JILL CARTER

No. 1005

report

CASE

SOAP III

Vol. I Series IV

February, 1958

D. E. KNUTH

Computing Center

CASE
Institute of Technology
Cleveland 6, Ohio

CASE SOAP III

Symbolic Optimum Assembly Program

for the IBM 650 Data-Processing System

COMPUTING CENTER
Case Institute of Technology
Cleveland, Ohio

Abstract

CASE SOAP III is a modified version of SOAP II which was originally written by Stan Poley of the Applied Science Division of IBM Service Bureau Corp. The major changes made in the preparation of CASE SOAP III were the addition of five-instruction-per-card output if desired and the use of "program points" as a greater convenience in writing programs. CASE SOAP III also offers twenty-four new pseudo operations and error correction routines which ease the task of writing and running SOAP programs. Several of these new pseudo-operations are included primarily to make CASE SOAP III of utility to the users of a 650 with extra attachments. The relocatable library routines have been omitted to provide room for the new material (however, ten additional regions are available as a replacement). CASE SOAP III is also designed for use with Compiler III (soon to be available from the Case Computing Center). The CASE SOAP III language is compatible with UNISAP, an assembly program for the Univac (also forthcoming shortly).

Acknowledgments

Credit should be given to the following persons who have helped considerably: to Melvin Conway, who is responsible for the "program point" concept; to george Haynam and Don Buyansky, for board wiring and diagrams: to Frederick Way III, Bill Lynch, George Petznick, and Angus Bond for many suggestions and criticisms; and to the Case Computing Center for thousands of cards used in the creation of the program.

Donald E. Knuth
Case Computing Center
January 31, 1958

C A S E S O A P I I I

INTRODUCTION. The user of CASE SOAP III should be familiar with the machine language instructions used in the 650. An assembly program such as SOAP greatly simplifies the task of machine language coding because it almost completely relieves the programmer of assigning actual storage locations. If he wants to refer to a certain location he gives it a symbolic name (Preferably one of high mnemonic value) and refers to it by means of this name. For example, salary might be stored in"WAGES"; sin x in "SIN X." The assembly program assigns actual locations to these names and produces the actual machine language program. The locations chosen are optimized; that is, they are put in the best possible place on the revolving drum for high-speed operation.

SOAP stands for Symbolic Optimum Assembly Programming and it is designed to assemble programs for any combination equipment used in the IBM 650 Data Processing System. Although SOAP itself operates from the basic 650 having only an alphabetic device as an added attachment, it will assemble programs written for tapes, printer, immediate access (core) storage, indexing registers, floating point arithmetic operations, disk storage and inquiry stations.

Machine language programs are coded in symbolic language, with one instructions per card, and processed by SOAP. The assembled output is then immediately reloadable as the desired machine language program.

INPUT CARD FORMAT. The Location Address (called the L-address) is punched in columns 43 through 47. It represents the location into which the assembled instructions will be loaded.

The Operation Code (OP-code fills columns 48 to 50. OP-codes for the 650 may be written either as a three-letter alphabetic mnemonic or in the

standard two-digit numeric form. A complete list of the alphabetic symbol codes to use with CASE SOAP III is given in Appendix A. When the two-digit number is given as the operation, column 48 is left blank, while columns 49 and 50 contain the number. Examples: RAL

or 65

The Data Address (or D-address) of the instruction to be assembled is specified by columns 51-55. Column 56 is the "tag" associated with the D-address, and is normally left blank unless an instruction is to be indexed or when using the PIK device (see INDEXING, PIK).

The Instruction Address, or I-address, is specified by the five columns 57 through 61. Column 62 is the I-address "tag" which is similar to the D-address tag described above.

Columns 63 through 72 may be used by the programmer for up to ten digits of comments. These columns are ignored by the SOAP program.

The sign of the instruction is specified in column 42. This column should be blank when a positive sign is desired. Any punch in this column will be interpreted as minus, and a negative instruction will be assembled as a result.

Column 41 specifies card "type" and is normally left blank unless the input card is to be treated as Type 1, 2, or 3, which are described in the sequel.

Columns 41 through 72 are reproduced on one-per-card output (see FORMATS, Appendix D). The remaining columns 1-40 and 73-80 are ignored by SOAP except that columns 2, 4, and 7 should not contain 12 (Y) punches.

Summary:

Column	Function
41	Type
42	Sign
43-47	Location Address
48-50	Operation Code
51-55	Data Address
56	Data Address Tag
57-61	Instruction Address
62	Instruction Address Tag
63-72	Comments

TYPES OF ADDRESSES. The L-, D- and I-addresses each use five columns of an input card. The left-most column of an address is called the "symbolizer part." There are five distinct types of addresses in SOAP language:

A. Absolute addresses. These are four-digit numerical addresses which are simply reproduced in the output. An absolute address is used when the programmer has a definite location in mind. For example, to reset add to the upper the contents of the upper accumulator, the instruction RAU 8003 may be used with the absolute D-address of 8003. Shift operations generally use absolute D-addresses.

Rules: 1. The symbolizer part of an absolute address must be blank. 2. The remaining four columns must be filled with the numeric address.

B. Regional addresses. A "region" of the drum is specified at the beginning of the program by a "REG" card (see below), and locations within this region are referred to with regional addresses. Each region has a letter or number associated with it, and there are thus 36 available regions. For example, region P may be the block of consecutive locations 1977-1986; the regional address P0001 refers to the first location in region P (namely 1977); R0012 is the twelfth word of region R. Regional addresses are a convenience for the programmer, saving him from making unnecessary calculations. They are also useful for relocating a routine coded in regional addresses, since the regional specification may be moved to any part of the drum.

Rules: 1. The alphabetic or numeric character which designates the region is punched in the symbolizer part of a regional address. 2. The remaining columns must be numeric.

C. Blank addresses. If the D- or I-address is to refer to the location of the next instruction, this address and the L-address of the next instruction may be left blank (provided the order of the cards is not changed when assembling). More than half of the addresses in a typical SOAP program are blank. This is a time-saver for the programmer and the keypuncher. Both D- and I-addresses may be blank if they both refer to the next location.[1]

1. Only the D-address is optimized in this case.

Rules: 1. When either the D- or I-address is blank, the L-address of the following instruction must be blank. 2. When neither D- nor I-address is blank, the L-address of the following instruction must be non-blank. 3. The first location of a program must not be blank.

