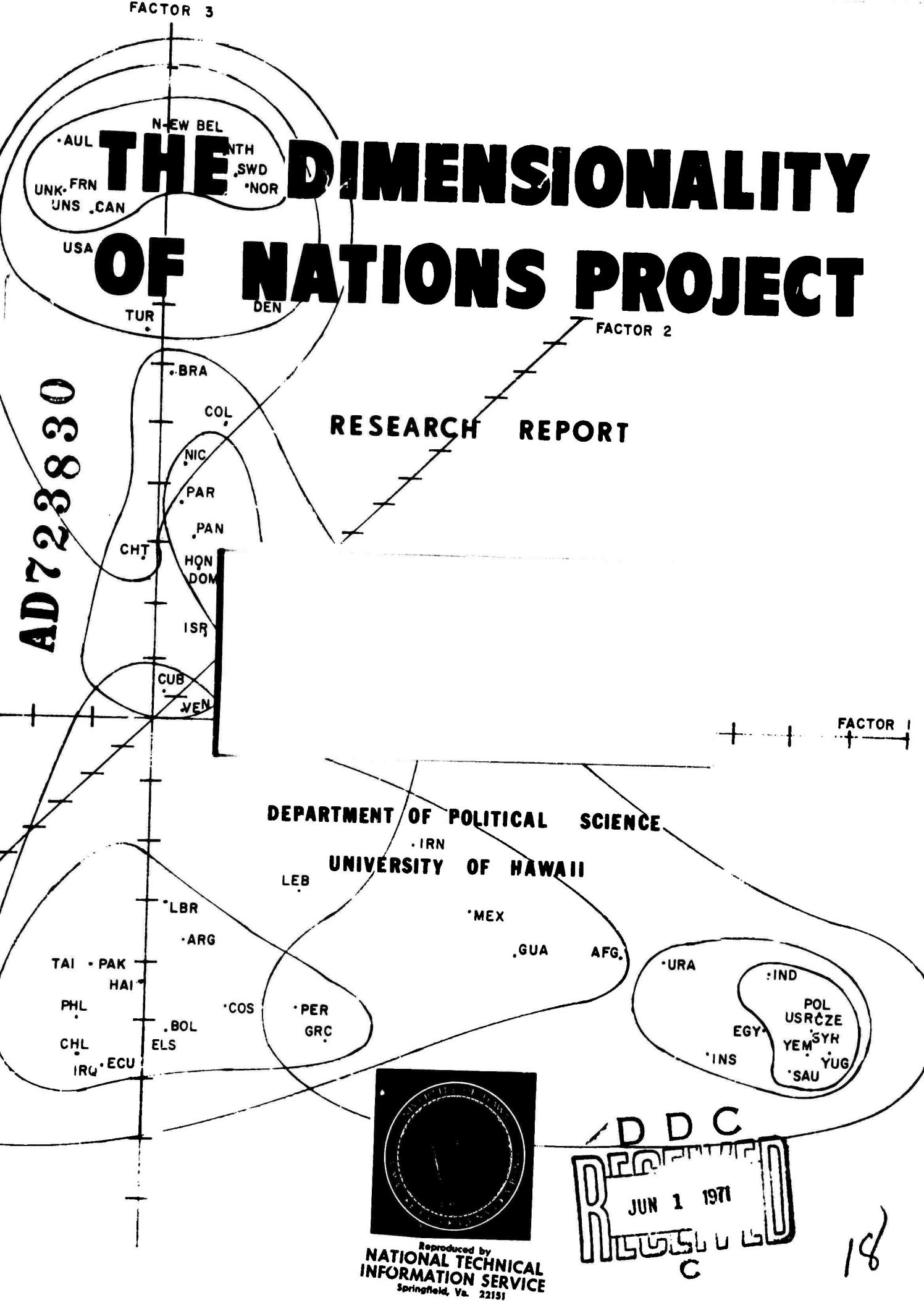


THE DIMENSIONALITY OF NATIONS PROJECT

RESEARCH REPORT

DEPARTMENT OF POLITICAL SCIENCE
UNIVERSITY OF HAWAII

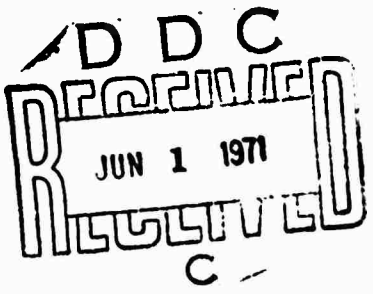
The Dimensionality of Nations Project
Department of Political Science
University of Hawaii

