

2.16

```
int fib(int n)
{
# Assume the argument n is in $a0, according to
# convention
    if (n == 0)
        return 0;
fib:   bne     $a0, $zero, cont    # Continue if not 0
        add     $v0, $zero, $zero # We want to return 0
        jr      $ra              # Return
    else if (n == 1)
        return 1;
cont:  addi   $v0, $zero, 1      # put 1 into $v0
        bne    $a0, $v0, cont2   # Continue if not 1
        jr     $ra
    else
        return fib(n-1) + fib(n-2);
# For each recursion, we need to know which values of n
# we are up to and the return address, so need to use
# the stack appropriately

    #Now compute fib(n-1)
cont2: addi   $sp, $sp, -8        # make (2) room on stack
        sw     $ra, 4($sp)       # push original return
        sw     $a0, 0($sp)       # push current value of n
        addi   $a0, $a0, -1      # set argument to n-1
        jal    fib              # do fib(n-1)
        addi   $t1, $zero, $v0   # store fib(n-1) in $t1
        lw     $a0, 0($sp)       # load n back from stack
        addi   $sp, $sp, +4      # adjust stack pointer
    #Now compute fib(n-2)
        addi   $sp, $sp, -8        # make (2) room on stack
        sw     $t1, 4($sp)       # push fib(n-1) on stack
        sw     $a0, 0($sp)       # push current value of n

        addi   $a0, $a0, -2      # n-2
        jal    fib
        add    $t2, $zero, $v0   # store fib(n-2) in $t2
        lw     $a0, 0($sp)       # Pop current value of n
        lw     $t1, 4($sp)       # Pop fib(n-1) into $t1
        lw     $ra, 8($sp)       # Pop original address
        addi   $sp, $sp, 12      # adjust stack pointer
    #Now compute fib(n-1)+fib(n-2)
        add    $v0, $t1, $t2     # store fib(n-2) in $t2
        jr     $ra              # Return that!
}
```

2.17

```
int fib_iter (int a, int b, int count)
{
  # Assume a is in $a0, b is in $a1 and count is in $a2
  if (count == 0)
    return b;
fib_iter: bne  $a2, $zero, cont      # carry on if not zero
          addi $v0, $zero, $a1
          jr   $ra                  # Return b
  else
    return fib_iter(a + b, a, count - 1);
cont:    add  $t0, $zero, $a0       # back up a
          add  $a0, $a0, $a1       # $a0 gets a+b
          add  $a1, $zero, $t0     # $a1 gets a
          add  $a2, $a2, -1        # $a2 gets count-1
          j   fib_iter             # tail call no return
}
```