Computer Physics Communications 42 (1986) 191–196
North-Holland, Amsterdam

VECTORIZATION OF THE THREE-DIMENSIONAL ISING MODEL PROGRAM ON THE CDC CYBER 205

Michael CREUTZ

Department of Physics, Brookhaven National Laboratory, Upton, NY 11973, USA

K.J.M. MORIARTY *

Consortium for Scientific Computing, Inc., John Von Neumann Center for Scientific Computing, P.O. Box 3717, Princeton, NJ 08543, USA

and

M. O'BRIEN

Institute for Computational Studies, Department of Mathematics, Statistics and Computing Science, Dalhousie University, Halifax, Nova Scotia B3H 3J5, Canada

Received 23 April 1986; in final form 27 June 1986

PROGRAM SUMMARY

Title of program: ISING

Catalogue number: AALU

Program available from: CPC Program Library, Queen's University of Belfast, N. Ireland (see application form in this issue)

Computer: CDC CYBER 205 (Model 682); *Installation:* CYBERNET Data Center, 1151 Seven Locks Road, Montrose Building, Rockville, MD 20854-2996, USA

Operating system: CDC CYBER 200 VSOS 2.1.6

Programming language used: CDC FORTRAN 200 (FORTRAN-77 plus CDC enhancements and vector instructions)

High speed storage required: 110 Kwords

Number of bits in a word: 64

* Permanent address: Institute for Computational Studies, Department of Mathematics, Statistics and Computing Science, Dalhousie University, Halifax, Nova Scotia B3H 3J5, Canada.

Peripherals used: terminal, line printer

Number of lines in combined program and test deck: 346

Reference to other published version of this program: CPC 39 (1986) 173

Keywords: Ising model, phase transitions, critical exponents, correlation functions, magnetization, microcanonical methods, vector processors, vectorization methods

Nature of the physical problem
Equilibrium configurations of the elementary magnets in the three-dimensinal Ising model are generated thus allowing the calculation of the critical exponents.

Method of solution
A previously published FORTRAN program [1] for the three-dimensional Ising model using the microcanonical method [2] is vectorized on the CDC CYBER 205 vector processor (with two vector pipelines) and an updating rate of 117 Mspins/s is obtained. On a four vector pipeline machine this rate would be increased to 234 Mspins/s.

Restrictions on the complexity of the program
The only possible restriction on the program is the run time which has to be considerable in order to reduce the errors in the measurements of the critical exponents. The storage re-

quirements for the program, which are IX × IY × IWIDE, are not severe and cause no problems on today's supercomputers.

Typical running time
The test run on 129 × 128 × 384 lattice with 10000 sweeps through the lattice took 9.1 min on the 8 Mword, 2 vector pipeline CDC CYBER 205 at Rockville, Maryland.

References
[1] M. Creutz and K.J.M. Moriarty, Comput. Phys. Commun. 39 (1986) 173.
[2] M. Creutz, Phys. Rev. Lett. 50 (1983) 1411.
G. Bhanot, M. Creutz and H. Neuberger, Nucl. Phys. B235[FS11] (1984) 417.

LONG WRITE-UP

1. Introduction

Numerical simulation of statistical systems is usually done by stochastic methods, mainly the "Metropolis" Monte Carlo scheme [1]. We have found that considerable increase in speed can be achieved by the use of an essentially deterministic method [2] which avoids real arithmetic and simulates randomness by the statistical fluctuations inherent in a large system. Fast programs for studying the three-dimensional Ising model using a scalar computer, the CDC CYBER 170-730, in both assembly language (COMPASS) [3] and standard ANSI FORTRAN-77 [4] have been published. The essentials of such a program for the two-dimensional model appeared in ref. [5]. In order to achieve a higher spin-flip rate than we obtained on the CYBER 170-730, we migrate our code to a vector processor, the CDC CYBER 205.