RESEARCH REPORT NO. 53

DYNA: DYNAMIC STORAGE ALLOCATION IN FORTRAN FOR THE IBM/360 OPERATING SYSTEM

Alan C. H. Kam
and
Charles F. Wall

April 1971



Prepared in connection with research supported by the Advanced Research Projects Agency, ARPA Order No. 1063, and monitored by the Office of Naval Research, Contract No. N00014-67-A-0387-0003.

BLANK PAGE

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)

DIMENSIONALITY OF NATIONS PROJECT

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

3. REPORT TITLE

DYNA: DYNAMIC STORAGE ALLOCATION IN FORTRAN FOR THE IBM/360 OPERATING SYSTEM.

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Research Report No. 53

5. AUTHOR(S) (First name, middle initial, last name)

KAM, Alan C. H. and Charles F. Wall

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
April 1971	15	5

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
N00014-67-A-0387-0003	Research Report No. 53
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited and reproduction in whole or in part is permitted for any purpose of the United States Government.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
2500 Campus Road Honolulu, Hawaii 96822	Advanced Research Projects Agency Washington, D.C.

13. ABSTRACT

This report describes an I.B.M./360 Assembly Language routine to allow for dynamic allocation of core storage in Fortran programs. The report describes in detail the implementation and use of the dynamic allocation subroutine DYNA.

A fairly detailed knowledge of FORTRAN is assumed in the report. The dynamic routine may be implemented by the FORTRAN programmer without a detailed understanding Assembly Language routine DYNA. A complete description of DYNA is included, for those programmers who The discussion is technical and assumes a knowledge of I.B.M./360 Operating System Macros and Assembly Language.

DD FORM 1 NOV 68 1473 (PAGE 1)

S/N 0101-807-6801

UNCLASSIFIED

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
DYNAMIC STORAGE ALLOCATION						

The Dimensionality of Nations Project
Department of Political Science
University of Hawaii

DYNA: DYNAMIC STORAGE ALLOCATION IN FORTRAN FOR THE IBM/360 OPERATING SYSTEM

ABSTRACT

This report describes an I.B.M./360 Assembly Language routine to allow for dynamic allocation of core storage in Fortran programs. The report describes in detail the implementation and use of the dynamic allocation subroutine DYNA.

A fairly detailed knowledge of FORTRAN is assumed in the report. The dynamic routine may be implemented by the FORTRAN programmer without a detailed understanding of the Assembly Language routine DYNA. A complete description of DYNA is included. The discussion is technical and assumes a knowledge of I.B.M./360 Operating System Macros and Assembly Language.

DYNA: DYNAMIC STORAGE ALLOCATION IN FORTRAN FOR THE IBM/360 OPERATING SYSTEM

Alan C. H. Kam
and
Charles F. Wall

University of Hawaii

1. INTRODUCTION

A major limitation of Fortran is that storage for data is allocated at compile-time by the programmer. Essentially, some opimum dimensions are selected, the program is compiled and fixed amount of storage is made available. If a particular problem requires less space than is allocated, storage is wasted. If more space is required than has been allocated, the program must be redimensioned and recompiled. In today's multiprogrammed systems, either situation is an inefficient use of system resources.

A solution to the problem is to allocate storage for data at execution time, dynamically. The general rationale is that allocation of storage for data is a dynamic function that varies with each use of a particular program.

2. DYNA: Dynamic Storage Allocation in Fortran for the IBM/360

DYNA is an assembly language subroutine that provides an interface between a Fortran program or subroutine and the operating system's (O.S.) main storage manager. Each time DYNA is called, storage is allocated and made available to the Fortran subroutine.

