;; original code:
;; (define f (n : int) (a : int) : int
;;     (if (= n 0)
;;         a
;; (f (- n 1) (+ a n))))
;; (f 10 0)

;; INITIAL CODE (cleaned up)

f:

pushq %rbp
movq %rsp, %rbp
subq $16, %rsp
movq %rdi, -16(%rbp) ; n
movq %rsi, -8(%rbp) ; a
cmpq $0, -16(%rbp) ; n = 0 ?
je done
movq -16(%rbp), %rax ; a = a + n
addq %rax, -8(%rbp)
subq $1,-16(%rbp) ; n = n - 1
movq -16(%rbp), %rdi ; n
movq -8(%rbp), %rsi ; a
callq f.
jmp conclusion.
done:
movq -8(%rbp), %rax ; return a

;; AFTER TAIL-CALL OPTIMIZATION:

f:

pushq %rbp
movq %rsp, %rbp
subq $16, %rsp
movq %rdi, -16(%rbp) ; n
movq %rsi, -8(%rbp) ; a
cmpq  $0, -16(%rbp) ; n = 0 ?
je done
movq  -16(%rbp), %rax ; a = a + n
addq  %rax, -8(%rbp)
subq  $1, -16(%rbp) ; n = n - 1
movq  -16(%rbp), %rdi ; n
movq  -8(%rbp), %rsi ; a
addq  $16, %rsp
popq  %rbp
jmp f
done:
    movq  -8(%rbp), %rax ; return a
conclusion:
    addq  $16, %rsp
    popq  %rbp
    retq

;; This works for any tail call, not necessarily recursive.
;; Can do better in this case -- loop optimizer could discover opportunities.