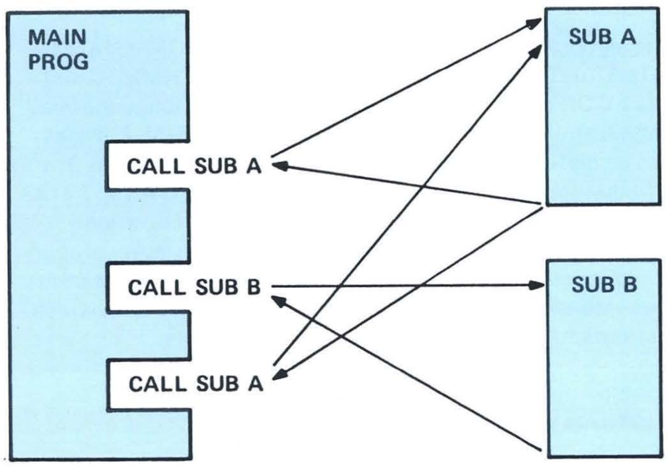
**PERKIN-ELMER** 32-BIT SOFTWARE

## UNIVERSAL OPTIMIZER



# The FORTRAN VII Family

### PRODUCT DESCRIPTION

Perkin-Elmer's family of FORTRAN VII products provides a spectrum of capabilities for FORTRAN programmers on Perkin-Elmer's 32-bit computer systems. State-of-the-art optimization techniques provide superior performance for technical, industrial and scientific applications written in industry standard FORTRAN. A suite of support software, including a high speed development compiler, maximizes programmer productivity during development and maintenance.

FORTRAN VII exceeds the American National Standard Full FORTRAN language (ANSI X3.9-1978). In addition, carefully selected language extensions provide increased programmer convenience and permit full exploitation of the Perkin-Elmer 32-bit architecture.

FORTRAN VII is augmented by comprehensive run-time support for user programs under OS/32, OS/32 MTM and/or Reliance. The run-time subprogram library includes mathematical functions, language extensions, real-time interfaces for multitasking and file access, and input/output. The ISA FORTRAN extensions for industrial processing and control are an integral part of the library.

### FEATURES

- State-of-the-art Optimizations
- Choice of Compilers:
  - FORTRAN VII Z, Universal Optimizer
  - FORTRAN VII O, Global Optimizer
  - FORTRAN VII D, Development Compiler
- Superior Operational Performance
- ANSI Full Standard Language with extensions
- Reentrant FORTRAN Programs
- Sharable, High-Speed Development Facility
- Comprehensive Run-Time Support

### UNIVERSAL OPTIMIZATION

FORTRAN VII Z provides a degree of optimization unparalleled in the industry. Previously, the maximum scope available from an optimizing compiler was Global, as is provided by FORTRAN VII O. Universal scope permits FORTRAN VII Z to eliminate the inefficiencies of inter-module interfaces, and to optimize subprograms within the context of their calls.

The results of this all encompassing scope are:

- Unexcelled Performance; FORTRAN VII programs, subjected to universal optimization outperform their globally optimized equivalents by a factor of up to four, depending on the complexity of the programs.
- Improved Programmer Productivity; By exploiting the power of the universal optimizer, programmers can write highly structured FORTRAN that is easier to create and debug. FORTRAN subprograms can be used in the same way an assembler programmer might use Macros, with comparable improvements in productivity.
- Reduced Life-Cycle Costs; The very same factors that contribute to programmer productivity during development, assure that FORTRAN VII programs are easy to read and maintain, thereby minimizing life-cycle maintenance and enhancement costs.

### LANGUAGE

The FORTRAN VII language is a powerful programming tool which exceeds the ANSI-FORTRAN-77 Full Language standard (X3.9-1978). In addition, FORTRAN VII provides a number of powerful extensions, which provide cost-effective portability of FORTRAN programs and allow the programmer to make optimal use of Perkin-Elmer's powerful 32-bit architecture.

REAL and DOUBLE PRECISION variables are supported by the processor floating point facilities, giving six digits of precision for REAL numbers and 16 digits for DOUBLE PRECISION, throughout the range $\pm\ 5.4\text{x}10^{-79}$ to $\pm\ 7.2\text{x}10^{75}$. In addition to 16 general-purpose registers, 16 floating point registers are available to the compiler for evaluating expressions and for temporary storage. COMPLEX variables are represented by two REAL values, or by two DOUBLE PRECISION values (COMPLEX *16).

