1958.06.11

TRANSOAP---SOAP III MODIFICATION (PUTS SUBROUTINES INTO "NORMAL FORM.")

THE USE OF TRANSOAP IS EXACTLY THE SAME AS THE USE OF SOAP III WITH THE FOLLOWING EXCEPTIONS:

1. ALL ADDRESSES WHICH ARE NOT TO BE TRANSLATED MUST BE ABSOLUTE, ~~REGIONAL~~, OR TAGGED WITH AN F.

2. THE PSEUDO-OPERATIONS OFF, ONN, ONE, FIV, BLR, BLA, AND PST ALF WILL NOT WORK IN THIS MODIFIED VERSION.

3. PRECEDE THE DECK TO BE TRANSOAPED WITH CARDS HAVING PSEUDO-OPERATION YYY AND HAVING THE CALLING NAMES OF THE SUBROUTINE (E.G., Q21 OR QSIN) PUNCHED ONE TO A CARD, HAVING THE Q IN COLUMN 56 AND THE REST OF THE LETTERS (EXCLUDING THE FINAL E OR F) PUNCHED IN THE FOLLOWING COLUMNS WITHOUT BLANK SPACES.

4. THE LAST YYY CARD MUST BE FOLLOWED BY A CARD WITH PSEUDO-OP 999 AND ALL OTHER ADDRESSES BLANK.

5. ALL 800X INSTRUCTIONS MUST BE PLACED ON TYPE 2 CARDS.

6. INCLUDE A CARD FOR EVERY TEMPORARY STORAGE LOCATION WHICH IS TO BE TRANSLATED. THIS IS IMPORTANT AS IT IS USED IN DETERMINING THE NUMBER OF LOCATIONS USED BY THE SUBROUTINE.

7. SUB-SUBROUTINES: ALL SUB-SUBROUTINES TO BE "DUP"D MUST BE PLACED AT THE VERY END OF THE SUBROUTINE. THEY MUST BEGIN WITH A DUP CARD WHICH HAS A D-ADDRESS OF ^^ABC (WHERE ABC IS ANY IDENTIFICATION FOR THE SUB-SUBROUTINE AS USUAL) AND AN <u>I-ADDRESS</u> WHICH IS THE LOCATION ADDRESS OF THE FIRST INSTRUCTION. (IF THIS IS A PROGRAM POINT A FORWARD PROGRAM POINT IS USED ON THE DUP CARD.) THE SUB-SUBROUTINE MUST END WITH A TYPE 3 CARD. DUP AND TYPE 3 MUST ~~[illegible]~~ NOT BE USED FOR ANY OTHER PURPOSE. DON'T NEST SUB-SUBROUTINES.

8. THE FIRST LOCATION OF THE SUBROUTINES AND SUB-SUBROUTINES MUST BE A PROGRAM ~~[illegible]~~ POINT OR SYMBOLIC ADDRESS.

9. THE ENTIRE DECK MUST BE FOLLOWED BY A BLANK CARD.

10. ON THE FIRST PASS THROUGH SOAP NOTHING WILL BE PUNCHED EXCEPT THE NUMBER OF LOCATIONS NEEDED BY THE SUBROUTINE. THIS WILL APPEAR IN THE L-ADDRESS OF A DUP CARD FOR EACH SUB-SUBROUTINE AND A 999 CARD FOR THE MAIN ROUTINE. SUBSTITUTE THESE CARDS THEN FOR ALL CORRESPONDING DUP CARDS AND FOR THE 999 CARD. THE SECOND PASS WILL YIELD THE DESIRED OUTPUT DECK READY FOR USE.

11. ERROR STOP 0333 MAY INDICATE A VIOLATION OF ONE OF THESE RULES. ALL RESTRICTIONS ABOVE ARE CERTAINLY REASONABLE AND EASY TO USE. JUST FOLLOW THE SIMPLE RULES AND EVERYTHING ELSE WILL BE DONE AUTOMATICALLY.

DON KNUTH JUNE 11, 1958

DK:SCP

Note: scp was the 'supervisory control printer' at the UNIVAC console. I probably was intrigued by its font! — Dec 95

1958.09.09

TIME STUDY ROUTINE --- September 9, 1958

This program is to be used with SOAP3 or SOAP2 output cards; the purpose is to discover how many word times the 650 must wait while executing the instructions of the program.

A card is punched for every instruction and type 2 card in a program. Pseudo-ops and comments cards are not reproduced but the card count is kept so that it will match the count on the original program. Exceptions to this rule are the pseudo ops ALF and REP since they result in machine language words; the pseudo-op NXT is also reproduced since it affects optimization to a great extent.

The SOAP language input is reproduced on the Time Study output cards, but the assembled "instruction" contains the timing information. The D-address contains the number of word times the machine waits to find the data address (a number from 0 to 49); and the I-address , the number needed to arrive at the instruction address of an instruction. On shift commands, the D- and I-addresses contain the number of words waiting time from the lower and upper limits, respectively. If an instruction is well optimized from the standpoint of the lower limit but not the upper limit, the number (upper limit minus 50) must be subtracted from the data address of the next instruction executed to see the true optimization. The optimization information punched is always based on the locations of the instructions alone, without regard to whether they are being executed from some other drum level.

