TO: MSPM Distribution List
FROM: L. L. Garthe
SUBJECT: BX.7.01
DATE: September 11, 1968

The following changes have been reflected in the attached writeup.

1. The command identifier has been changed from f1/1 to f1.

2. The command format has been changed by deleting parentheses, conforming to recent changes in the shell.

3. Although not significantly different from the user's point of view, the description is significantly changed primarily because of the use of working_segs. It is also more precise in that it reflects the actual implementation rather than as planned.

Published: 09/11/68
(Supersedes: BX.7.01 03/27/68)

Identification

The fl Command
L. L. Garthe

Purpose

This section describes the use of the fl command for MULTICS. A user invokes this command to assemble a single fl source program segment, producing an object program in the standard text, linkage and symbol segment formats.

For details on the fl language, or on the fl assembler, see MSPM BZ.5.00. A user's language manual is also being prepared.

Usage

When a user types a command line of one of the following forms,

```
fl   alpha
fl   alpha insymb beta
fl   alpha outsymb
```

the fl command is invoked by the Shell (MSPM BX.2), causing the fl source program contained in segment alpha to be assembled.

In the above formats, alpha and beta specify user selected segments and may be either an entry name, in which case the name should be in the user's working directory; or a path name followed by an entry name, which allows the user to specify segments in other directories of the system.

The "insymb beta" and "outsymb" control arguments may be used, by non casual users, to control assembler symbol table usage. For further details see "fl Symbol Table Usage" below.

System Option Commands

The fl command will not initially observe the options defined in MSPM BX.0.00 and BX.12.00.

fl Symbol Table Usage
Terminology Note

It should be understood, particularly by those familiar with the inner workings of MULTICS language processors, that the "fl symbol table" referred to here is not the

same as the "segment symbol table" turned out by MULTICS language processors (see MSPM BD.1.00). Although somewhat similar in function, their usage and formats differ, and they should not be confused.

Notation: In the following, ">pnam>" represents any legitimate MULTICS file system "path name", which can specify any directory in the system.

Similarly, "enam" represents a file system "entry name".

">pnam>enam" therefore represents a segment named enam in directory >pnam>.

If ">pnam>" is missing, i.e., "enam" appears alone, the directory is assumed to be the user's working directory, which is the most prevalent use.

Default Symbol Table Usage

The action for the most prevalent usage of fl, i.e.

```
fl >pnam>enam
```

is identical to the following:

```
fl >pnam>enam insymb >system_library_5>fl.symtab
```

which is defined in detail below.

Insymb Usage

The general format is:

```
fl >pnam1>enam1 insymb >pnam2>enam2
```

Assuming success, the segment named enam1.fl in directory >pnam1> would be assembled, using the fl symbol table named enam2.symtab in directory >pnam2>.

The following named segments would be produced and their names entered in directory >pnam1>.

```
enam1                    (text segment)
enam1.link
enam1.symbol
enam1.error
enam1.list
```

If there is a fatal error, only enam1.error will be produced.

Outsymb Usage

The general format for outsymb is:

fl >pnam>enam outsymb

Assuming successful assembly, the command assembles a segment named enam.fl, found in directory >pnam>. At the start of assembly the fl symbol table is empty. The assembly produces the following named segments in directory >pnam>:

enam.symtab
enam.error
enam.list

In the above, enam.symtab will be an fl symbol table which may be subsequently used in connection with the "insymb" variant of the fl command.

If fatal errors are encountered, only enam.error will be produced.

Implementation

Summary

The fl command, written in epl, provides an interface between the user and the assembler, as well as providing the principal interfaces between the assembler and MULTICS.

In general, the command analyzes the command arguments, sets up the assembler, input and output segments and calls the assembler. More details are as follows:

Preparation Prior to Assembler Call

1. Upon being called by the shell, the first action of the fl command is to analyze the strings passed (in effect the command arguments) by the shell to the command. The resulting information controls the handling of the optional "insymb" and "outsymb" arguments.

2. Pointers are then obtained to the following segments which should be available prior to the execution of the command:

   a. Source program segment

   b. fl upstairs segment

c. A segment containing the initial values of the assembler data segment.

d. If required, a segment containing the initial symbol table values to be used during assembly.

3. The array of information required by working_segs$init is then set up and working_segs$init is called (see BY.2.11) to establish the following segments used during assembly:

a. Working data segment (discarded after assembly)

b. Output Text Segment

c. Output Link Segment

d. Output Symbol Segment

e. Output Error Segment

f. Output Listing Segment

g. Output fl Symbol Table Segment, if required.

4. The initial values of the data segment, obtained in 2.c above, are copied into the working data segment, established in 3.a above.

5. If required, the initial value of the symbol table (2.d above) is copied into the temporary data segment (3.a above).

6. The outsymb argument is then checked to determine if this option exists and an indicator set to so inform the pops and the assembler, through the data segment.

7. The list option is then checked and appropriate internal switches set (deferred).

8. Pointers to all pertinent segments are then placed in the initial portion of the working data segment, allowing the POPS to reference these segments as required.

The Assembler Call

The assembler is now ready to be called. Since the assembler utilizes the 645 POPS interpreter, the call (in EPL) is:

```
call pop645$start (data_ptr);
```

where "data_ptr" is a pointer to the working data segment, which contains all other information, segment pointers, etc. necessary to properly drive the assembly.

Call Back For "Outsymb" Feature

At the end of pass 1 of the assembler the symbol table is in the proper state for the "outsymb" feature to be exercised. In order to do this the assembler logic activates a special POP which tests the contents of the symtab pointer position for null.

If so, it means that this feature is not to be exercised and control is returned to the next upstairs instruction without further action.

If not null, the special POP will call the command at a secondary entry point reserved for this purpose (fl$symout).

The action of the command will be to copy the appropriate information from the symbol table to the segment <unique_name.symtab>. Control is then returned to the calling point in the POPS, which in turn returns (in effect) to the upstairs.

The actual storing away of this segment, under the name specified by the user, is deferred until after completion of assembly.

After Return From Assembler To Command

1. A switch is available in the assembler data segment to allow the assembler to notify the command program of disastrous termination of assembly. This is the first thing checked by the command upon return from the assembler. If on, the command will dispose of the various working segments, provide an error message and return to the shell.

2. After appropriate disposition of output and working segments are determined, the command calls working_segs$finish, which destroy the working segments and files away the various output segments using the path and file names specified by the user.

3. If difficulty is encountered in filing away the various segments the following will be typed out on the user's console:

   "file system problem in accepting alpha.sffx"

   "file system error code is nnnn."

   (See MSPM BY.2.02 for error codes.)

4. Control is then returned to the Shell to await the next user command.

Errors

User errors fall into three categories each of which is covered below:

Command Format Errors

If the format of the fl command is not correct, the following message is typed out, the command is rejected and control returns to the shell to await the next command.

> "Format of fl_1 command is not correct.
>
> Check format and try again."

Segment Naming Errors

If segments specified by alpha or beta cannot be found the following message is typed out and control returns to the shell.

> "Unable to obtain a pointer to the following segment <name in error>"

If the names to be used for output segments have been previously used as directory names errors will be signalled as per working_segs$finish (BY.2.11).

Assembler Errors

Errors encountered by the assembler in the processing of fl source text normally do not result in termination of assembly. Such errors are placed in the error segment which is available after control returns to the command.

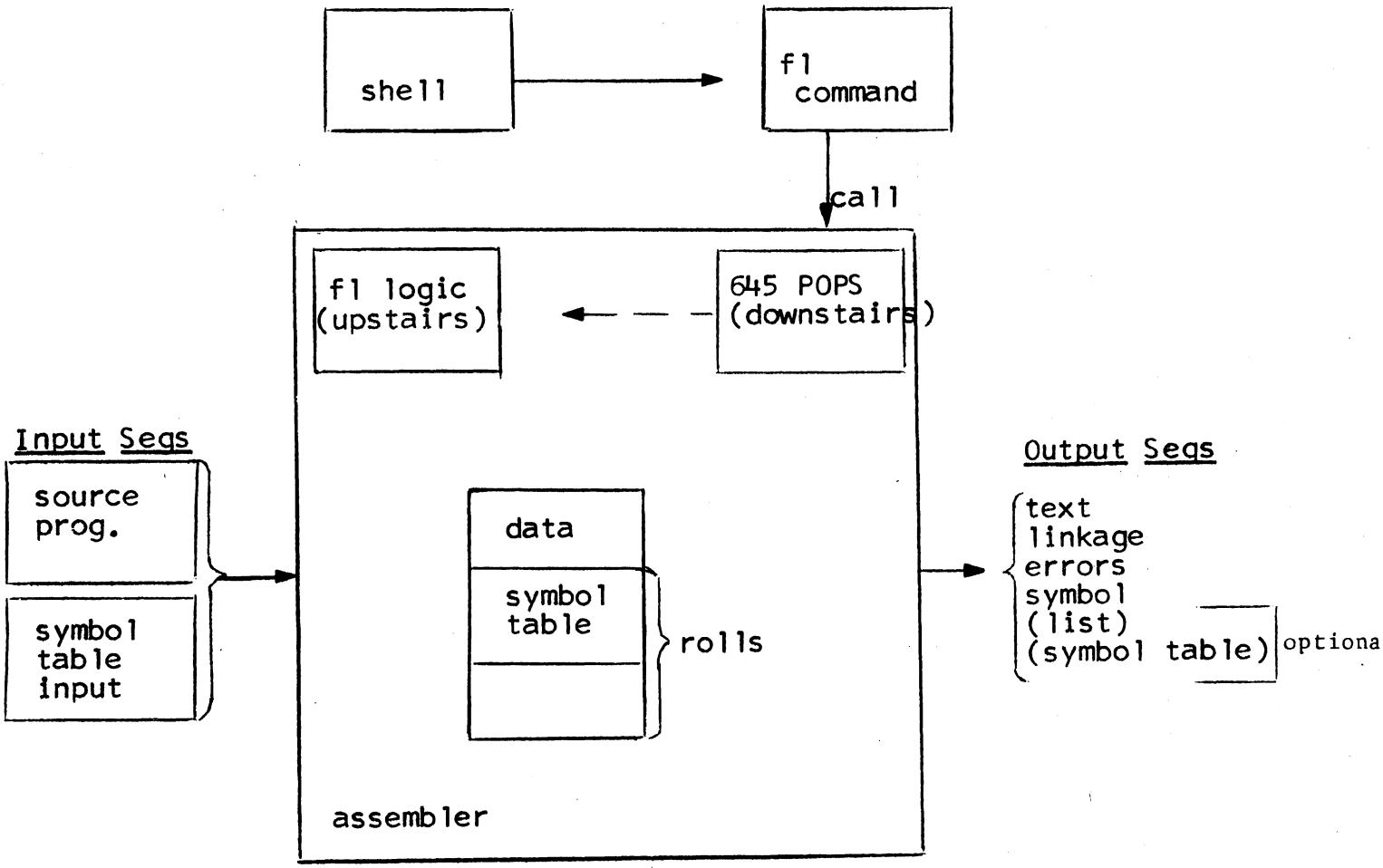
shell
fl command
call
fl logic (upstairs)
645 POPS (downstairs)
Input Segs
source prog.
symbol table input
data
symbol table
rolls
assembler
Output Segs
text
linkage
errors
symbol (list)
(symbol table)
optiona

Figure 1