European Centre
for Medium Range Weather Forecasts

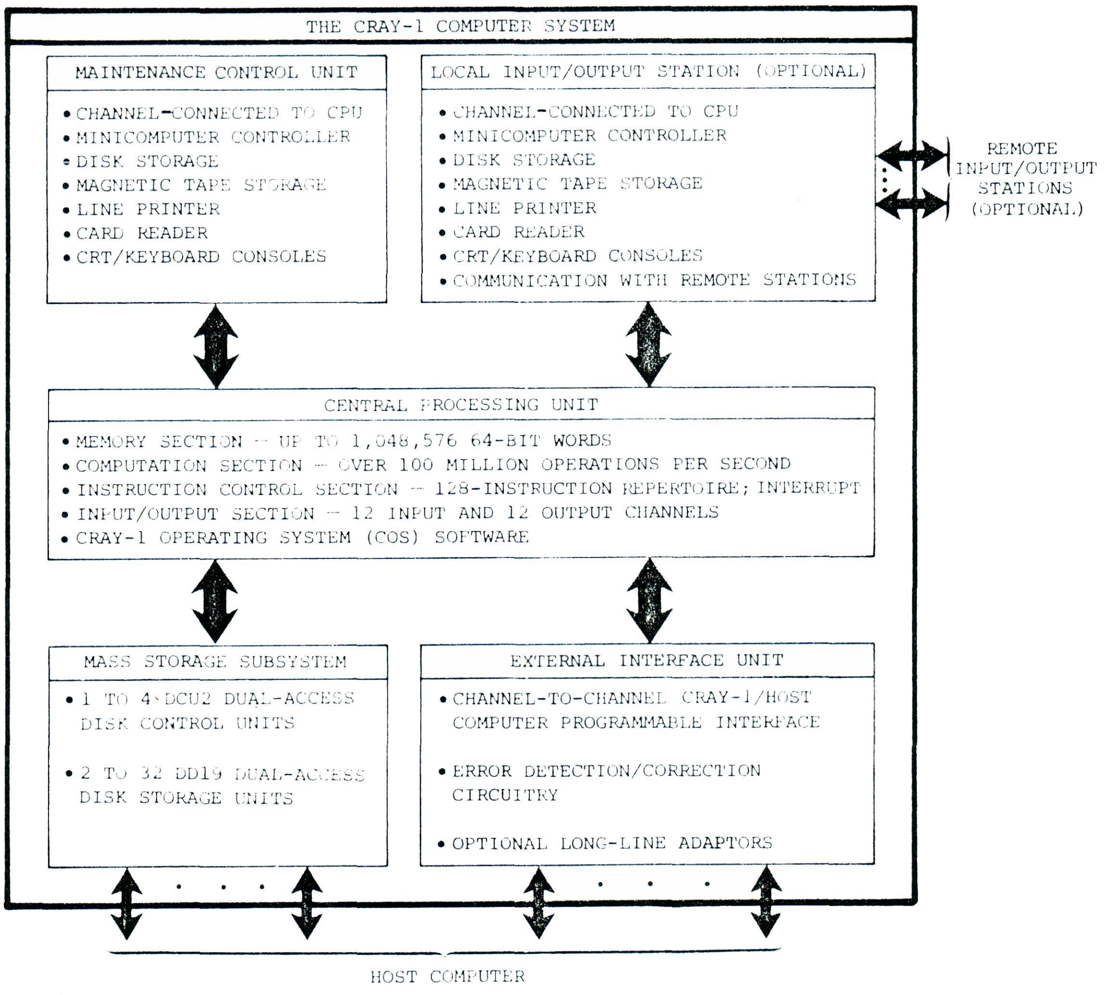
COMPUTER NEWSLETTER

Fitzwilliam House Skimped Hill Bracknell Berkshire England   Bracknell (0344) 24777 Telex 847908

In this issue: Number 4 - - November 1977

Serial-1 CRAY-1 Accepted on November 22	1
CRAY Acceptance Testing	1
The NEWS SHEET	2
6600 FTN Compiler Corrected	2
Q & A	3
Computer Division Seminars	4
CDC 6600 Performance Statistics	4
Ill-conditioning in FORTRAN Programs	4