The next four sections discuss the O.S. Getmain/Freemain facility, the interface for these functions, and the Error Conditions encountered.[1]

2.1 Fortran Convention

The standard Fortran call is used to call DYNA and to allocate storage. The order of the parameters is as follows:

CALL DYNA (SUB, ARRAY1, NSIZE1, ..., ARRAYN, NSIZEN)

where

SUB is the name of the Fortran subroutine the storage is allocated to

ARRAYN is the Nth array to be allocated

NSIZEN is the amount of storage in words of ARRAYN.

DYNA is called by the calling Fortran routine which in turn calls the Fortran subroutine SUB. SUB must be declared EXTERNAL in the calling routine and the parameters must not appear in COMMON.

The parameter list is a list of the addresses of each parameter in the subroutine call. The first parameter is the address of the subroutine to be called by DYNA. The last parameter has its high order byte equal to 128. The call to DYNA consists of loading the address of the parameter list in general register 1, loading the address of DYNA in register 15 and then branching to register 15 and linking register 14. Register 13 points to a save area within the calling routine (standard O.S. linkage convention).

---

[1] See References.

2.2 O.S. Getmain/Freemain

Getmain is an SVC which requests the control program to assign ownership of an area of main storage to a routine. This SVC has many options and is conveniently coded as a macro in assembly language. However, it has been hand coded in DYNA using the R and VU forms. The R-form requests a specific amount of main storage in register 0. The SVC returns the address of this area in general register 1. The VU-form requests an amount of main storage between two ranges. If the request cannot be satisfied, the control program will terminate the calling routine (see Error Conditions). For this form, register 1 points to a parameter list with the appropriate flag specifying the VU-request.

The Freemain SVC frees the storage assigned by the Getmain. Both the R and V forms are used and are hand coded.

2.3 The Interface

Upon entry to DYNA, the general registers are saved in an area provided by the calling routine. Then the Getmain SVC issued to obtain a work area is appropriately initialized.

The next phase is to process the parameter list and save the address of the calling routine. The parameter requests are rounded up in preparation for the double-word alignment. Once the total size of the arrays has been calculated, the second Getmain is issued. The main storage block allocated is used by DYNA to reallocate array addresses. The original array addresses are replaced with the new double-word aligned addresses. Thus, the initial parameters corresponding to the

addresses passed to DYNA of the arrays are replaced by the addresses of the main storage area to be used.

DYNA then calls the subroutine whose address is given by the first parameter in the list passed to DYNA. On return from the called subroutine DYNA frees the main storage allocated for the arrays and re-establishes the linkage to the calling program. DYNA then frees the work area allocated and returns to the calling routine.

Since DYNA obtains a work area (activation list) upon entry and frees this work at exit, DYNA is a re-entrant routine and may be called by more than one program in any order.[2]

2.4 Error Conditions

If there is no main storage available the system will abnormally end (ABEND) with a code of S80A or S804. Other codes are possible but have yet to be encountered.

If there is some storage available but not enough to satisfy the request, DYNA calls $ERR1, an error routine written in Fortran which has as an argument a vector with the following values: the address of the core that was allocated, the amount the system could allocate, the minimum requested size (usually zero) and the maximum size requested.

$ERR2 is called if the parameter list has the improper number of arguments. There should always be an odd number of arguments since the first argument is the name of the Fortran subroutine called by DYNA and all other arguments are the paired names and sizes. This

---

[2] See keferences.

error exit occurs in the debugging stage and does not return control. A message is printed and control is returned to the system.

3. PROGRAMMING DYNAMIC ROUTINES

The use of DYNA is relatively straightforward. The subroutine (DYNA) is reentrant and may be nested to any level. The tradeoff is in the time required to call DYNA in comparison to its efficient use of space.

3.1 Some Rules

As mentioned in 2.1 standard Fortran is used to call DYNA. The name of the subroutine to which DYNA will pass control must be declared EXTERNAL in the routine calling DYNA. The parameters in the call statement must not appear in COMMON. If an array is to be DOUBLE PRECISION, the size parameter must reflect this (e.g. ten words DOUBLE PRECISION must be passed as twenty words). DYNA always allocates on double-word boundaries. If a particular array is doubly subscripted the following form is not allowed in the called subroutine:

```
DIMENSION X(1,1)
```

Instead, the appropriate dimensions should be passed in COMMON and used in the called program in the following manner:

```
DIMENSION X(N,M)
COMMON N,M
```

For vectors, either form is acceptable.