Two forms of INTEGER are supported, fullword (*4) and halfword (*2), allowing the programmer to choose between the wide range of 32-bit values (± 2,147,483,647) and the space economy of the halfwords. LOGICAL variables are supported as Fullwords or single bytes (LOGICAL *1). Also provided are HOLLERITH and HEXADECIMAL constants.

In addition to arithmetic types, FORTRAN VII supports CHARACTER variables and expressions, including concatenation and substring notation and a powerful set of intrinsic functions for string manipulation.

The mixed mode arithmetic facilities add even more power to the generous repertoire of data types.

## COMPILATION

The FORTRAN VII system provides two compilation modes: operational mode, using the full facilities of either optimizer, and developmental mode, which provides high throughput compilation and debugging to multiple users of OS/32 MTM.

In operational mode, the VIIZ or VIIO compiler performs a wide range of state-of-the-art optimizations. The user can choose between global optimization, which operates one module at a time, or universal optimization, which encompasses up to the whole program. The resultant modules can be mixed at run-time.

Optimizations are made at the FORTRAN-source level. The compiler uses its knowledge of the rules of the language to generate an equivalent, optimum representation of the original program. To do this, it analyzes all flow paths through a user's program to ensure that program integrity is maintained under any possible operating condition.

Having achieved an optimum machine-independent version of the user's program, FORTRAN VII analyzes its execution resource requirements and allocates the computer resources to the elements of the program. Through these machine-dependent optimizations, FORTRAN VII ensures that the best possible use is made of the 32-bit architecture. Finally FORTRAN VII selects tailored code sequences to take optimum advantage of the powerful instruction repertoire for each section of the user's program.

In development mode, the FORTRAN VII programmer has a variety of program development facilities which foster efficient implementation of FORTRAN VII programs. Developmental facilities are provided under the auspices of the OS/32 Multi-Terminal Monitor (MTM) which provides interactive program development to up to 64 terminal users simultaneously.

During program development, individual FORTRAN VII programs are compiled, directly to linkable object code, in excess of 3000 lines per minute. Under normal load, terminal users will receive almost immediate turnaround for compilations.

## OPTIMIZATIONS — Machine Independent

In common with most high-order languages, FORTRAN provides many opportunities for optimization by a compiler. Many of the constructions in the language lead to very inefficient object code, if simplistically translated on an element-by-element basis.

The scope and scale of the optimizations performed by FORTRAN VII establish the state of the art in this field, and lead to unequalled execution speeds for FORTRAN programs.

The algebraic transformations, CONSTANT COMPUTATION, CONSTANT PROPAGATION, SYMBOLIC ARITHMETIC and TYPE CONVERSIONS, are the simplest of the language level optimizations. Their effects are simple in concept, but they can be quite far reaching because of hidden arithmetic. The introduction of the PARAMETER statement into ANSI-77 has further increased the value of this class of optimization because parameterized programs often have a larger proportion of constants. Universal optimization also benefits from these transformations because of the frequency with which subroutines are called with constant arguments.

**Machine-independent optimizations performed by FORTRAN VII are:**

### CONSTANT COMPUTATION

Expressions whose operands are explicit constants are evaluated by the compiler. CONSTANT COMPUTATION saves both time and space at run-time. This optimization is of particular importance in programs that access multidimensional arrays within DO-loops. Other optimizations frequently generate constant expressions, which are eliminated by CONSTANT COMPUTATION.

### CONSTANT PROPAGATION

This optimization propagates throughout the program values of any variables assigned to constants. CONSTANT PROPAGATION is important for parameterized programs and DO-loops. It often creates opportunities for CONSTANT COMPUTATION. Both run-time and space are saved.

### SYMBOLIC ARITHMETIC

Run-time and space are saved by applying simplifying SYMBOLIC ARITHMETIC, particularly extraction of a common multiplier (e.g., K1*X + K2*X becomes (K1 + K2)*X) and simplification of expressions involving 0 and 1. SYMBOLIC ARITHMETIC is of value during DO-loop degradation and in parameterized programs.

### TYPE CONVERSIONS

Compile time TYPE CONVERSIONS save both time and space. They are performed if the compiler can determine the value of operands in a mixed mode expression. They are most often valuable when integer constants are added to REAL or DOUBLE PRECISION expressions.

