(NASA-CR-163801) PRELIMINARY STUDY ON THE POTENTIAL USEFULNESS OF ARRAY PROCESSOR TECHNIQUES FOR STRUCTURAL SYNTHESIS Final Report (Rensselaer Polytechnic Inst., Troy, N. Y.) 153 p HC A12/MF A01 CSCL 20K G3/39

N81-12447

Unclas
29377

Final Report

on

"Preliminary Study on the Potential Usefulness of Array Processor Techniques for Structural Synthesis"

for

Research Grant No. NSG 1636

National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665

by

Principal Investigator
Larry J. Feeser
Professor and Chairman
Department of Civil Engineering
Rensselaer Polytechnic Institute
Troy, New York 12181

NASA Technical Officer
James L. Rogers, Jr.
Structures and Dynamics Division
NASA Langley

December 5, 1980

CONTENTS

	Page
INTRODUCTION	1
PREVIOUS WORK	3
SCOPE OF PRESENT WORK	4
SPAR CODE USED	5
EIG	6
INV	8
SSOL	8
K & M	9
TEST PROGRAM	9
TEST RESULTS	10
CONCLUSIONS	10
FIGURES	15
REFERENCES	22
APPENDICES	
Appendix A	23
Appendix B	31
Appendix C	40
Appendix D	48
Appendix E	57
Appendix F	65
Appendix G	89
Appendix H	100
Appendix I	111
Appendix J	140

INTRODUCTION

Recent developments in the structural synthesis area point to the developing use of advanced optimization techniques, combined with sophisticated structural analysis, to provide a system which can be used for production application. One such system has been described recently by Sobieski and Bhat (1). Their system makes use of the program, CONMIN, as the optimizer, which is based on mathematical nonlinear programming techniques of feasible-usable directions in combination with a structural analyzer called SPAR, (2,3) which is a program of high modularity and computer efficiency. These two programs have been combined through Optimizer-to-Analyzer and Analyzer-to-Optimizer Processors and the use of a standard computer operating system, in this case CDC NOS. This implementation provides a system which runs in a large computing environment (CDC Cyber). While this implementation of the system is quite workable, it has all of the problems of making use of a large computer utility, in that turnaround time becomes of concern, and the ability to interact with the system during execution is generally lost.

Recently, some attractive advances have been made in the computer area, as it relates to general structural analysis capability. The introduction of large scale mini-computers available at very reasonable prices, now makes it possible to perform extensive structural analysis on these machines. A number of computer manufacturers are now making what can be called maxi-mini-computers, with large central memories and virtual operating systems, whose speeds are in the medium range of all computing equipment. This type of computing hardware sells for, on the order of, one tenth the cost of some of the large systems, such as IBM and CDC. Consequently, even though the processing speeds may not be as fast as the large scale machines, the price performance certainly is far better than anything we have seen previously. This availability has led to decentralization of computing and allows a structural analysis-design group to have its own computing capability under their direct control. Even though the processing speed may not be as fast as the larger systems, the turnaround time and the interactiveness of this type of system is improved tremendously. It is just this kind of system which NASA Langley has installed in their structural analysis-design group and has implemented the program SPAR, a large general purpose linear structural analysis package, on this system.

The question then becomes whether or not it is viable to move the concept, as demonstrated by Sobieski and Bhat to the PRIME or similar computer as a total system. The problem generated is that the overall optimization problem generally involves a considerable number of reanalysis steps as one approaches an optimal solution in the design sense. Consequently, of concern is the operating speed of the analyzer program SPAR on the PRIME P400 hardware at Langley. It certainly appears that the analysis speeds must be increased by an order of 5 to 10, in order for the total system concept to be implemented on the PRIME and to provide reasonable turnaround and interactiveness.

One potential for providing this increased speed on the PRIME or similar hardware is that of parallel or array processing implemented in hardware which has become available recently at very reasonable prices. There has been considerable interest in array processing as illustrated by the early machines, such as ILLIAC IV and the commercially available CDC STAR. However, these two pieces of hardware are very expensive in their design and implementation. On the other hand, there are now manufacturers, such as Floating Point Systems, who make array processors with execution speeds on the order of 7 to 10 million instructions per second, and which are available for a price of around 50 to 60 thousand dollars and which can be interfaced directly to some of the mini-computers presently available.

It is the potential of the connection of a Floating Points Systems AP120 processor to a mini-computer which needs exploration. If, indeed, the connection of an array processor can provide significant increase in execution speed for the SPAR analysis program, then the potential for implementing the Sobieski-Bhat concept on a maxi-mini-computer would be greatly enhanced.

This research project proposes to perform research in the area of simulating the effects of the use of array processor techniques within the program SPAR, in order to evaluate the potential speedups which may result. This research is possible because of the availability of an AP120 simulator package which now executes on a PRIME computer and the availability of a PDP-11/40 with an AP120B attached at RPI. Consequently the potential exists for simulating portions of the SPAR analysis program both with the PRIME simulator and the PDP-11/40-AP120B hardware.

PREVIOUS WORK

During the 1978-79 academic year, RPI has worked with the Floating Point Systems array processor simulator on our PRIME computer to obtain some general information concerning potential speedups of small pieces of computer code as lifted from some SPAR Processors (4). This preliminary work shows results only of the floating point processor execution times, but does point towards considerable potential for significant speedup of some of the tight loop areas of the SPAR code. Several sections of code from SPAR have been recoded in Floating Point System's assembly language and run on the AP120 simulator on the PRIME. The code is that taken from a section of the matrix inversion processor of SPAR. A comparison of computation times between the PRIME P500 and the AP120 for completing identical routines for a given joint was made, varying several important parameters. The results are shown in Table 1.

Trial	CONRNG	NOF	NZERO	CPU(PRIME)	CPU(AP120)
1	2	3	3	3.00 ms	0.07 ms
2	6	3	3	24.24 ms	0.80 ms
3	10	3	3	84.80 ms	2.26 ms
4	15	3	3	187.88 ms	5.37 ms
5	6	6	3	90.90 ms	2.39 ms

NOF - Number of degrees of freedom per joint

CONRNG - Number of non-zero submatrices in a particular row of the stiffness matrix

NZERO - Number of non-zero degrees of freedom for the given calculation (Constraint conditions).

TABLE 1

These results, even though they do not include any I/O time required to pass the data arrays between the PRIME and the AP120, pointed to potential for significant speedup with the use of the array processor.

SCOPE OF PRESENT WORK

The present work has concerned itself with a systematic approach to simulating and measuring the AP120 performance in relationship to a number of SPAR processors. The past simulation data is reevaluated and some additional data generated to better understand the potential and feasibility of using an array processor such as the AP120 to speed-up the analysis process for large structural systems.

The past simulator study measured only the execution time of the array processor instruction set for some selected portions of translated SPAR code. There was not direct measuring of the input/output and data transfer times between the PRIME and the simulated AP120 connection. In this study, a limited amount of actual measurements of input/output and data transfer times were made using a PDP-11/40 with an AP120B attachment. Using these data, estimates are made as to the relative speedups that can be executed in a more complete implementation on an array processor - maxi-mini computer system.

SPAR CODE USED

Since the previous study showed that a significant amount of man power was required to rewrite existing FORTRAN routines from the SPAR Processors in the assembler language of the AP120, it was decided to test those sections of SPAR code which had already been translated for the simulation study. This involved the actual implementation on the AP120B of this code. Significant differences between the actual AP120B and the simulator codes had to be resolved by debugging.

Seven SPAR subroutines or portions of SPAR subroutines have been implemented in the AP120 assembler language and timing information gathered for FORTRAN execution and AP120 assembler execution. The sections of code were taken from five different SPAR Processors; EIG, INV, SSOL, K and M. Each piece of code will be described individually.

EIG

The EIG Processor is used to solve vibration and bifuncation buckling eigenproblems. Three short subroutines were selected from EIG and portions of each were coded in AP120 assembler and implemented on the PDP-11/40-AP120B system. The routines converted do not represent a significant contribution to the running time of EIG, but were selected for their relative ease of program conversion and the fact that they did use tightly coupled loops of vector floating point multiplications and additions. The experience gained in using these routines helped greatly in the conversion of more complex operations.

The first routine tested was subroutine EIGLD, which is used by SPAR to set problem dependent parameters. The particular code converted was seven lines of FORTRAN which generates a vector with random valued components. Figure 1 shows these seven lines of FORTRAN code. In order to perform timing testing, three stand alone programs were developed. The first represents the FORTRAN code to be converted. Dimension, timing and input/output have been built around the seven statements. This program is called FOR1.FOR and allows us to obtain timing information for the FORTRAN execution. The second program developed replaces the FORTRAN statements with FORTRAN calls to Array Processor utilities and to the AP assembler replacement code. This program also contains timing generating elements and is used to obtain timing information for the AP execution. This program is designated APFOR1.FOR. The third program is the AP assembler program which performs the operations on the AP120B. It is developed as a FORTRAN callable routine and is called from APFOR1. The assembler routine is named APEGLD. These three programs are presented in Appendix A.

The second routine tested was subroutine EXPND1, which is used by SPAR to perform the addition of one scaled vector to another and substitute the result into the original vector. The particular code converted was 13 lines of FORTRAN which is shown in Figure 2. The timing tests were performed using three stand alone programs. Program FOR2.FOR represents the 13 lines of code with dimensioning, timing calls and input/output added. This program provides timing information for the FORTRAN execution. The second program, APFOR2.FOR, replaces the FORTRAN statements with FORTRAN calls to AP utilities and the AP assembler code. This program also contains timing generating elements and is used to time the AP execution. The third program is the AP assembler routine which performs the operations on the AP120B and is named APXPD1.APM. These three programs are presented in Appendix B.

The third routine from EIG which was tested is NEWX, which performs a number of operations on vectors, including zeroing, normalization, and multiplication by a constant. Three separate portions of the subroutine NEWX have been tested. The first one is five lines of FORTRAN used to zero a vector. This code is shown as Figure 3. The three programs for this portion are FOR3.FOR, APFOR3.FOR and APNWX1.APM. These three programs are presented in Appendix C.

The second portion of NEWX is 11 lines of FORTRAN which performs a normalization operation on two vectors. This code is shown as Figure 4. The three programs for this portion are FOR4.FOR, APFOR4.FOR and APNWX2.APM. These three programs are presented in Appendix D.

The third portion of NEWX is 6 lines of FORTRAN which performs a multiplication of a vector by a constant. This code is shown as Figure 5. The three programs for this portion are FOR5.FOR, APFOR5.FOR and APNWX3.APM. These three programs are presented in Appendix E.

INV

INV is the stiffness matrix inversion or reduction processor for SPAR. Two routines which represent well over 50 percent of the total CPU time used in INV have been implemented. The first is the major portion of the Subroutine RED which performs the core of the reduction process. Eighty-two lines of FORTRAN code have been coded in AP assembler, representing the major test implementation for this project. Figure 6 lists this code. Again three programs have been developed for testing purposes. They are FOR6.FOR, APFOR6.FOR and APRED.APM. These three programs are presented in Appendix F.

The second routine from INV which has been implemented is AFEX. Seventeen lines of FORTRAN have been converted to AP assembler and this FORTRAN is shown as Figure 7. The three programs developed for testing this code are FOR8.FOR, APFOR8.FOR and APAFEX.APM. These three programs are presented in Appendix G.

SSOL

SSOL is the static solution processor for SPAR. It performs the matrix multiplication operations required to obtain the displacement vectors for each loading condition using the reduced matrix from INV. The basic routine used is MULTEX, a routine found in the

SPAR library. The eleven lines of FORTRAN shown in Figure 8 have been implemented by the usual three programs designated as FOR9.FOR, APFOR9.FOR and APMLTX.APM. These three programs are presented in Appendix H.

K & M

The K and M processors assemble the system stiffness and mass matrices, respectively. Both the K and M processors call the routine TRAN6 to perform the transformation

$$TK^T * GKL * TL$$

for each node. Fifty-two lines of FORTRAN code representing the major portion of TRAN6 have been implemented. This code is shown in Figure 9. The three programs used to test timing have been designated FORC.FOR, APFORO.FOR and APTRN6.APM. Listings of these three programs are presented in Appendix I.

TEST PROGRAM

Six of the nine program sets described above were tested extensively on the PDP-11/40 - AP120B system. The most important implementations are FOR6, which relates to the stiffness matrix reduction, FOR9, which relates to the matrix multiplication process and FORO, which performs rotational and translational transformations of matrices.

In order to obtain some comparison of the application to real problems, two problems were solved using SPAR. These are FUSEL and LUT, listings of which are provided in Appendix J. The CPU timing results of these runs are shown in Tables 2 and 3.

It is important to note that the INV processor uses the most CPU resources, representing approximately 40% of the total CPU time in LUT and approximately 75% of the total in FUSEL.

TEST RESULTS

The results of running comparisons between FORTRAN execution and AP execution of the same code are summarized in Table 4. These ratios include both the execution times and the data transfer times in all cases. It is this fact that accounts for the wide range of speedups obtained.

In the case of the INV reduction process implementation, the result is quite favorable. Since the execution of routine RED represents more than one half of the CPU time for the processor INV, the result indicates that one could expect on the order of doubling of the speed in the processor using only this one small section of code implementation.

CONCLUSIONS

The results of this limited testing program add to the evidence presented in Reference 4 as to the appropriateness of doing a full blown implementation. The evidence is not conclusive that sufficient speed can be obtained for an analysis-design speedup of sufficient magnitude to warrant full scale testing. However, the results generated here are for a very small part of the SPAR system being implemented on the AP. One must keep in mind that this limited implementation has taken a considerable amount of manpower. The

FUSEL run with SPAR on PRIME 650

Processor	CPU Time (Seconds)
TAB	13.6
ELD	6.9
E	15.5
EKS	79.4
TOFO	28.0
K	67.1
INV	768.2
AUS	2.7
DCU	0.5
SSOL	32.3
GSF	7.3
PSF	7.9

Table 2

LUT run with SPAR on PRIME 750

Processor	CPU Time (Seconds)
TAB	28.9
ELD	43.3
TOPO	12.3
E	26.9
EKS	32.7
K	76.9
INV	318.6
M	164.6
AUS	25.9
SSOL	33.1
GSF	30.7

Table 3

Run Time Comparisons

Program Set	FORTRAN TIME / AP TIME
FOR2 - XPD1	26
FOR4 - NWX2	19
FOR6 - RED	51
FOR8 - AFEX	6
FOR9 - MLTX	40
FOR0 - TRN6	22

Table 4

problem is that the level of sophistication of the computer programmer must be high enough to be able to work with the very complex assembler language of the AP. Even with an expert programmer, the programming of the AP must be described as tedious and tricky, due to the parallelism inherent in the system.

The final conclusion is that the purchase of an array processor for attachment to a maxi-mimi computer cannot yet be justified solely on the evidence to date in the structural analysis area. Other applications would have to be included to justify the purchase.

FIGURES

```
      IRB=0
     DO 1070 J=1,JT
     DO 1065 I=1,JDF
     K=INEX(I)
     IRBPI=IRB+I
1065 A(IRBPI)=RAN(0,0)*RSCALE(K)
1070   IRB=IRB+JDF
```

Figure 1

```
     DO 260 J=1,NV2
     JJZ=J+JZ
     DO 250 K=1,NV1
     KKZ=K+KZ
     QKJ=Q(KKZ,JJZ)
     IF(KTEST(JJZ).NE.0) GO TO 150
     IF(JJZ.NE.KKZ) GO TO 250
     QKJ=1.0
150  CONTINUE
     DO 200 I=1,LVEC
200  V2(I, J)=V2(I, J)+V1(I, K)*QKJ
250  CONTINUE
260  CONTINUE
```

Figure 2

```
      DO 70 N=1,NLOADS
      IF(LSDO(N) .EQ. 0)GO TO 70
      DO 60 I=1,LVEC
   60 V2(I,N)=0.
   70 CONTINUE
```

Figure 3

```
      DO 1300 N=1,NLOADS
      IF(LSDO(N).EQ.0) GO TO 1300
      SUM=.0
      DO 1100 I=1,LVEC
 1100 SUM=SUM+V2(I,N)*V1(I,N)
      SUM=1./SQRT(ABS(SUM))
      Z(N)=SUM
      DO 1200 I=1,LVEC
      V1(I,N)=V1(I,N)*SUM
 1200 V2(I,N)=V2(I,N)*SUM
 1300 CONTINUE
```

Figure 4

```
      DO 1850 N=1,NLOADS
      IF (LSDO(N) .EQ. 0) GO TO 1850
      SUM=Z(N)
      DO 1800 I=1,LVEC
 1800 V1(I,N)=V1(I,N)*SUM
 1850 CONTINUE
```

Figure 5

```
      DO 1000 K=1,NZERO
      M= IABS(MAP(K))
      DO 1000  J=1,CONRNG
      L= SUBMAP(J)
      DO 1000 I=1,NDF
 1000 B(I,J,K)= S(M,I,L)
C
C**** PRELIMINARY B MODIFICATION.
      DO 1400 K=1,NZERO
      M= MAP(K)
      IF(M.LT.0) GO TO 1400
      RA=BB(M,K)
      IF(RA.GT. ZEROD) GO TO 1025
C     NEX=INEX(M)
      IF(RA.LT.-ZEROD) GO TO 1015
C     KSING=KSING+1
C     NSING=KSING
C     WRITE(IOUT,1010) JOINT,NEX
C1010 FORMAT(49H *** WARNING. SYSTEM K SINGULAR. JOINT/COMPONENT=I5,I2)
      RA=.0
      GO TO 1030
C1015 KNEG=KNEG+1
C     NNEG=KNEG
C     IF(IPRT.LT.2) GO TO 1025
C     WRITE(IOUT,1020) JOINT,NEX
C1020 FORMAT(38HONEGATIVE DIAG TERM.  JOINT/COMPONENT= I5,I2)
C1025 RA=1./RA
1015   CONTINUE
1025   CONTINUE
 1030 BB(M,K)=RA
      IF(K.EQ.NZERO) GO TO 1200
      LA=K+1
      DO 1100 L=LA,NZERO
      IA=IABS(MAP(L))
      RAB= RA*BB(IA,K)
      DO 1100 I=IA,NDPCON
 1100 BB(I,L)= BB(I,L) -RAB*BB(I,K)
 1200 IF(M.EQ.NDP) GO TO 1400
      INEXT=M+1
      DO 1300 I=INEXT,NDP
 1300 BB(I,K)= BB(I,K)*RA
 1400 CONTINUE
C
      IF(CONRNG.EQ.1) GO TO 2500
```

Figure 6

```
      DO 2100 K=1,NZERO
      M= MAP(K)
      IF(M.LT.0) GO TO 2100
      RA= BB(M,K)
      NSUB= CONRNG
      DO 2000 I=2,CONRNG
      NSUB= NSUB+1
      LS= SUBMAP(NSUB)
C
C**** MODIFY EII
      DO 1600 ICOL=1,NDF
      RAB= RA*B(ICOL,I,K)
      DO 1600 IROW=1,ICOL
 1600 S(IROW,ICOL,LS)= S(IROW,ICOL,LS) -RAB*B(IROW,I,K)
COS   CALL CALCS(CONRNG,B,SUBMAP,NDF,S)
CDC   12 CDS OMITTED
C
C *** THE COMPASS ROUTINE CALCS REPLACES THE FOLLOWING ON CDC
C
      DO 1700 IROW=1,NDF
 1700 B(IROW,I,K)=RA*B(IROW,I,K)
      IF(I.EQ.CONRNG)GO TO 2000
C
C**** MODIFY EIJ S
      JA=I+1
      DO 1900 J=JA,CONRNG
      NSUB=NSUB+1
      LS=SUBMAP(NSUB)
      DO 1800 ICOL=1,NDF
      DO 1800 IROW=1,NDF
 1800 S(IROW,ICOL,LS)=S(IROW,ICOL,LS)-B(IROW,I,K)*B(ICOL,J,K)
 1900 CONTINUE
 2000 CONTINUE
 2100 CONTINUE
 2500 DO 2600 L=1,CONRNG
      K= SUBMAP(L)
      DO 2600 J=1,NDF
      DO 2600 I=1,NDF
 2600 S(I,J,K)=.0
```

Figure 6 (Continued)

```
      DO 1000 ISUB=1,NSUBS
      IF(ISUB.EQ.1) GO TO 600
      J=IFIX(AK(LK))
  400 IF(J.EQ.K4(LCON)) GO TO 600
      LCON=LCON+1
      IF(LCON.LT.LCONX) GO TO 400
      WRITE(6,500)
  500 FORMAT(29H0*** MPILE/KMAP INCONSISTENCY)
      STOP
  600 LSUB=LCON+CONRNG
      K=K4(LSUB)
      DO 700 J=1,NDF
      DO 700 I=1,NDF
      S(I,J,K)=S(I,J,K)+AK(LKSUB)
  700 LKSUB=LKSUB+1
      LCON=LCON+1
 1000 LK=LK+1
```

Figure 7

```
      I=JLIST(1)
      DO 100 L=1,N
      DO 100 M=1,N
  100 VOUT(L,I)=VOUT(L,I)+A(L,M,1)*VIN(M,I)
      IF(NSUBS.LT.2) GO TO 300
      DO 200 K=2,NSUBS
      J=JLIST(K)
      DO 200 L=1,N
      DO 200 M=1,N
      VOUT(L,I)=VOUT(L,I)+A(L,M,K)*VIN(M,J)
  200 VOUT(L,J)=VOUT(L,J)+A(M,L,K)*VIN(M,I)
```

Figure 8

```
      DO 2000 L=1,NNODES
      LL=ITRANS(L)
      DO 2000 K=1,L
      KK=ITRANS(K)
      N=N+1
C
      DO 100 J=1,6
      DO 100 I=1,6
      GKLTL(I,J)= .0
  100 HKL(I,J)=   .0
C     FORM HKL=  TK(TRANSPOSE) *GKL *TL
C     FIRST,  GKL*TL.
C
      DO 1100 J=1,3
      DO 1100 I=1,3
      DO 1100 M=1,3
      TLMJ=T(M,J,LL)
      GKLTL(  I,  J)= GKLTL(  I,  J) +S(  I,  M,N) *TLMJ
      GKLTL(  I,J+3)= GKLTL(  I,J+3) +S(  I,M+3,N) *TLMJ
      GKLTL(I+3,  J)= GKLTL(I+3,  J) +S(I+3,  M,N) *TLMJ
 1100 GKLTL(I+3,J+3)= GKLTL(J+3,J+3) +S(I+3,M+3,N) *TLMJ
C
C     TK(TRANSPOSE)*(GKL*TL)
      DO 1200 J=1,3
      DO 1200 I=1,3
      DO 1200 M=1,3
      TKMI=T(M,I,KK)
      HKL(  I,  J)= HKL(  I,  J) +TKMI*GKLTL(  M,  J)
      HKL(  I,J+3)= HKL(  I,J+3) +TKMI*GKLTL(  M,J+3)
      HKL(I+3,  J)= HKL(I+3,  J) +TKMI*GKLTL(M+3,  J)
 1200 HKL(I+3,J+3)= HKL(I+3,J+3) +TKMI*GKLTL(M+3,J+3)
C
C     TRANSPOSE, IF REQ.
      IF(MAP(N).GT.0) GO TO 1400
      DO 1300 J=2,6
      JM=J-1
      DO 1300 I=1,JM
      EIJ= HKL(I,J)
      HKL(I,J)=HKL(J,I)
 1300 HKL(J,I)=EIJ
 1400 LOC=IABS(MAP(N))
      IF(NDF.LT.6) GO TO 1600
      DO 1500 J=1,6
      DO 1500 I=1,6
 1500 H(I,J,LOC)= H(I,J,LOC)+HKL(I,J)
      GO TO 2000
 1600 DO 1700 I=1,NDF
      NROW=INEX(I)
      DO 1700 J=1,NDF
      NCOL=INEX(J)
 1700 H(I,J,LOC)=H(I,J,LOC)+HKL(NROW,NCOL)
 2000 CONTINUE
```

Figure 9

REFERENCES

1. J. Sobieszczanski-Sobieski and R.B. Bhat, "Adaptable Structural Synthesis Using Advanced Analysis and Optimization Coupled By A Computer Operating System", AIAA Paper No. 79-0723, Presented at AIAA/ASME/ASCE/AHS 20th Structures, Structural Dynamics and Materials Conference, St. Louis, Missouri, April 1979.

2. W.D. Whetstone et al., "SPAR Structural Analysis System Reference Manual", Vols. 1,2,3, * 4 NASA CR 158970-1,2,3,4, December, 1978.

3. T.R. Barone and L.J. Feeser, "Beginner's User Manual for SPAR", Report No. 78-1, Department of Civil Engineering, Rensselaer Polytechnic Institute, May 1978.

4. K.E. Ferson, "Performance Evaluation of a Minicomputer/Array Processor System for Finite Element Applications", Master of Engineering Report, Rensselaer Polytechnic Institute, Troy, NY, August 1979.

5. Programmers Reference Manual - Part One, AP-120B Array Processor, Floating Point Systems, Inc., Beaverton, OR, January 1978.

6. Programmers Reference Manual - Part Two, AP-120B Array Processor, Floating Point Systems, Inc., Beaverton, OR, January 1978.

7. Program Development Software Manual, AP-120B Array Processor, Floating Point Systems, Inc., Beaverton, OR, September 1978.

APPENDIX A

Listings of:

FOR1.FOR

APFOR1.FOR

APEGLD.APM

```
FORTRAN IV      V02.5-2    Tue 04-Nov-80 22:05:39             PAGE 001

      COM       05-14-80      RPI# 66666  66666  66666  77777  11111  99999
      C
      C    *********************************************
      C
      C     This is Program FOR1.FOR which represents a portion
      C     of Subroutine EIGLD in Processor EIG.  Used to obtain
      C     timing information for FORTRAN execution.
      C
      C     Corresponding Programs are:
      C         APFOR1.FOR - FORTRAN of FOR1 with AP Calls.
      C         APEGLD.APM - FP120 Assembler Program replacement
      C                 of FORTRAN portions.
      C         APEGLD.ABJ - Object code of APEGLD.APM
      C         APEGLD.SAV - Linked version of APEGLD.APM.
      C
      C    *********************************************
      C
      C     SUBROUTINE EIGLD
      C
            DIMENSION A(6000),INEX(6),RSCALE(6)
            DIMENSION ITIM1(2),ITIM2(2)
            DO 11 II=1,6
            INEX(II)=II
         11 RSCALE(II)=FLOAT(II)
            JT=1000
            JDP=6
            WRITE(6,3001)
       3001 FORMAT(' INPUT THE NUMBER OF TIMES TO BE EXECUTED')
            READ(5,*) NTEST
            CPU1=SECNDS(0.)
            CALL GTIM(ITIM1)
            DO 270 L=1,NTEST
              IRB=0
            DO 1070 J=1,JT
            DO 1065 I=1,JDP
            K=INEX(I)
            IRBPI=IRB+I
       1065 A(IRBPI)=RAN(0,0)*RSCALE(K)
       1070   IRB=IRB+JDP
        270 CONTINUE
            CALL GTIM(ITIM2)
            CPU2=SECNDS(0.)
            WRITE(6,50) CPU1
            WRITE(6,50) CPU2
            CALL CVTTIM(ITIM1,IHR,IMI,ISE,ITI)
            WRITE(6,70) IHR,IMI,ISE,ITI
            CALL CVTTIM(ITIM2,IHR,IMI,ISE,ITI)
            WRITE(6,70) IHR,IMI,ISE,ITI
```

```
FORTRAN IV      V02.5-2    Tue 04-Nov-80 22:05:39              PAGE 002

      50 FORMAT(' NUMBER OF SECONDS PAST MIDNIGHT = ',F15.7)
      70 FORMAT(' TIME = ',I2,':',I2,':',I2,':',I2)
         CPU=CPU2-CPU1
         WRITE(6,22)CPU
       22 FORMAT(5X,'Elapsed Time = ',F10.5,'Seconds')
         STOP
         END
```

```
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:17:38               PAGE 001

      COM       05-14-80      RPI# 66666  66666  66666  77777  11111  99999
      C
      C     ***********************************************
      C
      C     This is Program APFOR1.FOR which contains the AP Calls
      C     as replacements for FORTRAN code in FOR1.FOR.
      C     Represents a portion of Subroutine EIGLD in Processor
      C     EIG.  Obtains timing information for AP execution.
      C
      C     Corresponding Programs are:
      C         FOR1.FOR - Portion of Subroutine EIGLD in
      C                 Processor EIG.
      C         APEGLD.APM -
      C         APEGLD.ABJ -
      C         APEGLD.SAV -
      C
      C     ************************************************
      C
0001        DIMENSION A(6000),INEX(6),RSCALE(6)
0002        DIMENSION ITIM1(2),ITIM2(2),ITIM3(2),ITIM4(2)
0003        DO 11 II=1,6
0004        INEX(II)=II
0005     11 RSCALE(II)=FLOAT(II)
0006        JT=1000
0007        JDF=6
0008        SEED=0.
0009        WRITE(6,3001)
0010   3001 FORMAT(' INPUT THE NUMBER OF TIMES TO BE EXECUTED')
0011        READ(5,*) NTEST
0012        CPU1=SECNDS(0.)
0013        CALL GTIM(ITIM1)
0014        DO 270 L=1,NTEST
      C
      C     What follows are the AP calls which replace the
      C     commented code below.
      C
0015        CALL APCLR
0016        CALL APPUT(SEED,12,1,2)
0017        CALL APPUT(INEX(1),0,JDF,1)
0018        CALL APPUT(RSCALE(1),6,JDF,2)
0019        CALL APWD
      D     CALL GTIM(ITIM2)
0020        CALL APEGLD(JT,JDF)
0021        CALL APWR
      D     CALL GTIM(ITIM3)
0022        CALL APGET(A(1),13,6000,2)
0023        CALL APWD
      C
```

```
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:17:38                PAGE 002

           C      End of AP Calls.
           C
0024         270 CONTINUE
0025              CALL GTIM(ITIM4)
0026              CPU2=SECNDS(0.)
0027              CALL CVTTIM(ITIM1,IHR,IMI,ISE,ITI)
0028              WRITE(6,70)
0029              WRITE(6,75) IHR,IMI,ISE,ITI
           D      CALL CVTTIM(ITIM2,IHR,IMI,ISE,ITI)
           D      WRITE(6,80)
           D      WRITE(6,75) IHR,IMI,ISE,ITI
           D      CALLCVTTIM(ITIM3,IHR,IMI,ISE,ITI)
           D      WRITE(6,90)
           D      WRITE(6,75) IHR,IMI,ISE,ITI
0030              CALL CVTTIM(ITIM4,IHR,IMI,ISE,ITI)
0031              WRITE(6,100)
0032              WRITE(6,75) IHR,IMI,ISE,ITI
0033              WRITE(6,110) CPU1
0034              WRITE(6,110) CPU2
0035              CPU=CPU2-CPU1
0036              WRITE(6,120) CPU
0037          70  FORMAT('$TIME AT START OF DATA INPUT = ')
0038          80  FORMAT('$TIME AT COMPLETION OF DATA INPUT AND EXECUTION',
                 1' START = ')
0039          90  FORMAT('$TIME AT END OF EXECUTION AND START OF DATA'
                 1' OUTPUT = ')
0040         100  FORMAT('$TIME AT COMPLETION OF DATA OUTPUT = ')
0041          75  FORMAT('+',I2,':',I2,':',I2,':',I2)
0042         110  FORMAT(' NUMBER OF SECONDS PAST MIDNIGHT = ',F15.5)
0043         120  FORMAT(5X,'ELAPSED TIME = ',F10.5,' SECONDS')
           C
           C      FORTRAN Code replaced - Begin.
           C
           C         IRB=0
           C       DO 1070 J=1,JT
           C       DO 1065 I=1,JDF
           C       K=INEX(I)
           C       IRBPI=IRB+I
           C 1065 A(IRBPI)=RAN(0,0)*RSCALE(K)
           C 1070    IRB=IRB+JDF
           C
           C      FORTRAN Code replaced - End.
           C
0044              STOP
0045              END
```

```
                   $TITLE APEGLD
                   $ENTRY APEGLD,2
"
"
"
" THIS ROUTINE REPLACES A SECTION OF CODE LOCATED IN THE EIGLD
" SUBROUTINE IN THE EIG PROCESSOR
"
"      Corresponding Programs are:
"              FOR1.FOR
"              APFOR1.FOR
"
" AUTHOR: K. FERSON
" DATE: FEBRUARY 1979
" Revised: L.J. Feeser and K. Matis
" Date: May 1980
"
" ----USAGE----
" FORTRAN: CALL APEGLD(JT,JDF)
"
"        $PAGE
"
"
" ---MAIN DATA MEMORY MAP---
"
" ****************************  (STARTING ADDRESS)
" *                          *
" *     A ARRAY              *
" *                          *
" ****************************  13
" *                          *
" *     SEED                 *
" *                          *
" ****************************  12
" *                          *
" *     RSCALE ARRAY         *
" *                          *
" ****************************  6
" *                          *
" *     INEX ARRAY           *
" *                          *
" ****************************  0
"
"
" ---ARRAY DESCRIPTIONS---
"
" A(JDF*JT).......REAL ARRAY. RANDOM VECTOR OUTPUT
"                 THAT IS RETURNED ONLY ONCE.
"
```

```
" SEED........... RANDOM NUMBER STORED IN PROGRAM
"                 MEMORY.
"
" RSCALE(6)...... REAL ARRAY. PASSED ONCE.
"
" INEX(6)........ INTEGER ARRAY. PASSED ONCE.
"
"
" S-PAD PARAMETERS
"
                    JT     $EQU 0     "TOTAL NUMBER OF JOINTS
                    JDF    $EQU 1     "NUMBER OF DEGREES OF FREEDOM
                    K      $EQU 2     "ADDRESS POINTER FOR RSCALE
                    RADDR  $EQU 3     "BASE ADDRESS OF RSCALE ARRAY
                    CADDR  $EQU 4     "BASE ADDRESS OF OUTPUT ARRAY A
                    INEX   $EQU 5     "BASE ADDRESS OF INEX ARRAY
                    SADDR  $EQU 6     "ADDRESS OF SEED
                    INCNT  $EQU 7     "INNER LOOP COUNTER
                    OUTCNT $EQU 10    "OUTER LOOP COUNTER
"
"
"
" THE RANDOM NUMBER GENERATOR USED IN THIS ROUTINE IS IDENTICAL
" TO FLOATING POINT SYSTEMS RANDOM NUMBER ROUTINE
"
" FORTRAN: IRB=0
"          DO 1070 J=1,JT
"          DO 1065 I=1,JDF
"          K=INEX(I)
"          IRBPI=IRB+I
"     1065 A(IRBPI)=RND(0)*RSCALE(K)
"     1070 IRB=IRB+JDF
"
"
"
"
APEGLD:  LDSPI SADDR; DB=12.                      "LOAD SEED ADDR
         MOV SADDR,SADDR; SETMA                    "GET SEED
         RPSF B; DPX(0)<DB                        "GET MULTIPLIER B
         RPSF FMASK; DPX(2)<DB                     "GET FRACTION MASK
         DPX(1)<DB; DB=40000; WRTMAN
         DPX(1)<DB; DB=1015; WRTEX                "DPX(1)=1.
         FMUL DPX(0),MD                           "MPY SEED*B
                   MOV JT,OUTCNT                  "LOAD OUTER COUNT
                   LDSPI RADDR; DB=5.             "LOAD RSCALE ADDR -1
                   LDSPI CADDR; DB=12.            "LOAD A ADDRESS -1
                   CLR INEX
                   DEC INEX
OUTLOP:            MOV JDF,INCNT                  "LOAD INNER COUNT
INLOP:   FMUL;                                    "PUSH
```

```
                INC INEX; SETMA                     "GET K
        FMUL                                       "PUSH
        FSUB FM,DPX(1); DPY(0)<FM                   "A*B -1, SAVE A*B
        FADD;                                       "PUSH
                LDSPI K; DB=MD                      "STORE K
        FAND DPX(2),DPY(0);                        "ASSUME A*B>1
                                                   "FRACTION EXTRACTED WITH MASK
                ADD# K,RADDR; SETMA                 "GET RSCALE(K)
        FADD ZERO,DPY(0);                           "GET FRACTION DIRECTLY
                                                   "IF B*A<1
        BFGT GT1                                   "GET FRACTION IMMEDIATELY
                                                   "IF B*A>1
        FADD                                       "GET NEXT FA RESULT
GT1:    DPY(2)<FA                                  "SAVE FRACTION
                FMUL DPY(2),MD                        "MULT RSCALE*RND(0)
                FMUL
                FMUL;
                        DEC INCNT                   "DEC COUNT
                MI<FM; INC CADDR; SETMA;
                        BEQ CONT1                   "SAVE RESULT IN A
        FMUL DPX(0),DPY(2)                          "GET NEXT NUMBER
                JMP INLOP
CONT1:                  DEC OUTCNT
                        BEQ CONT2;
                SUB JDF,INEX                        "RESET INEX
        FMUL DPX(0),DPY(2)
                JMP OUTLOP
CONT2:  NOP
"
"
B:      $FP 27.0
FMASK:   $FP .9999999925
        $END
```

