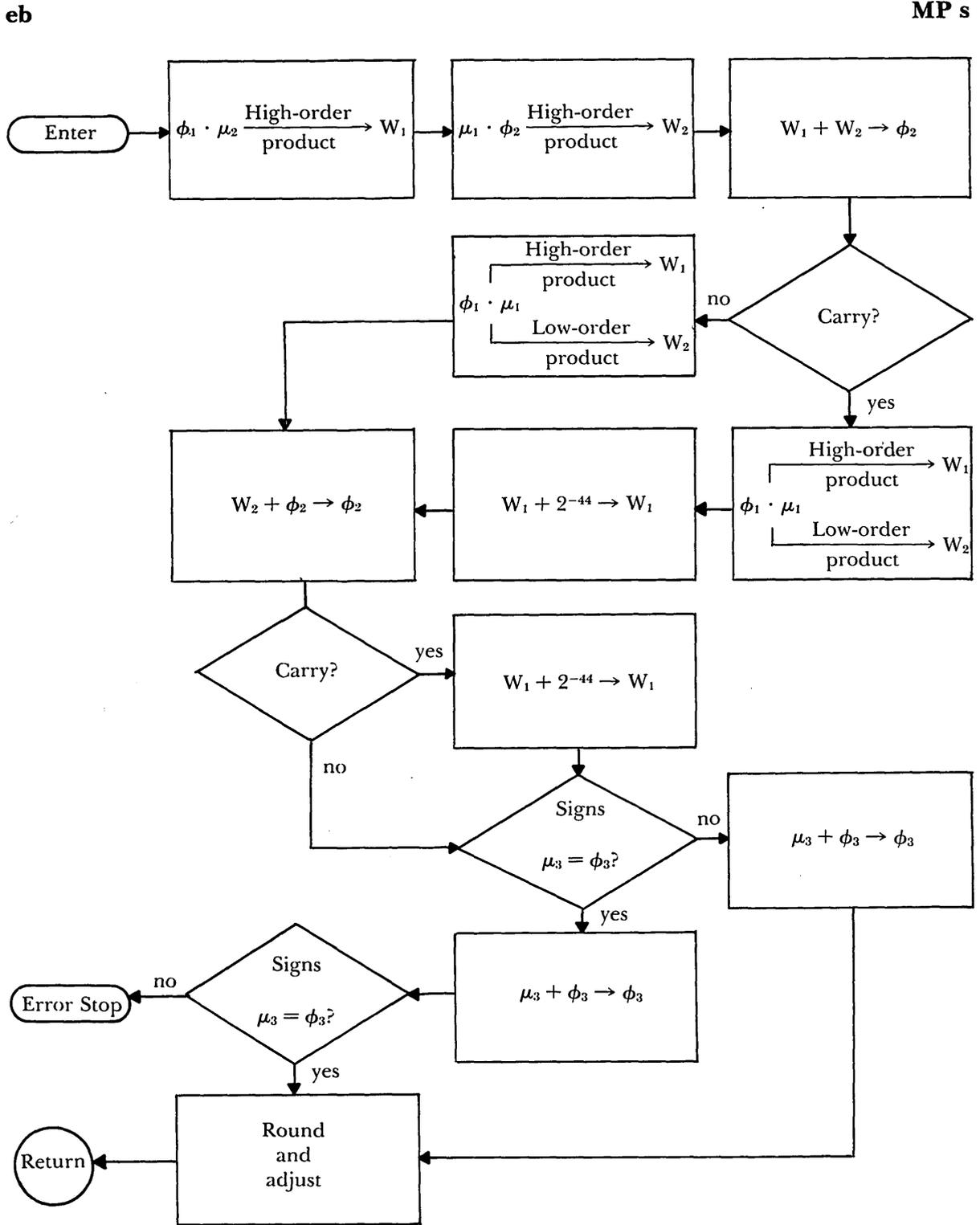


Fig. C3 - Steps required to form a product in extended precision (NIP II subroutine)

ec

P/s

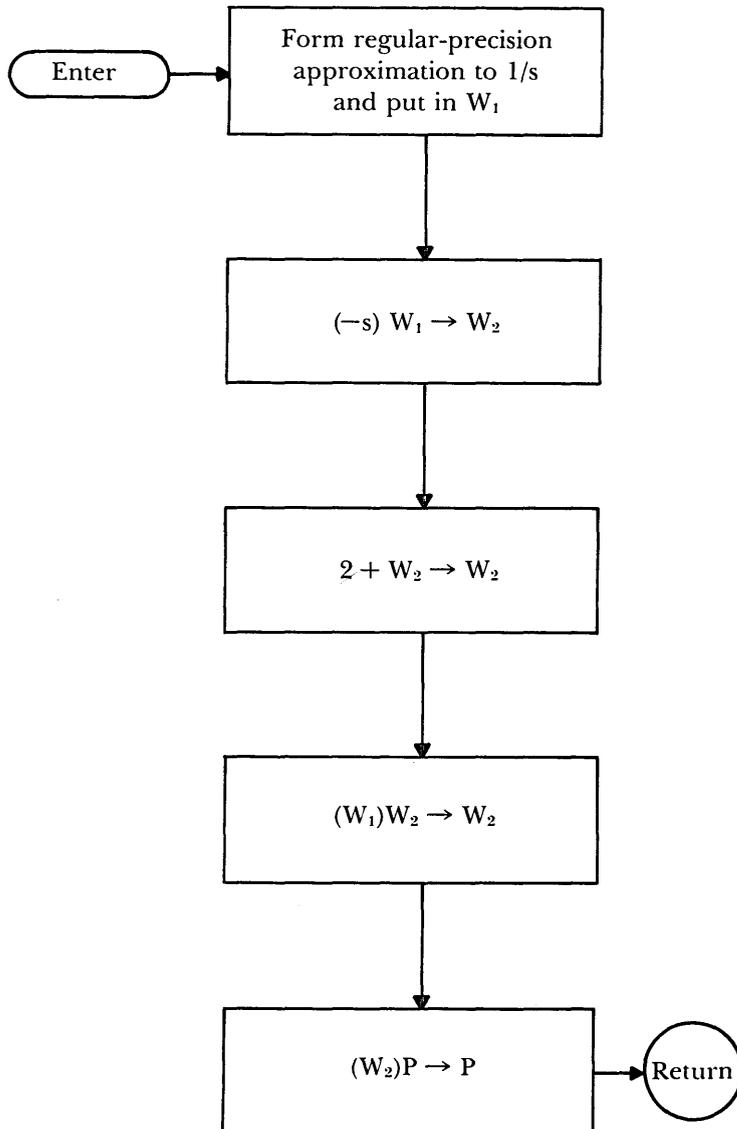


Fig. C4 - Steps required to form a quotient in extended precision (NIP II subroutine)

Appendix D

Difference Plots for the Elementary Functions: Square Root, Cosine, Arc Sine, Logarithm, and Exponential

Figures D1 through D5 are semilogarithmic graphs comparing NIP I and NIP II evaluations of the elementary functions, considering the latter as a reference. The functions included are square root (Fig. D1), cosine (Fig. D2), arc sine (Fig. D3), logarithm (Fig. D4), and exponential (Fig. D5). The term "argument" refers to the value of x

which is carried in the P register; the term "magnitude of difference" stands for the expression $|f_{II}(x) - f_I(x)|$, where the letter f symbolizes the function evaluated, and the subscripts I and II denote the regular-precision and extended-precision modes, respectively.