D. Program points. When either the D- or I-address is to refer to location "1" which appers later, the address "1F" (1 forward) is given; to refer to location "1" which appeared earlier in the program, the address "1B" (1 backward) is given. The location 1 would be called simply "1." Ten program points are available, 0-9, and each may be constantly redefined during the course of the program. If the program is written down on a coding form in sequence, "1F" will always refer to the next location 1 on the coding form; "3B" will always refer to the previous location "3" occurring on the form. In location 1, a D- or I-address of 1F may be given immediately, referring to a future location 1. A D- or I-address of 1B would still refer to the last location 1 (not the present one). A D- or I-address of "1" will refer to its own location.

Rules: 1. A D- or I-address will be interpreted by SOAP as a program point only if the symbolizer part is a number, the second column is a "B" or an "F", and the remaining columns are blank. 2. The L-address of an instruction will be interpreted by SOAP as a program point only if the symbolizer part is a number and all remaining columns are blank.[1] 3. A D- or I- address consisting of a number followed by four blanks will always refer to its own location, regardless of what number is used or what type of address is used in the L-address.[2]

E. Symbolic addresses. Every address which is not an absolute, regional, blank, or program point address will be interpreted as a symbolic address. A symbolic address may be any combination of characters acceptable to the

---

1. A program point address of the forward or backward type appearing in the L-address field will be interpreted as a symbolic address, but it will be impossible to refer to it in the D- or I-addresses of any instructions since it will be interpreted as a program point there. (This is of use only with the DUP pseudo-op; see Appendix C.)
2. When in drum-core mode (see DRC) this address will refer to the drum equivalent location; when used with EQU, SYN, REP or PIK, this address will be defined, referring to the location of the last assembled instruction.

alphabetic device (or special character attachment if it is used). A symbolic address is given an optimum equivalent address when it is first encountered, and succeeding use of this symbol will refer to its equivalent.[1]

Examples of addresses:

8000	Absolute
8000	Symbolic
2F	Program Point (in D- or I-address only)
2 F	Symbolic
R0123	Regional
R123	Symbolic
R 123	Symbolic
	Blank
START	Symbolic
9	Program Point (in L-address); location of instruction ($^{D-}_{I-}$addr.)
A	Symbolic
9	Symbolic
0009	Absolute
40009	Regional
/0009	Symbolic (used only with special character device)
9B	Program point (in D- or I-address only)
B9	Symbolic
B0009	Regional

DATA. Constants and data are easily assembled by SOAP. Numerical data are written using the absolute part of the OP-, D-, and I-fields; that is, writing the first two digits as an absolute machine code "operation," the next four as an absolute D-address, and the remaining four as an absolute I-address (thus using columns 49-50, 52-55, and 58-61).

>ALF: Alphabetic data of up to five characters per word may be assembled using the operation code "ALF." The five characters are written in the D-address, and the I-address is ignored. (Special characters may be used if the machine has the special character attachment.)

Examples:

```
Col.: 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
        0  N  E              0  0     0  0  0  1        0  0  0  0
     -  C  0  0  0  5        3  1     4  1  5  9        2  7  5  1
        5                 A  L  F  S  0  A  P  3
           9  0  1  0  A  L  F     7  0  9
```

1. This equivalent can be redefined (see EQU)

The four cards on the previous page will cause: 1. The number +0000010000 to be loaded into symbolic location ONE 2. The number -3141592751 to be loaded into the fifth location of region C 3. The number +8276617793 to be placed in the location of program point 5 4. The number +0097909900 to be placed in (core) location 9010.

Note: ALF is called a "pseudo-operation"; that is, it does not correspond to any regular 650 operation code, yet it has meaning in SOAP language. Pseudo-operations are the programmer's means of communicating with SOAP to control the assembly of his program. There are a total of 27 pseudo-ops. (Each pseudo-op is herein designated by the symbol > as its function is defined.)

AVAILABILITY. As SOAP begins to assemble a program, all 2000 locations of the drum are considered "available," that is, any of them may be chosen as the location assigned to a symbol, program point, or blank address. As soon as a location is chosen, however, it is made unavailable so it will not be used twice. Absolute addresses are not made unavailable when encountered by SOAP.[1] The availability status of all locations can be easily controlled by the programmer to his liking with several pseudo-ops.

>BLR: (Block Reservation) Sequential locations between two limits are made unavailable, regardless of their former status. The address of the first location (called FWA, the first-word address) is specified by the D-address and the last-word-address (LWA) is specified in the I-address field.

Rules: 1. Both FWA and LWA must be given as absolute addresses and must be in the range 0000-1999. 2. The L-address and sign are ignored. 3. A single location may be reserved by letting FWA = LWA.

>BLA: (Block availability) BLA is the same as BLR except that all locations from FWA to LWA are made available regardless of their previous status.

Example: The two cards BLR 1500 1900
BLA 1600 1602 (in that order) make locations 1500-1599 and 1603-1900 unavailable and locations 1600, 1601 and 1602 available.

---

1. If an absolute address is to be unavailable it should be reserved at the beginning of assembly with a BLR card.

;DRU: (Drum Unavailability) All 2000 locations are quickly made unavailable. (L-, D-, and I-addresses and sign on a DRU card are ignored.) It is considerably faster to execute DRU followed by BLA 0000 0300 than to block reserve 0301 to 1999 with BLR.

>REG: (Regional specification) A region on the drum must be defined by a REG card before any regional addresses in that region may be given. The number or letter designating the region is punched in the symbolizer part of the D-address (column 51). FWA and LWA of the region are given in the absolute part of the D- and I-address fields, respectively, just as in BLR. (L-address and sign are ignored.) As in BLR, all locations from FWA through LWA are made unavailable, and should be drum addresses. A region may, however, be defined for absolute addresses not on the drum if desired (in the core, for example.) In this case, LWA is left blank, and no locations are reserved; FWA is then enough to specify the region.

A region may be redefined; for example, region A might first be locations 0015 through 0017 and later be changed to 1015 through 1022. A regional address is always assembled according to the latest REG card defining that region.

Regions with number designations are normally used by standard relocatable routines to be used with CASE SOAP III; the programmer should confine himself to alphabetic regions when using these routines as part of his program.

