The X930 Program Set for Sigma 5 Assembly

C. C. Klimasauskas
Communications Systems Research Section

This article describes a set of programs that have been written to enable the Sigma 5 computer to assemble programs for the XDS 920/930 computers. It consists of two parts: a system procedure deck which allows SIGMA METASYMBOL to assemble a source language similar to the XDS 900-series METASYMBOL, and a secondary loader which reformats the Sigma 5 load module into the Universal Binary Language of the 900-series machines and writes it to cards or magnetic tape. The syntactic differences between this assembler and 900-series METASYMBOL are described, as well as the process of generating a 920 program using this program set and the Sigma 5.

I. Introduction

X930 is a program set consisting of a system procedure deck and a secondary loader. The procedure deck defines the algorithms required to convert an assembly deck similar in format to an XDS 920/930 METASYMBOL assembly deck to an intermediary format. This format, called ROM language (Relocatable Object Module), is converted by the Sigma 5 loader to a format (LMN—Load Module) which is an appropriate input form for the secondary loader. During this load operation, all references to external routines are satisfied from the Program Library. Then the secondary loader processes the load module, producing a binary format which can be loaded by the XDS 930 Monarch Loader, or by an XDS 930 Bootstrap Loader. This binary output can be to punch cards or magnetic tape. A flow chart of this is shown in Fig. 1.

II. The System Procedure Deck

The system procedure deck accepts a source deck which is similar in format to an XDS 930 METASYMBOL deck. The major differences between X930 source language and XDS 930 METASYMBOL source language lie in the format of literals (Holerith, Octal, and Decimal), the method for making external references and definitions and the use of the DO, PROC, and END directives. Three special directives in X930, described later, aid in alleviating the problem of generating literals for the XDS 930 computer.

The user gains the advantage of being able to employ the Sigma 5 METASYMBOL directives in his program.

III. The Secondary Loader

The secondary loader takes the core image code produced by the Sigma 5 loader and translates it to XDS 930 standard binary language. The secondary loader aborts whenever an input/output (I/O) error occurs, or the program detects an illegal input format, or an error has occurred during the load operation which generated the load module. An appropriate message is printed at the time of the abort.

The loader is not capable of detecting the absence of an end transfer record for a program, and will assume an end transfer address of zero. Great care must be exercised to always include an end transfer address on the program END card. The secondary loader is not capable of loading a load module longer than 2000_{16} words. Load modules with more than 2000_{16} words are rejected. All external references must be satisfied and no overlay structure is allowed. The binary output from the secondary loader will be absolute or relocatable contingent on the mode in which the load module was generated by the Sigma 5 loader.

IV. User Description

This is a user-oriented description of the X930 program set. It is assumed that the user is familiar with the XDS 930 computer and the Sigma 5 BPM operating system. Examples are given where it was felt greater clarity might be achieved and no attempt was made to be exhaustive. The description consists of two parts: (1) the PROC deck, and (2) the secondary loader.

A very important part of the X930 program set is the PROC deck. The PROC deck describes the 930 instructions, and specifies exactly what binary data are generated for each. The following sections describe the instruction format, available directives, and a comparison of the X930 (Sigma 5) assembler directives and 930 METASYMBOL directives.

A. X930 Directives

1. BCD directive. Sigma 5 BCD to 930 BCD conversion PROC. Similar to the BCD directive in 930 METASYMBOL:

BCD,C 'xxxx', 'xyyy', 'zzzz'

BCD Invokes the BCD PROC which converts text strings in the address field to 930 BCD format.

C Number of characters in each string. C must be between 1 and 4. If C is omitted, 4 characters/text string is assumed.

'xxxx' 'yyyy' 'zzzz' These are all Sigma 5 text strings (character literals). They may be at most 4 characters long. As many strings as desired may appear in the address field of the BCD directive.