APPENDIX B

Listings of:

FOR2.FOR

APFOR2.FOR

APXPD1.APM

```
TYPE FOR2
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:18:01                PAGE 001

      COM       05-14-80      RPI# 66666  66666  66666  77777  22222  11111
      C
      C     ***********************************************
      C
      C      This is Program FOR2.FOR which represents a portion of
      C      Subroutine EXPND1 in Processor EIG.  Used to obtain
      C      timing information for FORTRAN execution.
      C
      C      Corresponding Programs are:
      C          APFOR2.FOR - FORTRAN of FOR2.FOR with AP Calls.
      C          APXPD1.APM - FP120 Assembler Program replacement
      C                  of FORTRAN portions.
      C          APXPD1.ABJ - Object code of APXPD1.APM.
      C          APXPD1.SAV - Linked version of APXPD1.APM.
      C
      C     ******************************************************
      C
             DIMENSION Q(3,3),V2(1000,3),V1(1000,3),KTEST(3)
             DIMENSION ITIM1(2),ITIM2(2)
             JZ=0
             KZ=0
             NV1=3
             NV2=3
             LVEC=1000
             DO 1 JJ=1,NV1
             DO 1 II=1,NV2
           1 Q(II,JJ)=1.0
             DO 2 II=1,NV2
           2 KTEST(II)=1
             KK=LVEC
             WRITE(6,3001)
        3001 FORMAT(' INPUT THE NUMBER OF TIMES TO BE EXECUTED')
             READ(5,*) NTEST
             DO 3 JJ=1,NV1
             DO 3 II=1,KK
             V1(II,JJ)=2.0
           3 V2(II,JJ)=3.0
      C
      D      WRITE(6,2001)
      D      WRITE(6,2000) ((V2(I,J),J=1,3),I=1,3)
      C
             CPU1=SECNDS(0.)
             CALL GTIM(ITIM1)
             DO 270 L=1,NTEST
             DO 260 J=1,NV2
             JJZ=J+JZ
             DO 250 K=1,NV1
             KKZ=K+KZ
```

```
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:18:01          PAGE 002

            QKJ=Q(KKZ,JJZ)
            IF(KTEST(JJZ).NE.0) GO TO 150
            IF(JJZ.NE.KKZ) GO TO 250
            QKJ=1.0
        150 CONTINUE
            DO 200 I=1,LVEC
        200 V2(I, J)=V2(I, J)+V1(I, K)*QKJ
        250 CONTINUE
        260 CONTINUE
        270 CONTINUE
            CALL GTIM(ITIM2)
            CPU2=SECNDS(0.)
      C
      D     WRITE(6,2002)
      D     WRITE(6,2000) ((V2(I,J),J=1,3),I=1,3)
      C
            WRITE(6,50) CPU1
            WRITE(6,50) CPU2
            CALL CVTTIM(ITIM1,IHR,IMI,ISE,ITI)
            WRITE(6,70) IHR,IMI,ISE,ITI
            CALL CVTTIM(ITIM2,IHR,IMI,ISE,ITI)
            WRITE(6,70) IHR,IMI,ISE,ITI
         50 FORMAT(' NUMBER OF SECONDS PAST MIDNIGHT = ',F15.7)
         70 FORMAT(' TIME = ',I2,':',I2,':',I2,':',I2)
            CPU=CPU2-CPU1
            WRITE(6,22)CPU
         22 FORMAT(5X,F16.9)
       2000 FORMAT(3(1X,3F15.6,/))
       2001 FORMAT(' V2 MATRIX BEFORE CALLS')
       2002 FORMAT(' V2 MATRIX AFTER CALLS')
            STOP
            END
```

```
TYPE APFOR2
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:18:27                PAGE 001

      COM      05-14-80      RPI# 66666  66666  66666  77777  22222  11111
      C
      C    **********************************************
      C
      C     This is Program APFOR2.FOR which contains the AP Calls
      C     as replacements for FORTRAN code in FOR2.FOR.  Represents
      C     a portion of Subroutine EXPND1 in Processor EIG.  Obtains
      C     timing information for AP execution.
      C
      C     Corresponding Programs are:
      C         FOR2.FOR -
      C         APXPD1.APM -
      C         APXPD1.ABJ -
      C         APXPD1.SAV -
      C
      C    **********************************************
      C
            DIMENSION Q(3,3),V2(1000,3),V1(1000,3),KTEST(3)
            DIMENSION ITIM1(2),ITIM2(2),ITIM3(2),ITIM4(2)
            JZ=0
            KZ=0
            NV1=3
            NV2=3
            LVEC=1000
            DO 1 JJ=1,NV1
            DO 1 II=1,NV2
          1 Q(II,JJ)=1.0
            DO 2 II=1,NV2
          2 KTEST(II)=1
            KK=LVEC
            WRITE(6,3001)
       3001 FORMAT(' INPUT THE NUMBER OF TIMES TO BE EXECUTED')
            READ(5,*) NTEST
            DO 3 JJ=1,NV1
            DO 3 II=1,KK
            V1(II,JJ)=2.0
          3 V2(II,JJ)=3.0
      C
      D     WRITE(6,2001)
      D     WRITE(6,2000) ((V2(I,J),J=1,3),I=1,3)
      C
            CALL GTIM(ITIM1)
            DO 270 L=1,NTEST
      C
      C     What follows are the AP Calls which replace the
      C     commented code below.
      C
            K=NV1*NV2
```

```
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:18:27                PAGE 002

0024          KBASE=0
0025          CALL APCLR
0026          CALL APPUT(KTEST(1),0,NV2,1)
0027          CALL APPUT(Q(1,1),100,K,2)
0028          CALL APPUT(V1(1,1),109,3000,2)
0029          CALL APPUT(V2(1,1),3109,3000,2)
0030          CALL APWD
      D       CALL GTIM(ITIM2)
0031          CALL APXPD1(KZ,JZ,NV2,NV1,LVEC,100,0,109,3109)
0032          CALL APWR
      D       CALL GTIM(ITIM3)
0033          CALL APGET(V2(1,1),3109,3000,2)
0034          CALL APWD
      C
      C       End of AP Calls.
      C
0035      270 CONTINUE
0036          CALL GTIM(ITIM4)
      C
0037          WRITE(6,2002)
0038          WRITE(6,2000) ((V2(I,J),J=1,3),I=1,3)
      C
0039          CALL CVTTIM(ITIM1,IHR,IMI,ISE,ITI)
0040          WRITE(6,70) IHR,IMI,ISE,ITI
      D       CALL CVTTIM(ITIM2,IHR,IMI,ISE,ITI)
      D       WRITE(6,75) IHR,IMI,ISE,ITI
      D       CALL CVTTIM(ITIM3,IHR,IMI,ISE,ITI)
      D       WRITE(6,80) IHR,IMI,ISE,ITI
0041          CALL CVTTIM(ITIM4,IHR,IMI,ISE,ITI)
0042          WRITE(6,85) IHR,IMI,ISE,ITI
0043       70 FORMAT(' TIME AT START OF DATA INPUT = ',I2,':'
            1,I2,':',I2,':',I2)
0044       75 FORMAT(' TIME AT COMPLETION OF DATA INPUT AND '
            1,'EXECUTION START = ',I2,':',I2,':',I2,':',I2)
0045       80 FORMAT(' TIME AT END OF EXECUTION AND START ',
            1'OF DATA OUTPUT = ',I2,':',I2,':',I2,':',I2)
0046       85 FORMAT(' TIME AT COMPLETION OF DATA OUTPUT = ',
            1I2,':',I2,':',I2,':',I2)
0047     2000 FORMAT( 3(1X,3F15.6,/))
0048     2001 FORMAT(' V2 MATRIX BEFORE CALLS')
0049     2002 FORMAT(' V2 MATRIX AFTER CALLS')
0050          STOP
0051          END
```

```
                    $TITLE APXPD1
                    $ENTRY APXPD1, 9
"
" THIS ROUTINE PERFORMS A SECTION OF CODE IN THE EXPND1 SUBROUTINE
" LOCATED IN THE EIG PROCESSOR.
"
"
" AUTHOR: K. FERSON
" DATE: MARCH 1979
" Revised: L. J. Feeser and K. Matis
" Date: May 1980
"
" ----USAGE----
" FORTRAN: CALL APXPD1(KZ,JZ,NV2,NV1,LVEC,QBASE,KBASE,V1BASE,V2BASE)
"
"           ALL PARAMETERS ARE INTEGERS]
"
"         $PAGE
"
" ---MAIN DATA MEMORY MAP---
"
"  *****************************  (STARTING ADDRESS)
"  *                           *
"  *     V2 ARRAY              *
"  *                           *
"  *****************************  100+N*N+LR1
"  *                           *
"  *     V1 ARRAY              *
"  *****************************  100+N*N
"  *                           *
"  *     Q ARRAY               *
"  *                           *
"  *****************************  100
"  *                           *
"  *     KTEST ARRAY           *
"     K       R  Y             *
"  *****************************  0
"
"
"
" S-PAD PARAMETERS
"
"
                    KZ      $EQU 0      " NTEGER ADDRESS POINTER
                    JZ      $EQU 1      "INTEGER ADDRESS POINTER
                    NV2     $EQU 2      "OUTER LOOP COUNT
                    NV1     $EQU 3      "INNER LOOP CTUT
                    LVEC    $EQU 4      "VECTOR LENGTH
                    QBASE   $EQU 5      "BASE ADDRESS OF Q ARRAY
```

```
                QADDR   $EQU 5      "ADDRESS POINTER FOR Q
                KBASE   $EQU 6      "BASE ADDRESS OF KTEST ARRAY
                KADDR   $EQU 6      "ADDRESS POINTER OF KTEST
                CNT     $EQU 7      "INNER MOST COUNTER
                V1BASE  $EQU 7      "BASE ADDRESS OF V1 ARRAY
                V2BASE  $EQU 10     "BASE ADDRESS OF V2 ARRAY
                V1ADDR  $EQU 11     "ADDRESS POINTER OF V1
                V2ADDR  $EQU 12     "ADDRESS POINTER OF V2
                V2FADR  $EQU 13     "ADDRESS POINTER FOR V2 TARGET
                J       $EQU 14     "OUTER LOOP COUNT
                K       $EQU 15     "INNER LOOP COUNT
                VALUE   $EQU 16     "TEMPORARY STORAGE
                V1TEMP  $EQU 17      "BASE POINTER FOR V1
"
"
"
"
" FORTRAN:  DO 260 J=1,NV2
"           JJZ=J+JZ
"
APXPD1:         CLR J
        DEC QBASE                             "LOOP SET UP
        DPY(0)<SPFN; MOV V1BASE,V1BASE         "STORE V1 BASE IN DATA PAD
"
" OUTER LOOP
"
LOOP1:  CLR K
        INC JZ                                "GET JZ+J
        LDSPI V1BASE; DB=DPY(0)                "RESTORE V1 BASE
        MOV V1BASE,V1TEMP;                      "LOAD BASE POINTER FOR V1
        DPY(0)<SPFN                            "SAVE V1 BASE IN DATA PAD
                                              "THIS FREES SPAD 7
"
" FIRST INNER LOOP
"
" FORTRAN: DO 250 K=1,NV1
"          KKZ=K+KZ
"          QKJ=Q(KKZ,JJZ)
"          IF (KTEST(JJZ) .NE. 0)GO TO 150
"          IF (JJZ .NE. KKZ) GO TO 250
"          QKJ=1.0
"      150 CONTINUE
"
"
LOOP2:  MOV KADDR,KADDR; SETMA                 "GET KTEST(JJZ)
        INC KZ                                "GET KZ+K
        INC QADDR; SETMA                      "GET Q(KZ+Z,JZ+J)
        LDSPI VALUE; DB=MD                     "STORE KTEST(JJZ)
        MOV VALUE,VALUE                        "TEST FOR KTEST(JJZ)=0
        BEQ CONT1
```

```
        DPX(0)<MD                                    "STORE QJK=Q(KKZ,JJZ)
        JMP START3                                   "GOTO INNER MOST LOOP
CONT1:  SUB# JZ,KZ                                    "TEST JJZ=KKZ
        BNE CONT2
        RPSF ONE
        DPX(0)<DB                       "SET QJK=1.0
"
" DO THE 200 LOOP CALCULATIONS
"
" FORTRAN:      DO 200 I=1,LVEC
"            200 V2(I,J)=V2(I,J)+V1(I,K)*QKJ
"
"
START3: MOV V1TEMP,V1ADDR; SETMA                      "GET V1
        MOV LVEC,CNT                                 "LOAD INNER COUNT
        MOV V2BASE,V2ADDR; SETMA                     "GET V2
        FMUL DPX(0),MD;                              "DO QJK*V1
                   MOV V2ADDR,V2FADR                 "LOAD TARGET ADDRESS
        FMUL;                                        "PUSH
                   DEC V2FADR                         "LOOP SETUP
LOOP3:             FMUL                              "PUSH
        INC V1ADDR; SETMA;                           "GET NEXT V1
                   FADD FM,MD                         "DO V2+V1*QJK
                   FADD                              "PUSH
        INC V2ADDR; SETMA                           "GET  NEXT V2
        FMUL DPX(0),MD;                             "DO V1*QJK
                             DEC CNT                  "TEST CNT=0
END3:   INC V2FADR; SETMA; MI<FA;                     "STORE RESULT
        FMUL;                                          "PUSH
                             BNE LOOP3
"
" FORTRAN: 250 CONTINUE
"
"
CONT2:  INC K
        SUB# K,NV1                                   "TEST INNER LOOP
        BEQ CONT3;
                   ADD LVEC,V1TEMP                    "REAJUST V1 ADDRESS
        JMP LOOP2
"
"
" FORTRAN: 260 CONTINUE
"
CONT3:  INC J                                        "INC OUTER LOOP COUNT
        INC KADDR                                    "INCREMENT KTEST ADDRESS
        SUB# J,NV2                                    "TEST OUTER LOOP
        BEQ CONT4;
                   ADD LVEC,V2BASE                     "REAJUST V2 ADDRESS
        JMP LOOP1
ONE:    $FP 1.0
```

```
CONT4:   RETURN
         SEND
```

APPENDIX C

Listings of:

FOR3.FOR

APFOR3.FOR

APNWX1.APM

```
.TYPE FOR3
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:18:59                PAGE 001

      COM        05-14-80      RPI# 66666  66666  66666  77777  22222  44444
      C
      C    ***********************************************
      C
      C     This is Program FOR3.FOR which represents a portion
      C     of Subroutine NEWX in Processor EIG.  Used to obtain
      C     timing information for FORTRAN execution.
      C
      C     Corresponding Programs are:
      C         APFOR3.FOR - FORTRAN of FOR3.FOR with AP Calls.
      C         APNWX1.APM - FP120 Assembler Program replacement
      C                 of FORTRAN portions.
      C         APNWX1.ABJ - Object code of APNWX1.APM.
      C         APNWX1.SAV - Linked version of APNWX1.ABJ.
      C
      C    *******************************************************
      C
            DIMENSION LSDO(3),V2(2000,3)
            DIMENSION ITIM1(2),ITIM2(2)
            NLOADS=3
            LVEC=2000
            DO 10 II=1,NLOADS
         10 LSDO(II)=II
            KNTLS=0
            WRITE(6,3001)
       3001 FORMAT(' INPUT THE NUMBER OF TIMES TO BE EXECUTED')
            READ(5,*) NTEST
            CPU1=SECNDS(0.)
            CALL GTIM(ITIM1)
            DO 270 L=1,NTEST
            DO 70 N=1,NLOADS
            IF(LSDO(N) .EQ. 0)GO TO 70
            DO 60 I=1,LVEC
         60 V2(I,N)=0.
         70 CONTINUE
        270 CONTINUE
            CALL GTIM(ITIM2)
            CPU2=SECNDS(0.)
            WRITE(6,50) CPU1
            WRITE(6,50) CPU2
            CALL CVTTIM(ITIM1,IHR,IMI,ISE,ITI)
            WRITE(6,80) IHR,IMI,ISE,ITI
            CALL CVTTIM(ITIM2,IHR,IMI,ISE,ITI)
            WRITE(6,80) IHR,IMI,ISE,ITI
         50 FORMAT(' NUMBER OF SECONDS PAST MIDNIGHT = ',F15.7)
         80 FORMAT(' TIME = ',I2,':',I2,':',I2,':',I2)
            CPU=CPU2-CPU1
            WRITE(6,22)CPU
```

```
0033      22 FORMAT(5X,'TIME=',F16.9)
0034         STOP
0035         END
```

```
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:19:23              PAGE 001

      COM       05-14-80      RPI# 66666  66666  66666  77777  22222  44444
      C
      C     ****************************************************
      C
      C     This is Program APFOR3.FOR which contains the AP Calls
      C     as replacements for FORTRAN code in FOR3.FOR.  Represents
      C     a portion of Subroutine NEWX in Processor EIG.
      C
      C     Corresponding Programs are:
      C         FOR3.FOR -
      C         APNWX1.APM -
      C         APNWX1.ABJ -
      C         APNWX1.SAV -
      C
      C     ****************************************************
      C
            DIMENSION LSDO(3),V2(2000,3)
            DIMENSION ITIM1(2),ITIM2(2),ITIM3(2),ITIM4(2)
            NLOADS=3
            LVEC=2000
            DO 10 II=1,NLOADS
         10 LSDO(II)=II
            KNTLS=0
            WRITE(6,3001)
       3001 FORMAT(' INPUT THE NUMBER OF TIMES TO BE EXECUTED')
            READ(5,*) NTEST
            CPU1=SECNDS(0.)
            CALL GTIM(ITIM1)
            DO 270 L=1,NTEST
            CALL APCLR
            CALL APPUT(LSDO(1),0,NLOADS,1)
            CALL  APWD
      D     CALL GTIM(ITIM2)
            CALL APNWX1(NLOADS,KNTLS,LVEC,NLOADS)
            CALL APWR
      D     CALL GTIM(ITIM3)
            CALL APGET(V2(1,1),NLOADS,6000,2)
            CALL APWD
        270 CONTINUE
            CALL GTIM(ITIM4)
            CPU2=SECNDS(0.)
            CALL CVTTIM(ITIM1,IHR,IMI,ISE,ITI)
            WRITE(6,70)
            WRITE(6,75) IHR,IMI,ISE,ITI
      D     CALL CVTTIM(ITIM2,IHR,IMI,ISE,ITI)
      D     WRITE(6,80)
      D     WRITE(6,75) IHR,IMI,ISE,ITI
      D     CALL CVTTIM(ITIM3,IHR,IMI,ISE,ITI)
```

```
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:19:23                PAGE 002

      D      WRITE(6,90)
      D      WRITE(6,75) IHR,IMI,ISE,ITI
             CALL CVTTIM(ITIM4,IHR,IMI,ISE,ITI)
             WRITE(6,100)
             WRITE(6,75) IHR,IMI,ISE,ITI
             WRITE(6,110) CPU1
             WRITE(6,110) CPU2
             CPU=CPU2-CPU1
             WRITE(6,120) CPU
         70  FORMAT('$TIME AT START OF DATA INPUT = ')
         80  FORMAT('$TIME AT COMPLETION OF DATA INPUT AND EXECUTION',
            1' START = ')
         90  FORMAT('$TIME AT END OF EXECUTION AND START OF DATA'
            1' OUTPUT = ')
        100  FORMAT('$TIME AT COMPLETION OF DATA OUTPUT = ')
         75  FORMAT('+',I2,':',I2,':',I2,':',I2)
        110  FORMAT(' NUMBER OF SECONDS PAST MIDNIGHT = ',F15.5)
        120  FORMAT(5X,'ELAPSED TIME = ',F10.5,' SECONDS')
      C
      C      FORTRAN code replaced - Begin
      C
      C       DO 70 N=1,NLOADS
      C       IF(LSDO(N) .EQ. 0)GO TO 70
      C       DO 60 I=1,LVEC
      C    60 V2(I,N)=0.
      C    70 CONTINUE
      C
      C      FORTRAN code replaced - End
      C
             STOP
             END
```

```
"
"
                    $TITLE APNWX1
                    $ENTRY APNWX1,4
"
"
"
" THIS ROUTINE PERFORMS A SECTION OF CODE IN THE NEWX SUBROUTINE
" LOCATED IN THE EIG PROCESSOR
"
" AUTHOR: K. FERSON
" DATE: MARCH 1979
" Revised: L. J. Feeser and K. Matis
" Date: May 1980
"
" ----USAGE----
" FORTRAN: CALL APNWX1(NLOADS,KNTLS,LVEC,V2ADDR)
"
"         $PAGE
"
"
" ---MAIN DATA MEMORY MAP---
"
"  **************************  (STARTING ADDRESS)
"  *                        *
"  *     Z ARRAY            *
"  *                        *
"  **************************  NLOADS+LRM+2*(LVEC*NLOADS)
"  *                        *
"  *     V1 ARRAY           *
"  *                        *
"  **************************  NLOADS+LRM+LVEC*NLOADS
"  *                        *
"  *     V2 ARRAY           *
"  *                        *
"  **************************  NLOADS+LRM
"  *                        *
"  *     B ARRAY            *
"  *                        *
"  **************************  NLOADS
"  *                        *
"  *     LSDO ARRAY         *
"  *                        *
"  **************************  0
"
"
" ---ARRAY DESCRIPTIONS---
"
"  Z(NLOADS)......... REAL ARRAY. NEVER USED.
"
```

```
"  V1(LVEC*NLOADS)... REAL ARRAY. PREVIOUSLY PASSED.
"
"  V2(LVEC*NLOADS)... REAL ARRAY. RETURNED ONCE.
"
"  B(LRM)............ REAL ARRAY. NEVER USED.
"
"  LSDO(NLOADS)...... INTEGER ARRAY. PREVIOUSY PASSED.
"
"
"
"
"
"
"
"
"
" S-PAD PARAMETERS
"
                   NLOADS $EQU 0     "NUMBER OF LOADS
                   KNTLS  $EQU 1     "INTEGER PASSED
                   LVEC   $EQU 2     "TOTAL JOINTS*DEGREES OF FREEDOM
                   V2ADDR $EQU 3     "ADDRESS POINTER FOR V2
                   LSDO   $EQU 4     "INTEGER MAP
                   VALUE  $EQU 5     "TEMPORARY STORAGE
                   OUTCNT $EQU 6     "OUTER LOOP COUNT
                   INCNT  $EQU 7     "INNER LOOP COUNT
"
"
" FORTRAN: DO 70 N=1,NLOADS
"          IF(LSDO(N) .EQ. 0)GO TO 70
"          KNTLS=KNTLS+1
"          DO 60 I=1,LVEC
"       60 V2(I,N)=0.
"       70 CONTINUE
"
"
         DEC V2ADDR                               "SET UP V2ADDR FOR LOOP
         MOV NLOADS,OUTCNT                        "LOAD OUTER COUNT
         CLR LSDO                                 "CLEAR ADDRESS POINTER
         DEC LSDO
OUTLOP:  INC LSDO; SETMA                          "GET LSDO(N)
         MOV LVEC,INCNT                           "LOAD INNER COUNTER
         NOP
         LDSPI VALUE; DB=MD                       "VALUE=LSDO(N)
         MOV VALUE,VALUE                          "TEST VALUE=0
         BNE CONT1
         ADD LVEC,V2ADDR                          "AJUST V2ADDR
         JMP CONT2
"
CONT1:   INC KNTLS
```

```
LOOP:    DEC INCNT
         INC V2ADDR;SETMA;MI<ZERO;BNE LOOP        "CLR V2
CONT2:   DEC OUTCNT                               "TEST OUTER LOOP
         BNE OUTLOP
         $END
```

APPENDIX D

Listings of:

