Übungsblatt 2

Übungsgruppe Pentium

Stefan Schmauch, Anton Lydike

Donnerstag 07.05.2020

Aufgabe 1) **__ /5+1p.**

a)

```
        .data
buf:    .space 10
        .align 2

        .text
main:                                   # Programmbeginn

# Zeichenkette einlesen
        add     a0, zero, 0             # stdin
        add     a1, zero, buf           # Adresse des Puffers
        add     a2, zero, 10            # Maximal 10 Zeichen
        add     a7, zero, 63            # read
        scall
        nop                             # nach syscall 63 immer 3 Nops
        nop
        nop

# Zeichenkette in Zahl umwandeln (zahl = n)
        add     a1, zero, a0            # Länge der Zeichenkette
        add     a0, zero, buf           # Adresse der Zeichenkette
        jal     str2int
        add     s0, zero, a0            # s0: enthält das aktuelle teilprodukt n!/((n-s0)!)
        add     s1, zero, s0            # s1: enthält
        add     s2, zero, 3             # s2 = 3
# begin loop
fac_start:
        add     s1, s1, -1
        mul     s0, s0, s1
        bge     s1, s2, fac_start       # jump to fac_start, while s1 >= 3


# Ergebnis in string umwwandeln
        add     a0, zero, s0
        add     a1, zero, buf
        jal     int2str

# String ausgeben
        add     a2, zero, a0            # Länge steht in a0
        add     a0, zero, 1             # stdout
        add     a1, zero, buf           # Adresse des Puffers
        add     a7, zero, 64            # syscall 64: write
        scall


# Programm beenden
        add     a0, zero, a0            # eingelesene Zahl als Exitcode
        add     a7, zero, 93            # sycall 93: exit
        scall
```

b)

Betrachte:

$$12! = 479001600 < 2^{31} = 2147483648 < 2^{32} = 4294967296 < 13! = 6227020800$$

12! ist die letzte zahl die wir in einem signed 32 bit register speichern können. 32 bit unsigned lässt 13! immer noch nicht zu, da $13! > 2^{32} = 4294967296$

Aufgabe 2) **__ /5p.**

```
        .data
buf:    .asciiz "[Object object]"
        .align 2

len:    .space 4
        .align 2

        .text
main:                                   # Programmbeginn

# Registerbelegung:
# s0: das geladene byte
# s1: verschiebung im text buffer
# (wird zu beginn um 1 erhöht, deshalb beginnen wir mit -1)
        add     s1, zero, -1


loop:
        add     s1, s1, 1                   # increment pointer
        lbu     s0, buf(s1)                 # load byte from buff at position s1
```

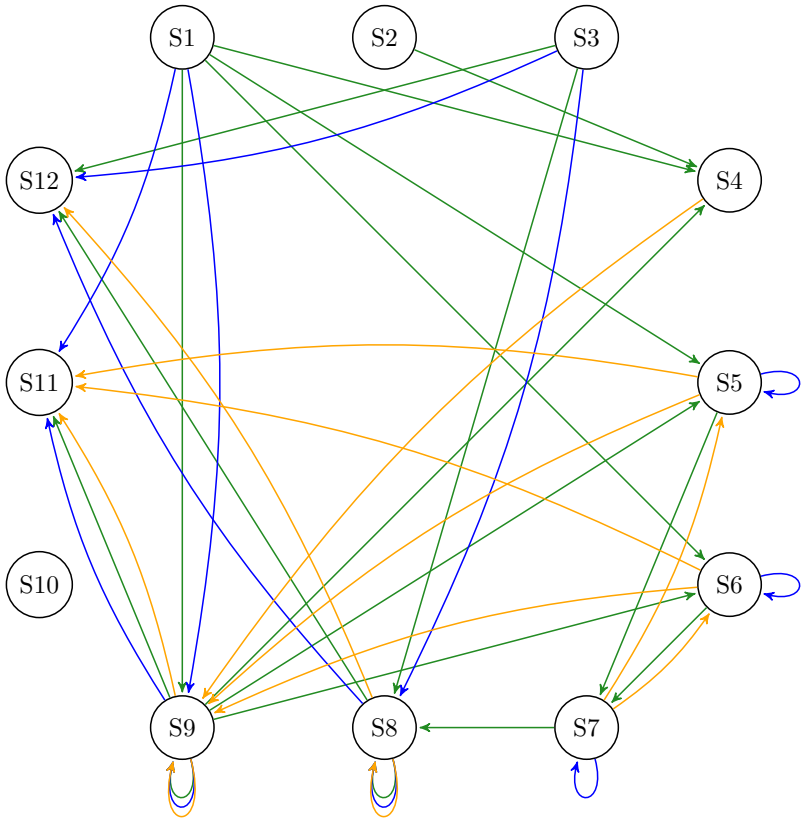
```
        bne     s0, zero, loop              # repeat if non-null byte

# Ergebnis in string umwwandeln
        add         a0, zero, s1
        add         a1, zero, len
        jal         int2str

# String ausgeben
        add     a2, zero, a0            # Länge steht in a0
        add     a0, zero, 1             # stdout
        add     a1, zero, len           # Adresse des Puffers
        add     a7, zero, 64            # syscall 64: write
        scall

# Programm beenden
        add     a0, zero, a0            # eingelesene Zahl als Exitcode
        add     a7, zero, 93            # sycall 93: exit
```

Aufgabe 3) **__ /2+2+1p.**

RISC	CISC
0x100	0x100
0x104	0x104
0x108	0x020
0x020	0x108
0x10c	0x30
0x110	-

CISC ist schneller, da es einen RAM Access weniger gibt.

Aufgabe 4) **__ /8p.**



Gesamtpunkte: **__ /24p.**