NPG Report No.1374

THE NAVAL ORDNANCE RESEARCH CALCULATOR (NORC) COMPILER

U. S. NAVAL PROVING GROUND
DAHLGREN, VIRGINIA

Date 17 May 1955

U. S. Naval Proving Ground
Dahlgren, Virginia

The Naval Ordnance Research Calculator (NORC) Compiler

by

Karl Kozarsky
Computation and Ballistics Department

NPG REPORT NO. 1374

Foundational Research
Project K-11011-9

17 May 1955

APPROVED: J. F. BYRNE
Captain, USN
Commander, Naval Proving Ground

E A Ruckner
E. A. RUCKNER
Captain, USN
Ordnance Officer
By direction

- -

CONTENTS

ABSTRACT

This report describes the current status of the Naval Ordnance Research Calculator (NORC) Compiler. This routine is an automatic coding technique designed to minimize the time spent on the coding of problems for solution on NORC. The compiler converts unordered symbolic or relative codes to ordered machine (absolute) codes, generates and assembles subroutines from a library file, provides a variety of other services for the coder, and writes the final coding on tape, ready to be run on NORC.

Use of the compiler enables a considerable amount of tedious coding time to be saved. The time required to compile a program is usually two minutes or less of NORC machine time.

FOREWORD

This is a report on Foundational Research Project NPG-K-11011-9, "Automatic Coding Techniques". This is the twenty-sixth partial report submitted under the Foundational Research Program of the Naval Proving Ground.

This routine was devised at intervals between November 1953 and December 1954, by Gene H. Gleissner and the author.

This report was reviewed by:

R. A. NIEMANN, Acting Director
Computation and Ballistics Department
R. H. LYDDANE, Assistant Director of Research
N. A. M. RIFFOLT, Director of Research

INTRODUCTION

According to the First Glossary of Programming Terminology (reference (a)), a compiler is "an executive routine which before the desired computation is started, translates a program expressed in pseudo-code into machine code (or into another pseudo-code for further translation by an interpreter). In accomplishing the translation, the compiler may be required to:

Decode - to ascertain the intended meaning of the individual characters or groups of characters in the pseudo-coded program....

Select - to choose a needed subroutine from a file of subroutines.

Generate - to produce a needed subroutine from parameters and skeletal coding.

Allocate - to assign storage locations to the main routines and subroutines, thereby fixing the absolute values of any symbolic addresses. In some cases allocation may require segmentation.

Assemble - to integrate the subroutines (supplied, selected, or generated) into the main routine, i.e., to:

a. Adapt - to specialize to the task at hand by means of preset parameters.

b. Orient - to change relative and symbolic addresses to absolute form.

c. Incorporate - place in storage.

Record - to produce a reference record."

The NORC compiler satisfies this definition for the most part and perhaps a bit more. The details are discussed below.

The compiler is one of several service routines (diagnostic, editing, etc.), designed to appreciably shorten the time between the origination and the solution of a problem. The compiler accomplishes this by relieving the coder of much of the routine work necessary for the preparation of a problem for solution on an automatic computer.

The form a compiler assumes is largely a function of the machine for which it is designed and the aims of the users. (The energy and ambition of the compiler's authors ought not to be omitted as a significant factor.) Thus, the floating decimal arithmetic 3-address logic of the NORC makes unnecessary a slow interpretive routine to cope with most problems. The ordering of the coding is left to the compiler since the fast tape system permits this accomplishment without increasing the total compiling time excessively. The advantages of using the NORC Compiler are not obtained at the cost of flexibility or increased running time of the program. Little restriction is imposed on the placing of information in the memory. For most problems the final machine coding hardly differs from coding performed directly in machine language.

Briefly, the coder and machine share their duties in the following manner:

The problem is coded on compiler coding sheets using symbolic codes for machine addresses and certain helpful compiler codes which shorten the task. Insertions in coding which has already been completed are not done by painful rearrangement of coding lines, but by the simple expedient of assigning appropriate symbolic locations to the insertions.

The compiler orders the coding, converts the symbolic coding to machine codes, assembles and incorporates subroutines (modifying them first if it is so required) into the problem; its output is a program on tape ready to be started by a standard operational procedure.

A minimal familiarity with the salient characteristics of the NORC is assumed in what follows and a brief description of the machine is included in Appendix (B).

DESCRIPTION

Coding without using a compiler is usually done first by assigning symbolic codes for machine locations. When the problem is completed in this way, storage assignments can then be made and the symbolic addresses are replaced by absolute machine addresses. Much rearrangement, erasures, and insertions are also generally required before the coding can be given to the keypunchers. However, to code a problem of any complexity directly in the machine codes requires an audacity and clairvoyance not found in ordinary coders.

Using the NORC Compiler, the coder still performs the initial symbolic coding; however, most of the remaining clerical duties are handled automatically, by the compiler, with far greater speed and accuracy. A discussion of the compiler coding sheets will indicate how this is done. (See Appendix (A), Figure 1). There are 7 fields on this form; twelve digits are allotted to each of the PQ, R, S and T fields, 8 to the location field and 8 to the C field. These 64 digits, comprising 1 line of compiler coding, are punched on one IBM card. The NORC card-to-tape-to-card machine (CTC) accepts 64 digits, 4 NORC words, per card in the preparation of magnetic tapes. The use of 64 digits per line of compiler coding allows ample digits for each field, and since a common keypunching error is the omission of a coding line, this arrangement permits a card to be inserted without rearranging all words subsequent to it in the same block.

The PQ, R, S and T fields may be lumped together for purposes of discussion. The expression field may be dispensed with. In this field the coder usually writes a hint (whose meaning to him rapidly vanishes) as to what operation is to be performed on that line, or what quantity is stored there.

There remain then three topics: (1) the location field, (2) the C field, and (3) the PQ, R, S and T fields.

The Location Field:

To the control section of a machine, the labels identifying each word are the absolute machine addresses; in NORC, these are the consecutive numbers 0001 to 2000. However, the coder may, and usually does, arbitrarily designate symbolic addresses to identify the words constituting his program. The coder benefits by the flexibility that symbolic addresses permit him during the construction of his program. This is particularly true from the point of view of organization of the information in the memory as well as in making corrections and insertions. Of course, a correspondence between these symbolic addresses and machine addresses must eventually be made, and all references to symbolic addresses in the program must be converted to their corresponding machine addresses. The coder need not spend his time performing this routine conversion since the machine, under the direction of the compiler, readily performs this task.

For the NORC compiler, symbolic locations may be designated numerically by as many as 8 digits. In practice it has been found convenient to place an arbitrary decimal point so that there are, say, 3 digits to the left and 5 to the right of the point, e.g., 107.3 and 894.26701. However, the same number of digits to the left of the point (the first digit being nonzero) should be used throughout a problem since, in order to simplify keypunching, the first significant digit of each symbolic address is punched in the same card column.