FOR4.FOR

APFOR4.FOR

APNWX2.APM

```
   TYPE FOR4
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:20:24              PAGE 001

      COM       05-14-80      RPI# 66666  66666  66666  77777  22222  55555
      C
      C     ************************************************
      C
      C      This is Program FOR4.FOR which represents a portion of
      C      Subroutine NEWX in Processor EIG.  Used to obtain
      C      timing information for FORTRAN execution.
      C
      C      Corresponding Programs are:
      C          APFOR4.FOR - FORTRAN of FOR4.FOR with AP Calls.
      C          APNWX2.APM - FP120 Assembler Program replacement
      C                  of FORTRAN portion.
      C          APNWX2.ABJ - Object code of APNWX2.APM.
      C          APNWX2.SAV - Linked version of APNWX2.ABJ.
      C
      C     ********************************************************
      C
0001         DIMENSION V2(1000,4),V1(1000,4),LSDO(4),Z(4)
0002         DIMENSION ITIM1(2),ITIM2(2)
0003         NLOADS=4
0004         LVEC=1000
0005         DO 11 II=1,NLOADS
0006      11 LSDO(II)=II
0007         DO 12 JJ=1,4
0008         DO 12 II=1,1000
0009         V1(II,JJ)=2.
0010         V2(II,JJ)=1.
0011      12 CONTINUE
0012         WRITE(6,3001)
0013    3001 FORMAT(' INPUT THE NUMBER OF TIMES TO BE EXECUTED')
0014         READ(5,*) NTEST
      C
      D      WRITE(6,90)
      D      WRITE(6,95) ((V1(I,J),J=1,4),I=1,3)
0015         CPU1=SECNDS(0.)
0016         CALL GTIM(ITIM1)
0017         DO 270 L=1,NTEST
0018         DO 1300 N=1,NLOADS
0019         IF(LSDO(N).EQ.0) GO TO 1300
0021         SUM=.0
0022         DO 1100 I=1,LVEC
0023    1100 SUM=SUM+V2(I,N)*V1(I,N)
0024         SUM=1./SQRT(ABS(SUM))
0025         Z(N)=SUM
0026         DO 1200 I=1,LVEC
0027         V1(I,N)=V1(I,N)*SUM
0028    1200 V2(I,N)=V2(I,N)*SUM
0029    1300 CONTINUE
```

```
0030     270 CONTINUE
0031         CALL GTIM(ITIM2)
0032         CPU2=SECNDS(0.)
        C
        D    WRITE(6,100)
        D    WRITE(6,95) ((V1(I,J),J=1,4),I=1,3)
        D    WRITE(6,110)
        D    WRITE(6,95) ((V2(I,J),J=1,4),I=1,3)
0033         WRITE(6,50) CPU1
0034         WRITE(6,50) CPU2
0035         CALL CVTTIM(ITIM1,IHR,IMI,ISE,ITI)
0036         WRITE(6,70) IHR,IMI,ISE,ITI
0037         CALL CVTTIM(ITIM2,IHR,IMI,ISE,ITI)
0038         WRITE(6,70) IHR,IMI,ISE,ITI
0039      50 FORMAT(' NUMBER OF SECONDS PAST MIDNIGHT = ',F15.7)
0040      70 FORMAT(' TIME = ',I2,':',I2,':',I2,':'I2)
0041      90 FORMAT(' V1 MATRIX BEFORE CALL')
0042      95 FORMAT(3(1X,4F15.7,/))
0043     100 FORMAT(' V1 MATRIX AFTER RETURN')
0044     110 FORMAT(' V2 MATRIX AFTER RETURN')
0045         CPU=CPU2-CPU1
0046         WRITE(6,23)CPU
0047      23 FORMAT(5X,'TIME=',F16.9)
0048         STOP
0049         END
```

```
 TYPE APFOR4
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:20:48                PAGE 001

      COM       05-14-80      RPI# 66666  66666  66666  77777  22222  55555
      C
      C     ******************************************************
      C
      C      This is Program APFOR4.FOR which contains the AP Calls
      C      as replacements for FORTRAN code in FOR4.FOR. Represents
      C      a portion of Subroutine NEWX in Processor EIG. Obtains
      C      timing information for AP execution.
      C
      C      Corresponding Programs are:
      C          FOR4.FOR
      C          APNWX2.APM
      C          APNWX2.ABJ
      C          APNWX2.SAV
      C
      C     ******************************************************
      C
0001         DIMENSION V2(1000,4),V1(1000,4),LSDO(4),Z(4)
0002         DIMENSION ITIM1(2),ITIM2(2),ITIM3(2),ITIM4(2)
0003         NLOADS=4
0004         LVEC=1000
0005         DO 11 II=1,NLOADS
0006      11 LSDO(II)=II
0007         DO 12 JJ=1,4
0008         DO 12 II=1,1000
0009         V1(II,JJ)=2.
0010         V2(II,JJ)=1.
0011      12 CONTINUE
0012         WRITE(6,3001)
0013    3001 FORMAT(' INPUT THE NUMBER OF TIMES TO BE EXECUTED')
0014         READ(5,*) NTEST
      D      WRITE(6,100)
      D      WRITE(6,90) ((V1(I,J),J=1,4),I=1,3)
      C
      C    THE FOLLOWING CALLS REPLACE THE FORTRAN COMMENTED BELOW
      C
0015         CALL GTIM(ITIM1)
0016         DO 270 L=1,NTEST
0017         CALL APCLR
0018         CALL APPUT(LSDO(1),0,NLOADS,1)
0019         CALL APPUT(V1(1,1),100,4000,2)
0020         CALL APPUT(V2(1,1),4101,4000,2)
0021         CALL APWD
      D      CALL GTIM(ITIM2)
0022         CALL APNWX2(100,4101,LVEC,NLOADS,8101)
0023         CALL APWR
      D      CALL GTIM(ITIM3)
0024         CALL APGET(V1(1,1),100,4000,2)
```

```
0025          CALL APGET(V2(1,1),4101,4000,2)
0026          CALL APWD
0027      270 CONTINUE
0028          CALL GTIM(ITIM4)
      C
      C       FORTRAN code replaced - Begin
      C
      C        DO 1300 N=1,NLOADS
      C        IF(LSDO(N).EQ.0) GO TO 1300
      C        SUM=.0
      C        DO 1100 I=1,LVEC
      C 1100   SUM=SUM+V2(I,N)*V1(I,N)
      C        SUM=1./SQRT(ABS(SUM))
      C        Z(N)=SUM
      C        DO 1200 I=1,LVEC
      C        V1(I,N)=V1(I,N)*SUM
      C 1200 V2(I,N)=V2(I,N)*SUM
      C 1300 CONTINUE
      C
      C       FORTRAN code replaced - End
      C
0029          CALL CVTTIM(ITIM1,IHR,IMI,ISE,ITI)
0030          WRITE(6,70) IHR,IMI,ISE,ITI
      D       CALL CVTTIM(ITIM2,IHR,IMI,ISE,ITI)
      D       WRITE(6,75) IHR,IMI,ISE,ITI
      D       CALL CVTTIM(ITIM3,IHR,IMI,ISE,ITI)
      D       WRITE(6,80) IHR,IMI,ISE,ITI
0031          CALL CVTTIM(ITIM4,IHR,IMI,ISE,ITI)
0032          WRITE(6,85) IHR,IMI,ISE,ITI
0033       70 FORMAT(' TIME AT START OF DATA TRANSFER = ',
             1I2,':',I2,':',I2,':'I2)
0034       75 FORMAT(' TIME AT COMPLETION OF DATA INPUT AND '
             1,'EXECUTION START = ',I2,':',I2,':',I2,':',I2)
0035       80 FORMAT(' TIME AT END OF EXECUTION AND START ',
             1'OF DATA OUTPUT = ',I2,':',I2,':',I2,':',I2)
0036       85 FORMAT(' TIME AT COMPLETION OF DATA OUTPUT = ',
             1I2,':',I2,':',I2,':',I2)
      D       WRITE(6,95)
      D       WRITE(6,90) ((V1(I,J),J=1,4),I=1,3)
      D       WRITE(6,97)
      D       WRITE(6,90) ((V2(I,J),J=1,4),I=1,3)
0037       90 FORMAT(3(1X,4F15.7,/))
0038       95 FORMAT(' V1 MATRIX AFTER RETURN')
0039       97 FORMAT(' V2 MATRIX AFTER RETURN')
0040      100 FORMAT(' V1 MATRIX BEFORE CALL')
0041          STOP
0042          END
```

```
"
"
                   $TITLE APNWX2
                  $ENTRY APNWX2, 5
                   $EXT DIV
                   $EXT SQRT
"
"
" THIS ROUTINE PERFORMS A SECTION OF CODE IN THE NEWX SUBROUTINE
" LOCATED IN THE FIG PROCESSOR
"
"
" AUTHOR: K. FERSON
" DATE: FEBRUARY 1979
" Revised: L. J. Feeser and K. Matis
" Date: May 1980
"
" ----USAGE----
" FORTRAN: CALL APNWX2(V1BASE,V2BASE,LVEC,NLOAD,ZBASE)
"
"            ALL PARAMETERS ARE INTEGERS]
"
"         $PAGE
"
"
" ---MAIN DATA MEMORY MAP---
"
"  **************************  (STARTING ADDRESS)
"  *                        *
"  *      Z ARRAY           *
"  *                        *
"  **************************  NLOADS+LRM+2*(LVEC*NLOADS)
"  *                        *
"  *      V1 ARRAY          *
"  *                        *
"  **************************  NLOADS+LRM+LVEC*NLOADS
"  *                        *
"  *      V2 ARRAY          *
"  *                        *
"  **************************  NLOADS+LRM
"  *                        *
"  *      B ARRAY           *
"  *                        *
"  **************************  NLOADS
"  *                        *
"  *      LSDO ARRAY        *
"  *                        *
"  **************************  0
"
"
```

```
" ---ARRAY DESCRIPTIONS---
"
"  Z(NLOADS)......... REAL ARRAY. NEVER PASSED.
"
"  V1(LVEC*NLOADS)... REAL ARRAY. PREVIOUSLY PASSED.
"                     RETURNED ONLY ONCE.
"
"  V2(LVEC*NLOADS)... REAL ARRAY. RETURNED ONCE.
"
"  B(LRM)............ REAL ARRAY. NEVER USED.
"
"  LSDO(NLOADS)...... INTEGER ARRAY. PREVIOUSY PASSED.
"
"
"
" S-PAD PARAMETERS
"
"
                  V1BASE $EQU 0     "BASE OF V1 ARRAY
                  V2BASE $EQU 1     "BASE OF V2 ARRAY
                  LVEC   $EQU 2     "TOTAL JOINTS * DEGREES OF FREDDOM
                  NLOAD  $EQU 3     "NUMBER OF LOADS
                  ZBASE  $EQU 4     "ADDDRESS OF Z ARRAY
                  V1ADDR $EQU 5     "ADDRESS OF V1
                  V2ADDR $EQU 6     "ADDRESS OF V2
                   CNT   $EQU 7     "INNER COUNTER
                  LSADDR $EQU 10    "ADDRESS OF LSDO ARRAY
                  OUTCNT $EQU 10    "OUTER LOOP COUNTER
                  VALUE  $EQU 11     "TEMPORARY STORAGE
                  V1DEST $EQU 12     "V1 ADDRESS FOR WRITE
                  V2DEST $EQU 13     "V2 ADDRESS FOR WRITE
"
"
" FORTRAN: DO 1300 N=1,NLOADS
"          IF(LSDO(N) .EQ. 0) GO TO 1300
"          SUM=0.
"
"
APNWX2:  CLR OUTCNT                          "CLEAR 1300 LOOP COUNTER
LOOP1:   MOV LSADDR,LSADDR;SETMA                 "GET LSDO(N)
         DPX(0)<DB; DB=ZERO                 "CLEAR DPX(0)=SUM
         NOP
         LDSPI VALUE; DB=MD                  "STORE LSDO(N)
         MOV VALUE,VALUE                    "TEST LSDO(N)
         BNE CONT1
         JMP CONT2
"
"
" 1100 LOOP CALCULATIONS
"
```

```
"
" FORTRAN:      DO 1100 I=1,LVEC
"          1100 SUM=SUM+V2(I,N)*V1(I,N)
"
"
"DPX(0) SHOULD BE ZERO
"
CONT1:   MOV V1BASE,V1ADDR; SETMA                  "GET V(1,N)
         MOV LVEC,CNT                              "LOAD INNER COUNTER
         MOV V2BASE,V2ADDR; SETMA                  "GET V2(1,N)
         DPX(1)<MD                                "SAVE V1
         INC V1ADDR; SETMA                        "GET V1(2,N)
         FMUL DPX(1),MD                           "DO V1*V2
         INC V2ADDR; SETMA;                       "GET V2(2,N)
                   FMUL                            "PUSH
LOOP2:             DPX(1)<MD;                      "SAVE V1(LVEC,N)
                             FMUL                 "PUSH
         INC V1ADDR; SETMA;                       "GET V2(LVEC,N)
                             FADD FM,DPX(0)        "V2*V1+SUM
                   FMUL DPX(1),MD;                "V1*V2
                             FADD;                "PUSH
                                       DEC CNT     "TEST LOOP
         INC V2ADDR; SETMA;                        "GET V2(LVEC,N)
                   FMUL;DPX(0)<FA;BNE LOOP2        "SAVE SUM
"
"
" FORTRAN: SUM=1.0/SQRT(ABS(SUM))
"          Z(N)=SUM
"
"
"
         JSR SQRT                                 "SQRT(ABS(SUM))
        LDTMA;DB=!ONE
        NOP
        DPY(0)<TM                                "SET DPY=1
         JSR DIV                                  "DPY(1) DIV DPX(0)
"
"               OR THE INVERSE OF SQRT(ABS(SUM))
"
         ADD# OUTCNT,ZBASE; SETMA; MI<DPX(0)        "STORE Z(N)=SUM
"
"
"                THE RESULT IS IN DPX(0)=SUM
"                DO THE 1200 LOOP CALCULATIONS
"
"
"  FORTRAN:    DO 1200 I=1,LVEC
"              V1(I,N)=V1(I,N)*SUM
"        1200 V2(I,N)=V2(I,N)*SUM
"        1300 CONTINUE
```

```
"
"
        NOP                                          "WAIT TO ACCESS MD
        NOP                                          "WAIT SOME MORE
          MOV V1BASE,V1ADDR; SETMA                     "GET V1(1,N)
          MOV V1ADDR,V1DEST                           "LOAD TARGET ADDRESS
          MOV V2BASE,V2ADDR; SETMA                     "GET V2(1,N)
          FMUL DPX(0),MD;                              "SUM*V1
                    MOV LVEC,CNT                       "LOAD INNER COUNT
          MOV V2ADDR,V2DEST                           "LOAD TARGET ADDRESS
          FMUL DPX(0),MD;                              "SUM*V2
          DEC V2DEST                                  "LOOP SETUP
          FMUL;                                        "PUSH
          DEC V1DEST                                   "LOOP SETUP
LOOP3:              FMUL; DPX(1)<FM                    "SAVE V1*SUM
          INC V1ADDR; SETMA;                          "GET NEXT V1
                    DPY(1)<FM                         "SAVE V2*SUM
          NOP
          INC V2ADDR; SETMA                            "GET NEXT V2
          FMUL DPX(0),MD                               "V1*SUM
          INC V1DEST; MI<DPX(1); SETMA                        "STORE V1*SUM
          FMUL DPX(0),MD                              "V2*SUM
                    DEC CNT                           "TEST LOOP COUNT
          INC V2DEST; SETMA; MI<DPY(1);                "STORE V2*SUM
                    BNE LOOP3;FMUL
CONT2:    ADD LVEC,V1BASE                             "INCREMENT BASE
          INC OUTCNT                                  "INC COUNTER AND LSDO ADDRESS
          SUB# NLOAD,OUTCNT                            "TEST COUNTER
          BEQ END; ADD LVEC,V2BASE
          JMP LOOP1
ONE:      $FP 1.0
END:     RETURN
          $END
```

APPENDIX E

Listings of:

FOR5.FOR

APFOR5.FOR

APNWX3.APM

```
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:21:09                PAGE 001

      COM       05-14-80      RPI# 66666  66666  66666  77777  22222  66666
      C
      C     ***********************************************************
      C
      C      This is Program FOR5.FOR which represents a portion of
      C      Subroutine NEWX in Processor EIG. Used to obtain timing
      C      information for FORTRAN execution.
      C
      C      Corresponding Programs are:
      C          APFOR5.FOR - FORTRAN of FOR5. FOR with AP Calls.
      C          APNWX3.APM - FP120 Assembler Program replacement
      C                  of FORTRAN portions.
      C          APNWX3.ABJ - Object code of APNWX3.APM.
      C          APNWX3.SAV - Linked version of APNWX3.ABJ.
      C
      C     ************************************************************
      C
0001             DIMENSION LSDO(3),Z(3),V1(1000,3)
0002             DIMENSION ITIM1(2),ITIM2(2)
0003             NLOADS=3
0004             LVEC=1000
0005             DO 1 II=1,3
0006             LSDO(II)=II
0007           1 Z(II)=FLOAT(II)
0008             DO 2 JJ=1,3
0009             DO 2 II=1,1000
0010           2 V1(II,JJ)=2.0
0011             WRITE(6,3001)
0012        3001 FORMAT(' INPUT THE NUMBER OF TIMES TO BE EXECUTED')
0013             READ(5,*) NTEST
0014             CPU1=SECNDS(0.)
0015             CALL GTIM(ITIM1)
0016             DO 270 L=1,NTEST
0017             DO 1850 N=1,NLOADS
0018             IF (LSDO(N) .EQ. 0) GO TO 1850
0020             SUM=Z(N)
0021             DO 1800 I=1,LVEC
0022        1800 V1(I,N)=V1(I,N)*SUM
0023        1850 CONTINUE
0024         270 CONTINUE
0025             CALL GTIM(ITIM2)
0026             CPU2=SECNDS(0.)
0027             WRITE(6,50) CPU1
0028             WRITE(6,50) CPU2
0029             CALL CVTTIM(ITIM1,IHR,IMI,ISE,ITI)
0030             WRITE(6,70) IHR,IMI,ISE,ITI
0031             CALL CVTTIM(ITIM2,IHR,IMI,ISE,ITI)
0032             WRITE(6,70) IHR,IMI,ISE,ITI
```

```
FORTRAN IV      V02.5-2     Tue 04-Nov-80 10:21:09                PAGE 002

      50  FORMAT(' NUMBER OF SECONDS PAST MIDNIGHT = ',F15.7)
      70  FORMAT(' TIME = ',I2,':',I2,':',I2,':',I2)
          CPU=CPU2-CPU1
          WRITE(6,22)CPU
       22 FORMAT(5X,'TIME=',F16.9)
          STOP
          END
```

```
TYPE APFOR5
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:21:34              PAGE 001

      COM       05-14-80      RPI# 66666  66666  66666  77777  22222  66666
      C
      C    **************************************************
      C
      C     This is Program APFOR5.FOR which contains the AP Calls
      C     as replacement for FORTRAN code in FOR5.FOR. Represents
      C     a portion of Subroutine NEWX in Processor EIG. Obtains
      C     timing information for AP execution.
      C
      C     Corresponding Programs are:
      C         FOR5.FOR
      C         APNWX3.APM
      C         APNWX3.ABJ
      C         APNWX3.SAV
      C
      C    **************************************************
      C
            DIMENSION LSDO(3),Z(3),V1(1000,3)
            DIMENSION ITIM1(2),ITIM2(2),ITIM3(2),ITIM4(2)
            NLOADS=3
            LVEC=1000
            DO 1 II=1,3
            LSDO(II)=II
          1 Z(II)=FLOAT(II)
            DO 2 JJ=1,3
            DO 2 II=1,1000
          2 V1(II,JJ)=2.0
            WRITE(6,3001)
       3001 FORMAT(' INPUT THE NUMBER OF TIMES TO BE EXECUTED')
            READ(5,*) NTEST
            CPU1=SECNDS(0.)
            CALL GTIM(ITIM1)
            DO 270 L=1,NTEST
            CALL APCLR
            CALL APPUT(LSDO(1),0,NLOADS,1)
            CALL APPUT(V1(1,1),10,3000,2)
            CALL APPUT(Z(1),6000,NLOADS,2)
            CALL APWD
      D     CALL GTIM(ITIM2)
            CALL APNWX3(10,6000,LVEC,NLOADS)
            CALL APWR
      D     CALL GTIM(ITIM3)
            CALL APGET(V1(1,1),10,3000,2)
            CALL APWD
        270 CONTINUE
            CALL GTIM(ITIM4)
            CPU2=SECNDS(0.)
            CALL CVTTIM(ITIM1,IHR,IMI,ISE,ITI)
```

```
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:21:34               PAGE 002

0030            WRITE(6,70)
0031            WRITE(6,75) IHR,IMI,ISE,ITI
      D         CALL CVTTIM(ITIM2,IHR,IMI,ISE,ITI)
      D         WRITE(6,80)
      D         WRITE(6,75) IHR,IMI,ISE,ITI
      D         CALLCVTTIM(ITIM3,IHR,IMI,ISE,ITI)
      D         WRITE(6,90)
      D         WRITE(6,75) IHR,IMI,ISE,ITI
0032            CALL CVTTIM(ITIM4,IHR,IMI,ISE,ITI)
0033            WRITE(6,100)
0034            WRITE(6,75) IHR,IMI,ISE,ITI
0035            WRITE(6,110) CPU1
0036            WRITE(6,110) CPU2
0037            CPU=CPU2-CPU1
0038            WRITE(6,120) CPU
0039       70   FORMAT('$TIME AT START OF DATA INPUT = ')
0040       80   FORMAT('$TIME AT COMPLETION OF DATA INPUT AND EXECUTION',
               1' START = ')
0041       90   FORMAT('$TIME AT END OF EXECUTION AND START OF DATA'
               1' OUTPUT = ')
0042      100   FORMAT('$TIME AT COMPLETION OF DATA OUTPUT = ')
0043       75   FORMAT('+',I2,':',I2,':',I2,':',I2)
0044      110   FORMAT(' NUMBER OF SECONDS PAST MIDNIGHT = ',F15.5)
0045      120   FORMAT(5X,'ELAPSED TIME = ',F10.5,' SECONDS')
      C
      C         FORTRAN code replaced - Begin
      C
      C          DO 1850 N=1,NLOADS
      C          IF (LSDO(N) .EQ. 0) GO TO 1850
      C          SUM=Z(N)
      C          DO 1800 I=1,LVEC
      C 1800 V1(I,N)=V1(I,N)*SUM
      C 1850 CONTINUE
      C
      C         FORTRAN code replaced - End
      C
0046            STOP
0047            END
```

```
"
"
                    $TITLE APNWX3
                    $ENTRY APNWX3,4
"
"
" THIS ROUTINE PERFORMS A SECTION OF CODE IN THE NEWX SUBROUTINE
" LOCATED IN THE EIG PROCESSOR
"
"
" AUTHOR: K. FERSON
" DATE: MARCH 1979
" Revised: L. J. Feeser and K. Matis
" Date: May 1980
"
" ----USAGE----
"
" FORTRAN: CALL APNWX3(V1BASE,ZBASE,LVEC,NLOAD)
"
"            ALL PARAMETERS ARE INTEGERS]
"
"        $PAGE
"
"
" ---MAIN DATA MEMORY MAP---
"
"  **************************  (STARTING ADDRESS)
"  *                        *
"  *     Z ARRAY            *
"  *                        *
"  **************************  NLOADS+LRM+2*(LVEC*NLOADS)
"  *                        *
"  *     V1 ARRAY           *
"  *                        *
"  **************************  NLOADS+LRM+LVEC*NLOADS
"  *                        *
"  *     V2 ARRAY           *
"  *                        *
"  **************************  NLOADS+LRM
"  *                        *
"  *     B ARRAY            *
"  *                        *
"  **************************  NLOADS
"  *                        *
"  *     LSDO ARRAY         *
"  *                        *
"  **************************  0
"
"
" ---ARRAY DESCRIPTIONS---
```

```
"
"  Z(NLOADS)......... REAL ARRAY. NEVER PASSED.
"
"  V1(LVEC*NLOADS)... REAL ARRAY. PASSED ONCE.
"                     RETURNED ONCE.
"
"  V2(LVEC*NLOADS)... REAL ARRAY.
"
"  B(LRM)............ REAL ARRAY. NEVER USED.
"
"  LSDO(NLOADS)...... INTEGER ARRAY. PREVIOUSY PASSED.
"
"
"
"
"
" S-PAD PARAMETERS
"
         V1BASE $EQU 0     "BASE OF V1 ARRAY
         ZBASE  $EQU 1     "BASE OF Z ARRAY
         LVEC   $EQU 2     "TOTAL NUMBER OF JOINTS * DEGREES OF FREEDOM
         NLOAD  $EQU 3     "TOTAL NUMBER OF LOADS
         LSDO   $EQU 4     "ADDRESS POINTER FOR LSDO ARRAY
         OUTCNT $EQU 4     "OUTER LOOP COUNTER
         CNT    $EQU 5     "INNER LOOP COUNTER
         VALUE  $EQU 6     "TEMPORARY STORAGE
         V1DEST $EQU 7     "V1 TARGET ADDRESS
         V1ADDR $EQU 10    "V1 ADDRESS POINTER
"
"
" FORTRAN:  DO 1850 N=1,NLOADS
"           IF(LSDO(N) .EQ. 0) GO TO 1850
"           SUM=Z(N)
"
"
         CLR OUTCNT
LOOP1:   MOV LSDO,LSDO; SETMA                     "GET LSDO(N)
         MOV LVEC,CNT                             "LOAD INNER COUNTER
         ADD# OUTCNT,ZBASE; SETMA                 "GET Z(N)
         LDSPI VALUE; DB=MD                       "SAVE LSDO(N)
         MOV VALUE,VALUE                          "TEST LSDO(N)=0
         BEQ CONT2
"
"
" DO LOOP CALCULATIONS
"
" FORTRAN:     DO 1800 I=1,LVEC
"         1800 V1(I,N)=V1(I,N)*SUM
"         1850 CONTINUE
"
```

```
"
CONT1:   MOV V1BASE,V1ADDR; SETMA; DPX(0)<MD          "SAVE SUM IN DPX(0)
         MOV V1BASE,V1DEST                        "LOAD ADDRESS REGISTERS
         DEC V1DEST
LOOP2:             FMUL DPX(0),MD                  "V1*SUM
         INC V1ADDR; SETMA;                       "GET NEXT V1
                   FMUL                           "PUSH
                   FMUL;                          "PUSH
                             DEC CNT              "TEST COUNTER
         INC V1DEST; SETMA; MI<FM;                 "SAVE RESULT
                             BNE LOOP2
"
"
CONT2:   INC OUTCNT                               "INC LSDO ADDRESS
         SUB# NLOAD,OUTCNT                        "TEST OUTER LOOP
         BEQ END;
                   ADD LVEC,V1BASE                 "REAJUST V1 BASE
         JMP LOOP1                                "GO TO 1300 LOOP
"
END:     $END
```

APPENDIX F

Listings of:

