AIPS MEMO NO. 3

ADAPTING RANCID TO THE U. OF MINN. CDC CYBER 74

-- a progress report

by Frank D. Ghigo

September 23, 1980

INTRODUCTION

I obtained a copy of the MODCOMP version of RANCID from E. Greisen during the last week of April 1980. The program consisted of about 130 routines. At present, I have gotten a subset of 70 subroutines to run. These include the basic POPS language implementation. I have ignored, up to now, all of the routines called by the VERBS subroutine.

The adaptation of the input and output abilities went fairly smoothly by writing the appropriate Z-routines, much as the authors of RANCID intended. But considerable changes were required to take care of character handling, and a few changes were required by the lack of I*2 and R*4 variables. In most cases, I was able to rewrite sections of non Z-routines in a way that makes them now independent of the number of characters per word and the relation between integer and real word size. Although RANCID was meant for byte-oriented machines, the changes I have made to adapt to arbitrary word length may be useful to install in the official version. The DEC-10, for example, has five characters per word (36-bit words), and many of the problems I have solved on the Cyber can probably be solved the same way on the DEC-10.

There are still some things which seem impossible to make machine independent, most notably many format statements, and some array size declarations, as discussed later.

There is also be a problem in POPS which may be a real bug in the program: hollerith constants cannot be passed through a procedure call if the procedure contains a RETURN statement.

I have been working on this off and on for the last $3\frac{1}{2}$ months. Probably I have spent the equivalent of about 6 full weeks on it. I regard this as a pretty speedy job, and would like to point out a number of circumstances which have been quite helpful in deciphering RANCID and adapting it to the Cyber 74.

1. There was adequate internal documentation which consisted of, at the beginning of each routine, a statement of its function and a description of the calling parameters, and during the routine, there were comments identifying the function of each group of statements. This made it fairly easy to follow what was going on. The documentation, though good, could still stand some improvement. Many of the routines make extensive use of parameters passed through common. These parameters and their functions should be explained at the beginning of each routine. Most of the comments in the body of the routines are overly terse.

2. The great modularity was helpful. Few routines are longer than a page or two, which makes it easier for us feeble-minded folk to encompass them.

3. Each routine follows good "Structured programming" practice with a fairly easy to follow flow of logic and no wild jumping about, except for the final leap for the exit. Each routine is structured similarly, with a standard way of exiting, whether a normal or error exit.

4. The extensive error checking and error traceback features built into RANCID were indispensable for debugging. Also, the debug option, which gives extra printing in a number of places, was very useful.

5. Sume's POPS manual for the Onsala system was extremely important for learning the organization of the K-array and the data blocks within it, which are, of course, the heart of POPS. It was especially useful that I had copied Eric's copy of Sume's manual which had handwritten addenda indicating the new symbol types that were not in the older POPS.

6. I had some previous experience with POPS in trying to puzzle out a few things in the Cyber version of TPOWER/SPOWER which George Martin installed at the U.of Minn. last year. A long conversation with G. Martin was informative in this regard. In a few cases the Cyber version of TPOWER/SPOWER provided some useful clues for putting RANCID on the Cyber. For the most part, I went in a different direction in the way I set up the I/O handling and character manipulation.

7. Since I am fairly well versed in the Cyber system, it was easy to write the appropriate machine-dependent parts, once I figured out what was needed.

CHARACTERISTICS OF THE CDC CYBER 74 WHICH WE HAVE TO RECKON WITH

It has 60-bit words. Fortran uses the same 60-bit word length for both real and integer variables. I*2, R*4, etc., are unheard of. The Cyber uses octal as the internal representation. It doesn't understand hexadecimal constants. The internal character set uses a 6-bit code, so that there are ten characters per word. There is an extended character set using an intermixture of 12-bit and 6-bit characters which I have avoided like the plague. Fortunately all the characters POPS needs seem to be in the basic set of 64 characters. So don't add any exotic characters, OK? Files are fairly simple entities, each having a unique file name of 7 characters. Extension files, version numbers, and members of files are unknown concepts.

I/O WOES

If one makes use of the Fortran read and write routines on the Cyber, the files must be named in the PROGRAM statement and of necessity remain open and exclusive for the duration of the program. The alternative is to use various subroutine calls to the Cyber Record Manager (CRM) which allow opening and closing of files which may be sequential or random access. Files handled by Fortran cannot be accessed by CRM calls, and vice versa. In VLAGEN, and possibly other places, files were opened with ZOPEN or ZTOPEN and then Fortran read or write statements were used. Such a mixture is not permitted on the Cyber.