3.2 An Added Advantage

If your program follows the normal Fortran convention of storing arrays columnwise, the arrays may be doubly-subscripted and calls to subroutines, as in the Scientific Subroutine Package, may be called without conversion. Since we always allocate the exact amount of storage needed by each array, the array may also be treated (in separate routines) as a vector. Since it is useful to use doubly-subscripted arrays for input/output and certain computations and singly-subscripted arrays (vectors) for other computations it is useful to be able to eliminate conversion problems. (For a more complete discussion see IBM/Scientific Subroutine Package-Programmers Guide).

3.3 A Sample Program

Consider a program which calculate means, standard deviations and correlations. A simple call to the IBM/SSP subroutine CØRRE will suffice. All I/Ø will not be shown as the purpose is to demonstrate the use of DYNA. We shall consider that we are calling the DOUBLE PRECISION version of CORRE.

```
C -- THIS IS THE MAIN ROUTINE --
      CØMMON N, M
      EXTERNAL SUBC
C -- READ CØNTRØL CARD AND ALLØCATE CØRE
      READ (5, 10) NRØWX, NCØLX
   10 FORMAT (2I3)
      N = NROWX
      M = NCOLX
```

```
      N1 = NROWX * 2
      N2 = NCØLX * 2
      N3 = N1 * N2
      N4 = (N2 * (N2 + 1))/2
C -- CALL SUBC VIA DYNA
      CALL DYNA(SUBC,X,N3,XBAR,N2,STD,N2,RX,N3,R,N4,B,N2,D,N2,T,N2)
      STØP
      END

C -- THIS IS SUBROUTINE SUBC
      SUBROUTINE SUBC(X,NA,XBAR,NB,STD,NC,RX,ND,R,NE,B,NF,D,NG,T,NH)
C -- NOTICE THAT IN THE SUBROUTINE WE
C -- CANNOT HAVE DUPLICATE DUMMY NAMES
      DIMENSIØN X(N,M),XBAR(1),STD(1),RX(1),R(1),B(1),D(1),T(1)
      CØMMØN N,M
C -- READ IN DATA
      DØ 20 I = 1, N
20    READ (5,10) (X(I,J), J = 1, M)
10    FORMAT (10F8.0)
C -- CALL CORRE
      CALL CORRE (N,M,I,X,XBAR,STD,RX,R,B,D,T)
C -- PRINT RESULTS
      .
      .
      .
      RETURN
      END
```

3.4 User Documentation

Primarily, the user must know:

a) The formula to calculate the dynamic area; and

b) the amount of static storage used by the program and buffers.

A simple test run with size = 1 for all dynamic data arrays will give the base (e.g. amount of static storage). To be safe this should be rounded up to the next even K bytes. The formula is determined by the particular program. For example, the formula for our sample program would be given as follows:

SIZE (in bytes) = (2 * N3 + 5 * N2 + N4) * 4

where

N1 = Number of rows in X times 2

N2 = Number of columns in X times 2

N3 = N1 * N2

N4 = (N2 * (N2 + 1))/2

the region is = BASE + SIZE rounded up to next even K bytes.

