2 ✓
RECEIVED BY TIC MAR 20 1975

UCID - 16697

This is an informal report intended primarily for internal or limited external distribution. The opinions and conclusions stated are those of the author and may or may not be those of the laboratory.

LAWRENCE LIVERMORE LABORATORY

University of California/Livermore, California

AN EXAMPLE OF CODE OPTIMIZATION

TIMOTHY RUDY

January 1975

MASTER

—NOTICE—

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

Prepared for U.S. Atomic Energy Commission under contract no. W-7405-Eng-48

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED
47

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

AN EXAMPLE OF CODE OPTIMIZATION

Timothy Rudy

January 1975

ABSTRACT

The performance of a fully explicit 1-D hydrodynamics program is presented for the various large scale computers at Lawrence Livermore Laboratory. The effect of programming techniques on performance is also examined through five versions of the program.

AN EXAMPLE OF CODE OPTIMIZATION

This report describes the evolution of an explicit 1-D hydrodynamics program from a basic FORTRAN origin to a vectorized conclusion. The execution time for each version is measured. The appendices consist of the following:

- 1) Appendix 1 version - Original FORTRAN II source
- 2) Appendix 2 version - Change branch logic
- 3) Appendix 3 version - "Better" FORTRAN
- 4) Appendix 4 version - Assembly language
- 5) Appendix 5 version - Vectorized version

The problem executed on all versions of the program consists of 100 zones. A pressure is applied to one boundary and a shock is allowed to propagate for 5001 time steps. Initially, all zones are equally spaced. Consequently, the running time is reduced by avoiding meaningless computations in zones which the shock has not effected.

Table 1 summarizes the time required to execute the problem on different versions of the program. The CHIP [2] (and PUTT) compilers are of CDC origin. The CHAT [1] compiler has been developed at LLL. It should be noted that more recent releases of the CDC compiler produce results comparable to CHAT [3]. The timings which are labeled "optimize" utilize an object-code optimization option available in the CHAT compiler [7] and "in-line sqrt" refers to performing the square root function in-line instead of a subroutine call.

The difference in performance of the CDC 7600 [6] versus the 6600 is well understood. The rather modest improved performance of version two over version one is accomplished by replacing the five three-branch "if tests" in the major loops of the program to a drop through operation.

In general, the use of labels should be kept to a minimum. The use of a STAR class computer for serial programs is obviously disastrous. However, an improved LLL compiler is under development and hopefully will close the differential between the 7600 and STAR. It is unlikely that serial usage of the STAR will improve beyond 1/3 7600 speed unless the register file can be artfully used to eliminate memory references.

The transition to "better FORTRAN" is a little more tedious. Appendix 3 contains the source for this version. In general the procedure used in obtaining this version involves reducing the number of operations within the do loops. For example, to fetch the variable X(KMN) requires two memory references. If X(KMN) is placed in a scalar, XLO, outside the do loop then only a single memory reference is required for X(KMN) within the loop. The following changes were made to the program to obtain the better FORTRAN version.

1) Do Loop 200

P+Q combined in array PQ.

Boundary conditions (PUB, PLB) defined initially and
RZAKI for KMN and KMX multiplied by two.

2) Do Loop 250

Recoded into 2 loops. Hopefully, a compiler will
optimize small loops more efficiently.

Q=FE=0. Moved from Do 350.

3) Do Loop 350

Fetch of X(KMN), X(KMX) moved outside of loop.

DU is stored in P(K) for later use.

Added PQ(K) calculation.

4) Do Loop 370

P+Q replaced by PQ.

Compliment of C2 available as C2M (+-C2).

FE(K) set to zero in Loop 250.

Removed store of WK1.

5) Do Loop 450

Since RZ is only used as a divisor, $RZ+1/RZ$ is defined initially.

$U(K)-U(K-1)$ previously computed and stored in P(K).

P+Q replaced by PQ.

EDOT is multiplied by DT when defined.

Replaced $X(K)-X(K-1)$ in DTCHK calculation with V(K).

The assembly version listed in Appendix 4 is not to be construed as the most optimal coding possible for the 7600. The criteria in designing and coding all versions of this code was that the time involved in changing from one version to the next should be limited to two hours. This does not include debug time. To obtain the optimum performance from the 7600 the instruction stack should be totally occupied with instructions. This would be possible with this program by either combining loops or processing more than one zone per pass through the mesh loop. Despite all of the above disclaimers, the assembly version is a factor or two faster than the FORTRAN. With a little more effort a factor of three would be obtained.

The final example in Appendix 5 is a STACKLIB [4] version of the code. STACKLIB utilized the concurrency and instruction stack features of the 7600 to effectively emulate a subset of the vector class instructions of the STAR. With minor modifications this version can be compiled

using the CHAT or STAR (a version of CHAT on the 7600 which produces STAR object code) compiler with vector extension facilities and execute on either machine. The performance on the 7600 of STACKLIB for short vectors (less than 180 elements) is understandably poor. The use of control vectors to inhibit storing elements in the result vector is similar to incorporating an "if test" with every store operation in a code for the 7600. As noted earlier, cold zones are not computationally significant for the serial version. However, using control vectors does not reduce the computation task for STACKLIB. When the optimization for the initialization phase of the STACKLIB vector routine is completed significant improvements are anticipated for short vector operations.

