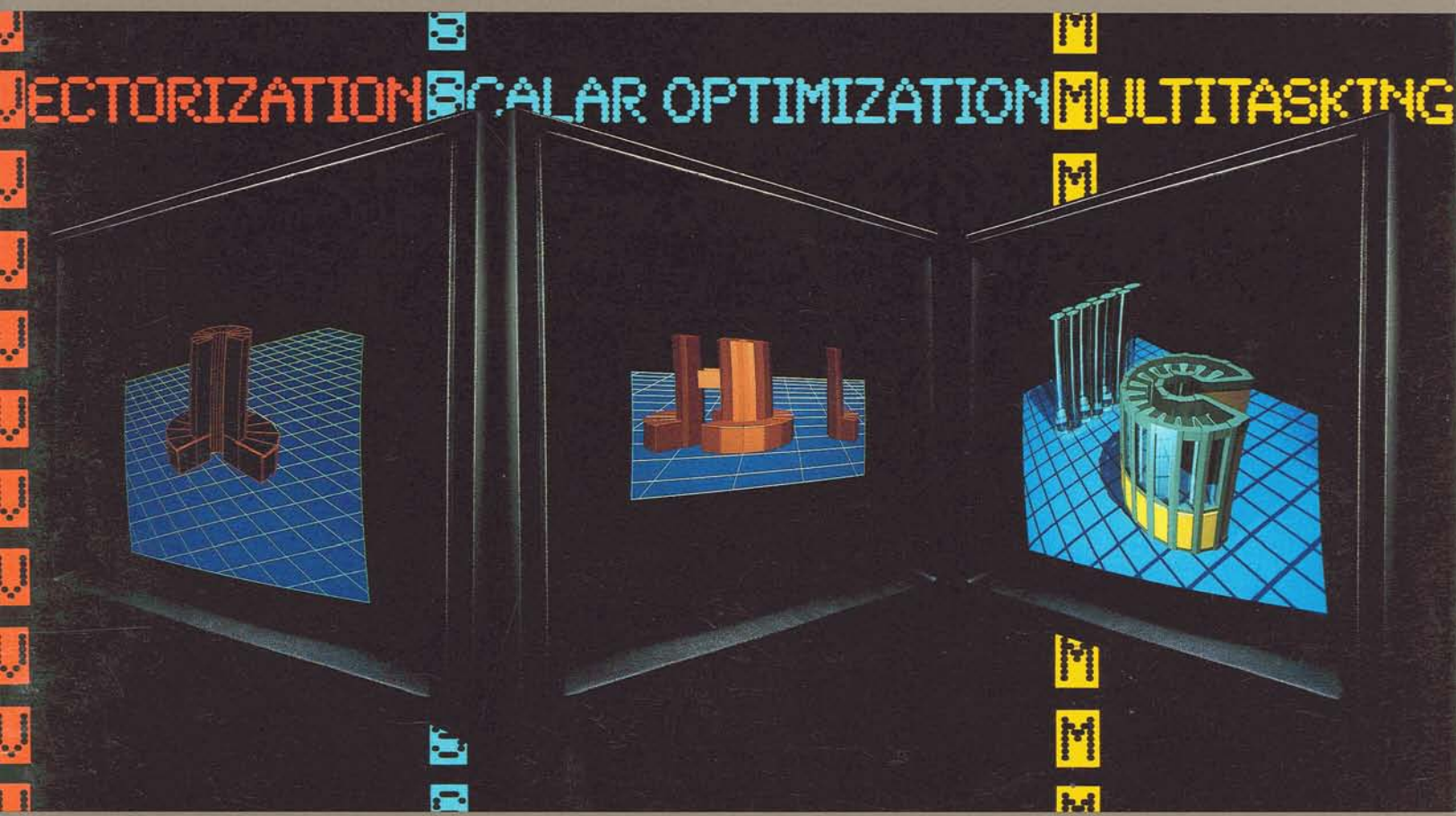
CFT77

Fortran compiler
for Cray supercomputers



CRAY

CFT77

VECTORIZATION & SCALAR OP

Cray Research has long recognized that high-performance hardware must be complemented with high-performance software to achieve the ultimate in high-speed scientific computing. Having pioneered the development of automatic optimizing and vectorizing compilers with the CFT Fortran compiler, Cray Research now proudly offers the CFT77 compiler, which represents the leading edge of compiler technology.

