```
# PROGRAM STARTS: TOP BLOCK

 0. addi sp sp 1      # variable x
 1. br 39             # jump to main function




# FUNCTION R()
                 # prologue for function r()
 2. psh ln sp        # save link register (return address)
 3. psh fp sp        # save old frame pointer
 4. mov fp sp        # update frame pointer to be the current stack pointer

                 # block of function r()
 5. br 1             # it does not have any nested function declarations, hence, br 1

                 # read statement
 6. read r0          # read statement: first read to a register
                     # read will be done to global value x
                     # since the scope is an ancestor, we need to emit code that follows the static
                     # .. link N time, where N is the difference in nesting depth, i.e., the difference
                     # .. between the levels.
 7. psh fp sp        # save the current frame pointer
 8. ld fp fp -2      # the difference in nesting depth is 1. load static parent frame pointer once.
 9. st r0 fp 1       # store the read value into the global variable, x. 1 is here the offset for var.
10. pop fp sp        # restore the current frame pointer

                 # epilogue function r()
11. mov sp fp        # tear down the stack frame
12. pop fp sp        # update to the old frame pointer
13. pop ln sp        # get the old instruction pointer (get return address)

14. ret ln           # return from function r()
```

```
# FUNCTION W()

                # prologue for function w()
15. psh ln sp         # save link register (return address)
16. psh fp sp         # save old frame pointer
17. mov fp sp         # update frame pointer to be the current stack pointer


                # block of function w()
18. br 1              # it does not have any nested function declarations, hence, br 1




                # write statement
                      # read will be done to global value x
                      # since the scope is an ancestor, we need to emit code that follows the static
                      # .. link N time, where N is the difference in nesting depth, i.e., the difference
                      # .. between the levels.
19. mov r0 fp         # get the value of the frame pointer
20. ld r0 r0 -2       # the difference in nesting depth is 1. load static parent frame pointer once.
21. ld r0 r0 1        # read the value of the global variable x to register 0
22. wr r0             # write the value of global var x, which lies in register 0

                # epilogue for function w()
23. mov sp fp         # tear down the stack frame
24. pop fp sp         # update to the old frame pointer
25. pop ln sp         # get the old instruction pointer (get return address)

26. ret ln            # return from function w()
```

```
# FUNCTION RW()

                # prologue for function rw()
27. psh ln sp         # save link register (return address)
28. psh fp sp         # save old frame pointer
29. mov fp sp         # update frame pointer to be the current stack pointer


                # block of function rw()
30. addi sp sp 1      # make space for local variable x

31. br 1              # it does not have any nested function declarations, hence, br 1

                # read statement
32. read r0           # read statement: first read to a register
                      # read will be done to a local variable, x
33. st r0 fp 1        # store the value in x (notice the stack position)

                # write statement
34. ld r0 fp 1        # load the value of x (notice the stack position)
35. wr r0             # write the value of x


                # epilogue for function rw()
36. mov sp fp         # tear down the stack frame
37. pop fp sp         # update to the old frame pointer
38. pop ln sp         # get the old instruction pointer (get return address)

39. ret ln            # return from function rw()
```

```
# MAIN FUNCTION

  # related to function call of r()
40. addi sp sp 1      # advance the stack pointer by 1 to make space for the return value
41. psh fp sp         # save the static link
42. bl -40            # branch to the address of the function r()
43. subi sp sp 1      # deallocate the space for the static link
44. pop r0 sp         # pop return value

  # related to function call of rw()
45. addi sp sp 1      # advance the stack pointer by 1 to make space for the return value
46. psh fp sp         # save the static link
47. bl -20            # branch to the address of the function rw()
48. subi sp sp 1      # deallocate the space for the static link
49. pop r0 sp         # pop return value

  # related to function call of w()
50. addi sp sp 1      # advance the stack pointer by 1 to make space for the return value
51. psh fp sp         # save the static link
52. bl -37            # branch to the address of the function w()
53. subi sp sp 1      # deallocate the space for the static link
54. pop r0 sp         # pop return value

55. hlt               # halt
```