An entry of 9876 in an address means the Time Study program makes no attempt to guess the waiting time or that there is no significance in the answer. Numeric OP-codes on the SOAP input are considered an indication of constants.

To run the program, read the deck in with 70 1951 XXXX±, overflow and error sense. Soap output cards follow the time study deck into the machine. Set the console to 36 0003 0006 ± when the 0363 stop occurs (see below). Use the HAND SOAP board.

Stops: 0222 Load card or availab'lity table card. Turning to Program Run will bypass all load cards.
0363 Set console to 36 0003 0006
8989 Normal machine error 8888 detected by Error Sense.
Rerun Procedure: Start at location 0250.

Donald Knuth
Case Institute of Technology

1959.01.31

C L O C K W A T C H   * * * * * * * * * * * * * * * * * *   Jan. 31, 1959

CLOCK WATCH is an added convenience for RUNCIBLE users when debugging a program, designed primarily for those who do not have a working knowledge of 650 basic machine language. The CLOCK WATCH routine punches a play-by-play report of the program as it is running, including all intermediate answers achieved.

I. When CLOCK WATCH is used, it replaces the Extra Clocking Features described in Appendix I of the RUNCIBLE I Manual (p. 33f). An additional 98 locations of memory are used but they need not be added to the amounts on the Header Card.

II. While compiling, digits m and n of the "klmn±" console code should be set to 88 for Clock Watching.

III. CLOCK WATCH, when in operation, will punch a card for almost every statement executed. (The only statements it misses are non-substitution statements which are followed by a statement numbered zero.) The output cards have the following format:

	Non-Substitution Statement	Substitution Statement
word 1:	the letters CLO	
word 2:	000000nnnn (where nnnn is the statement number)	
word 3:	0000000000	IDENT explained below
word 4:	trash	VALUE
wds. 5-8	0000000000000000000000000000000019571926	

The IDENT displayed is that of the variable at the left of the substitution arrow, which is being given the VALUE specified. The forms of the IDENT and VALUE are exactly the same as those used in ordinary input data card format* (see RUNCIBLE Manual, p. 16). A listing of these cards will tell the story of the compiled program.

IV. The Clock Watch deck is loaded at the beginning of step 16 in the operating instructions (when your deck and the basic package are in and the machine is stopped with 1999 in the ADDRESS lights). Now put the CLOCK WATCH deck in the read hopper, hit COMPUTER RESET and PROGRAM START, and END OF FILE at the proper time, and you are ready to start with step 16 again according to normal procedure.

V. During the running phase of your program, after all decks except the data cards have been run into the machine, the console setting is:

q X X X X X X A A A ±

where the X's and sign are arbitrary. (This setting is made as part of step 16, right after the CLOCK WATCH deck is in.) Digit q controls operation according to the rules

q=0,5,6,7,8, or 9: Minimum clocking only
q=1,2,3, or 4: Clock-Watching will always occur

after statement AAA is encountered for the first time. If AAA is zero, Clock-Watching will start immediately from the beginning.

VI. This routine will work with any basic package, whether type X or Y operation. It may not work with "doctored" RUNCIBLE output--see manual p. 33 note.

* If no variables are in core. The number k (last four digits of the IDENT) is the actual true machine location of the variable specified by the rest of the IDENT. Variables in core are merely called 00 0000 k.

1959.02.04

R U N C I B L E   Z E R O -- Brief explanation of its method

RUNCIBLE 0 places your program approximately into locations 0700-1384, and the compiler itself (filling the other locations) is later overlaid by variables, extensions, and the basic package. If variables and extensions do not all fit below location 0700, RUNCIBLE puts as many of them as possible into the space between 1385 and the first location of the basic package. Thus, the variables might not be in consecutive order (I,Y,C) and a statement such as "PUNCH Y1 THRU C10" may not work as it would with RUNCIBLE I. The Error Search, Clocking, and type X or Y options are all available in Runcible Zero. Optimization is sequential except for the first command in each non-zero statement. Constants are placed in the region 1900-1997 unless full clocking is used. In the latter case, they start at 1384 and work downwards. The initialization, filling of the statement dictionary, and error search procedures are all overlaid during operation.

Donald Knuth
Case Institute
February 4, 1959

1959.02.14

RUNCIBLE ZERO

Feb. 14, 1959

RUNCIBLE ZERO is a compiler which goes directly from statements to answers without taking time to punch any intermediate program cards. Thus, one less pass than you need for a so-called "one-pass" compiler is all that is required. Coding is exactly the same as for RUNCIBLE I--type B processing except for the few exceptions noted below.