CFT77 is a multipass, optimizing, vectorizing, and multitasking compiler that adheres to the American National Standards Institute (ANSI) standard 3.9-1978 (often called Fortran 77). CFT77 processes existing standard Fortran programs without modification.

The CFT77 compiler is available for the CRAY X-MP series of computer systems, the CRAY-2 computer system, and for CRAY-1 computer systems — and it operates under both COS and UNICOS, the Cray operating systems. CFT77 takes full advantage of the unique hardware architecture of Cray computer systems and by doing so greatly enhances their performance. Thus, users benefit from hardware and software that work together to achieve maximum performance.

The high degree of software portability and superior performance offered by the CFT77 compiler result in increased productivity of the programming staff and efficient use of computing resources.

As users of the first Cray Fortran compiler (CFT) know, application performance and portability are the top priorities for Cray compiler developers. These goals are paramount to CFT77, which applies the latest techniques in software design to continue in the tradition of excellence established by Cray Research with the CFT Fortran compiler.

Techniques for high performance

TIMIZATION MULTITASKING

CFT77 uses three techniques to improve the execution time of a FORTRAN program: vectorization, scalar optimization, and multitasking. These three techniques are key to the performance of Fortran programs.

Vectorization
The compiler automatically generates code that uses the vector registers and functional units of the Cray hardware. Speedups in the area of 10 to 1 are common when comparing vector processing to scalar processing. The programmer does not need to know the details of vectorization; CFT77 automatically vectorizes Fortran programs.

Scalar optimization
Even when CFT77 cannot vectorize code, it still optimizes scalar code using a variety of optimization techniques to improve execution time.

Multitasking
CFT77 permits the partitioning of a program among multiple processors, enabling different parts to execute at the same time. Future plans include the ability for CFT77 to partition automatically. Multitasking teamed with vectorization is a powerful combination.

CFT77

Vectorization

- **Basic vector arithmetic/assignments**

```
      REAL A(100), B(100), C(100)
      DO 40 I = 1,N
         A(I) = B(I) + SQRT(C(I))
40    CONTINUE
      END
```

- **Gather/scatter**

```
      REAL A(100), B(100), C(100)
      INTEGER INDEX1(100), INDEX2(100)
      DO 20 I = 1,N
         A(INDEX1(I)) = B(INDEX2(I)) + C(I)
20    CONTINUE
```

- **Reduction of an array to a scalar value**

```
      DIMENSION A(100)
      SUM = 0
      DO 10 I = 1, 100
         SUM = SUM + A(I)
10    CONTINUE
```

- **Loop with an ambiguity that will not be resolved until runtime**

```
      DIMENSION A(100)
      DO 10 I = 11, 100
         A(I) = A(I-K) + R
10    CONTINUE
```

- **Search loop**

```
      REAL A(180)
      DO 150 I = 1,N
         IF (A(I).LT.0) GO TO 155
150   CONTINUE
      •
      •
      •
155   CONTINUE
```

- **Vectorizable IF**

```
      REAL A(100), B(100), C(100)
      REAL BIG
      BIG = 1.0E100
      DO 220 I = 1, 100
         A(I) = B(I)+6.0
         IF (A(I).NE.0) THEN
            C(I) = C(I)/A(I)
         ELSE
            C(I) = BIG
         ENDIF
220   CONTINUE
```

Vectorization is a method for reducing the execution time of repetitive code. Following is an overview of the difference between scalar and vector processing.

Vectorization means that specialized hardware is used for greatly increasing program performance. CFT77 takes care of vectorizing for the users; without a vectorizing compiler, a programmer would have to use assembly language to manipulate the hardware. Vectorized loops include those containing nested IF statements, loops that use indirect (gather/scatter) addressing, and search loops, among others.