Regional addresses such as A0000 are permitted--the result is to assemble location (FWA - 1).[1] In the example above this would give first 0014, then 1014. A regional address A9999 will result in (FWA - 2). Similarly, regional addresses whose equivalents are larger than LWA are also permitted and properly assembled. Only locations FWA through LWA are made unavailable, however.

Examples

```
REG P1977  1986
REG C9049
```

The first card defines region P starting at 1977 and reserves locations 1977 through 1986. The second card defines region C starting at core location

---

1. If FWA = 0000, location 9999 is assembled.

9049, and does not reserve any locations. The erroneous card

REG C9049 9050

would define region C as before but would then stop the SOAP program (see under STOPS, 0999).

Availability may also be "doctored" using pseudo-ops COR, DRC, and PIK explained in Appendices B and C.

>PAT: (Punch Availability Table) SOAP "remembers" the availability status of all drum locations in a 200-word table. Pseudo-op PAT causes the punching of this table in highly readable format (see FORMATS, Appendix D). Available locations are listed as 1 and unavailable, 0. The availability table may be wholly or partially loaded as input to SOAP (merely by placing it together with the input program) thus restoring the availability which existed at some point of a prior assembly.

DEFINITION OF SYMBOLS AND PROGRAM POINTS. A symbol, program point, or regional address is thought of as being "defined" when it has a meaningful equivalent at that point in the program.

A symbol becomes defined after its first use in a SOAP program.[1] It then remains defined until the end of the program (or until execution of the pseudo-op SER; see below).

A forward program point such as 1F defines location "1"; the program point 1F is now defined until location 1 comes along. Then "1F" is immediately undefined until it appears again.

A backward program point such as 1B becomes defined after the first instruction in location 1 is assembled. Every succeeding location 1 will redefine the meaning of "1B."

If a location 1 occurs before the use of any 1F, the address 1B will still be defined in the future. (When undefined symbols or program points appear in the L-address of an instruction, SOAP assigns a random location.[2]

Regional addresses are defined by REG cards and remain defined until the

---

1. Within each instruction, the order of processing is location, data, instruction address.
2. In COR or DRC mode, SOAP picks a sequential location.

end of the program. Absolute addresses are always defined.

PREDEFINING SYMBOLS AND PROGRAM POINTS. A symbol or program point may be given any desired equivalent by means of pseudo-operations.

>EQU: (Equivalence) The symbol or forward program point in the D-address is assigned the equivalent of the I-address.

Rules: 1. The I-address may be absolute, regional, symbolic, or a program point, but it must be defined. The equivalent need not be in the range 0000-1999. 2. If a symbol in the D-address has been previously defined it will be redefined to the new equivalent. 3. If a program point is given in the D-address, it must be a forward program point which is currently undefined. It is thus impossible to predefine a backward program point; a forward program point, once defined, cannot be redefined until its location appears (or pseudo-op UND is used). 4. The sign (col. 42) and the location address must be blank. (Operation of EQU when these fields are non-blank is described in Appendix C.)

>SYN: (Synonym) Operation of SYN is exactly like EQU except that the I-address equivalent is also made unavailable (without regard to its former status). This equivalent must be in the range 0000-1999.

Examples:

```
EQU UPPER   8003
SYN START   1999
EQU 3F      1B
EQU R       R0001
```

In the first example, "UPPER" is assigned the equivalent 8003. Note that SYN could not be used. The second card sets the equivalent of START to 1999 and makes this location unavailable. Example # 3 defines 3F to the same equivalent as 1B; 1B must be currently defined, 3F undefined. The last example sets "R" to mean the same as R0001 (Region R must be defined). EQU was used in preference to SYN in the last two examples since the address equivalents have presumably been made unavailable already.

>UND: (Undefine) This pseudo-op frees a program point from its defined state, making it available for redefinition.

Rules: 1. The program point to be undefined is punched in the L-address

as a number followed by four blanks. 2. The D- and I-addresses and sign are ignored. 3. UND undefines both forward and backward program points.

Example 1 UND

>PST: (Punch Symbol Table) Every symbol currently defined is punched, together with its equivalent, on reloadable EQU cards. The symbol table of a long program can be conveniently inspected for errors in keypunching, which become obvious if unfamiliar symbols appear. PST is also necessary if the symbol table becomes full (see CAPACITY).

Rules: 1. The D- and I-addresses and sign are ignored; the L-address and comments are reproduced on every EQU card in the punched table. 2. When in five-per-card punch mode (see OUTPUT), PST also has the effect of a blank card. Normal punch mode is always restored after the table is punched.
>SER: (Symbole Erase) The entire symbol table is erased (similar to DRU).

CAPACITY. CASE SOAP III can remember 300 symbols and their current equivalents. With the use of program points, more than 300 symbols will rarely, if ever, occur. The SOAP program itself uses only 147 symbols (without program points it would have used 308); use of program points generally halves the number of symbols required.

If the symbol table does become full, it is possible to continue by punching the symbol table with PST, then giving the command SER and reloading only the symbols which are necessary for the remainder of the program.

SPEED. Assembly progresses at approximately 50 to 90 cards per minute; this will decrease if the symbol table becomes densely packed or if few drum locations are available. The processing of a program point is always much faster than the processing of a symbol.

OUTPUT. The CASE SOAP III programmer has his choice of two outputs: (for formats see Appendix D).

A. Normal output. The input card is reproduced in columns 41-72 and the assembled instruction is immediately reloadable as a single instruction load card. One-per-card output may also be used again as SOAP input.

B. Five-per-card output. All assembled instructions are punched on five-

per-card load cards, which may be loaded with the loading routine in Appendix E. When in five-per-card punch mode, assembly is in general much faster, more conservative of cards, and the final program will load into the machine much more rapidly. The disadvantages are in the lack of reuse as SOAP input and the increased difficulty in debugging a program.

>FIV: The pseudo-op FIV will cause CASE SOAP III to punch succeeding assembled instructions in five-per-card format. The FIV card is not reproduced in the output.

>ONE: This pseudo-op restores normal punch mode, and has the effect of a blank card except for the printing of the transfer instruction. The ONE card is not reproduced in the output.[1] Note: PAT and PST also restore normal punch mode after their execution.

