Published: 11/22/66

Identification

Summary of Multics PL/I Compiler Specifications
D. B. Wagner, R. M. Graham, and J. D. Harkins

Purpose

The entire BP section of the Multics System Programmers' Manual is an addition to the PL/I manual specifying (1) the "implementation defined" details of the language; (2) certain sensitive areas of implementation which must be standardized to allow compatibility with the EPL (early PL/I subset) compiler being used for the initial implementation of Multics; (3) some language changes necessary for general system programming and for programming certain sensitive areas of the system; and (4) a number of items required in order for the compiler and its output procedures to fit in the Multics environment. This section summarizes these specifications and indicates where more detailed information may be found.

The criterion for language changes has been that it should be almost automatic to edit a Multics PL/I program into a form usable in other PL/I systems, and vice-versa. By "almost automatic" is meant that a program could be used to make almost all the changes necessary and notify the user of any (and all) obscure differences he must handle himself.

The standard PL/I should be taken to be that described in IBM form C28-5671-3.

Much of the information contained in the following sections has been published in other places. Whenever possible the original source is indicated. In some instances the information contained herein differs from the source document. Again whenever possible these differences are noted. However, this document (MSPM Section BP) constitutes the specifications for the Multics PL/I Compiler and supersedes all previous documentation on the subject.

The Compiler Itself

1. The compiler should be a standard Multics (pure) procedure which is callable by the Shell (BX.0.00, BX.1.00), i.e., it has a symbol table (BD.1.00) and a linkage section (BD.7.01) The call should be:

```
call pl1 (source);
```

where source is a varying character string and source||".pl1"

is the name of a Multics file containing the ASCII source program, in the form of a based character string. The length of this string is obtained by calling "get_char_ct" (see the appendix)

2. All input and output files should be accessed through segment addressing. The Multics I/O system should not be used by the compiler, except for diagnostics. This restriction is made in order to decouple development of the compiler from the development of the I/O system. In addition, it will also make the compiler more efficient by avoiding the use of a large amount of machinery. Compiler diagnostics are an exception and should be emitted via the stream-name user_output.

3. The source program may include the statement,

   % include filename;

   The entire contents of the file, "filename"||".pl1", is to be inserted at this point in the compilation. It is assumed that the contents of this file is a based character (ASCII) string. Includes should be nestable to reasonable depth.

4. The compiler should as far as possible conform to BX.0.01, Standards for Command Writers.

5. The compiler should check the list option in the user profile and if it is on, the compiler should create a file source||".list" containing, as a based character string, the ASCII text of the "compilation listing" that would have been printed on-line in a more conventional system. The user profile is described in the appendix to this section.

6. The brief option in the user profile should be tested and used to temper the verbosity of on-line diagnostics. If brief is on, only "fatal error" diagnostics should be printed on-line.

Changes and Additions to PL/I

A complete description of the changes and additions to the PL/I language as specified in IBM form C28-6571-3 will be found in BP.0.02. The following is a brief summary.

1. Freeing a based variable allocated into an area is permitted.

2. Procedures take the recursive attribute by default.

3. The standard system action for programmer-named conditions is defined.

4. All key words in the program are in lower case letters. Elsewhere both upper and lower case letters are permitted. No upper-lower case mapping should be done, e.g., "a" and "A" are different identifiers.

5. The following options have been specified for use with the options attribute.

   executeonly

   validate (proc)

   callback (a,b,...)

   rename ((id1, id2),...)

6. A new form of external reference has been defined. An identifier of the form a$b refers to symbolic location b in segment a.

7. The condition prefixes nointerrupt and interrupt have been defined to control the inhibit-interrupt bit in the compiled code.

8. A number of built-in functions have been added. They are described in BP.0.03.

9. The collating sequence for characters will be that of ASCII (see BC.2.01).

10. The double quote (") is used for string constants.

11. The data type "relative pointer" has been added.

The Compiled Program

1. The result of the compilation of a file "a.pl1" is three files:

a	(procedure segment)
a.link	(linkage segment)
a.symbol	(symbol table and binding information)

2. The procedure segment must be pure (i.e. must not modify itself).

3. The linkage segment contains information necessary for inter-segment communication and is impure since it is

modified by the linker in the course of execution. The linkage section is described in MSPM BD.7.01.

4. The symbol table provides information required for the use of data-directed-input programs, debugging programs, the Shell, and Multics gatekeepers. The general format for symbol tables in Multics is described in MSPM BD.1.00 and the details of the symbol table to be produced by PL/I are given in MSPM BD.1.02. Every symbol used in a program should occur in the symbol table for the compiled segment. It is acceptable to define an additional option for use with the options attribute which causes the compiler to generate a "short" symbol table. This short symbol table must include at least the parameter type and binding information. However, the default case should be to produce the full symbol table.

5. The binding information is used when it is necessary to bind together two or more external procedures into a single segment. This requires, among other things, the relocation of certain addresses. The required relocation information is part of the binding information. It is described in MSPM BD.2.01. Note that a compiled program may not use relocation types other than those which can be represented as indicated in BD.2.01.

6. The standard Multics call, save, and return sequences should be used for all calls to external procedures. These are described in MSPM BD.7.02 (ordinary slave procedures) and MSPM BD.7.03 (execute-only and master-mode procedures). Note that any internal procedure whose name can be passed as an argument to an external procedure must be treated according to the standards described in BD.7.02.

7. The compiler may choose not to use the standard call sequences for internal procedures which cannot in any way be involved in inter-segment communication (e.g. are not passed as arguments), but if so the symbol-table format should be extended to include information to this effect and of course such extensions should be documented.

