TEM11

Übungen

[Algorithmus 2](#_Toc407571629)

[Zahlendarstellung in ALU und FPU 3](#_Toc407571630)

[Gleitkomma- (Floats) und IEEE-Maschinenzahlen 6](#_Toc407571631)

[Tabellen- und Zeichencodes 7](#_Toc407571632)

[Codegewinnung 11](#_Toc407571633)

[Fehlertolerante Codierung 15](#_Toc407571634)

[Holzi 15](#_Toc407571635)

[Hardware 18](#_Toc407571636)

[Sekundärspeicher 19](#_Toc407571637)

[RAID 25](#_Toc407571638)

[Rekursionen 26](#_Toc407571639)

[Algorithmenkomplexität 28](#_Toc407571640)

[Binärbäume 31](#_Toc407571641)

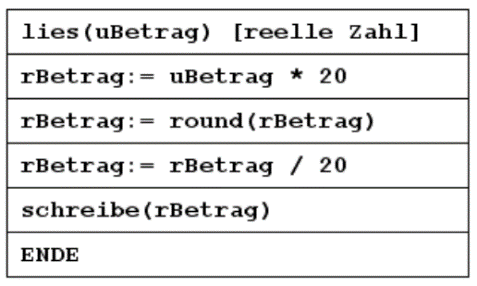
# Algorithmus

|  |  |  |
| --- | --- | --- |
| **Buch** | **Skript** | **Folien / Zusammenfassung** |
| 31 | 6 ff |  |

## Frage 1

Was "machen" die folgende Algorithmen?

Beispiel für eine reine Sequenz

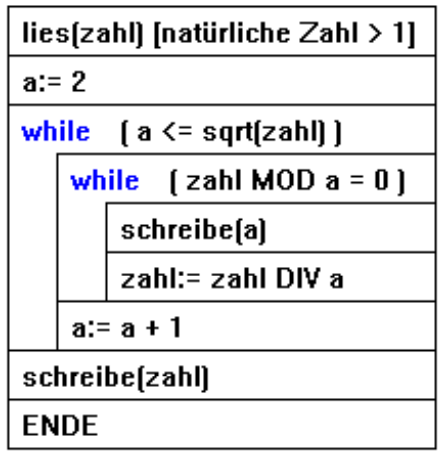
y:= round(x) ist eine Funktion zur mathematisch korrekten Rundung einer reellen Zahl x in eine Ganzzahl y

Machen Sie einen Versuch mit einer natürlichen Zahl und mehrere mit gebrochenen Zahlen.

* Rundung auf fünf Rappen.

## Frage 2

Beispiel für eine Iteration mit Kopfsteuerung

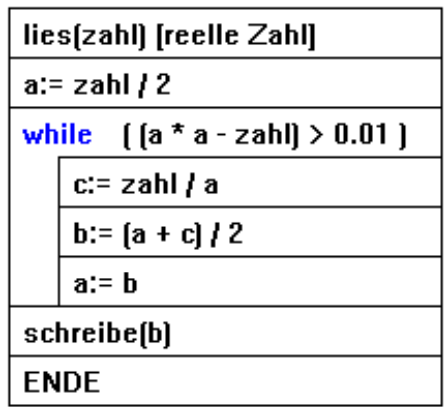
y:= sqrt(x) (square root) ermittelt die Quadratwurzel aus x.

Versuchen Sie durchaus auch grössere Werte, der Algorithmus konvergiert sehr schnell.

* Zerlegung einer Zahl in ihre Primfaktoren.

## Frage 3

Beispiel für eine Iteration mit Kopfsteuerung

Verwenden Sie Versuchszahlen bis 1000, um den Aufwand überschaubar zu halten.