It is not necessary to assign a symbolic location to every word on the coding sheet; its location field may be left blank (zero) in which case it is assumed to follow the word preceding it on the coding sheet. Indeed, it is often preferable to minimize the number of symbolic locations. Of course, if a word is explicitly referred to, it must have a symbolic location.

If it is desired to refer to a word whose location field is blank, it is possible to do so by relative referencing which is discussed in detail below.

A major advantage of the compiler is its ability to order the coding in ascending order on the location field, blank locations being treated as if they were rigidly attached to the symbolic location preceding them. It is not necessary, then, to deface coding sheets to make an insertion; one simply assigns appropriate symbolic locations to the insertions. Thus, if it is desirable to insert a word between the words whose locations are 102.3 and 102.4, it can be labelled 102.35 or, for that matter, 102.30001. (The latter choice would not be wise since it may be necessary later to insert words between 102.3 and 102.30001.)

If the inserted word is followed by any number of words with blank location fields, they will also be inserted. If it should be necessary to insert words between words with blank location fields, then appropriately sequential symbolic locations should be assigned to the blank words as well as the inserted ones. It has been found that the 8 digits allowed the location field are quite adequate for these purposes.

The PQ, R, S and T Fields:

Each of these fields contain allowance not only for 8 digit symbolic locations, but for 4 digits of absolute coding.

After the compiler has converted symbolic addresses which may be in the PQ, R, S and T fields to machine addresses, the absolute codes which may be in these fields are added to their respective converted symbolic codes. The use of this is apparent when considering the automatic address modifiers, M_4, M_6, and M_8 whose operation is selected by codes in the R, S, and T field exceeding 3999. (See example Appendix (A) Figure 1 line 3.) Of course, in an instruction word, the PQ field has a symbolic part of zero and an absolute part which is not zero. Furthermore, this scheme of allotting to the PQ, R, S and T fields an absolute as well as a symbolic portion allows relative referencing, e.g., the fourth word after the word whose symbolic location is 387.4 is noted by a 387.4 in the symbolic portion and a 0004 in the absolute portion of the field concerned.

In addition, since instructions which are not referred to need not have a symbolic location, the number of symbolic locations tends to be kept down to a minimum. (In a few problems coded so far the proportion of words with nonzero symbolic locations was only 1/4 to 1/3 the total number of compiler words, and a still smaller fraction of the absolute machine words.) This, coupled with the efficiency of the NORC instruction repertoire, makes the compiler's limitation of a maximum of 1300 nonzero symbolic locations for each problem adequate for even very long complicated problems.

The C Field:

The remaining field to be discussed is the C field, which contains codes which are interpreted by the compiler. The C codes identify the word associated with it as one of the following types:

00: The normal C code is zero which signifies only that the word in the PQ, R, S, and T fields is to be converted to machine coding.

01: Whenever a transfer of control to a subroutine is required an 01 C code is necessary. The form of this instruction must be obtained by looking up the subroutine in the subroutine directory, but the symbolic portion of the T field always contains the identification number of the subroutine. A typical example is Appendix (A) Figure 1, line 4.

If the subroutine is not already in this segment, the compiler will extract the subroutine from the subroutine library, assemble it so that it will function in the memory positions where it will be inserted (in the memory following the coder's coding), and make up the transfer of control. If the subroutine has already been incorporated in this segment, it will not be again included, but the transfer of control will be properly constructed.

The ample maximum of 200 different subroutines per segment is permitted by the compiler.

02: For subroutines which can make use of preset parameters, an 02 C code is used to identify the word as a preset parameter for the subroutines indicated by the 01 code preceding this word.

The 02 codes must immediately follow the associated 01 code and should have a blank location field.

This 02 word does not appear as such in the final machine coding, but is a parameter to modify the subroutine before the subroutine is incorporated into the final coding.

The subroutine directory contains the information as to the order and form of the 01 and following 02 codes.

03: This is an instruction to the compiling routine to assign a block of K + 1 consecutive locations to the current segment. K is in the absolute portion of the PQ field. (These K + 1 locations will be zero in the final machine coding.) This is intended to assist the coder in organizing his problem, and to save keypunching. (See Appendix (A)).

04: An example perhaps will best illustrate the use of this code: Assume 2 locations, say 241.7 and 384.2, are only used to store various quantities temporarily. But later it becomes apparent that they are not used simultaneously and that one of them, say 241.7, will suffice. Then 384.2 is to be deleted, but instead of searching through the coding and changing all references to 384.2 to 241.7, it suffices to do the following:

Change the C field of location 384.2 to 04 and the symbolic portion of the PQ field to 241.7.

The 04 code then states: (1) Do not enter this word into the final coding, and (2) convert any references to this location to the machine address equivalent to the location in the PQ field.

This code is designed to make this type of storage conservation simpler.

05: A constant may be written by placing 4 digits each in the absolute portions of the PQ, R, S, and T fields. Alternatively, one may use an 05 C code and write the 16 digits consecutively on the coding sheet, 8 digits each in the symbolic portions of the S and T fields. See Appendix (A) Figure 1, line 9.

08: This is a segment code, supplying necessary information to the compiler. The compiler will not segment a problem on its own initiative and the coder can choose a favorable point at which to terminate a segment.

The 08 code states that segment K (K in the PQ absolute field) contains words starting at the location specified in the symbolic portion of the R field; the first word of this segment to be placed in the memory over the location in the previous segment specified in the T field.

Thus everything in the memory up to the location specified by the T field in the 08 code can be saved from segment to segment. For a non-segmented problem no 08 code should be used; for a multi-segment problem there must be as many 08 codes as there are segments.

10: This code yields information as to how to start the problem. It signifies: start with segment K (in the PQ absolute field) at the location specified in the R field. The identification number of the problem is in the absolute portion of the S field and the problem is located on the tape mechanism whose tape code is in the absolute part of the T field (00XX). If this latter field is zero, tape code 09 will be used.

There must be one and only one 10 code for each problem.

11: This code is used when it is desired to read another segment into the memory and to transfer control to it. It signifies: Read segment K into the memory (K is in the PQ absolute field) and transfer control to the location specified in the R field. In addition, however, the S field must contain the location of this 11 code.

The 11 code should be used whenever the program tape is only used to read in the various segments of the problem. If the program tape is disturbed for other uses, employ the C code of 12, which is otherwise identical in form to the 11 code.

