SPERRY UNIVAC
Operating System/3 (OS/3)

Introduction to the Assembler

SPERRY UNIVAC

UP-8030 Rev. 1

This document contains the latest information available at the time of publication. However, Sperry Univac reserves the right to modify or revise its contents. To ensure that you have the most recent information, contact your local Sperry Univac representative.

Sperry Univac is a division of Sperry Rand Corporation.

AccuScan, FASTRAND, PAGEWRITER, SPERRY UNIVAC, UNISCOPE, UNISERVO, and UNIVAC are trademarks of the Sperry Rand Corporation.

 PRINTED IN U.S.A.

preface

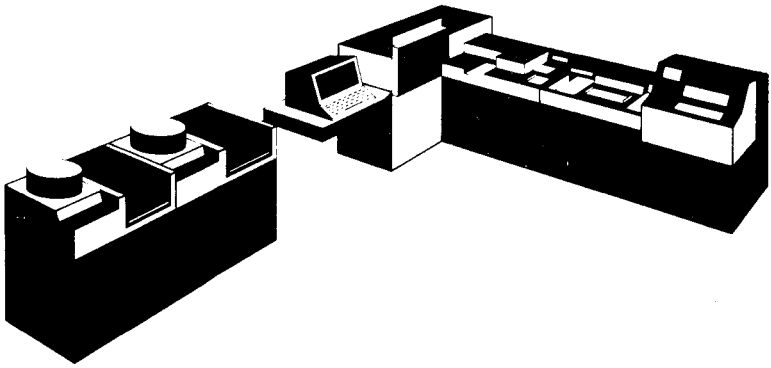
This manual is one of a series designed to introduce the software available with the SPERRY UNIVAC Operating System/3 (OS/3). The actual programming procedures required to use the software described are not included in this introductory series. Such detailed information is beyond the scope and intent of these manuals and is included in the appropriate User Guide and/or Programmer Reference.

assembly language and assemblers

Historically, the trend in programming computer systems has been away from machine language and toward high level programming languages. The first step along the way was the use of a mnemonic representation for each machine language instruction. This language was termed a "mnemonic machine language" (or an "assembly language"), and the program that translated the assembly language into code which could be executed was an "assembler".

Today's assemblers are more advanced programs, programs which are capable of much more than those first assemblers. No longer is the assembly process merely a one-for-one substitution of machine code for mnemonic instructions. Now assemblers translate very sophisticated assembly languages and the combination of language and modern processing has resulted in important advantages to the user. The advantages inherrent in the UNIVAC OS/3 Assembler are described below.

UNIVAC OS/3 Assembler



The UNIVAC OS/3 Assembler is an expandable, multi-phase language processor system that offers all the advantages of assembly language programming:

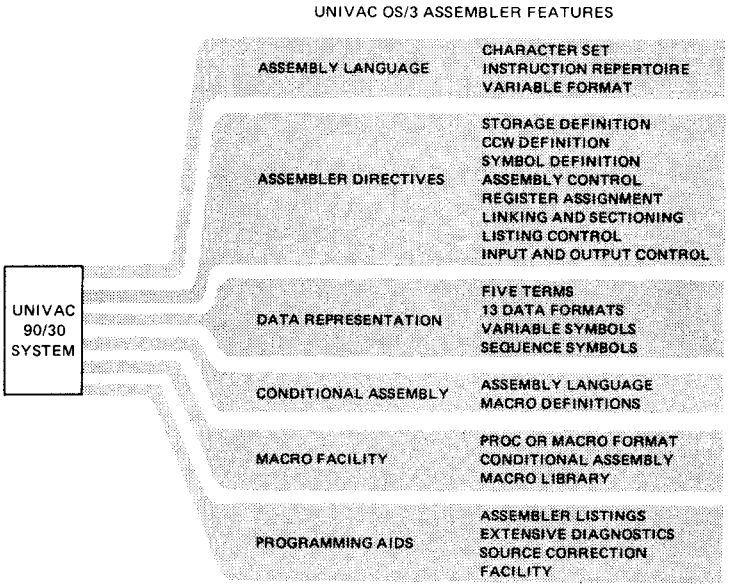
- mnemonic operation codes
- symbolic addressing and automatic storage assignment providing relocatable programs and program sectioning
- flexible data representation
- assembler control and conditional assembly

- macro and proc facilities
- source code correction facilities

With the UNIVAC OS/3 Assembler, however, these capabilities and facilities have been enhanced by the design of the assembler and the assembly language. Processing power has been increased without excessive penalties in the main storage occupied by the assembler. The UNIVAC OS/3 Assembler is a multi-phase processor written in a number of loadable sections and read into main storage as overlays during the assembly process. Should more than the minimum amount of storage be available, the assembler will expand to occupy the excess by extending its table areas and creating larger input and output buffers – greatly reducing assembly times.

