Chapter 11

Some Simple Programs

This appendix presents several programming examples in CM Fortran:

- a histogram program
- a convolution program
- a prime number sieve
- a solver for Laplace's equation

For comparison, several of these programs are presented in Fortran 77 versions as well.

NOTE

These programs are intended only to illustrate syntax and style in CM Fortran programming. They have not been tuned for performance.

Consult the *CM-5 CM Fortran Performance Guide* for information on optimizing programs.

Copyright © 1989–1994 Thinking Machines Corporation

11.1 Histogram

This subroutine for computing a histogram is shown in both serial and parallel versions.

11.1.1 Histogram: Fortran 77 Version

```
      SUBROUTINE HISTOGRAM(NPOINTS, POINTS, NBARS,
     $                         LOW, HIGH, BARS, ERR)
      INTEGER  NPOINTS, NBARS, BARS(NBARS), I, J
      REAL      POINTS(NPOINTS), LOW, HIGH, BARWIDTH
      LOGICAL  ERR
C
C     Initialization
C
      ERR = .FALSE.
      BARWIDTH = (HIGH-LOW)/NBARS
      DO 1 I=1,NBARS
         BARS(I) = 0
1     CONTINUE
C
C     Iterate over POINTS, updating BARS and checking for errors
C
      DO 2 I=1,NPOINTS
         IF (POINTS(I) .LT. LOW .OR. POINTS(I) .GT. HIGH) THEN
            ERR = .TRUE.
         ELSE
            J = IFIX((POINTS(I)-LOW)/BARWIDTH)+1
            BARS(J) = BARS(J) + 1
         ENDIF
2     CONTINUE

      RETURN
      END
```


Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

11.1.2 Histogram: CM Fortran Version

```
      SUBROUTINE HISTOGRAM(NPOINTS, POINTS, NBARS,
     $                     LOW, HIGH, BARS, ERR)
      IMPLICIT NONE
      INTEGER NPOINTS, NBARS, BARS(NBARS), BAR, PT
      REAL POINTS(NPOINTS), LOW, HIGH, BARWIDTH
      LOGICAL ERR
      INTEGER TEMP1(NPOINTS, NBARS), TEMP2(NPOINTS, NBARS)
C
C     Initialization
C
      ERR = .FALSE.
      BARWIDTH = (HIGH-LOW)/NBARS
C
C     First, we fill TEMP1 with values so that TEMP1(PT,BAR) =
C     BAR, then we fill TEMP2 such that TEMP2(PT,BAR) contains
C     the bar number in which POINTS(PT) falls. Finally, we
C     count the "intersections" of TEMP1 and TEMP2 for each bar
C
      FORALL (BAR=1:NBARS)  TEMP1(:,BAR) = BAR
      FORALL (PT=1:NPOINTS, BAR=1:NBARS)
     $   TEMP2(PT,BAR) = IFIX( (POINTS(PT)-LOW)/BARWIDTH )+1
      BARS = COUNT( TEMP1 .EQ. TEMP2, DIM=1 )
C
C     Check for errors
C
      ERR = ANY(POINTS .LT. LOW .OR. POINTS .GT. HIGH)

      RETURN
      END


C   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

      PROGRAM HISTTEST
      INTEGER NPOINTS, NBARS, I, J, K
      PARAMETER (NBARS = 10, NPOINTS = NBARS*(NBARS+1)/2)
      LOGICAL ERR
      INTEGER BARS(NBARS)
      REAL POINTS(NPOINTS)
```


Copyright © 1989–1994 Thinking Machines Corporation

```
 BARS = 0
 K = 1

 DO I=1,NBARS
    POINTS(K:K+I-1) = REAL(I)
    K = K + I
 END DO

 ERR = .FALSE.
 CALL HISTOGRAM(NPOINTS, POINTS, NBARS,
$                0.9, 10.1, BARS, ERR)
 DO I=1,NBARS
    PRINT *,I,BARS(I)
 END DO
 IF (ERR) PRINT *,"An error was detected"
 END
```

11.2 Convolution

This program illustrates 1-dimensional convolution over a 2-dimensional data set (done in column direction). It is shown in both serial and parallel versions.