The timings presented in Table 2 represent the results from the execution of the program for 10 000 time steps and 200 zones. The times specified for the STAR are obtained by using the job interval instructions. The STAR vector version is three times faster than the best 7600 FORTRAN program for the 100 zone problem and five times the 7600 for the 200 zone problem. This is very encouraging.

The major tasks associated with the 7600 to STAR conversion has been slighted in this report. These are boundary conditions, table look up, restructuring data bases and memory trade offs. If your code is in the two dimensional world you might consider vectorizing your 1-D sweeps through the data. It appears that a factor of three is possible with 100 element vectors even though such short vectors degrade the theoretical performance of the STAR by 1/2.

The conclusions to be drawn from this experiment are:

- 1) Recoding FORTRAN to obtain "better FORTRAN" is a logical step in preparing a program for a vector machine.
- 2) STACKLIB is an indispensable tool for preparing codes for the STAR.
- 3) Scalar programs on the STAR will perform poorly.
- 4) Significant improvement in running time for a program can be obtained via programming techniques.

REFERENCES

1. Martin, J. et al., LRLTRAN Language Used With the CHAT and STAR Compilers, Lawrence Livermore Laboratory, Rept. LTSS-207 (1973).
2. Carpenter, E. and Solbeck, S., The Chippewa Compiler, Lawrence Livermore Laboratory, Rept. LTSS-204 (1968).
3. Bailey, Dave, Private Communication.
4. McMahon, F. H., et al., Stacklib-A Vector-Function Library of Optimum Stackloops for the CDC 7600, Lawrence Livermore Laboratory, UCID-33083.
5. Control Data STAR-100 Computer System: Hardware Reference Manual, Control Data Corporation, St. Paul, Minnesota, Publication 60256000 06 (1973).
6. Control Data 7600 Computer System: Reference Manual, Control Data Corporation, St. Paul, Minnesota, Publication 60100000 L (1971).
7. R. G. Zwakenberg, CDC 6600/7600 Optimization Techniques, Lawrence Livermore Laboratory, Rept. UCID-30046 (1972).

AN EXAMPLE OF CODE OPTIMIZATION

5

TABLE 1

(ALL TIMES ARE IN SECONDS)

APPENDIX 1 VERSION

97.0	CHIP-6600
17.7	PUTT-7600
13.8	CHAT-7600
13.2	CHAT-7600-OPTIMIZE

APPENDIX 2 VERSION

78.8	CHAT-6600-OPTIMIZE-INLINE SQRT
78.0	STAR
13.1	CHAT-7600
12.8	CHAT-7600-OPTIMIZE

APPENDIX 3 VERSION

67.9	CHAT-6600-OPTIMIZE-INLINE SQRT
64.5	STAR
12.6	CHAT-7600
11.8	CHAT-7600-OPTIMIZE-
11.3	CHAT-7600-OPTIMIZE-INLINE SQRT

APPENDIX 4 VERSION

6.0	ASSEMBLY-7600
-----	---------------

APPENDIX 5 VERSION

17.3	STACKLIB-7600
3.6	STAR

AN EXAMPLE OF CODE OPTIMIZATION

6

TABLE 2

(ALL TIMES ARE IN SECONDS)

51.2	STACKLIB-7600
45.6	CHAT-7600-OPTIMIZE APPENDIX 3 VERSION
9.8	STAR

AN EXAMPLE OF CODE OPTIMIZATION

7

APPENDIX 1
ORIGINAL VERSION

```

111 CALL TIMR
    CONTINUE
    DO 200 K=KMNP,KMXZ
        ACCEL(K)=- (P(K-1)+Q(K+1)-P(K)-Q(K))*RZAKI(K)
        U(K)=U(K)+DT*ACCEL(K)
200    CONTINUE
        ACCEL(KMX)=-RZAKI(KMX)*(PUB-P(KMX)-Q(KMX))
        *2.
        U(KMX)=U(KMX)+DT*ACCEL(KMX)
        ACCEL(KMN)=0.
        IF (PLB) 210,205,205
205    CONTINUE
        ACCEL(KMN)=-RZAKI(KMN)*(P(KMNP)+Q(KMNP)-PLB)
        *2.
210    CONTINUE
        U(KMN)=U(KMN)+DT*ACCEL(KMN)
        DO 250 K=KMN,KMX
            IF (ABSF(U(K))-UCUT) 220,230,230
220        U(K)=0.
            GOT0240
230        X(K)=X(K)+U(K)*DT
240        W=X(K)-X(K-1)
            R(K)=R(K)*V(K)/W
            V(K)=W
250    CONTINUE
        FNUMX=-1.E+6
        DO 350 K=KMNP,KMX
            Q(K)=FE(K)=0.
            IF (U(K)-U(K-1)) 330,350,350
330        DU=U(K)-U(K-1)
            WK4=MIN1F(X(K)-X(KMN), X(KMX)-X(K-1))
            WK1=R(K)*(CSQ*DU - C1*SEUND(K))
            WK1=WK1/(1.+C3*(X(K)-X(K-1))/WK4)
            Q(K)=WK1*DU
350    CONTINUE
C
        DO 370 K=KMNP,KMXZ
            WK2=ABS(P(K+1)+Q(K+1)-P(K)-Q(K))
            FE(K)=0.
            IF (WK2-1.E-10) 370,370,365
365    CONTINUE
            WK3=.5*(R(K+1)+R(K))
            WK1=SQRT(WK3*WK2)
            FE(K)=-C2*WK1*(E(K+1)-E(K))
370    CONTINUE
C
        FE(KMN)=FE(KMX)=0.
        DTCHK=1.E+6
        DO 450 K=KMNP,KMX

