```
fib:
  addi        sp,sp,-36
  # callee saved registers
  sd        ra,24(sp)
  sd        s0,16(sp)
  sd        s1,8(sp) # s1 is later going to catch the return value
  addi        s0,sp,36
  # caller saved registers
  sd        a0,-36(s0)
  addi  a0,a0,-1
  bgt  a0,zero,.else
  addi a0,a0,1
  beq zero,zero,.end
.else:
  ld        a0,-36(s0)
  addi        a0,a0,-1
  jal ra,fib
  # because of the recursion any caller register would be overwrit
  # the next recursive function call if the called function isn't
  # function
  mv        s1,a0
  # caller saved registers loaded whenever needed afterwards
  ld        a0,-36(s0)
  addi        a0,a0,-2
  jal ra,fib
  add        a0,s1,a0
.end:
  # don't need to add to s0, as it's anyways going to be restored
  # callee saved registers need to be restored
  ld        ra,24(sp)
  ld        s0,16(sp)
  ld        s1,8(sp)
  addi        sp,sp,36
  jalr        zero,0(ra)
  .size        fib, .-fib
  .align        1
  .globl        main
  .type        main, @function
main:
  addi        sp,sp,-16
  sd        ra,8(sp)
  sd        s0,0(sp)
  addi        s0,sp,16
  addi  a0,a0,5
  jal ra,fib
  ld        ra,8(sp)
  ld        s0,0(sp)
  addi        sp,sp,16
  jalr zero,0(ra)
```