Additional C Codes:

If a problem is already on tape ready for compilation, a list of ordered deletions, insertions, or corrections may be placed on tape following the problem or on another tape mechanism, terminated by an end of file mark. Option switch 75 should be set to the transfer position during compilation and changes in the problem will be made via the following C codes:

21: Replace locations R through S by the following words. Following such a list of words must be a word with a 22 C code.

22: A sentinel terminating such a list of words.

23: Insert after location R the following words (must be followed by a word with a 22 C code).

24: Omit locations R through S.

If option switch 78 is set to the transfer position, the C codes of 21, 23, 24 will be recorded on the monitor printer.

Additional Features:

In order that all problems may be started in the same manner, so as to minimize operator error, the compiling routine inserts on the program tape, preceding the program blocks, a six-word block. This block is entered into CRT locations 0002 to 0007 by operation of the following instruction which is manually keyed into CRT location 0001:

P	Q	R	S	T
*09	94	0002	0007	0000

In the case of a nonsegmented program, the six-word block has the following form:

Loc	P	Q	R	S	T
0002	09	94	0008	1999	0000
3	--	68	--	--	0006
4	--	81	--	--	Start Line
5	--	--	--	--	--
6	--	61	--	--	0001
7	Problem Identification Number				

Note that these instructions read in the program tape, transfer control to the program, and print the identification number of the problem on the monitor printer where it may be checked. In the event of a tape check failure, the machine will be stopped.

*(09 is usually used as the tape code for the program tape unless otherwise specified (by the C code of 10 that indicates how to start the problem).)

In the case of a segmented problem, the program blocks are preceded by 2 blocks: (1) a six-word block similar to the one preceding a nonsegmented program, and (2) a 38 word block consisting of a routine common to multisegment problems.

The 6 word block is entered into CRT locations 0002 to 0007 in precisely the same fashion as indicated before and operate as follows:

Loc	P	Q	R	S	T
0002	09	94	0008	0045	0000
3	--	68	--	--	0006
4	--	81	0005	0024	0010
5	Start Loc.		0046	1999	Segment No.
6	--	61	--	--	0001
7	Problem Identification Number				

This reads in the second block, transfers control to it, and prints the identification number of the problem on the monitor printer where it may be checked. In the event of a tape check failure, the machine will be stopped.

The second block contains a routine which reads in the first program block required, and transfers control to it.

The 11 or 12 C code, which transfers control to subsequent segments, does so via this routine. The routine also includes a subroutine which rereads a block on tape if an error has been detected on first reading. If this subroutine is called for in a multisegment problem, the compiler will not insert the subroutine again into the program, but will make use of this one already present in the memory.

It must be emphasized that the coder does not code the above-mentioned preliminary blocks, but these are inserted by the compiler. Also, to start any problem, the operator always performs the same sequence of manual operations:

a. key into location 0001 this instruction

```
09   94   0002   0007   0000
```

b. with the source of instruction switch set to V, start computation.

Printing Options:

The coder has a choice of two forms of printed output during the course of compiling his problem.

a. A table of symbolic addresses versus their corresponding machine addresses. In each of the seven words printed on a line, the high order 8 digits will contain a symbolic location, the low order 4 digits the corresponding machine location.

If this output is desired, option switch 76 should be set to the transfer position, option switch 77 to the off position.

b. A list of compiler coding, the equivalent converted machine coding, the machine address of this coding and the segment in which it first appears.

The first four words on a printed line contain a line of compiler coding, that is the C, location, PQ, R, S, and T fields. The fifth word is zero. The sixth word contains the segment number in the high order 4 digits and the machine address in the low order 4 digits. (This latter is zero if the word is not entered in the final coding, e.g., C codes of 04 and 08.) The seventh word contains the converted machine word.

If this output is desired, option switch 77 should be set to the transfer position, option switch 76 to the off position.

Subroutines:

Subroutines in the library are coded in relative form in machine language, starting at any machine address.

The library of subroutines is on the same reel of tape as the compiler program, and follows it. When and if this reel is filled to capacity, allowance has been made for an additional reel of subroutines on another mechanism.

The first two words of the subroutine block are sentinel words. The eight low order digits of the first word contain the identification number of the subroutine, made up in the following way: the first 2 digits are the tape code that selects the mechanism on which the subroutine tape is placed, (tape code 10, thus far). The next four digits constitute the block number, x, $0001 \leq x \leq 9990$. The next digit is a 1 if any modification of the subroutine other than an orientation to the locations required by the problem is necessary (e.g., make use of preset parameters). Otherwise this digit is zero. The last digit is zero if the subroutine is in just one tape block; if there are $K > 1$ blocks, this last digit is K. Each block may have a maximum size of 1020 words.

The second sentinel word contains the length of the subroutine in the T field. In addition, the PQ field contains the machine address at which the first line of the subroutine was coded.

Then follows any "preliminary" coding which may be there for purposes of modifying, before incorporation into the problem, the n words of the subroutine which follow this coding. Any such "preliminary" coding is written in relative codes starting at location 8065. Easy access to routines useful to "preliminary" coding is provided by the compiler, such as read, write, and orient to absolute coding routines.

Following this are n "guide" words, each of which is associated with the corresponding subroutine word and serve to indicate how the subroutine word should be altered to be properly oriented.

Finally there is a terminating sentinel word consisting of 16 digits of 5's.

The above details concerning the form of the subroutine in the library file are of concern only to the author of a subroutine. A coder who wishes to use a library subroutine in his problem need enter on his coding sheet only the limited information indicated by the subroutine directory. The compiler automatically assembles the subroutine on the basis of the information obtained both from the coder and from the sentinel words and "preliminary" coding associated with the subroutine in the library.

Error Stops:

Coding errors that the compiler detects are signified by a coded machine stop and a line of printing on the monitor printer. The first word on the printed line will contain one of the error codes listed below. The second word contains, where applicable, the segment number involved.

1: This segment exceeds the memory capacity. Word 4 indicates the number of words in this segment.

2: More than 1300 symbolic locations used. (Word 2 indicated the segment reached.)

3: Reference is made to a word for which there is no location. Word 3 contains this address.

4: There are more 08 C codes than there are segments.

5: The R field of an 08 code has a nonzero absolute portion.

6: There is more than one 08 code for a given segment.

7: There is more than one 10 code for the problem.

8: Two words have been assigned the same symbolic location.

9: There is no 10 C code for the problem.

11: There is one and only one 08 code. There should be either no 08 codes (for a nonsegmented problem) or at least two 08 codes (for a segmented problem).

12: There is no 08 code for this segment.

13: A subroutine has been requested which is not in the subroutine library. The subroutine number is printed in word 3.

14: The input for a subroutine is not in the proper form as indicated by the subroutine directory. The subroutine number is printed in word 3.

15: Not used.

16: There are more than 200 different subroutines required for this segment.

