```
LOOP:   LODD PasCnt:            ; num of fibs to do in PasCnt
        JZER DONE:              ; no more passes, go to done
        SUBD c1:
        STOD PasCnt:            ; - - passes remaining
P1:     LODD daddr:             ; load a pointer to fib arg
        PSHI                    ; push arg for fib on stack
        ADDD c1:
        STOD daddr:             ; inc, store pointer for next d[n]
        CALL FIB:               ; call fib (arg on stack)
        INSP 1                  ; clear stack on fib return
P2:     PUSH                    ; put return AC (fib(n)) on stack
        LODD faddr:             ; load a pointer to result f[n]
        POPI                    ; pop result off stack into f[n]
        ADDD c1:
        STOD faddr:             ; inc, store pointer for next f[n]
        JUMP LOOP:              ; go to top for next pass


FIB:    LODL 1                  ; fib func loads arg from stack
        JZER FIBZER:            ; if fib(0) go to FIBZER
        SUBD c1:        ; dec arg value in AC (arg-1)
        JZER FIBONE:            ; if fib(1) go to FIBONE
        PUSH                    ; pushing arg-1
        CALL FIB:               ; fib(arg -1)
        PUSH                    ; push the above value
        LODL 1                  ; load value arg-1 to ac
        SUBD c1:        ; arg-1 is decremented i.e arg -2
        PUSH                    ; pushing arg-2
        CALL FIB:               ; calling fib(arg-2)
        ADDL 0                  ; ac = f1 + f2
        RETN;                   ; return


FIBZER: LODD c0:
        RETN                    ; AC = 0 for fib(0)
FIBONE: LODD c1:
        RETN                    ; AC = 1 for fib(1)
DONE:   HALT
.LOC    100                     ; locate data beginning at 100
d0:     3                       ; array of args for fib function
        9
        18
        23
        25
f0:     0                       ; array of result locs for fib
returns
        0
        0
        0
        0
```

```
daddr:  d0:                     ; start address of fib args
faddr:  f0:                     ; start address of fib results
c0:      0                      ; constants
c1:      1
PasCnt: 5                       ; number of data elements to process
LpCnt:   0                      ; number of fib iterations
tmp:     0                      ; initial value for fib(2)
fm1:     0                      ; at any point fib(n-1)
fm2:     0                      ; at any point fib(n-2)
```