2. Outline of the theory

A spin-$\frac{1}{2}$ Ising model on an IX × IY × (64 × IWIDE) simple cubic lattice is considered. Helical boundary conditions are used in the X-direction, with periodic boundary conditions in the Y and Z-directions. IX should be odd and IY even. There are 0.5 × (IMAX × 64) auxiliary variables called demons, which carry energy. The demons act on the spins and try to invert them, the process being allowed to go through if and only if permitted by energy constraints, described in detail in refs. [2–6]. After the demons, hopping from spin to spin (with a big hop in the Z-direction of about 100 steps), have covered the entire lattice, physical observables may be measured. Further details are given in refs. [3,4]. A description of some preliminary measurements of the critical exponents for the three-dimensional Ising model is contained in ref. [6].

3. Code description

The program consists of the following routines: ISING(main program), UPDATE, ENERGY, BETA, IBCOUNT and CORX. These routines have the following functions:

1) ISING is the main driver routine which initializes the parameters of the program and starts the simulation.
2) UPDATE carries out one sweep through the lattice.
3) ENERGY evaluates the average bond and demon energies.
4) BETA calculates the inverse temperature β from an average demon energy ED.
5) IBCOUNT counts the set bits in the bit string X, i.e. carries out the population count.
6) CORX counts the antiparallel spins, separated by given number of sites, in the X-direction.

A description of the function of each of these routines is contained in ref. [4] where the routine presently called UPDATE was named MONTE. For the present test run subroutine CORX is not called but it is a simple matter for the user to initiate these correlation measurements. A routine corresponding to CORZ of ref. [4] would have to be added by the user. The test run output is presented at the end of this paper.

In table 1 we present the spin update rate on 4 representative computers. The CDC CYBER 170-730 is a superminicomputer with a performance in the DEC VAX 11/750 range. The CDC 7600, which was introduced in 1968, was the supercom-

Table 1
The spin update rate for the three-dimensional Ising model with the microcanonical method, with no measurements, on some representative scalar and vector computers. (The results for the performance on the ETA-10 are estimated from the known properties of this computer.)

Computer	Type	Mflips
CDC CYBER 170-730	scalar uniprocessor	0.16
CDC 7600	scalar uniprocessor	29
CDC CYBER 205	vector uniprocessor (2 vector pipelines)	117
CDC CYBER 205	vector uniprocessor (4 vector pipelines)	234
ETA-10	vector multiprocessor	
	(on one processor)	332
	(on eight processors)	2658

puter of its time with a rated peak performance of 5 Mflops. Even by today's standards the performance on the CDC 7600 is impressive and this is because of such features as a bit population count instruction. The CDC CYBER 205 performance is achieved making extensive use of the CDC CYBER 200 FORTRAN Q8 hardware calls [7], such as Q8ORV, Q8XORV, Q8XORNV, Q8ANDV, Q8ANDNV, Q8NANDV, Q8SHIFTV and Q8CNTO. By careful arrangement of the lattice in memory, all need for gathers and scatters is eliminated. Our vector length is given by the variable IVL = 0.5 * IMAX. (For the test run the vector length is 8256 which takes optimal advantage of the CDC CYBER 205 memory-to-memory architecture in its vector unit.) Other authors who have adapted similar algorithms to the CDC CYBER 205, e.g. Schumacher [8] and Bhanot et al. [9] have achieved spin-flip rates of 83 and 98 Mflips, respectively. The ETA-10 performance is calculated on the basis of a 7 ns clock period and 2 vector pipelines for each of eight vector processors. The performance of the Ising model on several other machines is given in the recent paper by Reddaway et al. [10].

Two interesting points should be made about the CDC CYBER 205 architecture which makes it especially useful for Ising model calculations. First, the CYBER 205 is bit addressable which means that word boundaries are ignored and the number of lattice sites in any direction does not have to be a multiple of 64. Thus we can study a whole range of lattice sizes very efficiently. Second, the CYBER 205 can have up to 4 vector pipelines (the test run was carried out on a 2 vector pipeline machine). This doubling of the number of vector pipelines doubles the arithmetic rate and hence the Mflip rate.