Edited and Produced by User Support Section, Brandon House, ext. 286


THE CRAY-1 COMPUTER SYSTEM
MAINTENANCE CONTROL UNIT
• CHANNEL-CONNECTED TO CPU
• MINICOMPUTER CONTROLLER
• DISK STORAGE
• MAGNETIC TAPE STORAGE
• LINE PRINTER
• CARD READER
• CRT/KEYBOARD CONSOLES
LOCAL INPUT/OUTPUT STATION (OPTIONAL)
• CHANNEL-CONNECTED TO CPU
• MINICOMPUTER CONTROLLER
• DISK STORAGE
• MAGNETIC TAPE STORAGE
• LINE PRINTER
• CARD READER
• CRT/KEYBOARD CONSOLES
• COMMUNICATION WITH REMOTE STATIONS
REMOTE INPUT/OUTPUT STATIONS (OPTIONAL)
CENTRAL PROCESSING UNIT
• MEMORY SECTION -- UP TO 1,048,576 64-BIT WORDS
• COMPUTATION SECTION -- OVER 100 MILLION OPERATIONS PER SECOND
• INSTRUCTION CONTROL SECTION -- 128-INSTRUCTION REPERTOIRE; INTERRUPT
• INPUT/OUTPUT SECTION -- 12 INPUT AND 12 OUTPUT CHANNELS
• CRAY-1 OPERATING SYSTEM (COS) SOFTWARE
MASS STORAGE SUBSYSTEM
• 1 TO 4 DCU2 DUAL-ACCESS DISK CONTROL UNITS
• 2 TO 32 DD19 DUAL-ACCESS DISK STORAGE UNITS
EXTERNAL INTERFACE UNIT
• CHANNEL-TO-CHANNEL CRAY-1/HOST COMPUTER PROGRAMMABLE INTERFACE
• ERROR DETECTION/CORRECTION CIRCUITRY
• OPTIONAL LONG-LINE ADAPTORS
HOST COMPUTER

NEWSLETTER

Number 4, page 1
November 77

Serial-1 CRAY-1 Accepted on November 22

ECMWF accepted the serial-1 CRAY-1 interim system (installed at Rutherford Laboratories) on November 22. This system is now available for use by ECMWF and consists of

Cray-1 Computer (Serial 1) with 524288 words (64 bits) memory
DCU-2 Disk controller
4 DD-19 Disk drives
Data General Eclipse minicomputer subsystem.

This interim configuration will be in use until installation of serial 8 Cray-1 in October 1978 at the Shinfield Park site now under construction.

CRAY Acceptance Testing

Before a user service can commence on the CRAY-1, it is important that Cray and ECMWF agree that the machine is in a suitable state. In order to satisfy ourselves, a 24 hour acceptance test was agreed and has now been run.

asically the test comprises the following parts:

1. Two cycles, each lasting 2 hours when engineering test programs are run on the configuration to check that the hardware is performing to specification.

2. Several cycles, totalling 18 hours where user-type programs are run. The programs were drawn from the Milestone 1 suite and included the GCM and the GEM.

3. A demonstration that certain operational facilities, for example, operator control and permanent file dumping facilities exist.

The trial began at around noon on Monday 14 November, and the first 2 hour hardware test cycle passed without incident. The facilities demonstration was then run which showed that a minimum usable subset of facilities are available.

The software cycles were begun at around 16.30. Each cycle lasted around 2½ hours. In all, six cycles were run, operating through the night until around 09.30 on Tuesday 15 November. During these cycles, various problems occurred, the most significant ones being a bug in the scheduler which on two occasions stopped the CRAY from executing any jobs, and memory parities, one transient and one permanent. At the end of six cycles, it was clear that there were some problems which could not immediately be nswered and thus it was agreed that the trial would be suspended pending investigations y Cray staff.

By Thursday 17 November, it appeared that the machine was ready for retrial. The software problems had been removed and the memory problems had been resolved. However, 90 minutes into the first 2 hour hardware cycle we began to encounter disc problems. It was apparent that the problems would take some time to resolve so again the trial was suspended.

Cray then spent 3 days exhaustively testing the machine, and imported a hardware specialist to work on the disc problems. The problems were finally traced to the disc controller. The temperature sensing equipment on the disc controller had not been sounding the alarm, but it had been aboring the transfers. Unfortunately, the controller was running about 5^{o}F too warm. Opening up a refrigeration control valve about ½ a turn cured the overheating problem, and some work on a relay cured the alarm. These problems were finally cured at about 17.00 on Sunday 20 November.

Acceptance began at 19.00 on Sunday. This time there were few problems. Two system crashes (probably caused by memory parities) lost a total of 8 minutes, plus 21 minutes job rerun time. On each occasion, the permanent files and job queues were recovered by the system. A further two memory parities occurred which had no effect on the system or on any executing jobs.

Towards the end of the trial, one disc drive developed a solid hardware fault. The spare disc drive was quickly brought into service and little time was lost.

.../cont.