BLANK CARD: The input deck to be processed five-per-card should be followed by a blank card to insure that the last card will be punched (in case the number of instructions is not a multiple of five). The blank card also gives an optional transfer instruction: If the console switch is set to minus, the last instruction assembled will be 1998: 00 1999 1999 which will automatically transfer control from the loading routine in Appendix E to location 1999 as soon as the program is completely loaded. To use this feature, locations 1951-1960 and 1977-1998 should be block-reserved at the beginning of assembly (for the loading routine) and the program should start at 1999.

The pseudo-operations BOP, PAT, PST and ONE have the same function as a blank card (in addition to their normal functions) except ONE never prints the optional transfer instruction.

TYPE 1 CARD. If the programmer wishes to have more than the ten columns of an ordinary card for comments, a type 1 card may be used. SOAP will do nothing with this card except reproduce it (if in normal punching mode). The type 1 card may contain up to 30 characters punched in columns 43-72.

---

1. Unless it is redundant.

MISCELLANEOUS PSEUDO-OPS.

>OFF and ONN: After an OFF card appears, SOAP will not punch any pseudo-op cards or type 1 cards until the pseudo-op ONN restores normal procedure. Assembled instruction are punched as usual. Note that OFF is redundant when in five-per-card punch mode. Most extensive use of OFF and ONN is made when reloading the symbol table EQU cards.

>CCT: (Card Count) The card number on the one-per-card output card which follows the CCT card is set equal to the number specified by the D-address. (The D-address must be an absolute four-digit number.) For example, CCT 0000 causes SOAP to give the next card number 0000 and then to start again from 0001. (This can be used to advantage just before PST, for it will give a count of the symbols used.)

>BOP: (Beginning of Program) SOAP will initialize itself to begin assembling another program. This enables the operator of the machine to stack several programs to be SOAPed on top of each other in the card hopper, and the entire operation will proceed in one pass. BOP also has the effect of a blank card (see BLANK CARD).[1]

RUNNING THE PROGRAM. The assembly deck should have the following order:

1. CASE SOAP III omitted if SOAP is already on the drum)
2. Deck to be assembled (when five-per-card is desired, a blank card is usually placed at the end of the deck)

Due to the one pass nature of the assembly, priority for the choice of optimum locations diminishes as assembly progresses. Therefore frequently executed portions of the program should be placed toward the beginning of the assembly deck.

Machine Operator's Guide:

533 Read-Punch Unit: Insert CASE SOAP III Control Panel, place assembly deck in read feed, ready punch hopper with blanks.

---

1. A card with a zero in column 41 will have the effect of BOP except in its function as a blank; this card will not be reproduced in the output.

650 Console: programmed STOP, half cycle RUN, control RUN, display DISTRIBUTOR, overflow SENSE, error STOP. If CASE SOAP III is being loaded, set 70 1951 8888 on storage entry switches. If SOAP is already on the drum, set 00 1951 0000 on storage entry switches. (See BLANK CARD for instructions on the sign of the switches.)
1. Press COMPUTER RESET 2. Press PROGRAM START 3. Press both START keys on the 533 card reader 4. When read hopper empties, press END OF FILE.

The availability and symbol tables may be obtained manually by sending control to locations 0900 and 0800 respectively. The FIV pseudo-op may be duplicated manually by transferring to location 0123; ONE is accomplished by transferring to 0133.

A dry run of a new program may be made by inserting 00 1951 8002± into location 1997, which suppresses all punching except that of the availability table. SOAP will then merely check the input cards for errors, which can be corrected before the actual SOAPing of the program takes place.

STOPS:

0111. Symbol table full

0222. Drum packed; all locations unavailable (or PIK table empty; see under PIK, Appendix C)

0333. Invalid symbolic operation (or use of 999 or YYY pseudo-ops without special deck--see 999, Appendix C)

0444. Undefined regional address or REG card without proper character in column 51

0555. EQU, SYN, REP, or PIK with undefined address which should have been defined (see rules for these pseudo-ops)

0666. Blank location address when previous instruction had no blank D- or I-address.

0777. Filled location address when previous instruction had blank D- or I-address; or, meaningless location (as 8492)

0888. Occurence of undefined backward program point; or EQU, SYN trying to redefine a defined forward program point

0999. BLR, BLA, REG with FWA > LWA or BLR, BLA, REG, SYN attempting to reserve location greater than 1999.

1996. See OPT, Appendix C.

ERROR CORRECTION: On each of the programmed stops on the previous page, the error may be corrected by removing the cards from the read hopper, clearing the read feed, correcting the offending card (the third one out--or the fourth last if not on End Of File) and replacing it in the read feed. Continue with step 2 on the previous page.

>REP: (Repeat assembly) If the previous card contained an error, rather than the one on which SOAP is stopped, it may be corrected immediately with the pseudo-op REP instead of reSOAPing the entire deck.

Rules: 1. The equivalent of any absolute, regional, symbolic, or program point address in the D- or I-address of a REP card will be substituted for the corresponding D- or I-address of the last assembled instruction. (This equivalent must be defined.) 2. If the D- or I-address of the REP card is blank the corresponding address of the last assembled instruction will be untouched. 3. The reassembled instruction will be loaded into the same location as it was previously; the L-address of the REP card is ignored. 4. The sign of the reassembled instruction will be the sign of the REP card. 5. Any number of type 1 cards and pseudo-ops (excluding of course BOP) may intervene between the instruction to be reassembled and the REP card. 6. The instruction following REP must have a non-blank L-address.

Example: The coding

```
      LDD 1F
SUBR1 STD EXIT
```

will stop on the second card with a 0777 stop. Suppose the programmer wished the first instruction to read "LDD 1F SUBR1." This can now be corrected by inserting the card

```
REP   SUBR1
```

before the second card, and continuing with step 2 in the operator's instructions (depressing PROGRAM START, etc.). If SUBR1 has not been defined up to this point, checking of the output will show which address was assigned to the incorrect blank address, say 1234. Then the card EQU SUBR1 1234 followed by the REP card will overcome the difficulty.

Restarting after machine errors: If the 650 stops because of a bad read or a machine validity checking error, or for any other reason, SOAP III may be

restarted by replacing the deck (beginning with the card being processed) in the read hopper, setting 00 1951 0888 in the console switches, and continuing with step 1 in the operator's instructions. Any "harm" which might have been done during incomplete assembly will be undone, but for exceptional cases. (A symbol or forward program point which has just been defined will not always be dropped from memory, however.)