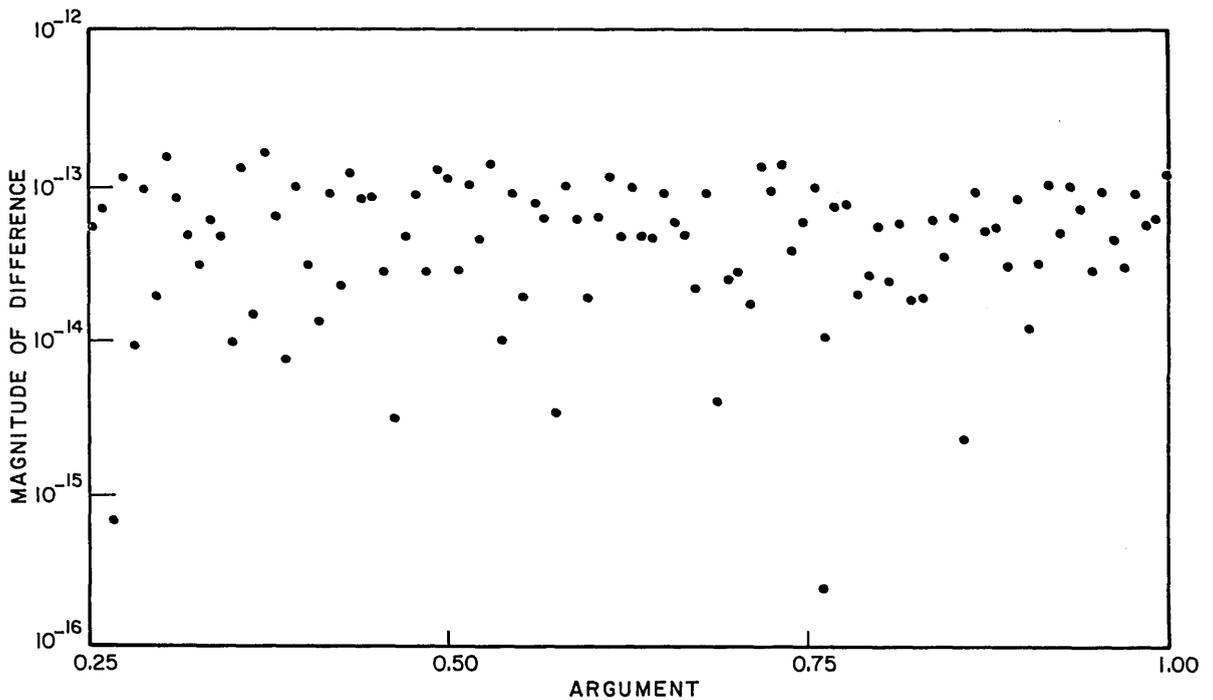


Fig. D1 - Plot of $|(\sqrt{x})_{II} - (\sqrt{x})_I|$ as a function of x

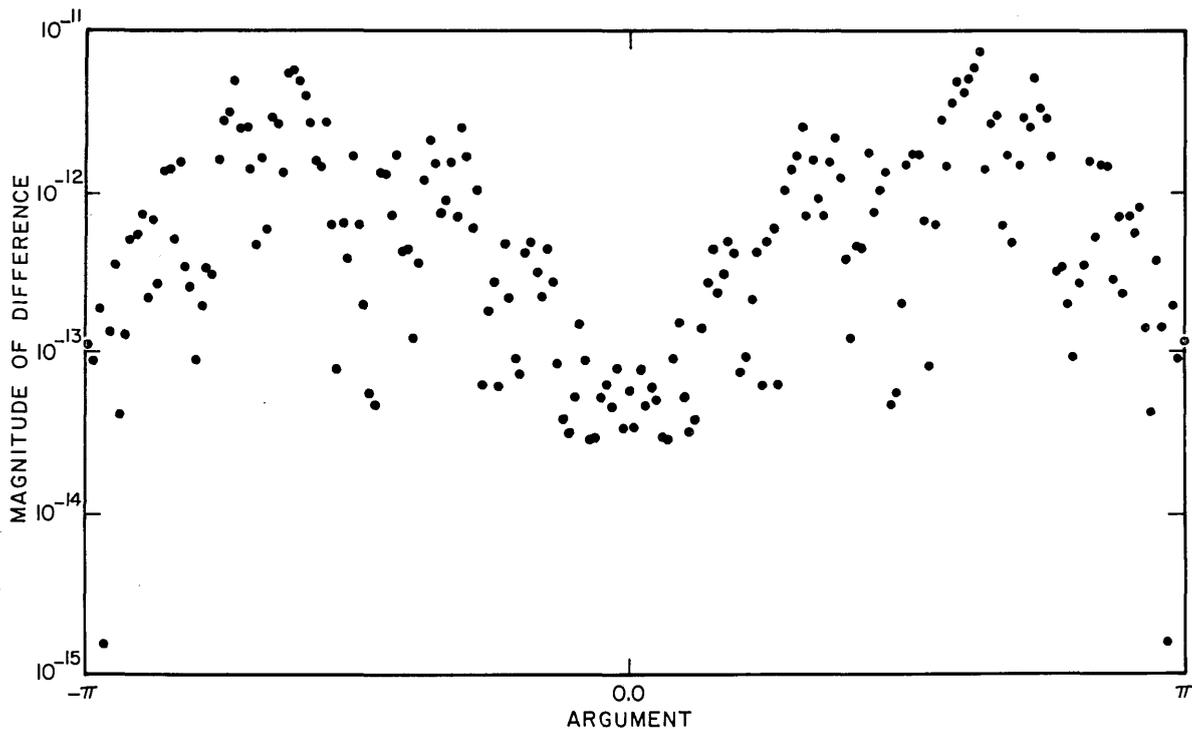


Fig. D2 - Plot of $|\cos_{II}(x) - \cos_I(x)|$ as a function of x

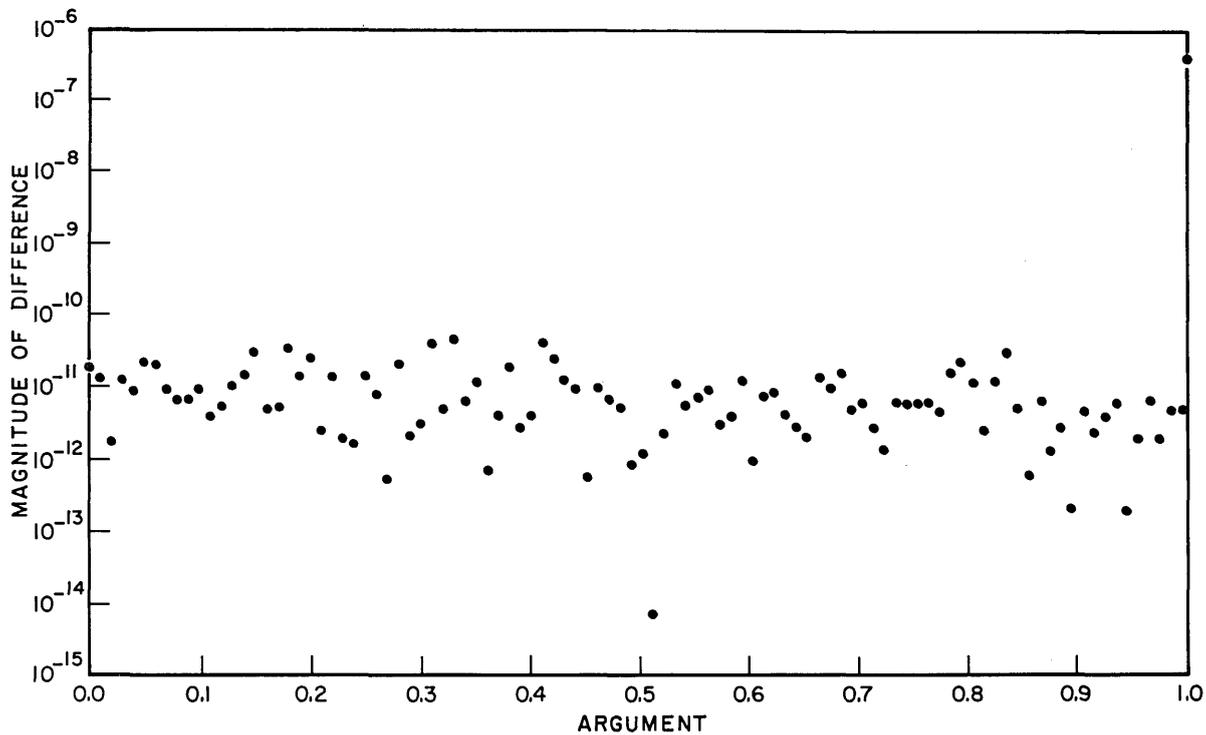
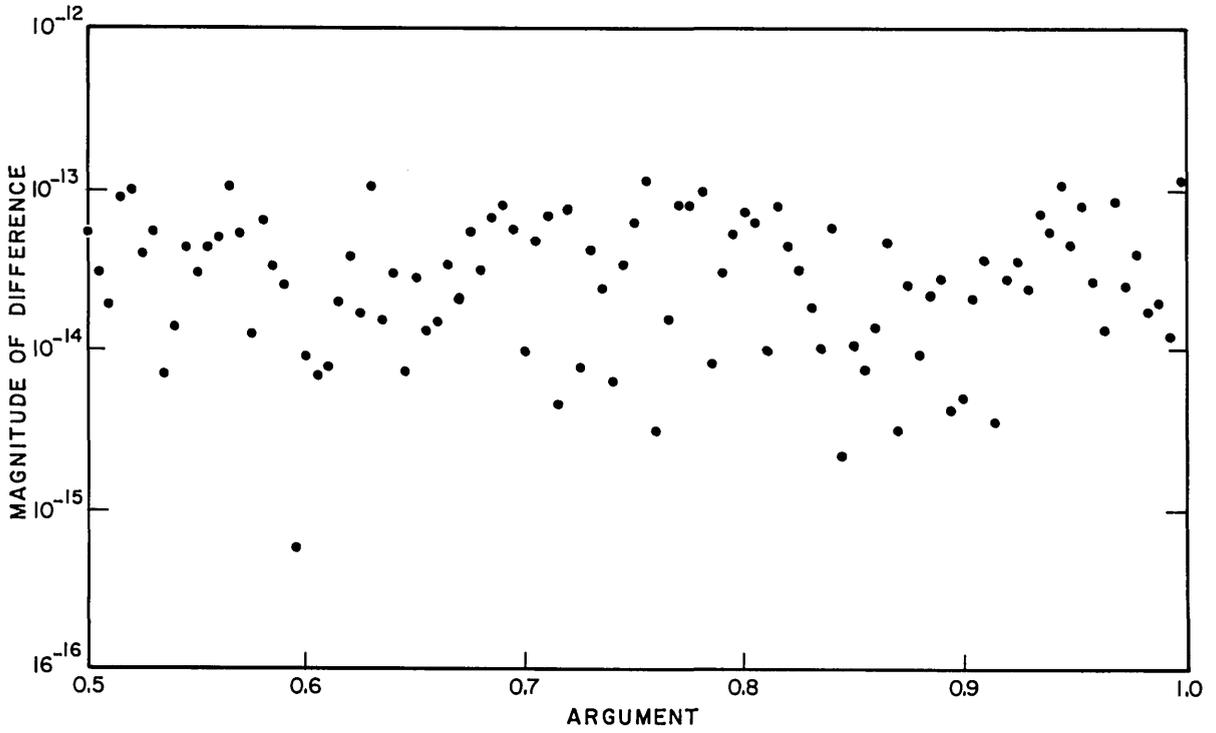
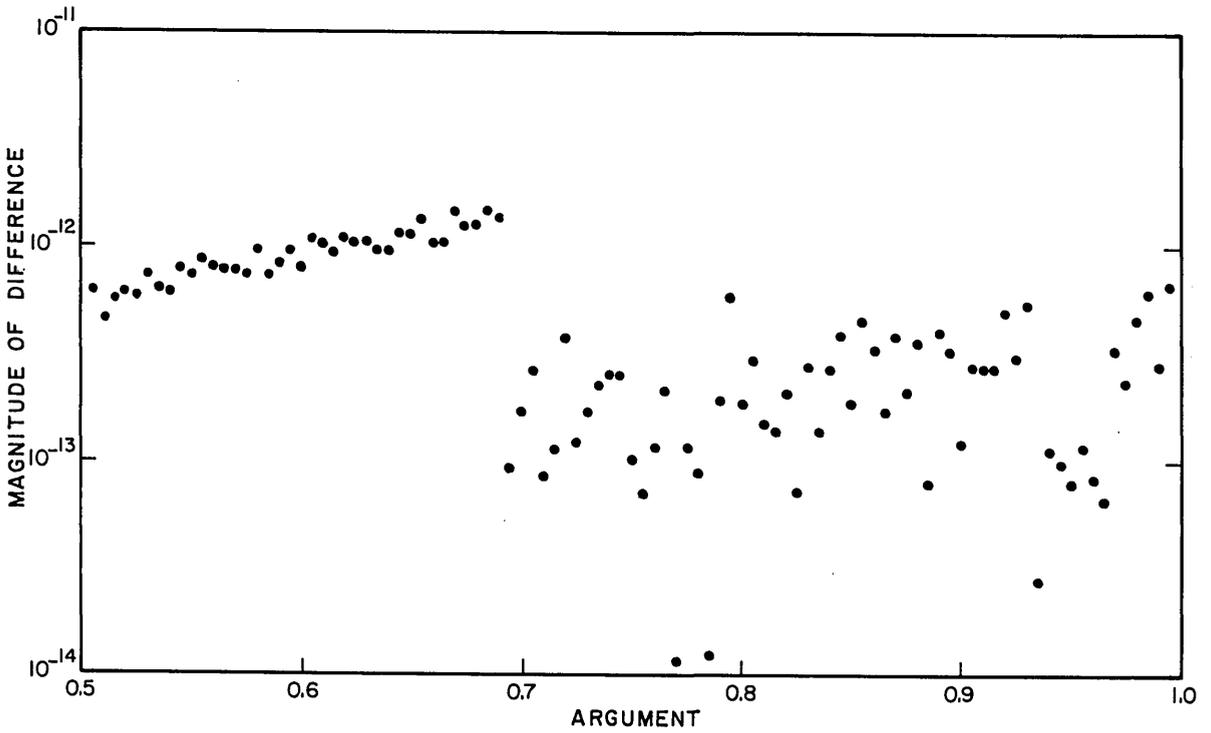


Fig. D3 - Plot of $|\arcsine_{II}(x) - \arcsine_I(x)|$ as a function of x

Fig. D4 - Plot of $|\ln_{II}(x) - \ln_I(x)|$ as a function of x Fig. D5 - Plot of $|\exp_{II}(x) - \exp_I(x)|$ as a function of x

Appendix E

Timing of NIP Pseudo-Orders

Table E1 is intended to give a rough idea of how much time is required to execute the NIP instructions on the NAREC.* The measurements were made in February, 1963, with the following arguments:

	Regular Precision	Extended Precision
(1)	0cccccccccd 00000000000	0cccccccccc 0000cccccccd
(2)	0cccccccccd 00060000000	0cccccccccc 0006cccccccd
(3)	00cccccccccd 00000000000	00cccccccccd 00000000000

*A. D. Anderson "Floating Point Time Test," NAREC Bulletin #5, Feb. 13, 1961, lists execution times for pseudo-orders in several other existing NAREC floating-point interpretive routines.

TABLE E1
Time Required to Perform NIP Instructions on the NAREC[†]