It was clear at an early stage that the message writing facility, MSGWRT, was all pervasive. Initial attempts to decipher MSGWRT and its attendant Z-routines led me to suspect the work of a madman. In order to make progress

in the early stages, I made the log file and the terminal into regular Fortran files so that MSGWRT became simply two write statements and two IF statements to select whether writing on the terminal, log file, or both is to occur. Perhaps someday I will put back in the more elaborate features of MSGWRT if I figure out what they are.

It would have been nice to have had better documentation in the Z-routines handling I/O. One needs to know how the MODCOMP I/O calls worked in order to put the equivalent functions in their place. An explanation of the FTAB and FDL tables might have been enlightening. Also helpful would have been an explanation of RANCID's general file philosophy. What is the intended use of the four types of files? (i.e. those with DEVTAB= 0,1,2,3) And how do these types relate to the different groups of file handling routines? Apparantly, DEVTAB=0 files are disk files which may either be binary files handled by routines ZOPEN, ZCLOSE, and ZFIO, or they may be text files handled by ZTOPEN, ZTCLOS and ZTREAD. DEVTAB=1 files seem to be files handled entirely by the standard Fortran calls, and DEVTAB= 2 files I suppose are double-buffered files handled by ZOPEN, ZCLOSE, and ZFIO, which I have not yet implemented. DEVTAB=3 files seem to be unused, whatever they are.

To implement the file handling routines based on the CRM calls, I made the FTAB table into the file information tables required by the Cyber. This may be parallel to their original use.

To implement the ZTREAD, ZTOPEN, ZTCLOS functions in which subsets of files, known as "members" are read, I defined a member of a file to be a portion of a file delimited by "end of record" marks, which are special symbols allowed by the Cyber in text files (since each line of a text file is also known as a record, this is a confusing nomenclature). I required a member name to appear as the first line after an end of record mark. Thus, a call to ZTOPEN will look for the first occurrence of the member name following an end of record mark. Subsequent calls to ZTREAD will read lines following that "member identifier" until the next end of record is encountered.

CHARACTER CAPERS

At first it seemed as if I could continue to put only 2 characters in each word and get away with few changes. But this is incompatible with the usage of ENCODE and DECODE which pack or unpack characters densely. For example, every ENCODE of a message to be printed would have to be decoded with a 40A2 before printing, so that it could be printed with a 40A2 format. Encoding and decoding is done so often that I decided to change things so that all character strings would be packed ten to a word. This required a number of changes, but it has the advantage that much less storage space in the K-array is required for symbols and character strings. Many format statements had to change from nA2 to mA10 as a result. Places where character strings were moved from one place to another by simply copying arrays had to be changed so that the subroutine ZMOVE moved the characters one at a time. These places occurred in the PRINT verb (subroutine QUICK) and the INPUTS verb (subroutine HELPS). As a result of these changes, RANCID is now less machine-dependent. This was done by introducing new variables in a common block (COMMON/CHARAC/) and using the variables in many places. They are NWPL (number of words per line on a CRT screen) and NCPW (number of characters per word). With these values set to NWPL=8 and NCPW=10, the program works on the Cyber, and with NWPL=40 and NCPW=2, it would work on the MODCOMP or VAX, assuming the appropriate Z-routines are substituted.

It might have been less work to have kept the 2-character-per-word structure and to have put in the extra DECODES, but doing it the way I have gives a more compact K-array, which may be a significant advantage in the long run.

Another problem was two cases in which a comparison of character strings was done by comparing 2-character words. One case occurred in VLAGEN in which a test for a comment line was done by comparing the first word of the input line with the string "C-". This clearly doesn't work if there are ten characters in the word. I therefore wrote a new Z-routine, called ZGETWD, to extract 2 characters from a word and put the two, left justified, into a different word, so that it would be directly comparable to the words it was to be compared with. This was also needed in the HELP verb, for matching the argument of the HELP command with the strings "VERBS", "ADVERBS", etc.

And, finally, some of the message generating parts required no changes at all. For example, messages in OERROR are set in data statements and printed with nA4 formats. Also, the error traceback required no alterations because although the subroutine names were set as 2-character per word strings, they were simply printed as is with nA2 formats.