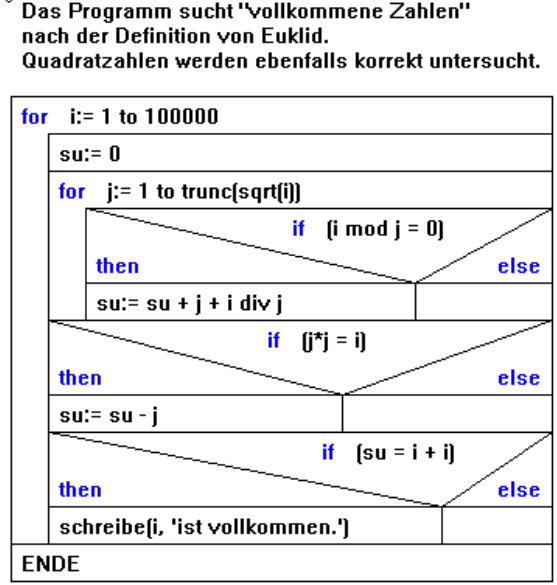
Auch dieser Algorithmus konvergiert schnell!

* Näherungsverfahren zur Berechnung der Quadratwurzel nach Heron von Alexandria.

## Frage 4

Der folgende Algorithmus ermittelt alle so genannt "vollkommenen Zahlen" (nach der Definition von Euklid) bis 100'000. Auch wenn Sie den Aufwand nicht auf sich nehmen können: Was ist nach Euklid eine vollkommene Zahl? Versuchen Sie nachzuvollziehen, was für Untersuchungen der Algorithmus an einer beliebigen natürlichen Zahlen i zwischen 1 und 100'000 vornimmt.

Beispiel für eine Iteration in Form einer Zählschleife sowie für eine Selektion

Sequenz, Selektion, Iteration bilden die drei grundlegenden Kontrollstrukturen der Strukturierten Programmierung. Strukturierte Programmierung bildet die Basis jeder Funktion bzw. Methode in der Objektorientierten Programmierung (OOP) und hat mit dem Siegeszug der OOP an Bedeutung eher gewonnen anstatt verloren!

* Wenn eine Zahl n gleich ist wie die Summe aller echten Teiler, dann nennen wir die Zahl n eine Euklidische vollkommene Zahl.

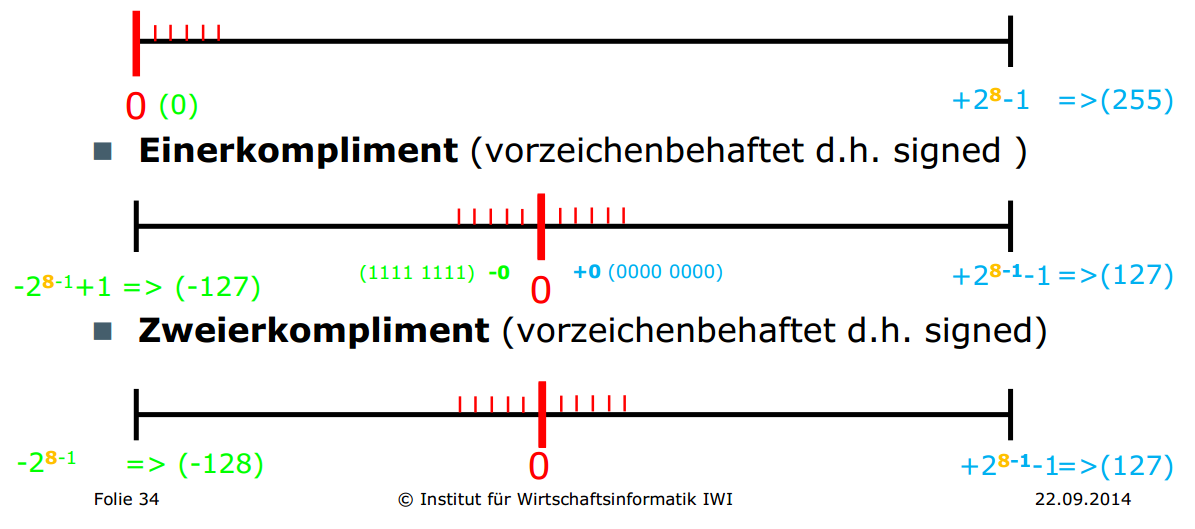
# Zahlendarstellung in ALU und FPU

|  |  |  |
| --- | --- | --- |
| **Buch** | **Skript** | **Folien / Zusammenfassung** |
| 47 ff | 11 ff |  |

