A MONTHLY PUBLICATION BY COMPUTER SCIENCES CORPORATION | VOL. 4 NO. 8

CSC REPORT

GENESIS FORTRAN COBOL PL/1 ALGOL JOVIAL

Ralph W. Pearson is Manager of Plans and Programs, Systems Programming Operations at Computer Sciences Division, and is a veteran of more than 18 years in the data processing industry.

He received his BS in Business Administration with concentration in Marketing at Northwestern University. He performed graduate studies in electronic computers at Illinois Institute of Technology. In addition, he has authored several articles for various periodicals, and has been a guest lecturer at numerous universities and AMA seminars. He is also an active member of ACM.

GENESIS: A COMPILER-COMPILER

By Ralph W. Pearson

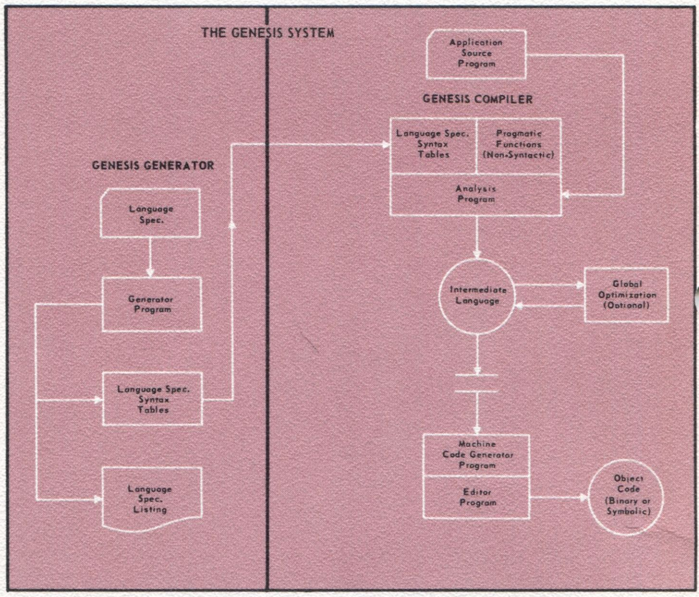
The GENESIS System provides a formalized method for the semi-automatic production of high-performance compilers and other systems software.

The primary purpose of the GENESIS System is to accelerate the delivery of these compilers at low total cost. One of the major attributes of GENESIS-produced compilers is that they are both machine- and language-independent, and can — and do — produce code running efficiently on any specified computer. No longer are compilers so closely tied to machine hardware that programs written in a specific compiler language will only run efficiently (if at all) on the computer for which the compiler was originally designed.

How GENESIS Works

Input to GENESIS is the detailed specification of a compiler source language (like FORTRAN, ALGOL, COBOL, etc.) written in an analytic grammar. This form of grammar was chosen in preference to the usual phrase-structure grammar because it is more easily processed by machines, it covers a broad spectrum of languages, and inherently forces the resolution of ambiguities. The GENESIS interpretation of this language specification results in the construction of a series of syntax tables which completely describe the language to be compiled. In addition, GENESIS at this point provides the capabilities of automatic printout of complete documentation of the language specification, and the running of test problems against the syntax tables to check for logical validity. During these trial compilations, syntax traces, intermediate dumps, and other debugging tools are provided.

The trial compilation, and the final GENESIS produced compiler utilize a standard analysis program to which are added the syntax tables and a set of pre-specified Pragmatic Functions (non-syntactic functions) for such processes as symbol table manipulating, number conversion, output, etc. Source-language programs are introduced into an Analyzer Program which, using the tables and functions, compiles a code string of the source program in the form of an Intermediate Language (IL). This


THE GENESIS SYSTEM
GENESIS GENERATOR
Language Spec.
Generator Program
Language Spec. Syntax Tables
Language Spec. Listing
Application Source Program
GENESIS COMPILER
Language Spec. Syntax Tables
Pragmatic Functions (Non-Syntactic)
Analysis Program
Intermediate Language
Global Optimization (Optional)
Machine Code Generator Program
Editor Program
Object Code (Binary or Symbolic)

IL code string represents an intermediate, albeit fully-compiled, output of the new compiler, and is at this point both language- and machine-independent.

Next, global optimization is (optionally) performed, by processing the code string through an optimization phase. This optimization across the whole program represents a significant contribution of the system, for it is this operation which increases the run-time efficiency of programs to a point where they compare very favorably with programs, hand-written by very good programmers. It surpasses human endeavors in the sense that it performs this optimization uniformly, consistently and in substantially less time.