Despite these incidents, the overall availability was 95.8% which exceeded the acceptance criteria.

One final point, the trial included both scalar and vector versions of the GEM, compiled from identical sources but using the CFT V option. A CPU speed factor of 5.2 improvement of vector version compared with scalar version was observed.

-- Peter Gray

The NEWS SHEET

With the CRAY-1 service just starting at Rutherford, we are faced with the problem of providing up-to-the minute technical (and logistical) information to users and staff on short notice. The NEWSLETTER, with its monthly publication schedule, is inadequate for this kind of immediacy, and probably inappropriate for any detailed, technical information... especially that which is subject to change.

So, we have inaugurated a new, irregular publication, called the COMPUTER NEWS SHEET, which is produced jointly by the computer operations, systems, and user support sections, and contains immediate information pertaining to the computer service. The NEWS SHEET is an internal publication and is directed specifically to those staff now using the computers.

We anticipate that material appearing in the NEWS SHEET will eventually make its way into the NEWSLETTER or some other more permanent technical document.

In order to be able to issue NEWS SHEETS as quickly as possible, we have restricted the production and distribution to about 50 copies. This can easily be run off on the Xerox machine and distributed within an hour after the text is ready and typed. The NEWS SHEET will be sent to Rutherford on the next van out. We intend to keep folders at strategic places in all 4 ECMWF sites to hold NEWS SHEETS (and other technical information) for public reference.

The first NEWS SHEET was distributed November 23rd, and contained information on job and data transfer between the 6600 in John Scott House and the CRAY-1 at Rutherford, and on the courier and bus service between the two sites.

-- Richard Friedman

6600 FTN Compiler Corrected

The FTN compiler on the 6600 has been generating object code optimised for a 6400 rather that a 6600 cpu! This was due to a trivial error in the installation procedure that created the NOS/BE system on the 6600 a number of months ago. Although theoretically it should only have affected execution speeds at OPT=2, some numerical differences have been observed. These are under investigation, but are most probably due to ill-conditioning (see article "Ill-conditioning in FORTRAN Programs" in this issue, page 3).

Since the 6400 cpu does not contain the instruction stack or overlapped functional units of the 6600 cpu, the code generated for this cpu by FTN at OPT=2 will be strictly sequential code. When run on the 6600, this code will execute slower than code properly optimised for the 6600 cpu.

The discrepancy was discovered recently when one programmer noticed minor differences in numerical results between runs using the standard system compiler and the MANTRAP compiler. It was found that the MANTRAP FTN was properly generating code for the 6600, which differed from the code generated by the system compiler.

The system resident FTN was re-installed (targetted properly for the 6600) onto the running NOS/BE system on November 22. Anyone experiencing sudden problems should contact User Support at ext. 286.

--- R. Friedman, M. Lewis,
N. Storer

Q & A

The following are some of the more interesting queries received by User Support over the past few weeks.

Q. When I did the following by error

```
REQUEST,B,*PF.
CATALOG,B,F,ID=EWXX,SN=DSET1.
```

why didn't CATALOG abort since I forgot to put SN=DSET1 on the REQUEST? As a result, the system set rather than my private pack was used.

A. The problem is with CATALOG, but not what you supposed. Actually, there is no SN= keyword parameter for CATALOG (see NOS/BE Reference Manual, pages 4-10 to 4-12). CATALOG assumes that the SN= designation has been made on the REQUEST prior to it. You are correct that the error was that you forgot it on the REQUEST. That would have been sufficient. However, CATALOG should have aborted in your case above with a control card syntax error, rather than just ignoring the inappropriate SN= parameter and proceeding mindlessly on to use the system set. An error report (PSR 77/06) has been submitted to CDC about this.

(R. Friedman)

Q. COPYBR seems to position a file written by FORTRAN binary (unformatted) WRITE(N) at EOI, ignoring the record structure of the file. Why? And how can I selectively COPY records off such a file onto another?

A. The record structure of files written by the unformatted WRITE(N) statement in the default case (RT=W) is defined by control words imbedded in the file. This record type has the advantage of saving space on disk and tape... the entire file (including endfile marks) will appear as a single "logical record" on the disk. The problem is with the COPY utilities, which assume only one kind of record structure, RT=S, or "SCOPE logical record". COPYBR on such a RT=W file will copy the entire file up to EOI. This, we agree is a serious limitation of the COPY utilities, which should be made to handle both RT=W and RT=S files; i.e. they should recognize a FILE card. Until such a utility can be provided, users have only two rather poor alternatives:

a) specify RT=S when writing the file to be processed by COPY by using a FILE directive to change the record type before executing the FORTRAN program;

b) write a FORTRAN COPY program to READ and WRITE RT=W files in a manner suitable to your specific needs. Both are distasteful, and for some may be impossible. However, this is just one of the worst deficiencies in NOS/BE that we are currently facing.

