Marvin Glaser
4424114

Gruppe 9
Wilhelm Esser

Programmieren 1 (PRG1) Übung 3

3.1

a)

1234_{10}

Umrechnung in Binärzähl

Rechnung: 1234 – 1024 = 210; 210 – 128 = 82; 82 – 64 = 18; 18 – 16 = 2; 2 – 2 = 0

10011010010_2

Umrechnung in Binärzahl über Aufspaltung der Hexadezimalzahlen In 4 Binärzahlen

0100 1101 0010

4 D 2

$1234_{10} = 10011010010_2 = 4D2_{16}$

$CAFE_{16}$

Umrechnung in Binärzahl über Unterteilung der Hexadezimalzahl in 4 Binärzählen pro Zeichen

C A F E

1100 1010 1111 1110

1100101011111110_2

Umrechnung in Binärzahl:

2 + 4 + 8 + 16 + 32 + 64 + 128 + 512 + 2048 + 16384 + 32768 = 51966

$CAFE_{16} = 1100101011111110_2 = 51966_{10}$

b)

-128

Rechnung: 128 – 128 = 0

10000000_2

Negative Zahl (-128) durch invertieren der Binärzahl:

01111111_2

Marvin Glaser
4424114

Gruppe 9
Wilhelm Esser

-1

Rechnung: 1 – 1 = 0

00000001_2

Negative Zahl durch invertieren der Binärzahl:

11111110_2

127

Rechnung: 127 – 64 = 63; 63 – 32 = 31; 31 – 16 = 15; 15 – 8 = 7; 7 – 4 = 3; 3 – 2 = 1; 1 – 1 = 0

01111111_2

Keine Invertierung, da Zahl nicht negativ

c)

127 01111111_2

+ 1 00000001_2

= -127 10000000_2

Bei der Addition von Binärzahlen wird 1 + 1 umgerechnet zu 0 + Übertrag 1. Dieser wird in der nächsten Spalte mit eingerechnet. Das führt in diesem Beispiel dazu, dass alle Zahlen der Binärzahl invertiert werden. Da die erste Stelle des Einerkomplements ausdrückt, ob die Zahl positiv oder negativ ist, ist das Ergebnis von 127 + 1 im Zweierkomplement -127.

3.2

Binär im Zweierkomplement (16Bit)	Oktal	Dezimal
0110011010100110	063246	26278
1011111100000000	-40376	-16638
0111111000111011	100703	33217
0111011111110111	4007	2055
0111111111010100	-52	- 42
0000000111110100	764	500

Marvin Glaser
4424114

Gruppe 9
Wilhelm Esser

Rechnung, Zeile 1:

Binär unterteilt in 3 Binärzahlen pro Zeichen für Umwandlung in Oktal

(00)0	110	011	010	100	110
0	6	3	2	4	6

Umwandlung in Dezimalzahl analog zu Rechnungen in 3.1

Rechnung, Zeile 2:

Umwandlung von 2er Komplement in 1er Komplement, dann Invertierung

2er: 1011111100000000

1er: 1011111100000001

Inv: 0100000011111110

Umwandlung von Inv in Oktal analog zu Zeile 1, aber Zahl negativ, da 1 erste Zahl von Binärzahl

Umwandlung von Inv zu Dezimal analog zu Zeile 1, aber auch negativ

Rechnung, Zeile 3:

Umwandlung von Oktal zu Binär durch Aufspaltung jeder Zahl in Binärzahl

1	0	0	7	0	3
001	000	000	111	000	011

Umwandlung in 1er Komplement durch Invertierung und dann Rechnung + 1 (letzte zwei Nullen entfernt, damit 16Bit)

Normal: 1000000111000011

1er: 0111111000111100

2er: 0111111000111011

Umwandlung von Binär in Hexadezimal mittels „Normal“, analog zu Rechnungen in 3.1

Rechnung, Zeile 4:

Analog zu Zeile 3

Normal: 100000000111

1er: 011111111000

2er: (0111)011111110111

Marvin Glaser
4424114

Gruppe 9
Wilhelm Esser

Rechnung Binär (Normal) in Dezimal analog zu 3.1

Rechnung, Zeile 5:

Umrechnung von Dezimalzahl in Binärzahl analog zu 3.1

Dezimal 42 zu Binär: 101010

Umwandlung in 1er und 2er Komplement wie in Zeile 3

Normal:

1er: 101010

2er: 010101

3.3 (0111111111)010100

Umwandlung Dezimal (Normal) in Oktal, analog zu Zeile 1

Rechnung, Zeile 6:

Umrechnung Dezimal in Binär, analog zu 3.1

Binär: (0000000)111110100

Umwandlung von Binär zu Oktal analog zu Zeile 1

a)

Casting: *Casting* bezeichnet Allgemein, die Umwandlung (*type conversion*) von einem Variablentyp zu einem Andern. Beispiele wären die Umwandlung eines String Objekts in ein Integer oder umgekehrt. Hierbei ist zu beachten, dass mit c*asting* im Normalfall die implizite Konvertierung gemeint ist. D.h. eine „manuelle" Umwandlung von Variablen mithilfe von zusätzlichem Code (z.B. der Funktion float(1))

Coercion: Unter *Coercion* versteht man im Normalfall die sog. implizite Konvertierung von Objekttypen. D.h. die Konvertierung von einem Objekttyp in einen anderen wird während der Ausführung des Programms vom *Interpreter* oder *Compiler* durchgeführt. Ein Beispiel hierfür ist das addieren von einem Float mit einem Integer in Python.

(Quelle: https://en.wikipedia.org/wiki/Type_conversion; Datum: 08.11.17; Uhrzeit: 17:10 Uhr)

Marvin Glaser
4424114

Gruppe 9
Wilhelm Esser

b)

Ein typischer Fehler, der bei *casting* und *coercion* auftritt, ist die Umwandlung eines Objekttypen in einen ungültigen anderen Typen (z.B. einen String aus Buchstaben in einen Float; float(„Hallo“)). Ein weiteres Problem kann der Verlust von Information während der Konvertierung darstellen. Ein Beispiel in Python, wäre der Verlust der Nachkommastelle, bei der Umwandlung eines Floats in einen Integer (a = Int(1.5) -->a = 1).

(Quelle: https://en.wikipedia.org/wiki/Type_conversion; Datum: 08.11.17; Uhrzeit: 17:20 Uhr)

c)

Python 3.6.3 Shell	Casting	Coercion	Nichts davon
>>> 3 * 3.14		X	
>>> int(float(str(ord(chr(123)))))	X		
>>> int (1 - - 1)	X	X	
>>> 4 << 1			X