Thomas Scharinger Matr. Nr. 11777710

Problem 1)

```
        add   x7, x0, x0
Jump1:  add   x29, x0, x0
        bge   x7, x5, jump2
        bge   x29, x6, jump2
jum2:   add   x30, x29, x7
        slli  x31, x30, 5
        sd    x30, 0(x31)
        addi  x29, x29, 1
        addi  x29, x29, 1
```

// i<a → cont
// j<b → cont
// i+j
// $2^2=4$ $2^3=8$ 2+3=5

~~Version 2.0~~:)

```
        add   x7, x0, x0
Jump1:  add   x29, x0, x0
Jump2:  add   x30, x29, x7
        slli  x31, x29, 5
        sd    x30, 0(x31)
        addi  x29, x29, 1
        blt   x29, x6, Jump2
        addi  x7, x7, 1
        blt   x7, x5, Jump1
```

// i=0
// j=0
// $x_{30} = i+j$
// $x_{31} = 2^5 \cdot j = j \cdot 4 \cdot 8$
// store
// j = j+1
// j < b
// i = i+1
// i < a

Problem 2)

Q1) No. There are no structural hazards because we have separate Instruction and Data-Memories.

Q2) Because the register source(s) and register destination are always located in the same positions of the instruction.

Problem 3)

1.) Offset: 4 bits $\Rightarrow 2^4 = 16$ bytes per line

2.) Index: 5 bits $2^5 = 32$

$\Rightarrow$ 32 lines

3.) Cache implementation

Data: $16 \cdot 64 \cdot 8 =$ 8192 bits

1 valid bit per line 64 bits

23 bit per line for tags $64 \cdot 23 =$ 1472 bits

= 1536 bits for cache implementation

4.)

byte adresses	line	Hit/Miss
9	0	Miss
12	0	Hit
100	6	Miss
256	16	Miss
1	0	Hit
270	16	Hit
275	17	Miss
268	16	Hit
12	0	Hit
2000	61	Miss
15	0	Hit
268	16	Hit

0 0 - 16
1 16 - 32
2 32 - 48
3 48 - 64
4 64 - 80
5 80 - 96
6 96 - 112
7 112 - 128
8 128 - 1
. . .

2000 / 16 = 125

125 mod 64 =
125 → 61
126 62
127 63
128 0

5.) Hit ratio: $\frac{7}{13}$

Problem 4) 16 byte cache line size

RisC 5 : 8 byte per data

=) Every second access is a miss

=) 512 Misses and 512 Hits

Double line size: => Every fourth access is a miss

=) 256 Misses and 768 Hits

Problem 5)

Q_1: CPI, Instruction count, cycle time

Q_2: When we have to load a page from the disk

Q_3: We use the TLB to get frequently or recently used data faster. Temporal locality!