```
*
* RUN-TIME SYSTEM
*
  0:     LD  5,1(0) ; read command-line arg into r5, the funcall arg slot
  1:    LDA  6,1(7) ; store return address in r6
  2:    LDA  7,5(0) ; branch to MAIN, at [r0]+5
  3:    OUT  4,0,0  ; print value in r4, the funcall return slot
  4:   HALT  0,0,0
*
* MAIN
*
  5:    MUL  5,5,5  ; compute result
  6:    ADD  4,0,5  ; store return value from r5 into r4
  7:    LDA  7,0(6) ; store return address in r7, for branch back
```

```
*
* RUN-TIME SYSTEM
*
  0:     LD  5,1(0) ; read command-line arg into r5, the funcall arg slot
  1:    LDA  6,1(7) ; store return address in r6
  2:    LDA  7,5(0) ; branch to MAIN, at [r0]+5
  3:    OUT  4,0,0  ; print value in r4, the funcall return slot
  4:   HALT  0,0,0
*
* MAIN
*
  5:    MUL  5,5,5  ; compute result
  6:    ADD  4,0,5  ; store return value from r5 into r4
  7:    LDA  7,0(6) ; store return address in r7, for branch back
```

Calling code sets up a function call

```
*
* RUN-TIME SYSTEM
*
  0:     LD  5,1(0) ; read command-line arg into r5, the funcall arg slot
  1:    LDA  6,1(7) ; store return address in r6
  2:    LDA  7,5(0) ; branch to MAIN, at [r0]+5
  3:    OUT  4,0,0  ; print value in r4, the funcall return slot
  4:   HALT  0,0,0
*
* MAIN
*
  5:    MUL  5,5,5  ; compute result
  6:    ADD  4,0,5  ; store return value from r5 into r4
  7:    LDA  7,0(6) ; store return address in r7, for branch back
```

Called code receives an argument

```
*
* RUN-TIME SYSTEM
*
  0:     LD  5,1(0) ; read command-line arg into r5, the funcall arg slot
  1:    LDA  6,1(7) ; store return address in r6
  2:    LDA  7,5(0) ; branch to MAIN, at [r0]+5
  3:    OUT  4,0,0  ; print value in r4, the funcall return slot
  4:   HALT  0,0,0
*
* MAIN
*
  5:    MUL  5,5,5  ; compute result
  6:    ADD  4,0,5  ; store return value from r5 into r4
  7:    LDA  7,0(6) ; store return address in r7, for branch back
```

Called code returns to caller

```
*
* RUN-TIME SYSTEM
*
  0:     LD  5,1(0) ; read command-line arg into r5, the funcall arg slot
  1:    LDA  6,1(7) ; store return address in r6
  2:    LDA  7,5(0) ; branch to MAIN, at [r0]+5
  3:    OUT  4,0,0  ; print value in r4, the funcall return slot
  4:   HALT  0,0,0
*
* MAIN
*
  5:    MUL  5,5,5  ; compute result
  6:    ADD  4,0,5  ; store return value from r5 into r4
  7:    LDA  7,0(6) ; store return address in r7, for branch back
```

Calling code responds to return from call

This is ...

```
function main(n : integer)
              : integer
    n*n
```

What about ...

```
function main(n : integer)
              : integer
    square(n)

function square(m : integer)
              : integer
    m * m
```

?