The DO-loop is one of FORTRAN's most powerful language facilities. It enables repetitive operations to be performed on logically associated data, for example, arrays. Frequently, DO-loops are nested inside each other with the innermost loop used to access multi-dimensional arrays. The next four optimizations combine to eliminate large amounts of hidden arithmetic.

**ARRAY LINEARIZATION**

This optimization is used to simplify the run-time calculations associated with accessing multi-dimensional arrays. The compiler replaces the complex calculations needed to access a multiple dimension array with the simpler calculation needed to access the equivalent one dimensional vector.

**INVARIANT CODE MOTION**

Frequently within loops code is included that does not depend on any other code in the loop. Such code is said to be invariant and is moved to just before the loop. INVARIANT CODE MOTION has no impact on space requirements, but can have a dramatic effect on run-time. Note that this optimization operates on any type of loop, not just DO-loops.

**STRENGTH REDUCTION**

Within DO-loops, many operations are dependent on the control variable. STRENGTH REDUCTION takes advantage of the repetitive nature of operations within DO-loops by using equivalent operations that execute faster. For example, multiplication by the control variable is replaced by addition to the value the last time around the loop.

**TEST REPLACEMENT**

A result of STRENGTH REDUCTION can be that a DO-loop control variable is no longer explicitly referenced within a loop. If so, TEST REPLACEMENT saves both time and space by eliminating the original control variable and replacing it with the most commonly used variable generated by STRENGTH REDUCTION.

The following optimizations are of value anywhere in a FORTRAN program:

**SCALAR PROPAGATION**

Most assignments of the form X = Y are eliminated and all uses of X are replaced by Y. SCALAR PROPAGATION saves both space and time.

**FOLDING**

FOLDING saves both space and time by eliminating intermediate assignments to unnecessary local variables.

**COMMON SUBEXPRESSION ELIMINATION**

Common sub-expressions are consolidated by this optimization, eliminating repeated calculation of the same value. Saves both time and space.

**DEAD CODE ELIMINATION**

Many of FORTRAN VII's optimizations can result in redundant or unreachable code. DEAD CODE ELIMINATION saves both space and execution time by eliminating such code.

The power of the language level optimizations is further enhanced by the algorithm used to apply them. Many of the optimizations create opportunities to further improve execution performance. For this reason the optimizations are applied repetitively until no further improvement is achieved.

**OPTIMIZATIONS — Machine Dependent**

These optimizations are performed to ensure that the object code generated by FORTRAN VII makes best possible use of the processing capabilities of the Perkin-Elmer 32-bit architecture.

**EXPRESSION REORDERING**

The sequence for evaluating expressions is ordered to minimize the number of registers for temporary storage of intermediate results.

**REGISTER ALLOCATION**

FORTRAN VII makes optimum use of the powerful register structure of the 32-bit architecture. The compiler minimizes memory accesses and takes best possible advantage of efficient register-to-register instructions. For smaller programs, FORTRAN VII keeps all variables and constants in registers, and for larger programs, those variables that are used most often are given priority.

**INSTRUCTION STRENGTH REDUCTION**

Many times, FORTRAN statements are written in a form which if directly translated, leads to inefficiencies. By performing some simplifying strength reductions at the instruction level, faster sequences of instructions are used to execute these common occurrences.

**MACHINE INSTRUCTION CHOICE**

Each sequence of instructions chosen by the FORTRAN VII code generator is carefully selected to ensure optimum efficiency for the Perkin-Elmer 32-bit architecture.

**DEVELOPMENT**

FORTRAN VII is a language system for use in developing large FORTRAN programs, whose execution speeds are required to be as fast as possible. It is expected that in most situations these FORTRAN programs will be developed by teams of cooperating programmers.

The FORTRAN VII developmental facilities promote swift implementation, testing and debugging of program modules and systems. The development environment is provided by OS/32 MTM, the multi-terminal timesharing system that supports a job mix of up to 64 interactive and batch users. The compiler is segmented Pure/Impure and is automatically shared by any number of concurrent users.

During program development, FORTRAN VII programs are compiled at speeds exceeding 3000 lines per minute directly into linkable object code. Automatic job control allows FORTRAN VII programs to be compiled, linked, loaded and executed with a single command input.