FOR6.FOR

APFOR6.FOR

APRED.APM

```
TYPE FOR6
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:22:05                 PAGE 001

      COM        05-14-80      RPI# 66666  66666  66666  77777  22222  77777
      C
      C    **************************************************
      C
      C     This is Program FOR6.FOR which represents a portion of
      C     Subroutine RED in Processor INV.  Used to obtain
      C     timing information for FORTRAN execution.
      C
      C     Corresponding Programs are:
      C         APFOR6.FOR - FORTRAN of FOR6.FOR with AP Calls.
      C         APRED.APM - FP120 Assembler Program replacement
      C                   of FORTRAN portion.
      C         APRED.ABJ - Object code of APRED.APM.
      C         APRED.SAV - Linked version of APRED.ABJ.
      C
      C    ******************************************************
      C
            DIMENSION BB(90,50),MAP(6),SUBMAP(250),S(6,6,50),B(6,15,50)
            DIMENSION ITIM1(2), ITIM2(2)
            INTEGER SUBMAP,CONRNG
            EQUIVALENCE (B(1,1,1),BB(1,1))
            NDPCON=90
            CONRNG=15
            ZEROD=.00005
            NDF=6
            NZERO=6
            DO 5 II=1,6
          5 MAP(II)=II
            DO 6 II=1,250,10
            SUBMAP(II)=1
            SUBMAP(II+1)=2
            SUBMAP(II+2)=3
            SUBMAP(II+3)=4
            SUBMAP(II+4)=5
            SUBMAP(II+5)=6
            SUBMAP(II+6)=7
            SUBMAP(II+7)=8
            SUBMAP(II+8)=9
            SUBMAP(II+9)=10
          6 CONTINUE
            ISTAGE=1
            NDPCON=NDF*CONRNG
            DO 10 II=1,6
            DO 10 JJ=1,6
            DO 10 KK=1,50
         10 S(II,JJ,KK)=1.
            WRITE(6,3001)
       3001 FORMAT(' INPUT THE NUMBER OF TIMES TO BE EXECUTED')
```

```
0032          READ(5,*) NTEST
      C       IF(NZERO.EQ.0) GO TO 2500
      C****  FILL B WITH UNMODIFIED DATA FROM SUBMATRIX LINE 1.
0033          CPU1=SECNDS(0.)
0034          CALL GTIM(ITIM1)
0035          DO 270 LL=1,NTEST
0036          DO 1000 K=1,NZERO
0037          M= IABS(MAP(K))
0038          DO 1000  J=1,CONRNG
0039          L= SUBMAP(J)
0040          DO 1000 I=1,NDF
0041     1000 B(I,J,K)= S(M,I,L)
      C
      C****  PRELIMINARY B MODIFICATION.
0042          DO 1400 K=1,NZERO
0043          M= MAP(K)
0044          IF(M.LT.0) GO TO 1400
0046          RA=BB(M,K)
0047          IF(RA.GT. ZEROD) GO TO 1025
      C       NEX=INEX(M)
0049          IF(RA.LT.-ZEROD) GO TO 1015
      C       KSING=KSING+1
      C       NSING=KSING
      C       WRITE(IOUT,1010) JOINT,NEX
      C1010 FORMAT(49H *** WARNING. SYSTEM K SINGULAR. JOINT/COMPONENT=I5,I2)
0051          RA=.0
0052          GO TO 1030
      C1015 KNEG=KNEG+1
      C       NNEG=KNEG
      C       IF(IPRT.LT.2) GO TO 1025
      C       WRITE(IOUT,1020) JOINT,NEX
      C1020 FORMAT(38HONEGATIVE DIAG TERM.  JOINT/COMPONENT= I5,I2)
      C1025 RA=1./RA
0053  1015     CONTINUE
0054  1025     CONTINUE
0055     1030 BB(M,K)=RA
0056          IF(K.EQ.NZERO) GO TO 1200
0058          LA=K+1
0059          DO 1100 L=LA,NZERO
0060          IA=IABS(MAP(L))
0061          RAB= RA*BB(IA,K)
0062          DO 1100 I=IA,NDFCON
0063     1100 BB(I,L)= BB(I,L) -RAB*BB(I,K)
0064     1200 IF(M.EQ.NDF) GO TO 1400
0066          INEXT=M+1
0067          DO 1300 I=INEXT,NDF
0068     1300 BB(I,K)= BB(I,K)*RA
0069     1400 CONTINUE
```

```
      C
0070        IF(CONRNG.EQ.1) GO TO 2500
0072        DO 2100 K=1,NZERO
0073        M= MAP(K)
0074        IF(M.LT.0) GO TO 2100
0076        RA= BB(M,K)
0077        NSUB= CONRNG
0078        DO 2000 I=2,CONRNG
0079        NSUB= NSUB+1
0080        LS= SUBMAP(NSUB)
      C
      C**** MODIFY EII
0081        DO 1600 ICOL=1,NDF
0082        RAB= RA*B(ICOL,I,K)
0083        DO 1600 IROW=1,ICOL
0084   1600 S(IROW,ICOL,LS)= S(IROW,ICOL,LS) -RAB*B(IROW,I,K)
      COS   CALL CALCS(CONRNG,B,SUBMAP,NDF,S)
      CDC   12 CDS OMITTED
      C
      C *** THE COMPASS ROUTINE CALCS REPLACES THE FOLLOWING ON CDC
      C
0085        DO 1700 IROW=1,NDF
0086   1700 B(IROW,I,K)=RA*B(IROW,I,K)
0087        IF(I.EQ.CONRNG)GO TO 2000
      C
      C**** MODIFY EIJ S
0089        JA=I+1
0090        DO 1900 J=JA,CONRNG
0091        NSUB=NSUB+1
0092        LS=SUBMAP(NSUB)
0093        DO 1800 ICOL=1,NDF
0094        DO 1800 IROW=1,NDF
0095   1800 S(IROW,ICOL,LS)=S(IROW,ICOL,LS)-B(IROW,I,K)*B(ICOL,J,K)
0096   1900 CONTINUE
0097   2000 CONTINUE
0098   2100 CONTINUE
0099   2500 DO 2600 L=1,CONRNG
0100        K= SUBMAP(L)
0101        DO 2600 J=1,NDF
0102        DO 2600 I=1,NDF
0103   2600 S(I,J,K)=.0
0104    270 CONTINUE
0105        CALL GTIM(ITIM2)
0106        CPU2=SECNDS(0.)
0107        WRITE(6,50) CPU1
0108        WRITE(6,50) CPU2
0109        CALL CVTTIM(ITIM1,IHR,IMI,ISE,ITI)
0110        WRITE(6,70) IHR,IMI,ISE,ITI
```

```
      CALL CVTTIM(ITIM2,IHR,IMI,ISE,ITI)
      WRITE(6,70) IHR,IMI,ISE,ITI
   50 FORMAT(' NUMBER OF SECONDS PAST MIDNIGHT = ',F15.7)
   70 FORMAT(' TIME = ',I2,':',I2,':',I2,':',I2)
      CPU=CPU2-CPU1
      WRITE(6,33)CPU
   33 FORMAT(5X,'TIME=',F16.8)
      WRITE(6,11)(BB(II,1),II=1,27)
   11 FORMAT(F16.7)
      STOP
      END
```

```
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:23:43              PAGE 001

      COM       05-14-80      RPI# 66666  66666  66666  77777  22222  77777
      C
      C    *****************************************************
      C
      C     This is Program APFOR6.FOR which contains the AP Calls
      C     as replacements for FORTRAN code in FOR6.FOR. Represents
      C     a portion of Subroutine RED in Processor INV. Obtains
      C     timing information for AP execution.
      C
      C     Corresponding Programs are:
      C         FOR6.FOR
      C         APRED.APM
      C         APRED.ABJ
      C         APRED.SAV
      C
      C    *****************************************************
      C
            DIMENSION BB(90,50),MAP(6),SUBMAP(250),S(6,6,50),B(6,15,50)
            DIMENSION ITIM1(2), ITIM2(2),ITIM3(2),ITIM4(2)
            INTEGER SUBMAP,CONRNG
            EQUIVALENCE (B(1,1,1),BB(1,1))
            NDFCON=90
            CONRNG=15
            ZEROD=.00005
            NDF=6
            NZERO=6
            DO 5 II=1,6
          5 MAP(II)=II
            DO 6 II=1,240,10
            SUBMAP(II)=1
            SUBMAP(II+1)=2
            SUBMAP(II+2)=3
            SUBMAP(II+3)=4
            SUBMAP(II+4)=5
            SUBMAP(II+5)=6
            SUBMAP(II+6)=7
            SUBMAP(II+7)=8
            SUBMAP(II+8)=9
            SUBMAP(II+9)=10
          6 CONTINUE
            ISTAGE=1
            NDFCON=NDF*CONRNG
            DO 10 II=1,6
            DO 10 JJ=1,6
            DO 10 KK=1,50
         10 S(II,JJ,KK)=1.
            WRITE(6,3001)
       3001 FORMAT(' INPUT THE NUMBER OF TIMES TO BE EXECUTED')
```

```
FORTRAN IV      V02.5-2     Tue 04-Nov-80 10:23:43                PAGE 002

0032          READ(5,*) NTEST
      C       IF(NZERO.EQ.0) GO TO 2500
      C****   FILL B WITH UNMODIFIED DATA FROM SUBMATRIX LINE 1.
0033          CPU1=SECNDS(0.)
0034          CALL GTIM(ITIM1)
0035          DO 270 L=1,NTEST
0036          CALL APCLR
0037          CALL APPUT(MAP(1),0,NZERO,1)
0038          CALL APPUT(SUBMAP(1),12,250,1)
0039          CALL APPUT(S(1,1,1),262,1800,2)
0040          CALL APWD
      D       CALL GTIM(ITIM2)
0041          CALL APRED(2062,262,12,NZERO,NDF,CONRNG,ISTAGE,NDFCON)
0042          CALL APWR
      D       CALL GTIM(ITIM3)
0043          CALL APGET(B(1,1,1),2062,4500,2)
0044          CALL APWD
0045      270 CONTINUE
0046          CALL GTIM(ITIM4)
0047          CPU2=SECNDS(0.)
0048          CALL CVTTIM(ITIM1,IHR,IMI,ISE,ITI)
0049          WRITE(6,70)
0050          WRITE(6,75) IHR,IMI,ISE,ITI
      D       CALL CVTTIM(ITIM2,IHR,IMI,ISE,ITI)
      D       WRITE(6,80)
      D       WRITE(6,75) IHR,IMI,ISE,ITI
      D       CALLCVTTIM(ITIM3,IHR,IMI,ISE,ITI)
      D       WRITE(6,90)
      D       WRITE(6,75) IHR,IMI,ISE,ITI
0051          CALL CVTTIM(ITIM4,IHR,IMI,ISE,ITI)
0052          WRITE(6,100)
0053          WRITE(6,75) IHR,IMI,ISE,ITI
0054          WRITE(6,110) CPU1
0055          WRITE(6,110) CPU2
0056          CPU=CPU2-CPU1
0057          WRITE(6,120) CPU
0058       70 FORMAT('$TIME AT START OF DATA INPUT = ')
0059       80 FORMAT('$TIME AT COMPLETION OF DATA INPUT AND EXECUTION',
             1' START = ')
0060       90 FORMAT('$TIME AT END OF EXECUTION AND START OF DATA'
             1' OUTPUT = ')
0061      100 FORMAT('$TIME AT COMPLETION OF DATA OUTPUT = ')
0062       75 FORMAT('+',I2,':',I2,':',I2,':',I2)
0063      110 FORMAT(' NUMBER OF SECONDS PAST MIDNIGHT = ',F15.5)
0064      120 FORMAT(5X,'ELAPSED TIME = ',F10.5,' SECONDS')
      C
      C       FORTRAN code replaced - Begin
      C
```

```
C       DO 1000 K=1,NZERO
C       M= IABS(MAP(K))
C       DO 1000  J=1,CONRNG
C       L= SUBMAP(J)
C       DO 1000 I=1,NDF
C 1000 B(I,J,K)= S(M,I,L)
CC
CC**** PRELIMINARY B MODIFICATION.
C       DO 1400 K=1,NZERO
C       M= MAP(K)
C       IF(M.LT.0) GO TO 1400
C       RA=BB(M,K)
C       IF(RA.GT. ZEROD) GO TO 1025
CC      NEX=INEX(M)
C       IF(RA.LT.-ZEROD) GO TO 1015
CC      KSING=KSING+1
CC      NSING=KSING
CC      WRITE(IOUT,1010) JOINT,NEX
CC1010 FORMAT(49H *** WARNING. SYSTEM K SINGULAR. JOINT/COMPONENT=I5,I2)
C       RA=.0
C       GO TO 1030
CC1015 KNEG=KNEG+1
CC      NNEG=KNEG
CC      IF(IPRT.LT.2) GO TO 1025
CC      WRITE(IOUT,1020) JOINT,NEX
CC1020 FORMAT(38HONEGATIVE DIAG TERM.  JOINT/COMPONENT= I5,I2)
CC1025 RA=1./RA
C1015   CONTINUE
C1025   CONTINUE
C 1030 BB(M,K)=RA
C       IF(K.EQ.NZERO) GO TO 1200
C       LA=K+1
C       DO 1100 L=LA,NZERO
C       IA=IABS(MAP(L))
C       RAB= RA*BB(IA,K)
C       DO 1100 I=IA,NDFCON
C 1100 BB(I,L)= BB(I,L) -RAB*BB(I,K)
C 1200 IF(M.EQ.NDF) GO TO 1400
C       INEXT=M+1
C       DO 1300 I=INEXT,NDF
C 1300 BB(I,K)= BB(I,K)*RA
C 1400 CONTINUE
CC
C       IF(CONRNG.EQ.1) GO TO 2500
C       DO 2100 K=1,NZERO
C       M= MAP(K)
C       IF(M.LT.0) GO TO 2100
C       RA= BB(M,K)
```

```
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:23:43              PAGE 004

      C       NSUB= CONRNG
      C       DO 2000 I=2,CONRNG
      C       NSUB= NSUB+1
      C       LS= SUBMAP(NSUB)
      CC
      CC**** MODIFY EII
      C       DO 1600 ICOL=1,NDF
      C       RAB= RA*B(ICOL,I,K)
      C       DO 1600 IROW=1,ICOL
      C 1600 S(IROW,ICOL,LS)= S(IROW,ICOL,LS) -RAB*B(IROW,I,K)
      CCOS    CALL CALCS(CONRNG,B,SUBMAP,NDF,S)
      CCDC    12 CDS OMITTED
      CC
      CC *** THE COMPASS ROUTINE CALCS REPLACES THE FOLLOWING ON CDC
      CC
      C       DO 1700 IROW=1,NDF
      C 1700 B(IROW,I,K)=RA*B(IROW,I,K)
      C       IF(I.EQ.CONRNG)GO TO 2000
      CC
      CC**** MODIFY EIJ S
      C       JA=I+1
      C       DO 1900 J=JA,CONRNG
      C       NSUB=NSUB+1
      C       LS=SUBMAP(NSUB)
      C       DO 1800 ICOL=1,NDF
      C       DO 1800 IROW=1,NDF
      C 1800 S(IROW,ICOL,LS)=S(IROW,ICOL,LS)-B(IROW,I,K)*B(ICOL,J,K)
      C 1900 CONTINUE
      C 2000 CONTINUE
      C 2100 CONTINUE
      C 2500 DO 2600 L=1,CONRNG
      C       K= SUBMAP(L)
      C       DO 2600 J=1,NDF
      C       DO 2600 I=1,NDF
      C 2600 S(I,J,K)=.0
      C
      C      FORTRAN code replaced - End
      C
0065         WRITE(6,200)(BB(II,1),II=1,27)
0066  200    FORMAT(F16.7)
0067         STOP
0068         END
```

```
                    $TITLE APRED
                 $ENTRY APRED,8
                    $EXT DIV
"
" THIS ROUTINE PERFORMS THE "RED" SUBROUTINE LOCATED IN THE INV
" MATRIX INVERSION PROCESSOR
"
" AUTHOR: K. FERSON
" DATE: DECEMBER 1978
" Revised: L. J. Feeser and K. Matis
" Date: May 1980
"
" ----USAGE----
"
" FORTRAN: CALL APRED(BBASE,SBASE,MPBASE,NZERO,NDF,CONRNG,ISTAGE,NDFCON)
"
"          ALL PARAMETERS MUST BE INTEGER]
"
" ----DATA PAD STORAGE---
"
"  DPX(2) ... CONTAINS NDF
"  DPX(3) ... CONTAINS NDF*(K-1)*CONRNG
"  DPY(1) ... CONTAINS NDF*CONRNG
"  DPY(2) ... CONTAINS RA
"  DPY(3) ... CONTAINS NDF*NDF
"
"
"
" ---MAIN DATA MEMORY MAP---
"
"  *****************************  (STARTING ADDRESS)
"  *                           *
"  *                           *
"  *      B OR BB ARRAY        *
"  *                           *
"  *****************************  12+LR4+ISIZE*NDF*NDF+LRK
"  *                           *
"  *                           *
"  *      AK ARRAY             *
"  *                           *
"  *****************************  12+LR4++ISIZE*NDF*NDF
"  *                           *
"  *                           *
"  *      S ARRAY              *
"  *                           *
"  *****************************  12+LR4
"  *                           *
"  *                           *
"  *      SUBMAP ARRAY         *
"  *                           *
```

```
"  *****************************  12
"  *                           *
"  *                           *
"  *      INEX ARRAY           *
"  *                           *
"  *****************************  6
"  *                           *
"  *                           *
"  *      MAP ARRAY            *
"  *                           *
"  *****************************  0
"
"
" ---ARRAY EXPLANATIONS---
"
" MAP(6) ............. INTEGER ARRAY. TRANSFERED ONCE FOR EACH CALL
"                      TO THE 'APRED' ROUTINE.
"
" INEX(6) ........... INTEGER ARRAY. TRANSFERED ONCE FOR
"                      EACH CALL TO THE 'APRED' ROUTINE.
"
" SUBMAP(LR4)......... INTERGER ARRAY. TRANSFERED ONLY AFTER
"                      BEING READ FROM THE DATA BASE.
"
" S(ISIZE*NDF*NDF) ... REAL ARRAY. NEVER TRANSMITTED.
"
" AK(LRK)............. REAL ARRAY. TRANSFERED ONLY AFTER BEING
"                       READ FROM THE DATA BASE.
"
" B(NZERO*NDFCON)...... REAL ARRAY. RETURNED ONCE FOR EVERY
"                       CALL TO THE ROUTINE.
"
"
" S-PAD PARAMETERS
"
"
                  BBASE  $EQU 0     "BASE ADDRESS OF BB ARRAY
                  SBASE  $EQU 1     "BASE ADDRESS OF S ARRAY
                  MPBASE $EQU 2     "BASE ADDRESS  FOR SUBMAP ARRAY
                  NZERO  $EQU 3     "NUMBER OF NONZERO SUBMATRICES
                  NDF    $EQU 4     "NUMBER OF DEGREES OF FREEDOM PER JOINT
                  CONRNG $EQU 5     "NUMBER OF NONZERO SUBMATRICES IN THIS ROW
                  NSUB   $EQU 6     "NUMBER OF SUBMATICES
                  SUBMAP $EQU 6     "ADDRESS POINTER FOR SUBMAP ARRAY
                  ISTAGE $EQU 6     "CURRENT JOINT NUMBER
                  NDFCON $EQU 7     "NDF*CONRNG
                  ICNT   $EQU 7     "LOOP COUNTER
                  K      $EQU 10    "LOOP COUNTER
                  N27    $EQU 11    "CONSTANT=27.
                  BBADDR $EQU 11    "ADDRESS POINTER OF BB ARRAY
```

```
                RADDR  $EQU 12     "ADDRESS POINTER FOR RA=BB(M,K)
                NDFS   $EQU 12     "SCRATCH REGISTER
                J      $EQU 12     "LOOP COUNTER
                BBADR1 $EQU 12     "ADDRESS POINTER FOR BB ARRAY
                BADDR  $EQU 12     "ADDRESS POINTER FOR BB ARAY
                BBADRF $EQU 12     "TARGET ADDRESS FOR BB ARRAY
                SADDR  $EQU 13     "ADDRESS POINTER FOR S ARRAY
                BBADR2 $EQU 13     "ADDRESS POINTER FOR BB ARRAY
                B1ADDR $EQU 14     "ADDRESS POINTER FOR BB ARRAY
                LA     $EQU 14     "LOOP COUNTER
                B2ADDR $EQU 15     "ADDRESS POINTER OF BB ARRAY
                ZERO   $EQU 15     "CONSTANT=0.
                II     $EQU 15     "LOOP COUNTER
                JCNT   $EQU 15     "LOOP COUNTER
                ICOL   $EQU 16     "LOOP COUNTER
                N      $EQU 16     "LOOP COUNTER
                CNT    $EQU 16     "LOOP COUNTER
                IA     $EQU 16     "LOOP COUNTER
                IROW   $EQU 17     "LOOP COUNTER
                COUNT  $EQU 17     "LOOP COUNTER
                M      $EQU 17     "INTEGER=MAP(K)
                INEXT  $EQU 17     "COUNTER
                I      $EQU 17     "LOOP COUNTER
"
" FIND NDF*NDF AND STORE THE RESULT IN DPY(3)
"
" FIND NDF*CONRNG AND STORE THE RESULT IN DPY(1)
"
" THESE DATA PADS MAY NOT BE USED FOR OTHER PURPOSES
"
"
 APRED: LDSPI N27; DB=27.
                MOV CONRNG,CONRNG; DPX(1)<SPFN            "FLOAT CONRNG
          MOV NDF,NDF; DPX(0)<SPFN                         "FLOAT NDF
          FADD ZERO,MDPX(0); MOV N27,N27
                FADD ZERO,MDPX(1); MOV N27,N27
          DPX(2)<FA;                                       "STORE NDF
                FADD
                FMUL DPX(2),FA                             "NDF*CONRNG
                     FMUL DPX(2),DPX(2)                   "NDF*NDF
                     FMUL
                DPY(1)<FM;                                 "NDF*CONRNG=DPY(1)
                     FMUL
                     DPY(3)<FM                            "SAVE NDF*NDF=DPY(3)
"
"
" PERFORM THE 1000 LOOP CALCULATIONS
"
" FORTRAN: DO 1000 K=1,NZERO
"          M=IABS(MAP(K))
```

```
"          DO 1000 J=1,CONRNG
"          L=SUBMAP(J)
"          DO 1000 I=1,NDF
"     1000 B(I,J,K)=S(M,I,L)
"
         MOV BBASE,BADDR                        "LOAD ADDRESS OF B(1,1,1)
         DEC BADDR                             "SET UP FOR LOOPS
         CLR K
LOOP1:    MOV K,K; SETMA                          "GET MAP(K)
         MOV CONRNG,JCNT                        "LOAD J LOOP COUNT
         MOV MPBASE,SUBMAP                      "LOAD SUBMAP BASE
         DEC SUBMAP                             "GET BASE-1
         LDSPI M; DB=MD                         "SAVE M=MAP(K)
         DEC M                                 "GET M-1
"
"
" PERFORM THE J LOOP BY FINDING THE
" ADDRESS OF S(M,I,L)
"
LOOP2:    INC SUBMAP; SETMA                      "GET SUBMAP(J)
         RPSF ONE; DPY(0)<DB                    "DPY(0)=1.
         LDSPI N27; DB=27.                      "LOAD CONSTANT=27.
         DPX(0)<MD                             "SAVE L=SUBMAP(J)
         FSUBR DPY(0),MDPX(0); MOV N27,N27        "FLOAT(L-1)
         FADD                                  "PUSH
         FMUL DPY(3),FA                         "(L-1)*NDF*NDF
         FMUL                                  "PUSH
         FMUL                                  "PUSH
         DPX(0)<FM                              "SAVE RESULT
         FIX DPX(0)                             "INT((L-1)*NDF*NDF)
         FADD;                                 "PUSH
                   MOV NDF,CNT                   "LOAD INNER COUNT
         DPX(0)<FA                             "SAVE RESULT
         LDSPI SADDR; DB=DPX(0)                  "LOAD ADDRESS POINTER
         ADD SBASE,SADDR                        "ADD BASE TO POINTER
         ADD M,SADDR                            "ADD (M-1) TO POINTER
         SUB NDF,SADDR                          "LOOP SET-UP
"
"
"
" PERFORM THE INNER I LOOP
"
"
LOOP3:   NOP
         ADD NDF,SADDR; SETMA                    "GET S(M,I,L)
         NOP
                   DEC CNT                      "TEST LOOP
END3:    INC BADDR; SETMA; MI<MD;                "STORE RESULT
                   BNE LOOP3
"
```

```
"
        DEC JCNT                              "TEST J LOOP
        BEQ CONT1                             "IF DONE, CONTINUE
        JMP LOOP2                             "IF NOT, BRANCH BACK
"
CONT1:  INC K                                 "INCREMENT COUNT
        SUB# NZERO,K                          "TEST K LOOP
        BEQ CONT2                             "IF DONE, CONTINUE
        JMP LOOP1                             "IF NOT, BRANCH BACK
"
"
" BEGIN THE 1400 LOOP
"
" FORTRAN: DO 1400 K=1,NZERO
"          M=MAP(K)
"          IF (M .LT. 0) GO TO 1400
"          RA=BB(M,K)
"          IF(RA .GT. ZEROD) GO TO 1025
"          NEX=INEX(M)
"          IF (RA .LT. -ZEROD) GO TO 1015
"
"
CONT2:  MOV BBASE,BBADDR                      "LOAD BB BASE ADDRESS
        CLR K
LOOP4:  MOV K,K; SETMA                        "GET M=MAP(K)
        INC K
        NOP
        LDSPI M; DB=MD                        "SAVE M
        MOV M,M                               "TEST M=0.
        BNE CONT3;                            "IF=0, CONTINUE
                  DEC M                       "GET M-1
        JMP CONT13                            "IF NOT, GO TO 1400
CONT3:  ADD M,BBADDR; SETMA                   "GET BB(M,K)
        RPSF ZEROD; DPY(0)<DB                 "DPY(0)=ZEROD
        NOP
        DPX(0)<MD                             "RA=DPX(0)
        FSUB DPX(0),DPY(0)                    "RA-ZEROD
        FADD DPX(0),DPY(0)                    "RA+ZEROD
        FADD                                  "PUSH
        BFGT CONT6                            "IF RA>ZEROD, GOTO 1025
        BFGE CONT4                            "IF RA>-ZEROD, CONTINUE
        JMP CONT5                             "RA<-ZEROD
"
"
" -ZEROD<RA<ZEROD
"
" FOR THIS CASE THE SYSTEM IS SINGULAR
"
" FORTRAN: KSING=KSING+1
"          NSING=KSING
```

```
"          WRITE(IOUT,1010) JOINT,NEX
"          FORMAT(49H ***WARNING. SYSTEM K SINGULAR. JOINT/COMPONENT=I5,I2
"          RA=0.
"          GO TO 1030
"
"
"
CONT4:   DPX(0)<DB; DB=ZERO                         "RA=0.
         JMP CONT7                                  "GO TO 1030
"
"
"
" RA<-ZEROD
"
" FOR THIS CASE THE DIAGONAL IS NEGATIVE
"
" FORTRAN: KNEG=KNEG+1
"          NNEG=KNEG
"          IF(IPRT .LT. 2) GO TO 1025
"          WRITE(IOUT,1020) JOINT,NEX
"    1020  FORMAT(38HO NONNEGATIVE DIAG TERM. JOINT/COMP=I5,I2)
"
"
CONT5:  NOP
"
"
" RA>ZEROD
"
" FOR THIS CASE THE DIAGONAL IS O.K.
"
" FORTRAN: RA=1.0/RA
"          BB(M,K)=RA
"          IF(K .EQ. NZERO) GO TO 1200
"
CONT6:   RPSF ONE; DPY(0)<DB                        "DPY(0)=1.0
        NOP
         JSR DIV                                    "DO DPY(0)/DPX(0) OR 1.0/RA
"
" DIV USES S-PADS 13,14, AND 15
" THE ANSWER IS RETURNED IN DPX(0)
"
CONT7:   MOV BBADDR,BBADDR; SETMA; MI<DPX(0)         "BB(M,K)=RA
         SUB# K,NZERO                               "IS K=0 ??
         BNE CONT8                                  "IF NOT, CONTINUE
         JMP CONT10                                 "IF YES, BRANCH TO 1200
"
"
"
" REAJUST BB(M,K) ADDRESS TO BB(1,K)
" BY SUBTRACTING M
```

```
"
" PERFORM THE 1100 LOOP CALCULATIONS
"
" FORTRAN: LA=K+1
"           DO 1100 L=LA,NZERO
"           IA=IABS(MAP(L))
"           RAB=RA*BB(IA,K)
"           DO 1100 I=IA,NDFCON
"    1100   BB(I,L)=BB(I,L)-RAB*BB(I,K)
"
"
"
"
"
"
"
"
CONT8:  SUB M,BBADDR                          "REAJUST BB(M,K) ADDRESS
        MOV BBADDR,BBADR1                     "LOAD BB(I,L)=BB(1,K)
        ADD NDFCON,BBADR1                     "GET BB(1,L) ADDRESS
        MOV K,LA                              "LA=K
LOOP5:  MOV LA,LA; SETMA                      "GET MAP(L)
        NOP
        CLR ZERO
        LDSPI IA; DB=MD                       "IA=MAP(L)
        MOV IA,IA                             "TEST FOR NEGATIVE IA
        BGE CONT9                             "IF >0, CONTINUE
        SUB IA,ZERO                           "OTHERWISE IA=-IA
        MOV ZERO,IA
"
"
CONT9:  DEC IA                                "IA=IA-1
        ADD IA,BBADDR; SETMA                  "GET BB(IA,K)
        NOP
        MOV IA,II                             "LOAD INNER COUNT
        FMUL DPX(0),MD;                       "RA*BB(IA,K)
                MOV BBADDR,BBADDR; SETMA      "GET BB(I,K)
        FMUL                                  "PUSH
        FMUL;                                 "PUSH
                ADD IA,BBADR1; SETMA          "GET BB(I,L)
        DPX(1)<FM;                            "SAVE DPX(1)=RAB
                FMUL FM,MD;                   "RAB*BB(I,K)
                MOV BBADR1,BBADR2             "LOAD BB(I,L) TARGET ADDRESS
                FMUL;                         "PUSH
                DEC BBADR2                    "LOOP SET-UP
"
"
"
LOOP6:          FMUL                          "PUSH
        INC BBADDR; SETMA;                    "GET NEXT BB(I,K)
```

```
                  FSUBR FM,MD                       "BB(I,L)-RAB*BB(I,K)
                  FADD;                             "PUSH
                              INC II                "INC COUNTER
         INC BBADR1; SETMA                          "GET NEXT BB(I,L)
         FMUL DPX(1),MD;                            "RAB*BB(I,K)
                              SUB# NDFCON,II        "TEST LOOP
         INC BBADR2; SETMA; MI<FA;                  "STORE RESULT
         FMUL;                                      "PUSH
                              BNE LOOP6
"
"
"
"
         SUB NDFCON,BBADDR                          "REAJUST BB(I,K) ADDRESS
         INC LA                                     "INC LA
         SUB# LA,NZERO                              "TEST OUTER LOOP
         BEQ CONT10                                 "IF DONE, CONTINUE
         JMP LOOP5                                  "IF NOT, BRANCH BACK
"
"
"
" FORTRAN: 1200 IF(M .EQ. NDF) GOTO 1400
"                INEXT=M+1
"                DO 1300 I=INEXT,NDF
"           1300 BB(I,K)=BB(I,K)*RA
"
"
CONT10:  INC M
         SUB# M,NDF                                 "TEST M=NDF
         BNE CONT11                                 "IF NOT EQUAL, CONTINUE
         JMP CONT12                                 "OTHERWISE, BRANCH
"
"
"
CONT11:  ADD INEXT,BBADDR                           "INEXT=M+1 ,GET ADDRESS
         MOV BBADDR,BBADRF; SETMA                   "GET BB(INEXT,K)
         DEC BBADRF                                 "LOOP SET-UP
         MOV INEXT,CNT                              "LOAD COUNTER
"
"
LOOP7:            FMUL DPX(0),MD;                   "RA*BB(I,K)
                              INC CNT               "INC COUNTER
         INC BBADDR; SETMA;                         "GET NEXT BB(I,K)
                  FMUL                              "PUSH
                  FMUL;                             "PUSH
                              SUB# CNT,NDF          "TEST LOOP
         INC BBADRF; SETMA; MI<FM;                  "STORE RESULT
                              BNE LOOP7
"
"
```

```
" TEST THE 1400 LOOP
"
" FORTRAN: 1400 CONTINUE
"
CONT12:  SUB NDF,BBADDR                               "REAJUST BB ADDRESS
         ADD NDFCON,BBADDR                            "GET BB(1,K+1) ADDRESS
         SUB# NZERO,K                                 "K=NZERO ?
         BEQ CONT13                                   "IF SO, CONTINUE
         JMP LOOP4                                    "IF NOT, BRANCH BACK
"
"
" PERFORM THE 2100 LOOP CALCUALATIONS
"
"
" FORTRAN: DO 2100 K=1,NZERO
"           M=MAP(K)
"           IF(M .LT. 0) GO TO 2100
"           RA=BB(M,K)
"           NSUB=CONRNG
"
"
CONT13:  LDSPI N27; DB=27.
         CLR K                                        "CLEAR OUTER COUNT
"
" CALCUALATE THE ADDRESS OF BB(M,K) BY FINDING
" M+(K-1)*NDFCON
"
LOOP8:   MOV K,K; SETMA;                                     "GET M=MAP(K)
                   DPX(1)<SPFN                               "STORE (K-1)
                   FADD ZERO,MDPX(1);MOV N27,N27             "FLOAT K-1
                   FADD
         DPX(0)<MD;                                          "FLOAT M
"
" DPY(1) HAS THE VALUE NDF*CONRNG FROM BEFORE
"
                   FMUL DPY(1),FA                            "K-1*NDF*CONRNG
         FADD ZERO,MDPX(0); MOV N27,N27;
                   FMUL
         FADD;
                   FMUL
         DPY(0)<FA;
                   FADD FM,FA                                "M+CON
         BFGE CONT14;                                        "TES
                   FADD;DPX(3)<FM                            "STORE             ONRNG
         JMP CONT17                                   "IF <0, BRANCH
CONT14:   DPX(0)<FA                                          "STORE M+CON*NDF*K-1
         FIX DPX(0)
         FADD;
                   MOV CONRNG,NSUB                           "LOAD NSUB
         DPX(0)<FA
```

```
        LDSPI RADDR;DB=DPX(0)                                "STORE NDF*CONRNG*(K-1)+(M-1
        DEC RADDR                                          "LOOP SET-UP
        ADD BBASE,RADDR; SETMA                                 "GET BB(M,K)=RA
"
" SET UP FOR THE 2000 LOOP
"
" FORTRAN: DO 2000 I=2,CONRNG
"          NSUB=NSUB+1
"          LS=SUBMAP(NSUB)
"
                  CLR ICNT                                           "RESET ICNT
                  INC ICNT
        DPY(2)<MD                                                "STORE RA IN DPY(2)
LOOP9:            MOV ICNT,ICNT; DPX(0)<SPFN                          "FLOAT ICNT-1
                  FADD ZERO,MDPX(0); MOV N27,N27
        ADD# MPBASE,NSUB; SETMA                                    "GET LS=SUBMAP(NSUB)
        DPY(-1)<DB; DB=40000; WRTMAN;                       "SET DPY(-1)=1.0
                            INC NSUB
        DPY(-1)<DB; DB=1015; WRTEX                           "SET DPY(-1)=1.0
        DPX(-1)<MD;
                            CLR ICOL                         "CLR COUNTER
"
" START ADDRESS CALCULATIONS FOR THE LOOPS
" TO FOLLOW. FIND (LS-1)*NDF*NDF FOR THE S ARRAY
" ADDRESS OFFSET. FIND (K-1)*CONRNG*NDF*NDF*(I-1)
" FOR THE ADDRESS OFFSET OF THE B ARRAY.
"
        FSUBR DPY(-1),MDPX(-1); MOV N27,N27                           "LS -1
        FADD;
                  FMUL DPX(2),FA;                                 "(I-1)*NDF
                            CLR IROW
        FMUL DPY(3),FA;                                        "(LS-1)*NDF*NDF
                            CLR NDFS
        FMUL
        FMUL;
                  FADD FM,DPX(3)                              "(I-1)*NDF+NDF*CONRNG*(K-1)
        DPX(-1)<FM;                                             "STORE LS-1 *NDF*NDF
                  FADD
        FIX DPX(-1);                                      "GET INTEGER OF (LS-1)*NDF*NDF
                  DPY(-3)<FA                                     "STORE I-1*NDF+NDF*CONRN
                  FIX DPY(-3)
        DPX(-1)<FA;                                       "STORE RESULT
                  FADD
        LDSPI SADDR; DB=DPX(-1);                               "STORE OFFSET ADDRESS FOR S
                  DPY(-3)<FA                                     "STORE RESULT
                  LDSPI B1ADDR; DB=DPY(-3)                    "STORE OFFSET FOR B ARRAY
"
" SET UP ADDRESSES FOR THE 1600 LOOPS
" AND PERFORM THE 1600 LOOP CALCULATIONS
"
```

```
" FORTRAN: DO 1600 ICOL=1,NDF
"          RAB=RA*B(ICOL,I,K)
"          DO 1600 IROW=1,ICOL
"     1600 S(IROW,ICOL,LS)=S(IROW,ICOL,LS)-RAB*B(IROW,I,K)
"
" CALCULATE THE ADDRESS OFFSETS FOR THE S AND B ARRAYS
"
"
         ADD SBASE,SADDR                              "ADD BASE AND OFFSET FOR S
         ADD BBASE,B1ADDR                            "ADD BASE AND OFFSET FOR B1
         MOV B1ADDR,B2ADDR                              "LOAD B2 ADDRESS
         DEC B1ADDR
LOOP10:    INC B1ADDR; SETMA                              "GET B1(ICOL,I,K)
                           INC ICOL                          "INC LOOP COUNT
                           SUB IROW,NDFS                       "GET NDFS-IROW
         FMUL DPY(2),MD;                                 "MPY RA*B1(ICOL,I,K)
                           ADD NDFS,SADDR                      "CHANGE BASE FOR S
         FMUL;
                           MOV B2ADDR,B2ADDR; SETMA      "GET B2(IROW,I,K)
         FMUL;
                           MOV NDF,NDFS                           "LOAD NDFS
         DPY(0)<FM;                                                "STORE RAB
                  MOV SADDR,SADDR; SETMA                 "GET S(IROW,ICOL,LS)
                  FMUL DPY(0),MD                              "MPY RAB*B2
                  FMUL;
                           CLR IROW                                "RESET IROW
"
" PERFORM THE INNER 1600 LOOP
"
LOOP11:            FMUL;
                           INC IROW                                 "INC INNER LOOP
         INC B2ADDR; SETMA;
                  FSUBR FM,MD                            "S-RAB*B2
                  FADD
         INC SADDR; SETMA;
                  DPX(0)<FA                            "SAVE RESULT IN DATA PAD
         FMUL DPY(0),MD;                           "MPY RAB*B2(+1)
                           SUB# IROW,ICOL           "TEST INNER LOOP
END11:     FMUL;
                  DECMA; MI<DPX(0);                  "STORE RESULTS IN S
                           BNE LOOP11
        SUB# ICOL,NDF                  "TEST OUTER LOOP
END10:  SUB ICOL,B2ADDR; BEQ GOT    "RESET B2 ADDRESS
        JMP LOOP10
"
"
" GET READY FOR THE 1700 LOOP
" B2ADDR HAS THE CORRECT ADDRESS FOR B(IROW,I,K)
"
" FORTRAN: DO 1700 IROW=1,NDF
```

```
"     1700 B(IROW,I,K)=RA*B(IROW,I,K)
"
GOT:    MOV B2ADDR,B1ADDR; SETMA                              "GET FIRST B(IROW,I,K)
" IROW ALREADY CONTAINS NDF
" IROW=COUNT (SAME SPAD)
"
                                 MOV ICNT,J                    . "ICNT-1 TO J
                                 INC J                        "J HAS VALUE OF I
"
LOOP12:   FMUL DPY(2),MD                                      "MPY RA*B(IROW,I,K)
          FMUL;
                                 INC B1ADDR; SETMA     "GET NEXT B1 ELEMENT
          FMUL;
                    DEC COUNT
END12:    MI<FM; DECMA;
                    BNE LOOP12
" STORE RESULT IN B(IROW,I,K)
" GET READY FOR THE 1900 LOOP
" BASE OF B1ADDR IS EQUAL TO THE BASE OF B2(ICOL,J,K)
" IN THE 1900 LOOP, ALSO B2(ICOL,J,K)-NDF WILL GIVE THE
" BASE OF B1(ICOL,I,K)
"
" FORTRAN: IF(I .EQ. CONRNG) GO TO 2000
"          JA=I+1     (ALREADY DONE)
"          DO 1900 J=JA,CONRNG
"          NSUB=NSUB+1
"          LS=SUBMAP(NSUB)
"
          MOV B1ADDR,B2ADDR                                "LOAD B2 ADDRESS
          SUB NDF,B1ADDR                                  "REAJUST B1 ADDRESS
          SUB# CONRNG,J                                  "TEST IF I=CONRNG
          BNE LOOP13;                                 "IF I IS NOT EQUAL, CONTINUE
                    DEC B1ADDR
          JMP CONT16                                 "IF I=CONRNG, GO TO  2000
"
"
LOOP13:   ADD# MPBASE,NSUB; SETMA                             "GET LS=SUBMAP(NSUB)
          DPY(-1)<DB; DB=40000; WRTMAN;                 "DPY(-1)=1.0
                                 INC NSUB
          DPY(-1)<DB; DB=1015; WRTEX                    "DPY(1)=1.0
          DPX(-1)<MD
          FSUBR DPY(-1),MDPX(-1); MOV N27,N27                  "LS-1
          FADD
"
" CALCUALTE THE OFFSET WITH THE NEW LS VALUE
"
          FMUL DPY(3),FA                               "MPY (LS-1)*NDF*NDF
          FMUL
          FMUL
          DPX(-1)<FM;                                   "STORE THE RESULT
```

```
                  MOV NDF,N                          "LOAD INNER COUNTER
         FIX DPX(-1);
                  MOV NDF,M                           "LOAD  OUTER COUNTER
"
" GET READY FOR THE 1800 LOOP CALCUALTIONS
"
" FORTRAN: DO 1800 ICOL=1,NDF
"          DO 1800 IROW=1,NDF
"    1800 S(IROW,ICOL,LS)=S(IROW,ICOL,LS)-B(IROW,I,K)*B(ICOL,J,K)
"
" PERFORM THE ADDRESS SET UP FOR THE LOOPS
"
         FADD
         DPX(-1)<FA;
                  MOV B2ADDR,B2ADDR; SETMA
         LDSPI SADDR; DB=DPX(-1)                       "STORE THE S ADRESS
         ADD SBASE,SADDR                                  "GET ADDRESS OF S
         DEC SADDR                                   "DEC FOR LOOP
"
" PERFORM THE OUTER 1800 LOOP CALCULATION
" SINCE B(ICOL,J,K) DOES NOT CHANGE IN THE INNER
" LOOP. IT IS ACCESSED ONLY IN THE OUTER LOOP
"
LOOP14:   DPY(0)<MD                                  "STORE B2
         INC B1ADDR; SETMA                           "GET B1
                  NOP                                "TIMING NOP
         INC SADDR; SETMA                            "GET S ARRAY ELEMENT
         FMUL DPY(0),MD                               "DO B1*B2
         FMUL;
                  DEC N                              "DEC INNER COUNT
"
" PERFORM THE INNER MOST 1800 LOOP CLALCULATIONS
"
LOOP15:   FMUL
         FSUBR FM,MD;                                "DO S-B1*B2
                  INC B1ADDR; SETMA                   "GET NEXT B1 ELEMENT
         FADD
         DPX(-1)<FA;                                 "STORE S-B1*B2
                           INC SADDR; SETMA            "GET NEXT S ELEMEMT
                  DEC N;                             "TEST INNER LOOP
                           FMUL DPY(0),MD             "DO (B1+1)*B2
END15:    MI<DPX(-1); DECMA;                         "STORE S=S-B1*B2
                  BNE LOOP15;
                           FMUL
                  MOV NDF,N;                          "RESET INNER LOOP COUNT
                           FMUL
                  SUB N,B1ADDR;                     "RESET B1 ADDRESS
                           FSUBR FM,MD                 "(S+1)-(B1+1)*B2
         INC B2ADDR; SETMA;                          "GET NEXT B2 ELEMENT
                           FADD
```

```
                    DEC M;                                "TEST OUTER 1800 LOOP
                         DPX(-1)<FA                         "STORE (S+1)-(B1+1)*B2
END14:               BEQ GO;                            "SAVE RESULT
                         MI<DPX(-1); MOV SADDR,SADDR; SETMA
        JMP LOOP14
"
"
" TEST THE OUTER LOOPS FOR COMPLETION
"
" FORTRAN: 1900 CONTINUE
"          2000 CONTINUE
"          2100 CONTINUE
"
GO:                 INC J                                 "TEST 1900 LOOP
                    SUB# J,CONRNG                          "TEST IF J=CONRNG
                    BEQ CONT16                             "IF YES, CONTINUE
                    JMP LOOP13                            "IF NOT, BRANCH BACK
CONT16:              INC ICNT                             "TEST 2000 LOOP
                    SUB# ICNT,CONRNG                       "IS I=CONRNG?
                    BEQ CONT17                             "IF YES,CONTINUE
                    JMP LOOP9                             "IF NOT, BRANCH BACK
CONT17:              INC K                                "TEST OUTER 2100 LOOP
                    SUB# K,NZERO                           "IS K=NZERO
                    BEQ CONT18                             "IF YES, CONTINUE
                    JMP LOOP8                             "IF NOT, BRANCH BACK
"
" GET READY FOR THE 2600 LOOP CALCULATIONS
"
" FORTRAN: 2500 DO 2600 L=1,CONRNG
"               K=SUBMAP(L)
"               DO 2600 J=1,NDF
"               DO 2600 I=1,NDF
"          2600 S(I,J,K)=0.
"               RETURN
"
" FIND THE ADDRESS OFFSET OF S(1,1,K) BY FINDING
" (K-1)*NDF*NDF
"
CONT18:    MOV CONRNG,CNT                                 "LOAD COUNTER
          DEC MPBASE                                      "LOOP SET-UP
"
" DPY(3) HAS THE VALUE NDF*NDF
"
OUTLOP:   INC MPBASE; SETMA;                                        "GETK
                    FIX DPY(3)                             "INT(NDF*NDF)
                    FADD
                    DPX(1)<FA
          DPX(0)<MD                                      "STORE K
          FSUBR DPY(-1),MDPX(0); MOV N27,N27                        "K-1
          FADD;
```

```
                 LDSPI COUNT; DB=DPX(1)                     "SET COUNT=NDF*NDF
        FMUL DPY(3),FA;                                     "K-1*NDF*NDF
                 DEC COUNT
        FMUL
        FMUL
        DPX(0)<FM                                    "STORE K-1*NDF*NDF
        FIX DPX(0)                                    "FIX RESULT
        FADD
        DPX(0)<FA
        LDSPI SADDR; DB=DPX(0)                           "LOAD S ADDRESS OFFSET
        ADD SBASE,SADDR; SETMA; MI<ZERO                "CLR FIRST ELEMENT
INLOP:  DEC COUNT                                    "CHECK LOOP
        INC SADDR;SETMA; MI<ZERO; BNE INLOP            "CLR S(I,J,K)
                 DEC CNT                              "CHECK OUTER LOOP
                 BEQ CONT19                            "IF DONE, GO TO END
                 JMP OUTLOP                           "IF NOT, BRANCH BACK
"
ONE:    $FP 1.0
ZEROD:  $FP .00001
CONT19: RETURN
         $END
```