CONCLUSION

A compiler of this type is no panacea for an installation's personnel problems. The analysis and general programming must still be performed as must the coding, though the latter can be done far more rapidly and efficiently via the compiler. In this way better use is made of the coder's time.

It is felt that some of the unique features of the NORC compiler, such as the ordering of the symbolic coding and the organizational flexibility permitted the coder, are very advantageous properties. In conjunction with the other characteristics of the compiler, a very efficient program can be coded in a relatively short time.

At the time of this report, only a few problems have been coded and compiled in the manner herein described. Experience with these problems has suggested some minor changes which have been made and further experience will probably indicate more. The machine time required for compilation will rarely exceed two minutes, although the choice of printing option b. can lengthen this time considerably.

REFERENCES

(a) Report to the Association for Computing Machinery, "First Glossary of Programming Terminology". Committee on Nomenclature, June 1954.

(b) Symposium on Automatic Programming for Digital Computers. Office of Naval Research, 13-14 May 1954.

(c) Proceedings of the Association for Computing Machinery, Toronto Meeting, 8-10 September 1952.

(d) A survey of Automatic Coding Techniques for Digital Computers. John L. Jones, (Master's Thesis), May 1954.

(e) The New York University Compiler System. Roy Goldfinger, 10 February 1954.

(f) The A-2 Compiler System Operations Manual. Remington Rand Inc., 15 November 1953.

(g) A Program for Translation of Mathematical Equations for Whirlwind 1. J. H. Laning, Jr., and N. Zierler. Engineering Memorandum E-364, January 1954.

(h) Digital Computers - Advanced Coding Techniques. Massachusetts Institute of Technology, Summer Session 1954.

(i) The Preparation of Programs for an Electronic Digital Computer. Wilkes, Wheeler and Gill. Addison-Wesley, 1951.

In addition there are a great number of reports on automatic coding techniques from the many installations using them - universities, aircraft companies, government centers, etc.

APPENDIX A

EXAMPLE OF COMPILER CODING AND CORRESPONDING MACHINE CODING

Only a long and complicated problem could illustrate all the advantages of the compiler, but is unlikely that anyone would read through such a problem. Instead the following artificial example only attempts to illustrate the use of some of the compiler codes.

The coding consists of one segment of a problem which reads in 3 numbers, x_i, from tape and computes $\sqrt{x_i + \pi}$, writing the 3-square roots on tape, The next segment is read in and control is transferred to it. The $\sqrt{x_i + \pi}$ remain in the memory for use by the next segment.

Figure 1 consists of the compiler coding, and Figure 2, the machine coding which the compiler would produce.

EXPRESSION	C	LOCATION	P	Q	R	S	T
START: READ IN BLOCK OF 3 NUMBERS		120·0	01	94	119·0	0002 119·0	0007
SETS MODIFIER M4				51	119·0		
ADD $X_i + \pi$			50	20	120·5	4000	0001 120·5
TRANSFER TO SQUARE ROOT SUBROUTINE	01	120·2	08	60	120·2		10·0036
INPUT AND OUTPUT LOCATIONS FOR SQUARE ROOT			120·5 00	 01			4000
ADD 1 TO MODIFIER SENSE IF LOOP IS TERMINATED				58	0001	0003 119·0	0002 120·0
WRITES 3 NUMBERS ON TAPE			02	90	119·0	0002 119·0	
READS IN NEXT SEGMENT AND TRANSFERS CONTROL TO IT	11		00	06	130·0	0005 120·2	
π	05	120·5				00031415	92653590
$X_i + \pi$							
	03	119·0	00	02			
	10		00	01	120·0	prob. no.	

FIGURE 1

THIRD BLOCK - SEGMENT 0001

EXPLANATION	LOCATION	P	Q	R	S	T
THESE CONTAIN THE Xi,	0046					
THEN $\sqrt{X_i + \pi}$	0047					
	0048					
START: READ IN THE 3 X_i	0049	01	94	0046	0048	0007
SET M4	0050		51	0046		
ADD $X_i + \pi$	0051	50	20	0058	4000	0059
TRANSFER TO SQ· RT· SUBROUTINE	0052	08	60	0052		0060
INPUT AND OUTPUT LOCATIONS FOR SQUARE ROOT	0053	00	60			4000
ADD ONE TO MODIFIER SENSE IF LOOP IS TERMINATED	0054		58	0001	0049	0051
WRITE 3 $\sqrt{X_i + \pi}$ ON TAPE	0055	02	90	0046	0048	0001
TRANSFER TO ROUTINE WHICH READS NEXT SEG· INTO MEMORY, STARTING	0056		60	0057	0024	0010
AT LOCATION 0049· (SAVING THE $\sqrt{X_i + \pi}$ IN 0046 - 0048)	0057	00	49	0049	1999	0006
π	0058	00	03	1415	9265	3590
$X_i + \pi$	0059					
	0060	00	40	0000	0085	0084
	0061	92	60			
THE SQUARE ROOT						
SUBROUTINE						
	0085	00	59	9999	9140	0002

FIGURE 2

APPENDIX B

GENERAL DESCRIPTION OF THE NAVAL ORDNANCE RESEARCH CALCULATOR (NORC)

General

The NORC carries out its calculations in decimal arithmetic on decimal numbers of 13 (or less) significant digits, under control of a stored program of the three-address type with automatic address modification. The instructions and the numbers on which the calculator operates are stored in a cathode ray tube storage unit.

Only two kinds of storage for both numbers and instructions are used in this calculator. The main storage element is the cathode ray tube storage unit holding 2,000 sixteen-decimal digit words, any word of which is accessible in an 8 microsecond calculator cycle. The cathode ray tube storage unit is supplemented by a group of eight high-speed magnetic tape reading and recording units with an aggregate storage capacity in excess of three million words. These tape units can be used interchangeably for input-output and intermediate storage of information including instructions and intermediate results.

Numbers and instructions are entered into the calculator on magnetic tapes which can be read into the cathode ray tube storage unit at an effective rate of approximately 2,500 words per second. Results of the computations of the computer are recorded on magnetic tape using any of the tape units.

Results of a computation can also be recorded directly in printed form on either one of two line-at-a-time page printers capable of recording 17.5 words per second (1,050 words a minute).

The arithmetic unit of the calculator operates serially digit by digit at a speed of one microsecond per digit. Actual addition and subtraction of 13-digit numbers is performed in the time taken to read the two numbers - i.e., 13 microseconds. Multiplication is performed with the aid of a serial form of multiplication table in approximately twice the time taken to read the 13-digit factors - i.e., 30 microseconds. Division requires approximately 250 microseconds. The arithmetic operations are checked by an independent modulo 9 computer which operates in parallel with the arithmetic unit and checks each arithmetic operation. The arithmetic unit requires only two registers which are composed of microsecond delay units which have proved extremely reliable at these high pulse rates.

