Submission #1

Assignment 1.1

hostname

```
rspi10.inf-ra.uni-jena.de
```

lscpu

```
 Architecture:            aarch64
  CPU op-mode(s):        32-bit, 64-bit
  Byte Order:            Little Endian
CPU(s):                  4
  On-line CPU(s) list:   0-3
Vendor ID:               ARM
  Model name:            Cortex-A72
    Model:               3
    Thread(s) per core:  1
    Core(s) per cluster: 4
    Socket(s):           -
    Cluster(s):          1
    Stepping:            r0p3
    CPU max MHz:         1500,0000
    CPU min MHz:         600,0000
    BogoMIPS:            108.00
    Flags:               fp asimd evtstrm crc32 cpuid
Caches (sum of all):
  L1d:                   128 KiB (4 instances)
  L1i:                   192 KiB (4 instances)
  L2:                    1 MiB (1 instance)
Vulnerabilities:
  Itlb multihit:         Not affected
  L1tf:                  Not affected
  Mds:                   Not affected
  Meltdown:              Not affected
  Mmio stale data:       Not affected
  Retbleed:              Not affected
  Spec store bypass:     Vulnerable
  Spectre v1:            Mitigation; __user pointer sanitization
  Spectre v2:            Vulnerable
  Srbds:                 Not affected
  Tsx async abort:       Not affected
```

Assignment 1.2

l_data1 = **00000001**

```
unsigned char l_data1  = 1;
1 = 00000001 in binär
```

l_data2 = **11111111**

```
unsigned char l_data2  = 255;
255 = 11111111 in binär
```

l_data3 = **00000000**

```
unsigned char l_data3  = l_data2 + 1;
255 + 1 = 100000000 in binär, die führende 1 wird fallen gelassen (größer als 8bit)
und übrig bleibt 00000000 (0).
Es ist ein Überlauf aufgetreten.
```

l_data4 = **10100001**

```
unsigned char l_data4  = 0xA1;
0xA1 = 16 * 10 + 1 * 1 = 161 = 10100001 in binär
Umwandlung von Hexadezimal in Binär
```

l_data5 = **01001011**

```
unsigned char l_data5  = 0b1001011;
0b1001011 ist bereits in binär (ohne die 0b am Anfang)
```

l_data6 = **01001000**

```
unsigned char l_data6  = 'H';
'H' = 01001000 in ASCII kodiert
```

l_data7 = **11111100**

```
char l_data7  = -4;
-4 = 11111100 in binär (Zweierkomplement / Radixkomplement)
```

l_data8 = **00000000000000000000100000000000**

```
unsigned int l_data8  = 1u << 11;
```

```
1u = 1 (unsigned int) um 11 Stellen nach links verschoben
also eine 1 mit 11 Nullen dahinter
```

l_data9 = **00000000000000000000000000000000**

```
unsigned int l_data9  = l_data8 << 21;
weiter 21 Stellen nach links verschoben macht insgesamt eine 1 um 32 Stellen nach
links verschoben
=> 1 mit 32 Nullen dahinter, 1 ist also auf Stelle 33
=> Overflow, fliegt runter, Rest sind nur Nullen
```

l_data10 = **00000111111111111111111111111111**

```
unsigned int  l_data10  = 0xFFFFFFFF >> 5;
0xFFFFFFFF = 32 bit mit 1en
>> 5 => sie werden um 5 Stellen nach rechts verschoben
die ersten 5 Stellen werden zu Füllnullen, der Rest sind Einsen
```

l_data11 = **00000000000000000000000000000110**

```
unsigned int  l_data11 = 0b1001 ^ 0b01111;
^ ist der xor Operator, das heißt jede Stelle wird mit gleicher Stelle aus der anderen
Zahl xor'ed
die Zahlen sind schon in Binär gegeben, also einfach die Stellen vergleichen
1001 xor 01111 = 0110, der Rest sind Füllnullen
```

l_data12 = **11111111111111111111111111110110**

```
unsigned int  l_data12 = ~0b1001;
~ ist der Einerkomplement Operator, das heißt jede Stelle wird invertiert
1001 = 0110, der Rest waren Füllnullen in 0b1001, die dann invertiert zu 1 werden
```

l_data13 = **00000000000000000000000001010000**

```
unsigned int  l_data13 = 0xF0 & 0b1010101;
& ist der bitwise und Operator, also nur 1, wenn beide 1 sind
0xF0 = 11110000, 0b1010101 = 01010101
11110000 & 01010101 = 01010000, Rest Füllnullen
```

l_data14 = **00000000000000000000000000000101**

```
unsigned int  l_data14 = 0b001 | 0b101;
| ist der bitwise oder Operator, also 1, wenn einer der beiden 1 ist
```

```
001 | 101 = 101, Rest Füllnullen
```

l_data15 = **00000000000000000001111000111111**

```
unsigned int  l_data15 = 7743;
7743 = 00000000000000000001111000111111 in Binär
```

l_data16 = **11111111111111111110000111000001**

```
int  l_data16 = -7743;
-7743 = 11111111111111111110000111000001 in Binär (Zweierkomplement / Radixkomplement)
```