Pseudo-Order Number	Basic Coding Symbol	Regular-Precision Operation Time (msec)	Extended-Precision Operation Time (msec)	Argument	
				p	s
d0	ADL (n,s)	1.2	1.2		
d1	ADR (n,s)	1.2	1.2		
d2	MCL s	0.7	0.7		
d3	MCR s	0.7	0.7		
d4	NCL s	0.6	0.6		
d5	NCR s	0.7	0.7		
d6	CDDB (s ₁ , s ₂)	NT	NT		
d7	RSL (s ₁ , s ₂)	1.4	1.4		
d8	LSL (s ₁ , s ₂)	1.1	1.1		
d9	LSR (s ₁ , s ₂)	1.1	1.1		
da	Sin P	7.3	1100.0	(1)	
db	ArcSin P	28.7	2100.0	(1)	
dc	CMP s	3.7	5.0	(1)	(2)
dd	BYP n	NT	NT		
de	REP n	NT	NT		
df	SRNI. s	NT	NT		
e0	SKIP	0.9	0.9		
e1	JP	1.8	6.1	(3)	
e2	FL s	0.6	0.6		
e3	FR s	0.9	0.9		
e4	NL s	1.0	1.0		
e5	NR s	1.0	1.0		
e6	s P	1.2	1.2		
e7	-s P	1.3	1.7		(1)
e8	P s	0.9	0.9		
e9	s Pd	2.8	6.8	(1)	(2)
ea	-s Pd	2.3	3.6	(2)	(1)
eb	MP s	1.8	4.1	(1)	(2)
ec	P/s	2.4	16.0	(2)	(1)
ed	Exp P	9.1	1800.0	(1)	
ee	Ln P	19.4	2300.0	(1)	
ef	POW n	3.0	31.6	(1)*	
f0	PT n	NT	NT		
f1	DPPT i,j	NT	NT		
f2	√P	5.0	200.0	(1)	
f3	PC n	NT	NT		
f4	FCL s	0.7	0.7		
f5	FCR s	0.7	0.7		
f6	CDB (s ₁ ,s ₂)	NT	NT		
f7	s/P	2.6	16.0	(1)	(2)
f8	-P	0.8	1.1	(1)	
f9	/P/	1.0	1.1	-(1)	
fa	√/P/	5.0	200.0	-(1)	
fb	XP	1.2	1.0		
fc	Cos P	5.2	1100.0	(1)	
fd	/s/ Pd	3.0	6.8	(1)	
fe	ArcCos P	26.7	2100.0	(1)	
ff	MON s	NT	NT		

*n = 2.
†Note that all times in the table are given in milliseconds, and NT indicates that the instruction was not timed.