4. Conclusions

We now have a very efficient code for calculations on the three-dimensional Ising model using the microcanonical method on a machine, the CDC CYBER 205, which has excellent cost/performance. Our future plans include running this code for extremely long runs, say 1 Msweeps per data point on a large lattice, to obtain accurate measurements of the critical exponents of this model. We have already published some measurements for the Ising model [6] – see also ref. [5], as well as the graphs in refs. [3,4]. We also intend to test universality by carrying out calculations on lattices other than the cubic lattice, e.g. the bcc and fcc lattices [11].

An interesting recent development has been the proposal for a deterministic method [12] which allows for studies of non-equililbrium phenomena [13]. By using the program described in the present text, giving the demons the same indices as the spins being updated, eliminating the scrambling shifts and rearranging the loops over I0 and J, we are immediately able to carry out that deterministic dynamics.

Acknowledgements

We would like to thank Lloyd M. Thorndyke, L. Kent Steiner and John E. Zelenka of ETA Systems, Inc. for access to the 2 Mword, 2 vector pipeline CDC CYBER 205 at Colorado State University at Fort Collins, Colorado, Robert M. Price of Control Data Corporation for his continued interest, support and encouragement and access to the CYBERNET 8 Mword, 2 vector pipeline CDC CYBER 205 at Rockville, Maryland, Dietrich

Stauffer for valuable correspondence, the Control Data Corporation PACER Fellowship Grants (Grant nos. 85PCR06 and 86PCR01) for financial support, the Natural Sciences and Engineering Research Council of Canada (Grant no. NSERC A8420) for further financial support and the DOE for time on the CDC CYBER 205 at Florida State University. This research was also carried out in part under the auspices of the US Department of Energy under contract no. DE-AC02-76CH00016.

References

[1] Monte Carlo Methods in Statistical Physics, second edition, ed. K. Binder (Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1986).

[2] M. Creutz, Phys. Rev. Lett. 50 (1983) 1411.

[3] M. Creutz, P. Mitra and K.J.M. Moriarty, Comput. Phys. Commun. 33 (1984) 361.

[4] M. Creutz and K.J.M. Moriarty, Comput. Phys. Commun. 39 (1986) 173.

[5] G. Bhanot, M. Creutz and H. Neuberger, Nucl. Phys. B235 (1984) 417.

[6] M. Creutz, P. Mitra and K.J.M. Moriarty, J. Stat. Phys. 42 (1986) 823.

[7] CDC CYBER 200 FORTRAN, Reference Manual, No. 60457040 (Control Data Corporation 215 Moffett Park Drive, Sunnyvale, CA 94086, 1983).

[8] H. Schumacher, Staatsexamensarbeit Thesis, University of Cologne (September 1985).

[9] G. Bhanot, D.W. Duke and R. Salvador, J. Stat. Phys. 44 (1986) 1005.

[10] S.F. Reddaway, D.M. Scott and K.A. Smith, Comput. Phys. Commun. 37 (1985) 351.

[11] M. Creutz, J.-M. Drouffe and K.J.M. Moriarty, FORTRAN Code for the Three-Dimensional Ising Model on b.c.c. and f.c.c. Lattices, Brookhaven National Laboratory Preprint in preparation.

[12] M. Creutz, Ann. Phys. 167 (1986) 62.

[13] M. Creutz, Microcanonical Monte Carlo, Invited Talk given at the 1986 Eastern Simulation Conferences, Simulations at the Frontier of Science, Norfolk, Virginia (10–12 March 1986).