CFT77 combines the practical knowledge gained in Cray Research's decade of vectorization experience with successful research programs from several universities.

CFT77 also provides an extensive set of vectorization diagnostics to indicate vectorized and unvectorized areas of code. Simple code changes or compiler directives often can help the compiler fully vectorize the unvectorized sections.

Scalar optimization

```
      SUBROUTINE SUMS (A,PSUM,SUM,M,N,L)
      INTEGER A(M,N), PSUM(M,N), SUM(M), L(M)
      IGET(IA,I,L) = SHIFTR(SHIFTL(IA,I),64-L)

      DO 20 I = 1,M
        K = 1
        DO 10 J = 1,N
          IF (IGET(A(I,J),0,32) .LT. L(I)) THEN
            SUM(I) = SUM(I) + IGET(A(I,J),0,32)
            PSUM(I,K) = SUM(I)
            K = K + 1 ENDIF
10      CONTINUE
20    CONTINUE
      END
```

The subroutine above, SUMS, sums the rows of array A into array SUM and saves each partial sum in array PSUM. Only elements of A that satisfy the IF test are summed. Also, the statement function IGET is used to extract a field (left half of word) from each element of A.

```
      SUBROUTINE SUMS(A,PSUM,SUM,M,N,L)
      INTEGER A(M * N), PSUM(M * N), SUM(M), L(M)
      m = M
      mn = m * N
      DO 20 i = 1,m
        k = i
        is = SUM(i)
        l = L(i)
        DO 10 j = 1, mn,m
          it = SHIFTR(A(j),32)
          IF (it.LT.l) THEN
            is = is + it
              PSUM(k) = is
              k = k + m
          ENDIF
10      CONTINUE
        SUM(i) = is
20    CONTINUE
      END
```

Subroutine SUMS is rewritten above to show the effect of global optimization. The variables in italics designate registers. An example of a register load is *m = M* and a store to memory is *SUM(i) = is*.

The multi-dimension arrays A and PSUM are shown as having one dimension to highlight the optimization of their address computations.

CFT77 efficiently optimizes scalar code. As with vectorization, CFT77 approaches scalar optimization by analyzing a complete program unit.

Scalar optimization transforms the internal representation of the Fortran program into a more efficient but functionally equivalent program. This is achieved by simplifying expressions and by detecting and eliminating redundant operations. The following optimization techniques recognized as being state-of-the art by today's compiler developers are incorporated into CFT77:

- ☐ Common subexpression elimination
- ☐ Forward propagation of constants and expressions
- ☐ Extracting invariant expressions from loops
- ☐ Strength reductions
- ☐ Hoisting and sinking
- ☐ Moving stores out of loops
- ☐ Store elimination
- ☐ Dead code elimination
- ☐ Arithmetic simplification
- ☐ Short circuiting of logical expressions
- ☐ Constant expression evaluation
- ☐ Bottom loading of loops

These scalar optimizations are always transparent to the user.

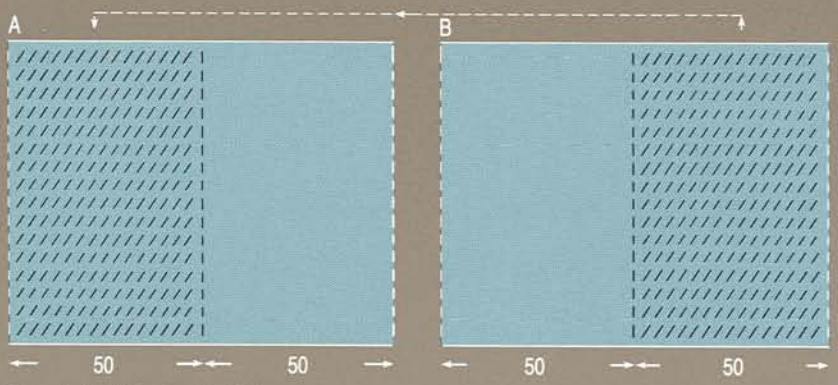
CFT77

Multitasking

Array processing is a CFT77-supported extension to Fortran 77 that allows representation of DO loops in a more compact fashion. Using it, any section or part of an array can be addressed.

