

# Übungsblatt 4

## Übungsgruppe Pentium

Stefan Schmauch, Nina Cami, Anton Lydike

Mittwoch 20.05.2020

**Aufgabe 1)**

\_\_\_ /4p.

```

    add t1, t0, 1      #S1
    sll t4, t1, t1     #S7 result overwritten by S4
    add t3, t0, 20     #S3
    sll t4, t1, 4      #S4 this is okay, since S7 is never executed in the
                        original
    add t2, t1, 100     #S2
    and s1, t1, t1     #S5
    sub s2, t3, t4     #S6
    j label
    add s2, t4, t3     #S8
label:
    add t0, t0, t0     #S9

```

**Aufgabe 2)**

\_\_\_ /5p.

```

    .data 0x200
fibs: .space 56

    .text
main:
    add s1, zero, 0    # storage index
    add s2, zero, 56   # last storage index
    add t0, zero, 1    # t0 = F_{i}
    add t1, zero, 1    # t1 = F_{i+1}

loop:
    sw t0, fibs(s1)    # save
    add t2, t1, t0     # t2 = F_{i+2}
    add t0, t1, 0      # t0 = t1
    add t1, t2, 0      # t1 = t2
    add s1, s1, 4      # increment storage pointer
    blt s1, s2, loop   # loop as long as we did not reach array length
    # exit gracefully
    add a0, zero, 0
    add a7, zero, 93
    scall              # exit with code 0

```

**Aufgabe 3)**

\_\_\_ /1+2+3p.

Name	Größe	Latenz
L1-Cache	2 <sup>11</sup> Bytes	8 Cycles
L2-Cache	2 <sup>17</sup> Bytes	30 Cycles
L3-Cache	2 <sup>22</sup> Bytes	120 Cycles
Arbeitsspeicher	2 <sup>27</sup> Bytes	< 400 Cycles

## Aufgabe 4)

\_\_\_ /5+3p.

```

        .data
seed_val:
        .space 4
array:   .space 40

        .text
main:
    # seed the generator
    add a0, zero, 42
    jal seed
    # generate numbers
    add s1, zero, 40          # address in out array
    add a1, zero, 1          # we want numbers from 0 to 255
main_loop:
    add s1, s1, -4            # one address to the left
    jal rand                  # generate a random number
    sw  a0, array(s1)         # save it to the array
    bne s1, zero, main_loop   # repeat until we saved array(0)
    add a7, zero, 93          # exit syscall
    add a0, zero, 0          # exit code 0
    scall

# seed the random number generator
# input register: a0 (read only)
seed:
    sw  a0, seed_val(zero)    # write a0 to seed_val
    ret

# generate a random number
# input register:  a1
# output register: a0
# output: if a1 is 0, a random 4byte integer. If a1 is not 0, a random 1byte integer
rand:
    lw  a0, seed_val(zero)    # load seed into a0 to save a register
    add t0, zero, 73          # get 73 into t0, we can override values here
                                # since these are not marked as save
    mul a0, a0, t0            # a0 = a0 * 73
    add a0, a0, 691           # a0 = a0 + 691
    sw  a0, seed_val(zero)    # set our new random number as seed
    beq a1, zero, rand_ret    # if a1 == 0, skip reduction
    and a0, a0, 0xFF
rand_ret:
    ret

```

Gesamtpunkte:

\_\_\_ /23p.