PROGRAM LISTING

```
      PROGRAM ISING(OUTPUT)
C     THIS PROGRAM SIMULATES THE THREE DIMENSIONAL ISING MODEL.
C     FOR DETAILED INFORMATION AND REFERENCES SEE THE PAPER
C     "VECTORIZATION OF THE THREE-DIMENSIONAL ISING MODEL PROGRAM
C     ON THE CDC CYBER 205"
C     BY M.CREUTZ, DEPARTMENT OF PHYSICS,
C     BROOKHAVEN NATIONAL LABORATORY,
C     UPTON, LONG ISLAND, NEW YORK 11973,
C     K.J.M. MORIARTY AND M. O'BRIEN, INSTITUTE FOR
C     COMPUTATIONAL STUDIES, DEPARTMENT OF MATHEMATICS, STATISTICS
C     AND COMPUTING SCIENCE, DALHOUSIE UNIVERSITY,
C     HALIFAX, NOVA SCOTIA B3H 3J5, CANADA
C
C  THE LATTICE IS IX BY IY BY 64*IWIDE. IX SHOULD BE ODD AND
C  IY SHOULD BE EVEN.
C    THE LATTICE IS STORED IN X AND Y DIMENSIONS AS (NUMBERS ARE
C  THE FIRST INDEX IN ISP):
C
C       IL    : IVL+IL    IL+1                                  :
C  -----------:                                                 :
C  IVL   IMAX :   1       IVL+1    2   IVL+2  ....  IVL+IS   IL:
C             :-------------------------------------------------:
C      IVL-IS : IMAX-IS  IVL-IS+1                               :
C
C  IN Z DIRECTION USE SECOND INDEX AS:
C
C    1,2,3, ... IWIDE, 1, 2, ... IWIDE, ..... (64 REPETITIONS)
C  WHERE REPETITIONS REPRESENT SUCCESSIVE BITS IN THE
C  CORRESPONDING WORD.
C
      DIMENSION ISP(16512,6),ID1(8256),ID2(8256)
C
C SWEEP NIT TIMES THROUGH THE LATTICE.
C REPEAT NBATCH TIMES AND FIND GRAND AVERAGE OF BETA.
C
C THE FOLLOWING LINE IS USED IN THE GENERATION OF
C AN APPROXIMATELY RANDOM WORD. NOTE THAT THE CDC COMPILER HAS
C A DEFICIENCY IN THAT IT WILL USE THE SAME RANDOM NUMBER WHEN
C RANF IS CALLED TWICE IN A LINE.
C
      RWORD(X)=XOR(RANF(X),SHIFT(RANF(X),31))
      IX=129
      IY=128
      IMAX=IX*IY
      IWIDE=6
      IZ=64*IWIDE
      PRINT 100,IX,IY,IZ
 100  FORMAT(1X,'LATTICE SIZE IS',I4,' BY ',I4,' BY ',I4)
      IE=0
      IVL=IMAX/2
      NIT=2000
      NBATCH=5
      ABATCH=1.*NBATCH-1.
      BS=0.
      BS2=0.
C INITIALIZE LATTICE.
      DO 1 I=1,IVL
      ID1(I)=0
      ID2(I)=0
      DO 1 J=1,IWIDE
      K=IVL+I
      ISP(I,J)=RWORD(X)
      ISP(K,J)=RWORD(X)
      ISP(I,J)=AND(ISP(I,J),RWORD(X))
1     ISP(K,J)=AND(ISP(K,J),RWORD(X))
C CALCULATE AVERAGE BOND ENERGY.
      CALL ENERGY(ISP,IX,IMAX,IWIDE,ID1,ID2)
C MAJOR LOOP STARTS.
      DO 3 MEAS=1,NBATCH
      IE=0
      T=SECOND(0)
      DO 2 I=1,NIT
2     CALL UPDATE(ISP,IX,IMAX,IWIDE,ID1,ID2,IE)
      T=SECOND(0)-T
      SPEED=IMAX*IWIDE*64.*NIT/(T*1.E6)
      PRINT 101,NIT,SPEED
101   FORMAT(//1X,'OVER ',I4,' ITERATIONS'/
     C       1X,'RUNNING AT ',F5.1,' MEGAFLIPS')
      CALL ENERGY(ISP,IX,IMAX,IWIDE,ID1,ID2)
