```
1  ##############################################################################
2  # Perfect number check                                                       #
3  # @param int the number you would like to check                              #
4  # @result int 0 if the number is not perfect, otherwise 1                    #
5  ##############################################################################
6
7  .data
8      prompt: .asciiz "Positive integer you would like to check : "
9      output: .asciiz "Is a perfect number (1: Yes, 0: No): "
10 .text
11 .globl main
12 main: li $v0 , 4          # |
13     la $a0 , prompt       # |
14     syscall               # |=> Print string "prompt"
15     li $v0 , 5            # |
16     syscall               # |=> Ask for integer A
17
18     # Initialise variables
19     move $s0 , $v0        # => Store A in $s0
20     li $s1 , 0            # => The sum of all proper divisors of A
21     li $s2 , 1            # => start here with checks for devisors
22
23 s:  bgeu $s2, $s0, eval # while $s2 < $s0
24     rem $t0, $s0, $s2    # $t0 = $s0 % $s2
25     bne $t0, $0, w
26     addu $s1, $s1, $s2   # $s1 += $s2
27 w:  addi $s2, $s2, 1     # $s2++
28     j s;                  # /endwhile
29
30 eval: seq $s0, $s0, $s1 # Compare the sum of divisors with A
31     li $v0 , 4            # |
32     la $a0 , output       # |
33     syscall               # |=> Print string "output"
34     la $v0 , 1            # |
35     move $a0 , $s0        # |
36     syscall               # |=> Print $s0
37     jr $ra
```

aufgabe–5.s

```
1  ##############################################################################
2  # @param string a \0 terminated string                                       #
3  # @return string the ROT-13 encrypted string                                 #
4  ##############################################################################
5
6  .data
7      prompt: .asciiz "Please enter string: "
8      output: .asciiz "ROT-13: "
9      plain:  .space 64
10 .text
11 .globl main
12 main:
13     li       $v0, 4       # |
14     la       $a0, prompt  # |
15     syscall               # |=> Print string "prompt"
16     li       $v0, 8       # |
17     la       $a0, plain   # | => Ask for string plain
18     li       $a1, 64      # |
19     syscall               # | => read a string with max. 64 chars
20     li       $t2, 10      # Stop by \n
21
22     # Loop over all characters
23     la       $t1, ($a0)   # => $t1: the current adress that gets modified
24
25 s:  lb       $t0, ($t1)   # => $t0: the current value (char)
26     beq $t0, $t2, out     # while $t1 != '\n'
27     li $t3, 64
28     bge $t3, $t0, w       # if $t0 <= 64: jump to w
29     li $t3, 123
30     bge $t0, $t3, w       # if $t0 >= 123: jump to w
31     li $t3, 90
32     bge $t3, $t0, big     # if $t0 <= 90: jump to big
33     li $t3, 96
34     bge $t3, $t0, w       # if $t0 <= 96: jump to w
35     j small
36 w:  addi $t1, $t1, 1      # $t1++
37     j s;                  # /endwhile
38
39 small:
40     addi     $t0, -84     # -97 + 13
41     rem      $t0, $t0, 26 # $t0 %= 26
42     addi     $t0, 97
43     sb       $t0, ($t1)
44     j w
45
46 big:
47     addi     $t0, -52     # -65 + 13
48     rem      $t0, $t0, 26 # $t0 %= 26
49     addi     $t0, 65
50     sb       $t0, ($t1)
51     j w
52
53 out:
54     li       $v0, 4       # |
55     la       $a0, output  # |
56     syscall               # |=> Print string "output"
57
```

```
58      la      $v0, 4          # |
59      la      $a0, plain      # |
60      syscall                 # |=> Print plain
61      jr      $ra
```

aufgabe–6.s