Problem 7.1:

```
fib:
    addi $sp, $sp, -4       # make room on stack
    sw $ra, 0($sp)      # save return address
    li $t0, 2           # $t0 = 2
    li $t1, 0           # f0 = 0
    li $t2, 1           # f1 = 1
    beq $a0, 0, is0     # if (n == 0) return f0;
    beq $a0, 1, is1     # if (n == 1) return f1;
    bgt $a0, 1, isg2        # if (n >= 2) loop
is0:
    move $v0, $t1       # store f0 in $v0
    j ret           # jump back to caller of fib
is1:
    move $v0, $t2       # store f1 in $v0
    j ret           # jump back to caller of fib
isg2:
    jal fib_iter        # jump and link to fib_iter
    j ret
ret:
    lw $ra, 0($sp)      # retrieve return address
    addi $sp, $sp, 4        # pop it off the stack
    jr $ra          # jump back to caller of fib


fib_iter:
    bgt $t0, $a0, end       # if t0 > $a0 break loop
    addi $t0, $t0, 1        # t0 ++
    add $v0, $t1, $t2       # f = f0 + f1
    move $t1, $t2       # f0 = f1
    move $t2, $v0           # f1 = f
    j fib_iter          # next loop
end:
   jr $ra
```

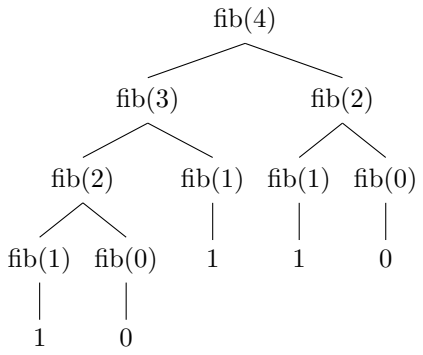
fib.asm

Problem 7.2:

Für die iterative Darstellung werden 10 Register benötigt.
Für die rekursive Darstellung werden 4 Register benötigt.

Problem 7.3:

Es werden 4 Funktionsaufrufe benötigt um auf das 4. Fibonacci-Element zu kommen.



Problem 7.4:

In der tiefsten Rekursionsstufe ist der Stack am größten (tiefste Ebene im Baum, s.o.). Für jeden Aufruf von recurse werden drei Register auf den Stack gepusht ($sp wird um 12 veringert). Dann wird die Return Address $ra in $sp + 0 und $a0 in $sp + 4 gespeichert. Das passiert bis $a0 = 1 ist. Dann wird direkt ein Wert (1) zurückgegeben und mit der Rekursion für $a0 - 2 begonnen. Dafür ist jeweils $sp + 8 reserviert.

address	value
$\$sp_0 - 36$	$\$ra_1$
$\$sp_0 - 32$	2
$\$sp_0 - 28$	0
$\$sp_0 - 24$	$\$ra_1$
$\$sp_0 - 20$	3
$\$sp_0 - 16$	0
$\$sp_0 - 12$	$\$ra_0$
$\$sp_0 - 8$	4
$\$sp_0 - 4$	0
$\$sp_0$	start value for stack pointer

Auf dem Rückweg werden in jeder Rekursion die Summe der beiden Kinder eines Knotens gebildet und in $sp + 8 gespeichert. Danach werden wieder drei Register gepopt in dem $sp um 12 erhöht wird.