TO: MSPM Distribution
FROM: R. H. Canaday
SUBJECT: BCPL
DATE: 06/18/68

This is the first of a set of six documents on the BCPL compiler for MULTICS. All documents assume that the reader is familiar with the BCPL language as implemented on CTSS.

The documents to be issued in this series are:

BZ.6.00	Overview
.6.01	Language description
.6.02	Library routines
.6.03	BCPL under MRGEDT
.6.04	Code generation strategies
.6.05	Executing BCPL code under MULTICS

Comments will be appreciated and should be communicated to me:

Rudd Canaday
BTL - Murray Hill
Room 2C-513
(201) 582-3038

Published: 06/18/68

Identification

A BCPL Compiler for MULTICS - Overview
R. H. Canaday

Purpose

The purpose of this document is to give an overview of the BCPL compiler being implemented for MULTICS. The reader is assumed to be familiar with reference (1).

The BCPL-MULTICS compiler will accept any program acceptable to the BCPL compiler on CTSS. In addition, the language has been extended as described in BZ.6.01.

Status

A compiler is now running under GECOS which will produce EPLBSA source. Compilations can be submitted through MRGEDT. TEXT, LINK, and SYMBOL files will be returned (see MSPM BZ.6.03). This compiler produces pure procedure, with static storage in the linkage segment, and global vector in a grown segment. The compiler is expected to run under MULTICS shortly. (The GECOS BCPL compiler is itself not pure procedure, although the EPLBSA it produces is, of course, pure procedure; the MULTICS BCPL compiler is pure procedure. Note that there is no BCPL compiler under 6.36).

Language

The language accepted by the compiler consists of BCPL as described in reference (1) plus extensions as described in MSPM BZ.6.01. The most important extensions are:

(1) A full complement of single-precision floating point operations.

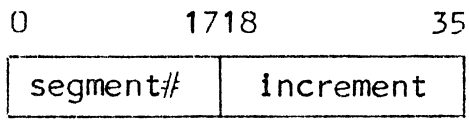
(2) String constants differentiated from character constants. This makes it possible to write a single-character string constant directly.

(3) Left-hand-side functions.

(4) Argument count available to callee.

Characteristics of BCPL Code Generators for MULTICS

(1) BCPL will handle full addresses. In BCPL an address will be packed into one word thus:



The command

```
x := lv y
```

will create such an address. The sequence

```
x := Fct

y := x(A,B)
```

which is equivalent to

```
y := Fct(A,B)
```

can refer to a BCPL function in any segment. Similarly, indirection (e.g., rv(rv A)) can cross segment boundaries.

(2) BCPL uses an abbreviated stack header and calling sequence. The header for a BCPL stack frame requires 8 words. The calling sequence is 4 instructions long. Save and return total 12 instructions (6 inline).

(3) Any BCPL routine can be called by name* from EPL or any language using the standard MULTICS conventions for argument lists and call-save-return.

(4) BCPL can call EPL through the library routine (`call') and later using the new reserved word call.

(5) Any sequence of calls between BCPL and EPL routines, including mutual recursion, is permitted.

(6) BCPL compiles pure procedure.

(7) Argument list conversion in BCPL-EPL or EPL-BCPL calls is address conversion. Thus, only single-word values can be passed directly. However, code can be written in BCPL to access or create EPL-compatible dope for any type of argument, if necessary. A library routine exists for string conversion.

---

* Initially a dummy name, e.g., "PRØ1," "PRØ2," etc., is being used in place of the source-language procedure name.

(8) BCPL procedure segments which reference global information have to be initialized prior to execution. However, additions may be made to BCPL later which will make the use of global declarations unnecessary.

(9) Any BCPL segment not communicating with other segments through global can be executed without prior initialization. All static storage is initialized on first reference to the procedure segment.

(10) BCPL segments may be bound, since BCPL uses EPLBSA and conforms to MULTICS standards in its use of the linkage segment.

Implementation

The BCPL-MULTICS compiler consists of the BCPL-GECOS compiler implemented at BTL, with extensive changes to the code generation routines. Thus the output code is essentially a transliteration from 635 code. Since the BCPL-GECOS compiler has been well tested, the MULTICS version of the compiler is expected to be fairly bug-free.

A more detailed description of the generated code is contained in MSPM BZ.6.04.

Execution of BCPL Programs

BCPL-produced TEXT and LINK may be executed under 6.36 or under MULTICS. However, the library routines cannot run under 6.36. In particular, the routines used to call EPL procedures from BCPL will execute only under MULTICS.

BCPL-to-BCPL calls do not conform to the MULTICS standard calling sequence. However, BCPL is fully MULTICS compatible in that EPL procedures are callable from BCPL (using the BCPL library routines `Getadr' and `Call'), and EPL-to-BCPL-to-EPL calling chains, including recursion, are handled correctly. Conditions arising during BCPL execution can be caught and handled by EPL procedures containing the proper ON-condition statements.

Further details are contained in MSPM sections BZ.6.02, .04, and .05.

Reference

1. M. Richards, The BCPL Reference Manual, Multics Repository Document M0099, 15 May 1968.