DEVELOPMENTS IN COMPILING TECHNIQUES TO 31 DECEMBER 1953

In any business or industry a large quantity of information exists on paper. This information in random, unprocessed form is of little or no value to management as a basis for decision-making. When information is organized and processed, it becomes "intelligence". In many cases, such intelligence is useful only if it is available rapidly. Many statistical studies and tabulations now known to be useful to management are not made, because they cannot be accomplished with sufficient speed to supply the intelligence before a decision must be made. Such information processing falls into the field of application of the electronic digital computers. It imposes quite stringent conditions on computer operations since the elapsed time between the submission of raw data and the delivery of results must be reduced to a minimum. Many such studies are made but once in the particular form desired, and they fall in the category of "one-time" problems for which programs are not immediately available.

Rarely will any computer installation be able to defend itself from the pleas for computation of the engineers, production managers, and actuaries associated with industry. The computing center will frequently find it advisable to fulfill such needs. Here again, the "one-time" problem will appear. It can only be economically executed if programming time and costs can be reduced to a minimum. Any installation of a service bureau nature will, of economic necessity, be required to reduce programming time and costs. Universal, automatic programming is necessary. Time and effort must never be devoted to specific programs for "one-time" use. A general program will not only handle specific problems but will be available for all variants likely to arise in the future. For example, if an application requires the prediction of Philadelphia Sales Tax collections, it can be so coded as to predict Sales Tax collections and be applicable in any other installation requiring the calculation of a similar prediction.

The compilers are so designed that they are general, and can also handle base problems and the variations, derivations, and changes arising during the life of a problem. The investment in programming of a commercial base problem may easily run beyond fifty man-years of effort by high-calibre trained personnel. A change in government regulations, city ordnances, union contracts, or requirements of company management, even a simple change, such as income tax exemptions, may invalidate not only parts of runs, but whole runs in such a base problem. Usually, by the nature of such changes these runs must be corrected or reprogrammed on extremely short notice. With the problem set up by compiler techniques, it is possible to instruct the computer to make the required alterations and changes at a moments notice. Thus the compiler permits changes to be made, which are automatically checked, at computer speed in a matter of minutes, and with complete elimination of the hazards of human error.

A general compiler, using the computer to generate its own self-checked programs, produces the following results,

1. reduction of staff with resultant reduction of costs,
2. increased speed of problem preparation supplying answers when needed,
3. problems prepared at speeds comparable with their execution,
4. checked routines produced by checked computer eliminating the human error factor.

These facts are being increasingly recognized throughout the computer industry. The existing UNIVAC installations (Census, Air Comptroller, Army Map, Livermore, NYU and BuShips) all have automatic coding projects and are basing their future plans on an increasing use of these techniques. Interpretive routines have been developed at many installations notably Whirlwind, MIDAC, ILLIAC, SEAC, and IBM-701 Speed Code. Our automatic programming project was originated in the fall of 1951. The effectiveness of the project can be validated by the fact that the organizations mentioned above have since started similar projects as ours has become known.

[illegible] main classes of routines are now in use in the field for this purpose: interpretive routines and compiling routines. Both types of routines depend upon the use of pseudo-codes and subroutine libraries. (A routine may be defined as an explicit set of instructions, coded in computer language and arranged in proper sequence to instruct the computer to perform a desired operation or series of operations such as the solution of a problem or the rearrangement of data. Hence a subroutine is defined as a set of instructions necessary to carry out a well-defined mathematical or logical operation - a subunit of a routine, usually stored in relative coding. A pseudo-code is an arbitrary instruction code, chosen for the convenience of the user, independent of the hardware of the computer, which must be transformed into the computer code in order to control the computer.)

In summary, the use of compiling routine techniques incorporating the application of pseudo-codes, permits the programmer in any installation to prepare problems in his own language, in his own terms, based on his own methods. The compiler then takes over the job of making an accurate translation to computer code.

When a problem is to be run using automatic techniques it is succinctly and efficiently defined in terms of a pseudo-code. If the problem is submitted to an interpretive routine or interpreter, the computation proceeds as follows:

1. A "word" of the pseudo-code is interpreted: i.e., the interpretive routine "extracts" the necessary information from the pseudo-code and transfers operands and control to the appropriate subroutine. The subroutine is either (a) available in storage and the transfer of control is accomplished by RU orders, or (b) the interpretive routine locates the subroutine on tape, reads it into storage and transfers operands and controls.
2. The subroutine executes the operation.
3. The result is transferred to working storage and control returned to the interpretive routine in order to process the next word of the pseudo-code.

It should be noted that steps 1 and 2 must be executed every time the operation is carried out; i.e., the interpretive process must be repeated for every iteration of the required operation. No permanent record of the operations, no "running tape" is produced.