      E=1.*IE/(64.*NIT*10*IWIDE)
      B=BETA(E)
      PRINT 102,B
102   FORMAT(1X,'BETA =',F8.6)
      IF (MEAS.NE.1) THEN
         BS=BS+B
         BS2=BS2+B**2
      ENDIF
3     CONTINUE
      BS=BS/ABATCH
      BS2=SQRT((BS2/ABATCH-BS**2)/(ABATCH-1.))
      PRINT 103,BS,BS2
103   FORMAT(//1X,' *** AVERAGE AFTER DISCARDING FIRST BATCH ***',/
     C       1X,'AV. BETA= ',F15.13,' +/- ',E9.3)
      STOP
      END

      SUBROUTINE UPDATE(ISP,IX,IMAX,IWIDE,ID1,ID2,IE)
      DIMENSION ISP(16512,6),NBR(8256,6),ID1(8256),ID2(8256)
      DIMENSION ICARRY(8256),IBIT1(8256),IBIT2(8256),IREJ(8256)
      IVL=IMAX/2
      I1=1
      I63=63
      I21=21
      I43=43
      IS=(IX-1)/2
      IL=IX-IS
      IVL4=IVL-1
      IVL5=IVL-IS
      IVL6=IVL-IL
      DO 1 J=1,IWIDE
      DO 1 IO=1,2
      IU=1+(IO-1)*IVL
      IN=1+(2-IO)*IVL
C FIRST NEIGHBOR.
      IF (J.EQ.IWIDE) THEN
        CALL Q8SHIFTV(X'08',,ISP(IU,1;IVL),,I1,,NBR(1,1;IVL))
        CALL Q8XORV(X'00',,ISP(IU,J;IVL),,NBR(1,1;IVL),,NBR(1,1;IVL))
      ELSE
        CALL Q8XORV
     +  (X'00',,ISP(IU,J;IVL),,ISP(IU,J+1;IVL),,NBR(1,1;IVL))
      ENDIF
C SECOND NEIGHBOR.
      IF (J.EQ.1) THEN
        CALL Q8SHIFTV(X'08',,ISP(IU,IWIDE;IVL),,I63,,NBR(1,2;IVL))
        CALL Q8XORV(X'00',,ISP(IU,J;IVL),,NBR(1,2;IVL),,NBR(1,2;IVL))
      ELSE
        CALL Q8XORV
     +  (X'00',,ISP(IU,J;IVL),,ISP(IU,J-1;IVL),,NBR(1,2;IVL))
      ENDIF
C THIRD NEIGHBOR.
      CALL Q8XORV(X'00',,ISP(IU,J;IVL),,ISP(IN,J;IVL),,NBR(1,3;IVL))
C SET UP FOR NEIGHBORS 4,5 AND 6.
      IF (IO.EQ.1) THEN
        N4=1
        NV4=2
        N5=IVL-IS+1
        NV5=1
        N6=1
        NV6=IL+1
      ELSE
        N4=IVL
        NV4=1
        N5=1
        NV5=IS+1
        N6=IVL-IL+1
        NV6=1
      ENDIF
C FOURTH NEIGHBOR.
      NBR(N4,4)=XOR(ISP(1,J),ISP(IMAX,J))
      CALL Q8XORV
     +  (X'00',,ISP(2,J;IVL4),,ISP(IVL+1,J;IVL4),,NBR(NV4,4;IVL4))
C FIFTH NEIGHBOR.
      CALL Q8XORV
     +  (X'00',,ISP(IVL+1,J;IS),,ISP(IVL-IS+1,J;IS),,NBR(N5,5;IS))
      CALL Q8XORV
     +  (X'00',,ISP(IVL+IL,J;IVL5),,ISP(1,J;IVL5),,NBR(NV5,5;IVL5))
C SIXTH NEIGHBOR.
      CALL Q8XORV
     +  (X'00',,ISP(IMAX-IS,J;IL),,ISP(1,J;IL),,NBR(N6,6;IL))
      CALL Q8XORV
     +  (X'00',,ISP(IVL+1,J;IVL6),,ISP(IL+1,J;IVL6),,NBR(NV6,6;IVL6))
C SUBTRACT 3 FROM THE DEMONS.
      CALL Q8XORV(X'00',,ID1(1;IVL),,ID2(1;IVL),,IBIT2(1;IVL))
      CALL Q8NANDV(X'03',,ID1(1;IVL),,ID1(1;IVL),,IBIT1(1;IVL))
      CALL Q8NANDV(X'03',,ID1(1;IVL),,ID2(1;IVL),,IREJ(1;IVL))
C ADD ANTIPARALLEL NEIGHBORS.
      DO 2 N=1,6
      CALL Q8ANDV(X'01',,IBIT1(1;IVL),,NBR(1,N;IVL),,ICARRY(1;IVL))