```

AN EXAMPLE OF CODE OPTIMIZATION

8

```

      EDOT=-(FE(K)-FE(K-1))/RZ(K)
      W=DT*(U(K)-U(K-1))/RZ(K)
      WE=E(K)-(P(K)+Q(K))*W
1    +EDOT*DT
      WP=GAMZ*WE*R(K)
      WP=MAX1F(WP, 0.)
      E(K)=E(K)-0.5*(P(K)+WP+Q(K)+Q(K))*W
1    +EDOT*DT
      E(K)=MAX1F(E(K), 1.E-30)
      P(K)=GAMZ*E(K)*R(K)
      SOUND(K)=0.
      IF(P(K)) 450,450,400
400   SOUND(K)=SQRT(GAM*P(K)/R(K))
      DTCHK=MIN1F((X(K)-X(K-1))/SOUND(K), DTCHK)
450   CONTINUE
      TIME=TIME+DT
      NYCL=NYCL+1
      IF(DT-DTCHK) 475,460,460
460   WRITE(NTED,462) NYCL,TIME,DT,DTCHK
462   FORMAT(*UNSTABLE *,I6,3E10.2, * LF TO DT/2*)
      GO TO 950
475   CONTINUE
490   CONTINUE
      IF(TIME-TIMEUP) 505,505,600
505   IF(TIME-EDTIME) 111,600,600
600   EDTIME=EDTIME+EDDT
      CALL TIMR
      WP=0.
      DO 603 K=KMNP,KMX
603   WP=WP+P(K)
      WRITE(NTTY,604)NYCL,WP
604   FORMAT(I6,E16.8)
      IF(TIME-TIMEUP)111,111,601
601   CONTINUE

```

AN EXAMPLE OF CODE OPTIMIZATION

9

APPENDIX 2
BRANCH LOGIC CHANGED

```

      CALL TIMR      $$C INITIALIZE TIMING
      IF (TIME-EDTIME), 601, 601
C EDIT INITIAL CONDITION
111  CONTINUE
      DO 200 K=KMNP, KMNZ
      ACCEL(K) = -(P(K+1)+Q(K+1)-P(K)-Q(K))*RZAKI(K)
      U(K) = U(K) + DT*ACCEL(K)
200  CONTINUE
      ACCEL(KMX) = -RZAKI(KMX)*(PUB-P(KMX)-Q(KMX))
      *2.
      U(KMX) = U(KMX) + DT*ACCEL(KMX)
      ACCEL(KMN) = 0.
      IF (PLB) 210,
      ACCEL(KMN) = -RZAKI(KMN)*(P(KMNP)+Q(KMNP)-PLB)
      *2.
210  CONTINUE
      U(KMN) = U(KMN) + DT*ACCEL(KMN)
      DO 250 K=KMN, KMX
      IF (ABSF(U(K))-UCUT) , 230, 230
      U(K) = 0.
      GO TO 240
230  X(K) = X(K) + U(K)*DT
240  W = X(K) - X(K-1)
      R(K) = R(K)*V(K)/W
      V(K) = W
250  CONTINUE
      FNUMX = -1.E+6
      DO 350 K=KMNP, KMX
      Q(K) = FE(K) = 0.
      IF (U(K)-U(K-1)) , 350, 350
      DU = U(K) - U(K-1)
      WK4 = MIN1F(X(K)-X(KMN), X(KMX)-X(K-1))
      WK1 = R(K)*(CSQ*DU - C1*SQND(K))
      WK1 = WK1/(1.+C3*(X(K)-X(K-1))/WK4)
      Q(K) = WK1*DU
350  CONTINUE
C
      DO 370 K=KMNP, KMNZ
      WK2 = ABS(P(K+1)+Q(K+1)-P(K)-Q(K))
      FE(K) = 0.
      IF (WK2-1.E-10) 370, 370,
      WK3 = .5*(R(K+1)+R(K))
      WK1 = SQRT(WK3*WK2)
      FE(K) = -C2*WK1*(E(K+1)-E(K))
370  CONTINUE
C
      FE(KMN) = FE(KMX) = 0.
      DTCHK = 1.E+6
      DO 450 K=KMNP, KMX

```


AN EXAMPLE OF CODE OPTIMIZATION

10

```

      EDOT=-(FE(K)-FE(K-1))/RZ(K)
      W=DT*(U(K)-U(K-1))/RZ(K)
      WE=E(K)-(P(K)+Q(K))*W
1      +EDOT*DT
      WP=GAMZ*WE*R(K)
      WP=MAX1=(WP, 0.)
      E(K)=E(K)-0.5*(P(K)+WP+Q(K)+Q(K))*W
1      +EDOT*DT
      E(K)=MAX1F(E(K), 1.E-30)
      P(K)=GAMZ*E(K)*R(K)
      SOUND(K)=0.
      IF(P(K)) 450,450,
      SOUND(K)=SQRT(GAM*P(K)/R(K))
      DTCHK=MIN1F((X(K)-X(K-1))/SOUND(K), DTCHK)
450    CONTINUE
      TIME=TIME+DT
      NYCL=NYCL+1
      IF(DT-DTCHK)475,,
460    WRITE(NTED,462) NYCL,TIME,DT,DTCHK
462    FORMAT("UNSTABLE ",I6,3E10.2, " LF TO DT/2")
      GO TO 950
475    CONTINUE
490    CONTINUE
      IF(TIME-TIMEUP),,600
      IF(TIME-EDTIME)111,,
600    EDTIME=EDTIME+EDDT
      CALL TIMR
      WP=0.
      DO 603 K=KMNP,KMX
      WP=WP+P(K)
603    CONTINUE
      WRITE(NITY,604)NYCL,WP
604    FORMAT(I5,E16.8)
      IF(TIME-TIMEUP)111,111,601
601    CONTINUE

```

AN EXAMPLE OF CODE OPTIMIZATION

11

APPENDIX 3
BEST FORTRAN VERSION

```

      CALL TIMR      $$C INITIALIZE TIMING
      IF (TIME-EDTIME), 601, 601
C EDIT INITIAL CONDITION
111  CONTINUE
C
C LOOP 1
      DO 200 K=KMN,KMX
      COMPUTE ACCELERATION AND VELOCITY
C EXTERNAL PRESSURES PUB, PLB FACTORED IN
      ACCEL(K)=RZAKI(K)*(FQ(K+1)-PQ(K))
      U(K)=U(K)+DT*ACCEL(K)
200  CONTINUE
C
      OP2C LO
      DO 250 K=KMN,KMX
C ELIMINATE NOISE DUE TO SMALL VELOCITIES
      Q(K)=FE(K)=0.
      IF (ABSF(U(K))-UCUT),, 250
      U(K)=0.
250  CONTINUE
C
C LOOP 3
      DO 260 K=KMN,KMX
C ADVANCE POSITION      X(N+)=X(N)+U(N+)
C COMPUTE NEW VOLUME (V) AND DENSITY (RH0)
      X(K)=X(K)+U(K)*DT
      W=X(K)-X(K-1)
      R(K)=R(K)*V(K)/W
      V(K)=W
260  CONTINUE
      XL0=X(KMN)
      XH1=X(KMX)
C
C LOOP 4
      DO 350 K=KMNP,KMX
      PQ(K)=P(K)
C DU TO P(K)
      P(K)=U(K)-U(K-1)
      IF (P(K)), 350, 350
C IF DU IS +, 0 NO ARTIFICIAL VISCOSITY (Q)
      PQ(K)=P(K)+Q(K)
      WK4=MIN1F(X(K)-XL0, XH1-X(K-1))
      WK1=R(K)*(CSQ*P(K) - C1*SOUND(K))
      WK1=WK1/(1.+C3*V(K)/WK4)
      Q(K)=WK1*P(K)
      PQ(K)=PQ(K)+Q(K)
350  SOUND(K)=0.
C
C LOOP 5

```

AN EXAMPLE OF CODE OPTIMIZATION

12

```

      DO 370 K=KMNP,KMXZ
      WK2=ABS(PQ(K+1)-PQ(K))
      IF(WK2-1.E-10)370,370,
      FE(K)=C2M*(E(K+1)-E(K))*SQRT(WK2*.5*(R(K+1)+R(K)))
370  CONTINUE
C
C PRESET DTCHK
      DTCHK=1.E+6
C
C LOOP 6
C
C
      DO 450 K=KMNP,KMX
      EDOT=-DT*(FE(K)-FE(K-1))*RZ(K)
      W=DT*P(K)*RZ(K)
      WE=E(K)-PQ(K)*W+EDOT
      WP=MAX1F(GAMZ*WE*R(K),0.)
      E(K)=E(K)-0.5*(PQ(K)+Q(K)+WP)*W+EDOT
C EDOT=EDOT*DT
      E(K)=MAX1F(E(K),1.E-30)
C COMPUTE NEW ENERGY (E), PRESSURE (P), AND SOUND SPEED (SOUND
      P(K)=GAMZ*E(K)*R(K)
      PQ(K)=P(K)+Q(K)
C FOR NEXT CYCLE
      IF(P(K)) 450,450,
      SOUND(K)=SQRT(GAM*P(K)/R(K))
      DTCHK=MIN1F(V(K)/SOUND(K),DTCHK)
C DELTA T CONTROL
450  CONTINUE
      TIME=TIME+DT
      NYCL=NYCL+1
C ADVANCE TIME AND CYCLE COUNTER
      IF(DT-DTCHK)475,,
C
C ***** PROBLEM IS UNSTABLE ,TIME STEP TOO LARGE*****
C
460  WRITE(NTD,462) NYCL,TIME,DT,DTCHK
462  FORMAT("UNSTABLE ",I6,3E10.2, " LF TO DT/2")
      GO TO 950
475  CONTINUE
490  CONTINUE
      IF(TIME-TIMEUP),,600
CHECK FOR MAXIMUM RUN TIME
      IF(TIME-EDTIME)111,,
CHECK FOR EDIT TIME
600  EDTIME=EDTIME+EDDT
      CALL TIMR
C      TIMR PRINTS OUT ELAPSED TIME
      WP=0.
      DO 603 K=KMNP,KMX
603  WP=WP+P(K)
C AS A DIAGNOSTIC PRINT OUT SUM OF PRESSURES FOR ALL ZONES
      WRITE(NTTY,604)NYCL,WP
604  FORMAT(I6,E16.8)
      IF(TIME-TIMEUP)111,111,601
601  CONTINUE

```

AN EXAMPLE OF CODE OPTIMIZATION

13

APPENDIX 4
ASSEMBLY LANGUAGE VERSION

```

      ID HID
      ENTRY HID
      ENTRY LOOPA
      ORG 0
HID   BSS 0
LOOPA CON 0          ' COMPUTE ACCELERATION
                        ' AND ADVANCE VELOCITY
      SA3 KMN
      SA4 KMX
      SA5 DT
      SA7 LOOPA
      SB7 X7
      SB5 X3          ' B5=KMN
      SA1 X3+PQ-1
      SA2 X3+PQ
      SB6 X4          ' B6=KMX
      BX0 X5          ' X0=DELTA T
      SA3 B5+RZAKI-1
      SA4 B5+U-1
      SB1 1
      FX6 X2-X1      ' PQ(K+1)-PQ(K)
      SB3 B5          ' B3=KMN
      SA5 UCUT
      SB4 B5          ' KMN
LPA   BX1 X2
      NX6 X6
      SA2 A2+B1      ' NEXT PQ
      FX6 X6*X3      ' ACCELERATION
      SA3 A3+B1      ' NEXT RZAKI
      SB5 B5+1        ' K=K+1
      FX7 X6*X0      ' DT*A
      SA6 B5+ACCEL-2
      FX7 X7+X4      ' U(N+1)
      SA4 A4+B1      ' NEXT U
      NX6 X7
      NX7 X7
      AX6 59
      BX6 X6-X7      ' ABSF(U)
      IX6 X6-X5      ' -U FLOOR
      PL X6,*+1
      MX7 0          ' SET U=0 IF BELOW FLOOR
      FX6 X2-X1      ' NEXT DEPO
      SA7 A4-B1
      GE B6,B5,LPA   ' STACK LOOP
      SA1 B4+U-1
      SA2 B4+XX-1
      SA3 B4+XX-2
      SA4 B4+R-1
      FX6 X1*X0
      SA5 B4+V-1
      ' X(K)
      ' X(K-1)
      ' DENSITY
      ' U*DT
      ' VOLUME

```

AN EXAMPLE OF CODE OPTIMIZATION

14

```

LPB  FX6 X6+X2      ' X(N+1)=X(N)+U(N+1)
     FX2 X4*X5      ' R*V
     SA4 A4+B1      ' NEXT R
     NX7 X6
     SA1 A1+B1      ' NEXT U
     SA5 A5+B1      ' NEXT V
     FX6 X7-X3      ' X(K)-X(K-1) (V)
     BX3 X7         ' IS NEXT X(K-1)
     SA7 A2
     NX6 X6
     FX7 X2/X6      ' A DIVIDE NEW RHO R=RV/V
     SA2 A2+B1      ' NEXT X
     SA6 A5-B1      ' V
     MX6 0
     SB4 B4+1       ' K=K+1
     SA6 B4+Q-2     ' Q=0
     SA6 B4+FE-2    ' FE=0
     FX6 X1*X0
     SA7 A4-B1
     GE B6,B4,LPB   ' STACK LOOP
     SB2 B1         ' B2=1
     SB5 B1+B3      ' B5=KMNP
     SA4 B5+P-1
     SA5 B5+U-2
     SA2 B5+U-1
     SB4 B3         ' B4=KMN
LPC  BX6 X4
     FX0 X2-X5      ' U(K)-U(K-1) DU
     BX5 X2
     SA2 A2+B2      ' NEXT U
     SA6 B5+PQ-1    ' PQ(K)=P(K)
     NX6 X0
     SA4 A4+B2      ' NEXT P
     PL X6,LPC1     ' ASSUME JUMP IS NORMAL PATH
     ZR X6,LPC1
     SA1 B4+XX-1    ' X(KMN)
     SA3 B5+XX-1    ' X(K)
     SA2 B6+XX-1    ' X(KMX)
     SA4 B5+XX-2    ' X(K-1)
     FX0 X3-X1      ' X(K)-X(KMN)
     FX7 X2-X4      ' X(KMX)-X(K-1)
     SA3 B5+V-1
     SA1 C3
     NX0 X0
     NX7 X7
     LX2 X0-X7
     PL X2,*+1
     BX7 X0
     FX0 X1*X3      ' MIN
     SA2 CSQ        ' C3*V
     FX0 X0/X7      ' C3*V/WK4
     SA4 C1
     SA3 B5+SOUND-1 ' SOUND(K)
     SA1 B5+R-1     ' R(K)
     FX7 X2*X6      ' CSQ*P (DU)
     NO
     SX2 17204B

```

AN EXAMPLE OF CODE OPTIMIZATION

15

```

FX4 X4*X3          ' C1*SOUND
LX2 45
SA3 B5+PQ-1
FX7 X7-X4          ' CSQ=DU-C1*SOUND
FX0 X0+X2
NX7 X7
NX0 X0
FX7 X7*X1          ' R*
FX0 X7/X0          ' WK1
SA2 B5+U
SA4 B5+P
FX7 X0*X6          ' Q=WK1*P (DU)
FX1 X7+X3          ' PQ+Q
SA7 B5+Q-1
NX7 X1
NO
LPC1 SA7 B5+PQ-1
MX7 0
SA6 A4-B2
SA7 B5+SOUND-1
SB5 B5+B2          ' K=K+1
NO
GE B6,B5,LPC
SB4 B3+B2          ' B4=KMNP
SB3 B2
SA1 B4+PQ-1
SA2 B4+PQ          ' PQ(K+1)
SA5 PQCUT
SB5 B6          ' B5=KMX
SB6 B6-B2
SB2 B4          ' B2=KMNP
LPFE1 FX6 X2-X1          ' PQ(K+1)-PQ(K)
BX1 X2
SA2 A2+B3
NX7 X6
NX6 X6
AX7 59
SB4 B4+B3
BX7 X6-X7          ' ABSF(DELPQ)
IX0 X7-X5
NO
PL X0,LPFE2
GE B6,B4,LPFE1          ' STACK LOOP
JP LPEP
LPFE2 SA1 B4+R-1          ' R(K+1)
SA2 B4+R-2          ' R(K)
SA3 B4+E-1          ' E(K+1)
SA4 B4+E-2          ' E(K)
SA5 C2M          ' -C2
SX0 17174B          ' .5
FX6 X1+X2          ' R(K+1)+R(K)
LX0 45
FX3 X3-X4          ' E(K+1)-E(K)
FX7 X7*X0          ' WK2*.5
NX6 X6
NX3 X3
FX1 X6*X7          ' INPUT TO SQRT

```

AN EXAMPLE OF CODE OPTIMIZATION

16

```

FX5 X3*X5          X5=C2M*(E(K+1)-E(K)) SAVED
SX4 17174B        BEGIN SQRT X1
BX3 X1
LX4 45
AX1 1
FX6 X4/X3
UX2 B1,X1
AX2 41
SA4 DIV2
SA2 X2+SQTB-32
PX3 B1,X3
FX1 X2*X3
LX2 29
IX3 X1+X2
FX0 X3*X3
IX7 X3-X4
FX1 X3*X6
FX3 X3+X7
FX2 X0*X1
FX0 X3-X2
NX3 X0
FX0 X3*X3
IX7 X3-X4
FX1 X3*X6
FX3 X3+X7
FX2 X0*X1
FX0 X3-X2
NX6 X0
SA1 B4+PQ-1      END SQRT X6
SA2 B4+PQ        NEXT PQ(K)
FX6 X6*X5        NEXT PQ(K+1)
SA5 PQCUT        FE
SA6 B4+FE-2
GE B6,B4,LPFE1
LPEP SA1 B2+FE-1  COMPUTE ENERGY,PRESSURE AND
SA2 B2+FE-2      SOUND SPEED
SA3 DT
SA4 B2+RZ-1
LPEP1 FX0 X2-X1   FE(K-1)-FE(K) CORRECT SIGN(-DT)
SA5 B2+P-1       DT*RZ
FX6 X4*X3        DT*RZ
SA1 B2+E-1       P+Q
SA2 B2+PQ-1      P+Q
NX0 X0           RH0
SA3 B2+R-1       W=DT*P*RZ
FX7 X6*X5
SA4 GAMZ
FX0 X0*X6        EDOT
FX6 X2*X7        PQ*W
SA5 B2+Q-1       Q(K)
FX0 X1+X0        E(K)+EDOT
FX4 X4*X3        GAMZ*R(K)
SA3 GAM
SX1 17174B      .5
NX0 X0
FX2 X2+X5       PQ(K)+Q(K)
LX1 45

```

AN EXAMPLE OF CODE OPTIMIZATION

17

```

FX6 X0-X6          ' E(K)+ECUT-PQ*W WE
FX7 X7*X1          ' .5*W
NX6 X6
NX2 X2
FX6 X6*X4          ' WP=WE*GAMZ*R
NO
+ PL X6,*+1        ' WP=MAX(WP,0)
MX6 0
+ FX6 X6+X2        ' PQ+Q+WP
SA2 B2+R-1
NX6 X6
SA1 ECUT
FX7 X6*X7          ' .5*W(PQ+Q+P)
FX7 X0-X7          ' E(N+1)
FX0 X3/X2          ' GAM/R(K)
NX7 X7
IX2 X7-X1
+ PL X2,*+1        ' E=MAX(E,ECUT)
BX7 X1
+ FX6 X7*X4        ' GAMZ*R(K)*E(K)
SA7 B2+E-1
FX3 X5+X6          ' PQ+P
SA5 B2+V-1
SA6 B2+P-1
NX7 X3
FX1 X0*X6          ' P*GAM/R
SA7 B2+PQ-1
ZR X6,LPEP4        ' P(N+1)=0.
SX4,17174B        ' BEGIN SQRT X1
BX3 X1
LX4 45
AX1 1
FX6 X4/X3
UX2 B1,X1
AX2 41
SA4 DIV2
SA2 X2+SQTB-32
PX3 B1,X3
FX1 X2*X3
LX2 29
IX3 X1+X2
FX0 X3*X3
IX7 X3-X4
FX1 X3*X6
FX3 X3+X7
FX2 X0*X1
FX0 X3-X2
NX3 X0
FX0 X3*X3
IX7 X3-X4
FX1 X3*X6
FX3 X3+X7
FX2 X0*X1
FX0 X3-X2
NX6 X0
+ SA4 DTCHK
FX7 X5/X6          ' SOUND(K)
END SQRT X6

```


AN EXAMPLE OF CODE OPTIMIZATION

18

```

      SB2 B2+B3          ' K=K+1
      SA1 B2+FE-1
      SA2 B2+FE-2
      SA3 DT
      IX0 X4-X7
      SA4 B2+RZ-1
      SA6 B2+SOUND-2
+      NG X0,*+1
      SA7 DTCHK          ' DTCHK=MIN(DTCHK,SOUND(K))
      GE B5,B2,LPEP1
      JP B7+0            ' RETURN
LPEP4 SB2 B2+1
      GE B5,B2,LPEP
      JP B7+0            ' RETURN

```

AN EXAMPLE OF CODE OPTIMIZATION

19

APPENDIX 5
VECTORIZED VERSION

```

C   DECLARATIVES.
    VECTOR (IDACCELV,ACCELV((0,1))), (IDRZAKIV,RZAKIV((0,1)))
    VECTOR (IDPQV,PQV((0,1))), (IDUV,UV((0,1)))
    VECTOR (IDQV,QV((0,1))), (IDFEV,FEV((0,1)))
    VECTOR (IDXV,XV((0,1))), (IDVV,VV((0,1)))
    VECTOR (IDRV,RV((0,1))), (IDT1V,T1V((0,1)))
    VECTOR (IDB1V,B1V((0,1)))
    VECTOR (IDB2V,B2V((0,1))), (IDB3V,B3V((0,1)))
    VECTOR (IDSOUNDEV,SOUNDEV((0,1))), (IDEV,EV((0,1)))
    VECTOR (IDT2V,T2V((0,1))), (IDT3V,T3V((0,1)))
    VECTOR (IDPV,PV((0,1))), (IDRZV,RZV((0,1)))
    BIT B1V,B2V,B3V
C   GENERATE DESCRIPTORS
    K=(KMX-KMN+1).SHL.1AD
    IDACCELV=(.LOC.ACCEL(KMN))*NWORD.UN.K
    IDRZAKIV=(.LOC.RZAKI(KMN))*NWORD.UN.K
    IDPQV=(.LOC.PQ(KMN))*NWORD.UN.K
    IDUV=(.LOC.U(KMN))*NWORD.UN.K
    IDQV=(.LOC.Q(KMN))*NWORD.UN.K
    IDFEV=(.LOC.FE(KMN))*NWORD.UN.K
    IDXV=(.LOC.X(KMN))*NWORD.UN.K
    IDVV=(.LOC.V(KMN))*NWORD.UN.K
    IDRV=(.LOC.R(KMN))*NWORD.UN.K
    IDT1V=(.LOC.ENT(KMN))*NWORD.UN.K
    IDB1V=(.LOC.MATSUR)*NWORD.UN.K
    IDB2V=(.LOC.AMX)*NWORD.UN.K
    IDB3V=(.LOC.IACT)*NWORD.UN.K
    IDSOUNDEV=(.LOC.SOUND(KMN))*NWORD.UN.K
    IDPV=(.LOC.P(KMN))*NWORD.UN.K
    IDEV=(.LOC.E(KMN))*NWORD.UN.K
    IDT2V=(.LOC.T2)*NWORD.UN.K
    IDT3V=(.LOC.T3)*NWORD.UN.K
    IDRZV=(.LOC.RZ(KMN))*NWORD.UN.K
    IDPQVP=(.LOC.PQ(KMN))*NWORD.UN.((KMX-KMN+2).SHL.1AD)
    CALL TIMR          $$C INITIALIZE TIMING
C   EDIT INITIAL CONDITION
111  CONTINUE
    ACCELV=PQVP(1;)-PQV
    ACCELV=ACCELV*RZAKIV
    T1V=ACCELV*DT
    UV=UV+T1V
    T1V=ABSF(UV)
    B1V=T1V.LT.UCUT      $$C BIT=1 IF U LESS THAN UCUT
    UV=B1V.CTRL.0.
    QV=0.
    FEV=0.
    T1V=UV*DT
    XV=XV+T1V          $$C ADVANCE POSITION (X)
    RV=RV+VV