The UNIVAC OS/3 Assembler is a logical extension of the assemblers used with the UNIVAC 9200/9300 Series Systems and it supports the capabilities found in the IBM System 360/20 assemblers. Programs written for these computer systems can be assembled and executed with UNIVAC OS/3, and UNIVAC OS/3 programs are upwardly compatible with the UNIVAC OS/7 Assembler as well.

The compatibility of the UNIVAC OS/3 Assembler is just one of its features. Many enhancements were made to its assembly language, its directives, its ability to represent data, its conditional assembly capabilities, its macro facility, and its programming aids.



ASSEMBLY LANGUAGE

The UNIVAC OS/3 assembly language is an enhanced, modern assembly language. It is one of the most flexible and expressive assembly languages available for a small scale computer system. This assembly language offers:

- Up to 148 instructions

  There are 83 to 148 assembly language instructions in six formats available to the UNIVAC OS/3 user.

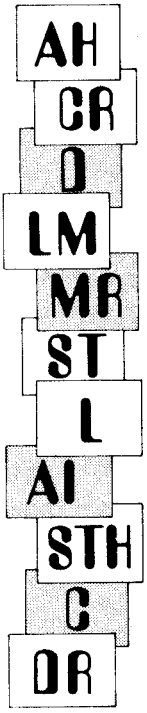
- A 54-character set

  The character set available in the assembly language contains: 26 alphabetic characters; the decimal digits, the four special letters ? $ # @, and 14 special characters:

  + – * / , = ( ) . blank & ' > <

- Variable formats

  The format of the assembly language source code instructions can be varied by using the ICTL input control assembler directive. This directive allows the UNIVAC OS/3 Assembler to accept source code which does not begin, continue, or end in the usual columns of the coding form.



All this is possible in an assembler language that is flexible enough to allow data in the instructions to be represented by five kinds of terms and 12 operators which can be used in absolute, relocatable, and complex expressions.

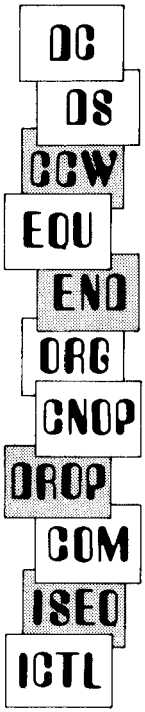
ASSEMBLER DIRECTIVES

In addition to the large repertoire of assembly language instructions available to the UNIVAC OS/3 programmer, there are three other levels of programming control: the assembler directives outlined here; the conditional assembly directives explained under the "Conditional Assembly" heading; and the macro facility directives described under the "Macro Facility" heading.

The assembler directives are written within the user's assembly language programs and are used to control the operations of the UNIVAC OS/3 Assembler. There are 26 assembler directives:

- two to define storage or constants (DS and DC)
- one to define channel command words (CCW)
- one directive for symbol definition (EQU)
- one directive for deleting an operation code (OPSYM)

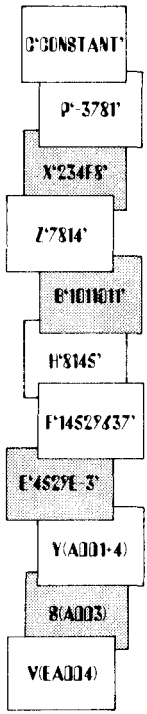
- five to direct the assembly process (START, END, ORG, LTORG, and CNOP)
- two that control base register assignments (USING and DROP)
- five linking and sectioning directives (ENTRY, EXTRN, CSECT, DSECT, and COM)
- four to control the listings printed by the assembler (TITLE, EJECT, PRINT, and SPACE)
- five for source code input and output control (ISEQ, ICTL, REPRO, PUNCH, and COPY)



The use of these assembler directives gives the UNIVAC OS/3 user programmer complete control of the assembler so that more efficient and more powerful assembly language programs can be designed.

DATA REPRESENTATION

The many forms of data representation available to the UNIVAC OS/3 user is another capability of the UNIVAC OS/3 Assembler — an extension that enables user programs to be coded with less effort because data does not have to be modified to make it fit into a limited number of possible formats.