8. One item not mentioned in BD.7.02 is the returned value of a function. The standard should be that the calling program provides the storage for the returned value and passes this as an additional argument. (Thus the count in the left half of the first word of the argument list header includes this argument.) It is acceptable to include this additional argument even in a call made by a call statement. In this case it should be a distinguishable dummy argument, since the returned value, if any, is not wanted. This permits, the compiler to generate calls in a uniform way if it wishes.

9. The argument list for a call to an internal procedure which is in any way involved in intersegment communication is embellished in the following way to allow access to variables in containing blocks: an extra ITS pair is added to the end of the argument list, as described in BD.7.02.

   This ITS pair is the value of the stack pointer (base pair sb ← sp) for the last generation of automatic storage associated with the called procedure's immediately containing block, and the first word of the argument list header contains 2 (i.e. the number of extra words) in the right half. Any call to a procedure parameter also includes this extra ITS pair, obtained in such a case from the second ITS pair in the parameter (a procedure parameter is passed as a label variable: see Representation of Data, below).

10. The argument list for a call from any procedure with the callback option specified has the following additional embellishment: ITS pairs pointing to descriptions of the arguments are added to the end of the argument list. The number of words occupied by such ITS pairs is placed in the left half of the second word of the header to the argument list. See BD.7.02 for further details.

11. All external names referenced by a compiled program which were not explicitly mentioned in the object program must be registered in the Multics External Name Registry. Registration is accomplished by submitting the names, in writing, to the MSPM editor. This includes the names of segments such as free_ and stat_ and the names of all run time routines. The External Name Registry is section BB.5.

12. It will eventually be necessary to define an additional option for use with the options attribute which causes the compiler to produce an optimized object program. However, the default case should be to avoid extra work at compilation time.

Implementation Defined Items

The implementation defined items are described in detail in BP.0.01 which is essentially the contents of repository document B0073. The following changes and additions were made, however:

1. The default linesize should be 136 (not 130) since that is the length of a line on the Anelex printer.

2. The subscript range should be bounded by -2**35 and 2**35-1.

3. All fixed-point numbers are stored as single (precision≤35 bits) or double (35 bits <precision ≤71 bits) precision binary integers. Fixed-point numbers are never stored in floating-point form.

4. The following maxima are defined:

fixed binary	≤ 72 bits
fixed decimal	≤ 21 digits
bit string	≤ 36*2**18 bits
character string	≤ 4*2**18 characters

5. The default for pagesize should be 54 lines. Assuming 6 lines per inch, this gives a one inch margin on the top and bottom of 11" forms.

6. The default area for allocation is located at

```
<free_>|[free_]
```

7. Input and output on the "standard files" SYSIN and SYSPRINT (PL/I manual, p. 100) should be connected to the standard Multics I/O system stream-names "user_input" and user_output".

8. Note specifically in BP.0.01, that during the execution of an on-unit, the on-unit current upon entry to the statically encompassing block at the time the on was executed is restablished.

Representation of Data

All data involved in any way in intersegment communication (i.e., data having the external attribute or which is passed as arguments to an external procedure) should have the representation, and be addressed, as specified in BP.2. The material in BP.2 is largely from repository documents B0022, B0056 and B0062. However, the following differences occur:

1. An area should be represented as a one-dimensional fixed-point array, and an n-dimensional array of areas should be represented as an n+1-dimensional fixed-point array.

2. Label variables should occupy six words, the two extra words to be used for special error-checking information designed and documented by Digitek.

3. The null pointer should be an ITS pair with its segment number equal to 777777(8), and its address equal to 1. That is,

```
EVEN
OCT       777777000043
OCT       000001000000
```

The segment numbered 777777(8) is a dummy segment and will never exist; any reference to it will result in a fault. Addresses other than 1 will be used for other special pointer values. For example, an address equal to 2 is an argument pointer to a null argument.

The compiler may choose to use different internal representations and different addressing schemes for data not involved in intersegment communication but if so the symbol-table format should be extended to include this information. Of course such extensions should be documented.

Asynchronous Operation

In the initial version it is not necessary to implement tasking.

Supporting Documents

The following MSPM Sections are necessary in addition to the BP sections for the complete specification of the Multics PL/I Compiler.

MSPM Section	Subject
BA.2	Summary of Multics Technical Policy
BB.5	Multics Name Registry
BB.5.01	Reserved Segment Name Suffixes
BB.5.02	Reserved External Symbols
BB.5.03	Null Pointer Assignments
BB.5.06	Segment Name Registry
BB.5.07	Reserved Stream Names
BC.2.00	Introduction to Character Input/Output
BC.2.01	The ASCII Character Set
BD.1.00	Format of Segment Symbol Table
BD.1.02	Segment Symbol Table Produced by PL/I
BD.2.01	Binding Information and Format
BD.7.01	The Linkage Section
BD.7.02	Call, Save, Return for Slave Procedures

MSPM Section	Subject
BD.7.03	Call, Save, Return for Execute Only and Master Mode Procedures
BX.0.00	Use of Commands in Multics
BX.0.01	Standards for Command Writing
BX.1.00	Multics Command Language
BX.2.00	The Shell
BX.10.00	Interactive Debugging Aids
BX.10.01	Machine- and PL/I-Oriented Interrogation and Modification of the Contents of Segments
BX.10.02	Program Tracing Under Interactive Control
BX.10.03	Breakpoint Processor
BX.10.04	Instruction-by-Instruction Interpretive Execution of Programs
BY.11.00	Overview of Error Handling
BY.11.01	Seterr
BY.11.02	Geterr
BY.11.03	Delerr
BY.11.04	Printerr