```

AN EXAMPLE OF CODE OPTIMIZATION

20

```

VV=XV-XV(-1;)
RV=RV/VV      $$C RHO(N+)
XL0=X(KMN)
XH1=X(KMX)
PQV(1;)=PV(1;)
PV(1;)=UV(1;)-UV      $$C DU
B1V=PV(1;).LT.0.
T1V=XV(1;)-XL0
T2V=XH1-XV
B2V=T2V.LT.T1V
T1V=B2V.CTRL.T2V
T2V=C3*VV(1;)
T1V=B1V.CTRL.T2V/T1V
T3V=CSQ*PV(1;)
T2V=C1*SQUNDV(1;)
T2V=T3V-T2V      $$C CSQ*P-C1*SQUND
T2V=T2V*RV(1;)      $$C WK1
T1V=T1V+1.
T2V=T2V/T1V
QV(1;)=B1V.CTRL.T2V*PV(1;)
PQV(1;)=PQV(1;)+QV(1;)
0350 CONTINUE
COMPUTE FE(K)
T1V=PQV(1;)-PQV
T1V=ABSF(T1V)
B1V=1.E-10.LT.T1V      $$C IF B1V=1 COMPUTE FE
T2V=RV(1;)+RV      $$C R(K+1)+R(K)
T2V=T2V*.5
T2V=T2V*T1V
T1V=SQRT(T2V)
T2V=EV(1;)-EV      $$C E(K+1)-E(K)
T2V=T2V*C2M      $$C -C2*(E(K+1)-E(K))
FEV(1;)=B1V.CTRL.T2V*T1V
T1V=R2V(1;)*DT
T2V=FEV-FEV(1;)
T3V=T1V*T2V      $$C EDOT
T1V=T1V*PV(1;)      $$C W
T2V=PQV(1;)*T1V
T2V=EV(1;)-T2V
T2V=T2V+T3V      $$C WE
T2V=T2V*GAMZ
T2V=T2V*RV(1;)
B1V=T2V.LT.0.
T2V=B1V.CTRL.0.      $$C WP=MAX(GAMZ*WE*R(K),0.)
T2V=T2V+QV(1;)
T2V=T2V+PQV(1;)
T2V=.5*T2V
T2V=T2V*T1V
T2V=T3V-T2V      $$C EDOT-
EV(1;)=EV(1;)+T2V      $$C E(N+)
B1V=EV(1;).LT.1.E-30
EV(1;)=B1V.CTRL.1.E-30      $$C APPLY FLOOR TO ENERGY
PV(1;)=GAMZ*EV(1;)
PV(1;)=PV(1;)*RV(1;)
PQV(1;)=PV(1;)+QV(1;)      $$C P+Q. FOR NEXT CYCLE
T1V=GAM*PV(1;)
SQUNDV(1;)=T1V/RV(1;)

```

AN EXAMPLE OF CODE OPTIMIZATION

21

```
SOUNDV(1;)=SQRTF(SOUNDV(1;))
T2V=VV(1;)/SOUNDV(1;)
CALL Q81NLINE(#D9,0,0,1DT2V,0,K,0,DTCHK)
C MINIMUM OF VECTOR T2V STORED IN DTCHK
TIME=TIME+DT
NYCL=NYCL+1
IF(DT-DTCHK)475,,
460 WRITE(NTED,462) NYCL,TIME,DT,DTCHK
462 FORMAT("UNSTABLE ",16,3E10.2, " LF TO DT/2")
GO TO 950
475 CONTINUE
490 CONTINUE
IF(TIME-TIMEUP)1600
IF(TIME-EDTIME)111,,
```

Distribution:

Internal

R. ABBOTT	L-307
R. Barker	L-92
R. Bell	L-72
A. Cecil	L-81
R. Cralle	L-81
B. K. Crowley	L-51
W. P. Crowley (5)	L-81
S. Fernbach	L-61
G. Foglesong	L-81
W. Grasberger	L-81
P. Gray	L-307
W. Grayson	L-24
C. Hendrickson	L-81
T. Kishi	L-61
W. Lokke	L-33
H. Nelson	L-328
J. Oldham	L-81
D. Pehrson	L-60
M. Richards	L-67
T. Rudy (5)	L-328
W. Schulz	L-81
G. Sutherland	L-60
K. Trigger	L-81
V. Waluch	L-81
T. Wainwright	L-71
W. Zagotta	L-81
G. Zimmerman	L-32
TID File (5)	L-9

External

J. P. Boris
K. Hain
G. Hain
Naval Research Lab.
Washington, D.C. 20390
C. E. Leith
W. Washington
National Center for Atmospheric Research
Boulder, Colorado 80302
D. Neeper
G. Spillman
Los Alamos Scientific Laboratory
Los Alamos, New Mexico 85744
C. P. Knowles
R. J. McLean
RDA Associates
Santa Monica, California 90403
N. Frazier
C. Purcell
Control Data STAR Systems
Minneapolis, Minnesota 55424

NOTICE

"This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately-owned rights."