Assuming A and B are both two-dimensional arrays (100 by 100), the following array syntax statement copies the shaded portion of array B into the shaded portion of array A, element by element.

```
A(1:100:1, 1:50:1) = B(1:100:1, 51:100:1)
```



The equivalent statements using DO loops are as follows:

```
      DO 10 I = 1,100
        DO 10 J = 1,50
10        A(I,J) = B(I,J,+50)
```

The multitasking capabilities of CFT77 permit the programmer to divide a single program among the multiple central processing units offered on CRAY X-MP and CRAY-2 computer systems.

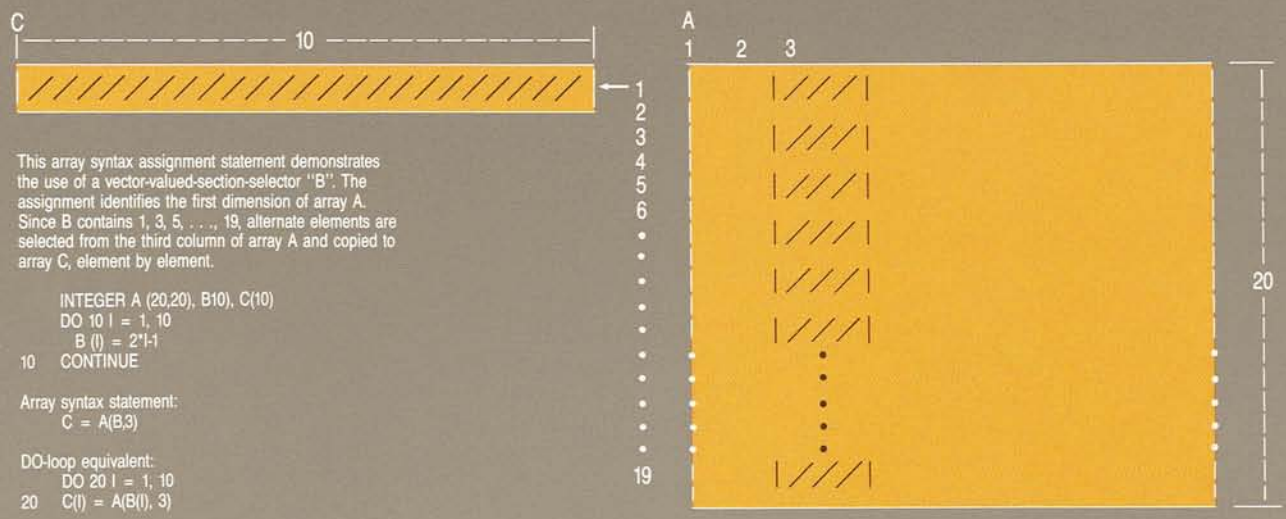
The speedup possible with multitasking is a function of the number of central processors available, the degree of parallel processing in the program, and the overhead inherent in multitasking. Speedup factors in the range of 3.6 to 3.8 have been achieved on four-processor systems and of up to 1.8 for two processors.

Cray Research currently supports two approaches to multitasking. In the first approach, the user multitasks a Fortran program by inserting calls to library routines that implement a basic set of multitasking functions. In the second approach, called microtasking, the user invokes the PREMULT preprocessor by inserting directives in the Fortran source code. PREMULT then generates the appropriate library calls.

CFT77 supports both approaches to multitasking, as does CFT. When stack storage allocation is specified, CFT77 generates reentrant code. The TASK COMMON statement allows the declaration of COMMON blocks known only to a single task; this is often useful in the first multitasking approach.

Currently under development and expected to be available in the near future as a feature of CFT77 is the ability to multitask some Fortran code automatically. Emphasis is placed on multitasking at the DO-loop level. Cray Research is exploring how best to implement multitasking so that the use of multiple processors is as easy as the use of vector processors.

