H. Melenk W. Neun

REDUCE User's Guide for the Cray 1/X-MP Series running COS

(Version 3.3)

Technical Report 87-5 (August 1987)

Konrad-Zuse-Zentrum für Informationstechnik;
Heilbronner Straße 10; D-1000 Berlin 31

TABLE OF CONTENTS

Abstract

This document describes operating procedures for running REDUCE specific to the CRAY 1 and CRAY X-MP computers running the Cray Operating System (COS). The document was derived from the corresponding document for Vax/UNIX prepared by A. C. Hearn and L. R. Seward, The Rand Corporation, Santa Monica, (CP85).

1. PRELIMINARY

This document describes operating procedures for running REDUCE specific to the Cray 1 / Cray X-MP computers with the COS operating system. It supplements the REDUCE User's Manual, describing features, extensions and limitations specific to this implementation of REDUCE.

The character ! (exclamation mark) is used in this document to represent the REDUCE syntax escape character. By using REDUCE on a Cray computer via a station computer this character (and others too) may have a different representation due to local code conversions. Please look at the special character table in the REDUCE User's Manual.

The modules that form the REDUCE system are stored in a number of files. The main entry to REDUCE is the executable file named "REDUCE", which is stored under system ownership and public access. At runtime the system needs access to REDUCE and LISP libraries. This access is achieved in an automatic and for the user invisible manner. To start REDUCE, simply use the command

REDUCE.

in your batch job or interactive session, after which REDUCE will respond with a banner line. The system then expects its input via the standard file $IN which normally is connected to the batch job data records or to the interactive terminal.

Example for a batch REDUCE job:

```
JOB,xxxxxx,MFL= 320000,T= 1.
ACCOUNT,xxxxxxxx.
REDUCE.
/EOF
p:= (x+ y)**10;
bye;
/EOF
```

note: the lower characters are converted to capital characters on input automatically by default (raise is on).

2. REDUCE DOCUMENTATION

For proper usage of REDUCE, the document

A.C. Hearn:
REDUCE User's Manual

should be consulted. As an addendum to the User's Manual, a number of modules first introduced by REDUCE 3.3 is described separately. If features of the underlying LISP system are to be used, the documents

Utah Symbolic Computation Group:
The Portable Standard LISP Users Manual

and

H. Melenk, W. Neun:
Portable Standard LISP Implementation for Cray X-MP computers

may be needed. All these documents are distributed by Konrad Zuse- Zentrum, Berlin. There is no on-line documentation available.

3. AN INTRODUCTION TO REDUCE

New users of REDUCE are advised to process the seven REDUCE Lessons, which should be used via a REDUCE implementation on a timesharing computer or a workstation (e.g. a VAX or a SUN). They are not available on line with Cray computers.

4. RESOURCE REQUIREMENTS

The minimum field length of a job running REDUCE is

MFL= 320000

In this field length the basic REDUCE features can be loaded; it includes a heap of total length 50000 words with about 40000 words available as free working memory. If additional modules of REDUCE are loaded at runtime or if due to the problem size a larger heap is needed, the MFL parameter has to be modified.

The areas for heap (data memory), bps (area for compiled code) and binding stack (BNDSTK) can be enlarged up to the limit defined by the MFL parameter.

The enlargement of binary program space while loading additional modules or compiling user programs is done automatically (the action is protocoled via $OUT).

If a larger heap is needed because the number of garbage collections is too high, the LISP function SET-HEAP-SIZE has to be called. This function expects the desired TOTAL heap length as parameter. It is called from REDUCE e.g. by

```
LISP SET!-HEAP!-SIZE 100000; % set heap to 100000;
```

(note the exclamation marks in front of special characters inside the name.) The heap size is adjusted in case of a lack of data area too, but this will happen if the heap is almost full, after many garbage collections. So we recommend to adjust heapsize when big data areas are needed.

The actual total heap size can be seen as value of the LISP variable

```
LISP heapsize;
```

The PSL parameters set by REDUCE normally condemn the garbage collector to do his work in total silence. This is necessary because asynchronous messages can destroy a handsome output image. In order to learn the amount of occupied heap space or in order to see at which points of execution the garbage collector is involved, simply turn on the switch

```
ON GC;
```

With GC on the garbage collector will be verbosed. If you only want to know the number of garbage collections during your calculation you ask for the value of the LISP variable

```
LISP GCKNT!*;    % number of garbage collections since start;
```

It would be a good strategy to inspect this variable after a large computation has been done the first time and to adjust the heap size so that this value remains in a reasonable range. For memory estimates: the underlying PSL uses two words to store one list element (= two pointers).

In case of stack overflow the stack size can be modified in a corresponding manner by calling the function SET-STACK-SIZE. The actual size of the stack is the value of the LISP variable StackSize. A similar function SET-BNDSTK-SIZE is supplied for BNDSTK.

IMPORTANT: when estimating the MFL the area needed by the local files have to be taken into account. Each file local to a job requires 4 K words of memory.