## Frage 1

Welches geschlossene Intervall auf der Zahlengeraden decken folgende ganzzahligen Datentypen moderner Programmiersprachen ab - je vorzeichenlos und vorzeichenbehaftet:

* byte: 8 Bits
* byte (tinyint in SQL) meist 8-Bit unsigned, also 0 ... 255
* short: 16 Bits
* short (unsigned): 216 – 1 = 65’535; (signed): -216-1 – 1 = 32’768 …. 216-1 – 1 = 32‘767
* int: 32 Bits
* int je nach Programmiersprache unterschiedlich breit ist!!!
* long: 64 Bits
* long int in Java und C# 64 Bits ist.



## Frage 2

Gehen Sie davon aus, es gäbe einen Ganzzahl-Datentyp bubu zu 18 Bits.

Welches geschlossene Intervall würde signed bubu abdecken?

* [-131'072 … 0 … +131'071]

Welches geschlossene Intervall würde unsigned bubu abdecken?

* [0 … 262'143]

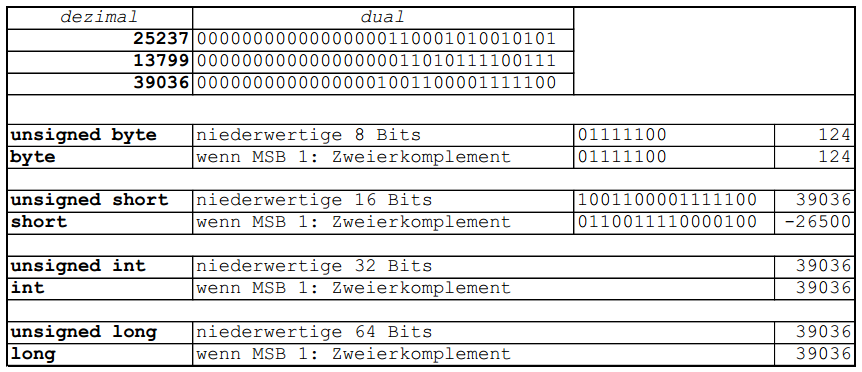
## Frage 3

"Stellen" Sie die natürlichen Zahlen 137 und -137 in folgende ALUs: 8 / 16 / 32 Bits, also die positive Zahl und ihr Zweierkomplement; beurteilen Sie die Einschränkungen und zu erwartenden Bildschirmausgaben.

* 8 Bits signed: 137 ist nicht darstellbar: 10001001 würde angezeigt als -119 (Vorlesungsbeispiel)
* 8 Bits unsigned: 137 ist 10001001 und wird korrekt angezeigt
* 16 Bits: 137 ist 00000000 10001001
* 16 Bits: -137 ist 11111111 01110111
* 32 Bits: 137 ist 00000000 00000000 00000000 10001001
* 32 Bits: -137 ist 11111111 11111111 11111111 01110111

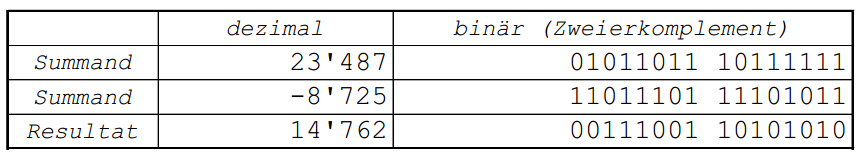
## Frage 4

(Zu den Wortbreiten siehe Übung 1.) Sie addieren in einem Ganzzahl-Rechenwerk die beiden Dezimalzahlen 25'237 und 13'799 - je als Long-Werte. Sie zwingen das Resultat danach in die acht Formate gemäss Tabelle (forced Type Casting). Welches dezimale Resultat erwarten Sie in den insgesamt acht Fällen auf dem Bildschirm? (Die Überlaufprüfung im Compiler sei deaktiviert.)