      CALL Q8XORV(X'00',,IBIT1(1;IVL),,NBR(1,N;IVL),,IBIT1(1;IVL))
      CALL Q8ANDV(X'01',,IBIT2(1;IVL),,ICARRY(1;IVL),,NBR(1,N;IVL))
      CALL Q8XORV(X'00',,IREJ(1;IVL),,NBR(1,N;IVL),,IREJ(1;IVL))
      CALL Q8XORV(X'00',,IBIT2(1;IVL),,ICARRY(1;IVL),,IBIT2(1;IVL))
  2   CONTINUE
C ACCEPT CHANGES.
      CALL Q8XORNV(X'07',,ISP(IU,J;IVL),,IREJ(1;IVL),,ISP(IU,J;IVL))
      CALL Q8ANDV(X'01',,ID1(1;IVL),,IREJ(1;IVL),,ID1(1;IVL))
      CALL Q8ANDNV(X'06',,IBIT1(1;IVL),,IREJ(1;IVL),,IBIT1(1;IVL))
      CALL Q8ORV(X'02',,ID1(1;IVL),,IBIT1(1;IVL),,ID1(1;IVL))
      CALL Q8ANDV(X'01',,ID2(1;IVL),,IREJ(1;IVL),,ID2(1;IVL))
      CALL Q8ANDNV(X'06',,IBIT2(1;IVL),,IREJ(1;IVL),,IBIT2(1;IVL))
      CALL Q8ORV(X'02',,ID2(1;IVL),,IBIT2(1;IVL),,ID2(1;IVL))
C HOP DEMONS IN THE Z DIRECTION.
      CALL Q8SHIFTV(X'08',,ID1(1:IVL),,I21,,ID1(1;IVL))
      CALL Q8SHIFTV(X'08',,ID2(1;IVL),,I43,,ID2(1;IVL))
C ACCUMULATE DEMON ENERGIES. ADJUST SECOND VARIABLE IN IBCOUNT
C TO GET DESIRED COMPROMISE OF ACCURACY AND UPDATING SPEED.
C APPROPRIATELY ADJUST NORMALIZATION IN CALLING PROGRAM. FASTEST
C UPDATING OCCURS WITH SMALL LENGTH WHILE LENGTH OF IVL COUNTS ALL
C DEMON ENERGIES.
      IE=IE+IBCOUNT(ID1(1),5)+2*IBCOUNT(ID2(1),5)
  1   CONTINUE
      RETURN
      END


       SUBROUTINE ENERGY(ISP,IX,IMAX,IWIDE,ID1,ID2)
C THIS SUBROUTINE CALCULATES AVERAGE BOND AND DEMON ENERGIES.
       DIMENSION ISP(16512,6),NBR(8256,6),ID1(8256),ID2(8256)
       IELAT=0
       IED=0
       IVL=IMAX/2
       I1=1
       I63=63
       IS=(IX-1)/2
       IL=IX-IS
       IVL4=IVL-1
       IVL5=IVL-IS
       IVL6=IVL-IL
       DO 1 J=1,IWIDE
       DO 1 IO=1,2
       IU=1+(IO-1)*IVL
       IN=1+(2-IO)*IVL
C FIRST NEIGHBOR.
       IF (J.EQ.IWIDE) THEN
         CALL Q8SHIFTV(X'08',,ISP(IU,1;IVL),,I1,,NBR(1,1;IVL))
         CALL Q8XORV(X'00',,ISP(IU,J;IVL),,NBR(1,1;IVL),,NBR(1,1;IVL))
       ELSE
         CALL Q8XORV
     C  (X'00',,ISP(IU,J;IVL),,ISP(IU,J+1;IVL),,NBR(1,1;IVL))
       ENDIF
C SECOND NEIGHBOR.
       IF (J.EQ.1) THEN
         CALL Q8SHIFTV(X'08',,ISP(IU,IWIDE;IVL),,I63,,NBR(1,2;IVL))
         CALL Q8XORV(X'00',,ISP(IU,J:IVL),,NBR(1,2;IVL),,NBR(1,2;IVL))
       ELSE
         CALL Q8XORV
     C  (X'00',,ISP(IU,J;IVL),,ISP(IU,J-1;IVL),,NBR(1,2;IVL))
       ENDIF
```

```
C THIRD NEIGHBOR.
      CALL Q8XORV(X'OO',,ISP(IU,J;IVL),,ISP(IN,J;IVL),,NBR(1,3;IVL))
C SET UP FOR NEIGHBORS 4, 5, AND 6.
      IF (IO.EQ.1)THEN
        N4=1
        NV4=2
        N5=IVL-IS+1
        NV5=1
        N6=1
        NV6=IL+1
      ELSE
        N4=IVL
        NV4=1
        N5=1
        NV5=IS+1
        N6=IVL-IL+1
        NV6=1