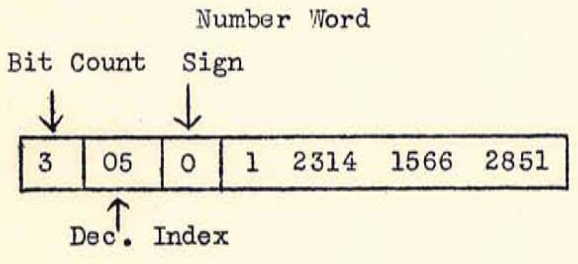
The correct operation of the magnetic tape input-output system and the cathode ray tube storage system is continuously checked by a bit count modulo 4 which is computed and recorded for each word on the magnetic tapes when they are originally prepared and for each new or modified word originating in the arithmetic unit. This bit count is compared with the actual number of bits in each word each time a word is read from a tape, read out of cathode ray tube storage, regenerated in cathode ray tube storage, or generated in the arithmetic unit. Bit counts are recorded on all tapes produced by the calculator, and the bit count is carried into the printers and compared with a bit count of each of the words printed by the printer. Any failure of the computed bit count to agree with the indicated bit count associated with a word of stored information stops the calculator, or in the case of tape failure can cause the calculator to perform a special series of instructions designed to correct it.

The modulo-9 arithmetic check and the modulo-4 bit count on storage and transmission of information, supplemented by various other checks, assure prompt indication of malfunction of the computer.

The card-to-tape-to-card machine, an auxiliary device, prepares magnetic tape suitable for input to the electronic calculator from conventionally punched IBM cards. It also prepares conventionally punched IBM cards from the output tapes produced by the calculator. The card-to-tape-to-card machine reads or punches four 16-digit words per card at the rate of approximately 100 cards per minute. In card-to-tape operation it produces the bit count for each input word, during tape-to-card operation it verifies the bit count of each output word.

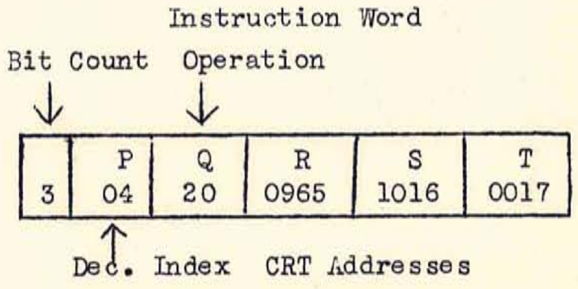
The organization of the number words and instruction words for this calculator is as indicated below. A number word consists of 17 digits; the bit count, a 2-digit decimal index, a 1-digit algebraic sign, and a 13-digit decimal number, in which the decimal point is located to the right of the highest digit, i.e., beteen positions 13 and 12. The decimal index represents the power of 10 by which the number in the word must be multiplied to give the number its true magnitude. The index can have values from 70 to 99, 0 to 30, corresponding to powers of 10 from -30 to +30. The algebraic sign plus or minus is represented by 0 or 1 respectively. The indicated bit count is the 3's complement of the bit count modulo-4 of the 16 other digits of the word.

An instruction word consists of the indicated bit count, two digits specifying the decimal index procedure to be applied to the operation, two digits to specify the operation to be performed, and three fields of four digits each normally specifying the CRT addresses of the numbers involved in the particular operation. Normally the first two address fields R and S designate the location of the operands, and the T field designates the location in which the result is to be stored.



$\equiv 10^5 \times (+\ 1.231415662851)$

$\equiv 123141.5662851$



Add (20) the numbers in storage locations 0965 and 1016.
Shift the result, if possible without overflow, so that it has a decimal index of 4 and store the result in location 0017.

The calculator will execute an average of 14,000 instructions of this type (other than input-output) per second. The execution of each of these instructions is equivalent to the execution of a number of conventional single-address fixed-point instructions; the calculator's performance is equivalent to the execution of 25,000 to 400,000 such instructions per second.

Input

Information is recorded on the magnetic tapes in blocks containing any desired number of words from one to 2,000, the size of the cathode ray tube memory. Each word of a block contains the bit count and 16 digits of information. The first and last word of each block contains a 4-digit number in the low order four digits, designating the block number used for identifying each block in searching and reading in to CRT storage.

The tape units read the information on the magnetic tapes via four parallel tracks representing binary coded decimal values of the digits recorded. The information is read at a digit spacing of 510 bits to the inch on each track and at a speed of 140 inches per second in both reading and writing.

Each tape block is separated from the next by a blank area of tape sufficient to start and stop the tape without loss of information. Each tape reel has a capacity of 1400 feet providing storage for from 350,000 to 450,000 words depending on the average block length.

The tape can be read in either direction. It is written or blocks are deleted only when moving in the forward direction. Rewinding is performed with the tape moving backwards. After initiating rewinding, the calculator can proceed with its computing or use other tapes while the tape unit completes its rewind.

Information is read from the tapes into cathode ray tube storage by means of instruction 94 or 96 of the form xx 94 R S T where xx designates the tape unit which is read and block T is read into CRT storage locations R through S. To read block 17 of tape 1 into CRT memory positions 500-599, this instruction would be recorded as 01 94 0500 0599 0017.

Cathode Ray Tube Storage

The cathode ray tube storage consists of 66 cathode ray tube storage units. Sixty-four of the CRT units carry the sixteen digits of information in the word, either a number or an instruction, four CRT units being used to represent each decimal digit in the binary coded decimal system. The two additional units contain the 3's complement of the modulo-4 bit count (the indicated bit count) used for checking the information stored each time there is a read-out from CRT storage or a regeneration of a word in the CRT storage.

In addition to the bit count for each word, a continuous odd-even count is kept for each of the 66 cathode ray tube units. This count is changed each time a number of bits in a given cathode ray tube unit changes. If, during the operation of the calculator, the word check indicates an error in a word in CRT, the odd-even count can be used to identify the particular column of the CRT storage in which the error has occurred, pinpointing the failure to a specific spot in a specific tube.

The normal addresses of the locations in the cathode ray tube storage unit are 0001 through 1999. These addresses increased by 4,000, 6,000 or 8,000 designate which of three address modifier numbers are to be added to the address before using it.

Arithmetic Unit

The arithmetic unit consists of two 16-digit universal registers designated Register 1 and Register 2, which receive numbers from and transmit numbers to the cathode ray tube storage, and which can also be stepped either right or left a digit at a time in one microsecond intervals.

Paralleling the arithmetic operations, the corresponding operation is performed on the modulo-9 value of the operands, and the modulo-9 value of the result of the arithmetic operation is checked with the modulo-9 computed result to determine correctness of the arithmetic operation.

