Chapter 7

Notes on the Implementation

7.1 CM Numbers

7.1.1 Floating-Point Representation

The representation of floating-point numbers on the CM-5 system conforms to the IEEE floating-point standard. Floating-point overflow and other conditions are not detected in the nodes; such errors are detected by the partition manager for scalar data only.

The approximate range of the IEEE single, **`F_floating`**, and **`D_floating`** formats is $0.29*10^{-38}$ through $1.7*10^{38}$, with about 7 decimal digits of precision for the single-precision forms and 15 for the double-precision form. The approximate range of the IEEE double format is $0.56*10^{-308}$ through $0.9*10^{308}$, with about 15 decimal digits of precision.

7.1.2 Arithmetic Exceptions

The CM-5 handles floating-point exceptions in an IEEE standard manner. Overflow, division by zero, inexact, and invalid operands are masked in CM arrays of floating-point types. Overflow turns into a properly signed infinity, and division by zero (except 0/0) turns into a properly signed infinity. Underflow turns into zero instead of a denormalized number.

Floating-point exceptions are handled differently by Sun control processors (partition managers), although the behavior still conforms to IEEE standards. In scalars and front-end arrays of floating-point types, the system executes traps for overflow, divide by zero, and invalid operand; underflow turns to zero; and inexact is masked.

Copyright © 1991–1994 Thinking Machines Corporation

The CM system does not detect integer overflow for **INTEGER*4** or **INTEGER*8** calculations. For example, multiplying 2**62 by 2**62 gives incorrect results without warning. Further, the partition manager and the vector units produce different incorrect results if overflow occurs while converting an **INTEGER*8** to an **INTEGER*4**.

7.2 Long-Term Restrictions

Fortran 90 requires that actual array arguments match their corresponding dummy arguments in type; CM Fortran adds the requirement for CM arrays that they must match in shape and home as well. These restrictions arise from the implementation of CM Fortran for distributed memory, and they hold for all CM hardware configurations.

7.2.1 Array Argument Home

As specified in the *CM Fortran Language Reference Manual*, the home of an actual array must match the home of a dummy array argument to which it is passed. Consequently, a failure occurs if a CM actual array is passed to a front-end dummy array, or if a front-end actual array is passed to a CM

Version 2.2, October 1994
Copyright © 1991–1994 Thinking Machines Corporation

dummy array. The programmer must ensure that the actual and dummy array homes match. An example illustrates this restriction:

```
PROGRAM ERRONEOUS
PARAMETER ( NA = 1000, NB = 500 )
REAL A(NA), B(NB)
A = [1:NA]          ! A is a CM array
DO I = 1, NB
  B(I) = NB         ! B is an FE array
END DO
CALL SQUARE( A, NA )! OKAY: CM actual to CM dummy
CALL SQUARE( B, NB )! ERROR: FE actual to CM dummy
...
END

SUBROUTINE SQUARE( X, N )
REAL X( N )
X = X*X             ! Dummy X is a CM array
END
```

The program above fails at the second call to subroutine **SQUARE** because of an array home mismatch. This will occur even if the main program and subprogram are compiled as part of the same file, since CM Fortran compiles program units completely independently of one another.

The compiler can detect this problem if interface blocks are used. Alternatively, compile with the switch **-safety=1** (or higher value) to catch the error at run time.

7.2.2 Array Argument Shape

CM Fortran requires that the shape of an actual argument match the shape of the corresponding dummy argument in two cases:

- in a reference to a procedure with an explicit interface (i.e., with an interface block present)
- if the actual argument and the dummy argument are CM arrays

The compiler enforces the first case in most situations. The programmer must enforce the second case.

Copyright © 1991–1994 Thinking Machines Corporation

A procedure reference fails at run time if a CM array actual argument is passed to a CM array dummy argument of a different shape. For example, the subroutine

```
SUBROUTINE SUB( A, N )
REAL A(N)
A = A - 1.0              ! A is a CM array
END
```

should be called passing an array section, as with

```
CALL SUB( X(1:J), J )
```

The subroutine can be called with the statement

```
CALL SUB( X, J )
```

only if array **X** actually has the shape **[J]**, which would be the case if **X** were declared using the statement

```
REAL X(J)
```

Compile with the switch **-safety=1** (or higher value) to catch mismatches in rank and shape at run time.

Version 2.2, October 1994
Copyright © 1991–1994 Thinking Machines Corporation