      ENDIF
C FOURTH NEIGHBOR.
      NBR(N4,4)=XOR(ISP(1,J),ISP(IMAX,J))
      CALL Q8XORV
    C  (X'OO',,ISP(2,J;IVL4),,ISP(IVL+1,J;IVL4),,NBR(NV4,4;IVL4))
C FIFTH NEIGHBOR.
      CALL Q8XORV
    C  (X'OO',,ISP(IVL+1,J;IS),,ISP(IVL-IS+1,J;IS),,NBR(N5,5;IS))
      CALL Q8XORV
    C  (X'OO',,ISP(IVL+IL,J;IVL5),,ISP(1,J;IVL5),,NBR(NV5,5;IVL5))
C SIXTH NEIGHBOR.
      CALL Q8XORV
    C  (X'OO',,ISP(IMAX-IS,J;IL),,ISP(1,J;IL),,NBR(N6,6;IL))
      CALL Q8XORV
    C  (X'OO',,ISP(IVL+1,J;IVL6),,ISP(IL+1,J;IVL6),,NBR(NV6,6;IVL6))
C ADD ANTIPARALLEL NEIGHBORS.
      DO 2 N=1,6
2     IELAT=IELAT+IBCOUNT(NBR(1,N),IVL)
1     CONTINUE
C ACCUMULATE DEMON ENERGIES.
      IED=IED+IBCOUNT(ID1(1),IVL)+2*IBCOUNT(ID2(1),IVL)
C NORMALIZE ENERGIES.
      ELAT=IELAT/(3.*IMAX*IWIDE*64)
      ED=1.*IED/(1.*IVL*64)
      ETOT=ELAT+2.*ED/(IWIDE*3.)
      PRINT 100,ELAT,ED
      PRINT 101,ETOT
100   FORMAT(1X,'AVERAGE BOND = ',F8.6,' AVERAGE ED = ',F8.6)
101   FORMAT(1X,'TOTAL ENERGY PER BOND = ',F8.6)
      RETURN
      END



      FUNCTION BETA(ED)
C THIS FUNCTION FINDS BETA GIVING AVERAGE DEMON ENERGY ED.
      F(X)=(X+2*X**2+3*X**3)/(1+X+X**2+X**3)
      XL=0
      XH=1.5
      DO 1 N=1,50
      XN=(XL+XH)*.5
      FN=F(XN)
      IF(FN.GT.ED) XH=XN
    1 IF(FN.LT.ED) XL=XN
      BETA=-ALOG(XN)/4
      RETURN
      END



      FUNCTION IBCOUNT(X,IVL)
C THIS FUNCTION COUNTS THE SET BITS IN IVL WORDS STARTING WITH X.
      DIMENSION X(10)
      IBCOUNT=0
      NLEFT=64*IVL
      ISTART=1
    1 IF (NLEFT.LT.1023*64) GOTO 2
      CALL Q8CNTO(X(ISTART;1023*64),,IC)
      IBCOUNT=IBCOUNT+IC
      ISTART=ISTART+1023
      NLEFT=NLEFT-1023*64
      GOTO 1
    2 CALL Q8CNTO(X(ISTART;NLEFT),,IC)
      IBCOUNT=IBCOUNT+IC
      RETURN
      END



       SUBROUTINE CORX(ISP,IMAX,IWIDE,N,IC)
C
C COUNTS ANTIPARALLEL SPINS SEPARATED BY N SITES
C IN THE X DIRECTION AND PLACES RESULTS IN IC.
C ONLY EVEN VALUES OF N ARE CONSIDERED HERE.
C
C FOR SEPARATION 'S' IN Y DIRECTION N MUST EQUAL S*IX.
C
       DIMENSION ISP(16512,6),IT(16512)
C
       IVL=IMAX/2
       IVLP1=IVL+1
       IC=0
       N2=N/2
       IVLMN2=IVL-N2
       DO 1 J=1,IWIDE
       CALL Q8XORV(X'OO',,ISP(1,J;IVLMN2),,ISP(N2+1,J;IVLMN2),,
     C             IT(1;IVLMN2))
       CALL Q8XORV(X'OO',,ISP(IVLMN2+1,J;N2),,ISP(1,J;N2),,
     C             IT(IVLMN2+1;N2))
       CALL Q8XORV(X'OO',,ISP(IVLP1,J;IVLMN2),,ISP(IVLP1+N2,J;IVLMN2),,
     C             IT(IVLP1;IVLMN2))
       CALL Q8XORV(X'OO',,ISP(IMAX-N2+1,J;N2),,ISP(IVLP1,J;N2),,
     C              IT(IMAX-N2+1;N2))
       IC=IC+IBCOUNT(IT(1),IMAX)
1      CONTINUE
       RETURN
       END
```