Appendix F

NIP Error Stops

There are twenty-one distinct automatic error stop sequences possible within the NIP system. When an error stop occurs, some diagnostic information is recorded via the RC (33) order before

the NAREC is halted. The reasons for stopping and the error stop sequences are shown in Tables F1 and F2 for NIP I and NIP II, respectively.

TABLE F1
NIP I Error Stops and Error Stop Sequences

Pseudo-Order Number	Reason for Stop	Error Stop Sequence
db, fe	$ P > 1$	Print P_0 , Print s_0 , Print σ_0 , and Stop at 017b
ec	$\div 0$	Print P_0 , Print s_0 , Print σ_0 , and Stop at 017b
ed	$P > 2^{11}$	Print P_0 , Print P_1 , Print σ_0 , and Stop at 017b
ee	$P \leq 0$	Print P_0 , Print P_1 , Print σ_0 , and Stop at 017b
f1	$(i+f) > a$	Print γ , Print σ_0 , and Stop at 017b
f2	$P < 0$	Print P_0 , Print P_1 , Print σ_0 , and Stop at 017b
f7	$\div 0$	Print s_0 , Print P_0 , Print σ_0 , and Stop at 017b
Any unallowed order		Print σ_0 , and Stop at 017b

TABLE F2
NIP II Error Stops and Error Stop Sequences

Pseudo-Order Number	Reason for Stop	Error Stop Sequence
da, fc	$ P > 2^{76}$ (Argument is effectively ∞)	Print σ_0 and Stop at 017b
db, fe	$ P > 1$	Print σ_0 and Stop at 017b
eb	Product will exceed the floating-point range	Print σ_0 and Stop at 0559
ec, f7	$\div 0$ or quotient will exceed the floating-point range	Print σ_0 and Stop at 017b or 0559
ed	$P > 2^{11}$	Print σ_0 and Stop at 017b
ee	$P \leq 0$	Print σ_0 and Stop at 017b
f0	$n > 13$	Print σ_0 and Stop at 017b
	$m > n$	Print σ_0 and Stop at 017b
f1	$(i+f) > a$	Print γ , Print σ_0 , and Stop at 017b
f2	$P < 0$	Print σ_0 and Stop at 017b
Any unallowed order		Print σ_0 and Stop at 017b

Appendix G

Summary of NAR Coding Symbols for NAREC Orders

TABLE G1
NAR Coding Symbols *vs* NAREC Order Number (8/10/62)

Order No.	NAR Coding Symbol(s)*	Order No.	NAR Coding Symbol(s)*
(00)	n zz	50	s a
		51	-s a
10	lo s, loiN	52	/s/ a, /s/a
11	ro s, roiN	53	-/s/ a, -/s/a
12	cl s, cliN	54	s ad
13	cr s, criN	55	-s ad
14	cul s, culiN	56	/s/ ad, /s/ad
15	cur s, curiN	57	-/s/ ad, -/s/ad
16	cs1 s, cs1iN		
17	csr s, csriN	60	m s
18	cel s, celiN	61	rm s
19	cer s, ceriN		
		70	d s, d/ s
20	lal s, laliN	71	xu, s xu, xu s
21	lar s, lariN		
22	il s, iliN	80	xa, s xa, xa s
23	ir s, iriN	81	t s1
24	s u		s2 m
26	s .au, s.au	82	stop, n stop, stop n
		83	bt s1
30	al n		s2 m
31	ar n	84	stm
32	rd s		
33	rc s	90	pcu
34	ul n	91	n pl
36	scu n, n scu		
38	ael n		
39	aer n		
40	ua, s ua, ua s		
41	au, s au, au s		
42	a s		
43	u s		
44	ya		

*Here s (or s1 or s2) generally represents a symbolic address but could be a decimal integer; n generally represents a decimal integer but could be a symbolic address; N represents a numbered instruction, *i.e.*, a decimal integer where $2 \leq N \leq 9999$; and m represents a decimal integer where $0 \leq m \leq 255$.

Appendix H

Summary of Basic and NAR Coding Symbols for NIP Pseudo-Orders, Including Brief Descriptions of the Associated NIP Operations

Pseudo-Order Number	Basic Coding Symbol	NAR Coding Symbol	Brief Descriptions of NIP Operations
d0	ADL (n,s) -fff ≤ n ≤ fff	adl n s -4095 ≤ n ≤ 4095	<u>Add n to the Left address of s.</u>
d1	ADR (n,s) -fff ≤ n ≤ fff	adr n s -4095 ≤ n ≤ 4095	<u>Add n to the Right address of s.</u>
d2	MCL s	mcl s	<u>Marker Conditional Left:</u> If P contains a marker, transfer interpretive control to s(L); otherwise, proceed to next order in sequence.
d3	MCR s	mcr s	<u>Marker Conditional Right:</u> If P contains a marker, transfer interpretive control to s(R); otherwise, proceed to next order in sequence.
d4	NCL s	ncl s	<u>NAREC Conditional Left:</u> If P is positive (or zero), exit the interpretive mode of control and transfer machine control to s(L); otherwise, proceed to next pseudo-order in sequence.
d5	NCR s	ncr s	<u>NAREC Conditional Right:</u> If P is positive (or zero), exit the interpretive mode of control and transfer machine control to s(R); otherwise, proceed to next pseudo-order in sequence.
d6	CDDB (s ₁ ,s ₂)	cddb s ₁ s ₂	<u>Convert the Decimal-point and fixed-point Decimal numbers between s₁ and s₂ to floating Binary (leaving markers unchanged) unless halted by a stopper.</u>
d7	RSL (s ₁ ,s ₂)	rsl s ₁ s ₂	<u>Place the Right address of s₁ in the Left of s₂.</u>

d8	LSL (s_1, s_2)	lsl s_1 s_2	Place the <u>Left</u> address of s_1 in the <u>Left</u> of s_2 .
d9	LSR (s_1, s_2)	lsr s_1 s_2	Place the <u>Left</u> address of s_1 in the <u>Right</u> of s_2 .
da	Sin P	sin p	Compute the <u>Sine of P</u> and place the result in the P register.
db	Arc Sin P	arcsin p	Compute the <u>Arc Sine of P</u> and place the result in the P register.
dc	CMP s	cmp s	<u>Compare P with s:</u> (1) If $P < s$, proceed to next order. (2) If $P = s$, proceed to order after next. (3) If $P > s$, proceed to second order after next.
dd	BYP n $0 \leq n \leq \text{fff}$	byp n $0 \leq n \leq \underline{4095}$	<u>Bypass</u> the next order n times.
de	REP n $0 \leq n \leq \text{fff}$	rep n $0 \leq n \leq \underline{4095}$	<u>Repeat</u> (perform n times) the loop starting at location $s+2(L)$ and terminating with the instruction $FR\ s+1$; then transfer control to $s(R)$. (The repeat order must appear only in $s(L)$, and $s+1$ must be reserved as a working location.)
df	SRNL s	srnl s	Enter the <u>Subroutine</u> at <u>NAREC Left of s</u> : Exit the interpretive mode of control and enter a subroutine at $s(L)$ with the right half (parameter portion) of the df instruction word in the left of the U register. Then, after a conventional subroutine exit, re-enter the interpretive mode of control and return to the next <i>left-hand</i> pseudo-order.