## Frage 5

Was wird im Folgenden berechnet? Basis sei eine 16-Bit-ALU. Füllen Sie die Tabelle vollständig aus:



# Gleitkomma- (Floats) und IEEE-Maschinenzahlen

|  |  |  |
| --- | --- | --- |
| **Buch** | **Skript** | **Folien / Zusammenfassung** |
| 68 ff | 14 ff |  |

## Frage 1

Welche Dezimalzahl entspricht folgender Maschinenzahl in einfacher Genauigkeit (die Leerstellen sind rein visuell)? 1100 0000 1110 0000 0000 0000 0000 00002

* -710

## Frage 2

Welche Maschinenzahl in einfacher Genauigkeit entspricht folgender Dezimalzahl? -0.62510

* 1011 1111 0010 0000 0000 0000 0000 00002

## Frage 3

Welchen Output vermuten Sie für das folgende Programm? Erklären und begründen Sie technisch einwandfrei.

using System;

class CalcTestMain

{

static void Main()

{

float x = 0.5f;

float y = 0.4f;

float r = x-y;

if (r == 0.1)

Console.WriteLine("Fortsetzung als Programmteil A, weil Ergebnis " + r);

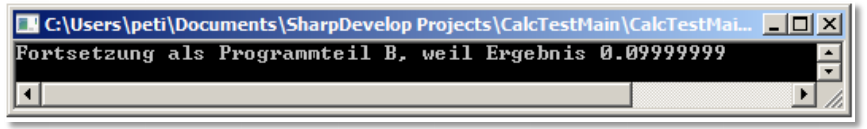
else

Console.WriteLine("Fortsetzung als Programmteil B, weil Ergebnis " + r);

}

}

* Die mathematische Logik erwartet Fortsetzung als Programmteil A. Die Realität ist:



# Tabellen- und Zeichencodes

|  |  |  |
| --- | --- | --- |
| **Buch** | **Skript** | **Folien** |
|  |  |  |

## Frage 1

Treffen Sie Aussagen bezüglich Ressourcenbedarf im Rechner: Speicherbedarf, Rechenbedarf beim Umformen zwischen Anzeige und Speicherablage.



* Speicherbedarf: UTF-8 am wenigsten, UTF-32 am meisten; konkret: 253 / 468 / 932 Bytes. (Die exakten Werte können aus der Aufgabenstellung nicht abgeleitet werden. Die Dateien liegen auf ILIAS.)
* Rechenbedarf: UTF-8 am meisten, UTF-32 am wenigsten.
* Hinweise zu den Browsern (Stand Frühjahr 2011, experimentieren Sie selbst!):
* IE 8 aktiviert bei allen Dateien ein ActiveX Script. Es werden zwei Zeichen nicht korrekt dargestellt. UTF-32 kann er nicht verarbeiten.
* Safari bringt für alle Dateien (richtigerweise) den ganzen Inhalt samt Umlauten usw.; er stellt indessen die Elementhierarchie nicht dar.
* Firefox kann alle Dateien laden. Alle Zeichen werden korrekt dargestellt.