The BCD directive takes the Sigma 5 literal string and converts it to 930 character code, packing it 4 characters/word. If fewer than four characters exist in each string, or C is less than 4, the characters will be left justified with trailing blanks inserted. Hence, each string generates one word of code.

2. FSC directive. Floating-point short to 930 decimal conversion PROC:

FSC FS'347.2',FS'−964.3E+8'

FSC Invokes the floating-point short constant PROC, which converts Sigma 5 floating-point short constants to 930 floating-point constants:

FS'347.2'

FS'−964.3E+8'—Sigma 5 floating-point short constants

This procedure does the same thing as the DED directive in 930 METASYMBOL. Any number of constants may appear in the address field. Constants are generated in the order they appear. Each constant occupies two words of core, and is in the standard 930 normalized floating-point format.

The 930 METASYMBOL equivalent for the above example is:

DED 347.2, 964.3* + 8

If other than a float-point short constant appears in the address field, it will be treated as if it were a float-point short constant, and an illegal instruction format error given.

3. FLC directive. Floating-point long to 930 decimal conversion PROC. This is identical to FSC except that the constants must be long format (64-bit) floating-point numbers. If other than a floating point long constant appears in the address field, it will be treated as if it were a floating-

point long constant, and an illegal instruction format error given.

4. Data9 directive. Generate 930 compatible data:

```
Data9   'ABCD',27,θ'15',FL'368.2',FS'4E−5',X'3F'
```

The Data9 PROC allows the generation of 930 compatible constants. All constants generated are truncated to 24 bits, truncation occurring on the left. If truncation of significant data occurs, a truncation error will be signalled. BCD and float-point (long and short forms) are converted exactly as described in the BCD, FSC, FLC directives. All other data are handled exactly as they are written. For example, the above statement has the 930 METASYMBOL equivalent:

```
BCD     4,ABCD
DATA    27,015
DED     368.2,4.*+(−5)
DATA    077
```

NOTE: BCD data are always converted as if they contained 4 characters.

5. OPD directive. This directive is identical to the OPD directive in 930 METASYMBOL; however, it must appear *before* the operation to be defined as referenced.

6. RORG directive. This directive is similar to the 930 METASYMBOL RORG directive in that it allows the programmer to set the location counter; however, the code generated will be absolute or relative as determined by the load card. No equivalent to the AORG directive exists. All areas which are not used are filled with zeroes. Hence, if a RORG 100 occurs followed by three words of data, and then a RORG 203 followed by the remainder of the program, zeroes will fill locations 103 to 202.

7. FORM directive. The FORM directive is identical to the FORM directive in 930 METASYMBOL. The FORM directive must precede any reference to the FORM which is defined.

8. External references and definitions. In 930 METASYMBOL, all undefined symbols are considered external references. In Sigma 5 METASYMBOL, external references must be explicitly defined by having those symbols appear in the address field of a REF directive. Externally defined symbols are those which appear in the address field of a DEF directive.

V. Using the Secondary Loader

After a load module has been generated by the Sigma 5 loader from the code generated under the X930 PROC deck, the secondary loader produces standard XDS 930 binary records. The secondary loader will not load a program in which any errors occurred during the generation of the load module.

First, a look at the options on the Sigma 5 load card is in order. The discussion applies particularly to BPM; however, the principles involved apply to other operating systems as well.

OPTION	Effect on the secondary loader.
ABS	Absolute binary records will be produced by the secondary loader. If ABS is not specified, relocatable records will be produced.
BIAS	A bias of zero allows the user to determine the origin of this program absolutely using the RORG directive if ABS is specified, or relatively using the RORG directive if ABS is not specified. A bias other than zero relocates the program to the nearest integer multiple of X'200'. This happens regardless of the ABS option.
NOTCB	Since the program is meant for the 930, no TCB (Task Control Block) is allowed. The secondary loader cannot load a program with a TCB.
NOSYSLIB	Since the program is meant for the 930, it is impossible for any of the routines for the Sigma 5 to work in the 930. Hence, external references should not be satisfied from the system library (absence of the NOSYSLIB option), but from the appropriate user element file.
LMN	This option must be specified. The name of the LMN must be assigned to M:EI for use by the secondary loader.

