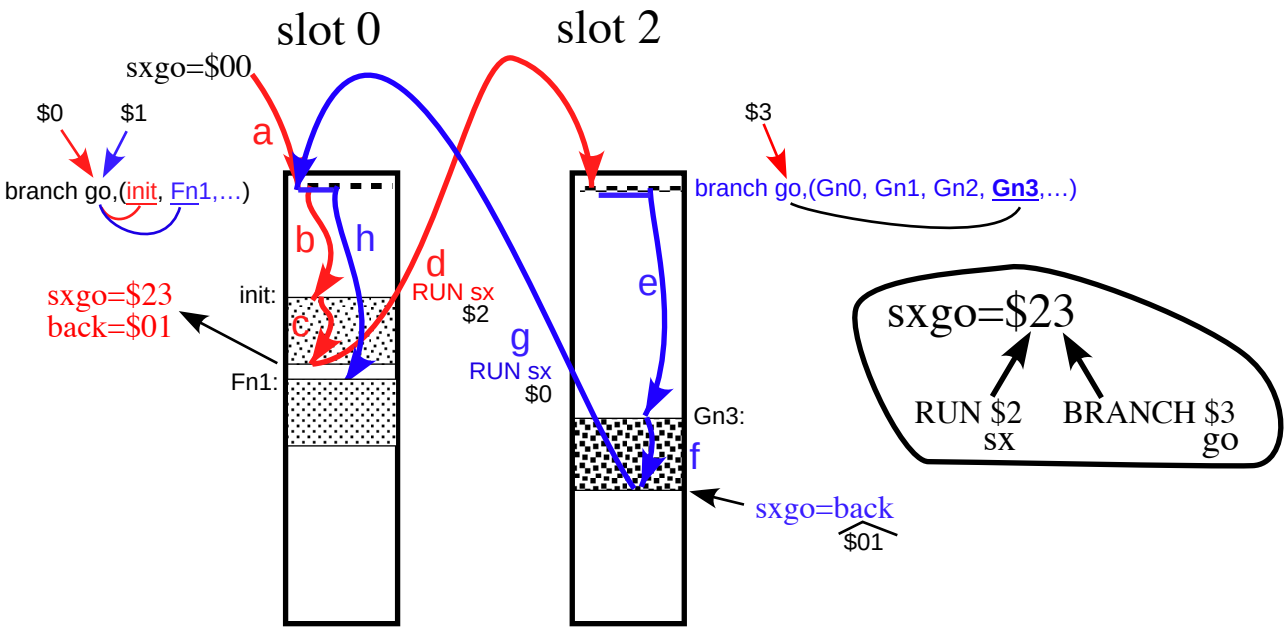
EME technique for cross-slot call

This descibes a systematic way to manage cross-slot calls, with up to 16 target or return labels per slot. It is possible to do this sort of thing ad-hoc, nonetheless, life is much easier when a consistent framework is already in place. I started using this when the BS2sx came, out and subsequently the '2e, the '2p and my current favorite the '2pe. The example shows the course of events for a cross-slot call from slot 0 to slot 2.



The technique uses two bytes, one as a forward pointer and one as a return pointer. It is like a single level return stack. Each of those bytes consists of two parts. The upper nibble is the destination slot for use with a RUN command, and the lower nibble is the index of a label within the destination slot.for use with a BRANCH command. I always make the forward pointer the first byte in memory, so that it is easy to maintain it at the same position in all slots, no matter what may happen with the other variables. The only way to be sure it is the first byte in memory is to define it as byte0 of the first word in memory.

```
wsx  VAR word
  sxgo VAR wsx.byte0   ' this is the slot swapping variable, part alias of wsx
    sx   VAR sxgo.nib1   ' for use with RUN, part aliases of both wsx and sxgo
    go   VAR sxgo.nib0   ' for use with BRANCH
back VAR wsx.byte1      ' this is the return pointer, I usually PUT it in location 126 in SPRAM.
```

The example shows a jump from slot 0 to a routine in slot 2 and then back to where it left off in slot 0.

a) Program starts in slot 0, all variables are initially zero, including the slot-swap byte, sxgo. The first program instruction is a BRANCH on the least significant nibble of $00,
b) which resolves to a branch to the label here called "init".
c) The init code executes through to a point where it needs to run a routine that happens to reside in slot 2. It points forward to the desired routine, sxgo=$23, and also sets the return pointer, back=$01, so that it can return to label Fn1 in slot 0.
d) It then executes the RUN $2 or RUN sx command, which brings it to the top of slot 2.
e) The $3 in the low nibble of sxgo is resolved by BRANCH go to label #3 in the list, "Gn3".
f) Code executes at Gn3 and eventually hits the instruction that retrieves the return pointer, sxgo=back. Now sxgo=$01.
g) Execution of RUN sx returns to the top of to slot 0.
h) The $1 in the low nibble of sxgo resolves BRANCH go to the label Fn1.

i) Execution continues and may involve several more cross slot calls with different destinations and different return points. In the process of program development it is easy enough to edit and add by changing the indices and matching destination labels. The back destination is not restricted to a specific return point. It can be used to chain several different routines with just a little advanced planning. The important thing is to have a mechanism is in place.