5. FILE HANDLING

For a detailed treatment of files consult the PSL documentation mentioned above. For simple applications the following notes may be sufficient:

—Use only filenames with capital letters and digits. With this rule quotation marks around filenames are superfluous. File names written without quotation marks are converted to upper case automatically.

—Save datasets with ID= PSL.

—Input of files which are saved already with your ownership and ID= PSL: Do not ACCESS the file; use the PDN as filename for opening the file (REDUCE will do an implicit ACCESS to connect the file).

```
REDUCE.
/EOF
in yesterdaydata; % read saved file with pdn specified;
```

—Input of local files or files already accessed:
Use the FN as filename for opening.

```
ACCESS,DN= TEMP,PDN= TOMORROW,OWN= FRIEND.
REDUCE.
/EOF
in temp; % read file using a local dn;
```

—Output of files:

Files opened output will be created as new files and will be saved as permanent files with your ownership (ID= PSL) and released when closed.

—Staging of files:"

The COS commands FETCH,DISPOSE,ACQUIRE,ACCESS,SAVE,RELEASE and some more can be invoked directly from REDUCE by using the LISP function COS-CMD (parameter is the full COS command enclosed by LISP string quotes), e.g.

```
LISP COS!-CMD
 "FETCH,DN= NEWDATA,TEXT= 'DSN= IBM.PDS(MEMBER),DISP= SHR'." ;
```

note the single quotes inside the string; they are parts of the COS command language.

6. INTERNAL PARAMETERS

6.1 Object Sizes

The maximum string and identifier lengths are limited only by the underlying PSL base. The current implementation allows several hundred characters in both identifiers and strings. However, we recommend that such names be limited to 24 characters or less for compatibility with other versions of REDUCE.

All fixed precision floating point numbers are printed in FORTRAN's "G23.8E4" format by default. This format may be changed as described in the PSL documentation.

Arbitrary precision integer and real arithmetic is supported.

6.2 Special Characters and Interrupts

Lower case input is permitted, but converted to upper case unless the switch RAISE is off.

^ may be used as an alternative to ** in expressions. This character is represented by the character "not" with some frontend computers.

The end-of-file is supplied by the EOF command.

If a terminal interrupt occurs (command ATTN), the current calculation is canceled. REDUCE immediately prompts for the next command. There is no means to continue the interrupted calculation. The LISP function USER!-ERROR!-Function is called in recovery procedure. It maybe redefined for own diagnostics. The error-code is the only parameter, e.g. 70 for terminal interrupt.

} is used to terminate strings in the REDUCE interactive editor, because ESC as in other implementations will not be handled via all stations.

6.3 Miscellaneous

There is no link currently to an external editor.

The internal ordering on alphabetic characters is from A through Z followed by a through z.

Times (as reported by ON TIME or SHOWTIME) are given in milliseconds, and measure execution time including garbage collection time. They do not include operating system overhead.

To exit REDUCE use "bye;" .

Per default there is echoing of input in batch jobs only. Echoing is controlled by the switch

on echo;

With echo turned on, the input is echoed to the output and a mixed protocol of incoming commands and outgoing results of their execution is produced.

In batch jobs the switch int is set to off by default. This switch indicates the batch/interactive status to the REDUCE system.

With int turned off REDUCE will generate no interactive requests and will stop algebraic execution when the first error occurs. To avoid this, you can use ON INT; even in batch jobs.

7. CUSTOMIZING THE REDUCE ENVIRONMENT

Implementation deferred. Datasets needed at startup time have to be processed explicitly.

8. IMPLEMENTATION DEPENDENT MESSAGES

A number of messages from the underlying PSL system and the Cray machine interface may be seen from time to time. These include error messages and informatory messages.

AB053 FLOATING POINT ERROR
Probably means a division by zero has been attempted

BPS will be automatically enlarged ... (Information)
BPS size is increased because a load module was needed by your application, e.g. the PSL compiler.

Heap space will be automatically enlarged (Warning)
Your problem is too large in its present form for the available workspace; either change your problem formulation or enlarge your heap (see above)

Non-numeric argument in arithmetic

This means that a LISP arithmetic routine has been called with an invalid argument

Hint: If you do not know what happened use
TR CONTINUABLEERROR; and run again.

AB054 OPERAND RANGE ERROR

This indicates an illegal memory reference. It can arise from applying the LISP function CAR to an atom in compiled code.

AB023 JOB TIME LIMIT EXCEEDED

The time parameter of the JOB card was too small for your batch job or you have coded an endless loop.

Stack overflow

The PSL stack has overflown. If your application needs a larger stack, please enlarge it via

Set!-Stack!-size < Size> ; call

Binding stack overflow

The PSL stack for special variables has overflown. If your application needs a larger BNDSTK, please do

Set!-Bndstk!-size < size> ; call

A hint: if you want a backtrace in case of an ABnnn error, please type
SYMBOLIC PROCEDURE USER!-ERROR!-FUNCTION(x); INTERPBACKTRACE[]
and run again.

9. FURTHER HELP

For further help with the Cray implementation of REDUCE, please contact:

Konrad-Zuse-Zentrum, Berlin

e mail: zb6260 @ db0zib21 . bitnet