Übung RA, Kapitel 1.2

Teil 1: Zahlen und Logik

A) Aufgaben zu den ganzen Zahlen

1. Konvertieren Sie die folgenden Zahlen in die Binärform:
   1984

Immer durch 2 teilen, der Rest ergibt das Bit. Jeweils mit dem LSB anfangen.
(LSB: Least Significant Bit, MSB: Most Significant Bit)

1984	:2 =	992	(LSB) Rest: 0
992	:2 =	496	0
496	:2 =	248	0
248	:2 =	124	0
124	:2 =	62	0
62	:2 =	31	0
31	:2 =	15	1
15	:2 =	7	1
7	:2 =	3	1
3	:2 =	1	1
1	:2 =	0	1 (MSB)

1 1 1 1 1 0 0 0 0 0 0

1984d = 11111000000b

4000

/2 = 2000 0 (LSB)
/2 = 1000 0
/2 = 500 0
/2 = 250 0
/2 = 125 0
/2 = 62 1
/2 = 31 0
/2 = 15 1
/2 = 7 1
/2 = 3 1
/2 = 1 1
/2 = 0 1 (MSB)
4000d = 111110100000b

8192

/2 = 4096 0 (LSB)
/2 = 2048 0
/2 = 1024 0
/2 = 512 0
/2 = 256 0
/2 = 128 0
/2 = 64 0
/2 = 32 0
/2 = 16 0
/2 = 8 0
/2 = 4 0
/2 = 2 0
/2 =1 0
/2 =0 1 (MSB)
8192d = 10000000000000b

2. Wie lautet 1001101001 (binär) dezimal, oktal, hexadezimal?

Dezimal: 2^n rechnen, addieren

1	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**1**
2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
512	256	128	64	32	16	8	4	2	1
512			**64**	**32**		**8**			**1**

512 + 64 + 32 + 8 +1 = 617d (=1001101001b)

Oktal: 3er Bit umrechnen:

001	**001**	**101**	**001**
1	1	5	1

1001101001b = 1151o

Hexadezimal: 4er Bit umrechnen:

0010	**0110**	**1001**
2	6	9

1001101001b = 269h

3. Welche der folgenden Darstellungen sind gültige hexadezimale Zahlen?
BED, CAB, DEAD, DECADE, ACCEDED, BAG, DAD.

Aller, ausser BAG! (G ist nicht gültig in Hex! Nur 0-9, A-F)

4. Stellen Sie die Dezimalzahl 100 unter Benutzung der Basiszahlen von 2 und 8 dar.

Basiszahl 2 => Binär => 100d = 1100100b

Basiszahl 8 => Oktal = > 100d = 144o

5. * Wie viele verschiedene positive Ganzzahlen können in k Ziffern mit Hilfe von r Basiszahlen ausgedrückt werden?

r^k

B) Aufgaben zu den Gleitkommazahlen

6. Konvertieren Sie die folgenden Zahlen in das einfachgenaue lEEE-Format. Geben Sie die Ergebnisse als acht hexadezimale Ziffern aus.

a) 9

*1) Zahl ist positiv => **SignBit = 0b** (0 = positiv, 1 = negativ)*

2) Exponent bestimmen: Zahl mit 2^x dividieren bis ein Wert zwischen 1 und 2 entsteht
9 / 2^1 => 9 / 2 = 4.5
9 / 2^2 => 9 / 4 = 2.25
*9 / 2^3 => 9 / 8 = 1.125 => **Exponent** ist 2^**3, Faction** ist **0.125** (Resultat – 1, die 1 ist „virtuell“)*

3) Exponent berechnen und in BIN wandeln => Bei IEEE 754, Single precision ist der Excess 127 (d.h 127 offset)
***Exponent**: 127 + **3** = **130d** => Dez2Bin: **1000 0010b** (Exponent ist 8Bit(single precision, double: 11bit))*

*4) Fraction(=Mantisse) berechnen und in BIN darstellen – **ggf runden=> ungenau!!!***
1. Bit nach dem Punkt: 2^-1 = 0.5
2. Bit nach dem Punkt: 2^-2 = 0.25
3. Bit nach dem Punkt: 2^-3 = 0.125
4. Bit nach dem Punkt: 2^-4 = 0.0625
5. Bit nach dem Punkt: 2^-5 = 0.03125
Etc.
***Fraction** 0.125 => 2^-3 => **0010 0000 0000 0000 0000 000b** (Fraction ist 23Bit (single precision, double: 52bit))*