EXAMPLE PROGRAM. In conclusion, we give a simple program which will prepare a table of. $F(x) = Ax^2 + Bx + C$ for $x = 1, 2, \ldots, 100$ assuming A, B, and C are integers and $|F(x)| < 10^{10}$. The output is to contain x in word 1, F(x) in word 2. Input as written on a SOAP coding form is shown below; the one-per-card assembled output is given on the next page.

TAG	SIGN	LOC	OP	DATA	TAG	INST	TAG	COMMENTS	COL.
41	42	43 44 45 46 47	48 49 50	51 52 53 54 55	56	57 58 59 60 61	62	63 64 65 66 67 68 69 70 71 72	
1		EXAMPLE		CALCULATE				F OF X	
1									
			BLR	1951		1960		READ AREA	
			REG	P0027		0028		PUNCH AREA	
		SETX	RAU	ONE		IF		SET X	
		1	STU	P0001				TO 1	
			MPY	A				CALCULATE	
			ALO	B				F	
			RAU	8002					
			MPY	P0001					
			ALO	C					
			STL	P0002					
			PCH	P0001				PUNCH	
			RAU	P0001					
			SUP	XMAX				IS X MAX	
			NZU			9876			
			AUP	101		1B		INCREASE X	
1									
		ONE		00	0000			0001	CONSTANTS
		XMAX		00	4000			0100	
		101		00	0000			-0-1 01	

EQU START 1999

$F(x) = Ax^2 + Bx + C$

```
1          EXAMPLE   CALCULATE  F OF X          1
1                                                2
                     BLR  1951   1960   READ AREA 3
                     REG  P0027  0028   PUNCH AREA 4
(START)  SETX        RAU  ONE    1F     SET X     5      0004  60 0007 0011
         1           STU  P0001         TO 1      6      0011  21 0027 0030
                     MPY  A             CALCULATE 7      0030  19 0033 0003
                     ALO  B             F         8      0003  15 0006 0061
                     RAU  8002                    9      0061  60 8002 0019
                     MPY  P0001                   10     0019  19 0027 0047
                     ALO  C                       11     0047  15 0000 0005
                     STL  P0002                   12     0005  20 0028 0031
                     PCH  P0001         PUNCH     13     0031  71 0027 0037
                     RAU  P0001                   14     0037  60 0027 0081
                     SUP  XMAX          IS X MAX  15     0081  11 0034 0039
                     NZU  9876                    16     0039  44 0043 9876
                     AUP  101    1B     INCREASE X 17     0043  10 0046 0011
1                                                18
          ONE        00  0000  0001  CONSTANTS    19     0007  00 0000 0001
          XMAX       00  0000  0100               20     0034  00 0000 0100
          101        00  0000  0101               21     0046  00 0000 0101
```

0004 Reset upper & add "1" to it
0011 Store "1" in loc. 0027
0030 Mpy 1 × A & put answer in lower
0003 add B to I×A
0061 add Ax + B to lower.
0019 Mpy times (Ax+B)×1
0047 add 1(Ax+B) + C
0005 Store 1(Ax+B)+C in 0028
0031 pch "1" & Ax+Bx+C
0037 add 1" to upper
0081 Sub 1 - 100 = -99
0039 go to 0043
0043 add to -99 in upper; 101 = 2

A - 0033
B - 0006
C - 0000

Lower = 8002

APPENDIX A

SYMBOLIC OPERATION CODES.

650-SOAP3-English		650-SOAP3-English	
00	NOP* No operation	40	NZA Non-zero Index Register A
01	HLT Halt	41	BMA Branch Minus IR A
02	UFA Unnormalized Floating Add	42	NZB Non-zero Index Register B
03	RTC Read Tape Check	43	BMB Branch Minus IR B
04	RTN Read Tape Numeric	44	NZU Non-zero Upper
05	RTA Read Tape Alphabetic	45	NZE Non-zero Accumulator
06	WTN Write Tape Numeric	46	BMI Branch Minus
07	WTA Write Tape Alphabetic	47	BOV* Branch on Overflow
08	LIB Load IAS Block (Core)	48	NZC Non-zero Index Register C
09	LDI Load IAS	49	BMC Branch Minus IR C
10	AUP Add Upper	50	AXA Add to IR A
11	SUP Subtract Upper	51	SXA Subtract from IR A
12	Not Used	52	AXB Add to IR B
13	Not Used	53	SXB Subtract from IR B
14	DIV Divide	54	NEF No End of File
15	ALO* Add Lower	55	RWD Rewind
16	SLO* Subtract Lower	56	WTM Write Tape Mark
17	AML Add Magnitude to Lower	57	BST Backspace Tape
18	SML Subtract Magnitude Lower	58	AXC Add to IR C
19	MPY Multiply	59	SXC Subtract from IR C
20	STL Store Lower	60	RAU Reset Add Upper
21	STU Store Upper	61	RSU Reset Subtract Upper
22	SDA Store Data Address	62	Not Used
23	SIA Store Instruction Address	63	Not Used
24	STD Store Distributor	64	DVR Divide and Reset
25	NTS No Tape Signal	65	RAL Reset Add Lower
26	BIN Branch Inquiry	66	RSL Reset Subtract Lower
27	SET Set Buffer Ring	67	RAM Reset Add Magnitude
28	SIB Store IAS Block	68	RSM Reset Subtract Magnitude
29	STI Store IAS	69	LDD,LOD* Load Distributor
30	SRT Shift Right	70	RD1,RCD Read Card (Input 1)
31	SRD Shift and Round	71	WR1,PCH Punch (Output 1)
32	FAD Floating Add	72	RC1 Read Conditional 1
33	FSB Floating Subtract	73	RD2 Read Input Area 2
34	FDV Floating Divide	74	WR2 Write Output Area 2
35	SLT Shift Left	75	RC2 Read Conditional 2
36	SCT Shift and Count	76	RD3 Read Input Area 3
37	FAM Floating Add Magnitude	77	WR3 Write Output Area 3
38	FSM Floating Subtract Mag.	78	RC3 Read Conditional 3
39	FMP Floating Multiply	79	RPY Reply