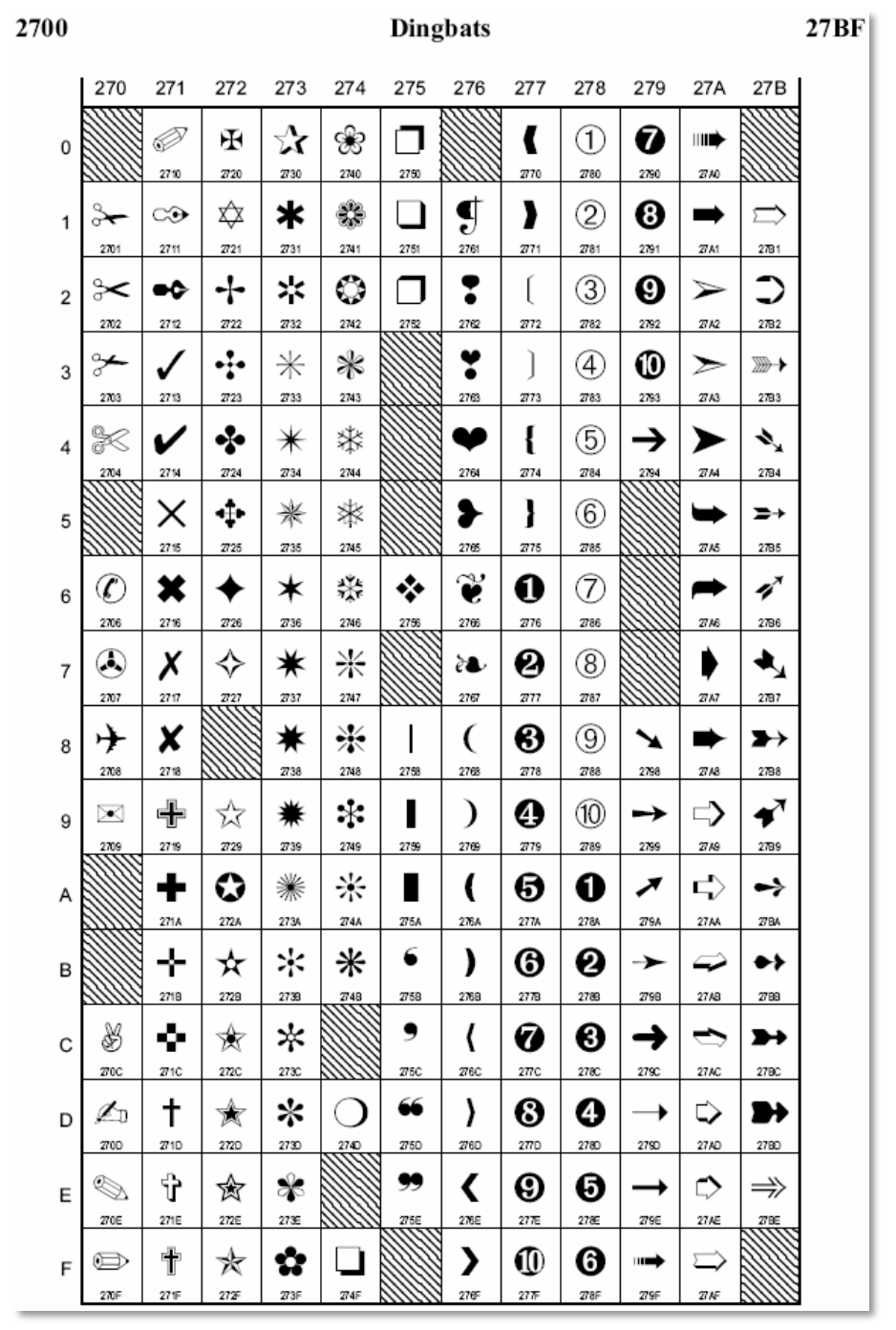
## Frage 2

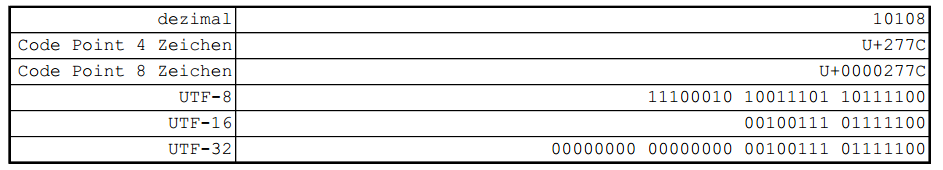
Der Cursor des XML Editors (siehe Abbildungen) ist ein Fähnchen mit nach rechts weisender Flagge. Was ist seine Bedeutung in unserem Dokument? Kann sich die Flagge wenden? Was hat das mit Unicode und den Code Points zu tun?

* Es hat arabische Zeichen, die von rechts nach links gelesen werden. Die Code Points von Unicode geben auch über die Leserichtung Auskunft. UTF übernimmt offensichtlich diese Hinweise.