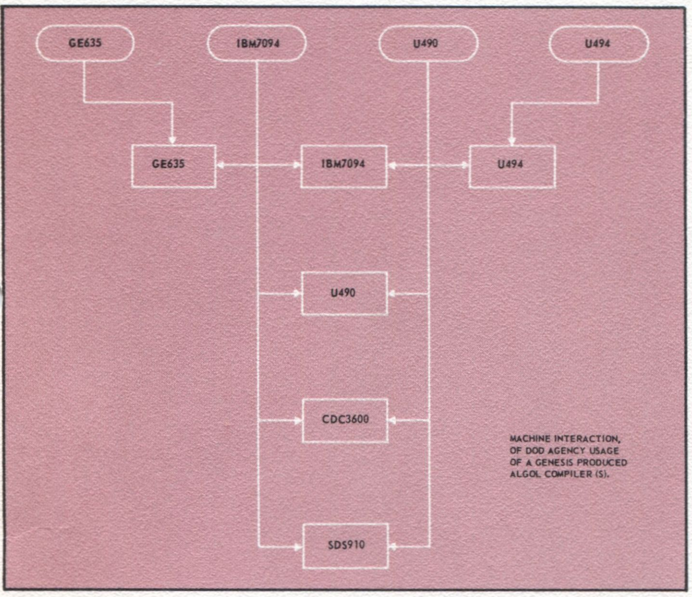
From this point, the optimized source string is processed by an Editor and a Code Generator program which tailor the program to any computer specified. The Editor and Code Generator are "pluggable" packages, and are especially written for each target machine for which a compiler and/or object code is desired. Local optimization is always provided in the code generator pass.

GENESIS itself is written in a special CSC-developed systems programming language called SYMPL. This language is designed to permit easy maintenance of the system, and facilitates the "bootstrapping" of compilers from machine to machine.

Simply stated then (no pun intended), the compiler portion of GENESIS is a skeleton model of a compiler, which, when provided with the "meat" of the language specification, and the instructional "brain" of the Pragmatic Functions, is capable of translating source language programs into an intermediate language which is both machine- and language-independent. Automatic global optimization is optionally provided. Editor and Code Generator programs then tailor this compiler's output to run efficiently on any computer specified.

Installations

The GENESIS System is a proven program. An agency of the DoD is currently using a GENESIS-produced ALGOL compiler on four different machines — i.e., essentially one and the same compiler operating on all of these machines. In addition to compiling programs for themselves, selected machines have been provided with the capability of producing code for other machines. Consider the unusual flexibility of this approach as illustrated by the diagram:


GE635
IBM7094
U490
U494
GE635
IBM7094
U494
U490
CDC3600
SDS910
MACHINE INTERACTION, OF DOD AGENCY USAGE OF A GENESIS PRODUCED ALGOL COMPILER (S).

In effect, this configuration represents 12 "compilers" operating on, or for, 6 separate machines. *Standard practice at this installation is for their programmers to submit programs for processing with little concern for which machine is to be used to compile or run the programs.* Other machines are currently being added to the configuration.

Performance

GENESIS compilers are not slow. Compiling speeds for JOVIAL on the System/360, Mod 65 in excess of 600 cards per minute, averaging 2-1/2 statements per card are expected. ALGOL users report object-code efficiency approaching — and in some instances exceeding — that of the "hand" code produced by some of their better programmers.

Other Languages

In addition to the ALGOL and JOVIAL languages, GENESIS language specifications for FORTRAN, COBOL, and PL/I, are being implemented. A high-performance JOVIAL compiler for the System/360 is under development.

Other Applications

The GENESIS System is not restricted to the production of compilers. Such tasks as translation from one compiler language to another; implementation of Data Management and Information Processing Languages; Command and Query Languages; Dynamic and Discrete Event Simulators — in fact, any program that requires language analysis — all are excellent candidates for GENESIS implementation. As an aside, GENESIS also compiles itself.

Benefits

The ability to produce compiler programs which are both machine- and language-independent, and then tailor them for various machines, provides many significant benefits:

Low Total Cost

Low total cost is the overriding benefit, and all other benefits of the GENESIS System relate to it. Cost will vary as a function of the complexity of the target machine and/or operating system. Typically, however, costs for high-performance systems are reduced by a third to a half, compared with conventional methods of compiler implementation.

Fast Delivery

GENESIS-produced compilers are delivered in from half to two-thirds the time required for comparable conventionally-produced compilers.

Compatibility

Compatibility of a given compiler language across a wide spectrum of machines is gained through the language- and machine-independence of GENESIS compilers. In this respect, conversion of programs, including the compiler itself, from one machine to another is greatly simplified.

Ease of Maintenance

The SYMPL language compiler, delivered with all GENESIS produced compilers, assures ease of compiler maintenance.

Hardware Independence

Access to the eventual target machine is unnecessary until the very last stages of compiler implementation. The GENESIS System now runs on a UNIVAC 1107 and will soon be "bootstrapped" through itself to an IBM 360. The availability of a target machine is relatively unimportant until final checkout.

High Performance

The GENESIS System is designed specifically for the production of high-performance compilers — that is, compilers which produce object programs which are consistently efficient in terms of speed and space utilization. This performance blend is assured by the quality of the optimization techniques provided.