The arithmetic unit is arranged to operate in either floating or specified decimal mode of operation depending upon the code in the index field of the instruction word. Any number in the range from 40 through 59 (normally 50) designates floating index operation of the arithmetic unit, and in addition and subtraction the two numbers are shifted to cause their indexes to correspond before the addition or subtraction is performed. Any high order zeros in the result are removed and the index of the result decreased accordingly. If the sum results in an additional high order digit, the result is shifted right one place and the index increased. For specified index operation, the desired result index is entered in the P field and can range from -30 through +30, the 100's complement being used for negative powers. The calculation is performed like a floating index operation with the result shifted to conform with the specified index. If an overflow results from any specified index operation, either the result can be shifted right and the index adjusted accordingly to retain the complete result, or the overflow digits can be dropped and the machine caused to signal the overflow condition as designated by the instruction code.

Arithmetic operations are rounded as designated by the various instruction codes by forcing a 1 in the lowest order digit of the retained portion of the result unless the retained portion or the discarded low order portion of the result is zero. Forcing a 1 in the lowest order digit increases its value if digit is even, but does not change it if the digit is odd.

A complete arithmetic operation on this calculator includes the securing of the instruction from the cathode ray tube storage, modification of the addresses of the instruction as indicated, the securing of the operand or operands from CRT storage, the inspection of the decimal indexes of the two operands, and the performance of the arithmetic operation, followed by a shifting of the result as indicated by the decimal index of the instruction, and the transfer of the result to a designated location in cathode ray tube storage. The results of one arithmetic operation are normally retained in register 1 and register 2, and 0000 in either or both of the operand fields in the next instruction will use the previous result as the corresponding operand, saving two or three references to cathode ray tube storage. Location 0000 in the result field retains the result in register storage only.

Instruction System

The instruction system of this calculator is based on the consecutive execution of three-address instructions stored in the cathode ray tube memory. As indicated earlier the instruction word contains 16 digits of information divided into five fields, P and Q being two digits each, and R, S and T four digits each. Field P normally specifies the decimal index procedure for arithmetic operations, field Q specifies the operation to be performed, fields R and S normally specify the location of the numbers to be used in the operation, and field T the location in which the result is to be stored.

The addresses in fields R, S and T are modified in any instruction where the address is greater than 3,999. Any one of three modifiers, designated M4, M6 and M8 will be automatically added to the addresses of 4,000 and above, M4 being added for addresses from 4,000 to 5,999, and M6 being added to addresses 6,000 to 7,999, and M8 being added to addresses 8,000 to 9,999. The resulting address modulo-2,000 after the addition of the address modifier is used as the actual CRT address in the instruction. Automatic address modification applies to all instruction except operations 50 through 57 and 90 through 98, which cover address modifier operations and tape operations.

Floating arithmetic is designated by a number in the range from 40 to 59 in the P field. This procedure causes automatic shifting of the result to remove high order zeros without overflow of any non-zero digit on the left. A specified or fixed index procedure is indicated by a number in the range from 70 through 30 in the P field, corresponding to powers of 10 from -30 to +30, the 100's complement being used for negative powers.

There are two sets of instructions covering the ordinary arithmetic operations of addition, subtraction, multiplication and division designated as standard arithmetic, codes 20 through 28, and special arithmetic codes 30 through 39. When using standard arithmetic instructions, if the execution of the specified index in the instruction would result in overflow digit or digits on the left of the result, the result is treated as a floated operation - i.e., the overflow is prevented, the 13 high order digits are retained in the result, and the index adjusted accordingly. The adjusted index indicator is turned on permitting a modification of the program to be executed to take in account the adjusted index condition.

In special arithmetic, when specified index procedure is indicated, and overflow occurs on the left, the overflow condition is indicated and any digits which have overflowed are lost, and the overflow indication may be used to adjust the calculation by program or stop the calculator.

The instructions available in the machine are described below. In specifying the operation performed by each instruction, the contents of the cathode ray tube storage at address X is designated as X'.

NORC CODES (Q FIELD)

Primed Letters (R', S') indicate contents of R, S.
Transfer: Shift R' P places into S, go to T for next instruction.

Overflow on left not allowed.	Overflow on left permitted.	
Rounded	**Un-Rounded**	**Arithmetic Instructions**
for 39<P<60 float result		
20	30	R' + S' to T
21	31	- (R' + S')
22	32	R' - S'
23	33	- (R' - S')
24	34	R' x S'
25	35	- (R' x S')
26	36	R' / S'
27	37	- (R' / S')
28	38	R' - S'
29	39	Truncating Transfer R' with S' to T
	40	Meta R' + S'
	41	Meta R' - S'

Modifier Instructions

50 no clearing	54 clear M8
51 clear M4	55 clear M4 and M8
52 clear M6	56 clear M6 and M8
53 clear M4 and M 6	57 clear M4, M6 and M8

The 50 to 57 codes add R, S, T to M4 M6, M8 respectively, after any indicated resets.

Print Instructions

80	Print Interlock & Transfer
81	Print 1 and Transfer
82	Print 2 and Transfer
83	Print 1 with special function and transfer
84	Print 2 with special function and transfer

Logical Instructions

58	M'4 + R to M4. If unequal to S go to T, otherwise continue
60	Transfer
61	Transfer and stop
62	Transfer with rounding
63	Program transfer on algebraic sign, +, 0, -.
64	Transfer on overflow indication
65	Transfer on adjusted index indication
66	Transfer on zero result
67	Transfer on end of file
68	Transfer on tape check failure
69	Transfer on printer ready indication
70	Program transfer on zero meta difference of R' and S'
71	Program transfer and stop on zero meta difference of R' and S'
72	Program transfer on non-zero meta difference of R' and S'
73	Program transfer and stop on non-zero meta difference of R' and S'
74	Transfer on condition switch 74
75	Transfer on condition switch 75
76	Transfer on condition switch 76
77	Transfer on condition switch 77
78	Transfer on condition switch 78
79	Transfer on condition switch 79

Tape	Instructions on TAPE P, words R to S, block T
90	Write
91	Write output (space between each 100 words)
92	Delete
93	Delete output
94	Read Forward
95	Read Backward
96	Verify Forward
97	Verify Backward
98	Rewind