CAUTION: The interpretive mode of control must not be re-entered within the subroutine without special care.

e0	SKIP	skip	<u>Skip</u> : Proceed to next order in sequence.
e1	JP	jp	<u>Adjust P</u> (if $P \neq 0$), and transfer control to the next order in sequence.
e2	FL s	fl s	<u>Floating Left</u> : Transfer interpretive control to the order at $s(L)$.

e3	FR s	fr s	<u>Floating Right</u> : Transfer interpretive control to the order at s(R).
e4	NL s	nl s	<u>NAREC Left</u> : Exit the interpretive mode of control and transfer machine control to s(L). Upon occurrence of the re-entry instruction LO 0390, for example, re-enter the interpretive mode of control and transfer control to the next pseudo-order in sequence.

CAUTION: The interpretive mode of control cannot necessarily be re-entered in this manner unless special care has been taken to preserve the contents of the sequencing register σ at locations 0202 and 0203.

e5	NR s	nr s	<u>NAREC Right</u> : Exit the interpretive mode of control and transfer machine control to s(R). Re-entry options are the same as those for NL s.
e6	s P	s p	Transfer the contents of <u>s to P</u> .
e7	-s P	-s p	Transfer <u>Minus s to P</u> .
e8	P s	p s	Transfer the contents of <u>P to s</u> .
e9	s Pd	s pd	<u>Add s to P</u> and place the result in P.
ea	-s Pd	-s pd	<u>Subtract s from P</u> and place the result in P.
eb	MP s	mp s	<u>Multiply P by s</u> and place the result in P.
ec	P/s	p/s	<u>Divide P by s</u> and place the result in P.
ed	Exp P	exp p	Form the <u>Exponential of P</u> (e^P) and place the result in P.
ee	Ln P	ln p	Form the <u>Natural Logarithm of P</u> and place the result in P.
ef	POW n	pow n	Form the <u>(±)nth Power of P</u> and place the result in P.
	$0 \leq n \leq \frac{4095}{\text{Log}_2 P } \leq 4095$		

f0	PT n $0 \leq n \leq b$ $n = \pm n $	pt n $-11 \leq n \leq 11$	Regular precision only—Depending on the sign of the signed-magnitude integer n, <u>Print</u> n digits of P: If n is positive, print P as a rounded floating-decimal number leaving P unchanged; if n is negative, simply round, convert, and place the result in P without printing. Let n = 0 mean n = b.
	PT m,n $1 \leq n \leq 13$ m < n		Extended precision only— <u>Print</u> n digits of P as a rounded floating-decimal number in m groups. Leave P unchanged. For m = 0, PT m,n operates in the same way as a PT n does in Regular Precision for positive n.
f1	DPPT i,j $0 \leq i \leq b$ $0 \leq j \leq b$ $0 \leq (i+j) \leq b$	dppt i j $0 \leq i \leq 11$ $0 \leq j \leq 11$ $0 \leq (i+j) \leq 11$	<u>Decimal-Point Print P</u> as a single decimal-point number with j digits in the fractional part and i digits allowed for the integer part, but suppress leading zeros. If, however, the number is not suitably scaled, revert to a f0-type printout with n = b. Note that the f1 order is given in a single-address instruction.
f2	\sqrt{P}	sr p	<u>Form the Square Root of P</u> and place the result in P.
f3	PC n $0 \leq n \leq 3f$	pc n	<u>Punch n</u> : Punch on the output tape the character represented by the last 6 bits of n.
f4	FCL s	fcl s	<u>Floating Conditional Left</u> : If P is positive, transfer interpretive control to s(L); otherwise, proceed to the next order in sequence.
f5	FCR s	fcr s	<u>Floating Conditional Right</u> : If P is positive, transfer interpretive control to s(R); otherwise, proceed to the next order in sequence.
f6	CDB(s ₁ ,s ₂)	cdb s ₁ s ₂	<u>Convert the floating-Decimal numbers between s₁ and s₂ to floating-Binary</u> (leaving markers unchanged) unless halted by a stopper.

f7	s/P	s/p	Divide the contents of <u>s</u> by <u>P</u> and place the result in P.
f8	$-P$	$-p$	Form the <u>Negative of P</u> and place the result in P.
f9	$/P/$	$/p/$	Form the <u>Absolute Value of P</u> and place the result in P.
fa	$\sqrt{P/}$	$sr/p/$	Form the <u>Square Root of the Absolute Value of P</u> and place the result in P.
fb	XP	xp	<u>Clear P.</u>
fc	$\text{Cos } P$	$\text{cos } p$	Compute the <u>Cosine of P</u> and place the result in P.
fd	$/s/ Pd$	$/s/ pd$	Add the <u>Absolute Value</u> of the contents of <u>s</u> to <u>P</u> and place the result in P.
fe	$\text{Arc Cos } P$	$\text{arcos } p$	Compute the <u>Arc Cosine of P</u> and place the result in P.
ff	$\text{MON } s$	$\text{mon } s$	Turn <u>Monitor</u> on: test, print, and stop monitoring according to the directory at s.

CURRENT LIST OF NAREC REFERENCES*

- | <u>Ref.</u>
<u>No.</u> | <u>Ref.</u>
<u>No.</u> |
|--|---|
| 1 - "Programming Manual for the NAREC," J. S. Seward, NRL Report 4652, Nov. 17, 1955 | 12 - "Subroutine for Evaluation of Bessel Functions of the First Kind, Using Real Arguments and Real Positive Orders," F. Weiner, Sep. 10, 1958 |
| 2 - "Exercises for In-Service Course in NAREC Programming," R. M. Mason, Summer, 1957 | 12A - "Subroutine for Evaluation of Bessel Functions of the First Kind, Using Real Arguments and Real Positive Orders—Addendum A," D. J. Ellis, Mar. 13, 1963 |
| 3 - "Logical Operations of the NAREC," (superseded by NAREC Reference #14) | 13 - "Subroutine for Evaluation of Bessel Functions of the Second Kind, Using Real Arguments and Real Positive Integer Orders," F. Weiner, Sep. 16, 1958 |
| 4 - Never written | 14 - "Logical Operations of the NAREC (Revised December 1958)," B. Lepson and R. M. Mason, May 18, 1959 (superseded by NAREC Reference #22) |
| 5 - "Subroutine Tapes for the NAREC," B. Lepson, July 23, 1956 | 15 - "Floating Point Runge-Kutta Integration Subroutine for Systems of First-Order Ordinary Differential Equations," A. D. Anderson, Feb. 4, 1959 |
| 6 - "Subroutine for Polynomial Interpolation," J. L. Hammersmith, July 31, 1956 | 16 - "Integration Subroutine," R. M. Mason and J. P. Mason, Feb. 5, 1959 |
| 7 - "Directions for Use of Subroutine for the Runge-Kutta Integration of n First-Order Ordinary Differential Equations," J. L. Hammersmith, Apr. 16, 1956 | 16A - "Integration Subroutine (Fixed-Point Version)—Addendum A," R. M. Mason and J. P. Mason, July 25, 1962 |
| 8 - "Directions for Use of Subroutine for the Runge-Kutta Integration of up to Three, Second-Order, Ordinary Differential Equations," J. L. Hammersmith, Apr. 19, 1956 | 16B - "Integration Subroutine (Floating-Point Version)—Addendum B," R. M. Mason, July 31, 1962 |
| 9 - "Subroutine for Evaluation of Spherical Bessel Functions Using Real Arguments," F. Weiner, Apr. 24, 1958 | 16C - "Integration Subroutine (NELIAC Floating-Point Version)—Addendum C," R. M. Mason, J. P. Mason, and A. B. Bligh, Oct. 7, 1963 |
| 9A - "Subroutine for Evaluation of Spherical Bessel Functions Using Real Arguments—Addendum A," D. J. Ellis, Dec. 26, 1962 | 17 - "Code and Format of Punched Paper Tape for NAREC Input," A. B. Bligh, Aug. 29, 1959 |
| 10 - "Parabolic Interpolation Subroutine," J. L. Hammersmith, Apr. 24, 1958 | |
| 11 - "Subroutine for Roots of Polynomials," A. D. Anderson, May 7, 1958 | |

*Copies of the NAREC References are available to qualified persons upon request. Direct inquiries to Director, U.S. Naval Research Laboratory, Washington, D.C. 20390, Attn: Code 4550.

