To: MTB Distribution

From: Frank C. Helwig Jr.

Date: 2/17/82

Subject: COBOL Data Types in the Sort-Merge Facility

Introduction

If a file containing data types supported by both COBOL and PL/1 is sorted, the PL/1 sort is one-third faster than the COBOL sort. This advantage vanishes when the COBOL program is changed to use a CALL statement

```
CALL "sort" USING ...
```

in place of the SORT verb.

The performance degradation occurs because of the method chosen to make key comparisons when the SORT verb is used. Code for making key comparisons is generated within the COBOL object module which is "called" at execution time by the sort-merge facility.

The data types acceptable to the sort-merge facility should be extended to include the COBOL data types so that comparison can be done within the PL/1 programs which comprise the sort-merge facility.

The notation used below to describe syntactical constructs is the same as the BNF notation used in AG94 (Multics PL/1 Language Specification) with a single extension. If a sequence of options is enclosed within choice indicators, {¦ ¦}, one or more of the options may be chosen, but a single option may be chosen only once.

The Existing PL/1 Data Types

A particular execution of the sort-merge facility requires a sort-merge description which describes the key fields involved in the sort-merge. This is normally supplied in a segment which contains a sequence of key field descriptions having the form:

```
<datatype> <position>
```

The following PL/1 data types are presently allowed:

(1) Real Fixed Binary Data:

```
{¦ fixed {bin ¦ binary} ¦}
     ( <number of digits> )
```

(2) Real Fixed Decimal Data

```
{¦ fixed {dec ¦ decimal} ¦}
     ( <number of digits> )
```

(3) Real Floating Binary Data

```
{¦ float {bin ¦ binary } ¦}
     ( <number of digits> )
```

(4) Real Floating Decimal Data

```
{¦ float {dec ¦ decimal } ¦}
     ( <number of digits> )
```

(5) Character Data

```
{char ¦ character } ( <number of digits> )
```

(6) Bit Data

```
bit ( <number of digits> )
```

Proposed Revision of the Data Types.

New data types are proposed which will be supported by the sort-merge facility. The new types consist of further PL/1 data types and the COBOL data types. The new types will become immediately available to the COBOL user via the CALL statement. The COBOL compiler interface with sort-merge can subsequently be modified to make use of of the new data types. The syntax of the new type descriptions is shown in Appendix 1. The M numbers which appear there refer to type codes for Multics descriptors. These are described in Appendix D of the Multics Programmers Reference Manual (AG91).

Possible Future Extensions

(1) The command user should be able to define the collating sequence to be used in comparing character data items. Standard names (ASCII,EDCDIC, ...) would be provided. The ability to define arbitrary collating sequences in the manner of the COBOL alphabet name clause would be considered. If a (1-1) transformation is described then translation need only occur during the input and output phases of the sort-merge.

(2) Nasty data (e.g. COBOL display data with a trailing overpunch sign) can be transformed into nice data (e.g. packed decimal) during the input phase and be restored during the output phase.

Appendix 1.

(1) Fixed Point Data

(a) Pl/1 Specification

```
{¦ fixed {bin¦binary} ¦}
[signed ¦ {uns ¦ unsigned} ]
[aligned ¦ {unal ¦ unaligned}]
( <number of digits> )
```

```
M1  short
M2  long
M33 short unsigned
M34 long unsigned
```

(b) COBOL Specification

```
[usage] { comp-6 ¦ comp-7 }
```

```
M1 short
```

(2) Fixed Decimal Data

(a) PL/1 Specification (Packed or Unpacked)

```
{¦ fixed {dec ¦ decimal} ¦}
[aligned ¦ {unal ¦ unaligned}]
( <number of digits> )
```

```
M9  signed
M43 leading sign
```

(b) COBOL Specification (Unpacked)

```
[ [usage] display]
pic [s]9( <number of digits> )
[ {leading ¦ trailing} [separate] ]

M9  leading separate
M29 leading overpunch
M36 trailing separate
M30 trailing overpunch
M35 unsigned
```

(c) COBOL Specification (Packed, Byte Aligned)

```
{ [usage] comp-5}
pic [s]9( <number of digits> )

M39 trailing sign
M40 unsigned
```

(d) COBOL Specification (Packed, Digit Aligned)

```
{ [usage] comp-8 }
pic [s]9( <number of digits> )

M41 leading sign
M38 unsigned
```

(3) Floating Binary Data

```
{¦ float {bin ¦ binary} ¦}
( <number of digits> )

M3 short
M4 long
```

(4) Floating Decimal Data (Packed or Unpacked)

```
{¦ float {dec ¦ decimal} ¦}
[aligned ¦ {unal ¦ unaligned}]
( <number of digits> )

M10 unpacked
M44 packed
```

(5) Character Data

```
{char ¦ character} [varying]
( <number of digits> )


M21 non-varying
M22 varying
```

(6) Bit Data

```
bit [varying]
( <number of digits> )

M19 non-varying
M20 varying
```