APPENDIX G

Listings of:

FOR8.FOR

APFOR8.FOR

APAFEX.APM

```
 TYPE FOR8
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:24:09              PAGE 001

      COM       05-14-80      RPI# 66666  66666  66666  77777  00000  33333
      C
      C     ****************************************************
      C
      C      This is Program FOR8.FOR which represents a portion of
      C      Subroutine AFEX in Processor INV. Used to obtain
      C      timing information for FORTRAN execution.
      C
      C      Corresponding Programs are:
      C          APFOR8.FOR - FORTRAN of FOR8.FOR with AP Calls.
      C          APAFEX.APM - FP120 Assembler Program replacement
      C                  of FORTRAN portion.
      C          APAFEX.ABJ - Object code of APAFEX.APM.
      C          APAFEX.SAV - Linked version of APAFEX.ABJ.
      C
      C     *********************************************************
      C
             DIMENSION AK(1000),S(6,6,10),K4(1000)
             DIMENSION ITIM1(2),ITIM2(2)
             INTEGER CONRNG
             DO 10 II=1,1000
             K4(II)=1
          10 AK(II)=1.
             DO 17 II=1,6
             DO 17 JJ=1,6
             DO 17 KK=1,10
          17 S(II,JJ,KK)=FLOAT(II)
             NSUBS=100
             LCON=1
             CONRNG=1
             NDF=6
             LK=1
             LCONX=0
             LKSUB=1
             LSUB=1
             WRITE(6,3001)
        3001 FORMAT(' INPUT THE NUMBER OF TIMES TO BE EXECUTED')
             READ(5,*) NTEST
      C
      C
             CPU1=SECNDS(0.)
             CALL GTIM(ITIM1)
             DO 270 L=1,NTEST
      C
             DO 1000 ISUB=1,NSUBS
             IF(ISUB.EQ.1) GO TO 600
             J=IFIX(AK(LK))
         400 IF(J.EQ.K4(LCON)) GO TO 600
```

```
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:24:09                PAGE 002

0031            LCON=LCON+1
0032            IF(LCON.LT.LCONX) GO TO 400
0034            WRITE(6,500)
0035        500 FORMAT(29H0*** MFILE/KMAP INCONSISTENCY)
0036            STOP
0037        600 LSUB=LCON+CONRNG
0038            K=K4(LSUB)
0039            DO 700 J=1,NDF
0040            DO 700 I=1,NDF
0041            S(I,J,K)=S(I,J,K)+AK(LKSUB)
0042        700 LKSUB=LKSUB+1
0043            LCON=LCON+1
0044       1000 LK=LK+1
0045        270 CONTINUE
0046            CALL GTIM(ITIM2)
0047            CPU2=SECNDS(0.)
0048            WRITE(6,50) CPU1
0049            WRITE(6,50) CPU2
0050            CALL CVTTIM(ITIM1,IHR,IMI,ISE,ITI)
0051            WRITE(6,70) IHR,IMI,ISE,ITI
0052            CALL CVTTIM(ITIM2,IHR,IMI,ISE,ITI)
0053            WRITE(6,70) IHR,IMI,ISE,ITI
0054         50 FORMAT(' NUMBER OF SECONDS PAST MIDNIGHT = ',F15.7)
0055         70 FORMAT(' TIME = ',I2,':',I2,':',I2,':',I2)
0056            CPU=CPU2-CPU1
0057            WRITE(6,12)CPU,NSUBS
0058         12 FORMAT(3X,'TIME=',F16.8,I10)
          D     WRITE(6,2001)(S(I,1,1),I=1,360)
0059       2001 FORMAT(F17.6)
0060            STOP
0061            END
```

```
      COM       05-14-80     RPI# 66666  66666  66666  77777  00000  33333
      C
      C    ***********************************************
      C
      C     This is Program APFOR8.FOR which contains the AP Calls
      C     as replacement for FORTRAN code in FOR8.FOR. Represents
      C     a portion of Subroutine AFEX in Processor INV. Obtains
      C     timing information for AP execution.
      C
      C     Corresponding Programs are:
      C         FOR8.FOR
      C         APAFEX.APM
      C         APAFEX.ABJ
      C         APAFEX.SAV
      C
      C    ****************************  *******************
      C
            DIMENSION AK(1000),S(6,6,10),K4(1000)
            DIMENSION ITIM1(2),ITIM2(2),ITIM3(2),ITIM4(2)
            INTEGER CONRNG
            DO 10 II=1,1000
            K4(II)=1
         10 AK(II)=1.
            DO 17 II=1,6
            DO 17 JJ=1,6
            DO 17 KK=1,10
         17 S(II,JJ,KK)=FLOAT(II)
            NSUBS=100
            LCON=1
            CONRNG=1
            NDF=6
            LK=1
            LCONX=0
            LKSUB=1
            LSUB=1
            WRITE(6,3001)
       3001 FORMAT(' INPUT THE NUMBER OF TIMES TO BE EXECUTED')
            READ(5,*) NTEST
      C
      C
            CPU1=SECNDS(0.)
            CALL GTIM(ITIM1)
            DO 270 L=1,NTEST
      C
            CALL APCLR
            CALL APPUT(K4(1),12,1000,1)
            CALL APPUT(S(1,1,1),1012,360,2)
            CALL APPUT(AK(1),1372,1000,2)
```

```
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:24:31                 PAGE 002

0029            CALL APWD
          D     CALL GTIM(ITIM2)
0030            CALL APAFEX(1012,1372,12,NSUB,LK,LCON,LCONX,NDF,
               1CONRNG,LKSUB)
0031            CALL APWR
          D     CALL GTIM(ITIM3)
0032            CALL APGET(S(1,1,1),1012,360,2)
0033            CALL APWD
0034        270 CONTINUE
0035            CALL GTIM(ITIM4)
0036            CPU2=SECNDS(0.)
0037            CALL CVTTIM(ITIM1,IHR,IMI,ISE,ITI)
0038            WRITE(6,70)
0039            WRITE(6,75) IHR,IMI,ISE,ITI
          D     CALL CVTTIM(ITIM2,IHR,IMI,ISE,ITI)
          D     WRITE(6,80)
          D     WRITE(6,75) IHR,IMI,ISE,ITI
          D     CALLCVTTIM(ITIM3,IHR,IMI,ISE,ITI)
          D     WRITE(6,90)
          D     WRITE(6,75) IHR,IMI,ISE,ITI
0040            CALL CVTTIM(ITIM4,IHR,IMI,ISE,ITI)
0041            WRITE(6,100)
0042            WRITE(6,75) IHR,IMI,ISE,ITI
0043            WRITE(6,110) CPU1
0044            WRITE(6,110) CPU2
0045            CPU=CPU2-CPU1
0046            WRITE(6,120) CPU
0047         70 FORMAT('$TIME AT START OF DATA INPUT = ')
0048         80 FORMAT('$TIME AT COMPLETION OF DATA INPUT AND EXECUTION',
               1' START = ')
0049         90 FORMAT('$TIME AT END OF EXECUTION AND START OF DATA'
               1' OUTPUT = ')
0050        100 FORMAT('$TIME AT COMPLETION OF DATA OUTPUT = ')
0051         75 FORMAT('+',I2,':',I2,':',I2,':',I2)
0052        110 FORMAT(' NUMBER OF SECONDS PAST MIDNIGHT = ',F15.5)
0053        120 FORMAT(5X,'ELAPSED TIME = ',F10.5,' SECONDS')
          C      DO 1000 ISUB=1,NSUBS
          C      IF(ISUB.EQ.1) GO TO 600
          C      J=IFIX(AK(LK))
          C  400 IF(J.EQ.K4(LCON)) GO TO 600
          C      LCON=LCON+1
          C      IF(LCON.LT.LCONX) GO TO 400
          C      WRITE(6,500)
          C  500 FORMAT(29H0*** MFILE/KMAP INCONSISTENCY)
          C      STOP
          C  600 LSUB=LCON+CONRNG
          C      K=K4(LSUB)
          C      DO 700 J=1,NDF
```

```
      C       DO 700 I=1,NDF
      C       S(I,J,K)=S(I,J,K)+AK(LKSUB)
      C   700 LKSUB=LKSUB+1
      C       LCON=LCON+1
      C 1000 LK=LK+1
0054          WRITE(6,2001)(S(I,1,1),I=1,360)
0055    2001  FORMAT(F17.6)
0056          STOP
0057          END
```

```
                $TITLE APAFEX
             $ENTRY APAFEX,10
"
" THIS ROUTINE SIMULATES A SECTION OF CODE IN THE AFEX SUBROUTINE
" THAT IS LOCATED IN THE MATRIX INVERSION ROUTINE INV
"
" AUTHOR: K. PERSON
" DATE: APRIL 79
" Revised: L. J. Feeser and K. Matis
" Date: May 1980
"
" ----USUAGE----
" FORTRAN: CALL APAFEX(SBASE,AKBASE,K4BASE,NSUB,LK,LCON,LCONX,NDF,...
"                    ....CONRNG,LKSUB)
"
"        ALL PARAMETERS MUST BE INTEGER VALUES]
"
"
" ---MAIN DATA MEMORY MAP---
"
"  *****************************  (STARTING ADDRESS)
"  *                           *
"  *                           *
"  *                           *
"  *                           *
"  *****************************  12+LR4+ISIZE*NDF*NDF+LRK
"  *                           *
"  *                           *
"  *       AK ARRAY            *
"  *                           *
"  *****************************  12+LR4+ISIZE*NDF*NDF
"  *                           *
"  *                           *
"  *       S ARRAY             *
"  *                           *
"  *****************************  12+LR4
"  *                           *
"  *                           *
"  *       K4 ARRAY            *
"  *                           *
"  *****************************  12
"  *                           *
"  *                           *
"  *                           *
"  *                           *
"  *****************************  6
"  *                           *
"  *                           *
"  *                           *
"  *                           *
```

```
"   ******************************  0
"
"
" ---ARRAY EXPLANATIONS---
"
"  AK(LRK)................ REAL ARRAY. TRANSFERED ONLY AFTER
"                          BEING READ FROM THE DATA BASE.
"
"  S(ISIZE*NDF*NDF)....... REAL ARRAY. NEVER TRANSFERED.
"
"  K4(LR4)................ INTEGER ARRAY. TRANSFERED ONLY AFTER
"                          BEING READ FROM THE DATA BASE.
"
"
"
" S-PAD PARAMATERS
"
                    SBASE  $EQU 0     "BASE ADDRESS OF S(I,J,K)
                    AKBASE $EQU 1     "BASE ADDRESS OF AK(LK) ARRAY
                    K4BASE $EQU 2     "BASE ADDRESS OF K4(LCON) ARRAY
                    NSUB   $EQU 3     "OUTER LOOP COUNT
                    LK     $EQU 4     "ADDRESS POINTER FOR AK ARRAY
                    LCON   $EQU 5     "ADDRESS POINTER FOR K4 ARRAY
                    LCONX  $EQU 6     "CONSTANT=L4+CONRNG
                    NDF    $EQU 7     "NUMBER OF DEGREES OF FREEDOM
                    NDF2   $EQU 7     "NDF*NDF
                    CONRNG $EQU 10    "NUMBER OF NONZERO SUBMATRICES
                    LKSUB  $EQU 11     "ADDRESS POINTER FOR AK ARRAY
                    TEMP   $EQU 11     "SCRATCH REGISTER=K4(LCON)
                    J      $EQU 12     "SCRATCH REGISTER=IFIX(AK(LK))
                    S1ADDR $EQU 12     "ADDRESS POINTER FOR S(I,J,K)
                    S2ADDR $EQU 13     "ADDRESS POINTER FOR TARGET S(I,J,K)
                    AKADDR $EQU 14     "ADDRESS POINTER FOR AK ARRAY
                    CNT    $EQU 15     "INNER LOOP COUNT
                    N27    $EQU 15     "CONSTANT=27.
                    ISUB   $EQU 16     "OUTER LOOP COUNTER
                    LSUB   $EQU 17     "ADDRESS POINTER FOR K4 ARRAY
"
"
" CALCULATE NDF*NDF AND STORE THE RESULT
" THIS VALUE IS USED TO CALCULATE THE ADDRESS OF S(NDF,NDF,K)
" FOR ANY GIVEN K.
"
"
APAFEX: LDSPI N27; DB=27.                         "LOAD CONSTANT=27.
        MOV NDF,NDF; DPX(0)<SPFN                   "DPX(0)=NDF
        FADD ZERO,MDPX(0); MOV N27,N27              "FLOAT NDF
        FADD                                      "PUSH
        DPX(0)<FA                                 "SAVE FLOAT(NDF)
        FMUL DPX(0),DPX(0)                         "NDF*NDF
```

```
        FMUL;                                            "PUSH
                  CLR ISUB                               "CLEAR OUTER COUNT
        FMUL;                                            "PUSH
                  MOV AKBASE,AKADDR                       "LOAD ADDRESS POINTER
        DPX(2)<FM;                                       "STORE NDF*NDF IN DPX(0)
                  ADD LKSUB,AKADDR                        "POINT TO AK(LKSUB)
        FIX DPX(2)                                       "FIX(NDF*NDF)
        FADD                                            "PUSH
        DPX(1)<FA                                       "SAVE NDF*NDF
        LDSPI NDF2; DB=DPX(1)                                "NDF2=NDF*NDF
        JMP LOOP3
"
"
" FORTRAN:  DO 1000 ISUB=1,NSUB
"           IF (ISUB .EQ. 1) GO TO 600
"           J=IFIX(AK(LK))
"       400 IF (J .EQ. K4(LCON)) GO TO 600
"
"
LOOP1:  ADD# LK,AKBASE; SETMA                             "GET AK(LK)
        NOP
                  ADD# LCON,K4BASE; SETMA                   "GET K4(LCON)
        FIX MD                                          "INT(AK(LK))
        FADD                                            "PUSH
        DPX(1)<FA                                       "SAVE J
        LDSPI J; DB=DPX(1)                                   "STORE J=AK(LK)
                  LDSPI TEMP; DB=MD                       "STORE K4(LCON)
        SUB# J,TEMP                                      "J=K4(LCON) ?
        BEQ LOOP3                                       "BRANCH IF YES
"
" FORTRAN:  LCON=LCON+1
"           IF (LCON .LT. LCONX) GO TO 400
"
"
LOOP2:  INC LCON
        SUB# LCON,LCONX                                 "LCON .LT. LCONX ?
        BGT CONT                                        "IF YES, CONTINUE
        JMP ERROR                                      "IF NOT, ERROR
"
"
CONT:   ADD# LCON,K4BASE; SETMA                           "GET K4(LCON+1)
        NOP
        NOP
        LDSPI TEMP; DB=MD
        SUB# J,TEMP                                      "K4(LCON+1)=J ?
        BEQ LOOP3                                       "IF YES, BRANCH
"
"
        JMP LOOP2                                       "CONTINUE TEST
"
```

```
"
" FIND THE STARTING ADDRESS OF THE S ARRAY
" GIVEN THAT DPX(2)=NDF*NDF
"
"
" FORTRAN:  K=K4(LSUB)
"
"
LOOP3:  MOV K4BASE,LSUB                         "FIND LSUB
        ADD LCON,LSUB
        ADD CONRNG,LSUB; SETMA                   "GET K4(LSUB)
        RPSF ONE; DPY(0)<DB                     "DPY(0)=1.
        LDSPI N27; DB=27.                       "LOAD CONSTANT
        DPX(0)<MD                               "SAVE K
        FSUBR DPY(0),MDPX(0); MOV N27,N27         "FLOAT K-1
        FADD                                    "PUSH
        FMUL DPX(2),FA                          "DO K-1 * NDF**2
        FMUL                                    "PUSH
        FMUL;                                   "PUSH
                INC ISUB                        "INC OUTER LOOP COUNT
        DPX(0)<FM;                              "SAVE (K-1)*NDF**2
                INC LCON                        "INC LCON
        FIX DPX(0)                              "INT((K-1)*NDF**2)
        FADD                                    "PUSH
        DPX(1)<FA                               "SAVE ADDRESS POINTER
        LDSPI S1ADDR; DB=DPX(1)                     "SAVE ADDRESS POINTER
"
"
" PERFORM THE ADDITION OF AK AND S
"
"
" FORTRAN:    DO 700 J=1,NDF
"             DO 700 I=1,NDF
"             S(I,J,K)=S(I,J,K)+AK(LKSUB)
"         700 LKSUB=LKSUB+1
"
"
        ADD SBASE,S1ADDR; SETMA                       "GET S(1,1,K)
        MOV S1ADDR,S2ADDR                       "LOAD TARGET ADDRESS
        MOV AKADDR,AKADDR; SETMA                 "GET AK(LKSUB)
        DPX(0)<MD;                              "SAVE S(1,1,K)
                MOV NDF2,CNT                     "LOAD COUNTER
        DEC S2ADDR                              "LOOP SET-UP
LOOP4:          FADD DPX(0),MD                   "S(I,J,K)+AK(LKSUB)
        INC S1ADDR; SETMA;                      "GET NEXT S(I,J,K)
                FADD                            "PUSH
        NOP
        INC AKADDR; SETMA                       "GET NEXT AK(LKSUB)
        DPX(0)<MD;                              "SAVE S(I,J,K)
                DEC CNT                         "TEST COUNT
```

```
          INC S2ADDR; SETMA; MI<FA;                 "SAVE RESULT
                    BNE LOOP4
"
"
"
          SUB# NSUB,ISUB                           "TEST OUTER MOST LOOP
          BEQ CONT1
          JMP LOOP1                               "START AGAIN
ONE:      $FP 1.0
ERROR:    NOP
CONT1:    RETURN
         $END
```

APPENDIX H

Listings of:

FOR9.FOR

APFOR9.FOR

APMLTX.APM

```
.TYPE FOR9
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:25:36                 PAGE 001

      COM       05-14-80      RPI# 66666  66666  66666  77777  22222  33333
      CCOMDECK MULTEX
      C
      C    ***:**********************************************
      C
      C     This is Program FOR9.FOR which represents a portion of
      C     Subroutine MULTEX in CDEC Library. Used to obtain
      C     timing information for FORTRAN execution.
      C
      C     Corresponding Programs are:
      C         APFOR9.FOR - FORTRAN of FOR9.FOR with AP Calls.
      C         APMLTX.APM - FP120 Assembler Program replacement
      C                 of FORTRAN portion. Called from APSMLT.
      C         APMLTX.ABJ - Object code of APMLTX.APM.
      C         APMLTX.SAV -
      C
      C         FOR7.FOR
      C         APFOR7.FOR
      C         APSMLT.APM
      C         APSMLT.ABJ
      C         APSMLT.SAV
      C
      C    **************************************************
      C
      C     SUBROUTINE MULTEX
      C    $(   N,NSUBS,JLIST,A,VIN,VOUT)
      C     8/74 WD WHETSTONE
0001        DIMENSION A(6,6,50),VIN(6,50),VOUT(6,50),JLIST(50)
0002        DIMENSION ITIM1(2),ITIM2(2)
0003        NSUBS=50
0004        N=6
      C
0005        DO 11 II=1,50,5
0006        JLIST(II)=1
0007        JLIST(II+1)=2
0008        JLIST(II+2)=3
0009        JLIST(II+3)=4
0010        JLIST(II+4)=5
0011     11 CONTINUE
0012        DO 12 II=1,6
0013        DO 12 JJ=1,6
0014        DO 12 KK=1,50
0015        IF(II.NE.1) GO TO 12
0017        VIN(JJ,KK)=FLOAT(JJ)
0018     12 A(II,JJ,KK)=FLOAT(II)
0019        WRITE(6,3001)
0020   3001 FORMAT(' INPUT THE NUMBER OF TIMES TO BE EXECUTED')
0021        READ(5,*) NTEST
```

```
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:25:36                PAGE 002

            CPU1=SECNDS(0.)
            CALL GTIM(ITIM1)
            DO 270 LL=1,NTEST
            I=JLIST(1)
            DO 100 L=1,N
            DO 100 M=1,N
      100   VOUT(L,I)=VOUT(L,I)+A(L,M,1)*VIN(M,I)
            IF(NSUBS.LT.2) GO TO 300
            DO 200 K=2,NSUBS
            J=JLIST(K)
            DO 200 L=1,N
            DO 200 M=1,N
            VOUT(L,I)=VOUT(L,I)+A(L,M,K)*VIN(M,J)
      200   VOUT(L,J)=VOUT(L,J)+A(M,L,K)*VIN(M,I)
      270   CONTINUE
            CALL GTIM(ITIM2)
            CPU2=SECNDS(0.)
            WRITE(6,50) CPU1
            WRITE(6,50) CPU2
            CALL CVTTIM(ITIM1,IHR,IMI,ISE,ITI)
            WRITE(6,70) IHR,IMI,ISE,ITI
            CALL CVTTIM(ITIM2,IHR,IMI,ISE,ITI)
            WRITE(6,70) IHR,IMI,ISE,ITI
      50    FORMAT(' NUMBER OF SECONDS PAST MIDNIGHT = ',F15.7)
      70    FORMAT(' TIME = ',I2,':',I2,':',I2,':',I2)
            CPU=CPU2-CPU1
            WRITE(6,33)CPU
       33   FORMAT(5X,5HTIME=,F16.7)
            STOP
     300    STOP 'NSUBS.LT.2'
            END
```

```
 TYPE APFOR9
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:26:13                  PAGE 001

      COM       05-14-80      RPI# 66666  66666  66666  77777  22222  33333
      CCOMDECK MULTEX
      C
      C     ***********************************************
      C
      C      This is Program APFOR9.FOR which contains the AP Calls
      C      as replacements for FORTRAN code in FOR9.FOR. Represents
      C      a portion of Subroutine MULTEX in CDEC Library. Obtains
      C      timing information for AP execution.
      C
      C      Corresponding Programs are:
      C          FOR9.FOR
      C          APMLTX.APM
      C          APMLTX.ABJ
      C          APMLTX.SAV
      C
      C     ***********************************************
      C
      C      SUBROUTINE MULTEX
      C     $(   N,NSUBS,JLIST,A,VIN,VOUT)
      C      8/74 WD WHETSTONE
0001         DIMENSION A(6,6,50),VIN(6,50),VOUT(6,50),JLIST(50)
0002         DIMENSION ITIM1(2),ITIM2(2),ITIM3(2),ITIM4(2)
0003         NSUBS=50
0004         N=6
      C
0005         DO 11 II=1,46,5
0006         JLIST(II)=1
0007         JLIST(II+1)=2
0008         JLIST(II+2)=3
0009         JLIST(II+3)=4
0010         JLIST(II+4)=5
0011      11 CONTINUE
0012         DO 12 II=1,6
0013         DO 12 JJ=1,6
0014         DO 12 KK=1,50
0015         IF(II.NE.1) GO TO 12
0017         VIN(JJ,KK)=FLOAT(JJ)
0018      12 A(II,JJ,KK)=FLOAT(II)
0019         WRITE(6,3001)
0020    3001 FORMAT(' INPUT THE NUMBER OF TIMES TO BE EXECUTED')
0021         READ(5,*)NTEST
0022         CPU1=SECNDS(0.)
0023         CALL GTIM(ITIM1)
0024         DO 270 LL=1.NTEST
0025         CALL APCLR
0026         CALL APPUT(JLIST(1),0,50,1)
0027         CALL APPUT(A(1,1,1),50,1800,2)
```

```
              CALL APPUT(VIN(1,1),1850,300,2)
              CALL APWD
      D       CALL GTIM(ITIM2)
              CALL APMLTX(50,2150,1850,0,N,NSUBS)
              CALL APWR
      D       CALL GTIM(ITIM3)
              CALL APGET(VOUT(1,1),2150,300,2)
              CALL APWD
        270      CONTINUE
              CALL GTIM(ITIM4)
              CPU2=SECNDS(0.)
              CALL GTIM(ITIM4)
              CPU2=SECNDS(0.)
              CALL CVTTIM(ITIM1,IHR,IMI,ISE,ITI)
              WRITE(6,70)
              WRITE(6,75) IHR,IMI,ISE,ITI
      D       CALL CVTTIM(ITIM2,IHR,IMI,ISE,ITI)
      D       WRITE(6,80)
      D       WRITE(6,75) IHR,IMI,ISE,ITI
      D       CALLCVTTIM(ITIM3,IHR,IMI,ISE,ITI)
      D       WRITE(6,90)
      D       WRITE(6,75) IHR,IMI,ISE,ITI
              CALL CVTTIM(ITIM4,IHR,IMI,ISE,ITI)
              WRITE(6,100)
              WRITE(6,75) IHR,IMI,ISE,ITI
              WRITE(6,110) CPU1
              WRITE(6,110) CPU2
              CPU=CPU2-CPU1
              WRITE(6,120) CPU
         70   FORMAT('$TIME AT START OF DATA INPUT = ')
         80   FORMAT('$TIME AT COMPLETION OF DATA INPUT AND EXECUTION',
             1' START = ')
         90   FORMAT('$TIME AT END OF EXECUTION AND START OF DATA'
             1' OUTPUT = ')
        100   FORMAT('$TIME AT COMPLETION OF DATA OUTPUT = ')
         75   FORMAT('+',I2,':',I2,':',I2,':',I2)
        110   FORMAT(' NUMBER OF SECONDS PAST MIDNIGHT = ',F15.5)
        120   FORMAT(5X,'ELAPSED TIME = ',F10.5,' SECONDS')
      C        I=JLIST(1)
      C        DO 100 L=1,N
      C        DO 100 M=1,N
      C   100 VOUT(L,I)=VOUT(L,I)+A(L,M,1)*VIN(M,I)
      C        IF(NSUBS.LT.2) GO TO 300
      C        DO 200 K=2,NSUBS
      C        J=JLIST(K)
      C        DO 200 L=1,N
      C        DO 200 M=1,N
      C        VOUT(L,I)=VOUT(L,I)+A(L,M,K)*VIN(M,J)
```

```
      C   200 VOUT(L,J)=VOUT(L,J)+A(M,L,K)*VIN(M,I)
0056          STOP
0057     300  STOP 'NSUBS.LT.2'
0058          END
```

```
                    $TITLE APMLTX
                    $ENTRY APMLTX,6
"
" SIMULATES SUBROUTINE MULTEX(N,NSUBS,JLIST,A,VIN,VOUT)
"
" AUTHOR: K. PERSON
" DATE: FEB 79
" Revised: L. J. Feeser and K. Matis
" Date: May 1980
"
"
" ---USAGE---
"
" FORTRAN: CALL APMLTX(ABASE,INBASE,OTBASE,JBASE,NDF,NSUB)
"
"              ALL PARAMETERS MUST BE INTEGER]
"
"
" ---MAIN DATA MEMORY MAP---
"
"
"  *****************************  (STARTING ADDRESS)
"  *                           *
"  *      IN ARRAY             *
"  *                           *
"  *****************************  NV*LR+NV*LVOUT
"  *                           *
"  *      OUT ARRAY            *
"  *                           *
"  *****************************  NV+LR
"  *                           *
"  *      A OR IA ARRAY        *
"  *                           *
"  *****************************  NV
"  *                           *
"  *      LSDO ARRAY           *
"  *                           *
"  *****************************  0
"
"
" ---ARRAY EXPLANATIONS---
"
" IN(NV*LVIN)..... REAL ARRAY.
"
" OUT(NV*LVOUT)... REAL ARRAY.
"
" IA(LR).......... REAL ARRAY.
"
" LSDO(NV)........ INTEGER ARRAY.
"
```

```
"
" S-PAD PARAMETERS
"
                    A       $EQU 0      "ADDRESS OF A OR A(L,M,K)
                    ABASE   $EQU 0      "BASE ADDRESS OF ARRAY A
                    INBASE  $EQU 1      "BASE ADDRESS OF ARRAY VIN
                    OTBASE  $EQU 2      "BASE ADDRESS OF ARRAY VOUT
                    JBASE   $EQU 3      "BASE ADDRESS OF ARRAY JLIST
                    NDF     $EQU 4      "NDF=N  NUMBER OF DEGREES OF FREEDOM
                    NSUB    $EQU 5      "NUMBER OF SUBMATRICES
                    VOUT    $EQU 6      "ADDRESS OF VOUT
                    C       $EQU 6      "ADDRESS OF C OR VOUT(L,I)
                    VIN     $EQU 7      "ADDRESS OF VIN
                    F       $EQU 7      "ADDRESS OF F OR VOUT(L,J)
                    NDFSQ   $EQU 10    "NDF SQUARED
                    N27     $EQU 11    "CONSTANT
                    OUTCT   $EQU 12    "LOOP COUNTER
                    CNT     $EQU 12    "LOOP COUNTER
                    COUNT   $EQU 13    "LOOP COUNTER
                    CT2     $EQU 13    "LOOP COUNTER
                    D       $EQU 14    "ADDRESS OF D OR A(M,L,K)
                    E       $EQU 15    "ADDRESS OF E OR VIN(M,I)
                    B       $EQU 16    "ADDRESS OF B OR VIN(M,J)
                    CNTER   $EQU 17    "LOOP COUNTER
"
"CALCULATE THE STARTING ELEMENT OF THE VOUT AND VIN ARRAYS
" THEREFORE FIND (I-1)*NDF
"
" FORTRAN:   I=JLIST(1)
"
APMLTX:  LDSPI N27;DB=27.                                   "LOAD CONSTANT
         DPY(1)<DB; DB=40000; WRTMAN                         "DPY(1)=1
         DPY(1)<DB; DB=1015; WRTEX
         MOV JBASE,JBASE; SETMA                              "GET I=JLIST(1)
                   MOV NDF,NDF; DPX(2)<SPFN                  "FLOAT NDF
                   FADD ZERO,MDPX(2); MOV N27,N27
         DPX(0)<MD                                          "STORE I
         FSUBR DPY(1),MDPX(0); MOV N27,N27                   "I-1
         FADD;
                   DPX(2)<FA                                "STORE NDF
         FMUL DPX(2),FA                                     "(I-1)*NDF
                   FMUL DPX(2),DPX(2)                        "NDF*NDF
         FMUL
         DPX(0)<FM;                                         "STORE I-1*NDF
                   FMUL
         FIX DPX(0);                                        "INT(I-1*NDF)
                   DPX(0)<FM                                "STORE NDF*NDF
                   FIX DPX(0)
         DPX(0)<FA                                         "STORE(I-1)*NDF
         LDSPI VIN; DB=DPX(0);                               "LOAD VIN
```

```
                    FADD
                    DPX(0)<FA                                 "STORE NDF*NDF
                    LDSPI NDFSQ; DB=DPX(0)                    "STORE NDF*NDF
"
" SET UP FOR 100 LOOP
"
" FORTRAN:    DO 100 L=1,N
"             DO 100 M=1,N
"         100 VOUT(L,I)=VOUT(L,I)+A(L,M,1)*VIN(M,I)
"
          MOV NDF,OUTCT                                       "LOAD COUNTER
          MOV VIN,VOUT                                        "LOAD VOUT
          DEC VOUT
          ADD INBASE,VIN                                      "GET VIN ADDRESS
          ADD OTBASE,VOUT                                     "GET VOUT ADDRESS
          DEC ABASE
          ADD NDF,VIN
"
"DO .... VOUT(L,I)=VOUT(L,I)+A(L,M,K)*VIN(M,J)
"
LOOP1:    INC VOUT;SETMA                                      "GET VOUT(L,I)
          NOP
          INC ABASE;SETMA                                     "GET A(L,M,I)
          FADD ZERO,MD                                        "VOUT(L,I)+0
          SUB NDF,VIN; SETMA; FADD                            "GET VIN(M,I)
          DPX(1)<MD             "STORE A(L,M,I)
                                MOV NDF,COUNT                 "LOAD COUNT
LOOP2:              FMUL DPX(1),MD                            "VIN*A
          ADD NDF,ABASE; SETMA;                               "GET A(L,M+1,I)
                    FMUL
                    FMUL
          INC VIN;SETMA;                                      "GET VIN(M+1,I)
                    FADD FM,FA                                "VOUT+VIN*A
          DPX(1)<MD;                                          "STORE A(L,M+1,I)
                    FADD;
                                DEC COUNT
END2:              MI<FA; MOV VOUT,VOUT; SETMA;               "STORE VOUT
                                BNE LOOP2                     "TEST M=NDF
                                DEC OUTCT
                                BEQ CONT1                     "TEST L=NDF
        SUB NDFSQ,ABASE                                       "REAJUST A ADDRESS
END1:                           JMP LOOP1
"
"CHECK FOR NSUB LESS THAN TWO   (IE..NSUB=1)
"
" FORTRAN: IF(NSUBS .LT. 2 ) GO TO 300
"
CONT1:    DEC# NSUB
          BNE CONT2;
          DEC NDFSQ                                           "GET NDF**2-1
```

```
        JMP END
"
" SET UP FOR THE 200 LOOPS
"
" FORTRAN:    DO 200 K=2,NSUBS
"             J=JLIST(K)
"             DO 200 L=1,N
"             DO 200 M=1,N
"             VOUT(L,I)=VOUT(L,I)+A(L,M,K)*VIN(M,J)
"         200 VOUT(L,J)=VOUT(L,J)+A(M,L,K)*VIN(M,I)
"
CONT2:  SUB NDF,ABASE
        MOV ABASE,D                                      "ADD OF A(M,L,K)
        SUB NDF,ABASE                                   "REAJUST A
        LDSPI CNTER; DB=1.                             "LOAD CNTER=1
        INC A
        INC C
        MOV VIN,E                                       "LOAD VIN(M,I)
        SUB NDF,E
        DEC E                                          "REAJUST C
        MOV NDF,CT2                                     "LOAD COUNTER
"
"CALCULATE THE STARTING ELEMENTS OF VOUT(L,J) AND VIN(M,J)
" THEREFORE.... (J-1)*NDF
"
LOOP3:  INC JBASE; SETMA                                "GET J
                          INC CNTER                    "INCREMENT K COUNT
        NOP
        DPX(0)<MD; SUB NDF,C                             "STORE J, INC VOU
        FSUBR DPY(1),MDPX(0); MOV N27,N27                " J-1
        FADD;
                          MOV NDF,CNT                   "LOAD COUNT
        FMUL DPX(2),FA                                 "J-1*NDF
        FMUL
        FMUL
        DPX(0)<FM                                      "STORE J-1*NDF
        FIX DPX(0)
        FADD
        DPX(0)<FA
        LDSPI F; DB=DPX(0)                             "STORE INCREMENT IN F
        MOV F,B                                        "STORE INCREMENT IN B
        ADD OTBASE,F                                   "GET F ADDRESS
        ADD INBASE,B                                   "GET B ADDRESS
        DEC B
"
"
" DO ...  VOUT(L,I)=VOUT(L,I)+A(L,M,K)*VIN(M,J)
"         VOUT(L,J)=VOUT(L,J)+A(M,L,K)*VIN(M,I)
"
"
```

```
LOOP4:  ADD NDF,ABASE; SETMA                                    "GET A
        NOP
        INC B; SETMA                                            "GET B
        DPX(1)<MD                                               "STORE A
        MOV C,C; SETMA                                          "GET C
                FMUL DPX(1),MD                                  "A*B
        INC D; SETMA;                                           "GET D
                FMUL
        DPY(2)<MD;                                              "STORE C
                FMUL
        INC E; SETMA;                                           "GET E
                FADD FM,DPY(2)                                  "C+A*B
        DPX(0)<MD;                                              "STORE D
                FADD
                MOV C,C;SETMA; MI<FA                            "STORE C=C+A*B
                FMUL DPX(0),MD                                  "E*D
        MOV F,F;SETMA;                                          "GET F
                FMUL
                FMUL
                NOP
                FADD FM,MD                                      "F+D*E
                FADD;
                        DEC CNT                                 "TEST INNER LOOP
                MI<FA; MOV F,F; SETMA;                          "STORE F=F+D*E
                        BEQ CONT3
END4:                   JMP LOOP4
CONT3:  INC F                                                   "INC F ADDRESS
        MOV NDF,CNT                                             "RELOAD COUNTER
        INC C
        SUB NDF,E                                               "REAJUST E ADDRESS
        SUB NDFSQ,ABASE                                         "REAJUST A ADDRESS
        DEC CT2
                        BEQ CONT4;                              "TEST OUTER LOOP
        SUB NDF,B                                               "REAJUST B ADDRESS
END3:           JMP LOOP4
"
" TEST FOR K=NSUBS
"
CONT4:  MOV D,ABASE
        MOV NDF,CT2
        SUB NDF,ABASE                                           "REAJUST A
        SUB# NSUB,CNTER                                         "TEST K
        BEQ END;  INC ABASE
        JMP LOOP3
"
" FORTRAN:  300 RETURN
"
END:    RETURN
       $END
```