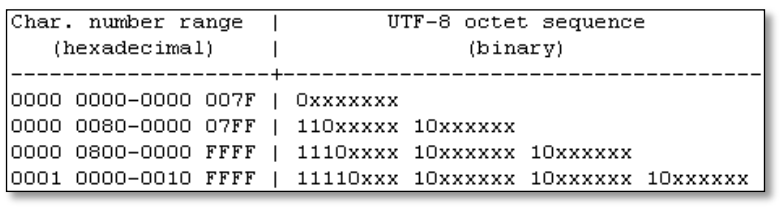
## Frage 3

Diese Aufgabe betrifft das Zeichen  aus obiger Dingbats Tabelle. Füllen Sie folgende Tabelle aus. Für UTF-8 genügen Ihnen obige Darlegungen, für UTF-16 benötigen Sie noch etwas Recherchen ... und  
UTF-32?

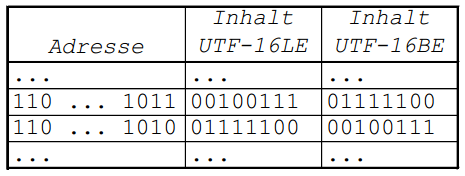




* Umrechnung in UTF-8:



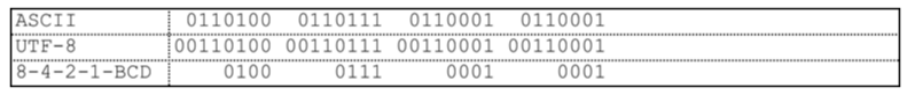
* Hinweis: Bei der Speicherung in der Byte-Maschine ergäbe sich am Beispiel UTF-16 folgende Stapelung (beachten Sie dazu die Thematik Little/Big Endian):



* In der aktuellen Skript-Version ist dies nicht Teil der Fragestellung.
* Die Adressen sind reine Fantasieadressen.

## Frage 4

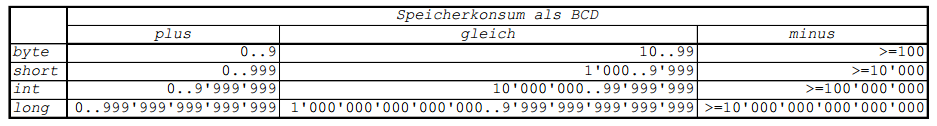
Stellen Sie die Zeichenkette "4711" (alphanumerisch!!!) dar in den in der Tabelle genannten Formaten. (Das LSB des "Einers" befindet sich ganz rechts.)



* Hinweis: Die Leerschläge sind rein visuell gemeint ... und zeigen gleichzeitig Gesetzmässigkeiten auf.

## Frage 5

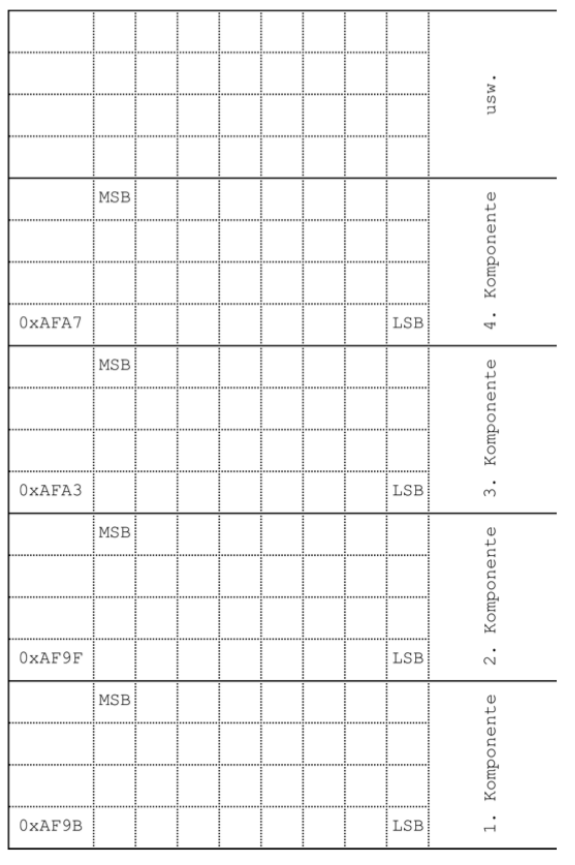
Die Umrechnung BCD <> Dezimalzahl ist mit Bestimmtheit rechenintensiver als die reine Dualzahl <> Dezimalzahl Umrechnung. Betrachten Sie im Folgenden aber nur die Speicherökonomie. Verglichen werden BCD Zahlen mit dem jeweiligen Ganzzahl-Datentypen: Bis zu welcher (natürlichen) Zahl "lohnt sich" BCD (Vermerk: plus), in welchem Bereich sind beide gleich (gleich) und ab welcher Zahl konsumieren wir mehr Speicher (minus)? Für den Ganzzahl-Datentyp byte ist die Aufgabe als Muster schon gelöst.