Features



This array syntax assignment statement demonstrates the use of a vector-valued-section-selector "B". The assignment identifies the first dimension of array A. Since B contains 1, 3, 5, . . ., 19, alternate elements are selected from the third column of array A and copied to array C, element by element.

```
      INTEGER A (20,20), B10), C(10)
      DO 10 I = 1, 10
        B (I) = 2*I-1
10    CONTINUE

Array syntax statement:
      C = A(B,3)

DO-loop equivalent:
      DO 20 I = 1, 10
20    C(I) = A(B(I), 3)
```

CFT77 is a language rich in features. It contains all features described in the Fortran 77 standard as well as a number of extensions to the language. Some of these extensions, such as stack storage allocation and TASK COMMON, are necessary to support multitasking. Other features such as NAMELIST I/O and Hollerith constants are frequently used in existing Fortran programs; CFT77 supports these features so that existing codes can be moved to CFT77 without extensive conversions. A few features that are expected to be in the next Fortran standard have been implemented in CFT77; these include a subset of the array syntax (see example above).

The same language features are supported in CFT77 on all Cray computer systems. The extensions supported include the following:

- Array processing, which permits operations on whole arrays or array sections (a subset of the proposed Fortran 8X standard array processing)
- Automatic arrays, with flexible bounds
- Recursive functions and subroutines
- Pointer data type
- Hollerith constants
- Boolean constants (octal and hexadecimal)
- Variable names of up to 31 characters and external and COMMON block names containing up to 8 characters
- Comments embedded within a line
- Compiler directives for listing output control, vectorization control, dynamic common blocks, and array bounds checking
- A choice of static or stack storage allocation methods
- TASK COMMON storage for multitasking
- On the CRAY-2, COMMON blocks allocated to local memory, permitting faster access to frequently used variables
- Asynchronous I/O, which allows I/O operations to execute simultaneously with other program statements
- Mixed formatted and unformatted records in a file under the COS operating system
- NAMELIST I/O
- Extra edit descriptors, including those for right justification and octal or hexadecimal output

CFT77

The Cray Fortran environment

The environment surrounding the CFT77 compiler contains a wealth of library routines and tools that make the user's job both easier and faster.

Library routines
Supporting the CFT77 compiler and the high-performance hardware inherent in a Cray computer system is a library of highly optimized subroutines to aid scientific and engineering computation.

Regardless of the machine and operating system, a wide variety of library routines are callable from CFT77. They include:

- Mathematical routines that are intrinsic to Fortran
- Scientific application routines
- I/O and utility routines

Routines in these libraries perform random number generation, Fourier analysis, sorting, and many other operations. Fortran programs that need a frequently used operation can be served by Cray's standard libraries.

All of the library routines are optimized. They have been coded to keep execution time to a minimum; many are coded in assembly language to maximize efficiency.