APPENDIX C

DISTRIBUTION

Bureau of Ordnance:	
Ad3	1
Re3d	1
Armed Services Technical Information Agency Document Service Center Knott Building Dayton 2, Ohio	5
Commanding General Aberdeen Proving Ground Aberdeen, Maryland Attn: Technical Information Section Development and Proof Services	2
Commander, Operational Development Force U. S. Atlantic Fleet, U. S. Naval Base Norfolk 11, Virginia	1
Naval Ordnance Laboratory White Oak, Silver Spring 19, Maryland	5
Director Naval Research Laboratory Anacostia 20, D. C.	2
Solid Propellant Information Agency Applied Physics Laboratory Johns Hopkins University Silver Spring, Maryland	3
Director David Taylor Model Basin Washington 37, D. C.	2
Naval Ordnance Test Station Inyokern, California	3
Chief of Naval Research Washington 25, D. C. Attn: Mathematical Section	1

DISTRIBUTION (Continued)

Chief of Bureau of Aeronautics Department of the Navy Washington 25, D. C. Attn: Section RS-3	1
Chief of Bureau of Ships Department of the Navy Washington 25, D. C.	1
Comptroller, U. S. Air Force Attn: Dr. George B. Dantzig Mathematical Consultant 3E1048 Pentagon Building Washington, D. C.	1
The Bureau of the Census Washington 25, D. C.	1
National Bureau of Standards Division 11 National Applied Mathematics Laboratories Washington, D. C.	2
Harvard Computation Laboratory Harvard University Cambridge, Mass.	1
Head, Postgraduate School Monterey, California	1
Superintendent U. S. Naval Academy Annapolis, Maryland Attn: Department of Mathematics	1
Atomic Energy Commission 1901 Constitution Avenue Washington 25, D. C.	2
National Advisory Committee for Aeronautics Langley Memorial Aeronautical Laboratory Langley Field, Virginia	1

DISTRIBUTION (Continued)

Atomic Energy Commission Technical Information Service P. O. Box 62 Oak Ridge, Tennessee	1
Los Alamos Scientific Laboratory Box 1663 Los Alamos, New Mexico Attn: Nicolas Metropolis Paul Stein	3
Commanding General Picatinny Arsenal Dover, New Jersey Attn: Technical Division	3
Commanding Officer Frankford Arsenal Philadelphia 37, Penna. Attn: Ernest C. Casale Head Mathematics Section Pitman Dunn Laboratory	2
Commanding General White Sands Proving Grounds Las Cruces, New Mexico Attn: Flight Determination Laboratory John Titus	3
Operation Research Office 7100 Connecticut Avenue Chevy Chase, Maryland Washington 15, D. C. Attn: Joseph O. Harrison, Jr. W. E. Cushen	2
University of Michigan Willow Run Research Center Willow Run Airport Ypsilanti, Michigan	2

DISTRIBUTION (Continued)

Massachusetts Institute of Technology 211 Massachusetts Avenue Cambridge 39, Massachusetts Attn: Digital Computer Laboratory	5
University of Pennsylvania Moore School of Electrical Engineering Philadelphia, Pennsylvania	2
Raytheon Manufacturing Company Equipment Engineering Division 148 California Street Newton 58, Massachusetts Attn: Dr. H. L. Thomas, Documents Section	1
Dr. Richard Block	1
General Electric Company 1 River Road Schenectady 5, New York Attn: H. R. Koenig	1
W. B. Jordan	1
H. Poritsky	1
Engineering Research Associates 1902 W. Minnehaha Avenue St. Paul, Minnesota	1
Remington Rand Inc. Eckert-Mauchly Division 2300 W. Allegheny Ave. Philadelphia 29, Penna. Attn: Herbert F. Mitchell, Jr.	2
Armour Research Foundation Chicago 13, Illinois	1
Cornell Aeronautical Laboratory, Inc. 4455 Genesee Street Buffalo 21, New York	1

DISTRIBUTION (Continued)

University of Illinois Urbana, Illinois Attn: A. H. Taub, Dept. of Math.	1
J. P. Nash, Electronic Digital Computer Project	3
University of Maryland Department of Mathematics College Park, Maryland Attn: Monroe H. Martin	1
David M. Young	1
Charles Warlick	1
IBM Corporation Technical Computing Bureau 1111 Connecticut Avenue, N. W. Washington 6, D. C. Attn: George Petrie	1
IBM Corporation Engineering Laboratory Poughkeepsie, New York Attn: E. R. Lancaster Advanced Engineering Education Department	1
Sandia Corporation Sandia Base P. O. Box 5800 Albuquerque, New Mexico Attn: Library	1
Watson Scientific Computing Laboratory 612 W. 116th Street New York 27, New York Attn: Dr. L. H. Thomas	1
Dr. W. Eckert	1
Radio Corporation of America Programming and Analysis Unit Camden, New Jersey Attn: A. S. Kranzley	1
M. Macchia	1

DISTRIBUTION (Continued)

Rand Corporation 1700 Main Street Santa Monica, Calif. Attn: John C. Shaw	2
Knolls Atomic Power Laboratory Schenectady, New York Attn: Dr. Richard H. Stark	2
Westinghouse Electric Corporation Physics Dept. P. O. Box 1468 Pittsburgh, Pa. Attn: George Crane	2
Rich Electronic Computing Center Georgia Institute of Technology Atlanta, Georgia Attn: Dr. E. K. Ritter	1
University of North Carolina Dept. Mathematical Statistics Chapel Hill, N. C. Attn: Dr. H. Hotelling	1
IBM Corporation 590 Madison Avenue New York 22, N. Y. Attn: J. C. McPherson	1
Douglas Aircraft Co., Inc. El Segundo, Calif.	1
Douglas Aircraft Co., Inc. Santa Monica, Calif.	1
General Motors Corp. Detroit, Mich.	1
Boeing Airplane Company Seattle, Washington	1

DISTRIBUTION (Continued)

Lockhead Aircraft Corp. Van Nuys, Calif.	1
North American Aviation, Inc. Los Angeles, Calif.	1
United Aircraft Corp. East Hartford, Conn.	1
Institute for Advanced Study Princeton, New Jersey Attn: Prof. J. von Neumann	1
Dr. H. H. Goldstine	1
University of California 942 Hillsdale Ave. Berkeley, Calif. Attn: D. H. Lehmer	1
Commander Wright Air Development Center Wright-Patterson Air Force Base, Ohio Attn: WCRRN-4	2
Commanding Officer Naval Air Missile Test Center Point Mugu, Calif.	1
University of Delaware Newark, Delaware Attn: G. C. Webber	1
R. Remage	1
Local:	
OKCP	100
OK	1
CR	1
File	1

NPG 1374

Naval Proving Ground, Dahlgren, Virginia

THE NAVAL ORDNANCE RESEARCH CALCULATOR (NORC) COMPILER, by Karl Kozarsky. 17 May 1955. 16p. append. A-C. (NPG Report No. 1374. Foundational Research Project K-11011-9).

This report describes the current status of the Naval Ordnance Research Calculator (NORC) Compiler. This routine is an automatic coding technique designed to minimize the time spent on the coding of problems for solution on NORC. The compiler converts unordered symbolic or relative codes to ordered machine (absolute) codes, generates and assembles subroutines from a library file, provides a variety of other services for the coder, and writes the final coding on tape, ready to run on NORC.

Use of the compiler enables a considerable amount of tedious coding time to be saved. The time required to compile a program is usually two minutes or less of NORC machine time.

I. Digital computers - Coding

I. NORC

II. Kozarsky, K.

NPG 1374

Naval Proving Ground, Dahlgren, Virginia

THE NAVAL ORDNANCE RESEARCH CALCULATOR (NORC) COMPILER, by Karl Kozarsky. 17 May 1955. 16p. append. A-C. (NPG Report No. 1374. Foundational Research Project K-11011-9).

This report describes the current status of the Naval Ordnance Research Calculator (NORC) Compiler. This routine is an automatic coding technique designed to minimize the time spent on the coding of problems for solution on NORC. The compiler converts unordered symbolic or relative codes to ordered machine (absolute) codes, generates and assembles subroutines from a library file, provides a variety of other services for the coder, and writes the final coding on tape, ready to run on NORC.

Use of the compiler enables a considerable amount of tedious coding time to be saved. The time required to compile a program is usually two minutes or less of NORC machine time.

I. Digital computers - Coding

I. NORC

II. Kozarsky, K.

NPG 1374

Naval Proving Ground, Dahlgren, Virginia

THE NAVAL ORDNANCE RESEARCH CALCULATOR (NORC) COMPILER, by Karl Kozarsky. 17 May 1955. 16p. append. A-C. (NPG Report No. 1374. Foundational Research Project K-11011-9).

This report describes the current status of the Naval Ordnance Research Calculator (NORC) Compiler. This routine is an automatic coding technique designed to minimize the time spent on the coding of problems for solution on NORC. The compiler converts unordered symbolic or relative codes to ordered machine (absolute) codes, generates and assembles subroutines from a library file, provides a variety of other services for the coder, and writes the final coding on tape, ready to run on NORC.

Use of the compiler enables a considerable amount of tedious coding time to be saved. The time required to compile a program is usually two minutes or less of NORC machine time.

I. Digital computers - Coding

I. NORC

II. Kozarsky, K.

NPG 1374

Naval Proving Ground, Dahlgren, Virginia

THE NAVAL ORDNANCE RESEARCH CALCULATOR (NORC) COMPILER, by Karl Kozarsky. 17 May 1955. 16p. append. A-C. (NPG Report No. 1374. Foundational Research Project K-11011-9).

This report describes the current status of the Naval Ordnance Research Calculator (NORC) Compiler. This routine is an automatic coding technique designed to minimize the time spent on the coding of problems for solution on NORC. The compiler converts unordered symbolic or relative codes to ordered machine (absolute) codes, generates and assembles subroutines from a library file, provides a variety of other services for the coder, and writes the final coding on tape, ready to run on NORC.

Use of the compiler enables a considerable amount of tedious coding time to be saved. The time required to compile a program is usually two minutes or less of NORC machine time.

I. Digital computers - Coding

I. NORC

II. Kozarsky, K.

NPG 1374

Naval Proving Ground, Dahlgren, Virginia

THE NAVAL ORDNANCE RESEARCH CALCULATOR (NORC) COMPILER, by Karl Kozarsky. 17 May 1955. 16p. append. A-C. (NPG Report No. 1374. Foundational Research Project K-11011-9).

This report describes the current status of the Naval Ordnance Research Calculator (NORC) Compiler. This routine is an automatic coding technique designed to minimize the time spent on the coding of problems for solution on NORC. The compiler converts unordered symbolic or relative codes to ordered machine (absolute) codes, generates and assembles subroutines from a library file, provides a variety of other services for the coder, and writes the final coding on tape, ready to run on NORC.

Use of the compiler enables a considerable amount of tedious coding time to be saved. The time required to compile a program is usually two minutes or less of NORC machine time.

I. Digital computers - Coding

I. NORC

II. Kozarsky, K.

NPG 1374

Naval Proving Ground, Dahlgren, Virginia

THE NAVAL ORDNANCE RESEARCH CALCULATOR (NORC) COMPILER, by Karl Kozarsky. 17 May 1955. 16p. append. A-C. (NPG Report No. 1374. Foundational Research Project K-11011-9).

This report describes the current status of the Naval Ordnance Research Calculator (NORC) Compiler. This routine is an automatic coding technique designed to minimize the time spent on the coding of problems for solution on NORC. The compiler converts unordered symbolic or relative codes to ordered machine (absolute) codes, generates and assembles subroutines from a library file, provides a variety of other services for the coder, and writes the final coding on tape, ready to run on NORC.

Use of the compiler enables a considerable amount of tedious coding time to be saved. The time required to compile a program is usually two minutes or less of NORC machine time.

I. Digital computers - Coding

I. NORC

II. Kozarsky, K.

NPG 1374

Naval Proving Ground, Dahlgren, Virginia

THE NAVAL ORDNANCE RESEARCH CALCULATOR (NORC) COMPILER, by Karl Kozarsky. 17 May 1955. 16p. append. A-C. (NPG Report No. 1374. Foundational Research Project K-11011-9).

This report describes the current status of the Naval Ordnance Research Calculator (NORC) Compiler. This routine is an automatic coding technique designed to minimize the time spent on the coding of problems for solution on NORC. The compiler converts unordered symbolic or relative codes to ordered machine (absolute) codes, generates and assembles subroutines from a library file, provides a variety of other services for the coder, and writes the final coding on tape, ready to run on NORC.

Use of the compiler enables a considerable amount of tedious coding time to be saved. The time required to compile a program is usually two minutes or less of NORC machine time.

I. Digital computers - Coding

I. NORC

II. Kozarsky, K.

NPG 1374

Naval Proving Ground, Dahlgren, Virginia

THE NAVAL ORDNANCE RESEARCH CALCULATOR (NORC) COMPILER, by Karl Kozarsky. 17 May 1955. 16p. append. A-C. (NPG Report No. 1374. Foundational Research Project K-11011-9).

This report describes the current status of the Naval Ordnance Research Calculator (NORC) Compiler. This routine is an automatic coding technique designed to minimize the time spent on the coding of problems for solution on NORC. The compiler converts unordered symbolic or relative codes to ordered machine (absolute) codes, generates and assembles subroutines from a library file, provides a variety of other services for the coder, and writes the final coding on tape, ready to run on NORC.

Use of the compiler enables a considerable amount of tedious coding time to be saved. The time required to compile a program is usually two minutes or less of NORC machine time.

I. Digital computers - Coding

I. NORC

II. Kozarsky, K.