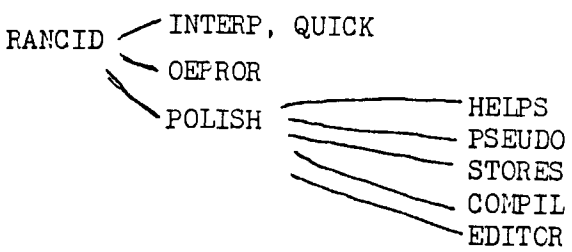
WORD-LENGTH WORRIES

Because integers are the same size as reals in the Cyber rather than half the length, a number of problems with equivalencing integer with real arrays were encountered. One problem which I do not see how to fix in a machine-independent way involves the equivalencing of the adverb names to their locations in the K-array, which is done by means of the COMMON/CORE/ declaration in HELPS and AU1, AU2, etc. For the real values there is no problem, but the string variables are declared as integers, so their size declaration must be halved for the Cyber version. The machine-independent solution would be to require all adverbs to be real-sized variables and arrays.

Most of the potential problems with using the K-array as both an integer and a real array are taken care of by the use of the functions IREALP and IINITP. These two should be Z-routines. For the Cyber, I altered them to return the same number they were given. That is, IREALP(IP) = IP and IINITP(IP) = IP. IREALP is called in four places and the above definition works in two of them (in subroutines ASSIGN and SYMBOL). However, in the routines LTSTOR and INIT, the wrong address is calculated unless IREALP(IP) = IP + 1. I took care of this in a quasi-machine-independent way by the introduction of a fudge parameter (in COMMON/CHARAC/) to be added to the appropriate address in LTSTOR and INIT, and whose value would be set appropriately for whichever computer is being used. It would be nice to think of a more elegant way to handle this problem.

OVERLAYING

To make RANCID fit in the 25K word space allowed for Cyber interactive jobs, I have overlayed the program as follows:



In this form, the most core is needed when INTERP and QUICK are loaded, and that amount is 15K words. Of course, things will become tighter when VERBS and all the things it calls are installed. One should note that I decreased the required core considerably by making the size of LISTF be 1024 and the size of the K-array be 3072, rather than 2048 and 7680. These lower sizes are reasonable since they refer to 60-bit words.

A FEW RANDOM SUGGESTIONS

1. The equating of string constants seems overly limited since only strings defined with identical lengths can be equated with each other. Surely it should always be OK to equate, e.g., STR1*n = STR2*m if n > m. And the case of m > n should be allowed with perhaps an error message saying "string truncated".

2. It would be nice to be able to SCRATCH adverbs as well as procedures.

CATALOG OF CHANGES MADE TO NON-Z ROUTINES

Most of these were required to make the program work on the Cyber. Some, however, were done merely for aesthetic or other metaphysical reasons. I have not detailed changes in RANCID and VLAGEN since these main programs are expected to be a little machine-dependent.

comments	is changed routine now machine-independent?
COMMON/CHARAC/ installed in many routines. this common includes NWPL,NCPW,NIPR,IFUDGE (= # words per line, # char.per word, # integers per real, and the fudge parameter) values for Cyber = 8,10,1,2 values for MODCOMP/VAX = 40,2,2,1 These values are initialized in RANCID. They cannot be initialized in INIT because RDUSER needs them. -- routines needing COMMON/CHARAC/ : RANCID VLAGEN FILZCH FRMT GETSTR HELPS INIT IWPC IWPR KPACK LTSTOR MSGWRT OERROR PREAD PSEUDO QUICK RDUSER SPFIL SYMBOL UNPACK	yes
All calls to UNPACK -- The call: CALL UNPACK(40, JBUF, KARBUF) now becomes: CALL UNPACK(NWPL,JBUF,KARBUF)	yes
Falling through computed GO TO statements On the Cyber, unnumbered statements immediately following a computed GO TO statement are unreachable. I thus preceded the GO TO statements with an IF statement to check whether the index was in the allowed range, and if not, to jump to the immediately following code, which was usually an error message setup.	yes?
Declarations changed REAL V(40) became V(60) and X(5) became X(10) in many routines, since these are equivalenced to integer arrays. -- routines affected: (routines in parentheses have these arrays defined but do not use them) (ASSIGN) BCLEAN COMPIL EDITOR GETFLD (GETNAM) (GTLINE) HELPS INIT INTERG INTERP LTSTOR (MASSGN) (OERROR) (POLISH) PSEUDO QUICK STORES SUBS (SYMBOL)	no