Optional Optimization Level

Local Optimization is always provided. Global optimization is available as desired. The maximum obtainable utilizes the best of the techniques currently available.

Modification and "Bootstrapping"

The GENESIS System allows relatively easy modification of an existing GENESIS compiler to process a different language. It also provides for the efficient adaptation ("bootstrapping") of an existing GENESIS compiler to a new machine. To a great degree, these facilities are bonuses conferred by the method of implementation of the SYMPL language.

(Continued)

Documentation and Debugging

Automatic documentation of the language specification is provided, as are the debugging tools used in running sample programs testing the validity of each new compiler implementation.

Those who receive the greatest benefits from the GENESIS System are:

- Users who desire a number of languages compiling for a related series of machines — e.g., computer manufacturers.
- Users who require absolutely compatible compilation and running of programs on a wide variety of machines — e.g., the Federal Government, and other multiple-machine users.
- Any user seeking fast delivery of a high-performance compiler at low total cost.
- The user of a GENESIS compiler who wishes to convert an existing library of programs to a new machine.

Conclusion

The GENESIS System represents a breakthrough in the art of producing high-performance compilers. It not only cuts the cost and time of development of these complex programs, but compilers produced by it can be made to run efficiently on a broad spectrum of machines. It is not the answer to all of the problems in the computer industry, but it has been proven in use to be fundamentally sound and capable of performing to the degree specified. Machine-compiler language compatibility is now a practical reality.

H. L. TERWILLIGER TO HEAD COLORADO SPRINGS OFFICE

The appointment of key managerial personnel for Computer Sciences Corporation's new Colorado Springs facility was announced recently by William R. Hoover, president of Computer Sciences Division.

Hoover named H. L. Terwilliger as manager of the new office, located in the Holly Sugar Building, 100 Chase Stone Center. At the same time, Paul M. Botting was named technical manager.

Computer Sciences provides computer system design, consultation in data systems, scientific and commercial programming, and related services.

The appointments were announced by Hoover to coincide with the official opening. A group of Computer Sciences executives from the Los Angeles headquarters also attended the ceremonies.

Terwilliger will direct the expansion of Computer Sciences' services to military and governmental organizations in the Mountain States, as well as to commercial and industrial concerns.

"The new office will enable us to respond rapidly to the specialized requirements of the military commands headquartered here," Hoover said.

In addition, he said, "The extensive business development throughout this area has created a strong demand for professional services in the design of information systems and the application of computers for commercial purposes."

Terwilliger comes to CSC following eight years with International Business Machines Corporation. He has specialized in the development of military systems such as missiles, intelligence and range instrumentation.

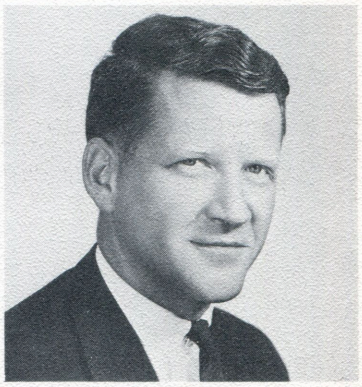
Prior to joining the company, Terwilliger managed advanced programs in the Real Time Systems Department of IBM's Federal Systems Division. In this capacity he directed functional studies related to the equipment and programming requirements of the North American Air Defense Command and the U.S. Air Defense Command.

Botting has been with Computer Sciences for two years. He was a group manager in the Applied Sciences Department of its Eastern Operations Center in Washington, D. C.

He has directed computer systems design and the solution of numerous operational problems pertaining to satellite projects for NASA's Goddard Space Flight Center.

Earlier, Botting was with the U.S. Naval Weapons Laboratory at Dahlgren Virginia, for nine years. During this period he designed and produced computer programs for the Navy's space surveillance activities and for various projects related to the Polaris missiles carried by a large number of U.S. nuclear-powered submarines.

G. H. THOMAS TO HEAD APPLICATIONS SYSTEMS



Geoffrey H. Thomas has joined Computer Sciences Corporation as manager of Application Systems in the company's Computer Sciences Division, it was disclosed recently.

He will direct the development of new proprietary product lines together with their marketing, to commercial and industrial organizations. These proprietary items consist of programs for common applications of computers in business.

Thomas will be based at the company's El Segundo, California, headquarters.

Prior to joining Computer Sciences, he was associated with International Business Machines Corporation for 13 years. In his most recent capacity, he was IBM's representative to educational organizations in the San Diego area.

Earlier, he was marketing manager in IBM's Data Processing Division, with responsibility for the division's marketing activities in the State of Arizona.

Thomas also served as an instructor in IBM's management training program at San Jose, California, where he conducted programs to educate senior executives from IBM's customer companies in the use of computer-based information systems in financial and industrial management.

CSC REPORT — AUGUST 1967
Published by Computer Sciences Corporation
650 North Sepulveda Boulevard
El Segundo, California 90245
Houston/Huntsville/New York/Richland, Wash.
San Diego/San Francisco/Hawaii/Wash., D.C.