Published: 12/13/68

Identification

Short Call/Save/Return
R. M. Graham

Purpose

The standard call/save/return is of such length that it is impractical for language translators to compile calls to short supporting procedures. In order to provide language translators with greater flexibility in their choice of tactics, extremely short call/save/return sequences have been defined. The following section defines the sequences. These sequences must be used with great caution. The final section is a short discussion of the cautions to be observed when using the short sequences.

The Short Sequences

The short call is

```
[eapap  arglist]      optional loading of arg ptr

 tsbbp  callee        transfer to callee, return is in
                      bb<-bp base pair
```

Note that no registers or bases are saved and the contents of base pair bb<-bp (and ab<-ap if used) are lost.

The short save is empty, i.e., the address callee in the tsb instruction is the actual entry point in the procedure segment and not in the linkage section as in the standard call. The stack frame is not bumped and the value of lb<-lp is unchanged.

The short return is,

```
tra bp|0
```

This returns to the instruction following the tsb instruction in the call. Note that base pair bb<-bp (and ab<-ap if arguments were passed) is not restored.

Cautions Regarding the Use of Short Sequence

The motivation for the short sequences is the desire to make it practical for compilers to call short support procedures. The following comments are made with that objective in mind.

The four ways in which the short call/save/return differs from the long are:

1. The stack is not bumped, i.e., the called procedure is not provided with a new stack frame.

2. The called procedure does not have access to its own linkage section, i.e., lp remains pointing to the caller's linkage section.

3. The return location is not left in the stack.

4. Registers and bases are not saved and restored.

These differences have several pertinent implications.

1. The two sequences are incompatible (the return information is in different places). Thus, a short (long) call may not be used to call a procedure which uses a long (short) return, respectively.

2. The called procedure cannot call another procedure or have any internal static storage. Both of these require that the procedure be able to refer to its own linkage section.

3. The called procedure may not refer to automatic storage (stack) in the usual way since the sp base is pointing to the callers stack frame.

4. The caller is responsible for saving and restoring any registers or bases whose contents he wishes preserved.

5. A standard argument list need not be used. However, whenever an argument list is used (even a nonstandard one) its location should be transmitted via the ab<-bp base pair.

6. The registers may be used to transmit both input and output arguments.

7. The stack may be used by the called procedure, but only in a parasitic fashion. The contents of sp|18 point to the beginning of the unused portion of the stack. Before using any of this space the callee must update the contents of sp|18 so that it points beyond the area to be used. This is necessary since the system assumes that the stack is unused beginning with the word pointed to by the contents of sp|18 and may use the space when processing a fault or interrupt. Before return to the caller sp|18 must be restored to its original value. The called procedure must not change the contents of the sb<-sp base pair, i.e., the stack discipline specified in BD.7.02 must be observed.

System procedures, especially the debugging aids, are written under the assumption that all procedures use the long call/save/return. In particular, the system assumes stack discipline is observed. If anything happens while control is in a procedure reached by a short call the system will not be able to provide much assistance in diagnosing the trouble. For this reason, use of the short call is suggested only for short support routines called by compiled procedures.