(R. Friedman)

Q. How can I safely and gracefully abort a running program from FORTRAN?

A. On the 6600 with FTN, a subroutine called SYSTEM is provided such that

```
CALL SYSTEM (52, message)
```

where message is a hollerith character string (like 20H XYZ FOUND BAD DATA) will cause the message to be written to the DAYFILE, a traceback printed on OUTPUT, and the program aborted. Control passes next to an EXIT control card. However, caution must be observed on the CRAY. Although there currently is no routine to gracefully abort a program, there is a subroutine called SYSTEM, which does something else. Until a compatible routine can be provided on the CRAY, use SYSTEM only on the 6600.

(R. Friedman)

Please direct your questions, statements, complaints, etc. to User Support, Brandon House, Room 102, extension 286.

Computer Division Seminars

November 24	--	F. Königshofer	: "ECMWF's Telecommunication sub-system"
December 8	--	T. Stanford	: Cyber/Cray Spooling System
January 12	--	P. Gray	: Cyber/Cray Link
January 26	--	K. Petersen	: Graphics
February 9	--	R. Friedman	: CRAY CFT FORTRAN Compiler
February 23	--	M. Lewis	: MNF FORTRAN Compiler

Time and place for these meetings will be announced in the weekly ECMWF Calendar.

C.D.C. 6600 PERFORMANCE STATISTICS

WEEK ENDING	16/10	23/10	30/10	6/11	13/11	20/11
JOBS CENTRAL SITE	826	1033	810	806	1013	1052
JOBS REMOTE	180	225	200	207	301	394
PLOTS	132	99	35	76	73	97
C.P. HOURS	59.0	68.9	89.7	49.8	57.2	77.7
M.T.B.F. (hours)	56	84	33.6	28	168	42
Scheduled Availability	99	100	100	87.1(2)	100	100
Overall Availability	72.8(1)	93.7	93.8	66.1(2)	88.7	87.4

-- Eric Walton

(1) System failed at week-end during unattended operation
(2) Mainly resulting from power cuts.

Ill-conditioning in FORTRAN Programs

The following is adapted from an article that appeared recently in one of CDC's technical publications. It may be of particular interest to programmers now involved in converting production programs from one compiler and library to another (e.g., FTN to CFT). In some cases, as outlined below, over-sensitivity to compiler or math library changes may be indicative of bad programming practice, or the use of an improper computation algorithm. Procedures for detection and elimination of such oversensitivities are described in this article:

"The execution of a FORTRAN program is affected by a number of variables: the word-size of the machine on which the program runs, the code generated by the compiler, the presence or absence of rounding in the evaluation of arithmetic operations, the precision of the input data, the precision of storage of temporary results, the object-time routines employed, and any system-supplied data. If the output from execution of a FORTRAN program undergoes large changes when small changes are made in any single one of these variables, the program is called ill-conditioned with respect to that variable changed. In the case of interest, a program will be proved ill-conditioned with respect to the mathematical routines if traps in both old and new versions of some math routines indicate an acceptable relative error (say about 1.E-13 for single—precision, 1.E-28 for double precision) when run against the binary produced by the compiler, but the output shows an appreciably bigger variation.

We list some methods for detecting and eliminating ill-conditioning in programs.

1. Ill-conditioning due to inaccurate representation of input and intermediate results. This will be noticed when the program runs against different sets of data which are nearly equal, producing output differing appreciably more than input differences. Such ill-conditioning may be eliminated by increasing the precision of the data being input, or the precision of intermediate results.

2. Ill-conditioning due to inaccurate computation of arithmetical results. This is most conveniently detected by compiling the program, once with the FTN option ROUND=+*/ specified, and once without, then looking for differences in the program's output from execution. This type of ill-conditioning can arise in a variety of ways. One common way is the evaluation of a long sum to the same precision as each of its components, e.g., in the computation of inner products, variances. (The current FTN compiler inserts code inline for exponentiation with a constant fixed exponent between -16 and 16, and the multiplications are rounded or not according to the ROUND=* specification's appearance on the control card). Another common way is in the subtraction of nearly equal quantities, whose difference scales some output data. This may be important in certain iterations where the iteration count is a function of the data. Ill-conditioning arising from inaccurate differences may be checked by casting relevant variables and operations into higher precision.

.../cont.