80	RAA	Reset Add to IR A
81	RSA	Reset Subtract from IR A
82	RAB	Reset Add to IR B
83	RSB	Reset Subtract from IR B
84	TLU	Table LookUp
85	SDS	Seek Disk Storage
86	RDS	Read Disk Storage
87	WDS	Write Disk Storage
88	RAC	Reset Add to IR C
89	RSC	Reset Subtract from IR C

650-SOAP3-English

90	BDO+	Branch Distributor 10
91	BD1	Branch Distributor 1
92	BD2	Branch Distributor 2
93	BD3	Branch Distributor 3
94	BD4	Branch Distributor 4
95	BD5	Branch Distributor 5
96	BD6	Branch Distributor 6
97	BD7	Branch Distributor 7
98	BD8	Branch Distributor 8
99	BD9	Branch Distributor 9

* O's must be alphabetic O's, not numeric zeroes.
+ Either O or zero is acceptable.

SUMMARY OF PSEUDO OPERATIONS.

Sign-	Loc -	OP -	Data -	Inst	
i	i	ALF	XXXXX	i	
i	i	BLA	FWA	LWA	
i	i	BOP	i	i	
i	i	CCT	nnnn	i	
i	i	COR	FWA	i	
i	nnnn	DRC	FWA	i	
i	i	DRU	i	i	
i	i	DUP	SYMBL	i	
Sign	α	EQU	SYMBL	equiv	
i	i	FIV	i	i	*
i	i	HED	ε	i	
i	i	MIX	i	i	
i	i	NXT	nnnn	nnnn	
i	i	OFF	i	i	
i	i	ONE	i	i	*
i	i	ONN	i	i	
i	i	OPT	nnnn	i	
i	i	PAT	i	i	
i	equiv	PIK	equiv	equiv	
i	α	PST	i	i	
i	i	PUD	SYMBL	i	
i	i	REG	β FWA	LWA	
Sign	i	REP	DA	IA	
i	i	SCR	i	i	
i	i	SER	i	i	
Sign	α	SYN	SYMBL	equiv	
i	i	TRY	i	i	
i	n	UND	i	i	
i	i	UNP	i	i	
(various meanings)		YYY 999	(various meanings)		

i-ignored nnnn-four digit number -determines type of EQU or SYN desired
*-card does not appear in output -heading charactor -alphabetic or numeric

80 RAA Reset Add to IR A
81 RSA Reset Subtract from IR A
82 RAB Reset Add to IR B
83 RSB Reset Subtract from IR B
84 TLU Table LookUp
85 SDS Seek Disk Storage
86 RDS Read Disk Storage
87 WDS Write Disk Storage
88 RAC Reset Add to IR C
89 RSC Reset Subtract from IR C

90 BDO+ Branch Distributor 10
91 BD1 Branch Distributor 1
92 BD2 Branch Distributor 2
93 BD3 Branch Distributor 3
94 BD4 Branch Distributor 4
95 BD5 Branch Distributor 5
96 BD6 Branch Distributor 6
97 BD7 Branch Distributor 7
98 BD8 Branch Distributor 8
99 BD9 Branch Distributor 9

* O's must be alphabetic O's, not numeric zeroes.
+ Either O or zero is acceptable.

SUMMARY OF PSEUDO OPERATIONS.

Sign	Loc	OP	Data	Inst	
i	i	ALF	XXXXX	i	
i	i	BLA	FWA	LWA	
i	i	BOP	i	i	
i	i	CCT	nnnn	i	
i	i	COR	FWA	i	
i	nnnn	DRC	FWA	i	
i	i	DRU	i	i	
i	i	DUP	SYMBL	i	
Sign	α	EQU	SYMBL	equiv	
i	i	FIV	i	i	*
i	i	HED	ε	i	
i	i	MIX	i	i	
i	i	NXT	nnnn	nnnn	
i	i	OFF	i	i	
i	i	ONE	i	i	*
i	i	ONN	i	i	
i	i	OPT	nnnn	i	
i	i	PAT	i	i	
i	equiv	PIK	equiv	equiv	
i	α	PST	i	i	
i	i	PUD	SYMBL	i	
i	i	REG	β FWA	LWA	
Sign	i	REP	DA	IA	
i	i	SCR	i	i	
i	i	SER	i	i	
Sign	α	SYN	SYMBL	equiv	
i	i	TRY	i	i	
i	n	UND	i	i	
i	i	UNP	i	i	
(various meanings)		YYY	(various meanings)		
		999			

i-ignored nnnn-four digit number -determines type of EQU or SYN desired
*-card does not appear in output -heading charactor -alphabetic or numeric

APPENDIX B

ADDITIONAL FEATURES OF CASE SOAP III FOR USE WITH EXTRA 650 ATTACHMENTS.

INDEXING. When the D- or I- address is to be modified by an indexing register, the corresponding TAG column (56 or 62) is used to indicate the appropriate indexing register. Permissible tags are A, B, and C or 1, 2, and 3.[1] SOAP will automatically add the proper multiple of 200 or 2000 to all tagged addresses unless they are not in the range 0000-1999 or 9000-9059.

Regions in the core may be defined using a REG card with the I-address blank.

>COR: (Core Mode) This pseudo-op causes SOAP to stop optimum programming and begin programming sequentially. While in COR mode all drum locations are considered unavailable.[2] Every undefined address is given a sequential equivalent (starting with FWA), while previously defined addresses are assembled as usual. FWA is punched as an absolute address in the D-field.[3] COR mode is terminated by a COR card with D-address blank.

>DRC: (Drum-Core Mode) DRC is used to program sequentially on the drum, when the program will actually be block-transferred with LDI and executed in the core. Rules: 1. FWA must be a core address; it is punched absolute in the D-field. 2. The absolute drum address which corresponds to FWA is punched in the L-field. 3. The instructions are assembled exactly as they are when in COR mode, except that all L-addresses in the range 9000-9059 are translated down to the corresponding drum location in the output. 4. DRC is stopped by a DRC card with blank D-address or any COR card. When DRC is stopped, the translation of location addresses is stopped and all equivalents which were assigned are the core-equivalents, not the drum equivalents.[4]

---

1. The letters J, K, L, S, and T should not appear in a tag column in any case. Other letters and numbers, except the permissible tags above and P, are ignored.
2. This may be superseded by a P-tag (see PIK)
3. FWA need not be a core location, but if it is a drum location the sequential equivalents assigned are not reserved. The machine will stop if a meaningless location is encountered.
4. All drum locations used by DRC should be block-reserved at the beginning of assembly. Symbols may be redefined to the corresponding drum addresses after DRC with the special EQU cards described in Appendix C.

