

Rechnerstrukturen: Übungsblatt 5

Florian Ludewig (185722)

15. Juni 2020

Aufgabe 1

Aufgabe 1 (Ich nehme an wir können die Zahlen in Bitlänge 8 umrechnen)

$$\begin{aligned}
 B_0 &= 1 & *30 &= 30 \\
 B_1 &= -\frac{1}{2} \left[\left(\frac{2}{0} \right) \cdot 1 \right] = -\frac{1}{2} & *30 &= -15 \\
 B_2 &= -\frac{1}{3} \left[\left(\frac{3}{0} \right) \cdot 1 + \left(\frac{2}{1} \right) \cdot B_1 \right] = -\frac{1}{3} \cdot \left(-\frac{1}{2} \right) = \frac{1}{6} & *30 &= 5 \\
 B_3 &= -\frac{1}{4} \left[\left(\frac{4}{0} \right) \cdot 1 + \left(\frac{3}{1} \right) \cdot \left(-\frac{1}{2} \right) + \left(\frac{2}{2} \right) \cdot \frac{1}{6} \right] = 0 & *30 &= 0 \\
 B_4 &= -\frac{1}{5} \left[\left(\frac{5}{0} \right) \cdot 1 + \left(\frac{4}{1} \right) \cdot \left(-\frac{1}{2} \right) + \left(\frac{3}{2} \right) \cdot \frac{1}{6} + \left(\frac{2}{3} \right) \cdot 0 \right] = -\frac{1}{30} & *30 &= -1
 \end{aligned}$$

a) 30: 00011110_2
 $\begin{array}{r} 11100001 \text{ EK} \\ + 00000001 \\ \hline 11100010 \text{ ZK} \end{array}$

-15: 0001111_2
 $\begin{array}{r} 10001111 \\ 01110000 \text{ EK} \\ + 00000001 \\ \hline 01110001 \text{ ZK} \end{array}$

5: 0000101_2
 $\begin{array}{r} 11111010 \text{ EK} \\ + 00000001 \\ \hline 11111011 \text{ ZK} \end{array}$

0: 0000000_2
 $\begin{array}{r} 11111111 \text{ EK} \\ + 00000001 \\ \hline 11111111 \text{ ZK} \end{array}$
 ... Overflow

-1: 10000001_2
 $\begin{array}{r} 01111110 \text{ EK} \\ + 00000001 \\ \hline 01111111 \text{ ZK} \end{array}$

b) 30: $30/8 = 3 \text{ R } 6 \uparrow$
 $3/8 = 0 \text{ R } 3 \uparrow$
 036_8

-15: $11110001_2 \text{ ZK(15)}$
 $= 241_{10}$
 $241/8 = 30 \text{ R } 1 \uparrow$
 $\text{R } 6$
 $\text{R } 3$
 361_8

5: $5/8 = 0 \text{ R } 5$
 005_8

0: $0/8 = 0 \text{ R } 0$
 000_8

-1: 01111111_2 ZK(1)
 $= 255_{10}$
 $255/8 = 31 \text{ R } 7 \uparrow$
 $31/8 = 3 \text{ R } 7 \uparrow$
 $3/8 = 0 \text{ R } 3$
 377_8

c) 30: $30/16 = 1 \text{ R } 14 \uparrow$
 $1/16 = 0 \text{ R } 1 \uparrow$
 $1E_{16}$

-15: $11110001_2 \text{ ZK(15)}$
 $= 241_{10}$
 $241/16 = 15 \text{ R } 1 \uparrow$
 $15/16 = 0 \text{ R } 15 \uparrow$
 $F1_{16}$

5: $5/16 = 0 \text{ R } 5$
 05_{16}

0: $0/16 = 0 \text{ R } 0$
 00_{16}

-1: 01111111_2 ZK(1)
 $= 255_{10}$
 $255/16 = 15 \text{ R } 15 \uparrow$
 $15/16 = 0 \text{ R } 15 \uparrow$
 FF_{16}

Aufgabe 2

```
1  #include <stdio.h>
2  #include <limits.h>
3  #include <time.h>
4  #include <stdlib.h>
5
6  unsigned W = sizeof(unsigned) * 8;
7  unsigned x, y;
8
9  int arithmeticRightShift(int x, int n) {
10     if (x < 0 && n > 0)
11         return x >> n | ~(~0U >> n);
12     else
13         return x >> n;
14 }
15
16 unsigned a() {
17     return ~(arithmeticRightShift((x|(~x+1)), W-1))&1;
18 }
19 unsigned b() {
20     return ~(arithmeticRightShift(x, W-1) << 1);
21 }
22 unsigned c() {
23     return ~(~x|(y^(INT_MIN+INT_MAX)));
24 }
25 unsigned d() {
26     return x^(INT_MIN+INT_MAX);
27 }
28 unsigned e() {
29     return ((x^y)&~y)|(~(x^y)&y);
30 }
31 unsigned f() {
32     return arithmeticRightShift(((x<0) ? (x+3) : x), 2);
33 }
34
35 void test_all(unsigned expected_value) {
36     printf("a: %0x ?== %0x (match: %d)\n", expected_value, a(), expected_value
37         == a());
38     printf("b: %0x ?== %0x (match: %d)\n", expected_value, b(), expected_value
39         == b());
40     printf("c: %0x ?== %0x (match: %d)\n", expected_value, c(), expected_value
41         == c());
42     printf("d: %0x ?== %0x (match: %d)\n", expected_value, d(), expected_value
43         == d());
44     printf("e: %0x ?== %0x (match: %d)\n", expected_value, e(), expected_value
45         == e());
46     printf("f: %0x ?== %0x (match: %d)\n", expected_value, f(), expected_value
47         == f());
48 }
49
50 int main(void) {
51     srand(time(NULL));
52     x = rand();
53     y = rand();
54
55     printf("1:\n");
56     test_all(x);
57
58     printf("\n2:\n");
59     test_all(x&y);
60
61     printf("\n3 positive x:\n");
62     test_all((x<0?1:-1));
63     x = (~-1 * x)+1;
```

```

58     printf("\n3 negative x:\n");
59     test_all((x<0?-1));
60
61     return 0;
62 }

```

Die Ausgabe des obigen C-Programms ist:

```

1  1:
2  a: 47ad6e0f ?== 0 (match: 0)
3  b: 47ad6e0f ?== ffffffff (match: 0)
4  c: 47ad6e0f ?== 43800800 (match: 0)
5  d: 47ad6e0f ?== b85291f0 (match: 0)
6  e: 47ad6e0f ?== 47ad6e0f (match: 1)
7  f: 47ad6e0f ?== 11eb5b83 (match: 0)
8
9  2:
10 a: 43800800 ?== 0 (match: 0)
11 b: 43800800 ?== ffffffff (match: 0)
12 c: 43800800 ?== 43800800 (match: 1)
13 d: 43800800 ?== b85291f0 (match: 0)
14 e: 43800800 ?== 47ad6e0f (match: 0)
15 f: 43800800 ?== 11eb5b83 (match: 0)
16
17 3 positive x:
18 a: ffffffff ?== 0 (match: 0)
19 b: ffffffff ?== ffffffff (match: 1)
20 c: ffffffff ?== 43800800 (match: 0)
21 d: ffffffff ?== b85291f0 (match: 0)
22 e: ffffffff ?== 47ad6e0f (match: 0)
23 f: ffffffff ?== 11eb5b83 (match: 0)
24
25 3 negative x:
26 a: ffffffff ?== 0 (match: 0)
27 b: ffffffff ?== ffffffff (match: 1)
28 c: ffffffff ?== 0 (match: 0)
29 d: ffffffff ?== fffffffe (match: 0)
30 e: ffffffff ?== 1 (match: 0)
31 f: ffffffff ?== 0 (match: 0)

```

Wodurch sich folgende Lösungen ergeben:

- 1) e
- 2) c
- 3) b