- | <u>Ref.</u>
<u>No.</u> | <u>Ref.</u>
<u>No.</u> |
|---|---|
| 18 - "Universal Data Conversion Program for NAREC Data Plotter," W. A. McCool, July 1959 | 24 - "Basic Fixed-Point Subroutine Collection for the Core Memory," E. Dent and M. Brinkman, Dec. 20, 1961 |
| 19 - "Fourier and Frequency Analysis Program with Application to the Analysis of Shock Motions," W. A. McCool, Oct. 1959 | 25 - "A Programming Glossary for NAREC Users," R. M. Mason, NRL Report 5779, Aug. 6, 1962 |
| 20 - "Solution of Systems of Simultaneous Linear Equations on the NAREC," B. Lepson and J. P. Mason, NRL Report 5372, Oct. 23, 1959 | 26 - "A NAREC Program for Least Square Error Curve Fitting," J. P. Falvey, NRL Report 5788, June 15, 1962 |
| 21 - "A Category Classification System for Computer Programs," A. B. Bligh, Mar. 27, 1961 | 27 - "A Program for the Execution of LCP-30 Machine-Language Codes on the NAREC Computer," E. E. Wald and B. Wald, NRL Report 5919, July 1, 1963 |
| 22 - "Logical Operations of the NAREC (Revised April 1961)," B. Lepson, A. B. Bligh, and R. M. Mason, Apr. 17, 1961 | 28 - "NIP—A Floating Interpretive Programming System for the NAREC Computer," E. A. Stennett, L. S. Bearce, and R. M. Mason, NRL Report 6106, Dec. 18, 1964 |
| 22A - "Correction to Description of Read Operation (Order 32)," B. Lepson and R. M. Mason, Sep. 25, 1962 | 29 - "NELIAC-N—A Tutorial Report," J. W. Kallander and R. M. Thatcher, NRL Report 5976, June 17, 1963 |
| 23 - "Memory Dump and Associated Subroutines," R. M. Mason, Nov. 15, 1961 | 30 - "NABUR—A NAREC Assembler for the Burroughs D825 Modular Data-Processing Computer," R. M. Mason and I. G. Fishman, NRL Report 5974, Jan. 13, 1964 |
| 23A - "Memory Dump and Associated Subroutines—Addendum A," J. W. Kallander, July 26, 1962 | |

CURRENT LIST OF NAREC BULLETINS*

<u>Bull.</u> <u>No.</u>	<u>Bull.</u> <u>No.</u>
1 - "New Left Shift Order (Logical)," B. Lepson, Oct. 28, 1959	10A - "Floating-Point Complex Number Arithmetic and Square Root Subroutines—Revision A," I. G. Fishman, May 2, 1962
2 - "Preliminary Discussion of the NAREC Magnetic Core Memory," B. Lepson, Oct. 28, 1959	11 - "Summary of Headings in the Standardized Format for Reporting on Subroutines and Programs," A. B. Bligh, May 18, 1961
3 - "Some Elementary Numerical-Analytic Considerations Involved in Preparing a Problem for Computation," J. L. Hammersmith, May 27, 1960	12 - "Memory Recording Subroutine," J. P. Falvey, May 25, 1961
4 - "Floating-Point Binary to Decimal Conversion without Printout," A. T. Hind, Aug. 29, 1960	13 - "Program Readdressing Subroutine," L. S. Bearce and J. P. Falvey, June 22, 1961
5 - "Floating Point Time Test," A. D. Anderson, Feb. 13, 1961	14 - "Numerical Fourier Analysis I," C. G. Myers, June 5, 1961
6 - "Description of Revised Fixed-Point Subroutine Tape (Tape 3400)," E. Dent and A. B. Bligh, Mar. 1, 1961	15 - "Numerical Fourier Analysis II," C. G. Myers, June 5, 1961
7 - "New Punched Tape Input Orders (Single Character Read—Order Number 36) (Stop Tape Motion—Order Number 84)," B. Lepson and R. M. Mason, Mar. 10, 1961	16 - "Basic Fixed-Point Subroutine Tape for the Core Memory," E. Dent and M. Brinkman, June 12, 1961
8 - "A Standardized Format for Reporting on Subroutines and Programs," A. B. Bligh, Apr. 17, 1961	17 - "Possible Change in Name of the T Register," A. B. Bligh, June 22, 1961
9 - "Numeric Record Subroutine No. 1," A. B. Bligh, Apr. 20, 1961	18 - "Sine-Cosine Source Program," E. Dent and M. Brinkman, June 22, 1961
10 - "Floating-Point Complex Number Arithmetic and Square Root Subroutines," E. Wingfield, May 17, 1961	19 - "A Proposed Convention for Labelling NAREC Punched Tapes," A. B. Bligh, July 19, 1961
	20 - "Subroutine for Evaluation of Bessel Functions of the First Kind for Orders Zero and One, Using Complex Arguments," J. P. Mason, July 12, 1961
	20A - "Subroutine for Evaluation of Bessel Functions of the First Kind for Orders Zero and One, Using Complex Arguments—Addendum A," J. P. Mason, Mar. 13, 1962

*Copies of the NAREC Bulletins are available to qualified persons upon request. Direct inquiries to Director, U.S. Naval Research Laboratory, Washington, D.C. 20390, Attn: Code 4550.

- | <u>Bull.</u>
<u>No.</u> | <u>Bull.</u>
<u>No.</u> |
|---|---|
| 20B - "NELIAC Function to Evaluate Bessel Functions of the First and Second Kinds for Orders Zero and One, Using Complex Arguments," J. P. Mason, Aug. 3, 1964 | 29 - "English Text Readin Subroutine," A. B. Bligh, Sep. 8, 1961 |
| 21 - "Output Flexowriter Format Information," D. E. Tillett, June 28, 1961 | 30 - "English Text Record Subroutine," A. B. Bligh, Oct. 12, 1961 |
| 22 - "Subroutine for Evaluation of Bessel Functions of the First Kind for Orders Zero and One, Using Real Arguments," J. P. Mason, July 26, 1961 | 31 - "NAR IA," A. B. Bligh, Dec. 19, 1961 |
| 22A - "Subroutine for Evaluation of Bessel Functions of the First Kind for Orders Zero and One, Using Real Arguments—Addendum A," J. P. Mason, Mar. 13, 1962 | 32 - "Subroutine for Finding the Zeros of J_0 and J_1 ," J. P. Mason, Oct. 6, 1961 |
| 22B - "NELIAC Function to Evaluate Bessel Functions of the First and Second Kinds for Orders Zero and One, Using Positive Real Arguments," J. P. Mason, Sep. 18, 1964 | 32A - "Subroutine for Finding the Zeros of J_0 and J_1 —Addendum A," J. P. Mason, Mar. 13, 1962 |
| 22C - "NELIAC Function to Evaluate Bessel Functions of the First Kind for Orders Zero and One, Using Positive Real Arguments," J. P. Mason, Nov. 13, 1964 | 33 - "Memory Recording Subroutine (Revision No. 1)," J. Falvey, Dec. 27, 1961 |
| 23 - "Square Root and Square Root Absolute Source Program," E. Dent and M. Brinkman, July 24, 1961 | 33A - "Memory Recording Subroutine (Revision No. 2)," J. P. Falvey, Sep. 27, 1962 |
| 24 - "Negative Exponential Source Program," E. Dent and M. Brinkman, Aug. 11, 1961 | 34 - "Snapshot System (Preliminary Version)," J. W. Kallander, Oct. 17, 1961 |
| 25.1 - "Natural Logarithm Source Program," E. Dent and M. Brinkman, Dec. 7, 1961 | 35 - "Equation Solving Routine," R. E. McGill, Dec. 29, 1961 |
| 26 - "Source Program for Multiple or Single Word Conversion from Decimal to Binary," E. Dent and M. Brinkman, Aug. 24, 1961 | 36 - "Data Read In and Conversion Subroutine No. 1," J. P. Falvey, Dec. 28, 1961 |
| 27 - "NIP I," L. S. Bearce, E. A. Stennett, and R. M. Mason, July 27, 1961 | 37 - "Programmer's Glossary for NAREC Users (Preliminary Version)," R. M. Mason, Dec. 8, 1961 |
| 28 - "Source Program for Arithmetic Overflow Check," E. Dent and M. Brinkman, Aug. 24, 1961 | 38 - "Arcsine and Arccosine Source Program," E. Dent and M. Brinkman, Feb. 19, 1962 |
| | 39 - "Binary to Decimal Rounded n Significant Digits Record and Omit Record Source Program," E. Dent and M. Brinkman, Feb. 19, 1962 |
| | 40 - "Binary to Decimal Rounded Record and Omit Record Source Program," E. Dent and M. Brinkman, Feb. 19, 1962 |
| | 41 - "Programs and Subroutines for Solving Systems of Simultaneous Linear Equations," J. P. Mason, Apr. 5, 1962 |