APPENDIX I

Listings of:

FORO.FOR

APFORO.FOR

APTRN6.APM

```
 TYPE FORO
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:26:52              PAGE 001

      CCOMDECK TRAN6
      C
      C     **************************************************
      C
      C      This is Program FORO.FOR, which represents a portion of
      C      Subroutine TRAN6 in the SPAR Library.  Used to obtain timing
      C      information for FORTRAN execution.
      C
      C      Corresponding Programs are:
      C          APFORO.FOR - FORTRAN of FORO with AP Calls
      C          APTRN6.APM - FP120 Assembler Program replacement of
      C                  FORTRAN portions.
      C          APTRN6.ABJ - Object code of APTRN6.APM
      C          APTRN6.SAV - Linked version of APTRN6.APM
      C
      C     **************************************************
      C
      C      SUBROUTINE TRAN6(MAP,H,NDF,NNODES,ITRANS,T)
      C      4/73 WD WHETSTONE
0001         COMMON/CONSTR/ JT,JDF,JDDF,INEX(6),MEXIN(6),KSYM(3)
0002         COMMON/TEMPS/
            $ HKL(6,6),GKLTL(6,6)
0003         COMMON/LOCAL/S(6,6,10)
0004         DIMENSION MAP(10),H(6,6,1),ITRANS(4),T(3,3,1)
0005         DIMENSION ITIM1(2),ITIM2(2)
      C H(NDF,NDF,1) WHERE THE TOTAL DIM=NDF*NDF*KSIZE
      C     $INSERT SYSCOM>A$KEYS
0006         CALL ASSIGN(4,'RK1:SFILE.DAT',13,'RDO')
0007         CALL ASSIGN(2,'RK1:TFILE.DAT',0,'RDO')
0008         CALL ASSIGN(3,'RK1:HFILE.DAT',0,'RDO')
0009      44 FORMAT(F16.7)
      C       READ(7,44)Z,S
      C       READ(8,44)Z,T
      C       READ(9,44)Z,H
0010         WRITE(6,3001)
0011    3001     FORMAT('  INPUT THE NUMBER IF TIMES TO BE EXECUTED')
0012         READ(5,*) NTEST
0013         READ(4,44) (((S(I,J,K),I=1,6),J=1,6),K=1,10)
0014         READ(2,44) ((T(I,J,1),I=1,3),J=1,3)
0015         READ(3,44) ((H(I,J,1),I=1,6),J=1,6)
0016         CPU1=SECNDS(0.)
0017         CALL GTIM(ITIM1)
0018         DO 3000 ITEST=1,NTEST
0019         NDF=3
0020         NNODES=3
0021         DO 4 I=1,4
0022       4 ITRANS(I)=1
0023         KK=1
```

```
      DO 7 I=1,10
      MAP(I)=KK
    7 KK=-KK
      DO 9 I=1,6
    9 INEX(I)=1
      N=0
   50 FORMAT(' NUMBER OF SECONDS PAST MIDNIGHT = ',F15.7)
      DO 2000 L=1,NNODES
      LL=ITRANS(L)
      DO 2000 K=1,L
      KK=ITRANS(K)
      N=N+1
C
      DO 100 J=1,6
      DO 100 I=1,6
      GKLTL(I,J)= .0
  100 HKL(I,J)=   .0
C     FORM HKL=  TK(TRANSPOSE) *GKL *TL
C     FIRST,  GKL*TL.
C
      DO 1100 J=1,3
      DO 1100 I=1,3
      DO 1100 M=1,3
      TLMJ=T(M,J,LL)
      GKLTL(  I,  J)= GKLTL(  I,  J) +S(  I,  M,N) *TLMJ
      GKLTL(  I,J+3)= GKLTL(  I,J+3) +S(  I,M+3,N) *TLMJ
      GKLTL(I+3,  J)= GKLTL(I+3,  J) +S(I+3,  M,N) *TLMJ
 1100 GKLTL(I+3,J+3)= GKLTL(I+3,J+3) +S(I+3,M+3,N) *TLMJ
C
C     TK(TRANSPOSE)*(GKL*TL)
      DO 1200 J=1,3
      DO 1200 I=1,3
      DO 1200 M=1,3
      TKMI=T(M,I,KK)
      HKL(  I,  J)= HKL(  I,  J) +TKMI*GKLTL(  M,  J)
      HKL(  I,J+3)= HKL(  I,J+3) +TKMI*GKLTL(  M,J+3)
      HKL(I+3,  J)= HKL(I+3,  J) +TKMI*GKLTL(M+3,  J)
 1200 HKL(I+3,J+3)= HKL(I+3,J+3) +TKMI*GKLTL(M+3,J+3)
C
C     TRANSPOSE, IF REQ.
      IF(MAP(N).GT.0) GO TO 1400
      DO 1300 J=2,6
      JM=J-1
      DO 1300 I=1,JM
      EIJ= HKL(I,J)
      HKL(I,J)=HKL(J,I)
 1300 HKL(J,I)=EIJ
 1400 LOC=IABS(MAP(N))
```

```
      IF(NDF.LT.6) GO TO 1600
      DO 1500 J=1,6
      DO 1500 I=1,6
 1500 H(I,J,LOC)= H(I,J,LOC)+HKL(I,J)
      GO TO 2000
 1600 DO 1700 I=1,NDF
      NROW=INEX(I)
      DO 1700 J=1,NDF
      NCOL=INEX(J)
 1700 H(I,J,LOC)=H(I,J,LOC)+HKL(NROW,NCOL)
 2000 CONTINUE
 3000 CONTINUE
      CALL GTIM(ITIM2)
      CPU2= SECNDS(0.)
      WRITE(6,50) CPU1
      WRITE(6,50) CPU2
      CALL CVTTIM(ITIM1,IHR,IMI,ISE,ITI)
      WRITE(6,70) IHR,IMI,ISE,ITI
   70 FORMAT(' TIME = ',I2,':',I2,':',I2,':',I2)
      CALL CVTTIM(ITIM2,IHR,IMI,ISE,ITI)
      WRITE(6,70) IHR,IMI,ISE,ITI
      CPU = CPU2 - CPU1
      WRITE(6,99)CPU
   99 FORMAT('  EXECUTION TIME =',F12.7,'  SECONDS')
      WRITE(6,88) (H(I,1,1),I=1,36),(GKLTL(I,1),I=1,36),
     X(HKL(I,1),I=1,36)
   88 FORMAT(3E16.7)
      STOP
      END
```

```
 TYPE APFORO
FORTRAN IV      V02.5-2    Tue 04-Nov-80 10:27:20               PAGE 001

      CCONDECK TRAN6
      C
      C     ***********************************************
      C
      C      This is Program APFORO.FOR which contains the AP Calls
      C      as replacements for FORTRAN Code in FORO.FOR.  Represents
      C      a portion of Subroutine TRAN6 in the SPAR Library.
      C      Obtains timing information for AP execution.
      C
      C      Corresponding Programs are:
      C          FORO.FOR
      C          APTRN6.APN
      C          APTRN6.ABJ
      C          APTRN6.SAV
      C
      C     ***********************************************
      C
      C      SUBROUTINE TRAN6(MAP,H,NDF,NNODES,ITRANS,T)
      C      4/73 WD WHETSTONE
0001         COMMON/CONSTR/ JT,JDF,JDDF,INEX(6),NEXIN(6),KSYM(3)
0002         COMMON/TEMPS/
            $ HKL(6,6),GKLTL(6,6)
0003         COMMON/LOCAL/S(6,6,10)
0004         DIMENSION MAP(10),H(6,6,1),ITRANS(4),T(3,3,1),CLR(600)
0005         DIMENSION ITIM1(2),ITIM2(2),ITIM3(2),ITIM4(2),ITIM5(2)
0006         DIMENSION ITIM6(2),ITIM7(2)
      C H(NDF,NDF,1) WHERE THE TOTAL DIM=NDF*NDF*KSIZE
      C     $INSERT SYSCOM>A$KEYS
0007         CALL APCLR
0008         CALL APPUT(0,15000,1,1)
0009         CALL APPUT(0,15500,1,1)
0010         DO 20 LOOP=1,200
0011         IADRO=(LOOP-1)*9
0012         CALL APPUT(IADRO,15000+LOOP,1,1)
0013         IADRO=(LOOP-1)*6
0014         CALL APPUT(IADRO,15500+LOOP,1,1)
0015  20     CONTINUE
0016         CALL APWD
0017         DO 18 LOOP=1,600
0018         CLR(LOOP)=0.0
0019  18     CONTINUE
0020         CALL APPUT(CLR,0,600,2)
0021         CALL ASSIGN(1,'RK1:SFILE.DAT',0,'RDO')
0022         CALL ASSIGN(2,'RK1:TFILE.DAT',0,'RDO')
0023         CALL ASSIGN(3,'RK1:HFILE.DAT',0,'RDO')
0024      44 FORMAT(F16.7)
      C       READ(7,44)Z,S
      C       READ(8,44)Z,T
```

```
C     READ(9,44)Z,H
      WRITE(6,3001)
3001     FORMAT('  INPUT THE NUMBER IF TIMES TO BE EXECUTED')
      READ(5,*) NTEST
      READ(1,44) (((S(I,J,K),I=1,6),J=1,6),K=1,10)
      READ(2,44) ((T(I,J,1),I=1,3),J=1,3)
      READ(3,44) ((H(I,J,1),I=1,6),J=1,6)
      CPU1=SECNDS(0.)
      CALL GTIM(ITIM7)
      DO 3000 ITEST=1,NTEST
D     CALL GTIM(ITIM1)
      NDF=3
      NNODES=3
      DO 4 I=1,4
    4 ITRANS(I)=1
      KK=1
      DO 7 I=1,10
      MAP(I)=KK
    7 KK=-KK
      DO 9 I=1,6
    9 INEX(I)=1
      N=0
   50 FORMAT(' NUMBER OF SECONDS PAST MIDNIGHT = ',F15.7)
C
      CALL APPUT(INEX,0,6,1)
      CALL APPUT(ITRANS(1),6,4,1)
      CALL APPUT(S(1,1,1),82,360,2)
      CALL APPUT(H(1,1,1),442,36,2)
      CALL APPUT(T(1,1,1),488,9,2)
      CALL APPUT(MAP(1),478,10,1)
      CALL APWD
D     CALL GTIM(ITIM2)
      CALL APTRN6(488,NDF,NNODES,478)
      CALL APWR
D     CALL GTIM(ITIM3)
      CALL APGET(H(1,1,1),442,36,2)
      CALL APWD
D     CALL GTIM(ITIM4)
C
C    *********************************************
C
C     FORTRAN Code replacement - Begin
C
C      DO 2000 L=1,NNODES
C      LL=ITRANS(L)
C      DO 2000 K=1,L
C      KK=ITRANS(K)
C      N=N+1
```

```
CC
C       DO 100 J=1,6
C       DO 100 I=1,6
C       GKLTL(I,J)= .0
C  100 HKL(I,J)=   .0
CC      FORM HKL=  TK(TRANSPOSE) *GKL *TL
CC      FIRST,  GKL*TL.
CC
C       DO 1100 J=1,3
C       DO 1100 I=1,3
C       DO 1100 M=1,3
C       TLMJ=T(M,J,LL)
CNC       GKLTL(  I,  J)= GKLTL(  I,  J) +S(  I,  M,N) *TLMJ
C       GKLTL(  I,J+3)= GKLTL(  I,J+3) +S(  I,M+3,N) *TLMJ
C       GKLTL(I+3,  J)= GKLTL(I+3,  J) +S(I+3,  M,N) *TLMJ
C 1100 GKLTL(I+3,J+3)= GKLTL(I+3,J+3) +S(I+3,M+3,N) *TLMJ
CC
C      TK(TRANSPOSE)*(GKL*TL)
C       DO 1200 J=1,3
C       DO 1200 I=1,3
C       DO 1200 M=1,3
C       TKMI=T(M,I,KK)
C       HKL(  I,  J)= HKL(  I,  J) +TKMI*GKLTL(  M,  J)
C       HKL(  I,J+3)= HKL(  I,J+3) +TKMI*GKLTL(  M,J+3)
C       HKL(I+3,  J)= HKL(I+3,  J) +TKMI*GKLTL(M+3,  J)
C 1200 HKL(I+3,J+3)= HKL(I+3,J+3) +TKMI*GKLTL(M+3,J+3)
CC
CC      TRANSPOSE, IF REQ.
C       IF(MAP(N).GT.0) GO TO 1400
C       DO 1300 J=2,6
C       JM=J-1
C       DO 1300 I=1,JM
C       EIJ= HKL(I,J)
C       HKL(I,J)=HKL(J,I)
C 1300 HKL(J,I)=EIJ
C 1400 LOC=IABS(MAP(N))
C       IF(NDF.LT.6) GO TO 1600
C       DO 1500 J=1,6
C       DO 1500 I=1,6
C 1500 H(I,J,LOC)= H(I,J,LOC)+HKL(I,J)
C       GO TO 2000
C 1600 DO 1700 I=1,NDF
C       NROW=INEX(I)
C       DO 1700 J=1,NDF
C       NCOL=INEX(J)
C 1700 H(I,J,LOC)=H(I,J,LOC)+HKL(NROW,NCOL)
C 2000 CONTINUE
C
```

```
C       FORTRAN Code replacement - End
C
C      ******************************************
C
D       CALL GTIM(ITIM5)
D       WRITE(6,4000) ITEST
D       CALL CVTTIM(ITIM1,IHR,IMI,ISE,ITI)
D       WRITE(6,70) IHR,IMI,ISE,ITI
D       CALL CVTTIM(ITIM2,IHR,IMI,ISE,ITI)
D       WRITE(6,70) IHR,IMI,ISE,ITI
D       CALL CVTTIM(ITIM3,IHR,IMI,ISE,ITI)
D       WRITE(6,70) IHR,IMI,ISE,ITI
D       CALL CVTTIM(ITIM4,IHR,IMI,ISE,ITI)
D       WRITE(6,70) IHR,IMI,ISE,ITI
D       CALL CVTTIM(ITIM5,IHR,IMI,ISE,ITI)
D       WRITE(6,70) IHR,IMI,ISE,ITI
  3000 CONTINUE
       CALL GTIM(ITIM6)
       CPU2= SECNDS(0.)
       WRITE(6,50) CPU1
       WRITE(6,50) CPU2
       CALL CVTTIM(ITIM7,IHR,IMI,ISE,ITI)
       WRITE(6,70) IHR,IMI,ISE,ITI
  4000 FORMAT(' CYCLE = ',I5)
    70 FORMAT(' TIME = ',I2,':',I2,':',I2,':',I2)
       CALL CVTTIM(ITIM6,IHR,IMI,ISE,ITI)
       WRITE(6,70) IHR,IMI,ISE,ITI
       CPU = CPU2 - CPU1
       WRITE(6,99)CPU
    99 FORMAT('  EXECUTION TIME =',F12.7,'  SECONDS')
       WRITE(6,88) (H(I,1,1),I=1,36)
    88 FORMAT(E16.7)
       STOP
       END
```

```
"MODIFIED 29-MAY-80 TO USE M.D. INSTEAD OF TMRAM FOR ADDRESS
"TABLE
"
                    $TITLE TRAN6
                 $ENTRY APTRN6,4
"
"
" THIS ROUTINE PERFORMS THE TRAN6 SUBROUTINE LOCATED IN THE
" SPAR LIBRARY. IT IS CALLED EXTENSIVELY IN THE K AND M PROCESSORS.
"
" AUTHOR: K. FESON
" DATE: MAY 1979
"
" ----USUAGE----
"
" FORTRAN: CALL APTRN6(TBASE,NDF,NNODES,MAPBAS)
"
"        $PAGE
"
"
" ----MAIN DATA MEMORY MAP----
"
"  *******************************      (STARTING ADDRESS)
"  *                             *
"  *                             *
"  *       T ARRAY OR B3(LE)     * <-- LR (ADDRESS RELATIVE TO BASE)
"  *                             *
"  *                             *
"  *******************************   442+NDF*NDF*KSIZE+LREC5
"  *                             *
"  *                             *
"  *     K4(LREC5) OR MAP        * <-- L' (ADDRESS RELATIVE TO BASE)
"  *                             *
"  *                             *
"  *******************************  442+NDF*NDF*KSIZE
"  *                             *
"  *     H(NDF*NDF*KSIZE)        *
"  *                             *
"  *******************************  442
"  *                             *
"  *     SLOC(360)               *
"  *                             *
"  *******************************  82
"  *                             *
"  *     GKLTL (36)              *
"  *                             *
"  *******************************  46
"  *                             *
"  *     HKL(36)                 *
"  *                             *
```

```
"  ********************************  10
"  *                              *
"  *      ITRANS(4)               *
"  *                              *
"  ********************************  6
"  *                              *
"  *      INEX(6)                 *
"  *                              *
"  ********************************  0
"
"
"
"
"
"
" ----ARRAY DESCRIPTIONS----
"
" INEX(6). ........ INTEGER ARRAY. TRANSFERED ONLY ONCE
"
" ITRANS(4)........ INTEGER ARRAY. TRANSFERED ONLY ONCE.
"
" HKL(36)........... REAL ARRAY. NEVER TRANSMITTED.
"
" GKLTL(36)......... REAL ARRAY. NEVER TRANSMITTED.
"
" SLOC(360)......... REAL ARRAY. TRANSFERED  ONCE FOR EVERY CALL TO
"                    THE ROUTINE.
"
" H(NDF,NDF,KSIZE).. REAL ARRAY. TRANSFERED AND RETURNED FOR EVERY
"                    CALL TO THE ROUTINE.
"
" K4(LREC5) OR MAP.. INTEGER ARRAY. TRANSFERED ONLY AFTER BEING READ
"                    FROM THE DATA BASE.
"
" B3(LE) OR T(3,3,_) REAL ARRAY. TRANSFERED ONLY AFTER BEING READ
"                    FROM THE DATA BASE.
"
"
"
"
" ---TABLE MEMORY USAGE---
"
" A TABLE OF INDIRECT ADDRESS OFFSETS MUST BE LOADED AT
" TMRAM LOCATION 4096 (DECIMAL).
" THE TABLE IS USED TO CALCULATE AN ADDRESS OFFSET FOR
" AN ARRAY OF THE FORM X(3,3,LOC). THE VALUE 'LOC' IS
" ADDED TO 4096 TO FIND THE ADDRESS OFFSET IN THE
" TABLE. FOR EXAMPLE IF LOC=3, THE TABLE LOCATION
" 4096+3 OR 4099 IS READ AND A VALUE OF 18 IS LOADED
" AS THE ADDRESS OFFSET FOR ARRAY X(3,3,3). THIS
```

```
" REPRESENTS THE ADDRESS OF X(1,1,3) WHEN ADDED TO THE
" BASE ADDRESS OF X(1,1,1) IN THE MAIN DATA MEMORY.
"
"
"        4096   --     0
"        4097   --     0
"        4098   --     9
"        4099   --    18
"        4100   --    27
"        4101   --    36
"        4102   --    45
"        4103   --    54
         4104   --    63
"        4105   --    72
"        4106   --    81
"        4107   --    90
"        4108   --    99
"        4109   --   108
"        4110   --   109
"        4111  --    117
"        4112  --    126
"           .          .
"           .          .
"           .          .
"
"    TABLE ADDRESS    OFFSET ADDRESS
"
"
"   ( THE TABLE SHOULD CONTAIN SEVERAL HUNDRED ENTRIES)
"
"
"
" ---TABLE MEMORY USAGE---
"
" ANOTHER TABLE OF INDIRECT ADDRESS MUST BE LOADED AT
" TMRAM LOCATION 4196 (DECIMAL).
" THE TABLE IS USED TO LOCATE AN ADDRESS OFFSET FOR AN
" ARRAY OF THE FORM X(6,NCOL). THE VALUE OF 'NCOL' IS
" ADDED TO 4196 TO FIND THE TABLE MEMORY LOCATION OF
" THE ADDRESS OFFSET
"
"
"        4196  --   0
"        4197  --   6
"        4198  --  12
"        4199  --  18
"        4200  --  24
"        4201  --  30
"        4202  --  36
"        4203  --  42
```

```
"         4204  --  48
"         4205  --  54
"         4206  --  60
"         4207  --  66
"           .   --  .
"           .   --  .
"           .   --  .
"
"
"    TABLE ADDRESS  ADDRESS OFFSET
"
"    (THE TABLE SHOULD CONTAIN SEVERAL HUNDRED ENTRIES)
"
"
"
" S-PAD PARAMETERS
"
                    TBASE  $EQU 0     "BASE ADDRESS OF T ARRAY
                    NDF    $EQU 1     "DEGREES OF FREEDOM PER JOINT
                    NNODES $EQU 2     "NUMBER OF NODES IN ELEMENT
                    T2INC  $EQU 3     "OFFSET ADDRESS TO LOCATE T(1,1,KK)
                    THTSIX $EQU 3     "CONSTANT OF 36
                    TWNTY1 $EQU 3     "CONSTANT OF 21
                    EIGHTN $EQU 3     "CONSTANT OF 18
                    MAPBAS $EQU 3     "BASE ADDRESS OF MAP ARRAY (PASSED AS ARGUM
                    NROW   $EQU 3     "LOOP COUNTER
                    GKLTLB $EQU 4     "BASE ADDRESS OF GKLTL ARRAY
                    BASEGK $EQU 4     "BASE ADDRESS OF GKLTL ARRAY
                    HBASE  $EQU 4     "BASE ADDRESS OF H ARRAY
                    HKINC  $EQU 4     "ADDRESS POINTER OF H ARRAY
                    THREE  $EQU 5     "CONSTANT OF 3
                    HKADDR $EQU 5     "ADDRESS POINTER OF HKL ARRAY
                    N      $EQU 6     "INDEX FOR MAP ARRAY
                    JCNT   $EQU 6     "LOOP COUNTER
                    J      $EQU 6     "LOOP COUNTER
                    GKLTLA $EQU 7     "ADDRESS POINTER FOR GKLTL ARRAY
                    GKLTL1 $EQU 7     "ADDRESS POINTER FOR GKLTL(1,J) ARRAY
                    HKL1   $EQU 7     "ADDRESS POINTER FOR HKL(I,J)
                    HADDR1 $EQU 7     "ADDRESS POINTER FOR H ARRAY
                    S1ADDR $EQU 10    "ADDRESS POINTER FOR S(I,M,N) ARRAY
                    GKLTL2 $EQU 10    "ADDRESS POINTER FOR GKLTL(2,J)
                    HKL2   $EQU 10    "ADDRESS POINTER FOR HKL(J,I) ARRAY
                    HADDR2 $EQU 10    "ADDRESS POINTER FOR H(I,J,LOC)
                    TMA    $EQU 11    "TABLE ADDRESS POINTER
                    S2ADDR $EQU 11    "ADDRESS POINTER FOR S(I,M+1,N)
                    GKLTL3 $EQU 11    "ADDRESS POINTER FOR GKLTL(3,J)
                    HKTMP  $EQU 11    "ADDRESS POINTER FOR HKL ARRAY
                    ITRAN  $EQU 12    "BASE ADDRESS OF ITRAN ARRAY
                    LL     $EQU 12    "INDICY FOR T ARRAY = ITRAN(L)
                    KK     $EQU 12    "INDICY FOR T ARRAY = ITRAN(K)
```

```
                  S3ADDR $EQU 12     "ADDRESS POINTER FOR S(I,M+2,N)
                  HKL    $EQU 12     "ADDRESS POINTER FOR HKL ARRAY
                  HADDR  $EQU 12     "ADDRESS POINTER FOR H ARRAY
                  K      $EQU 13     "INNER LOOP COUNTER FOR 2000 LOOP
                  ICNT   $EQU 13     "LOOP COUNTER
                  JM     $EQU 13     "LOOP COUNTER
                  CNT    $EQU 13     "LOOP COUNTER
                  ZERO   $EQU 13     "CONSTANT OF 0
                  I      $EQU 13     "LOOP COUNTER
                  L      $EQU 14     "OUTER LOOP COUNTER FOR 2000 LOOP
                  DPA    $EQU 14     "DATA PAD REGISTERS BASE
                  LOC    $EQU 14     "TEMPORARY STORAGE OF MAP(N)
                  INEX   $EQU 14     "ADDRESS POINTER FOR INEX ARRAY
                  SIX    $EQU 15     "CONSTANT OF 6
                  T1INC  $EQU 16     "OFFSET ADDRESS TO LOCATE T(1,1,KK) BASE
                  SADDR  $EQU 16     "TEMPORARY STORAGE OF S BASE
                  N27    $EQU 16     "CONSTANT OF 27
                  NCOL   $EQU 16     "LOOP COUNTER
                  MAP    $EQU 16     "ADDRESS OFFSET FOR MAP ARRAY
                  SBASE  $EQU 17     "BASE ADDRESS OF S ARRAY
                  BASEHK $EQU 17     "BASE ADDRESS OF HKL ARRAY
                  TADR1 $EQU 15000.     "ADDRESS OF ADDRESS TABLE 1
                  TADR2 $EQU 15500. "ADDRESS OF ADDRESS TABLE 2
"
"
"
" FORTRAN: N=0
"          DO 2000 L=1,NNODES
"          LL=ITRANS(L)
"
"
"
APTRN6:  LDDPA; DB=12.                             "GET HIGH DATA PADS
         CLR N                                    "SET N=0
         DEC N; DPX(2)<SPFN                        "INIT N TO -1
         CLR L; DPX(0)<SPFN                        "INIT L
         MOV MAPBAS,MAPBAS; DPY(0)<SPFN            "STORE BASE OF MAP ARRAY
         LDSPI SBASE; DB=46.                       "LOAD SBASE - 36
         MOV SBASE,SBASE; DPX(3)<SPFN              "SAVE IN HIGH DATA PAD
"
"
LOOP1:   LDSPI L; DB=DPX(0)                        "RESTORE L
         LDSPI ITRAN; DB=6.                       "LOAD BASE OF ITRAN ARRAY
         ADD# L,ITRAN; SETMA                       "GET LL=ITRAN(L)
         LDSPI TMA; DB=TADR1                        "GET BASE OF TABLE
         CLR K; DPX(1)<SPFN                           "RESET INNER LOOP COUNT
         LDSPI LL; DB=MD                           "SAVE LL
         ADD# LL,TMA; SETMA                       "FIND ADDRESS OFFSET IN TABLE
         INC L; DPX(0)<SPFN                        "SAVE L+1 IN HIGH DATA PAD
        NOP
```

```
        LDSPI T2INC;DB=MD                      "SAVE ADDRESS OFFSET
         MOV T2INC,T2INC; DPY(2)<SPFN           "SAVE IN HIGH DATA PAD
"
"
"
" FORTRAN: DO 2000 K=1,L
"          KK=ITRANS(K)
"
"          ( N=N+1 IS PERFORMED LATER IN THE CODE )
"

"
LOOP2:   LDSPI ITRAN; DB=6.                    "LOAD BASE OF ITRAN ARRAY
         LDSPI K; DB=DPX(1)                     "RESTORE K
         ADD# K,ITRAN; SETMA                    "GET KK=ITRAN(K)
         LDSPI TMA; DB=TADR1                     "GET BASE OF TABLE
        NOP                                    "WAIT FOR M.D.
         LDSPI KK; DB=MD                        "SAVE KK
         ADD# KK,TMA; SETMA                     "FIND ADDRESS OFFSET IN TABLE
         INC K; DPX(1)<SPFN                     "SAVE K+1 IN HIGH DATA PAD
        NOP                                    "WAIT FOR M.D.
         LDSPI T1INC; DB=MD                     "STORE ADDRESS OFFSET
         MOV T1INC,T1INC; DPY(1)<SPFN           "SAVE IN HIGH DATA PAD
"
"
"
" GET TLMJ=T(M,J,LL)
"
" TBASE+T2INC GIVES THE ADDRESS OF T(1,1,LL)
" FOUND IN THE PREVIOUS SECTION
"
" THE NINE VALUES OF T FOR M=1,3 AND J=1,3
" ARE STORED IN THE DATA PAD X REGISTERS
"
"
         LDSPI T2INC; DB=DPY(2)                  "RESTORE T2INC
         LDSPI THREE; DB=3.                       "LOAD CONSTANT
         ADD TBASE,T2INC; SETMA                    "GET T(1,1,LL)
         CLR DPA; SETDPA                                 "CLEAR DATA PAD BASE AD
         INC T2INC; SETMA                        "GET T(2,1,LL)
         DPX(0)<DB; DB=MD                        "SAVE T(1,1,LL)
         INC T2INC; SETMA                        "GET T(3,1,LL)
         DPX(1)<DB; DB=MD                        "SAVE T(2,1,LL)
         INC T2INC; SETMA                        "GET T(1,2,LL)
         DPX(2)<DB; DB=MD;                       "SAVE T(3,1,LL)
                    ADD THREE,DPA; SETDPA          "GET NEXT SET OF DATA PADS
         INC T2INC; SETMA                        "GET T(2,2,LL)
         DPX(0)<DB; DB=MD                        "SAVE T(1,2,LL)
         INC T2INC; SETMA                        "GET T(2,3,LL)
         DPX(1)<DB; DB=MD                        "SAVE T(2,2,LL)
```

```
        INC T2INC; SETMA                               "GET T(1,3,LL)
        DPX(2)<DB; DB=MD;                              "SAVE T(3,2,LL)
                  ADD THREE,DPA; SETDPA                  "GET NEXT SET OF DATA PADS
        INC T2INC; SETMA                               "GET T(2,3,LL)
        DPX(0)<MD                                      "SAVE T(1,3,LL)
        INC T2INC; SETMA                                "GET T(3,3,LL)
        DPX(1)<MD                                      "SAVE T(2,3,LL)
        NOP
        DPX(2)<MD                                      "SAVE T(3,3,LL)
"
"
"
"
" PERFORM THE 1100 LOOP CALCULATIONS IN A SERIES OF STEPS
"
" BECAUSE THE INDIVIDUAL CALCULATIONS ARE INDEPENDENT, EACH GIVEN
" CALCULATION IS PERFORMED IN A SEPERATE IDENTICAL ROUTINE
"
" THE INNER MOST LOOP ACTUALLY PERFORMS THE FORTRAN M LOOP INTERNALLY
" WITH THE CALCULATIONS
"
" THE FORTRAN I LOOP IS THE INNER LOOP FOR THE ROUTINE
"
" THE J LOOP IS OUTSIDE BOTH AS NORMAL
"
"
"
" FORTRAN: DO 100 J=1,6
"          DO 100 I=1,6
"          GKLTL(I,J)=.0
"      100 HKL(I,J)=.0
"          DO 1100 J=1,3
"          DO 1100 I=1,3
"          DO 1100 M=1,3
"          TLMJ=T(M,J,LL)
"          GKLTL(I,J) = GKLTL(I,J) + S(I,M,N)*TLMJ
"          GKLTL(I,J+3) = GKLTL(I,J+3) + S(I,M+3,N)*TLMJ
"          GKLTL(I+3,J) = GKLTL(I+3,J) + S(I+3,M,N)*TLMJ
"     1100 GKLTL(I+3,J+3) = GKLTL(I+3,J+3) + S(I+3,M+3,N)
"
" CALCULATE GKLTL(I,J) = GKLTL(I,J) + S(I,M,N)*TLMJ
"
"
        LDSPI SIX; DB=6.                                "LOAD CONSTANT
        CLR DPA; SETDPA                                        "CLEAR DATA PAD BASE
        MOV THREE,JCNT                                  "LOAD J LOOP COUNT
        LDSPI GKLTLB; DB=46.                           "LOAD BASE ADR OF GKLTL
        MOV GKLTLB,GKLTLA                               "LOAD BASE ADDRESS
        DEC GKLTLA                                      "LOOP SET-UP
        LDDPA; DB=12.                                   "GET HIGH DATA PADS
```

```
         LDSPI SBASE; DB=DPX(3)                          "RESTORE SBASE
         LDSPI THTSIX; DB=36.                            "LOAD CONSTANT
         ADD THTSIX,SBASE; DPX(3)<SPFN                   "GET S(I,M,N+1) BASE
         LDDPA; DB=0.                                    "GET LOW DATA PADS
         MOV SBASE,SADDR                                  "LOAD S(I,M,N) BASE
LOOP3:   MOV SADDR,S1ADDR; SETMA                           "LOAD S(I,M,N)
         MOV S1ADDR,S2ADDR
         ADD SIX,S2ADDR; SETMA                             "LOAD S(I,M+1,N)
         MOV S2ADDR,S3ADDR;
                  FMUL DPX(0),MD                           " S1*TLMJ(1,J,LL)
         ADD SIX,S3ADDR; SETMA;                            "LOAD S(I,M+2,N)
                  FMUL                                    "PUSH
                  FMUL DPX(1),MD                           "TLMJ(2,J,LL)*S(I,M+1,N)
         MOV THREE,ICNT;                                 "LOAD INNER COUNT
                  FMUL; DPY(0)<FM                          "STORE S1 PRODUCT
LOOP4:            FMUL DPX(2),MD                             " S(I,M+2,N)*TLMJ(3,J,LL)
         INC S1ADDR; SETMA;                               "GET NEXT S1
                  FADD FM,DPY(0); FMUL                       "S2 PRODUCT+ S1 PRODUCT
                  FADD; FMUL
         INC S2ADDR; SETMA;                               "GET NEXT S2
                  FADD FM,FA                              "S2+S1+S3 (PRODUCTS)
         FMUL DPX(0),MD;                                  "S1+1*TLMJ
                  FADD
         INC S3ADDR; SETMA; FMUL                           "GET NEXT S3
         FMUL DPX(1),MD;                                  "S2+1*TLMJ
                            DEC ICNT                       "CHECK LOOP
         FMUL; DPY(0)<FM;                                 "SAVE S1+1 PRODUCT
                  INC GKLTLA; SETMA;MI<FA;                   "STORE RESULT
                            BNE LOOP4
"
" THE INNER I AND M LOOPS ARE COMPLETE
" REAJUST AND CHECK THE OUTER J LOOP
"
"
         ADD THREE,DPA; SETDPA                             "GET NEXT DATA PAD UNITS
         DEC JCNT                                        "CHECK OUTER LOOP
         BEQ CONT1; ADD THREE,GKLTLA                       "REAJUST GKLTLA ADDRESS
         JMP LOOP3
"
"
" CALCULATE GKLTL(I+3,J) = GKLTL(I+3,J) + S(I+3,M,N)*TLMJ
"
" TLMJ IS STILL IN THE DATA PADS
"
"
CONT1:   CLR DPA; SETDPA                                        "GET FIRST SET OF DATA
         MOV THREE,JCNT                                   "LOAD COUNTER
         MOV GKLTLB,GKLTLA                                "LOAD BASE ADDRESS
         ADD THREE,GKLTLA                                 "GET GKLTL(I+3,J) ADDRESS
         DEC GKLTLA                                       "LOOP SET UP
```

```
        MOV SBASE,SADDR                                     "LOAD S BASE
        ADD THREE,SADDR                                      "GET S(I+3,M,N) ADDRESS
LOOP5:  MOV SADDR,S1ADDR; SETMA                              "LOAD S1 ADDRESS
        MOV S1ADDR,S2ADDR
        ADD SIX,S2ADDR; SETMA                                "LOAD S2 ADDRESS
        MOV S2ADDR,S3ADDR;
                  FMUL DPX(0),MD                             "DO S1*TMLJ
        ADD SIX,S3ADDR; SETMA;                               "GET S3 ADDRESS
                  FMUL
                  FMUL DPX(1),MD                             "DO S2*TLMJ
        MOV THREE,ICNT                                      "LOAD INNER COUNT
                  FMUL; DPY(0)<FM                            "STORE S1*TLMJ
LOOP6:            FMUL DPX(2),MD                             "DO S3*TLMJ
        INC S1ADDR; SETMA;                                  "GET NEXT S1 ELEMENT
                  FADD FM,DPY(0); FMUL                        "S2 PRODUCT + S1 PRODUCT
                  FADD; FMUL
        INC S2ADDR; SETMA;                                  "GET NEXT S2 ELEMENT
                  FADD FM,FA
        FMUL DPX(0),MD;                                     "S1+1*TLMJ
                  FADD
        INC S3ADDR; SETMA; FMUL                              "GET NEXT S3 ELEMENT
        FMUL DPX(1),MD;                                     "DO S2+1*TLMJ
                            DEC ICNT                         "CHECK LOOP
        FMUL; DPY(0)<FM;                                    "SAVE S1+1 PRODUCT
                  INC GKLTLA; SETMA;MI<FA;                    "STORE RESULT
                            BNE LOOP6
"
" RECHECK THE OUTER J LOOP
"
        ADD THREE,DPA; SETDPA                                "GET NEXT DATA PADS
        DEC JCNT                                            "CHECK OUTER LOOP
        BEQ CONT2; ADD THREE,GKLTLA                          "REAJUST GKLTLA ADDRESS
        JMP LOOP5
"
"
" CALCULATE GKLTL(I,J+3) = GKLTL(I,J+3) + S(I,M+3,N)*TLMJ
"
" TLMJ IS STILL IN DATA PADS
"
"
CONT2:  CLR DPA; SETDPA                                            "GET FIRST SET OF DATA
        MOV THREE,JCNT                                      "LOAD OUTER COUNT
        MOV GKLTLB,GKLTLA                                   "LOAD BASE ADDRESS
