digital INTEROFFICE MEMORANDUM

SUBJECT: PDP-11 Subprogram Calling Conventions

DATE: November 10, 1970

TO: PDP-11 List C
PDP-11 Master List

FROM: Hank Spencer

DEPARTMENT: Programming

Meetings were held on the above subject on 14 Oct. and 22 Oct., attended by Ad van de Goor, Hank Spencer, Larry Wade, Ken Stapleford, Larry McGowan, Ron Brender, Nick Pappas, Bruce Delagi, Lou Cohen, Jim Murphy, Jim Bell, Harlan Shepardson. It was determined that the standard for calling subroutines, as defined for the Fortran statement

```
      CALL  SUBR (A,B,...)
```

be

```
      JSR   R5, SUBR
      BR    .+N             ;N is the number of arguments
      #A                    ;followed by the addresses
      #B                    ;of the arguments.
       .
       .
       .

SUBR:  .
       .
       .
      RTS   R5
```

The calling standard for functions, i.e.

```
      X = FCTN (A,B,...)
```

was left to be determined by the Fortran implementers (N. Pappas, et al) to best suit their convenience.

A discussion of the requisites the subroutine calling standard must satisfy led to stated agreement on the following, in order of priority:

1) fast execution
2) easy to document, explain, interface to
3) short
4) provide for reentrant calls and the ability to have argument lists remote from the call
5) handle variable length argument lists
6) fail-soft (e.g. provide trace-back, user protection)

7) potential for specifying modes or uses of arguments, as might be used by Algol or Bliss.

The previous subroutine calling convention,

```
MOV #A, -(SP)        ;push argument
MOV #B, -(SP)        ;addresses on stack
  .
  .
  .
JSR  PC,SUBR
```

was said to fail in items 1), 3) and 5) and was therefore discarded in favor of the following.

The calling sequence

```
JSR  R5,SUBR
BR  .+N
A
B
.
.
.
```

provides directly for items 1) (at least for the first level of argument passing), 2), 3) and 5). (Note that the low-order byte of the BRanch instruction contains the number of arguments.)

Item 6) is provided for in that the BRanch is always safe to return to, even if the subroutine was called with the wrong number of arguments. Furthermore, the use of R5 in the call provides a trail of calling locations on the R6 stack which could be followed by a run-time trace-back routine in case of a low-level subroutine crash.

Item 7) was more or less ignored. Some vague statements were made that argument types could be specified as additional arguments if needed.

Item 4) was said to be acceptably met by the following modifications of the simple call, all of which may call the same subroutine. The reasoning is that the most common case should be the simplest, and that additional complexity is warranted if additional features are expected.

Argument list remote from the call (potentially sharable, i.e. code is separate from data in memory):

```
CALL:   .
        .                       ARGLST: JSR  R5,SUBR
        .                               BR   .+N
        JSR  PC,ARGLST                  #A
         .                              #B
         .                               .
         .                               .
                                         .
                                        RTS  PC
```

Reentrant call (the arguments are pushed on the stack and return to the main line is executed from the stack, simulating JSR R5)

```
CALL:   .
        .
        .
        MOV  R5,-(SP)           ;save R5

        MOV  #(JSR R6,@R5),-(SP)

         .
         .
         .

        MOV  #B,-(SP)           ;#B would be on the stack for reentrancy

        MOV  #A,-(SP)

        MOV  #BR.+N,-(SP)

        MOV  SP,R5              ;R5 now points at argument list

        JSR  PC,SUBR

        MOV  (SP)+,R5           ;restore R5
```

The stack now contains

```
        Ø  .
           .
           .
           PC                   following the call
           BR .+n               return is to here from SUBR
           #A
           #B
            .
            .
            .
```

```
        JSR  R6,@R5      this instruction cleans
        old R5           up the stack and returns
                         to main line via (R5)=PC
                         resulting from RTS R5 in
                         SUBR
```

The simplest call (JSR followed by in-line argument addresses) will be implemented in the present Fortran Compiler. The above illustrates the possibility of meeting additional requirements, and may be implemented in future compilers.