TEST RUN OUTPUT

```
LATTICE SIZE IS 129 BY  128 BY  384
AVERAGE BOND =  .750572 AVERAGE ED =  .000000
TOTAL ENERGY PER BOND =  .750572


OVER 2000 ITERATIONS
RUNNING AT 116.8 MEGAFLIPS
AVERAGE BOND =  .685223 AVERAGE ED =  .588139
TOTAL ENERGY PER BOND =  .750572
BETA = .220077


OVER 2000 ITERATIONS
RUNNING AT 116.8 MEGAFLIPS
AVERAGE BOND =  .685307 AVERAGE ED =  .587387
TOTAL ENERGY PER BOND =  .750572
BETA = .218410


OVER 2000 ITERATIONS
RUNNING AT 116.8 MEGAFLIPS
AVERAGE BOND =  .685061 AVERAGE ED =  .589596
TOTAL ENERGY PER BOND =  .750572
BETA = .219922


OVER 2000 ITERATIONS
RUNNING AT 116.9 MEGAFLIPS
AVERAGE BOND =  .685199 AVERAGE ED =  .588351
TOTAL ENERGY PER BOND =  .750572
BETA = .219266


OVER 2000 ITERATIONS
RUNNING AT 116.9 MEGAFLIPS
AVERAGE BOND =  .685109 AVERAGE ED =  .589162
TOTAL ENERGY PER BOND =  .750572
BETA = .218770


 *** AVERAGE AFTER DISCARDING FIRST BATCH ***
AV. BETA=  .2190919158126 +/-  .328E-03
```