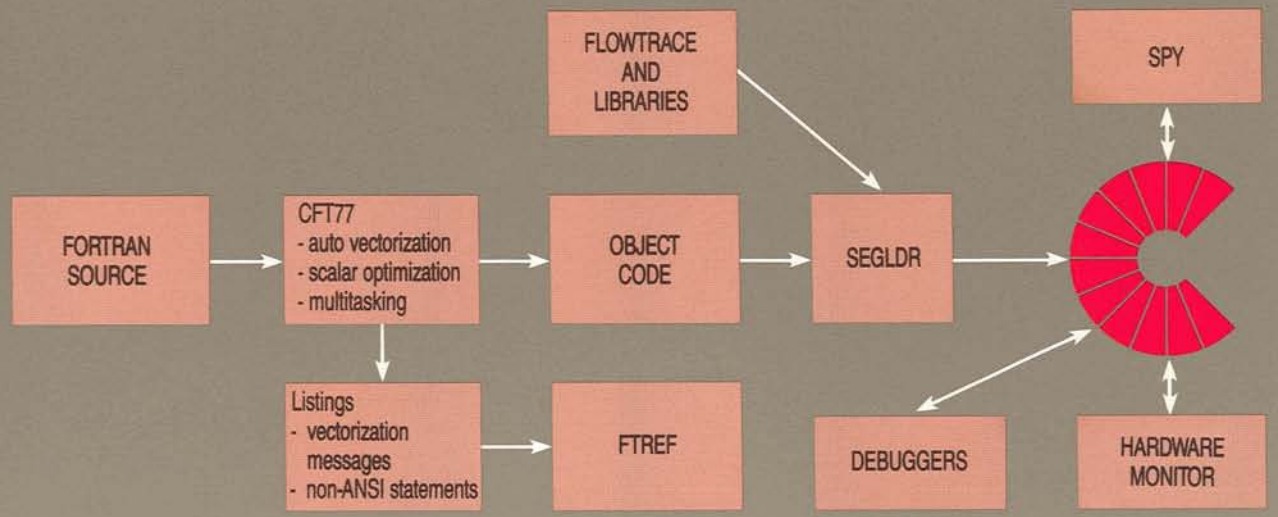
Linking to non-Fortran routines
CFT77 is compatible with other Cray Research language processors. Routines compiled with CFT77 may call or be called by routines compiled by the Pascal, C, or CFT compilers, or routines assembled by the CAL assembler.

Segment loader
SEGLDR, the segment loader, allows control over memory use at run time. This is particularly useful for large codes with several distinct sections, such as initialization, computation, and output.

Symbolic debug package
Included with CFT77 is a debug package to help users locate errors in their applications. The package consists of the following parts:

- DEBUG, which analyzes a memory dump of a job and provides listings of variable names and values
- DRD, which is a powerful interactive symbolic debugger that analyzes the memory of an executing job based on user directives
- DDA, which allows interactive analysis of a memory dump of a job using a subset of DRD directives

Multitasking tools
A multitasking history trace buffer provides for the accumulation of a history of multitasking events. An associated tool, MTDUMP, interprets this data and reports the sequence of execution, task history, and processor history. These tools aid the user in understanding multitasking behavior, identifying bottlenecks, and debugging programs.



Non-ANSI flags
At the user's request, CFT77 will flag features that are not part of Fortran 77.

List options
Many options are available for generating output listings, including a source statement listing with any of five levels of error messages and a listing of assembly code generated by CFT77. Diagnostic messages are issued on the source listing for loops that are not vectorized.

Cross referencing
CFT77 has an extensive cross-reference facility. The listing includes addresses, references and definitions of variables, statement labels, subroutine names, and so on. All are keyed to the Fortran line number.

FTREF
The FTREF program, a global cross-reference utility, provides a static analysis of program flow and common block use. The latter is provided in both summary and detailed formats. FTREF also has options specifically oriented to multitasked applications.

FLOWTRACE
The FLOWTRACE option is a useful tool for fine-tuning program performance. It shows where the code spends its time and helps locate the sections where special optimization could be applied for increased performance.

Hardware performance monitor
On CRAY X-MP computer systems, the hardware performance monitor allows users to identify bottlenecks and to compute MFLOPS (millions of floating point operations per second). The monitor accumulates statistics on the following hardware activities:

- Instructions executed
- Floating-point operations
- Hold issue conditions
- Reference conflicts
- Vector operations

SPY
This is a code-level profiler available for the CRAY X-MP computer systems. Like FLOWTRACE, it is useful for fine-tuning program performance. SPY samples the hardware program address register to build a map of where the program spends its time and can provide information at a lower level of detail than that provided by FLOWTRACE.

Data conversion
I/O library routines convert data and tape/disk formats during Fortran I/O operations. Data is converted to and from Cray formats and IBM, CDC, or DEC VAX formats. Users may also disable data conversion during I/O operations and perform the conversion by calls to special library routines.

CFT77

CFT77 design philosophy

Transportability
Through its many features and because of its compliance with the 1978 ANSI standard, CFT77 assures that programs written for other computer systems have maximum portability with a minimum of effort.