A P P E N D I X C

ADDITIONAI FEATURES OF CASE SOAP III FOR USE IN ADVANCED PROGRAMMING.

800X INSTRUCTIONS AND TYPE 2 CARDS. Instructions which take place in locations 8000-8003, 8005-8007 may be optimized by using the address as an absolute location address. Instructions which take place in eraseable locations but do not need to be loaded into these locations may be optimized by placing them on Type 2 cards. Type 2 cards are treated just as ordinary cards except that they (along with 800X instructions) are not loaded into the location specified when the output deck is loaded. (Word 1 on normal output cards is 69 1954 1953; on 800X and Type 2 cards it is 69 1954 8000. In five-per-card output the instructions do not appear at all.)[1]

HEADING.

>HED: Heading is used to avoid duplicity of symbols when several programs or several sections of a single program are to be assembled together. The need for heading is paramount if several persons have contributed to a program. Heading is accomplished in SOAP by the automatic insertion of a heading character into the right-most position of symbolic addresses which have this position blank. (When the right-most column is non-blank, the symbol is not headed. Program points are not headed.) The heading character is punched in the symbolizer part of the D-address of a HED card (it may be any character acceptable to the alphabetic device or special character attachment if used) and all other columns are ignored. This heading is applied to every symbol thereafter until another HED[2] card. appears or the program ends. To "turn off" the heading, a HED card with no punch in column 51 may be used. If within the section head by "A" it is desired to refer to the symbol "TAXbb" (b indicates a blank column) which appears in a section headed by "F", the symbol "TAXbF" should be used.

---

1. Caution: No 800X or Type 2 card should be the last instruction before a blank card, ONE, PAT, PST, or BOP operation if five-per-card punching is being used. If REP is to be a correction to either a TYPE 2 or an 800X instruction, it should be punched Type 2.
2. Or SCR

>PIK: PIK provides better optimization by keeping an alternate availability table which may be entered when desired. Suppose, for example, the programmer wishes to place a certain number into temporary storage, and he has several locations to choose between for this storage. He does not know in advance which will be placed most optimally of these. By entering these locations into the PIK table, he can then get the one which is best suited. The PIK table is used when an undefined symbol or forward program point in D- or I-address is tagged with a "P." Locations are placed into the PIK table with the PIK pseudo-op, which takes the equivalent of any absolute, regional, program point, or symbolic address in any of the L-, D-, or I-addresses, and puts it into the PIK table. Any of these three fields may be left blank; up to three locations can thus be put into the table per PIK card.[1]

In our example, suppose the programmer wants to choose between the temporary storage locations HOLD1, HOLD2, and B0004. He first gives the pseudo-op

HOLD1 PIK HOLD2 B0004

and then follows with his instruction

STL 7F P

where 7F was currently undefined. Later, when he wants to bring his number back into the lower accumulator, he instructs

RAL 7F

which may then be followed by 7 UND , freeing program point 7 for future use. The equivalent assigned to 7 was removed from the PIK table as soon as it was picked; two locations still remain in the table.

>UNP: (Un-PIK) The entire PIK table is cleared (similar to DRU).

An undefined address tagged with a P will take precedence over COR mode (see Appendix B) and may thus be used to obtain a drum address during COR processing. The PIK table will accommodate a maximum of five locations in each of the 50 drum levels, or a total of 250 locations. If a tagged address is given and the PIK table is empty, SOAP will stop (0222).

---

1. All program points must be of the forward or backward variety on a PIK card, even when placed in the L-address. All addresses on a PIK card must be defined. Caution: A given location must not be entered into the PIK table twice before clearing the table, for it may then be used twice. The PIK table differs from the ordinary availability table in this respect. Processing of a PIK card goes from left to right; so, for example, if a 0555 stop occurs because of an undefined symbol in the D-address, any address written in the L-address has already been processed and should not be rewritten on the correction card (the I-address should be left unaltered in this case).

>NXT: Another aid to optimization is the pseudo-op NXT, which specifies in what way the next instruction is to be optimized. Occasionally an unusual situation arises in which optimization should take place contrary to the usual rules. The new optimizing rule for the next instruction only (specified on a NXT card) is given in the form:

L even, D = L + αα	D even, I = D + γγ
L odd, D = L + ββ	D odd, I = D + δδ

where αα, ββ, γγ, and δδ are two-digit numbers. The four-digit numbers ααββ and γγδδ are punched in the absolute parts of the D- and I-address fields, respectively. Examples: To optimize the instruction

BOV

where both D- and I-addresses are blank, (only the D-address will be optimized by normal rules, and if the instruction takes the I-address branch, requiring two extra word-times, an entire drum revolution will be wasted) give the pseudo-instruction NXT 0505 0000 (the I-address could have been anything in this case). The sample program given at the end of the main portion of this manual may perhaps run a little faster if the pseudo-command NXT 0000 4747 were given before the PCH instruction.

>EQU extended: 1. When the sign of an EQU card is minus, the complement (modulo 10000) of the equivalent in the I-address is assigned to the symbol (or program point) in the D-address.

2. When the L-address of an EQU card is non-blank, the sum of the equivalents of the D- and I-addresses is assigned to the symbol or program point in the L-address.[1] A program point in the L-address should be a forward program point (in spite of its being in the L-address).

3. When the L-address of an EQU card is non-blank and the sign is minus, the difference of the two equivalents, D-address minus I-address, is assigned to the symbol or program point in the L-address.

---

1. Processing of EQU and SYN proceeds from right to left (I-address first, etc.) rather than from left to right as in ordinary operations.

All equivalents are positive modulo 10000 (carries are dropped).