Since RUNCIBLE I requires 2000 memory locations and RUNCIBLE 0 must be cut to considerably less, a few restrictions on programs have been made. Capacity has remained almost the same--any program which would put out less than 750 SOAP cards with type A processing and RUNCIBLE I can be handled by RUNCIBLE 0 except in highly unusual cases. Running time is about 25% faster with type A processing, however.

Changes to RUNCIBLE I rules:

1. The following statements will not be acceptable:

   (a) Iteration statements
   (b) Matrix definition statements or any matrix notation
   (c) "Set error correction" statements--now treated as jump statements

Note that (a) and (b) were superfluous statements which were convenient but could be programmed in terms of other statements. (c) was omitted because RUNCIBLE I Type Ais recommended for long production runs.

2. Each statement must fit on a single card. However, no final F need be punched in column 70, and columns 43-72 may all be used for the statement. (Because of this one-card restriction, parentheses nesting has been held to a maximum of four deep.) The last statement should still terminate in FF in the normal way, however. If you want to punch the F's in column 70 on any or all of your statements, you may do so without difficulty; in this way your statements are admissible to RUNCIBLE I also.

3. When running the program, rules are the same except the input deck (for step 3) is stacked in the following order:

   1) RUNCIBLE ZERO deck
   2) Header Card
   3) Comments Card
   4) Statements
   5) Relocation Package
   6) Any extensions used in relocatable form
   7) The basic package you want
   8) BLANK CARD

Omit steps 8 through 15 in the operator instructions. There are no rerun procedures.

4. New Error Stops: The "1234" Error Stops now appear as "0123". New stops are

8700 More than ten extensions and not in Error Search Mode
8701 Program too large for storage available and not in Error Search Mode
8702 More than 98 constants and not in Error Search Mode with Full Clocking
8703 Relocation Package not following "FF" statement

When using Error Search Mode the stops are the same as with RUNCIBLE I.

1959.08.15

N U M B E R   P E R V E R T E R   D E M O N S T R A T I O N   C A R D

Instructions for use:

Prepare console as follows:

Storage Entry	70 9000 9001 +
Half Cycle	RUN
Address	8000
Control	ADDRESS STOP
Display	UPPER ACCUM

Place Perversion Card in read hopper.
Depress Computer Reset, Program Start.
Depress START and END OF FILE simultaneously on card reader.
The program should now be stopped with 8000 on the Address Lights.
Change storage entry switches to 60 8004 9001 +.
Now the program is satisfactorily initialized.

Set 8004 to any number. Press Program Start. When the machine stops, the number will appear with its digits reading from right to left instead of from left to right.
You may reset 8004 and depress Program Start again as often as you wish.

```
The card:  NZA   40 9007 8000
           STL   20 9009 9002
           RAL   65 8003 9003
           DIV   14 9004 9005
                 00 0000 0010
           AUP   10 9009 9006
           AXA   50 1000 9000
           MPY   19 9004 9001
```

-Donald Knuth 8/15/59

1959.09.00

AXOLOTL AXOLOTL

As anyone could discover from reading Appendix IV of the SuperSoap Manual, it is possible to put your assembled program immediately onto disk tracks. Now if you happen to have a mistake in that program, it will be difficult to correct the mistake on the disk storage. THIS IS WHY AXOLOTL WAS INVENTED! Axolotl will take any drum load and put it onto 34 consecutige tracks of disk storage -- with its own loading routine automatically attached.

Here's how AX~~@&~~OTL works:

1. We start out when you have the drum filled with 2000 words of good information. (All of this information must be valid, since it must be loaded into the core by the Axolotl routine.) Your object is to put these 2000 words into "safe" keeping on the ram unit, in a self-loading form.

2. Set the console to 70 9000 FDA, where FDA (first disk address) is the disk and track number of the first of 34 consecutive disks AXolotl is to use. FDA may not be 0000.

3. Set the 8004 switches to the instruction which you will want to execute after the drum has been loaded by its self-loading routine.

4. Read in the Axolotl card.

5. Axolotl now takes over, puts the whole drum away, and punches a card. (Caution: when it tries to punch a card, it is not done storing the drum yet-- there is one more track to write!)

6. When Axolotl is finished it goes immediately to the instruction on 8004. The drum is unchanged from the way it was when Axolotl began.

7. The card that was punched will, at some future time, bring in the information which was stored on the drum and will continue with the instruction you put on the 8004 switches when you originally Axolotlled the program.

Comparison between AXOLOTL and MOXIE:

Axolotl does not restore the accumulators or index registers. Moxie does.
Axolotl does not restore the core. Moxie does.
Axolotl uses 34 disk tracks. Moxie uses 41.
Axolotl takes 9 seconds to store your program. Moxie takes 44 seconds.
Axolotl takes 7 seconds to load your program. Moxie takes 14 seconds.
When you reload your program it takes off immediately. With Moxie, you must set the 8004 switches and hit program start.
Moxie destroys the drum after it has dumped it; Axolotl leaves it untouched.
Moxie checks the drum for invalid information. Axolotl doesn't, but if there are enough complaints, Axolotl might.