The IBM interpretive routine is known as Speed Code I. A manual was issued 19 June 1953 by the IBM Scientific Computing Service. The first paragraph of the Introduction states:

> "The IBM-701 Speedcoding System was designed to minimize the amount of time spent in problem preparation. It is applicable to small computing problems and to many large computing problems. A description of Speed-Coding is hereby made available in an informal fashion, from the standpoint of printing and distribution, because programming for the IBM Electronic Data-Processing Machines, Type 701 and Associated Equipment, is developing rapidly and it is advantageous to make changes in the system easily and informally.
>
> . . . Once the IBM-701 was announced, scientists concerned with preparing for these machines, actively considered the problems of reducing problem preparation".

An analysis of the IBM-701 philosophy and logic shows that it is mandatory that an automatic technique (in this case interpretive) be used to make the equipment commercially satisfactory.

If the problem is submitted to a compiling routine or compiler, the computation proceeds as follows:

1. A word of the pseudo-code is interpreted.
2. The compiler locates the required subroutine and records in the running-tape the instructions for the transfer of the operands and control to the subroutine, and the transfer of the results and final controls.

a) If the subroutine is to be internally stored while running the problem, the compiler proceeds to step 4.

b) If the subroutine is to be tape-stored while running the problem, the compiler transforms the relative coding and enters the subroutine in the running tape.

4. The compiler records the operation and the running-tape position of the subroutine in the record.

5. The compiler continues processing the next word of the pseudo-code.

Thus, the result of the compilation procedure is a "running tape" specifically coded for the particular problem.

It should be noted that the interpretive steps 1 and 2 are not repeated again and again if the compilation method of programming is used. Because of the advantages and time-saving offered by the compiler technique, efforts were directed toward compilers rather than toward interpretive routines.

Compiler A-0

Compiler A-0 was completed and tested prior to 1 May 1952. A paper "Education of a Computer" covering the compiler techniques was presented by Dr. Hopper at the meeting of the Association for Computing Machinery in Pittsburg, 3 May 1952. (Exhibit A

Though our initial effort in the design and development of this technique, the A-0 Compiler, was completely logical in organization and arrangement, it was cumbersome in its application techniques. It is now of historical interest only (a very few valid copies are available).

In September 1952, Richard K. Ridgway of the Programming Research group reported the further developments in compiler techniques in a paper "Compiler Routines" at the meeting of the Association for Computing Machinery in Toronto. (Exhibit B) A further discussion of automatic programming was presented by Dr. Hopper at the Midwest Research Institute in January 1953. (Exhibit C)

Compiler A-1

The A-0 Compiler, complete as it was, with fundamental concepts sound and applicable, served as a base for the development of the A-1 and A-2 Compilers. Compiler A-1 was completed and proven by 1 January 1953, and much of the cumbersomeness of A-0 had been refined and eliminated. It was supplied with a general (eleven significant digit, unlimited exponent) two-word floating-decimal library. It was used to program the "Carne" problem concerning the response of a particular R-C circuit to a pulsed signal as well as several shorter problems. Experience with A-1 led to further refinements which have been reflected in the A-2 Compiler. Some of the conditions corrected were:

1. the pseudo-code has been made less cumbersome;
2. segmenting has been made automatic;
3. input and output subroutines are generative rather than static.

In May 1953, Computers and Automation printed an article by Dr. Hopper on "Compiling Routines". Reprints of this article were later circulated by the Electronic Computer Department to all Branch Offices. (Exhibit D)

First Workshop on Automatic Programming

On 16 July 1953, at the First Workshop sponsored jointly by the Bureau of the Census and Remington Rand was held in Washington, D. C. Mr. Savidge's efforts resulted in attendance by over ninety people, not only from the UNIVAC installations, but from all branches of the government as well as from certain nearby civilian activities. The A-1 Compiler was presented in lectures and demonstrated on the UNIVAC. (Exhibit E) Comments were favorable and great interest was expressed in the development of the techniques. This led to offers from the existing installations to use the compilers in actual day-to-day operation. Such application of the compilers required however the production of a manual telling; how to use a compiler, how to write information, how to write subroutines to be added to the systems, as well as how to operate the computer for compilation.

Compiler A-2

By August 1953, Compiler A-2 was completed, tested and operating. Drafts of a manual have been prepared and reproduced in ditto form. A limited quantity is available.

MIT-ACM Meeting, September 1953

Two papers presented by members of the Programming Research Group at this conference,

"Analytical Differentiation by a Digital Computer" by Harry G. Kahrimanian

"An Editing Generator" by Adele Mildred Koss and John H. Waite, Jr.

do not directly enter the history of compilers, though they both contributed to, and received contributions from, the compiler techniques. Also, Dr. Hopper was a member of the panel on automatic programming.