- | <u>Bull.</u>
<u>No.</u> | <u>Bull.</u>
<u>No.</u> |
|---|---|
| 41A - "NELIAC Functions to Solve Systems of Simultaneous Linear Equations," J. P. Mason, Feb. 5, 1964 | 51 - "Least Square Smoothing of Equally Spaced Data," A. T. Hind, Aug. 28, 1962 |
| 42 - "Subroutines to Transform a System of Complex Equations into the Form Required by the Simultaneous Equations Routine," J. P. Mason, Mar. 28, 1962 | 52 - "Standard Input Form Memory Recording Subroutine," G. D. Harlow, Sep. 19, 1962 |
| 42A - "NELIAC Function to Transform a System of Complex Equations into the Form Required by the Simultaneous Linear Equations Functions," J. P. Mason, Mar. 6, 1964 | 53 - "Source Program for the Evaluation of Bessel Functions of the First and Second Kinds for Order Zero and One, Using Real Arguments," R. E. McGill, Oct. 4, 1962 |
| 43 - "Program for Finding the Inverse of a Matrix," J. P. Mason, Apr. 4, 1962 | 54 - "A Subroutine for Converting Floating-Point Data into Linear or Semi-Log Plotter Coordinates," E. A. Stennett, Dec. 6, 1962 |
| 43A - "NELIAC Function to Find Inverse of a Matrix," J. P. Mason, Feb. 3, 1964 | 55 - "A Subroutine for Preparing a Log-Log Plot from a Given Set of Floating-Point Data," G. Chayt, Dec. 27, 1962 |
| 44 - "Simulation of the LCP-30 Computer," E. E. Wald and B. Wald, Mar. 6, 1962 | 56 - "Decimal Point Data Conversion Subroutine," M. Brinkman, Dec. 26, 1962 |
| 45 - "Snapshot System," J. W. Kallander, Oct. 31, 1962 | 57 - "Translating Readin Subroutine," D. J. Ellis and A. B. Bligh, Mar. 12, 1963 |
| 45A - "NELIAC-N Snapshot," M. Brinkman and J. W. Kallander, Sep. 15, 1964 | 58 - "Subroutine for Evaluating Elliptic Integrals," J. P. Mason, Feb. 18, 1963 |
| 46 - "Recent Progress in Equipment Development and Programming," A. B. Bligh, Aug. 10, 1962 | 59.1 - "Extended-Precision Solution of Systems of Simultaneous Linear Equations," J. P. Mason, Aug. 16, 1963 |
| 47 - "New Logical Comparison Orders 18 and 19," A. B. Bligh, I. J. Levy, R. M. Mason, and B. Lepson, Aug. 10, 1962 | 59A - "NELIAC Bridge to Solve a Set of Simultaneous Linear Equations in Extended Precision," J. P. Mason, Aug. 3, 1964 |
| 48 - "New Logical Shift Orders 38 and 39 (A Entire Left and A Entire Right)," B. Lepson, A. B. Bligh, I. J. Levy, and R. M. Mason, Aug. 10, 1962 | 60 - "Alphanumeric Tape Readin Subroutine," A. B. Bligh and I. C. Fishman, Apr. 22, 1963 |
| 49 - "New Y Register Sense Order 44," B. Lepson, A. B. Bligh, I. J. Levy, and R. M. Mason, June 30, 1964 | 61 - "NELIAC-N Definitive Memoranda," J. W. Kallander, Apr. 27, 1963 |
| 50.1 - "Character Code Translation Tables," I. J. Levy, Mar. 24, 1964 | 61A - "NELIAC-N Definitive Memorandum #1," J. W. Kallander, Apr. 27, 1963 |
| | 61B - "NELIAC-N Definitive Memorandum #2," J. W. Kallander, Apr. 29, 1963 |

- | <u>Bull.</u>
<u>No.</u> | <u>Bull.</u>
<u>No.</u> |
|--|--|
| 61C - "NELIAC-N Definitive Memorandum #3,"
J. W. Kallander, Apr. 30, 1963 | 74 - "NELIAC Function for Fixed Point Integer
Division with Roundoff," J. W. Kallander,
Apr. 2, 1964 |
| 62 - "Subroutines to Find Eigenvalues of a 3×3
and a 4×4 Real Matrix," D. J. Ellis, May
14, 1963 | 75 - "Precompiled NELIAC Packages," J. W.
Kallander, June 29, 1964 |
| 63 - "Time Unit Conversion Subroutines," D. J.
Ellis, May 14, 1963 | 76 - "BCD Data Conversion Functions," M.
Brinkman, Oct. 6, 1964 |
| 64 - "Reports on NELIAC Work" A. B. Bligh,
Oct. 4, 1963 | 77 - "Numerical Fourier Analysis," C. G. Myers,
Oct. 5, 1964 |
| 65 - "NELIAC Memory Dump," M. Brinkman,
Oct. 24, 1963 | 78 - Forthcoming |
| 66.1 - "NELIAC-NIP Conversion Functions," M.
Brinkman, Oct. 6, 1964 | 79 - "Magnetic Tape Functions," S. L. Hill,
June 24, 1964 |
| 67 - "Binomial Coefficient," A. B. Bligh, Oct. 29,
1963 | 80 - "New Integer Substitution Orders 28 and
29," I. J. Levy, June 29, 1964 |
| 68 - "NELIAC Function for Matrix Multiplica-
tion," J. P. Mason, Feb. 3, 1964 | 81 - "New Line Printer Top of Form Order 92,"
I. J. Levy, June 29, 1964 |
| 69 - "Recent Developments in NAREC Hard-
ware," A. B. Bligh and I. J. Levy, Feb. 10,
1964 | 82 - "NELIAC Function to Solve a Cubic Equa-
tion Having Real Coefficients," J. P. Mason,
July 1, 1964 |
| 70 - "Determinant Evaluation Function," E. A.
Stennett, Apr. 1, 1964 | 83 - "NELIAC Functions to Generate Plotter
Tapes (Simplified)," G. J. Flenner, Nov.
25, 1964 |
| 70A - "Extended Precision Determinant Evalua-
tion Function," E. A. Stennett, Sep. 15,
1964 | 84 - "NELIAC Functions to Generate Plotter
Tapes (General)," G. J. Flenner, Nov. 4,
1964 |
| 71 - "Repeat Loop Subroutine," A. B. Bligh,
Mar. 17, 1964 | 85 - "NELIAC Function to Evaluate Struve
Functions for Orders Zero and One,"
M. Brinkman and J. P. Mason, Nov. 13,
1964 |
| 72 - "NELIAC Function for Bioctal Dumps," J.
W. Kallander, Apr. 1, 1964 | 86 - "NELIAC Functions to Evaluate Sine and
Cosine Integrals," M. Brinkman, Nov. 13,
1964 |
| 73 - "Linear Interpolate Function," A. B. Bligh,
Apr. 1, 1964 | 87 - "NELIAC Functions to Execute Complex
Arithmetic," C. M. Howe, Nov. 12, 1964 |

CURRENT LIST OF NELIAC REFERENCES*

Ref.
No.

- 1 - "NELIAC-N-A Tutorial Report," J. W. Kallander and R. M. Thatcher,
NRL Report 5976, June 17, 1963, NAREC Reference #29

*Each NELIAC Reference is also assigned a NAREC Reference number as its primary number.