3. Ill-conditioning due to the library of mathematical routines in use during execution. If the random number generator is called, check that the same initial seed and multiplier are being used. If the random number generator is not called, then changes in output across executions with two different libraries are due in almost all cases to one of the other forms of ill-conditioning affecting computation using a mathematical routine. To determine which routine is affected, substitute new math routines for old routines, one by one, when executing repeated runs with the same data. When the affected routine is detected, determine the occurrence affected by the ill-conditioning by replacing occurences of that function, one by one, with calls for the function in higher precision, during repeated executions with the same data. An example of ill-conditioning associated with the library of math routines is as follows. If a program calls for computation of cosine near pi/2 (or sine near pi), and the result is used to scale output quantities, the program may be ill-conditioned. The reason is that the function has a zero at the point, but the result of the function evaluated near the point may lose significance because of the great magnification of machine error there.

4. Classes of programs known to be ill-conditioned. Included here are the programs to invert matrices, solve systems of linear equations, extract roots of polynomials, etc. An adequate literature exists to enable the program's user to draw conclusions on the degree of significance of his output, for most cases here. Certain thechniques may be safer than others, even though the cost in execution time is higher. For example, the Crout reduction is preferred (for error control) to the Gauss-Seidel reduction in the inversion of matrices, although the Gauss-Seidel is quicker on all programs. Other methods of detecting ill-conditioning in such programs include replacement by equivalent expressions, and forward error analyses.

We give some examples of FORTRAN sequences where ill-conditioning can occur.

<u>Example 1.</u> (Insufficient precision in a stored constant).

```
      PRINT 10, (I-1, TAN((I-1)*3.1416/180.),I=1,90)
   10 FORMAT(1H1,10X,*TANGENT FOR ANGLES 0 TO 90 DEGREES*//
      + 11X,*ANGLE*,SX,*TANGENT*//(11X,13,7X,F10.4))
```

In this example, pi is given (as 3.1416) to insufficient accuracy, since TAN has a singularity at pi. The approximation of 3.141562654 lowers the relative error of the last output item by several orders of magnitude. Instead, write

```
      PRINT 10, (I-1,TAN((I-1)*3.141592654/180.),I=1,90)
```

<u>Example 2.</u> (Insufficient precision in temporaries).

```
      REAL, SX,SXX,FCN
      SX=SXX=O.
      DO 10 I=1,100000
      X=FCN(I)
      SX=SX+X
   10 SXX=SXX+X*X
      AV=SX/100000.
      SD=SQRT(SXX/100000.-AV**2)
      PRINT 20,SD
   20 FORMAT(11X,E17.10)
```

This example of a sequence to compute standard deviations may be ill-conditioned from two causes. First, the accumulation of running totals in SX and SXX may allow machine error to build up to significant levels. Second, if the standard deviation SD is small compared to the mean AV, significant accuracy loss may occur in the evaluation of the expression SXX/100000-AV**2, which is then small in comparison to subtrahend and minuend. A better sequence is

```
      DOUBLE SX,SXX
      REAL FCN
      SX=SXX=O.
      DO 10 I=1,100000
      X=FCN(I)
      SX=SX+X
   10 SXX=SXX+X*X
      AV=SX/100000.
      SD=SQRT(SNGL(SXX/100000.-DBLE(AV)**2))
      PRINT 20,SD
   20 FORMAT(11X,E17.10)
```

.../cont.

Example 3. (Precision loss in arithmetic operations).

```
      REAL MX,MY,VX,VY,MASS
      READ*,SPEED,APEX,MASS,NO
      DO 10 I=1,NO
      ANG=APEX*(RANF(O)*2.-1.)
      VX=.5*MASS*(SPEED*(1.-COS(ANG)))**2
      VY=.5*MASS*(SPEED*SIN(ANG))**2
      MX=MX+VX
   10 MY=MY+VY
      PRINT *,MX,MY
```

For values of APEX less than 1.E-7 in absolute value, any precision of MX is completely lost. This occurs in the evaluation of the expression 1.-COST(ANG), since the small x, cos(x)=1.-x^2. However, some equivalent expressions preserve precision. For instance:

```
      REAL MX,MY,VX,VY,MASS
      READ *,SPEED,APEX,MASS,NO
      DO 10 I=1,NO
      ANG=APEX*(RANF(O)*2.-1.)
      VX=.5*MASS*(SPEED*2.*SIN(ANG*.5)**2)**2
      VY=.5*MASS*(SPEED*SIN(ANG))**2
      MX=MX+VX
   10 MY=MY+VY
      PRINT *, MX,MY              ..."
```

-- Richard Friedman