This is the estimate the user should supply on the EXEC card.

```
C -- TO SUPPLY LESS.
C --        RESULTS IN AN 80A;
C -- TO SUPPLY MORE:
C --        IS A WASTE FOR SURE:
```

```
*****************************************************************
*
*       DYNAMIC CORE ALLOCATION PROGRAM FOR FORTRAN PROGRAMS
*       USING OS MACRO GETMAIN FOR THE ALLOCATION MECHANISM.
*
*                                   2. DOUBLE-WORD BOUNDARIES
*
*                   CORE GUESTIMATION ALGORITHM
*
*                   LINKAGE EDITOR MAP  TOTAL LENGTH
*                   SYSIN/SYSPRINT          4K BYTES
*                   FOR N DD CARDS          N*2K BYTES
*                   SUM (BLKSIZE*BUFNO)     M
*                   TOTAL REQUIREMENTS  (TOTAL LENGTH)+4K+N*2K+M
*
*****************************************************************
DYNA     CSECT
         STM       14,12,12(13)          SAVE THE GENERAL REGISTERS
         LR        12,15                 USE GR 12 AS THE BASE REGISTER
         USING     DYNA,12               TELL THE ASSEMBLER THE BASE IS
         LR        2,1                   SAVE REGISTER 1 FOR LATER.
         LA        0,SAVESIZE            GR 0 HAS THE SIZE OF SAVE AREA
         BAL       1,*+4                 IBM CONVENTION FOR GETMAIN (R)
         SVC       10                    GET MAIN SVC
         USING     DYNASAVE,11
         LR        11,1                  USE GR 11 AS SAVE BASE REGISTER
         LR        1,2                   RESTORE GR 1 TO ORIGINAL VALUE
         ST        13,SAVEAREA+4         SAVE GR 13 FOR LINKAGE
         LA        13,SAVEAREA           RESET THE LINKAGE
         ST        1,SAVEGR1
*
*        INITIALIZATION OF THE PARAMETER LISTS, AND CONSTANTS
*
         LA        3,MINSIZE
         ST        3,GETSIZE
         LA        3,FREEAREA
         ST        3,GETAREA
         ST        3,ERRLIST
         MVI       GETMODE,X'E0'
         MVI       GETSP,X'00'
         XC        FREEAREA(16),FREEAREA
         EJECT
*****************************************************************
*
*        NOW START ALLOCATING THE CORRECT ADDRESSESS.
*
*****************************************************************
         L         5,SAVEGR1
         L         4,0(0,5)              GET ADDRESS OF REAL MAIN PROGRAM
         L         4,0(0,4)              GR 4 HAS THE ADDRESS OF REAL ONE
         ST        4,SAVEMAIN
         LA        5,4(0,5)              GR 5 POINTS TO THE ARRAY ADDRESS
         ST        5,SAVEGR1             SAVE THE NEW PARMLIST PNTR.
*
*        WANDER THRU THE PARAMETER LIST TO DETERMINE THE MAX CORE REQ'D
*
         LA        0,0                   ZERO THE WORK REGISTER.
         LA        4,0                   ZERO THE WORK REGISTER.
LOCATES  EQU       *
```

```
         CLI     0(5),X'80'          CHECK FOR ERRONEOUS PARM LIST
         BE      ERROR1
         L       4,4(0,5)            GET THE NUMBER OF WORDS REQ'D
         L       4,0(0,4)            THE ACTUAL NUMBER
         SRL     4,1                 NUMBER OF WORDS/2
         LA      4,1(0,4)            SAFETY FACTOR
         SLL     4,3                 PREPARE FOR DOUBLE WORDS
         AR      0,4                 SUM UP THE NUMBER OF DOUBLE WORD
         CLI     4(5),X'80'          CHECK FOR END OF LIST
         LA      5,8(0,5)            STEP FOR THE EXT SET
         BNE     LOCATES
*
*        FORCING DOUBLE WORD ALIGNMENT FOR DOUBLE PRECISION ARRAYS.
*
         ST      0,MAXSIZE           FOR THE GET MAIN ROUTINE
         LA      1,GETLIST
         SVC     4                   GETMAIN SVC CODE
         L       2,FREEAREA          GR 2 HAS THE ADDRESS OF FREECORE
         L       3,FREESIZE          GR 3 HAS THE AMOUNT FO FREE CORE
         C       3,MAXSIZE           TEST IS ALLOCATED ENOUGH
         BL      ERROR
         L       5,SAVEGR1           RESET GR 1
LOOP     EQU     *
         ST      2,0(0,5)            CHANGE THE OLD ARRAY ADDRESS
         L       6,4(0,5)            PICK UP THE ADDRESS OF =BYTES
