```
power:
  sub sp, sp, #24       // push 3 items to stack
  mov x9, x30           // save return loc to temp register
  stur x9, [sp, #16]    // store return loc in memory
  stur x1, [sp, #8]     // store y arg in memory
  stur x0, [sp, #0]     // store x arg in memory

  cbz x0, retzr         // branch on x == 0

  subs x10, x1, xzr     // compare y with zero
  blt retzr             // handle y < 0

  cbnz x1, recur        // branch on y > 0
  mov x1, #1            // return 1 on y == 0
  add sp, sp, #24       // pop 3 items from stack
  br x30                // return to caller

  retzr:
    mov x2, #0          // return 0 on x == 0 or y < 0
    add sp, sp, #24     // pop 3 items from stack
    br x30              // return to caller

  recur:
    sub x1, x1, 1        // decrement y
    bl power             // recursive call with (y - 1)

  ldur x0, [sp, #0]      // load x arg from memory
  ldur x1, [sp, #8]      // load y arg from memory
  ldur x9, [sp, #16]     // load return loc from memory
  mov x30, x9            // restore return location
  add sp, sp, #24        // pop 3 items from stack

  mul x2, x0, x2         // set return val to x * power(x, y - 1)
  br x30                 // end of recursive call
```