        LDSPI EIGHTN; DB=18.                                 "LOAD CONSTANT
        ADD EIGHTN,GKLTLA                                   "GET ADDRESS OF GKLTL(I,J+3)
        DEC GKLTLA                                          "LOOP SET UP
        MOV SBASE,SADDR                                     "LOAD S BASE ADDRESS
        ADD EIGHTN,SADDR                                    "GET S(I,M+3,N) AS BASE
LOOP7:  MOV SADDR,S1ADDR; SETMA                              "LOAD S1 WITH BASE
        MOV S1ADDR,S2ADDR
```

```
        ADD SIX,S2ADDR; SETMA                          "GET S2
        MOV S2ADDR,S3ADDR;
                  FMUL DPX(0),MD                       "DO S1*TLMJ
        ADD SIX,S3ADDR; SETMA;                         "GET S3
                  FMUL
                  FMUL DPX(1),MD                       "DO S2*TLMJ
        MOV THREE,ICNT;                               "LOAD INNER COUNT
                  FMUL; DPY(0)<FM                      "SAVE S1*TLMJ
LOOP8:            FMUL DPX(2),MD                       "DO S3*TLMJ
        INC S1ADDR; SETMA;                            "GET NEXT S1 ELEMENT
                  FADD FM,DPY(0); FMUL                  "S1 PRODUCT +S2 PRODUCT
                  FADD; FMUL
        INC S2ADDR; SETMA;                            "GET NEXT S2 ELEMENT
                  FADD FM,FA                           "S1+S2+S3 PRODUCTS
        FMUL DPX(0),MD;                               "S1+1*TLMJ
                  FADD
        INC S3ADDR; SETMA; FMUL                        "GET NEXT S3
        FMUL DPX(1),MD;                               "DO S2+1*TLMJ
                            DEC ICNT                   "TEST LOOP
        FMUL; DPY(0)<FM;                              "SAVE S1+1 PRODUCT
                  INC GKLTLA; SETMA;MI<FA;              "STORE RESULT
                            BNE LOOP8
"
" CHECK THE OUTER J LOOP
"
        ADD THREE,DPA; SETDPA                          "GET NEXT DATA PAD SET
        DEC JCNT                                     "CHECK LOOP
        BEQ CONT3; ADD THREE,GKLTLA                    "REAJUST ADDRESS
        JMP LOOP7
"
"
" CALCULATE GKLTL(I+3,J+3) = GKLTL(I+3,J+3) + S(I+3,M+3,N)*TLMJ
"
" TLMJ IS STILL IN DATA PADS
"
"
CONT3:  CLR DPA; SETDPA                                     "RESET DATA PAD ADDRESS
        MOV THREE,JCNT                                 "LOAD OUTER COUNT
        MOV GKLTLB,GKLTLA                              "LOAD BASE
        LDSPI TWNTY1; DB=21.                           "LOAD CONSTANT
        ADD TWNTY1,GKLTLA                             "GET GKLTL(I+3,J+3) ADDRESS
        DEC GKLTLA                                    "LOOP SETUP
        MOV SBASE,SADDR                               "LOAD BASE FOR S
        ADD TWNTY1,SADDR                              "GET S (I+3,M+3,N) ADDRESS
LOOP9:  MOV SADDR,S1ADDR; SETMA                        "GET S1
        MOV S1ADDR,S2ADDR
        ADD SIX,S2ADDR; SETMA                          "GET S2
        MOV S2ADDR,S3ADDR;
                  FMUL DPX(0),MD                       "S1*TLMJ
        ADD SIX,S3ADDR; SETMA;                         "GET S3
```

```
                FMUL
                FMUL DPX(1),MD                      "S2*TLMJ
        MOV THREE,ICNT;                             "LOAD INNER COUNT
                FMUL; DPY(0)<FM                     "SAVE S1*TLMJ
LOOP10:          FMUL DPX(2),MD                      "DO S3*TLMJ
        INC S1ADDR; SETMA;                          "GET NEXT S1 ELEMENT
                FADD FM,DPY(0); FMUL                  "S1 PRODUCT + S2 PRODUCT
                FADD; FMUL
        INC S2ADDR; SETMA;                          "GET NEXT S2 ELEMENT
                FADD FM,FA                           "ADD S1 + S2 +S3 PRODUCTS
        FMUL DPX(0),MD;                             "DO S1+1*TLMJ
                FADD
        INC S3ADDR; SETMA; FMUL                      "GET NEXT S3 ELEMENT
        FMUL DPX(1),MD;                             "DO S2+1*TLMJ
                        DEC ICNT                     "CHECK INNER COUNT
        FMUL; DPY(0)<FM;                            "SAVE S1+1 PRODUCT
                INC GKLTLA; SETMA;MI<FA;              "STORE RESULT
                        BNE LOOP10
"
" CHECK THE OUTER J LOOP
"
        ADD THREE,DPA; SETDPA                        "GET NEXT DATA PADS
        DEC JCNT                                    "CHECK LOOP
        BEQ CONT4; ADD THREE,GKLTLA
        JMP LOOP9
"
" THIS ROUTINE PERFORMS THE 1200 LOOP CALCULATIONS
"
"
" GET TLMJ=T(M,J,KK)
"
" TBASE+T1INC GIVES THE ADDRESS OF T(1,1,KK)
" FOUND IN THE PREVIOUS SECTION
"
" THE NINE VALUES OF T FOR M=1,3 AND J=1,3
" ARE STORED IN THE DATA PAD X REGISTERS
"
" EACH CALCULATION IS PERFORMED INDEPENDENTLY IN A SEPERATE
" AND INDENTICAL ROUTINE JUST LIKE THE 1100 LOOP.
"
"
" FORTRAN: DO 1200 J=1,3
"          DO 1200 I=1,3
"          DO 1200 M=1,3
"          TKMI=T(M,I,KK)
"          HKL(I,J)   =  HKL(I,J) +TKMI*GKLTL(M,J)
"          HKL(I,J+3) = HKL(I,J+3) +TKMI*GKLTL(M,J+3)
"          HKL(I+3,J) = HKL(I+3,J) + TKMI*GKLTL(M+3,J)
"     1200 HKL(I+3,J+3) = HKL(I+3,J+3) + TKMI* GKLTL(M+3,J+3)
"
```

```
"
"
CONT4:   LDDPA; DB=12.                          "GET HIGH DATA PADS
         LDSPI T1INC; DB=DPY(1)                 "RESTORE T1INC
         ADD TBASE,T1INC; SETMA                 "GET T(1,1,KK)
          CLR DPA; SETDPA                                "CLEAR DATA PAD BASE AD
           INC T1INC; SETMA                      "GET T(2,1,KK)
         DPX(0)<DB; DB=MD                       "SAVE T(1,1,KK)
         INC T1INC; SETMA                       "GET T(3,1,KK)
         DPX(1)<DB; DB=MD                       "SAVE T(2,1,KK)
         INC T1INC; SETMA                       "GET T(1,2,KK)
         DPX(2)<DB; DB=MD;                      "SAVE T(3,1,KK)
                 ADD THREE,DPA; SETDPA           "GET NEXT SET OF DATA PADS
         INC T1INC; SETMA                       "GET T(2,2,KK)
         DPX(0)<DB; DB=MD                       "SAVE T(1,2,KK)
         INC T1INC; SETMA                       "GET T(2,3,KK)
         DPX(1)<DB; DB=MD                       "SAVE T(2,2,KK)
         INC T1INC; SETMA                       "GET T(1,3,KK)
         DPX(2)<DB; DB=MD;                      "SAVE T(3,2,KK)
                 ADD THREE,DPA; SETDPA           "GET NEXT SET OF DATA PADS
         INC T1INC; SETMA                       "GET T(2,3,KK)
         DPX(0)<MD                              "SAVE T(1,3,KK)
         INC T1INC; SETMA                        "GET T(3,3,KK)
         DPX(1)<MD                              "SAVE T(2,3,KK)
         NOP
         DPX(2)<MD                              "SAVE T(3,3,KK)
"
"
" CALCULATE HKL(I,J) = HKL(I,J) + TKMI*GKLTL(M,J)
"
"
"
         LDSPI SIX; DB=6.                       "LOAD CONSTANT OF 6
         LDSPI THREE; DB=3.                     "LOAD CONSTANT OF 3
         LDSPI BASEGK; DB=46.                   "LOAD GKLTL BASE
         MOV BASEGK,GKLTL1                      "LOAD GKLTL(M,J) ADDRESS
         LDSPI BASEHK; DB=10.                   "LOAD BASE OF HKL
         MOV BASEHK,HKL                         "LOAD BASE ADDRESS OF HK
         DEC HKL                                "LOOP SET UP
         MOV THREE,JCNT                         "LOAD OUTER COUNTER
"
" BEGIN THE J LOOP CALCULATIONS
"
LOOP11:   CLR DPA; SETDPA                                "GET FIRST SET OF DATA
         MOV GKLTL1,GKLTL2; SETMA                "GET GKLTL(M,J)
         INC GKLTL2                             "GET GKLTL(2,J) ADDRESS
         MOV GKLTL2,GKLTL3; SETMA                "GET GKLTL(2,J)
         FMUL DPX(0),MD                         "TKMI(1,J,LL)*GKLTL1
         INC GKLTL3; SETMA; FMUL                 "GET GKLTL(3,J)
         FMUL DPX(1),MD;                        "TKMI(2,J,LL)*GKLTL2
```

```
                MOV THREE,ICNT                    "LOAD INNER COUNT
        DPY(0)<FM; FMUL                           "SAVE GKLTL1 PRODUCT
        FMUL DPX(2),MD                            "TKMI(3,J,LL)*GKLTL3
        FADD FM,DPY(0); FMUL                      "GKLTL1 PRODUCT + GKLTL2 PROD
LOP11A:  MOV GKLTL1,GKLTL1; SETMA;                 "GET GKLTL(1,J)
                FADD; FMUL                        "PUSH
                FADD FM,FA;                        "GKLTL 1+2+3 PRODUCTS
                        ADD THREE,DPA; SETDPA            "GET NEXT DATA PADS
        MOV GKLTL2,GKLTL2; SETMA;                 "GET GKLTL(2,J)
                FADD                              "PUSH
        FMUL DPX(0),MD                            "GKLTL(1,J)*TKMI(1,J,LL)
        MOV GKLTL3,GKLTL3; SETMA;                 "GET GKLTL(3,J)
        FMUL                                      "PUSH
        FMUL DPX(1),MD                            "GKLTL(2,J)*TKMI(2,J,LL)
        DPY(0)<FM; FMUL                           "SAVE GKLTL1 PRODUCT
        FMUL DPX(2),MD;                           "GKLTL3*TKMI(3,J,LL)
                        DEC ICNT                  "CHECK INNER LOOP
        FADD FM,DPY(0); FMUL;                     "GKLTL 1+2 PRODUCT
                INC HKL; SETMA; MI<FA;              "SAVE RESULT
                        BNE LOP11A
"
"
" CHECK THE OUTER LOOP AND REAJUST THE ADDRESSES
"
"
        ADD SIX,GKLTL1                            "GET GKLTL(M,J+1)
        ADD THREE,HKL                             "GET HKL(I,J+1)
        DEC JCNT                                 "CHECK OUTER LOOP
        BEQ CONT4A                                 "IF DONE, CONTINUE
        JMP LOOP11                                "IF NOT, JUMP BACK
"
"
"
" PERFORM THE 1200 LOOP CALCULATIONS FOR THE SECOND EQUATION
"
"     HKL(I+3,J) = HKL(I+3,J) + TKMI*GKLTL(M+3,J)
"
"
CONT4A:  MOV BASEHK,HKL                            "LOAD BASE ADDRESS
        MOV BASEGK,GKLTL1                          "LOAD BASE ADDRESS
        ADD THREE,HKL                             "GET HKL(I+3,J) ADDRESS
        ADD THREE,GKLTL1                          "GET GKLTL(M+3,J) ADDRESS
        MOV THREE,JCNT                            "LOAD OUTER COUNTER
        DEC HKL                                  "LOOP SET UP
"
" BEGIN THE J LOOP CALCULATIONS
"
LOP11B:  CLR DPA; SETDPA                                  "GET FIRST SET OF DATA
        MOV GKLTL1,GKLTL2; SETMA                   "GET GKLTL(M,J)
        INC GKLTL2                                "GET GKLTL(5,J) ADDRESS
```

```
        MOV GKLTL2,GKLTL3; SETMA                        "GET GKLTL(5,J)
        FMUL DPX(0),MD                                  "TKMI(1,J,LL)*GKLTL1
        INC GKLTL3; SETMA; FMUL                         "GET GKLTL(6,J)
        FMUL DPX(1),MD;                                 "TKMI(2,J,LL)*GKLTL2
                 MOV THREE,ICNT                         "LOAD INNER COUNT
        DPY(0)<FM; FMUL                                 "SAVE GKLTL1 PRODUCT
        FMUL DPX(2),MD                                  "TKMI(3,J,LL)*GKLTL3
        FADD FM,DPY(0); FMUL                            "GKLTL1 PRODUCT + GKLTL2 PROD
LOP11C:  MOV GKLTL1,GKLTL1; SETMA;                      "GET GKLTL(4,J)
                 FADD; FMUL                             "PUSH
                 FADD FM,FA;                            "GKLTL 1+2+3 PRODUCTS
                          ADD THREE,DPA; SETDPA         "GET NEXT DATA PADS
        MOV GKLTL2,GKLTL2; SETMA;                       "GET GKLTL(5,J)
                 FADD                                   "PUSH
        FMUL DPX(0),MD                                  "GKLTL(4,J)*TKMI(1,J,LL)
        MOV GKLTL3,GKLTL3; SETMA;                       "GET GKLTL(6,J)
        FMUL                                            "PUSH
        FMUL DPX(1),MD                                  "GKLTL(5,J)*TKMI(2,J,LL)
        DPY(0)<FM; FMUL                                 "SAVE GKLTL1 PRODUCT
        FMUL DPX(2),MD;                                 "GKLTL3*TKMI(3,J,LL)
                          DEC ICNT                      "CHECK INNER LOOP
        FADD FM,DPY(0); FMUL;                           "GKLTL 1+2 PRODUCT
                 INC HKL; SETMA; MI<FA;                 "SAVE RESULT
                          BNE LOP11C
"
"
" CHECK THE OUTER LOOP AND REAJUST THE ADDRESSES
"
"
        ADD SIX,GKLTL1                                  "GET GKLTL(M,J+1)
        ADD THREE,HKL                                   "GET HKL(I,J+1)
        DEC JCNT                                        "CHECK OUTER LOOP
        BEQ CONT4B                                      "IF DONE, CONTINUE
        JMP LOP11B                                      "IF NOT, JUMP BACK
"
" PERFORM THE 1200 LOOP CALCULATIONS FOR THE THIRD EQUATION
"
"     HKL(I,J+3) = HKL(I,J+3) + TKMI*GKLTL(M,J+3)
"
"
CONT4B:  LDSPI EIGHTN; DB=18.                           "LOAD CONSTANT
        MOV BASEHK,HKL                                  "LOAD BASE ADDRESS
        MOV BASEGK,GKLTL1                               "LOAD BASE ADDRESS
        ADD EIGHTN,HKL                                  "GET HKL(I,J+3) ADDRESS
        ADD EIGHTN,GKLTL1                               "GET GKLTL(M,J+3) ADDRESS
        DEC HKL                                         "LOOP SET UP
        MOV THREE,JCNT                                  "LOAD OUTER COUNTER
"
" BEGIN THE J LOOP CALCULATIONS
"
```

```
LOP11D    CLR DPA; SETDPA                                  "GET FIRST SET OF DATA
          MOV GKLTL1,GKLTL2; SETMA                       "GET GKLTL(M,J+3)
          INC GKLTL2                                     "GET GKLTL(2,J+3) ADDRESS
          MOV GKLTL2,GKLTL3; SETMA                       "GET GKLTL(2,J+3)

          FMUL DPX(0),MD                                 "TKMI(1,J,LL)*GKLTL1
          INC GKLTL3; SETMA; FMUL                        "GET GKLTL(3,J+3)
          FMUL DPX(1),MD;                                "TKMI(2,J,LL)*GKLTL2
                    MOV THREE,ICNT                       "LOAD INNER COUNT
          DPY(0)<FM; FMUL                                "SAVE GKLTL1 PRODUCT
          FMUL DPX(2),MD                                 "TKMI(3,J,LL)*GKLTL3
          FADD FM,DPY(0); FMUL                           "GKLTL1 PRODUCT + GKLTL2 PROD
LOP11E:   MOV GKLTL1,GKLTL1; SETMA;                      "GET GKLTL(1,J+3)
                    FADD; FMUL                           "PUSH
                    FADD FM,FA;                          "GKLTL 1+2+3 PRODUCTS
                              ADD THREE,DPA; SETDPA      "GET NEXT DATA PADS
          MOV GKLTL2,GKLTL2; SETMA;                      "GET GKLTL(2,J+3)
                    FADD                                 "PUSH
          FMUL DPX(0),MD                                 "GKLTL(1,J+3)*TKMI(1,J,LL)
          MOV GKLTL3,GKLTL3; SETMA;                      "GET GKLTL(3,J+3)
          FMUL                                           "PUSH
          FMUL DPX(1),MD                                 "GKLTL(2,J+3)*TKMI(2,J,LL)
          DPY(0)<FM; FMUL                                "SAVE GKLTL1 PRODUCT
          FMUL DPX(2),MD;                                "GKLTL3*TKMI(3,J,LL)
                              DEC ICNT                   "CHECK INNER LOOP
          FADD FM,DPY(0); FMUL;                          "GKLTL 1+2 PRODUCT
                    INC HKL; SETMA; MI<FA;               "SAVE RESULT
                              BNE LOP11E
"
"
" CHECK THE OUTER LOOP AND REAJUST THE ADDRESSES
"
"
          ADD SIX,GKLTL1                                 "GET GKLTL(M,J+1)
          ADD THREE,HKL                                  "GET HKL(I,J+1)
          DEC JCNT                                       "CHECK OUTER LOOP
          BEQ CONT4C                                     "IF DONE, CONTINUE
          JMP LOP11D                                     "IF NOT, JUMP BACK
"
" PERFORM THE 1200 LOOP CALCULATIONS FOR THE FOURTH EQUATION
"
"     HKL(I+3,J+3) = HKL(I+3,J+3) + TKMI*GKLTL(M+3,J+3)
"
"
CONT4C:   LDSPI TWNTY1; DB=21.                           "LOAD CONSTANT
          MOV BASEHK,HKL                                 "LOAD BASE ADDRESS
          MOV BASEGK,GKLTL1                              "LOAD BASE OF GKLTL1
          ADD TWNTY1,HKL                                 "GET HKL(I+3,J+3) ADDRESS
          ADD TWNTY1,GKLTL1                              "GET GKLTL(M+3,J+3) ADDRESS
          DEC HKL                                        "LOOP SET UP
```

```
        MOV THREE,JCNT                                  "LOAD OUTER COUNTER
"
" BEGIN THE J LOOP CALCULATIONS
"
LOP11F:  CLR DPA; SETDPA                                      "GET FIRST SET OF DATA
        MOV GKLTL1,GKLTL2; SETMA                        "GET GKLTL(M+3,J+3)
        INC GKLTL2                                      "GET GKLTL(5,J+3) ADDRESS
        MOV GKLTL2,GKLTL3; SETMA                        "GET GKLTL(5,J+3)
        FMUL DPX(0),MD                                  "TKMI(1,J,LL)*GKLTL1
        INC GKLTL3; SETMA; FMUL                         "GET GKLTL(6,J+3)
        FMUL DPX(1),MD;                                 "TKMI(2,J,LL)*GKLTL2
                  MOV THREE,ICNT                        "LOAD INNER COUNT
        DPY(0)<FM; FMUL                                 "SAVE GKLTL1 PRODUCT
        FMUL DPX(2),MD                                  "TKMI(3,J,LL)*GKLTL3
        FADD FM,DPY(0); FMUL                            "GKLTL1 PRODUCT + GKLTL2 PROD
LOOP12:  MOV GKLTL1,GKLTL1; SETMA;                      "GET GKLTL(6,J+3+3)
                  FADD; FMUL                            "PUSH
                  FADD FM,FA;                           "GKLTL 1+2+3 PRODUCTS
                            ADD THREE,DPA; SETDPA            "GET NEXT DATA PADS
        MOV GKLTL2,GKLTL2; SETMA;                       "GET GKLTL(5,J+3)
                  FADD                                  "PUSH
        FMUL DPX(0),MD                                  "GKLTL(6,J+3+3)*TKMI(1,J,LL)
        MOV GKLTL3,GKLTL3; SETMA;                       "GET GKLTL(6,J+3)
        FMUL                                            "PUSH
        FMUL DPX(1),MD                                  "GKLTL(5,J+3)*TKMI(2,J,LL)
        DPY(0)<FM; FMUL                                 "SAVE GKLTL1 PRODUCT
        FMUL DPX(2),MD;                                 "GKLTL3*TKMI(3,J,LL)
                            DEC ICNT                    "CHECK INNER LOOP
        FADD FM,DPY(0); FMUL;                           "GKLTL 1+2 PRODUCT
                  INC HKL; SETMA; MI<FA,                "SAVE RESULT
                            BNE LOOP12
"
"
" CHECK THE OUTER LOOP AND REAJUST THE ADDRESSES
"
"
        ADD SIX,GKLTL1                                  "GET GKLTL(M,J+1)
        ADD THREE,HKL                                   "GET HKL(I,J+1)
        DEC JCNT                                        "CHECK OUTER LOOP
        BEQ CONT5                                       "IF DONE, CONTINUE
        JMP LOP11F                                      "IF NOT, JUMP BACK
"
"
"     FORTRAN: IF (MAP(N) .GT. 0) GOTO 1400
"
"
CONT5:  LDDPA; DB=12.                                   "GET HIGH DATA PADS
        LDSPI MAPBAS; DB=DPY(0)                         "RESTORE BASE OF MAP
        LDSPI N; DB=DPX(2)                              "RESTORE N
        INC N                                           "N=N+1
```

```
        ADD# N,MAPBAS; SETMA                         "GET MAP(N)
        MOV N,N; DPX(2)<SPFN                          "SAVE N
        LDPA; DB=0                                    "GET NORMAL DATA PADS
        LDSPI MAP; DB=MD                              "SAVE MAP(N)
        CLR ZERO                                      "SET ZERO=0
        SUB MAP,ZERO                                  "TEST MAP(N) .GT.0
        BGT CONT6                                     "CONTINUE IF FALSE
        JMP CONT7                                      "OTHERWISE, GOTO 1400
"
"
"
"
" PERFORM THE 1300 LOOP CALCULATIONS
"
" FORTRAN: DO 1300 J=2,6
"          JM=J-1
"          DO 1300 I=1,JM
"          EIJ=HKL(I,J)
"          HKL(I,J)=HKL(J,I)
"     1300 HKL(J,I)=EIJ
"
"
CONT6:  MOV BASEHK,HKTMP                               "LOAD BASE
        LDSPI SIX; DB=6.                              "LOAD CONSTANT
        LDSPI J; DB=1.                                "LOAD OUTER COUNTER - 1
"
" BEGIN THE OUTER LOOP
"
LOOP13:  MOV J,JM                                     "LOAD THE INNER COUNTER
        MOV BASEHK,HKL2                               "LOAD BASE
        ADD SIX,HKTMP                                  "GET HKL(I,J) ADDRESS
        MOV HKTMP,HKL1                                "LOAD HKL(I,J)
        DEC HKL1                                     "LOOP SET UP
        ADD J,HKL2                                    "GET HKL(J,I) ADDRESS
        SUB SIX,HKL2                                  "LOOP SET UP
        INC J                                        "GET J
"
" PERFORM THE INNER LOOP
"
LOOP14:  NOP
        ADD SIX,HKL2; SETMA                             "GET HKL(J,I)
        NOP
        INC HKL1; SETMA                               "GET HKL(I,J)
        NOP
        MOV HKL1,HKL1; SETMA; MI<MD                    "STORE HKL(I,J)=HKL(J,I)
                DEC JM
        MOV HKL2,HKL2; SETMA; MI<MD;                   "HKL(J,I)=HKL(I,J)
                BNE LOOP14                             "LOOP DONE?
"
" TEST OUTER LOOP
```

```
"
        SUB# SIX,J                                  "TEST J=6
        BEQ CONT7                                   "IF EQUAL, END
        JMP LOOP13
"
" FORTRAN: 1400 LOC=IABS(MAP(N))
"
"
CONT7:  LDSPI SIX; DB=6                             "LOAD CONSTANT
        LDSPI N27; DB=27.                           "LOAD CONSTANT
        LDDPA; DB=12.                               "GET HIGH DATA PADS
        LDSPI N; DB=DPX(2)                          "RESTORE N
        LDSPI MAPBAS; DB=DPY(0)                     "RESTORE BASE OF MAP ARRAY
        ADD# N,MAPBAS; SETMA                        "GET MAP(N)
        LDDPA; DB=0.                                "GET NORMAL DATA PADS
        CLR ZERO                                "GET 0
        LDSPI LOC; DB=MD                          "LOC=MAP(N)
        MOV LOC,LOC                           "IS LOC NEGATIVE
        BGE CONT8                            "IF POSITIVE, CONTINUE
        SUB LOC,ZERO                             "GET -LOC
        MOV ZERO,LOC                             "STORE -LOC
"
" CALCULATE THE BASE ADDRESS OF H(I,J,LOC)
"
"
CONT8:  DEC LOC                                     "GET LOC -1
        MOV LOC,LOC; DPX(0)<SPFN                      "FLOAT LOC-1
        FADD ZERO,MDPX(0); MOV N27,N27
        FADD;
                  RPSF THYSIX
        DPY(0)<DB                                    "DPY(0)=36.
        FMUL DPY(0),FA                              "36*LOC-1
        FMUL
        FMUL
        DPX(0)<FM
        FIX DPX(0)                                   "GET INT(36*LOC-1)
        FADD
        DPX(0)<FA;                                   "STORE RESULT
        LDSPI HBASE; DB=442.                        "LOAD H BASE
        LDSPI HADDR; DB=DPX(0)                        "GET H(1,1,LOC) ADDRESS
"
" FORTRAN: IF(NDF .LT. 6) GOTO 1600
"
" THE ADDRESS OF H(1,1,LOC) HAS BEEN CALCULATED FOR USE
" IN EITHER THE 1500 OR 1700 LOOPS
"
"
        SUB# SIX,NDF                                "TEST FOR NDF .LT. 6
        BGE CONT9                                   "IF TRUE, CONTINUE
        JMP CONT10                                  "OTHERSWISE, GOTO 1600
```

```
"
"
" BEGIN THE 1500 LOOP FORTRAN
"
"
"
" FORTRAN: DO 1500 J=1,6
"          DO 1500 I=1,6
"    1500  H(I,J,LOC) = H(I,J,LOC) + HKL(I,J)
"          GOTO 2000
"
"
CONT9:   ADD HBASE,HADDR; SETMA                    "GET H(1,1,LOC)
         MOV HADDR,HADDR2                          "LOAD DESTINATION ADDRESS
         MOV BASEHK,HKL1; SETMA                    "GET HKL(1,1)
         DPX(0)<MD;                                "STORE H(1,1,LOC)
                   DEC HADDR2                      "LOOP SET UP
         LDSPI CNT; DB=36.
"
" PERFORM THE 1500 LOOP CALCULATIONS
"
LOOP15:            FADD DPX(0),MD                  "H(I,J,LOC)+HKL(I,J)
         INC HADDR; SETMA;                         "GET NEXT H(I,J,LOC)
                   FADD                            "PUSH
         INC HKL1; SETMA                           "GET NEXT H(I,J)
         DPX(0)<MD;                                "STORE H(I,J,LOC)
                             DEC CNT               "TEST LOOP
         INC HADDR2; SETMA; MI<FA;                 "SAVE RESULT
                             BNE LOOP15            "FINISHED?
"
"
         JMP CONT11                                "GOTO 2000
"
" PERFROM THE 1700 LOOP CALCULATIONS
"
" FORTRAN: 1600 DO 1700 I=1,NDF
"               NROW=INEX(I)
"               DO 1700 J=1,NDF
"               NCOL=INEX(J)
"          1700 H(I,J,LOC)=H(I,J,LOC)+HKL(NROW,NCOL)
"
"
CONT10:  LDSPI SIX; DB=6.                          "LOAD CONSTANT OF 6.
         CLR INEX                                  "GET INEX BASE ADDRESS
         CLR I                                     "CLEAR OUTER COUNTER
         ADD HBASE,HADDR                           "GET H(1,1,LOC) ADDRESS
         DEC HADDR                                 "GET H(1,1,LOC) ADDRESS -1
         LDSPI TMA; DB=TADR2                       "SET ROM BASE ADDRESS
"
" OUTER LOOP
```

```
"
LOOP16:   ADD# I,INEX; SETMA                    "GET INEX(I)
          MOV BASEHK,HKADDR                     "GET HKL(1,1) ADDRESS
          DEC HKADDR                            "LOOP SET UP
          LDSPI NROW; DB=MD                     "STORE NROW=INEX(I)
          ADD NROW,HKADDR                       "GET HKL(NROW,1) ADDRESS
          INC HADDR                             "GET H(I+1,J,LOC) ADDRESS
          MOV HADDR,HADDR1                      "LOAD H ADDRESS
"
"
" INNER LOOP SET UP
"
          CLR J                                 "CLEAR INNER COUNT
          ADD# J,INEX; SETMA                    "GET INEX(J)
          NOP
          MOV HADDR1,HADDR2; SETMA              "LOAD TARGET ADDRESS
          LDSPI NCOL; DB=MD                     "NCOL=INEX(J)
          ADD# NCOL,TMA; SETMA                  "GET INCREMENT FROM ROM
          DPX(0)<MD;                            "STORE H(NCOL,1)
                    SUB SIX,HADDR2              "LOOP SET UP
          NOP                                   "WAIT FOR M.D.
          LDSPI HKINC; DB=MD                    "STORE TABLE INCREMENT
          ADD# HKINC,HKADDR; SETMA              "GET HKL(NROW,NCOL)
          INC I                                 "INC OUTER LOOP COUNT
          NOP
"
"
" INNER LOOP CALCULATIONS
"
"
LOOP17:   INC J;                                "INCREMENT INNER LOOP COUNT
                    FADD DPX(0),MD              "H(I,J,LOC)+HKL(NROW,NCOL)
          ADD# J,INEX; SETMA;                   "GET INEX(J)
                    FADD                        "PUSH
          NOP
          ADD SIX,HADDR1; SETMA                 "GET H(I,J+1,LOC)
          LDSPI NCOL; DB=MD                     "STORE NCOL=INEX(J)
          ADD# NCOL,TMA; SETMA                  "GET INCREMENT IN ROM TABLE
          NOP
          DPX(0)<MD                             "SAVE H(I,J+1,LOC)
          LDSPI HKINC; DB=MD                    "SAVE INCREMENT
          ADD# HKINC,HKADDR; SETMA              "GET HKL(NROW,NCOL)
                    SUB# NDF,J                  "TEST INNER LOOP
          ADD SIX,HADDR2; SETMA; MI<FA;         "STORE RESULT
                    BNE LOOP17
"
"
          SUB# NDF,I                            "TEST OUTER LOOP
          BEQ CONT11                            "IF DONE, CONTINUE
          JMP LOOP16                            "IF NOT, GBRANCH BACK
```

```
"
"
"
"
"
CONT11:   LOOPA; DB=12.                          "GET HIGH DATA PADS
         LDSPI L; DB=DPX(0)                      "RESTORE L
         LDSPI K; DB=DPX(1)                      "RESTORE K
         SUB# K,L                                "IS INNER K LOOP DONE
         BEQ CONT12                              "IF YES, CONTINUE
         JMP LOOP2                               "OTHERWISE, BRANCH BACK
CONT12:   SUB# L,NNODES                          "TEST OUTER MOST 2000 LOOP
         BEQ CONT13                                 "IF FINISHED, END
         JMP LOOP1                               "OTHERWISE, BRANCH BACK
THYSIX:   $FP 36.
CONT13:     RETURN
           $END
```

