Journal of Statistical Physics, Vol. 28, No. 4, 1982

Monte Carlo Technique for Very Large Ising Models

C. Kalle[1] **and V. Winkelmann**[1]

Received February 18, 1982

Rebbi's multispin coding technique is improved and applied to the kinetic Ising model with size $600*600*600$. We give the central part of our computer program (for a CDC Cyber 76), which will be helpful also in a simulation of smaller systems, and describe the other tricks necessary to go to large lattices. The magnetization M at $T = 1.4 * T_c$ is found to decay asymptotically as $\exp(-t/2.90)$ if t is measured in Monte Carlo steps per spin, and $M(t = 0) = 1$ initially.

KEY WORDS: Monte Carlo; Glauber kinetic Ising model; Multispin coding; CDC computers.

1. INTRODUCTION

The Ising magnet on a simple cubic lattice serves as a model for many cooperative phenomena. Its simulation on a computer uses the Metropolis Monte Carlo method,[(1)] usually with Glauber kinetics.[(1)] Rebbi and co-workers[(2)] improved the efficiency of this computer simulation by "multispin coding"; a complete program was published by Zorn *et al.*,[(2)] and was further improved by Ottavi and Chakrabarti *et al.*[(4)] These latter authors used it also for a simulation of large lattices up to $360*360*360$. The present work started from the program used in that paper,[(4)] reduced its computer time by about 40%, and applied it to larger systems. Basically we made a feasibility study only; the application to unsolved problems is left for the future.

In CDC 6000 and 7000 series computers each memory unit ("word") has 60 bits; in a (spin-1/2) Ising model, one bit is sufficient to represent a single spin. Multispin coding allows three bits for every spin and thus

[1] University of Cologne, Computing Center, Robert–Koch–Strasse 10, D-5000 Cologne 41, West Germany.

0022-4715/82/0800-0639$03.00/0 © 1982 Plenum Publishing Corporation

makes it possible to count the number of antiparallel neighbor spins, from zero to six in the simple cubic lattice, with these three bits. Thus 20 spins (or 20 interaction energies) can be stored in one word and calculations for these can be made with one word-oriented instruction. Therefore, compared with conventional one-word-per-spin methods,[1] the memory required is reduced by a factor of 20, the computer time by a factor of 2 (Zorn *et al.*[2]). Basically the program performs an exclusive-or operation (XOR) between a word representing 20 spins and their neighbors; the result gives in each of the 20 three-bit parcels the number of antiparallel neighbors,[2] and thus the interaction energy. Thereafter each spin has to be flipped separately according to the probability with a random number (FORTRAN function RANF) distributed homogeneously between zero and one. (For computers with 32-bit words, the advantages of multispin coding are less drastic.)

In the following section we describe the tricks to speed up computation in our CDC Cyber 76 M and include a list of that part of the program. Section 3 describes how a hierarchy of three storage devices is used for the simulation of systems larger than $100*100*100$. Section 4 gives and discusses our results.

2. THE CENTRAL ALGORITHM

For each Monte Carlo step per spin, the computer program examines regularly every spin with coordinates i, j, k and flips and checks if it should be flipped ($i, j, k = 1, 2, \ldots, L$). The outermost loop with variable K is left out in our program listing for simplicity (see Section 3), the second loop (DO 1 $J = 1, L$) starts our program listing in Table I. As the reader may see, periodic boundary conditions were employed. The innermost loop is reduced to a loop over $LL = L/20$ steps only, since in multispin coding we will treat with $II = 1$ the 20 spins $I = 1,\ 1 + LL,\ 1 + 2*LL, \ldots, 1 + 19*LL$, with $II = 2$ the 20 spins $I = 2, 2 + LL, 2 + 2*LL, \ldots, 2 + 19*LL$, until finally with $II = LL$ we treat the 20 spins $I = LL,\ 2*LL,\ 3*LL, \ldots, 20*LL (= L)$. The two statements after labels 10 and 11 incorporate the special cases $II = 1$ and $II = LL$, where the left or right neighbors in the lattice row are not characterized by $II \pm 1$ but have to be formed by shift operations.[3] Essentially we first calculate, apart from differences due to periodic boundary conditions, the sum of the six neighbors and the central spin:

$$IEN = IS(II-1,J,K) + IS(II+1,J,K) + IS(II,J-1,K) \\ + IS(II,J+1,K) + IS(II,J,K-1) + IS(II,J,K+1)$$

For reasons of programming technique we denote $K+1$ and $K-1$ by the

Table I. Central Part of Computer Program, Showing for One Sweep through the Lattice the Loops in Two of Three Dimensions in the L * L * L Lattice[a]

```
      DO 1 J = 1, L
      JP1 = J + 1 $ IF(JP1.GT.L) JP1 = 1
      JM1 = J − 1 $ IF(JM1.EQ.0) JM1 = L
      DO 1 II = 1, LL
      IF(II.EQ.1) GOTO 10
      IF(II.EQ.LL) GOTO 11
      IEN = ISC(II + 1, J) + ISC(II − 1, J)
      GOTO 12

10    CONTINUE
      IEN = ISC(2, J) + SHIFT(ISC(LL, J), 57)
      GOTO 12

11    CONTINUE
      IEN = ISC(LLM1, J) + SHIFT(ISC(1, J), 3)
12    ICI = ISC(II, J)
      IEN = IEN + ISC(II, JM1) + ISC(II, JP1) +
      .    IS(II, J, PLLCMU) + IS(II, J, PLLCML) + ICI
      IEN = XOR(ICI.OR.SHIFT(ICI, 1).OR.SHIFT(ICI, 2), IEN) + IEN1
C     IEN COUNTS FOR THE 20 SPINS IN ICI HOW MANY OF THEIR
C     NEIGHBORS ARE ANTIPARALLEL TO THEM (EXCLUSIVE OR)
C     ICH MARKS THOSE SPINS WHICH HAVE TO BE FLIPPED
      ICH = 0
      DO 2 J1 = 1, 10
      IF(RANF(J1).LE.EX(IEN.AND.7)) ICH = ICH.OR.7
      ICH = SHIFT(ICH, 3)
      IEN = SHIFT(IEN, 3)
      IF(RANF(J1).LE.EX(IEN.AND.7)) ICH = ICH.OR.7
      ICH = SHIFT(ICH, 3)
2     IEN = SHIFT(IEN, 3)
C     SHIFT PRODUCES A CIRCULAR SHIFT TO THE LEFT BY 3 BITS
      ICI = ISC(II, J) = XOR(ICI, ICH.AND.IEN1)
      M = M + COUNT(ICI)
C     COUNT COUNTS THE NUMBER OF SET BITS IN A COMPUTER WORD
1     CONTINUE
```

[a] RANF is a random number generator and EX(n) the probability to flip a spin if $n-1$ of its neighbors are antiparallel. Basically, PLLCMU = $K-1$ and PLLCML = $K+1$ if the investigated plane K is stored in ISC(II, J). This part can be applied to smaller systems, at least $L = 60$. IEN1 adds unity to every 3-bit parcel.

integer "plane-in-large-core-memory (LCM)" variables PLLCML and PLLCMU (see Section III), and we store central plane K also in small-core-memory, the usual working storage, as a copy of the plane K resident in LCM, such as we can reference $IS(II,J,K)$ by ISC(II,J). ICI is an abbreviation[3] for $IS(II,J,K)$.

The crucial statement

$$\text{IEN} = \text{XOR(ICI.OR.SHIFT(ICI, 1).OR.SHIFT(ICI, 2), IEN)} + \text{IEN1}$$

gives in every 3-bit group the number of antiparallel neighbors of the central spin in ICI, as the reader may check in an example. [The two "OR"-operations give 111 for every up spin (001) and 000 for every down spin (000).] This method works if each spin interacts with six neighbors and has to be modified for different numbers of interacting neighbors. The addition of IEN1 to this result adds unity to every number of antiparallel neighbors; this IEN1 has 001 in each of its 20 3-bit parcels. Earlier work[2–4] used an XOR operation for each of the six directions of interaction, which is less effective.

The statements "IF (RANF(J1).LE.EX(IEN.AND.7)) ICH = ICH. OR.7" will result in ORing a 7 (= 111) to the changer word ICH if the spin has to be flipped, and a 000 otherwise. Since the number of hardware registers in the computer is exhausted by this loop construction, it was not useful to mark spins to be flipped by an additional constant 1 (= 001), which would be more convenient for the flip process. In contrast to earlier work [2–4], we investigated two spins during one trip of loop labeled 2, since the small number of instructions allows the loop still to reside in the Cyber 76 instruction stack. In order to get a feeling for the time consumed in this loop, the reader may consider that each machine instruction in this innermost loop will be executed 216.000.000 times for each Monte Carlo step per spin (600 * 600 * 600 system). The cycle time of a Cyber 76 computer is 27.5 nsec, thus to save one cycle's time in this loop means saving 5.8 sec per Monte Carlo step. So we took great care to optimize the code of this loop. Thereafter the statement ICI = XOR (ICI, ICH. AND.IEN1) produces the changed spin word; and finally M = M + COUNT(ICI) counts the number of spins 001 in the changed configuration and so gives the magnetization more simply than in earlier versions.

We found that in our test runs for 240**3 the CPU time for one Monte Carlo step was reduced to 14 sec compared with 22 sec in the program of Chakrabarti *et al.*[4] given to us. For the 600**3 system we needed less than 230 sec for one sweep through the lattice. Another factor of 3 can be gained in speed if one takes $\exp(-4J/k_B T) = 1/2$ corresponding to $T/T_C = 1.279$ and treats 20 spin flips simultaneously[5].

3. HIERARCHY OF MEMORIES FOR VERY LARGE SYSTEMS

Chakrabarti *et al.*[4] used already disk storage to store larger systems up to 360**3, with test runs up to 600**3. Information to and from the disk storage can be transferred asynchronously by the FORTRAN "BUFFER IN" and "BUFFER OUT" statements; using this method a program may perform computation and input/output at the same time. We corrected an error in the given program[4], due to which parts of the boundary spins were always down, and further refined the method.

We used the three available types of memory in a strongly hierarchical way: SCM, the small-core memory; LCM, the large-core memory; and RMS, the disk storage system. As mentioned above, the plane K is stored with dimension ISC(LL, L) in SCM, this plane K, its neighbors $K+2$, $K+1$, $K-1$ and the first plane $K=1$ are stored in direct access LCM ("LEVEL 2") in the area IS($LL, L, 5$). Information between SCM and LCM was transferred by the fastest possible method, the block copy (FORTRAN "MOVLEV" routine). The third index n of IS(II, J, n) was denoted as $n=$ PLSCM for the central plane K, as $n=$ PLLCMU for the upper neighbor plane $K-1$, $n=$ PLLCML for the lower neighbor plane $K+1$, and $n=$ PLLCMI for the input plane $K+2$ read in from disk storage.

At this point, it may be interesting to the reader, in order to understand the following, to learn a little about single-word access to LCM in the Cyber 76 computer.

If a program references a word in LCM this and the following 15 words are read from core into a hardware register. Successive references to words following that word are satisfied from that register, which is quite fast. So only every 16 times a real storage access is required, if consecutive words are read using single-word access to LCM. This is the clue to make LCM access fast. Since the minimum LCM configuration consists of two banks of LCM, each having such a hardware register, this configuration is presently available at the Cologne Cyber 76, the only possibility to use LCM in the fastest way is to read the PLLCML plane using one bank and to read the PLLCMU plane using the other bank. Thus computation time is optimal if the product of $LL = L/20$ and L is an odd multiple of 16, allowing $L = 200$, 280, 360, 440, 520, 600, and 680 for the Cologne configuration. ($L = 120$ and smaller systems can be computed without RMS usage.)

All L planes are stored on RMS, accessed by "BUFFER IN" and "BUFFER OUT" statements. We did not test if we could fit 680**3 into the disk storage system under normal multiuser conditions; for 600**3 we consumed already nearly one fifth of the total available RMS space, that is about 20 million (60-bit) words. (One may save memory by a factor of 3, losing in speed by a factor of 1.6, if one compresses three 20-spin words into one before storing them on the disk. Only test runs were made, however.[5]) It is not sufficient to store only the planes $K - 1$ and $K + 1$ in LCM, because, as we mentioned above, we want to do asynchronous input/output, so during the time the lowest plane $K + 2$ is read in from RMS by "BUFFER IN," the plane $K + 1$ is already needed for computation, and during the time the uppermost plane $K - 1$ is output to RMS, its LCM storage can be used by spins of the upper plane. Table III explains this usage of plane indices. Effectively three processes are done at a time: input of a new plane, computation, and output of an already newly computed plane.

When we have investigated one plane, we move to the lower neighboring plane not by shifting the spins IS(II, J, n) within storage but merely by rotating the pointers (third index n), e.g., setting PLSCM = PLLCML. If these variables are set up as PLLCMU = 4, PLSCM = 2, PLLCML = 1, and PLLCMI = 3 the single-word access to LCM via the two banks is used best. Table II indicates schematically the way the information is shifted between memories. Additional statements are required for the special cases $K = 1$, $K = L - 1$, $K = L$, $K = L + 1$ and during the first Monte Carlo sweep through the lattice. After each such Monte Carlo step per spin, the magnetization $(M - L{*}{*}3/2)/L{*}{*}3$ is printed along with other useful information.

To protect the computation against a breakdown of the computer, the (very limited) output was also printed on the dayfile (which is usually not lost), and the RMS information was saved for later reuse. Thus we could

Table II. Schematic Listing of Storage Handling Statements

```
      M = -L3/2
      DO 8 K = 1, L
      CALL MOVLEV(IS(1, 1, PLSCM), ISC(1, 1), LLL)
        .
        .
      Computation as in Table I
        .
        .
      CALL MOVLEV(ISC(1, 1), IS(1, 1, PLSCM), LLL)
C        MOVE SCM PLANE BACK TO LCM
      SCRATCH = PLLCMU
      PLSCM = PLLCML
      PLSAVEI = PLLCMI
      PLLCML = PLONE
      PLLCMI = SCRATCH
      BUFFEROUT (OTAPE, 1) (IS(1, 1, PLLCMU), IS(LL, L, PLLCMU))
      BUFFERIN (ITAPE, 1) (IS(1, 1, PLLCMI), IS(LL, L, PLLCMI))
8     CONTINUE
```

Table III. Sketch of Plane Index Usage for Asynchronous Disk Input/Output during Computation

Disk storage	
--------------------------------	PLLCMU (output)
--------------------------------	PLSCM (used in SCM)
--------------------------------	PLLCML
--------------------------------	PLLCMI (input)
Disk storage	

stretch the calculation over an extended period of time, and were not forced to compute all steps during one long run. No special arrangements with the general operating system were necessary, our program ran under usual conditions, as did numerous other programs at that time.

4. RESULTS

Figure 1 shows that the magnetization after t Monte Carlo steps per spin at a temperature 40% above the critical temperature. We see that the magnetization decays for large t as $\exp(-t/\tau)$ with $\tau = 2.90$, until finally fluctuations take over. This decay is seen here much better than in Ref. 4, either due to an unfortunate fluctuation there or due to the above-mentioned programming error in Ref. 4. For small times, deviations occur as expected. For the initial slope $-dM/dt$ [at $t = 0$, $M(t = 0) = 1$] is equal to the probability of flipping one spin down when all neighbor spins are up. That probability remains finite at the critical temperature whereas $1/\tau$ vanishes there(4). [The flip probability was taken as

$$e^{-\beta\Delta E}/(1 + e^{-\beta\Delta E})$$

where $\beta = 1/k_B T$ and ΔE is the energy change of the flip.]

Thus we have simulated the decay of magnetization in a very large Ising lattice on a general-purpose computer under normal operating conditions, using less than four minutes central processor time for every Monte Carlo step per spin in a 600*600*600 lattice. While we used many features special to large CDC computers, the whole program was written in FORTRAN. It remains to be seen what new results can be obtained from large systems; for this purpose we can send our complete program (200 statements) to the interested reader.

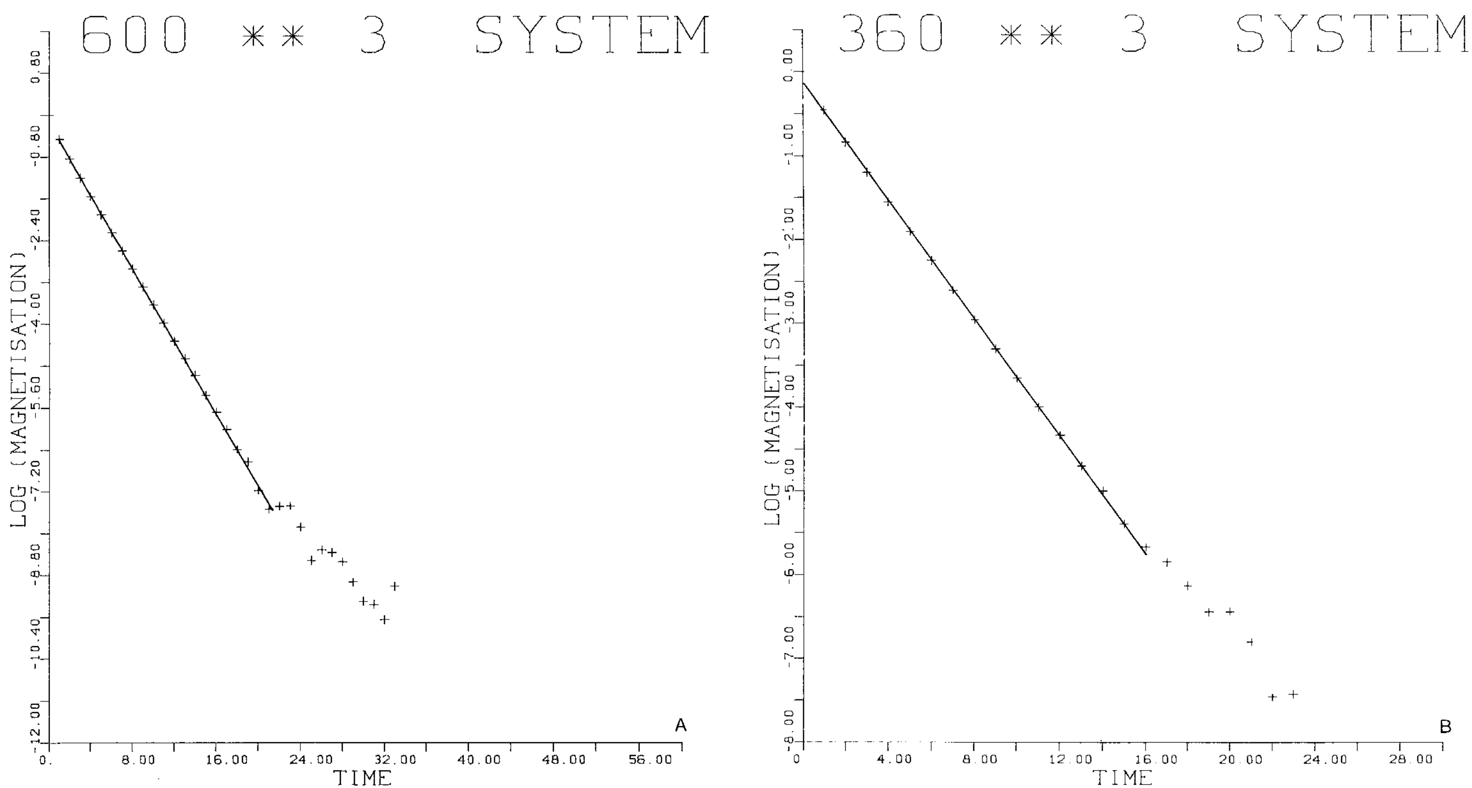

Fig. 1. Decay of magnetization as function of time t (= number of Monte Carlo steps per spin), for 360**3 and 600**3 Ising models at $T/T_c = 1.4$. In both figures we give the natural logarithm, not the decadic one.

```
ABØØ2SØ AF DC=40 FS=0001
              II       IIIIIII     IIIIII     IIIIII     IIIIII     IIIIII     IIIII I
             IIII      II     II  II     II  II     II  II     II  II     II  II     I
            II  II     II     II  II     II  II     II         II  II         II   III
           II    II    IIIIIII    II     II  II     II      IIII   IIIIII     II  I II
           II    II    II     II  II     II  II     II     II           II  II II II
           IIIIIIII    II     II  II     II  II     II    II            II  II I  II
           II    II    II     II  II     II  II     II   II             II  III   II
           II    II    II     II  II     II  II     II   II        II   II    I   II
           II    II    IIIIIII     IIIIII     IIIIII    IIIIIIII    IIIIII   I IIIII


UKLN SCOPE 2.1.5 LVL 533A2 , 28/09/81  21.07.34.. 04/02/82       82035

SYS DEVICES 844/ 7/PF FLS=377K FLL=2000K MXS=170K MXL=400K MXB=1000B

HH.MM.SS CPU SECOND ORIGIN
          22.49.54.MFA, AF     UNI KOELN NOS/BE1.4 LEV. 508 03.02.1982
22.49.31 00000,004 MFZ,       -AB002,STMFZ.
22.49.31 00000,005 MFZ,        ACCOUNT,AB002,*****.
22.49.31 00000,006 LOD,       -FTN,OPT=2.
22.49.34 00000,229 USR,             .221 CP SECONDS COMPILATION TIME
22.49.34 00000,229 LOD,       -LGO.
22.49.35 00000,357 MFZ,          LD610 - FLS REQUIRED TO LOAD - 011445 OU.COG
22.49.37 00000,361 MFZ,          LD603 - EXECUTION INITIATED OS.EXP
22.49.37 00000,362 USR,         FORTRAN LIBRARY 528            25/05/81
22.49.40 00001,669 USR,            STOP
22.49.40 00001,669 USR,            146400   FINAL EXECUTION FL.
22.49.40 00001,669 USR,             1.307 CP SECONDS EXECUTION TIME.
22.49.40 00001,670 JOB,       -EXIT,S.
22.49.40 00001,670 MFZ,          MAX MS                     6   +4KW
22.49.40 00001,671 MFZ,          JM166 - MAXIMUM USER SCM           146400B WORDS
22.49.40 00001,672 MFZ,          JM167 - MAXIMUM USER LCM            300008 WORDS
22.49.40 00001,672 MFZ,          JM170 - MAXIMUM JS+IO LCM             176B BUFFERS
22.49.40 00001,673 MFZ,          SCM                   75.896  KWS
22.49.40 00001,673 MFZ,          LCM                    2.722  KWS
22.49.40 00001,673 MFZ,          I/O                    0.001  MW
22.49.40 00001,673 MFZ,          USER                   1.441  SEC
22.49.40 00001,674 MFZ,          JOB                    1.675  SEC
22.49.40 00001.674 MFZ.          SC050 - 000006 SC/LC SWAPS


          PROGRAM IS3D         76/76    OPT=2                          FTN 4.8+5


                   PROGRAM IS3D(OUTPUT,TAPE5=OUTPUT)
                   DIMENSION IS(5,100,100),EX(7),INDM1(100),INDP1(100)
                   DATA IS/50000*0/,IEN1/11111111111111111111B/
                   T=1.27923/0.221673 $ LL=3 $ CALL RANSET(1) $ L=LL*20$LL
                   DO 1 I=1,L $ INDM1(I)=I-1
          1        INDP1(I)=I+1
                   INDM1(1)=L $ INDP1(L)=1
                   DO 2 I=1,7
          2        EX(I)=EXP((4*I-16)/T)
                   DO 6 ITIME=1,6 $ M=0
                   DO 4 K=1,L $ KM1=INDM1(K) $ KP1=INDP1(K)
                   DO 4 J=1,L $ JM1=INDM1(J) $ JP1=INDP1(J)
                   DO 4 I=1,LL
                   IF(I.EQ.1) GOTO 10 $ IF(I.EQ.LL)      GOTO 11
                   IEN=IS(I-1,J,K)+IS(I+1,J,K)           $ GOTO 12
          10       IEN=IS(2,J,K)+SHIFT(IS(LL,J,K),57) $ GOTO 12
          11       IEN=IS(LLM1,J,K)+SHIFT(IS(1,J,K),3)
          12       ICI=IS(I,J,K)
                   IEN=IEN+ICI+IS(I,JM1,K)+IS(I,JP1,K)+IS(I,J,KM1)+IS(I,J,
                   IEN=IEN1+XOR(ICI.OR.SHIFT(ICI,1).OR.SHIFT(ICI,2),IEN)
                   ICH=0 $ DO 3 II=1,10
                   ICH=SHIFT(ICH,3) $ IEN=SHIFT(IEN,3)
                   IF(RANF(II).LE.EX(IEN.AND.7)) ICH=ICH.OR.7
                   ICH=SHIFT(ICH,3) $ IEN=SHIFT(IEN,3)
          3        IF(RANF(II).LE.EX(IEN.AND.7)) ICH=ICH.OR.7
          4        ICI=IS(I,J,K)=XOR(ICI,ICH.AND.IEN1)
          6        WRITE(5,5) M,ITIME
          5        FORMAT(2I20)
                   STOP $ END
         0                    1
         0                    2
         0                    3
         0                    4
         0                    5
         0                    6

*AB002S0 AF DC=40 FS=0001
```

Fig. 2. A complete program which ran on our Cyber 76 to test computing times of multispin coding in three-dimensional Ising models up to $100 \times 100 \times 100$. It does not calculate E and M since that should be done at the end only.

ACKNOWLEDGMENTS

We thank D. Stauffer for suggestions and help with the manuscript, and H. G. Baumgaertel for a copy of his computer program of Ref. 4.

REFERENCES

1. K. Binder, *Monte Carlo Methods in Statistical Physics* (Springer Verlag, Heidelberg, 1979).
2. M. Creutz, L. Jacobs, and C. Rebbi, *Phys. Rev. Lett.* **42**:1390 (1979); R. Zorn, H. J. Herrmann, and C. Rebbi, *Computer Phys. Commun.* **23**:337 (1981); See also R. Friedberg and J. E. Cameron, *J. Chem. Phys.* **52**:6049 (1970).
3. H. Ottavi, *Z. Phys. B* **44**:203 (1981).
4. B. K. Chakrabarti, H. G. Baumgaertel, and D. Stauffer, *Z. Phys. B* **44**:333 (1981).
5. D. Stauffer, Private Communication.