Examples:

```
-       EQU NEG    8002
  A     EQU B      C
- 3F    EQU 4F      0001
-       EQU SYMBL SYMBL
  X     EQU X      X
```

In the first example, NEG is given the address equivalent 1998. The second sets the equivalent of A to the equivalent of B plus the equivalent of C. In the third example, the equivalent of 4F minus 1 is assigned to program point 3F. (4F must be currently defined, 3F undefined as in ordinary EQU.) Example number four sets SYMBL's equivalent equal to the complement of its former value. The last example sets X's equivalent to twice its previous value. It must be remembered that this arithmetic is done on the addresses, not on the values of the numbers in those addresses.

SYN is exactly like EQU extended except that the final equivalent assigned is reserved and must be in the range 0000-1999.

LIBRARY ROUTINES. Several pseudo-ops have been included for writers of library routines which are to be SOAPed with the programs of others.

>SCR: (Scramble) This makes very sure that no symbols of a library program will match symbols of another program. The constant 4040050598 is subtracted from the alphabetic representation of all symbols which have the right-most character blank, thus making the resulting "symbol" different from any that could possibly be reproduced in alphabetic representation. (SCR is similar to HED, only with a more complex "heading character." It is stopped by a HED card with column 51 blank.) All symbols under the influence of the SCR card must have a number or letter in the symbolizer part if they are to be scrambled (to avoid negative symbols).

>DUP: Groups of library subroutines often have common instructions which should not be repeated if two similar subroutines are both used simultaneously. DUP allows these common portions to be coded in all symbolic decks, but used only the first time they occur in the SOAPing.

Rules: If the symbol specified by the D-address is not in the symbol table, procedure is normal; if this symbol is already defined, all succeeding cards are completely ignored by SOAP except types 1 and 3. The type 3 stops the effect of DUP (see below).

>OPT: This is for portions of a library routine which are optional and can be omitted; OPT avoids the making of several decks, some with and some without these portions. An absolute four-digit number, nnnn, is punched in the D-address; the machine will halt, displaying this number (as 000000nnnn) and 1996 in the address lights. The operator at the console now chooses whether he wishes to include the next portion of the program or not, by adjusting the sign of the console switches and pushing program start. If the sign is minus or the switches are all zero, all succeeding cards are ignored until a type 3 card comes along as in DUP. If the switches are positive and non-zero, normal operation occurs, and the optional portion of the program is included. The number nnnn is used to distinguish between several OPT stops which might occur in the same program.

Type 3 Card: A type 3 card is exactly like a type 1 except that it stops the rejection mode which might exist because of DUP or OPT.

EXTRA PSEUDO-OPS

>999 and YYY: Additional pseudo-ops which might be found useful for local requirements, may be easily added to the user's CASE SOAP III deck merely by the addition of several cards just before the last card of the SOAP deck. Space in the symbolic operation dictionary has been left to accommodate pseudo-ops having the call-letters 999 and YYY. There are approximately 15 storage locations available on the drum, 50 of them in sequence, and the use of the more than thirty subroutines within SOAP will facilitate coding of these routines. Examples of such additions would be a "Block-PIK" or a block reservation or availability between symbolic address limits, etc.

A P P E N D I X D

OUTPUT CARD FORMATS.

Normal. col. 1: 12 punch
cols. 1-10:69 1954 1953+
cols.17-20:card count
cols.21-30:24 L 8000+
cols.31-40: instruction
col. 40: sign of instruction
cols.41-72:input reproduced

Type 2 and 800X. (same as normal except cols. 1-10: 69 1954 80000+)

Pseudo-ops(except ALF and REP) and types 1 and 3.

(same as normal except cols. 1-10: 00 0000 8000+
cols.21-40: blank
col. 73: 9 punch)

PST output: (same as pseudo-ops except an additional 9 punch in col. 74)

PAT output: col. 41: 12 punch
cols. 1-10: 00 α α+450 (α is the drum level 00-49 in each band)
cols.11-20: availability for α, α+50, α+100, ..., α+450
cols.21-30: 00 α+500 α+950
cols.31-40: availability for α+500, ..., α+950
⋮
cols.71-80: availability for α+1500, ..., α+1950

Five-per-card output:

cols. 1-6: 888888
cols. 7-10: card count
cols.11-20: I-1
cols.21-30: I-2
cols.31-40: I-3
cols.41-50: I-4
cols.51-60: I-5
cols.61-64: L-1
cols.65-68: L-2
cols.69-72: L-3
cols.73-76: L-4
cols.77-80: L-5
cols.7,10,70,80: 12 punches.

I-1 through 5 are the assembled instructions, with signs over the units digits

L-1 though 5 are the locations corresponding to the instructions

A P P E N D I X E

Five-per-card loading routine. Uses locations 1951-1960 and 1977-1998.

All signs are positive. Console setting to load is 70 1951 8888 +

Card 1	Card 2	Card 3	Card 4
69 1951 1955	10 1955 1998	65 1957 1987	22 1980 1986
69 1952 1956	69 1982 1985	30 0002 1996	65 1989 1993
69 1953 1957	69 1951 1980	69 1980 1988	15 1978 1984
69 1954 1958	24 1989 1991	24 8888 1990	00 0001 0000
24 1997 1952	70 1997 8888	24 1991 1998	24 1988 1998
24 1998 1953	24 1979 1999	24 1987 1999	24 1986 1999
24 1999 1958	24 1982 8002	24 1996 8002	24 1993 8002
65 1954 8000	24 1985 8003	24 1980 8003	24 1978 8003

Card 5	Card 6	Card 7	Card 8
20 1989 1992	10 1957 1981	35 0004 1996	(first
30 0004 1977	16 8002 8001	70 1979 8888*	card of
15 1983 1989	30 0008 1996	~~XXXXXXXXXXX~~	five-per
15 1889 0042	30 0004 1996	~~XXXXXXXXXXX~~	-card
24 1984 1998	24 1990 1998	24 1997 1998	load
24 1992 1999	24 1981 1999	24 1998 8001	cards)
24 1977 8002	24 1994 8002	~~XXXXXXXXXXX~~	
24 1983 8003	24 1995 8003	~~XXXXXXXXXXX~~	

* This I-address may be changed to transfer to another location on non-load if desired.

All addresses in the 1800's and 1900's may be translated into another band by subtracting the proper multiple of 50; the addresses in card 1 and in word one of card 2 should be left untranslated (along with the I-addresses of words five and six of <u>all</u> cards) if the console setting is to remain the same.

All above cards must be load cards.

INDEX