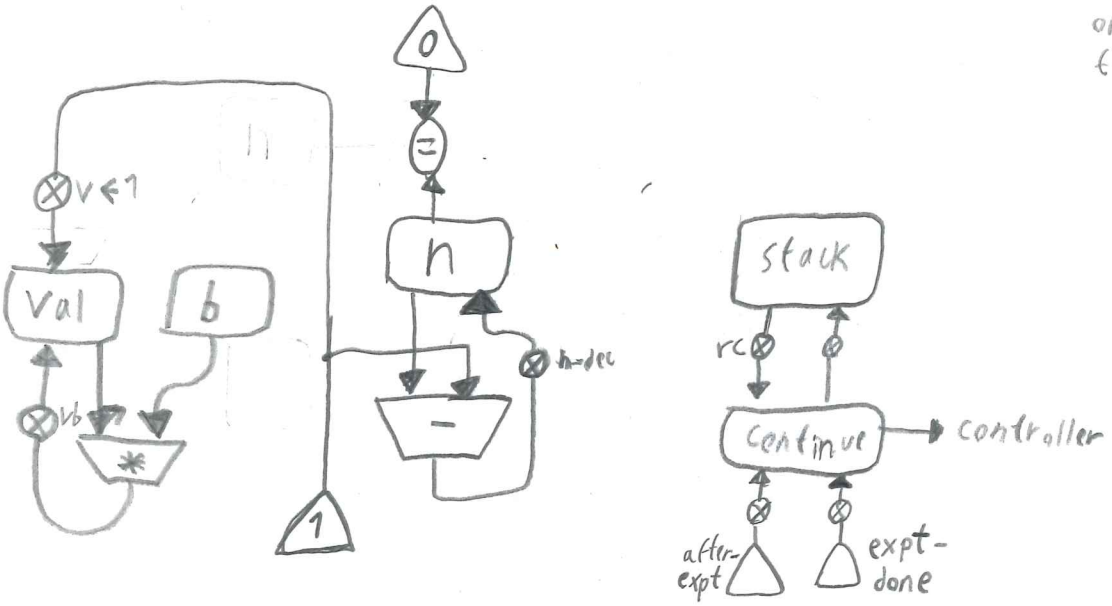
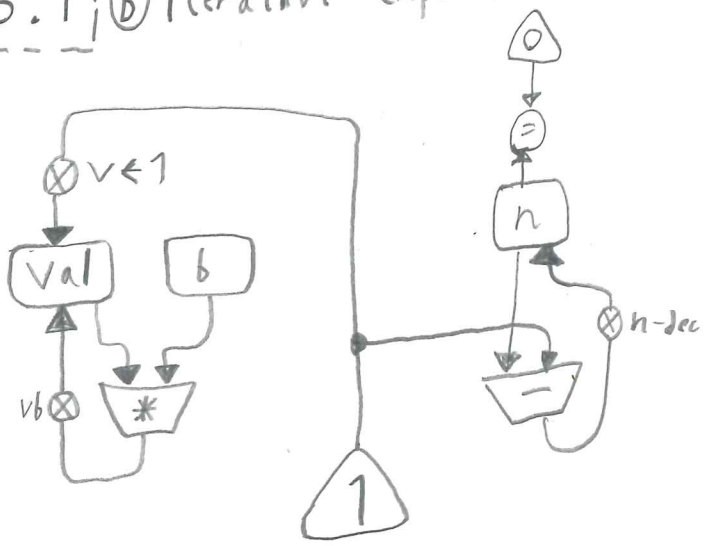
5.4 @ Recursive exponentiation

Note: n not part of calculation - don't need to save it.




```
(controller
    (assign continue (label expt-done))
expt-loop
    (test (op =) (reg n) (const 0))
    (branch (label base-case))
    (save continue)
    (assign n (op -) (reg n) (const 1))
    (assign continue (label after-expt))
    (goto (label expt-loop))
after-expt
    (restore continue)
    (assign val (op *) (reg val) (reg b))
    (goto (reg continue))
base-case
    (assign val (const 1))
    (goto (reg continue))
expt-done)
```

5.4 | (b) Iterative exponentiation




```
(controller
   (assign val (const 1))
 expt-loop
   (test (op =) (reg n) (const 0))
   (branch (label expt-done))
   (assign n (op -) (reg n) (const 1))
   (assign val (op *) (reg val) (reg b))
   (goto (label expt-loop))
 expt-done)
```

Observation: The iterative approach is simpler. It's almost as if the recursion creates a second machine to run afterwards. Note, though, that the scheme recursive expt function is simpler than the iterative expt.

Does recursion "create" potentially complexer iteration machines? w/ho knows!

Seeing register machine versions really hammer home the "build-up" versus "in-place partial solution" descriptions from chapter 1, though.