11.2.1 Convolution: Fortran 77 Version

```
PROGRAM CONVOLVE
INTEGER NT,NX,NFP
PARAMETER (NX=2048)
PARAMETER (NT=128)
PARAMETER (NFP=16)
REAL F(NFP)
REAL P(NT,NX), Q(NT,NX)
```


Copyright © 1989–1994 Thinking Machines Corporation

```
C     Initialize P and Q

      DO I=1,NX
        DO J=1,NT
          P(J,I) = FLOAT(J)
          Q(J,I) = 0.0
        END DO
      END DO

C     Initialize F

      DO I=1,NFP
        F(I) = I
      END DO

C     Shift the array P by one over the first dimension
C     Compute Q after the shift on P

      DO IFP=1,NFP
        DO I=1,NX
          DO J=NT-1,1,-1
            P(J+1,I) = P(J,I)
          END DO
          P(1,I) = 0.0
        END DO

        DO I=1,NX
          DO J=1,NT
            Q(J,I) = Q(J,I) + P(J,I) * F(IFP)
          END DO
        END DO
      END DO

C     Show results

      PRINT *, ( Q(I,1), I = 1,NT )

      STOP
      END
```


Copyright © 1989–1994 Thinking Machines Corporation

11.2.2 Convolution: CM Fortran Version

```
      PROGRAM CONVOLVE
      IMPLICIT NONE
      INTEGER, PARAMETER :: NX=16, NT=16, NFP=16
      INTEGER I, IFP
      REAL F(NFP)
      REAL, DIMENSION(NT,NX) :: P, Q
C
C     Initialize P and Q on the CM.
C
      Q = 0.0
      FORALL(I=1:NT)  P(I,:) = FLOAT(I)
C
C     Initialize F on the front end.
C
      DO IFP = 1, NFP
         F(IFP) = IFP
      END DO
C
C     Shift the array P by one over the first dimension.
C     Compute Q after the shift on P.
C
      DO IFP = 1, NFP
         P = EOSHIFT(P, DIM=1, SHIFT=-1)
         Q = Q + P * F(IFP)
      END DO
C
C     Show results
C
      PRINT *, ( Q(I,1), I = 1, NT)

        END
```


Copyright © 1989–1994 Thinking Machines Corporation

11.3 Prime Number Sieve

This program for finding the prime numbers in a set of numbers is presented in three versions: a serial version and two alternative parallel versions.

11.3.1 Primes: Fortran 77 Version

```
      PROGRAM FINDPRIMES
      INTEGER I, J, N
      PARAMETER (N = 999)
      LOGICAL PRIMES(N), CANDID(N)
C
C     Initialization
C
      DO 1 I=1,N
         PRIMES(I) = .FALSE.
         CANDID(I) = .TRUE.
 1    CONTINUE
      CANDID(1) = .FALSE.
C
C     Loop: Find next valid candidate, mark it as prime,
C           invalidate all multiples as candidates, repeat.
C
 2    DO 4 I=1,SQRT(REAL(N))
         IF (CANDID(I)) THEN
            PRIMES(I) = .TRUE.
            DO 3 J=I,N,I
               CANDID(J) = .FALSE.
 3          CONTINUE
         ENDIF
 4    CONTINUE
C
C     At this point, all valid candidates are prime
C
      DO 5 I=SQRT(REAL(N))+1,N
         PRIMES(I) = CANDID(I)
 5    CONTINUE
```


Copyright © 1989–1994 Thinking Machines Corporation

```
C
C       Print results
C

        DO I=1,N
            IF (PRIMES(I)) PRINT *,I
        END DO
        END
```

11.3.2 Primes: First CM Fortran Version

This parallel version is a straightforward translation of the serial program.

```
        PROGRAM FINDPRIMES
        IMPLICIT NONE
        INTEGER I, N, NEXTPRIME
        PARAMETER (N = 999)
        LOGICAL PRIMES(N), CANDID(N)
C
C       Initialization
C
        PRIMES = .FALSE.
        CANDID = .TRUE.
        CANDID(1) = .FALSE.
C
C       Loop: Find next valid candidate, mark it as prime,
C             invalidate all multiples as candidates, repeat.
C
        NEXTPRIME = 2
        DO WHILE ( NEXTPRIME .LE. SQRT( REAL(N) ) )
            PRIMES( NEXTPRIME ) = .TRUE.
            CANDID( NEXTPRIME:N:NEXTPRIME ) = .FALSE.
            NEXTPRIME = MINVAL( [1:N], DIM=1, MASK=CANDID )
        END DO
C
C       At this point, all valid candidates are prime.
C
        PRIMES( NEXTPRIME:N ) = CANDID( NEXTPRIME:N )
C
```


Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

```
C       Print results
C
        PRINT *, "Number of primes:", COUNT(PRIMES)
        DO I=1, N
            IF (PRIMES(I)) PRINT *, I
        ENDDO
        END
```

11.3.3 Primes: Second CM Fortran Version

This parallel version uses a different algorithm from the serial program, one that allows it to use more parallel processors.

```
        PROGRAM FINDPRIMES
        IMPLICIT NONE
        INTEGER I, N, NN
        PARAMETER (N = 500)
        INTEGER TEMP1(N,N), TEMP2(N,N)
        LOGICAL CANDID(N,N), PRIMES(N)
C
C       Initialization
C
        CANDID = .FALSE.
        FORALL (I=1:N)   TEMP1(I,:) = 2*I+1
        FORALL (I=1:N)   TEMP2(:,I) = 2*I+1
C
C       At this point, the temporary 2-dimensional arrays are
C       modulated element by element. If an element in TEMP1 is
C       not a multiple of the corresponding element of TEMP2, or
C       (for the sake of an easily generated argument to the up-
C       coming ALL instrinsic) the element in TEMP1 is greater
C       than or equal to the corresponding element in TEMP2, then
C       the corresponding element in the CANDID array is set.
C
        WHERE  ( ((MOD( TEMP1,TEMP2 ) .NE. 0)
      $                .AND. (TEMP1 .GT. TEMP2))
      $                .OR.    (TEMP1 .LE. TEMP2) )
      $         CANDID = .TRUE.
```


Copyright © 1989–1994 Thinking Machines Corporation

```
C
C       The following statement performs an AND across the second
C       dimension of CANDID and stores the results in PRIMES.
C
        PRIMES = ALL(CANDID, DIM=2)
C
C       Print results
C
        PRINT *, "Number of primes:", COUNT(PRIMES)+1
        PRINT *, 2
        DO I=1,N
            IF (PRIMES(I)) PRINT *, 2*I+1
        ENDDO
        END
```


Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

11.4 Laplace Solver

This program solves Laplace's equation

$$\nabla^2 f = 0$$

on the unit square ([0,1] x [0,1]), subject to the boundary condition that $f = 1$ at $y = 1$ and $f = 2$ along the rest of the boundary. This program uses the 5-point Jacobi relaxation method, with f initially set to 0 on the interior.

```
      PROGRAM LAPLACE
      PARAMETER (MAXX=32)
      PARAMETER (MAXY=MAXX)
      REAL F(MAXX,MAXY),DF(MAXX,MAXY)
      LOGICAL CMASK(MAXX,MAXY)
      REAL RMS_ERROR,MAX_ERROR
      INTEGER ITERATION
C
C     Initialize the mask for the interior points
C
      CMASK = .FALSE.
      CMASK(2:MAXX-1,2:MAXY-1) = .TRUE.
C
C     Initialize F
C
      F = 2.
      F(:,MAXY) = 1.
      WHERE (CMASK) F = 0.
C
C     Set a dummy value for MAX_ERROR
C
      MAX_ERROR = 1.
      ITERATION = 0
C
```


Copyright © 1989–1994 Thinking Machines Corporation

```
C       Iterate until MAX_ERROR < 1.E-3
C
        DO WHILE (MAX_ERROR.GT.1.E-3)
            ITERATION = ITERATION + 1
C
C           Compute DF, the change at each iteration, and update
C
            DF = 0.
            WHERE (CMASK)
                 DF = 0.25*(CSHIFT(F, DIM=1, SHIFT=1)
       &                  + CSHIFT(F, DIM=1, SHIFT=-1)
       &                  + CSHIFT(F, DIM=2, SHIFT=1)
       &                  + CSHIFT(F, DIM=2, SHIFT=-1)) - F
                 F = F + DF
            ENDWHERE
C
C       Compute the RMS and Maximum errors.
C
            RMS_ERROR = SQRT(SUM(DF*DF)/((MAXX-2)*(MAXY-2)))
            MAX_ERROR = MAXVAL(DF,MASK=CMASK)
C
C       See if we should print things out
C
            IF (MOD(ITERATION,10).EQ.0) THEN
                 WRITE (6,*) ITERATION,RMS_ERROR,MAX_ERROR
            ENDIF
        ENDDO
C
C       Write the final iteration count
C
        WRITE (6,*) ITERATION,RMS_ERROR,MAX_ERROR

        END
```


Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation