Published: 11/17/67
(Supersedes: BD.7.04, 04/27/67)

Identification

Link_fault
D. L. Boyd, D. H. Johnson

Purpose

The link_fault procedure establishes intersegment references. It is designed to accomplish this in two different situations. First, whenever an initial external reference is made during the execution of a process, link_fault is called by the Fault Interceptor Module (FIM) (BK.3.03) to complete the link. Secondly, link_fault is able to force a link to be set (completed) when called as a library procedure for forcing links (BY.13.01).

Introduction

Link-fault exists in the administrative ring of Multics. It has two entry points, fim and force.

The fim entry is accessed by the following method. When a process is executing, all external references go indirectly through the linkage section of a segment (BD.7.01). The first time an external reference is made, a Fault Tag 2 (ft2 or sometimes fi) modifier in the linkage section is recognized by the GE645 hardware during address modification. This causes a fault which executes a pair of instructions in the fault vector locations for ft2 faults. The FIM is entered which in turn calls the link_fault procedure at entry fim. Link_fault changes the fault to an ITS or an ITB pair which points to the desired reference. Control is returned to the FIM which returns to the faulting procedure and execution of the external reference is then completed.

The link_fault entry for forcing links is force. It works essentially the same within link_fault as the fim entry except for the differences described in Usage. Link_fault first checks if the link is already set. If it is, link_fault does nothing and returns. After the link has been set, control is returned to the calling procedure.

Usage

There are two entries to link_fault. The call to entry fim is:

```
link_fault$fim(machcond, code)
```

```
dcl machcond (0:22) bit(36), code fixed bin(35);
```

The arguments are:

1.	machcond	bases, registers, and scu data at the time of the linkage fault.
2.	code	error code returned by link_fault.

The call to force a link is:

```
link_fault$force (pointlp, option, bases);

dcl pointlp ptr, option fixed bin(17), bases (8) bit (36);
```

The arguments are:

1.	pointlp	pointer to link which is to be forced
2.	option	if = 0, ignore trap before link if = 1, allow trap before link
3.	bases	bases to use for ITB type link

Link_fault$fim assumes that a fault occurred in a linkage section. It uses the store control unit data (scu, machine conditions) found in argument one, machcond, to determine the link pair. Link_fault$force uses pointlp, the first argument, to determine the link pair. Link_fault then looks through the linkage section pointers for the designated unlinked reference. Linking may temporarily be halted if a trap (call) before link or definition has been requested. (Note exception in next paragraph.) A trap before link means that the construction of the link is suspended while another procedure is executed. A trap before definition means that use of the definition is suspended while another procedure is executed. Link_fault ultimately constructs either an ITS or an ITB pair in the link word pair. In the case of an entry at fim it modifies the machine conditions so that the fault will not occur again when the machine is restored to its state before the fault.

When link_fault$force is called, there is an option as specified by argument two. The option is designed to regulate the use of the trap before link facility. If the argument equals 1, a trap before link is allowed. If the argument equals 0, a request to trap before link is ignored. The user must be cautious about allowing traps in this case. As an example, if link_fault, due to a trap before link calls out to another procedure which in turn calls link_fault$force to set the link that was left waiting by the original trap, an infinite loop will have been created.

In calls to entry fim, traps before link and definition are always allowed. In calls to entry force, traps before definition are always allowed.

The third argument for the entry force, bases, contains the base register information necessary for an ITB external reference. This argument is ignored if the reference is not an ITB. If the bases are needed and they are zero or not given, it is one of the errors described below.

Whenever link_fault executes a trap before link or definition request, an argument list is included in the call to the trap procedure. Both kinds of traps have the first two arguments described below. A third argument needed for a trap before definition is also described.

A pointer to the users argument list, if there is one, is passed as the first argument of the call to the trap procedure. Otherwise, a pointer to zero is given as the argument list count.

If a trap before link or definition request is executed when link_fault is entered at entry fim, a pointer to the scu data is passed as the second argument of the call to the trap procedure. The scu data is argument one, machcond. The scu data is subdivided in the following order:

user bases (words 0 through 7)

user registers (words 8 through 15)

machine conditions at time of fault (words 16 through 22)

If a trap before link or definition request is executed when the force entry is used, the scu data is not known. If there are any bases known, a pointer to them is used instead. If the bases are not known, a null pointer is used. When the trap before definition request is executed, a third argument is included in the call to the trap procedure, a flag bit. On returning from the definition trap, if the flag bit equals zero, the trap pointer is left unchanged. If the flag bit equals one, the ability to trap at the definition trap just executed is removed. If the flag bit is not set by the called procedure, link_fault sets it equal to one and proceeds as above.

Link_fault makes calls to the procedure getseg in the Segment Management Module (SMM) (BD.3.02). Getseg makes available, if possible, the segment number of the procedure

segment that is being referenced as well as the segment number of its linkage section. If the SMM returns any errors, link_fault notes an error as described below.

All calls to link_fault at entry fim are assumed to be correct and no checking is done. When link_fault is called at entry force, a check is made of the arguments given. The argument ring validating procedure, validate_arg (BD.9.03), is called to see if the procedure calling link_fault is allowed to reference the segment containing the link to be forced as well as the segment containing the bases, if given.

When an error is detected, and link_fault was called at entry force, seterr (BY.11.01) is called to put identifying information in <error_out>. The condition "link_fault_err" is then signaled. If link_fault was called at entry fim, a return is made to the FIM with the appropriate error code. The following errors are detected:

Error Number	Meaning
11	Tried to trap before link or definition with call pointer equal to 0.
12	Illegal external reference type code.
13	Fault occurred in a linkage section with no link definitions.
14	External symbol definition not found in linkage section.
15	Segment not found.
21	The second argument option, in the call to [force] was undefined.
22	Bases needed and not supplied or incorrect in call to entry [force].
31	Link not set. Illegal ring access involved in arguments of call to [force].
41	The scu data (machine conditions) were not valid.