Creation of FORTRAN source programs is performed using standard operating system facilities. OS/32 EDIT is a powerful interactive editor ideally suited to the terminal user. The input spooler can be used to create disc files from card decks.

FORTRAN VII includes optional debugging facilities which provide FORTRAN source level testing of programs. Versatile trace facilities and run-time array bounds checking provide immediate feed-back on program problem areas.

**RUN-TIME LIBRARY**

The comprehensive FORTRAN VII run-time library gives exemplary run-time support to FORTRAN VII programs.

The **mathematical functions** employ modern numerical techniques and take full advantage of the power and flexibility of the 32-bit processors to provide results as accurate as is practical in the shortest possible time.

The accuracy is better than five decimal digits for the REAL functions and better than 14 decimal digits for DOUBLE PRECISION.

As an option, many commonly used mathematical functions are implemented as microcode*, for execution in Writable Control Store at approximately twice the speed.

The **language extensions** provide FORTRAN VII programmers with access to data types and operations not available within the language itself. These include the ISA logical facilities:

- Logical operations on bit strings
- Logical shift operations in integers
- Manipulation of individual bits
- Byte processing
- Queueing and pushdown operations on Perkin-Elmer circular list structures.

The **real time extensions** give FORTRAN VII programs access to the real-time facilities of OS/32. Real-time application systems of FORTRAN written tasks are a practical reality with FORTRAN VII in the OS/32 multi-tasking environment. The facilities, based on the ISA proposals, include:

- Intertask communication and control
- Interaction with external events
- Time of day and interval clock awareness
- File creation and access
- Analog/Digital conversion and Digital I/O
- Internal fault detection and response

The **input/output system** supports FORTRAN READ, WRITE, PRINT, TYPE, and ACCEPT statements for performing formatted, unformatted, binary, list-directed, and NAMELIST I/O, as well as extremely flexible auxiliary I/O statements. (OPEN, CLOSE, and INQUIRE) which support access to the operating system file manager. The formatter performs run-time execution of pre-translated FORMAT statements to provide conversion and editing of information between internal representation and external character strings. The pre-translation of the FORMAT statements ensures maximum run-time efficiency and reduces run-time memory requirements. The formatter also provides support for ANSI-FORTRAN 77 internal files which provide storage-to-storage data manipulations as well as the ENCODE and DECODE statements which are part of the defacto industry standard FORTRAN.

**OPERATING ENVIRONMENTS**

FORTRAN VII is fully integrated with other Perkin-Elmer 32-bit software products. The development mode of operation is ideally suited to the OS/32 MTM terminal user. The compiler requires approximately 100KB over and above the operating system, and each concurrent user needs from 5 to 10KB for compilations. Automatic job control facilities simplify the operation of the compilation process, through the OS Command Substitution System.

Optimizing compilations are best performed as a sequence of batch jobs, either under control of MTM or directly under OS/32. The compiler requires approximately 200KB over and above the operating system, plus up to 500KB table space. In addition, FORTRAN VII requires 1200KB of disc space for overlays. All compilers automatically page tables to disc if allocated memory is not sufficiently large to compile in memory.

**PRODUCT NUMBER**

| | |
|---|---|
| **S80-225*** | Universal FORTRAN Language System |
| **S80-216** | Global FORTRAN Language System |

**RELATED DOCUMENTATION**

**S80-225 BCM**
**S80-216 BCM** } Documentation Package

*Not available Models 7/32 and 8/32.

**PERKIN-ELMER**

**Computer Operations**
2 Crescent Place
Oceanport, N.J. 07757
(201) 870-4712
(800) 631-2154

Manufacturing facilities and Sales/Service offices located throughout the world; major subsidiaries located in AUSTRALIA: Adelaide, Albury, Brisbane, Canberra, Melbourne, Perth, Sydney, and Wellington, New Zealand; CANADA: Calgary, Montreal, Ottawa, Toronto, Vancouver; ENGLAND: Manchester, Slough; FRANCE: Arcueil, Bordeaux, Grenoble, Lille, Lyon, Perigueux, Toulouse; WEST GERMANY: Dusseldorf, Frankfurt, Munich, and Vienna, Austria; NETHERLANDS: Gouda; SINGAPORE: Singapore, Hong Kong; JAPAN: Tokyo. Other countries are serviced by a network of distributors.

Printed in U.S.A. January, 1982