Next, let us consider the input and output for the secondary loader. Input of the load module is through the system M:EI DCB. This must be assigned before the secondary loader is executed. The standard XDS 930 binary, which is produced by the secondary loader, can be output to cards or magnetic tape through the M:EO system DCB.

If output is to magnetic tape, the PACK option must be specified. When the binary has been produced, two end-of-file marks are written and control returned to the monitor.

The length of the 930 program must not exceed 8000 words. It must be remembered that the Sigma 5 loader always starts a load module at the bias value (rounded to the nearest page) and, therefore, a RORG directive causes zeroes to be inserted. (RORG 100 would cause the program to be preceded by 100 words of zeroes.) When producing the XDS 930 binary, all leading zeroes are ignored.

Only protection type (00) code is output to the M:EO file. All other protection types are ignored. The total program length (including leading zeroes) will always be even; hence, a final word of zero may occur at the end of the program.

An end transfer record is always generated. If no end transfer address was specified on the program, an end transfer address of zero is assumed. A warning is printed for absolute programs with a transfer address of zero.

VI. Progress

The X930 Program set has been successfully used to assemble on the XDS Sigma 5 computer and execute on the XDS 930 computer simple stand-alone test programs which type pre-selected BCD strings and perform simple arithmetic functions, typing the result. The programs have been loaded by the XDS 930 computer equally well from magnetic tape and cards. Programs with external references which can be satisfied by the X930 program library (making the resultant code stand alone) have also been successfully executed on the XDS 930 computer. The library routine in this case was a core-dump subroutine used to display the contents of the 930's memory on the printer and thus help verify the performance of X930.

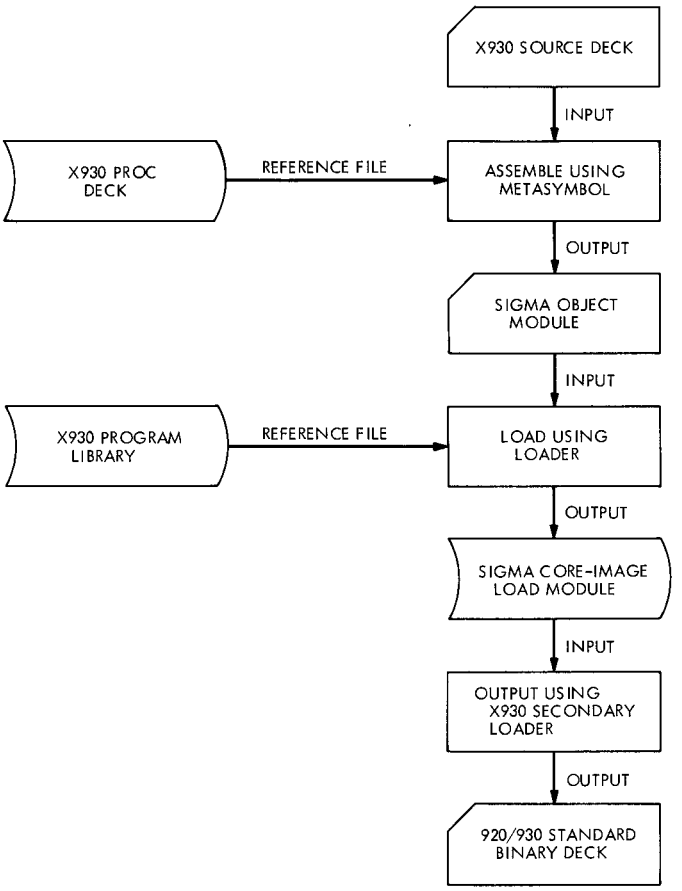

Fig. 1. Operational flow chart for the X930 program package