As mentioned, data can be expressed in five kinds of terms in assembly language instructions. These terms are:

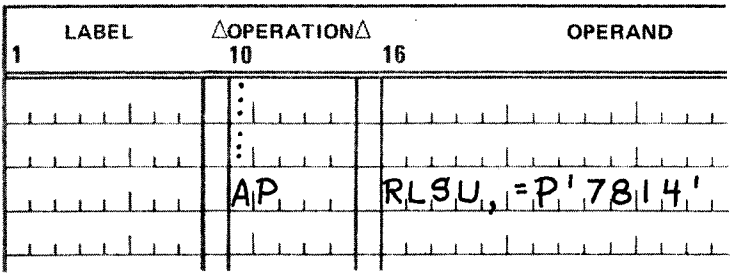
- Self-defining terms (SDT)

  Self-defining terms are fixed values coded by the user programmer that specify immediate data, masks, and so forth. SDTs may be specified as binary, hexadecimal, decimal, or character formats. The following example shows a binary SDT appearing in a compare instruction:

```
LABEL      △OPERATION△      OPERAND
1          10               16
           :
           CLI              TAG,B'00100000'
           BNER             8
           :
```

- Literals

Literals have their values specified by the user programmer and computed by the assembler. They are provided to save programming steps since they may be used to specify constants without coding a DC assembler directive. There are 12 kinds of literals, allowing the user programmer to specify everything from relocatable symbol literals to binary normalized floating-point doubleword literals. A packed decimal literal is shown in the following arithmetic instruction:

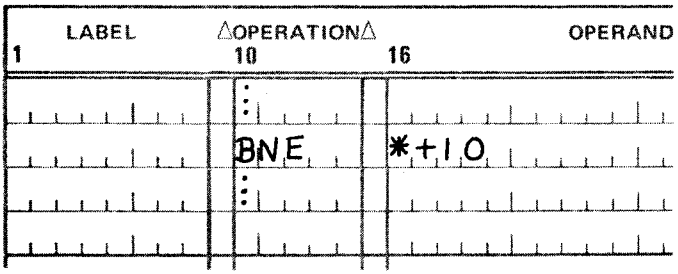


- Symbols

  The user programmer can also use symbols in the assembler language instructions. Symbols are alphanumeric characters which are assigned a value by the assembler. RLSU in the preceding example is a symbol.

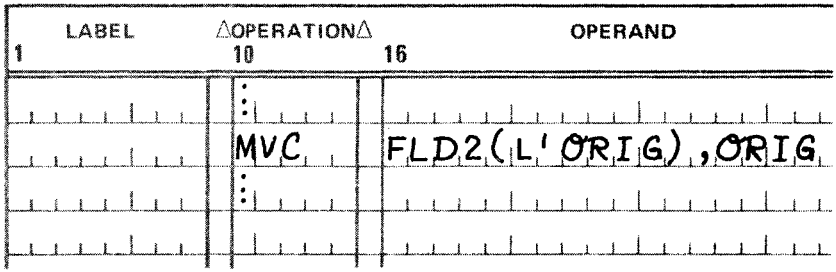
- Location counter reference

  A reference to the value of the assembler's location counter is another type of term permitted in assembly language programs. The following branching instruction contains an asterisk, which is a reference to the value of the location counter. This instruction tells the assembler to branch to a location at the location counter value plus 10 bytes.



- Length attribute

  The length assigned to a term, instruction, or storage area may be used as a term in the assembly language. In the following example, the length of the ORIG symbol is referenced:



Any type of term may be combined to form an expression by using one or more of the 12 operators available in the UNIVAC OS/3 assembly language.

Data used in, or established by, the assembler directives can take all the forms used within assembly language instructions. Actually more possible forms of data representation are available when using the assembler directives because by using the DC assembler directive, constants can be defined in character, hexadecimal, binary, packed or zoned decimal, floating-point, binary address values, base register and displacement address, and external address formats.

Additional data representations are possible at the conditional assembly and macro facility levels. The directives used at these levels can use data in all the forms discussed so far (again with the exception of the literals). In addition, the macro facility permits the use of variable and sequence symbols. Variable symbols may be symbolic parameters, set symbols, labels of DO directives, or system variable symbols. Sequence symbols are used to define branch destination points in conditional assembly directives. Variable and sequence symbols are further described in subsequent sections.

CONDITIONAL ASSEMBLY

One of the advantages of the UNIVAC OS/3 Assembler is the conditional assembly facility. By using the conditional assembly techniques outlined here, the user programmer can design one source program that can be assembled to produce one of several object programs depending on the current requirements.

The conditional assembly facility uses techniques and directives that exclude lines of coding from the output of the assembly process, that include sets of coding lines more than once, and that establish and alter values used to determine the course of assembly. Conditional assembly is possible within macro definitions and at any point within a program.

There are several techniques available to the UNIVAC OS/3 user programmer:

- Conditional assembly with variable symbols

  A variable symbol can represent:

  - a symbolic parameter, or the label or an operand of a macro instruction; by varying the label or an operand of a macro instruction, the programmer can vary the code generated when the macro is processed;

  - a set symbol, which is a value set by one of the conditional assembly SET directives and used as a counter, a switch, or a value to control the sequence of object code generation;

  - the label of a DO conditional assembly directive; or

  - one of the system variable symbols, i.e., symbols used within macro definitions to access the current program section name, a count of the number of macro instructions

processed; to refer to a positional parameter by its position within the operand field of the macro instruction; to substitute the date, time, or Julian date of the assembly; or to generate a null character string.