At this session, as well as from published material, and from the seminars held up to that time, it was evident that Remington Rand's Programming Research Group had progressed considerably further in the development and applicaltion of automatic programming techniques than had any other single or combined effort in this field. It can be definitely stated that this progress depended on several important considerations:

1. the compiling-generating technique is far more powerful than the interpretive technique;
2. since the UNIVAC is an alpha-numeric system, it permits great flexibility in the designation of pseudo-codes and specifications;
3. the readily available, checked, and practically limitless secondary memory provided by the UNISERVOs and tapes supplies the necessary resources for translators, tape libraries, etc. to a degree not attainable in any other computer to date;
4. the UNIVAC, being a checked computer will permit the production of computer

proved routines, taking over from the programmer the detailed and time-consuming effort required to insure reliable programs.

Second Workshop on Automatic Programming

On 1 December 1953 at the Second Workshop sponsored by the Office of the Air Comptroller and Remington Rand at the Pentagon approximately one hundred people were present. Not only were reports presented by the Programming Research Group but existing UNIVAC installations reported successful automatic programming. Lectures and demonstrations of the A-2 Compiler were given, including the programming and running of an optical ray problem. (Exhibit F)

Recent Developments

The Compiler A-2 has been successfully applied at Army Map (Exhibit G), Air Comptroller (Exhibit H and I), and Livermore (Exhibit J). Copies have been transmitted to NYU and BuShips. With the delivery in January of preliminary manuals (Exhibit K), it is to be expected that further suggestions, criticisms, and comments will be received. Thus as the A-0 Compiler led to the development of A-1, and likewise A-0 and A-1 to A-2, data is being accumulated and studied for further refinement of this technique which could lead to the A-3 Compiler.

The work with the three compilers, A-0, A-1, and A-2, has received unanimous acceptance from the existing UNIVAC installations. This leads to the conclusion that the approach is correct and that every effort should be put forth to correlate the data on hand for the development of an A-3 Compiler.

The A-2 Compiler was slanted by its available subroutines towards application to scientific and engineering problems. However, the basic concepts of A-2 as well as its logical composition are completely general. The transition of the compiler technique from A-0 through A-1 and A-2 has been so guided as to insure the development of a technique of great generality. With proper libraries of subroutines, including those required by commercial problems, the A-2 Compiler will operate just as satisfactorily, and function with the same degree of effectiveness, in commercial applications as it will in scientific applications.

Summary

1. Automatic programming techniques have been accepted by the computer industry. This statement is best supported by reproducing a table included in the section on computing devices of "1953 Engineering Developments" reviewed by the AIEE Technical Committees in the January 1954 issue of Electrical Engineering. A monumental effort is being expended by IBM, RCA, Raytheon, CPC (National Cash), Consolidated Engineering, Burroughs and all their users and prospective users, and by their affiliated university installations, to develop similar techniques for use with their digital computers. It is our understanding that the ERA organization is developing techniques of this type for application of the 1101 and 1103.

2. The response to the two Workshops on Automatic Programming confirmed the importance and effectiveness of this technique. The Workshops were attended not only by the UNIVAC users but also by representatives of many other government agencies including those contemplating 701's.

3. The response to the publications concerning compilers (including the article by J. W. Mauchly and G. M. Hopper on "Influence of Programming Techniques on the Design of Computers" in the Proceedings of the IRE, Computer Issue, vol. 41, no. 10, October 1953) and to other seminars conducted by independent organizations demonstrates the emphasis being placed on these programming techniques.

4. The number of contacts by individuals and organizations searching for information and knowledge concerning these techniques confirms the universal acceptance of what has been accomplished.

5. Through personal contacts, it has been determined that our closest competitors are making an all-out effort to approach the level that Remington Rand has attained and to surpass us

AIDS TO MAN'S WORK

	Physical		Mental and Clerical
a.	The Tool, man provides power and intelligence.	a.	Desk Calculator, slide rule, and so forth.
b.	Power Tool, man provides intelligence.	b.	Business machine -- processes masses of similar documents.
c.	Automatically Controlled Power Tool (includes primitive computing elements.) Example: steel mill drive.	c.	Large-Scale Automatic Computer Complete sequences of mathematical or business calculations are manually programmed.
d.	Automatic Factory or Process Line. The computer is mature co-ordinating groups of machines and making intricate decisions at high speed.	d.	Self-Programmed Automatic Computer. Interprets simple commands into extensive programs for science and industry.

Taken from Electrical Engineering, January 1954, page 24

This report is being supplemented as noted by transmittal of material and publications on work done by the Programming Research Group.

Recommendations

1. That a comprehensive review and report be prepared on the compiling techniques developed to date covering UNIVAC, 1103, 701, 702, RayCom, Elecom, Burroughs, CRC, MIT, Illinois, Michigan, Ferranti, EDSAC (LEO), Consolidated Engineering, etc.
2. That based on the findings of the above report, personnel be devoted 100% of their time to the development of the compiler techniques and particularly to the development of the A-3 Compiler.
3. That periodic reports of the developments in the field of automatic programming and compiling techniques as applied to all computers be submitted to support the evaluation and determination of the direction and scope the development of these techniques should take within Remington Rand.

