Author(s) : J.Zoll, R.Brun et al.
Submitter : J. Zoll
Language : Fortran or C or Assembler

Library: KERNLIB
Submitted: 27.04.1988
Revised: 20.02.1995

Calling a Subroutine by its Address

The purpose of this package is to provide a (limited) tool to connect what is called a user-routine with an arbitrary name to a `CALL` in a package, pre-existing on a library.

Because on most machines `JUMPXn` is implemented in Fortran or C, separate entries are needed for calling the user-routine with zero, one, two, ..., nine parameters.

Structure:

`SUBROUTINE` subprogram
User Entry Names: `JUMPAD`, `JUMPST`, `JUMPXn`, ($n = 0, 1, \ldots, 9$)
Internal Entry Names: `JUMPYn` (Z042) ($n = 0, 1, \ldots, 9$) (if not Assembler or C)

Usage:

Three steps are necessary:

1) Get the transfer address `IAD` of the routine (for example `TARGET`) to be called:

```
EXTERNAL TARGET
IAD=JUMPAD(TARGET)
```

2) Set the transfer address for the next transfer(s):

```
CALL JUMPST(IAD)
```

3) Execute a transfer, for a call with $n = 0, 1, \ldots, 9$ parameters:

```
    CALL JUMPX0
or  CALL JUMPX1(P1)
    ...
or  CALL JUMPX9(P1,P2,P3,P4,P5,P6,P7,P8,P9)
```

Restrictions:

Since on most machines `JUMPXn` is written in Fortran or C, the call to `JUMPXn` will be found in the trace-back of routine `TARGET`, and `RETURN` from `TARGET` will pass through `JUMPXn`. Hence, normally (i.e. unless recursion is handled by a particular machine), `TARGET` or any of its called routines may not again call `JUMPXn`.

•