- Branching

  Both conditional and unconditional branch facilities are provided by the UNIVAC OS/3 Assembler. Branching is accomplished by using a sequence symbol to define a branch destination point and then using an AGO or AIF branching directive. The ANOP directive is provided to facilitate branching to a point in the program where no statement is available to define the branch destination.

- Repeating code

  The generation of object code from one set of source code lines can be repeated by using the DO, ENDO, or ACTR conditional assembly directives. These directives allow the user programmer to avoid repetitive coding; DO and ENDO are used to control conditional or repeated generation and ACTR is used to limit the number of conditional assembly directives executed within a macro or a source program.

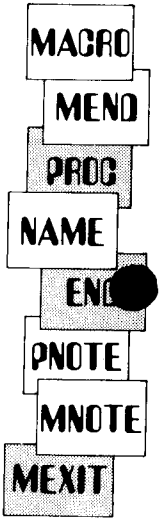
- Conditional assembly with attributes

  Symbols and macro instruction operands are assigned attributes as the source code is assembled. These attributes — type, length, scaling, integer value, count, and number — may then be used as variables in conditional assembly directives to control the assembly process.

The conditional assembly techniques provided by the UNIVAC OS/3 Assembler give the user programmer source code flexibility while ensuring that the object code generated by the assembler is precisely tailored to the data processing operations to be performed by the user's program.

MACRO FACILITY

The single most important advantage of a modern assembler is the macro facility. By using one macro instruction in the source code, many lines of code can be automatically included in the object program produced by the assembler. Macros may be written in either macro or proc format. The macros called by the macro instructions can be written by the user or can be supplied in the UNIVAC OS/3 macro library. The macros can be divided into three groups:



- System interface macros

  These macros are supplied in the system library and by using them, the UNIVAC OS/3 user programmer may access interfaces supplied by Sperry Univac between his program and all the system's supervisor and job control functions, data management software, maintenance routines, and communications facilities.

- Shorthand macros

  The macro library supplied to UNIVAC OS/3 users includes various shorthand macros that the user may call instead of writing his own sequences of code. The library contains macros for such common chores as calling standard subroutines or performing functions like moving large groups of characters in main storage.

- User-written macros

  To make it easier for the UNIVAC OS/3 user programmer to write his own macros, the macro facility features include:

  – conditional assembly within macro definitions

  – the ability to write macros in either macro or proc format

  – the ability to nest macro instructions

  – the capability of copying lines of code from library sources

The macro facility provided with the UNIVAC OS/3 Assembler is a highly sophisticated programming device that makes the assembly language more convenient without sacrificing control over the program structure. The macro facility simplifies program debugging, modification, and standardization for the UNIVAC OS/3 user.

PROGRAMMING AIDS

The UNIVAC OS/3 Assembler provides many aids to facilitate assembly language programming. The assembler provides program listings, diagnostics, and a source correction facility to reduce the user's efforts to write and debug assembly language programs.

During the assembly process, the assembler can be directed to produce several types of listings including:

- an options listing, to identify the assembler and date and time of assembly, and to produce a list of all the assembler options used;

- a listing containing all the External Symbol Dictionary Items used in the program;

- a program listing showing both the source and the object code;

- a cross reference listing in which each symbol in the source program is listed with the numbers of the defining instruction and all referencing instructions; and

- an extensive diagnostic listing.

The diagnostic listing provided by the UNIVAC OS/3 Assembler provides a detailed account of the errors the assembler encounters; the instruction line number, an error code, and a detail message are printed for each error. Every effort has been made to make this diagnostic service comprehensive by extending the number of possible messages and by including information to help the user programmer locate errors in the source code.

A source correction facility is also available to UNIVAC OS/3 users. This facility allows the programmer to use library service directives to cause the assembler to select source code from the control stream. The programmer can use the source correction facility to temporarily correct or update his source programs without a separate library services run, thereby eliminating an entire job step and further reducing the programming effort necessary to develop and maintain UNIVAC OS/3 assembly language programs.

summary

The UNIVAC OS/3 Assembler offers a flexible assembly language, comprehensive assembler directives, complete data representation, conditional assembly features, programming aids, and a versatile macro facility. And yet these features and capabilities are offered in a package small enough to run on a minimally configured UNIVAC 90/30 System. Such power and flexibility at such a low cost makes the UNIVAC OS/3 Assembler unique.

A detailed description of the features and capabilities of the UNIVAC OS/3 Assembler, complete with illustrations and examples drawn from actual practice is available in the *UNIVAC Operating System/3 Assembler User Guide.* A review of the programming used with the assembler, written for experienced personnel, is also available; see the *UNIVAC Operating System/3 Assembler Programmer Reference.*