CURRENT LIST OF NELIAC BULLETINS*

<u>Bull.</u> <u>No.</u>	<u>Bull.</u> <u>No.</u>
1 - "NELIAC-N Definitive Memoranda," J. W. Kallander, Apr. 27, 1963, NAREC Bulletin #61	Mason, Feb. 5, 1964, NAREC Bulletin #41A
1A - "NELIAC-N Definitive Memorandum #1," J. W. Kallander, Apr. 27, 1963, NAREC Bulletin #61A	8A - "NELIAC Bridge to Solve a Set of Simultaneous Linear Equations in Extended Precision," J. P. Mason, Aug. 3, 1964, NAREC Bulletin #59A
1B - "NELIAC-N Definitive Memorandum #2," J. W. Kallander, April 29, 1963, NAREC Bulletin #61B	9 - "NELIAC Function for Matrix Multiplication," J. P. Mason, Feb. 3, 1964, NAREC Bulletin #68
1C - "NELIAC-N Definitive Memorandum #3," J. W. Kallander, Apr. 30, 1963, NAREC Bulletin #61C	10 - "Determinant Evaluation Function," E. A. Stennett, Apr. 1, 1964, NAREC Bulletin #70
2 - "Reports on NELIAC Work," A. B. Bligh, Oct. 4, 1963, NAREC Bulletin #64	10A - "Extended Precision Determinant Evaluation Function," E. A. Stennett, Sep. 15, 1964, NAREC Bulletin #70A
3 - "Integration Subroutine (NELIAC Floating-Point Version)—Addendum C," R. M. Mason, J. P. Mason, and A. B. Bligh, Oct. 7, 1963, NAREC Reference #16C	11 - "NELIAC Function to Transform a System of Complex Equations into the Form Required by the Simultaneous Linear Equations Functions," J. P. Mason, Mar. 6, 1964, NAREC Bulletin #42A
4 - "NELIAC Memory Dump," M. Brinkman, Oct. 24, 1963, NAREC Bulletin #65	12 - "NELIAC Function for Biocatal Dumps," J. W. Kallander, Apr. 1, 1964, NAREC Bulletin #72
5.1 - "NELIAC-NIP Conversion Functions," M. Brinkman, Oct. 6, 1964, NAREC Bulletin #66.1	13 - "Linear Interpolate Function," A. B. Bligh, Apr. 1, 1964, NAREC Bulletin #73
6 - "Binomial Coefficient," A. B. Bligh, Oct. 29, 1963, NAREC Bulletin #67	14 - "NELIAC Function for Fixed Point Integer Division with Roundoff," J. W. Kallander, Apr. 2, 1964, NAREC Bulletin #74
7 - "NELIAC Function to Find Inverse of a Matrix," J. P. Mason, Feb. 3, 1964, NAREC Bulletin #43A	15 - "Precompiled NELIAC Packages," J. W. Kallander, June 29, 1964, NAREC Bulletin #75
8 - "NELIAC Functions to Solve Systems of Simultaneous Linear Equations," J. P.	16 - "BCD Data Conversion Functions," M. Brinkman, Oct. 6, 1964, NAREC Bulletin #76
	17 - "Magnetic Tape Functions," S. Hill, June 24, 1964, NAREC Bulletin #79

*Each NELIAC Bulletin is also assigned a NAREC Bulletin or Reference number as its primary number.

- | <u>Bull.</u>
<u>No.</u> | <u>Bull.</u>
<u>No.</u> |
|---|---|
| 18 - "NELIAC Function to Solve a Cubic Equation Having Real Coefficients," J. P. Mason, July 1, 1964, NAREC Bulletin #82 | Zero and One, Using Positive Real Arguments," J. P. Mason, Nov. 13, 1964, NAREC Bulletin #22C |
| 19 - "NELIAC Function to Evaluate Bessel Functions of the First and Second Kinds for Orders Zero and One, Using Complex Arguments," J. P. Mason, Aug. 3, 1964, NAREC Bulletin #20B | 22 - "NELIAC Functions to Generate Plotter Tapes (Simplified)," G. J. Flenner, Nov. 25, 1964, NAREC Bulletin #83 |
| 20 - "NELIAC-N Snapshot," M. Brinkman and J. W. Kallander, Sep. 15, 1964, NAREC Bulletin #45A | 23 - "NELIAC Functions to Generate Plotter Tapes (General)," G. J. Flenner, Nov. 4, 1964, NAREC Bulletin #84 |
| 21 - "NELIAC Function to Evaluate Bessel Functions of the First and Second Kinds for Orders Zero and One, using Positive Real Arguments," J. P. Mason, Sep. 18, 1964, NAREC Bulletin #22B | 24 - "NELIAC Function to Evaluate Struve Functions for Orders Zero and One," M. Brinkman and J. P. Mason, Nov. 13, 1964, NAREC Bulletin #85 |
| 21A - "NELIAC Function to Evaluate Bessel Functions of the First Kind for Orders | 25 - "NELIAC Functions to Evaluate Sine and Cosine Integrals," M. Brinkman, Nov. 13, 1964, NAREC Bulletin #86 |
| | 26 - "NELIAC Functions to Execute Complex Arithmetic," C. M. Howe, Nov. 12, 1964, NAREC Bulletin #87 |

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) U.S. Naval Research Laboratory Washington, D.C. 20390		2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	
		2b. GROUP	
3. REPORT TITLE NIP - A Floating-Point Interpretive Programming System for the NAREC Computer			
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) This is an interim report on the problem.			
5. AUTHOR(S) (Last name, first name, initial) Stennett, Edwin A., Bearce, Loren S., and Mason, Robert M.			
6. REPORT DATE December 18, 1964		7a. TOTAL NO. OF PAGES 54	7b. NO. OF REFS
8a. CONTRACT OR GRANT NO. 54R01-08; 45B02-03		9a. ORIGINATOR'S REPORT NUMBER(S) NRL Report 6106 NAREC Reference #28	
b. PROJECT NO. RF 006-02-41-4351; RR 003-09-41-5101		9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) None	
c.			
d.			
10. AVAILABILITY/LIMITATION NOTICES Unlimited availability - available from CFSTI			
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY Office of Naval Research	
13. ABSTRACT An interpretive programming system has been written for the U.S. Naval Research Laboratory's medium-speed electronic digital computer (NAREC). This system, known as NIP (acronym for NAREC Interpretive Programming system), provides for automatic floating-point computation over the range 10^{-1233} to 10^{+1233} in either of two modes. One mode, called "extended precision," allows for computational results containing as many as nineteen significant decimal digits. The other (and historically the first) mode, called "regular precision," allows for as many as eleven significant decimal digits. Using the same program, it is possible to compute in either mode, which provides a basis for determining the cumulative effect of computational precision on results. This report discusses conventions and defines instructions which are available for writing a program in the NIP language. A brief description of the floating-point system, an example of a program written in extended precision, and a table of instruction execution times are included in the appendixes.			

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Digital computers NAREC Mathematical computer programming NIP (NAREC Interpretive Program) Programming manual Interpretive system Floating-point arithmetic Pseudo-instruction Order code Double precision arithmetic Computational aids Automatic scaling						

INSTRUCTIONS

1. **ORIGINATING ACTIVITY:** Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization (*corporate author*) issuing the report.
- 2a. **REPORT SECURITY CLASSIFICATION:** Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.
- 2b. **GROUP:** Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.
3. **REPORT TITLE:** Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.
4. **DESCRIPTIVE NOTES:** If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.
5. **AUTHOR(S):** Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.
6. **REPORT DATE:** Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.
- 7a. **TOTAL NUMBER OF PAGES:** The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.
- 7b. **NUMBER OF REFERENCES:** Enter the total number of references cited in the report.
- 8a. **CONTRACT OR GRANT NUMBER:** If appropriate, enter the applicable number of the contract or grant under which the report was written.
- 8b, 8c, & 8d. **PROJECT NUMBER:** Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.
- 9a. **ORIGINATOR'S REPORT NUMBER(S):** Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.
- 9b. **OTHER REPORT NUMBER(S):** If the report has been assigned any other report numbers (*either by the originator or by the sponsor*), also enter this number(s).
10. **AVAILABILITY/LIMITATION NOTICES:** Enter any limitations on further dissemination of the report, other than those

imposed by security classification, using standard statements such as:

- (1) "Qualified requesters may obtain copies of this report from DDC."
- (2) "Foreign announcement and dissemination of this report by DDC is not authorized."
- (3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."
- (4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."
- (5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. **SUPPLEMENTARY NOTES:** Use for additional explanatory notes.
12. **SPONSORING MILITARY ACTIVITY:** Enter the name of the departmental project office or laboratory sponsoring (*paying for*) the research and development. Include address.
13. **ABSTRACT:** Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.
 It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as (TS), (S), (C), or (U).
 There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.
14. **KEY WORDS:** Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, roles, and weights is optional.