5) Floating-Point Sign+Exponent+Fraction(Mantisse) zusammensetzen: (single precision total 32bit, double 64bit)
01000001000100000000000000000000b => 0100 0001 0001 0000 0000 0000 0000 0000b

6) Bin2Hex

Bin	0100	0001	0001	0000	0000	0000	0000	0000
Hex	4	1	1	0	0	0	0	0

9d** => als IEEE 754 Floating-point in HEX: **41100000h

b) 5/32

1) Sign Bit:
*5/32 = 0.15625 ist positiv => **SignBit = 0***
2) Exponent:
*0.15625 * 2 = 0.3125 2^-1*
*0.3125 * 2 = 0.625 2^-2*
*0.625 * 2 = 1.25 2^-3*
*0.15625 = 1.25 * 2^-3 => **Exponent** ist -3 => Excess-127 => 127 + (-3) = **124d => 0111 1100b***
3) Fraction/Mantisse:
0.25 = 2^-2 = 010…0b
4) Floating-Point (single precision)
0011 1110 0010 0000 0000 0000 0000 0000b

Bin	0011	1110	0010	0000	0000	0000	0000	0000
Hex	3	E	2	0	0	0	0	0

5/32 = 0.15625d = 3E200000h

c) -5/32

1) Sign Bit:

-5/32 = -0.15625 ist negativ => ***SignBit = 1***

2) Exponent:

*0.15625 * 2 = 0.3125 2^-1*

*0.3125 * 2 = 0.625 2^-2*

*0.625 * 2 = 1.25 2^-3*

*0.15625 = 1.25 * 2^-3 =>* ***Exponent*** *ist -3 => Excess-127 => 127 + (-3) =* ***124d => 0111 1100b***

3) Fraction/Mantisse:

0.25 = 2^-2 = 010…0b

4) Floating-Point (single precision)

1011 1110 0010 0000 0000 0000 0000 0000b

Bin	*1011*	*1110*	*0010*	*0000*	*0000*	*0000*	*0000*	*0000*
Hex	*B*	*E*	*2*	*0*	*0*	*0*	*0*	*0*

-5/32 = -0.15625d = BE200000h

d) 6.125

1) Sign Bit:

6.125 ist positiv=> ***SignBit = 0***

2) Exponent:

6.125 / 2 =3.0625 2^1

3.0625 / 2 = 1.53125 2^2

*6.125 = 1.53125 * 2^2 =>* ***Exponent*** *ist 2 => Excess-127 => 127 + 2 =* ***129d => 1000 0001b***

3) Fraction/Mantisse:

0.53125

2^-1 = 0.5 1 (MSB)

2^-2 = 0.25 0

2^-3 = 0.125 0

2^-4 = 0.0625 0

2^-5 = 0.03125 1 (LSB)

0.53125 = 2^-1 + 2^-5 = 100010…0b

4) Floating-Point (single precision)

0100 0000 1100 0100 0000 0000 0000 0000b

Bin	*0100*	*0000*	*1100*	*0100*	*0000*	*0000*	*0000*	*0000*
Hex	*4*	*0*	*C*	*4*	*0*	*0*	*0*	*0*

6.125d = 40C40000h

7. Konvertieren Sie die folgenden einfachgenauen IEEE-Gleitkommazahlen von Hexadezimal in Dezimal:

a) 42E48000h

Hex	4	2	E	4	8	0	0	0
Bin	0100	0010	1110	0100	1000	0000	0000	0000

0 1000 0101 1100 1001 0000 0000 0000 000

SignBit: 0 => positive Zahl

Exponent: 1000 0101 => 133 Excess-127 => 133 – 127 = 6 => 2^6

Mantisse: 1100 1001

*=> (1 * 2^-1) + (1 * 2^-2) + (0 * 2^-3) + (0 * 2^-4) + (1 * 2^-5) + (0 * 2^-6) + (0 * 2^-7) + (1 * 2^-8)*

=> 0.5 + 0.25 + 0.03125 + 0.00390625 = 0.78515625

Alternative: 1100 1001 = 201 / 256 (da 8bit) = 0.78515625

Zusammensetzen:

*(+1) * (**1** + 0.78515625) * (2^6) = **114.25***

42E48000h = 114.25d

b) 3F880000h

Hex	3	F	8	8	0	0	0	0
Bin	0011	1111	1000	1000	0000	0000	0000	0000

0 0111 1111 0001 0000 000000000000000

SignBit: 0 => positive Zahl

Exponent: 0111 1111 => 127 => Excess-127 => 127-127 = 0 => 2^0

Mantisse: 0001 => 2^-4 = 0.0625

*(+1) * (1 + 0.0625) * (2^0) =1.0625*

3F880000h = 1.0625d

c) 00800000h

Hex	0	0	8	0	0	0	0	0
Bin	0000	0000	1000	0000	0000	0000	0000	0000

0 0000 0001 0000 0000 0000 0000 0000 000

SignBit: 0 => positive Zahl

Exponent: 0000 0001 => 1 => Excess-127 => 1-127 = -126 => 2^-126

Mantisse: 0000 => 0

*(+1) * (1 + 0) * (2^-126) =1.17549 * 10^-38*

00800000h = 2^-126 (= 1.17549 * 10^-38d)

d) C7F00000h

Hex	C	7	F	0	0	0	0	0
Bin	1100	0111	1111	0000	0000	0000	0000	0000

1 1000 1111 1110 0000 0000 0000 0000 000

SignBit: 1 => negative Zahl

Exponent: 1000 1111 => 143 => Excess-127 => 143-127 = 16=> 2^16

Mantisse: 1110 => 0.5 + 0.25 + 0.125 = 0.875

*(-1) * (1 + 0.875) * (2^16) = -122‘880.0*

C7F00000h = -122‘880.0d

8. In einer bestimmten Situation kann eine Operation auf zwei Gleitkommazahlen eine drastische Reduzierung der Anzahl der signifikanten Bits im Ergebnis verursachen. Warum ist das der Fall?

Bei Annäherungswerten, fast gleicher Zahlen.

*Bsp: 0.21347 – 0.21345 = 0.00002 => 0.2 * 10^-4 (nur noch 1 signifikante Stelle!)*

C) Aufgaben zur Bool'schen Algebra

9. Aufgabe 1 Hamburger oder Hot-Dog

Ein Logiker fährt in ein Drive-in-Restaurant und sagt: "Ich möchte einen Hamburger oder einen Hot-Dog und Pommes Frites." Der Koch hat die Schule zu häufig geschwänzt und weiss nicht (oder kümmert sich nicht darum), ob "UND" Vorrang "ODER" hat. Für ihn ist das eine so gut wie das andere. Welche der folgenden Fälle sind gültige Interpretationen der Bestellung? (Im Englischen hat das Wort "OR" ["ODER"] die Bedeutung von "EXCLUSIVE OR" [ausschliessliches oder].)

a. Nur ein Hamburger. - gültig
b. Nur ein Hot-Dog. - ungültig
c. Nur Pommes Frites. - ungültig
d. Ein Hot-Dog und Pommes Frites. - gültig
e. Ein Hamburger und Pommes Frites. – gültig (sofern OR vor AND gilt)
f. Ein Hot-Dog und ein Hamburger. - ungültig
g. Alle drei. - ungültig
h. Nichts - der Logiker bleibt zwar klug, fährt aber hungrig von dannen. – ungültig (war eine gültige Aussage)

10. Aufgabe 2 ** Missionar

Ein Missionar hat sich in Südkalifornien verirrt und stoppt an einer Kreuzung. Er weiss, dass zwei Motorradbanden die Gegend unsicher machen. Eine davon sagt immer die Wahrheit, und die andere lügt immer. Er möchte wissen, welche Strasse nach Disneyland führt. Welche Frage sollte er stellen?

Er fragt die Motorradbande A welche Antwort Motorradbande B sagen würde.

Bsp 1: A lügt und B sagt die Wahrheit:

A sagt B würde nach links sagen => Da B die Wahrheit sagt, A jedoch lügt geht es nach rechts.

Bsp 2: A lügt und B sagt die Wahrheit:

A sagt B würde nach rechts sagen => Da B die Wahrheit sagt, A jedoch lügt geht es nach links.

Bsp 3: A sagt die Wahrheit und B lügt:

A sagt B würde nach links sagen => Da B Lügt, A jedoch die Wahrheit sagt, geht es nach rechts.

Bsp 4: A sagt die Wahrheit und B lügt:

A sagt B würde nach rechts sagen => Da B Lügt, A jedoch die Wahrheit sagt, geht es nach links.

➔ Es ist immer die andere Richtung als die Antwort ist.
Da eine Gruppe immer lügt wird die Antwort „invertiert“.

11. Aufgabe 3 * Bool'sche Funktionen

Es gibt vier boolesche Funktionen einer einzigen Variablen und 16 Funktionen von zwei Variablen.
Wie viele Funktionen von drei Variablen gibt es?

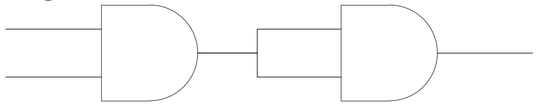
256

Bei 3 Variablen gibt es 8 Zeilen (mögliche Bit-Muster). Somit 2^8 = 256.

12. Aufgabe 4 Wahrheitstabelle
Benutzen Sie eine Wahrheitstabelle, um folgendes zu zeigen:
P = (P AND Q) OR (P AND NOT Q).

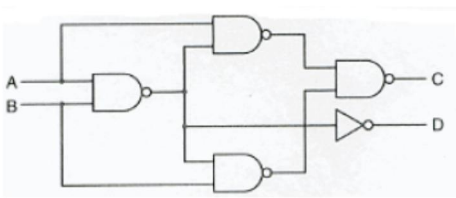
P	***Q***	***P AND Q***	***NOT Q***	***P AND NOT Q***
0	*0*	***0***	*1*	***0***
0	*1*	***0***	*0*	***0***
1	*0*	***0***	*1*	***1***
0	*1*	***1***	*0*	***0***

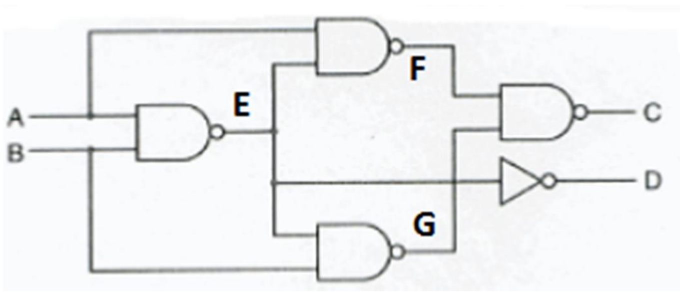
13. Aufgabe 5 AND aus NAND
Zeigen Sie, wie die AND-Funktion aus zwei NAND - Gates gebildet werden kann.



14. Aufgabe 6 De Morgan
Benutzen Sie die De Morganschen Gleichungen, um das Komplement von AB zu ermitteln.
$\bar{A} + \bar{B}$

15. Aufgabe 7 * Schaltungsanalyse
Was macht diese Schaltung?





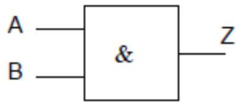
Wahrheitstabelle:

A	**B**	**D (A and B)**	E (not A&B)	F (not A&E)	G (not B&E)	**C (not F&G) = XOR A&B**
0	**0**	**0**	1	1	1	0
0	**1**	**0**	1	1	0	1
1	**0**	**0**	1	0	1	1
1	**1**	**1**	0	1	1	0

⇨ ***Halbaddierer***
⇨ ***C = Summe, D = Übertrag***

16. Aufgabe 8 Logik-Bausteine

(a) Ergänzen Sie die fehlenden Namen und die zugehörigen Wahrheitstabellen. Geben Sie ein Schalteräquivalent an.

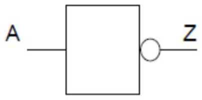


***AND**-Glied*

Wahrheitstabelle:

A	***B***	***Z***
0	***0***	*0*
0	***1***	*0*
1	***0***	*0*
1	***1***	*1*

Schalter-Modell:

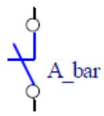


***NOT**-Glied*

Wahrheitstabelle

A	***Z***
0	*1*
1	*0*

Schalter-Modell:



(b) * Wie sieht ein Schalteräquivalent für eine EXOR Verknüpfung aus?