The MPS Debugger (this is, of course, a first draft)

I. Facilities Provided to the User

All Debugger facilities will be available to a user either through his console or via calls to Debugger routines.

A. Direct execution. Essentially any statement or procedure may be invoked. Examining values, setting values, examining MPS state information, etc., may be done with the aid of "service" routines ~~which are~~ included in the debugger.

1. Service Routines:

a. Symbolic to address mapping.
b. Address to symbolic mapping.
c. Symbolic to type mapping (using symbol table segments).
d. Typed value to output string conversion.
e. Input string to typed value conversion.
f. Stack history display.

In addition, other facilities must be available to establish a context for execution of direct statements, but it is not clear exactly what facilities are needed.

B. Breakpoints. Breakpoints may be inserted before and after any statement or procedure. A breakpoint may refer to an arbitrary procedure, which will be executed each time control arrives at the breakpoint. Facilities will be provided to set and clear breakpoints, and to list breakpoints.

C. ON-conditions and tracing. These are the extreme cases of breakpoints. In the former case, a boolean condition is evaluated after execution of each statement; if the condition ever becomes true a break is effected. These facilities are slow, but they are a real aid to finding obscure data smashing bugs.

+ Implementation schemes

II. System Facilities Needed by the Debugger

A. Find the top frame of a process.
B. Find the stack frame which called a given stack frame.
C. Find the current control location in a process.
D. Find the return location in a frame.
E. Map from a machine address to a pair (a,b) where a identifies a code module and b is a displacement within that module.
F. Map identifiers a of code modules into symbol tables for those modules.

N.B.1. Going directly from a frame pointer to a symbol table might be better if it could be arranged.

N.B.2. The ideal would be to replace E and F above with a function f which maps machine addresses into functions g from the set of base registers to symbolic record descriptors.

G. Expunge the top frame of a process.

H. Find statement boundaries in code.

I. Move an instruction out of line, fixing it up for correct execution as appropriate; replace an instruction in line. In general this requires allocating storage.

III. Questions

1. Are breakpoints to be associated with code or processes?
2. What to do about multiple names in different code segments for the same data structure?