APPENDIX J

Listings of:

FUSEL

LUT

```
[XQT TAB
START 80, 5$ ROTATIONS ABOUT  Y EXCLUDED
TITLE" FUSELAGE MODEL,FSPAR1
TEXT
" MEMBRANE-ROD-BEAM FUSELAGE MODEL
"NONREPEATABLE PART
JLOC$ FUSELAGE DIA. 800. CM.,LENGTH=800. CM.
FORMAT=2$CYLINDRICAL COORDINATES
1 400. 0. 0. 400. 337.5 0. 16 1 5
16 400. 0. 800. 400. 337.5 800.
MREF
FORMAT=2
1 -2 0. 0. 10000000.
2 1 0. 0. 10000000.
MATC
1 .72+6 0.3 .0028 22.-6$ AL-ALLOY,METRIC UNITS
E23 SECTION PROPERTIES $ROD ELEMENTS
1 4.168$AREA OF THE RODS
SHELL SECTION PROPERTIES
1 0.1$SKIN THICKNESS
E21 SECTION PROPERTIES$BEAM ELEMENTS
 DSY 1 16804. 0. 1262.7 0. 108. 144. 0. 6.0784 0. 0.
 0. 0. 0. -8.7778 17. 3.2222 17. 3.2222 -17. -8.7778 -17.
CONSTRAINT CASE 1
ZERO 1,2,3;1,16$ CANTILEVER THE FUSELAGE
[XQT ELD
E23$ROD ELEMENTS
NSECT=1$
NREF=2
1 17 1 4 3 1 $
4 20$
5 21$
6 22$
52 68$
53 69$
54 70$
7 23 1 4 10 1$
E41$ MEMBRANE PANELS
NSECT=1$
1 17 18 2 2 16 1 $
49 65 66 50 2 16 1$
17 33 34 18 1 1 2 2 16$
23 39 40 24 1 1 9 2 16$
32 48 33 17$
48 64 49 33$
E21
NSECT=1$
NREF=1
1 2 2 16 2 16 $
49 50 2 16 2 16$
```

```
33 34$
34 35$
39 40$
40 41$
41 42$
42 43$
43 44$
44 45$
45 46$
46 47$
47 48$
48 33$
[XQT E
[XQT EKS
[XQT TOPO
[XQT K
[XQT INV
[XQT AUS
 SYSVEC;APPLIED FORCES 1
  I=1;J=65; -10000.
 SYSVEC;APPLIED FORCES 2
  I=1;J=69;77; -20000. 20000.
 SYSVEC;UNIT VEC
  I=1; J=1,80; 1.0
 DEFINE WT=DEM DIAG 0 O
 DEFINE UN=UNIT VEC
 OBJF AUS 1 1=XTY(UN,WT)
[XQT DCU
 PRINT 1 OBJF AUS 1 1
[XQT SSOL
[XQT GSF
[XQT PSF
[XQT VPRT
 PRINT APPL FORC 1 1
 PRI T STAT DISP 1 1
[XQT SSOL
 RESET SET=2
[XQT GSF
 RESET SET=2
[XQT PSF
 RESET SET=2
[XQT VPRT
 PRINT APPL FORC 2 1
 PRINT STAT DISP 2 1
[XQT EXIT
```

```
[TQ TAB                            . GENERATE BASIC TABLES DEFINING STRUCTURE
START 372$
TITLE"SATURN V LAUNCHER UMBILICAL TOWER (LUT)
TEXT$
"SATURN V LAUNCHER UMBILICAL TOWER (LUT)
"
"DAT ESIGNEDS'IO EXEMPLIFY THE USE OF "INC"
 "AND "MOD" COMMANDS IN THE "ELD" PROCESSOR.
MATERIAL CONSTANTS$
$                                      MATERIAL PROPERTY IS DEFINED WITH A
$                                      WEIGHT DENSITY
      1 3.+7 .29982668 .28$
CONSTRAINT CASE 1$
$                                      JOINTS 1-4 ARE COMPLETELY CONSTRAINED
      ZERO 1 2 3 4 5 6; 1 4 1$
BEAM ORIENTATION SPECIFICATIONS$
      1 1 3 1 0.$
      2 1 1 1 0.$
E21 SECTION PROPERTIES$
       GIVN  1  9012.1  0.   250.4  0.  44.16     9.47$
      GIVN  2  2096.4  0.    76.5  0.  22.4      8.4 $
      GIVN  3    63.35 0.    63.35 0.   7.3    126.7 $
      GIVN  4  4461.   0.   135.1  0.  31.77     4.66$
      GIVN  5  1814.5  0.    63.8  0.  20.       1.7 $
      GIVN  6  9012.1  0.   250.4  0.  44.16     9.47$
      GIVN  7  2096.4  0.    76.5  0.  22.4      8.4 $
      GIVN  8    28.14 0.    28.14 0.   5.58    56.28$
      GIVN  9  2824.8  0.    95.7  0.  24.71     2.57$
      GIVN 10  1140.7  0.    44.   0.  16.18     1.14$
      GIVN 11  1140.7  0.    44.   0.  16.18     1.14$
      GIVN 12    21.7  0.     2.89 0.   3.53      .08$
      GIVN 13   105.3  0.     2.79 0.   4.86      .1 $
      GIVN 14   446.3  0.    22.1  0.  10.59      .5 $
      GIVN 15    15.16 0.    15.16 0.   4.3     30.32$
      GIVN 16 21200.   0. 21200.   0. 276.   31100.  $
      GIVN 17   475.7  0.   475.7  0.  26.04   951.4 $
      GIVN 18  3988.6  0.   116.9  0.  29.11     3.47$
      GIVN 19 16667.   0. 16667.   0. 228.   24554.  $
      GIVN 20   475.7  0.   475.7  0.  26.04   951.4 $
      GIVN 21  3988.6  0.   116.9  0.  29.11     3.47$
      GIVN 22 14284.   0. 14284.   0. 205.   21538.  $
      GIVN 23   475.7  0.   475.7  0.  26.04   951.4 $
      GIVN 24  3266.7  0.   115.1  0.  27.65     3.8 $
      GIVN 25 12609.   0. 12609.   0. 182.   18683.  $
      GIVN 26   361.5  0.   361.5  0.  19.24   723.  $
      GIVN 27  3266.7  0.   115.1  0.  27.65     3.8 $
      GIVN 28 10773.   0. 10773.   0. 160.   16000.  $
      GIVN 29   361.5  0.   361.5  0.  19.24   723.  $
      GIVN 30  3266.7  0.   115.1  0.  27.65     3.8 $
      GIVN 31  9068.   0.  9068.   0. 138.   13476.  $
```

```
        GIVN 32    361.5  0.    361.5  0.  19.24    723.  $
        GIVN 33   3266.7  0.    115.1  0.  27.65      3.8 $
        GIVN 34   7459.   0.   7459.   0. 117.    11233.  $
        GIVN 35    361.5  0.    361.5  0.  19.24    723.  $
        GIVN 36   2364.3  0.     88.3  0.  24.7       3.57$
        GIVN 37   5969.   0.   5969.   0.  96.     8917.  $
        GIVN 38    361.5  0.    361.5  0.  19.24    723.  $
        GIVN 39   2364.3  0.     88.3  0.  24.7       3.57$
        GIVN 40   4585.   0.   4585.   0.  76.     6859.  $
        GIVN 41    248.5  0.    248.5  0.  12.88    497.  $
        GIVN 42   2096.4  0.     76.5  0.  22.4       8.4 $
        GIVN 43   2402.4  0.    930.1  0.  56.73     34.45$
        GIVN 44    248.5  0.    248.5  0.  12.88    497.  $
        GIVN 45   2096.4  0.     76.5  0.  22.4       8.4 $
        GIVN 46   2402.4  0.    930.1  0.  56.73     34.45$
        GIVN 47    248.5  0.    248.5  0.  12.88    497.  $
        GIVN 48   1814.5  0.     63.8  0.  20.        1.7 $
        GIVN 49   1266.5  0.    454.9  0.  32.65      7.63$
        GIVN 50    192.3  0.    192.3  0.   9.84    384.6 $
        GIVN 51   1814.5  0.     63.8  0.  20.        1.7 $
        GIVN 52   1266.5  0.    454.9  0.  32.65      7.63$
        GIVN 53    192.3  0.    192.3  0.   9.84    384.6 $
        GIVN 54   1814.5  0.     63.8  0.  20.        1.7 $
        GIVN 55    641.5  0.    107.3  0.  17.94      2.01$
        GIVN 56    192.3  0.    192.3  0.   9.84    384.6 $
        GIVN 57   1814.5  0.     63.8  0.  20.        1.7 $
        GIVN 58    641.5  0.    107.3  0.  17.94      2.01$
        GIVN 59    192.3  0.    192.3  0.   9.84    384.6 $
        GIVN 60   1814.5  0.     63.8  0.  20.        1.7 $
        GIVN 61   2987.3  0.    203.5  0.  29.43      4.52$
        GIVN 62   1140.7  0.     44.   0.  16.18      1.14$
        GIVN 63   1326.8  0.     53.1  0.  18.23      1.71$
        GIVN 64    446.3  0.     22.1  0.  10.59       .5 $
        GIVN 65     15.16 0.     15.16 0.   4.3      30.32$
        GIVN 66   5886.9  0.    170.3  0.  34.71      4.9 $
        GIVN 67   2987.3  0.    203.5  0.  29.43      4.52$
        GIVN 68   2987.3  0.    203.5  0.  29.43      4.52$
        GIVN 69   2987.3  0.    203.5  0.  29.43      4.52$
        GIVN 70   2987.3  0.    203.5  0.  29.43      4.52$
        GIVN 71     15.16 0.     15.16 0.   4.3      30.32$
        GIVN 72 26478.    0. 26478.    0. 326.    38584.  $
        GIVN 73    562.   0.    562.   0.  18.41   1124.  $
        GIVN 74    732.   0.    732.   0.  24.35   1464.  $
        GIVN 75   1157.   0.   1157.   0.  40.19   2314.  $
        GIVN 76 26300.    0. 26300.    0. 294.    38700.  $
        GIVN 77    562.   0.    562.   0.  18.41   1124.  $
        GIVN 78   1556.   0.   1556.   0.  56.6    3112.  $
  JOINT LOCATIONS$
          1  360.   907. -2880.  270.   589. -2160. 2 48$
          2 -360.   907. -2880. -270.   589. -2160. 2 48$
```

```
  3 -360. -427. -2880. -270. -109. -2160. 2 48$
  4  360. -427. -2880.  270. -109. -2160. 2 48$
  5  316.  348. -2520.  270.  348. -2160. 2 24$
  6 -316.  348. -2520. -270.  348. -2160. 2 24$
  7 -316.  132. -2520. -270.  132. -2160. 2 24$
  8  316.  132. -2520.  270.  132. -2160. 2 24$
  9  316.  480. -2520.  270.  480. -2160. 2 24$
 10 -316.  480. -2520. -270.  480. -2160. 2 24$
 11 -316.    0. -2520. -270.    0. -2160. 2 24$
 12  316.    0. -2520.  270.    0. -2160. 2 24$
 13  108.  480. -2520.  108.  480. -2160. 2 24$
 14 -108.  480. -2520. -108.  480. -2160. 2 24$
 15  108.  348. -2520.  108.  348. -2160. 2 24$
 16 -108.  348. -2520. -108.  348. -2160. 2 24$
 17  108.  132. -2520.  108.  132. -2160. 2 24$
 18 -108.  132. -2520. -108.  132. -2160. 2 24$
 19  108.    0. -2520.  108.    0. -2160. 2 24$
 20 -108.    0. -2520. -108.    0. -2160. 2 24$
 21  316.  240. -2520.  270.  240. -2160. 2 24$
 22    0.  751. -2520.    0.  589. -2160. 2 24$
 23 -316.  240. -2520. -270.  240. -2160. 2 24$
 24    0. -271. -2520.    0. -109. -2160. 2 24$
 25  316.  751. -2520.$
 26 -316.  751. -2520.$
 27 -316. -271. -2520.$
 28  316. -271. -2520.$
 53  240.  348. -1920.  240.  348.  1440. 15 20$
 54  108.  480. -1920.  108.  480.  1440. 15 20$
 55 -108.  480. -1920. -108.  480.  1440. 15 20$
 56 -240.  348. -1920. -240.  348.  1440. 15 20$
 57 -240.  132. -1920. -240.  132.  1440. 15 20$
 58 -108.    0. -1920. -108.    0.  1440. 15 20$
 59  108.    0. -1920.  108.    0.  1440. 15 20$
 60  240.  132. -1920.  240.  132.  1440. 15 20$
 61  108.  348. -1920.  108.  348.  1440. 15 20$
 62 -108.  132. -1920. -108.  132.  1440. 15 20$
 63 -108.  348. -1920. -108.  348.  1440. 15 20$
 64  108.  132. -1920.  108.  132.  1440. 15 20$
 65  240.  240. -1920.  240.  240.  1680. 16 20$
 66    0.  480. -1920.    0.  480.  1680. 16 20$
 67 -240.  240. -1920. -240.  240.  1680. 16 20$
 68    0.    0. -1920.    0.    0.  1680. 16 20$
 69  240.  480. -1920.  240.  480.  1680. 16 20$
 70 -240.  480. -1920. -240.  480.  1680. 16 20$
 71 -240.    0. -1920. -240.    0.  1680. 16 20$
 72  240.    0. -1920.  240.    0.  1680. 16 20$
353  240.  290.  1680.$
354   50.  480.  1680.$
355  -50.  480.  1680.$
356 -240.  290.  1680.$
```

```
        357 -240.  190.  1680.$
        358  -50.    0.  1680.$
        359   50.    0.  1680.$
        360  240.  190.  1680.$
        361   50.  290.  1680.$
        362  -50.  190.  1680.$
        363  -50.  290.  1680.$
        364   50.  190.  1680.$
RMASS$
$                                         RIGID LUMPED MASSES
        REPEAT   4 1$
         49 100.025$
        REPEAT   4 1$
         89  43.880$
        REPEAT   4 1$
        109  24.788$
        REPEAT   4 1$
        129  47.480$
        REPEAT   4 1$
        149  30.632$
        REPEAT   4 1$
        169  64.710$
        REPEAT   4 1$
        189  56.938$
        REPEAT   4 1$
        209  84.460$
        REPEAT   4 1$
        229  72.472$
        REPEAT   4 1$
        249 110.585$
        REPEAT   4 1$
        269 116.048$
        REPEAT   4 1$
        289 121.750$
        REPEAT   4 1$
        309 106.598$
        REPEAT   4 1$
        329 151.378$
        REPEAT   4 1$
        349 231.478$
        REPEAT   4 1$
        369 308.282$
[XQT ELD                                 . READ ELEMENT DEFINITIONS
E21 $
 GROUP 1       "FLOOR 2
     NSECT=1  ;  9 25   1  1 4 1    $
                 5  9   1  1 4 1    $
                 5 21   1  1 2 2    $
                 8 21               $
                 6 23               $
```

```
          NSECT=2   ; 22 26   1  1 2 2    $
                      22 25   1  1 2 2    $
          NSECT=3   ; 13 25   1  1 2 1    $
                      13 22   1  1 2 6    $
                      14 22               $
                      20 27               $
                       9 15   1  1 2 1    $
                      11 18   1  1 2 5    $
                      12 17               $
                      17 21   1  1 2 3    $
                      15 21               $
                      18 23   1  1 2 1    $
          NSECT=4   ;  5 15   1  1 2 1    $
                       7 18               $
                       8 17               $
                      15 16   1  1 2 2    $
          NSECT=5   ;  9 13   1  1 2 1    $
                      11 20               $
                      12 19               $
                      13 14   1  1 2 6    $
      GROUP 2       "FLOOR 3
        MOD JOINT=24 ; MOD NSECT=5 $
          NSECT=1   ;  9 25   1  1 4 1    $
                       5  9   1  1 4 1    $
                       5 21   1  1 2 2    $
                       8 21               $
                       6 23               $
          NSECT=2   ; 22 26   1  1 2 2    $
                      22 25   1  1 2 2    $
          NSECT=3   ; 13 25   1  1 2 1    $
                      13 22   1  1 2 6    $
                      14 22               $
                      20 27               $  240.
       4148.     18.      0.      0.
      12372.      0.      0.      0.
      12372.      0.   7168.  32768.
        584.    224.   2080.    240.
          1.      0.      0.      0.***
          3.      1.      0.      0.***TITLE
     VMAX         0.
          4.      1.      0.      0.***
     VMAX         0.      7.
          0.     13.      0.      0.***
      16384.      0.      0.     48.
      16520.      0.      0.     48.
          0.      0.  18948.      0.
      12624.      0.      0.      0.
          1.  10240.  19205.      0.
       8256.      0.      0.     48.
          1.  32768.      0.      0.13 14   1  1 2 6    $
```

```
GROUP 3      "FLOORS 4-19
 INC NSECT=3 $
  MOD JOINT=0 ; MOD NSECT=0 $
    NSECT=18 ; 54 66    1 1 15 20 $
    NSECT=18 ; 55 66    1 1 15 20 $
    NSECT=18 ; 55 70    1 1 15 20 $
    NSECT=18 ; 54 69    1 1 15 20 $
    NSECT=18 ; 56 70    1 1 15 20 $
    NSECT=18 ; 56 67    1 1 15 20 $
    NSECT=18 ; 57 67    1 1 15 20 $
    NSECT=18 ; 57 71    1 1 15 20 $
    NSECT=18 ; 58 71    1 1 15 20 $
    NSECT=18 ; 58 68    1 1 15 20 $
    NSECT=18 ; 59 72    1 1 15 20 $
    NSECT=18 ; 59 68    1 1 15 20 $
    NSECT=18 ; 60 72    1 1 15 20 $
    NSECT=18 ; 60 65    1 1 15 20 $
    NSECT=18 ; 53 69    1 1 15 20 $
    NSECT=18 ; 53 65    1 1 15 20 $
 INC NSECT=0 $
  MOD JOINT=300 $
    NSECT=66 ; 54 66    1 1  2  6 $
               55 66    1 1  2  1 $
               54 69    1 1  2  1 $
               56 70    1 1  2  1 $
               57 67    1 1  2  1 $
               58 71    1 1  2  1 $
               59 68              $
               60 65              $
               53 69              $
               53 65              $
  MOD JOINT=0$
    NSECT=11 ; 53 61    1 1 14 20 $
               61 63    1 1 14 20 $
               56 63    1 1 14 20 $
               57 62    1 1 14 20 $
               62 64    1 1 14 20 $
               60 64    1 1 14 20 $
    NSECT=12 ; 55 63    1 1 14 20 $
               58 62    1 1 14 20 $
               59 64    1 1 14 20 $
    NSECT=13 ; 62 63    1 1 14 20 $
               61 64    1 1 14 20 $
    NSECT=14 ; 54 61    1 1 14 20 $
    NSECT=15 ; 61 65    1 1 14 20 $
               63 66    1 1 14 20 $
               63 67    1 1 14 20 $
               62 67    1 1 14 20 $
               63 70    1 1 14 20 $
               62 71    1 1 14 20 $
```

```
                  62 68     1 1 14 20 $
                  64 68     1 1 14 20 $
                  64 72     1 1 14 20 $
                  64 65     1 1 14 20 $
                  61 65     1 1 14 20 $
                  61 69     1 1 14 20 $
 INC NSECT=6 $
  MOD JOINT=280$
    NSECT=61 ; 53 61     1 1  2 20 $
    NSECT=61 ; 61 63     1 1  2 20 $
    NSECT=61 ; 56 63     1 1  2 20 $
    NSECT=61 ; 57 62     1 1  2 20 $
    NSECT=61 ; 62 64     1 1  2 20 $
    NSECT=61 ; 60 64     1 1  2 20 $
    NSECT=62 ; 55 63     1 1  2 20 $
    NSECT=62 ; 58 62     1 1  2 20 $
    NSECT=62 ; 59 64     1 1  2 20 $
    NSECT=63 ; 62 63     1 1  2 20 $
    NSECT=63 ; 61 64     1 1  2 20 $
    NSECT=64 ; 54 61     1 1  2 20 $
    NSECT=65 ; 61 66     1 1  2 20 $
    NSECT=65 ; 63 66     1 1  2 20 $
    NSECT=65 ; 63 67     1 1  2 20 $
    NSECT=65 ; 62 67     1 1  2 20 $
    NSECT=65 ; 63 70     1 1  2 20 $
    NSECT=65 ; 62 71     1 1  2 20 $
    NSECT=65 ; 62 68     1 1  2 20 $
    NSECT=65 ; 64 68     1 1  2 20 $
    NSECT=65 ; 64 72     1 1  2 20 $
    NSECT=65 ; 64 65     1 1  2 20 $
    NSECT=65 ; 61 65     1 1  2 20 $
    NSECT=65 ; 61 69     1 1  2 20 $
GROUP 4       "INTER-FLOOR COLUMNS AND DIAGONALS
 INC NSECT=0 $
  MOD JOINT=0$
   NREP=2     $
    NSECT=72 ;  1 25     1 1  4  1 $
               25 49     1 1  4  1 $
               25 45               $
    NSECT=73 ;  2 22     1 1  2 24 $
                1 22     1 1  2 24 $
                3 24     1 1  2 24 $
                4 24     1 1  2 24 $
    NSECT=74 ;  2 23               $
                4 21               $
                1 21     1 1  2  2 $
    NSECT=75 ; 26 47               $
               27 47               $
               28 45               $
    NSECT=76;  49 69     1 1  4  1 $
```

```
     NSECT=77 ; 49 66    1 1  2  2 $
                50 66    1 1  2  2 $
     NSECT=78 ; 50 67              $
                49 65    1 1  2  2 $
                52 65              $
  INC NSECT=3 $
     NSECT=16 ; 69 89    1 1 15 20   $
     NSECT=16 ; 70 90    1 1 15 20   $
     NSECT=16 ; 71 91    1 1 15 20   $
     NSECT=16 ; 72 92    1 1 15 20   $
     NSECT=17 ; 69 86    1 1 15 20   $
     NSECT=17 ; 70 86    1 1 15 20   $
     NSECT=17 ;     1. 39424.  8192. 45056.
     1. 37376.  6616. 22016.
  8257. 37376.   232.  5680.
     1. 53760.   416.  5632.
     1. 37376.   160.  5632.
  8393. 39424.     0.  7024.
   657. 39424.  8240. 31488.
     1. 37797.  6504. 38400.
     0.   224.     0.     0.
     1.     0.     0.     0.***
     3.     1.     0.     0.***TITLE
RECUR4      0.
     4.     1.     0.     0.***
RECUR4      C.     6.
     0.    15.     0.     0.***
 16724.     0.     0.    48.
     0.     0.     0.    16.
     0.T INV                           . FACTOR K
$
[XQT M                               . FORM SYSTEM M
$                                      FORM SYSTEM M WITH MASS ENTRIES BY
$                                      CONVERTING MATERIAL PROPERTY WEIGHT
$                                      DENSITY INTO MASS DENSITY
      RESET G=386.088$
[XQT AUS                             . GENERATE LOAD DATA
     M+RM=SUM(CEM,RMAS)$
$
     ALPHA; CASE TITLES
       1"INERTIA LOAD FOR PRESTRESS
$
      UNIT VECTORS=RIGIDS
      DEFINE X=UNIT VECTORS 1 1 3,3$
$
      APPLIED FORCES=PRODUCT(-386.088,M+RM,X)$
[XQT SSOL                            . COMPUTE STATIC SOLUTION
[XQT GSF
     RESET EMBED=1$
[XQT KG                              . FORM SYSTEM KG, PRESTRESS SOLUTION
```

```
$
[XQT AUS
     K+KG=SUM(K,KG)$
$
[XQT INV                                . FACTOR K+KG
      RESET K=K+KG$
[XQT EIG                                . SOLVE SYSTEM EIGENPROBLEM
$                                         PRESTRESS VIBRATION ANALY 70 87    1 1 15 20
$
     NSECT=17 ; 71 87    1 1 15 20   $
     NSECT=17 ; 71 88    1 1 15 20   $
     NSECT=17 ; 72 88    1 1 15 20   $
     NSECT=17 ; 72 85    1 1 15 20   $
     NSECT=17 ; 69 85    1 1 15 20   $
[XQT TOPO                               . ANALYZE ELEMENT INTERCONNECTIVITY
$
[XQT E                                  . FORM ELEMENT DATA PACKETS
$
[XQT EKS                                . INSERT K, S INTO ELEMENT DATA PACKETS
$
[XQT K                                  . FORM SYSTEM K
$
```