|  |  |
| --- | --- |
| TEA  PRIMITIVE | SEMANTICS |
| F: | |  |  | | --- | --- | | NAME | Fork | | PURPOSE | Conditionally Jump across a TEA program | | SYNTAX  & SEMANTICS | |  | | --- | | f: | | INERT | | f:REGEX:LA  f:REGEX:LA:LB | | If the AI matches the regular expression REGEX then jump to the block in the TEA Program under the label LA, otherwise to LB. Only the first two parameters are mandatory. And LA, LB should be valid labels declared somewhere in the program. | | f!: | | INERT | | f!:REGEX:LA  f!:REGEX:LA:LB | | Similar to f:, but the logic works same if AI does NOT match the regular expression REGEX | |  | | NOTES | It is important to note that f: is the equivalent of an IF-Statement in other languages. We shall look at some illustrative examples:  i:TEST  f:TEST:A:B  l:B  x:\_OK  q!:  l:A  r:^T:B  Would return “BEST” if AI at line#2 is “TEST”, otherwise “TEST\_OK”  Note that properly using conditional branching in TEA via the f: construct **requires some careful thought**. First, because TEA has no explicit code block construct such as using {…code} in some languages. Also, it is important to note that Label-Block can overlap based on which Label occurs first. For example, try to re-write the above program with the B-block (lines #3-5), which is our “Else-Block” being preceded by the A-block (lines #6-7), and see if it would produce the same test results as what we expect in the above example.  Another simpler, but still enlightening example is the following short program:  #shall return input shuffled if it matches “TEST”  i:TEST | f!:TEST:A:B | l:B |a!: | l:A | v: #(=TTSE, VAULTS:{"":"TTSE"}) |  | |