Additionally, CFT77 contains a number of extensions to the ANSI standard, including those already supported in CFT. Some of the extensions add helpful features that make Fortran richer and more flexible. Others enhance portability by reflecting features added to Fortran by other computer manufacturers, such as IBM and CDC.

Cray Research took portability one step beyond ANSI compliance by designing CFT77 to run on all of its machines and under all Cray supported operating systems. It runs on CRAY-1, CRAY X-MP, and CRAY-2 computer systems and executes under COS and UNICOS (the Cray operating systems).

A Fortran program that compiles and runs on one Cray system will compile and run on all Cray systems. Different codes do not need to be maintained for each machine. Upgrading to a new Cray system, therefore, is easy.

Changing from CFT to CFT77 is also easy. In general, programs that compile and execute correctly with the old CFT compiler also compile and execute correctly with CFT77.

Cray Research will implement CFT77 on future generations of its computers. The compiler has been structured so this can be done quickly, without sacrificing the performance of generated code. Therefore, the program optimized today for a CRAY X-MP or CRAY-2 computer system will move easily to the new Cray systems of tomorrow.

Structure of the compiler
CFT77 is designed for the future. The compiler is structured for easy adaptation to new Cray hardware as it becomes available and to new optimization techniques as they evolve. Because it is written in Cray's extended Pascal, CFT77 is both portable and maintainable.

The structure of CFT77 is organized around three major functions: source input and semantic analysis; optimization and vectorization; and code generation.

In the first phase of the compilation, CFT77 reads the Fortran statements and translates them into an intermediate form used in later processing. This section of CFT77 is virtually the same on all Cray machines, meaning source code that compiles on one machine will compile on the others.

Additional information

The intermediate code consists of text and a dictionary. The text is a representation of the executable Fortran statements. The dictionary is a collection of the attributes associated with the text items.

During the second phase, CFT77 performs optimization transformations on the intermediate text and determines the vectorizable sections of the code. This phase is optional. Bypassing it slows down the execution speed of the generated code, but the corresponding speedup in compilation time can be valuable during development and debugging.

In its final phase, CFT77 generates machine instructions from the intermediate text and dictionary. The instructions are scheduled to take advantage of the asynchronous execution of the independent functional units common to all Cray computers. Each code generator also takes advantage of specific hardware features, such as chaining, local memory, or gather/scatter operations. Upon completion of this third phase, the machine language code is ready for loading and execution.

Documentation and training
Cray Research supports all of its software products with technical manuals and training. Programmers may be interested in the following:

- The CFT77 Reference Manual, which describes the entire CFT77 language and its interface to the Cray operating systems
- The Progammer's Library Reference Manual, which describes the routines available and how they can be called from CFT77
- A course on CFT77 offered by Cray Research at the Mendota Heights, Minnesota, training facility, which provides information and practical experience in code conversion, debugging, and programming to take advantage of vector processing and other basic optimization techniques

Additional information on CFT77 is available from any Cray Research sales office.

CRAY RESEARCH, INC.
608 Second Avenue South
Minneapolis, MN 55402
612/333-5889

Domestic sales offices

Albuquerque, New Mexico
Atlanta, Georgia
Beltsville, Maryland
Boston, Massachusetts
Boulder, Colorado
Chicago, Illinois
Cincinnati, Ohio
Colorado Springs, Colorado
Dallas, Texas
Detroit, Michigan
Houston, Texas
Huntsville, Alabama
Laurel, Maryland
Los Angeles, California
Minneapolis, Minnesota
Pittsburgh, Pennsylvania
Pleasanton, California
Rochester, New York
Seattle, Washington
St. Louis, Missouri
Sunnyvale, California
Tampa, Florida
Tulsa, Oklahoma

International subsidiaries

Cray Canada Inc.
Toronto, Canada

Cray Research France S.A.
Paris, France

Cray Research GmbH
Munich, West Germany

Cray Research Japan, Limited
Tokyo, Japan

Cray Research S.R.L.
Milan, Italy

Cray Research (UK) Ltd.
Bracknell, Berkshire, U.K.

MP-1009