CATALOG OF CHANGES (continued)

comments	is changed routine now machine-independent?
Quote Symbol -- Cyber Fortran quote symbol for delimiting hollerith strings is " not '. Thus all single quotes from C-ville were changed to double quotes. But POPS quote symbol was and should be kept as a single quote, which entailed slight changes in the data statements which set constants containing the quote symbol. Appropriate changes were made to the following items in these subroutines:	no

variable	subroutine
IQUOTE	GETFLD
IQUOTE	GETSTR
IOPNQ	HELPS
ICLSQ	HELPS

Format Statements -- numerous A2's became A10's. I'm not going to list them all.		no

subroutine	comments	is changed routine now machine-independent?
FILZCH	installed COMMON/CHARAC/. loop now runs: DO 10 J= 1 , NCPW	yes
FRMT	installed COMMON/CHARAC/, code altered to make use of NCPW. Also the DATA statement setting hexadecimal constants was changed to the equivalent Cyber form (octal constants). Thus my changed data statement is machine dependent, but this could be make independent by setting these constants in regular hollerith form and using ZGTBYT to extract the wanted characters in right-justified form.	yes
GETSTR	installed COMMON/CHARAC/ , code altered to make use of NCPW, etc.	yes
HELPS	-verb INPUT- moving of text now done by character, not by word. also, in the printing of the INPUTS header, we now generate the header by ENCODE statements, rather than by copying arrays HEAD and IDASH to MSGTXT.	yes
HELPS	-verb DEFAULT- I made the file name read by the DEFAULT command dependent on the user number, so each user can have his own default file. The file name was formed thus: DEvssvv, but is now formed thus: DE10nnn, where nnn = user ID number.	

CATALOG OF CHANGES (continued)

subroutine	comments	is changed routine now machine-independent?
HELPS	-verb HELP- To compare IXMEM with TYPNAM (this is how we figure out which HELP listing to do) IXMEM first had to be torn apart into 2-byte words by use of the new Z-routine ZGETWD.	yes
HELPS	Halved the array sizes for the integer arrays that appear in COMMON/CORE/. This is required so that the adverb name in COMMON/CORE/ contains the correct value which was put into the K-array earlier.	no
IINITP	Changed to IINITP = IP. This should probably be a Z-routine.	no
INIT	installed COMMON/CHARAC/. The two statements involving IREALP changed to: TAG = IREALP(L) + IFUDGE	yes
	The constants ITRUE and IFALSE were not recognized while hunting the K-array, if they were set as 2-character-per-word arrays. Changed to: DATA ITRUE, IFALSE / "TRUE", "FALSE" /	no
IREALP	Changed to IREALP = IP. Should probably be a Z-routine.	no
IWPC	COMMON/CHARAC/ installed. Now reads: IWPC = (IP-1)/ NCPW + 1	yes
IWPR	COMMON/CHARAC/ installed. Now reads: IWPR = NIPR*IP	yes
KDUMP	This should be a Z-routine since the optimum way to dump an array depends on the machine.	no
KPACK	COMMON/CHARAC/ installed. Program altered to make use of NCPW.	yes
LLOCAT	The requirement that all blocks in the K-array occupy an even number of words was removed, to allow denser use of the K-array.	no
LTSTOR	COMMON/CHARAC/ installed. Statement 30 changed to : TAG = IREALP(L) + IFUDGE	yes
MSGWRT	Somewhat simplified, as mentioned under I/O Woes. I also surpressed the annoying "RANCID:" beginning of each line, though leaving this prefix for error messages.	no

CATALOG OF CHANGES (continued)

subroutine	comments	is changed routine now machine-independent?
OERROR	COMMON/CHARAC/ installed. NWPL, NCPW inserted in appropriate places. Format 1000 changed to (3A4,8A10)	yes no
PREAD	COMMON/CHARAC/ installed. NWPL inserted in UNPACK call.	yes
PSEUDO	COMMON/CHARAC/ installed. The setting of the string size is now more general: NSIZE (= number of real words needed to hold the string) = (NCPW-1 + INT(C(TAG)))/(NCPW*NIPR)	yes
QUICK	COMMON/CHARAC/ was installed.	
	-PRINT verb- KARLM1 set to 72, not 32, and all moving of text is done by ZMOVE (i.e., by character) rather than by COPY. Numerous counters were changed to facilitate this.	yes
	-DUMP verb- Most of the code in this section was replaced by a call to KDUMP.	yes
RDUSER	COMMON/CHARAC/ installed, and NWPL, NCPW were put in.	yes
SPFIL	COMMON/CHARAC/ installed, etc.	yes
SYMBOL	COMMON/CHARAC/ installed. Setting of TAG under "variable storage" section is now: TAG = LLOCAT(NIPR,K(KXORG),KXLINK) + KXORG - 1	yes
UNPACK	COMMON/CHARAC/ installed, etc.	yes