         L       6,0(0,6)            GR6 = NUMBER OF WORDS IN ARRAY
         SRL     6,1                 =WORDS/2
         LA      6,1(0,6)            =WORD==WORDS+1
         SLL     6,3                 =WORDS=MULTIPLE OF 8
         AR      2,6                 GR 2 -> NEW FREE AREA
         CLI     4(5),X'80'          CHECK IF LAST ARGUMENT
         LA      5,8(0,5)            GR 5 -> NEW ARRAY LIN LIST
         BNE     LOOP
         EJECT
*
*        NOW CALL THE REAL MAIN PROGRAM WITH THE NEW PARAMETER LIST
*
         L       1,SAVEGR1
         L       15,SAVEMAIN
         BALR    14,15               GO TO IT.
*
*        FIRST FREE THE ALLOCATED CORE.
*        RETURN TO THE MAIN CALLER NORMALLY.
*
RETRUNS  EQU     *
         L       0,FREESIZE
         L       1,FREEAREA
         SVC     10                  FREE THE ALLOCATED CORE.
RETURNQ  EQU     *
         L       13,SAVEAREA+4
         LA      0,SAVESIZE          FREE UP THE SAVE AREA
         LR      1,11                RETURN THE ALLOCATED ADDRESS
         SVC     10                  FREE MAIN
         LM      14,12,12(13)
         LA      15,0
         BR      14
*
*        NOT ENOUGH CORE IS ALLOCATED FOR THIS JOB
*
```

```
ERROR    EQU      *
         LA       1,ERRLIST
         L        15,=V($ERR1)
         LA       14,RETRUNS
         BR       15
*
*        INCORRECT SETUP OF THE PARAMETER LIST, MISSING =OF BYTES
*
ERROR1   EQU      *
         LA       14,RETURNQ
         L        15,=V($ERR2)
         BR       15
         EJECT
*****************************************************************
*
*        DYNAMIC SAVE AREA AND CONSTANTS FOR PROGRAM
*        THE ADDRESSESES AND CONSTANTS ARE FILLED IN AT EXECUTION TIME
*
*****************************************************************
         LTORG
DYNASAVE DSECT
SAVEBEG  EQU      *
         DS       0F
GETLIST  EQU      *
GETSIZE  DS       AL4(MINSIZE)          PARAMETER LIST FOR GETMAIN
GETAREA  DS       AL4(FREEAREA)
GETMODE  DS       XL1'E0'               VC MODE
GETSP    DS       XL1'00'               SP 0
*
         DS       0F
FREEAREA DS       AL4(0)                ADDRESS OF FREEE CORE AREA
FREESIZE DS       1F'0'                 NUMBER OF BYTES ALLOCATED
MINSIZE  DS       1F'0'                 MINIMUM CORE REQUIREMENT
MAXSIZE  DS       1F'0'                 MAXIMUM CORE REQUIREMENT
*
ERRLIST  DS       AL4(FREEAREA)
SAVEAREA DS       18F'0'
SAVEGR1  DS       01F'0'
SAVEMAIN DS       1F'0'
SAVEEND  EQU      *
SAVESIZE EQU      SAVEEND-SAVEBEG
         END
```

```
      SUBROUTINE $ERR1(ICORE)                                          $ERR1010
      DIMENSION ICORE(4)                                               $ERR1020
      PRINT 10,ICORE(1),ICORE(2),ICORE(4)                              $ERR1030
      FORMAT('ODYNA ERROR: INSUFFICIENT CORE FOR PROGRAM '//           $ERR1040
     1  ' ALLOCATED CORE AT ',Z8,' OF ', I6,' BYTES, BUT REQUIRED '    $ERR1050
     2  ,I6,' BYTES.'//)                                               $ERR1060
      RETURN                                                           $ERR1070
      END                                                              $ERR1080

      SUBROUTINE $ERR2                                                 $ERR2010
      PRINT 10                                                         $ERR2020
      FORMAT('ODYNA ERROR: INCORRECT PARAMETER LIST SETUP')            $ERR2030
      STOP                                                             $ERR2040
      END                                                              $ERR2050
```

BIBLIOGRAPHY

IBM System/360 Operating System, Assembler Language, C28-6514-5.

IBM System/360 Operating System, Concepts and Facilities, C28-6535-4.

IBM System/360 Opetating System Fortran IV (G and H) Programmers Guide, GC28-6817-1.

IBM System/360 Scientific Subroutine Package, (360A-CM-03X) Version III, Programmers Manual, H20-0205-3.

IBM System/360 Operating System, Supervisor and Data Management Services, C28-6646-0.