DR. GRACE MURRAY HOPPER

31 December 1953

CARNE PROBLEM

Response of a Particular R-C Circuit
to a Pulsed Signal

I Statement of Problem

$$E(t) = -A \sum_{\alpha\beta\gamma} \frac{\alpha e^{-\alpha t}}{(\alpha-\beta)(\gamma-\alpha)} \quad \text{for } 0 \leq t \leq 20\,\mu \text{ seconds by } \Delta t = 1$$

where α, β, γ are roots of the cubic equation

$$p^3 + ap^2 + bp + c = 0$$

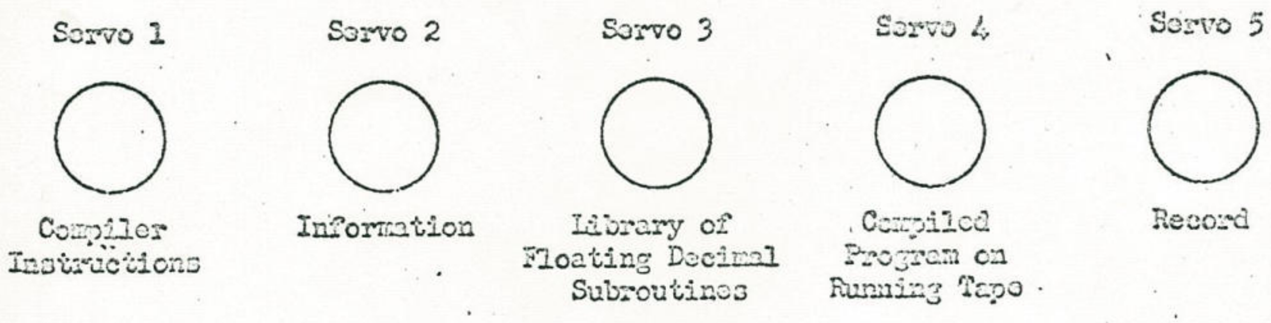
and $A = f(a, b, c)$

A. Final Formula for Calculation

$$E(t) = -A\left[\frac{e^{-at}\left\{(\alpha b)\cos bt + \left[b^2 - a(\alpha - a)\right]\sin bt\right\} - (\alpha b)\, e^{-\alpha t}}{b\left[b^2 + (\alpha - a)^2\right]}\right]$$

(39,375 values of E)

II Compilation



A. Compilation Time – 7 1/2 minutes

1. Compiler is read into computer

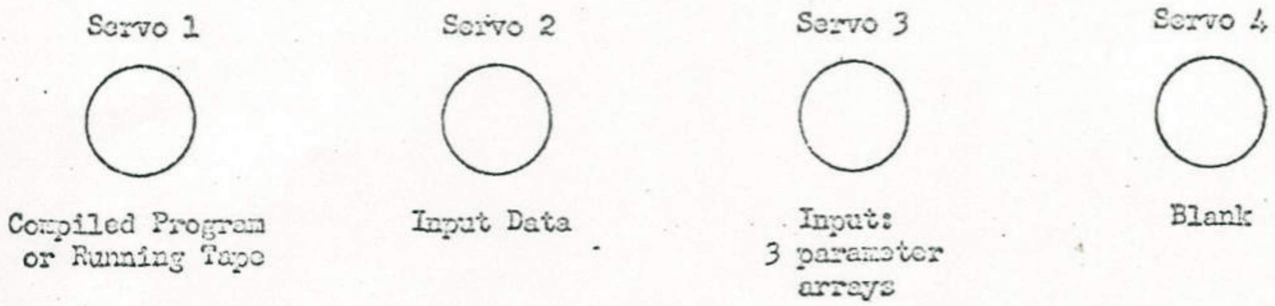
2. Information is read in secondly.

3. Compiler begins to process information.

4. Compiler looks for proper subroutine on Servo 3; finds it; prepares proper data transfer instructions; adjusts all addresses in the relative coded subroutines that are to be modified; then writes the subroutine on Servo 4 in the form of a finished part of the whole program.

5. Tape on Servo 5 keeps a record of each subroutine processed and the line number on which the routine begins in the program.

III <u>Running Program</u>



A. Description

1. Compiled program on Servo 1 is read into computer

2. Input data (3 parameter arrays) is read into the computer secondly.

3. Calculation of the Cubic coefficients then commences Time is 30 seconds.

(3 parameter arrays are values needed to calculate coefficients)

4. Coefficients are written on blank tape on Servo 4.

FRANK M. DELANEY