* Hinweis: "plus" meint: spricht für eine Speicherung als BCD und damit gegen eine Speicherung als Integer.

## Frage 6

Ein Array von short Integers ist im Speicher nicht gepackt abgelegt - also als LangwortIntegers. Startadresse ist 0x0000AF9B. (Langworte sind bei Intel im Format Little Endian abgelegt.) Berechnen sie die dezimale Adresse des LSB der 48. Array-Komponente!



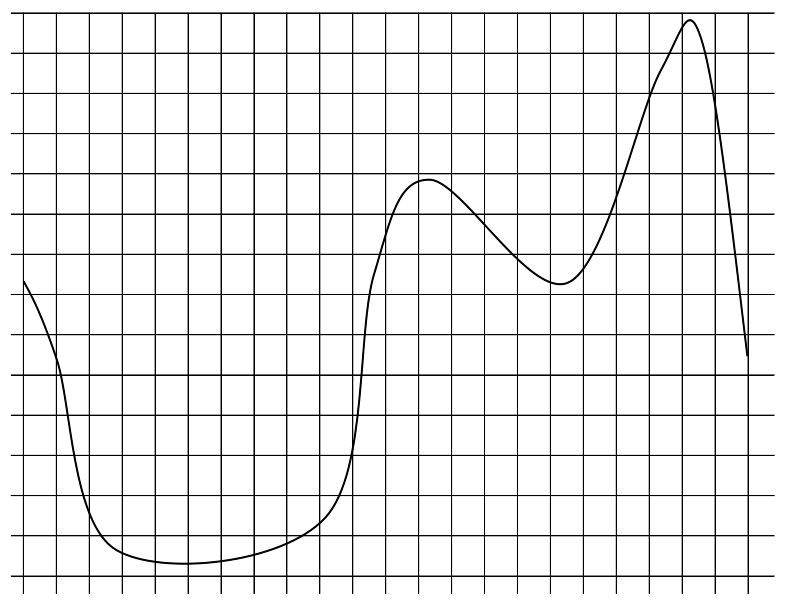
* Die 48. Komponente beginnt an Adresse: 0xAF9B + 0x2F \* 4 = 44'955 + 47 \* 4 = 0xB057 = 45'143
* Hinweise zu dieser Übung und allgemeine Bemerkungen:
* Als Speicherökonomie noch eine wirtschaftliche Notwendigkeit war, haben Systeme Daten gepackt (z.B. packed array), d.h. die führenden Nullen eliminiert.
* Beachten Sie die Unterschiede in den Bedeutungen von Lang/long - je nach Programmiersprache oder Digitaltechnik. Verbindlich sind die Angaben im Lexikon.
* Für eine sprachliche Benennung als 48. Array-Komponente gilt natürliche Zählung, also eine mit 1 beginnende.
* Als Speichermodell für die "Bytemaschine" verwende ich das "Hochhausmodell" mit von unten nach oben steigenden Adressen.
* Ein LSB hat in einer Bytemaschine nie eine eigene Adresse. Es hat die Adresse seines ganzen Bytes.

# Codegewinnung

|  |  |  |
| --- | --- | --- |
| **Buch** | **Skript** | **Folien** |
|  |  |  |

## Frage 1: Diskretisierung / Digitalisierung (analog-digital-Wandlung)

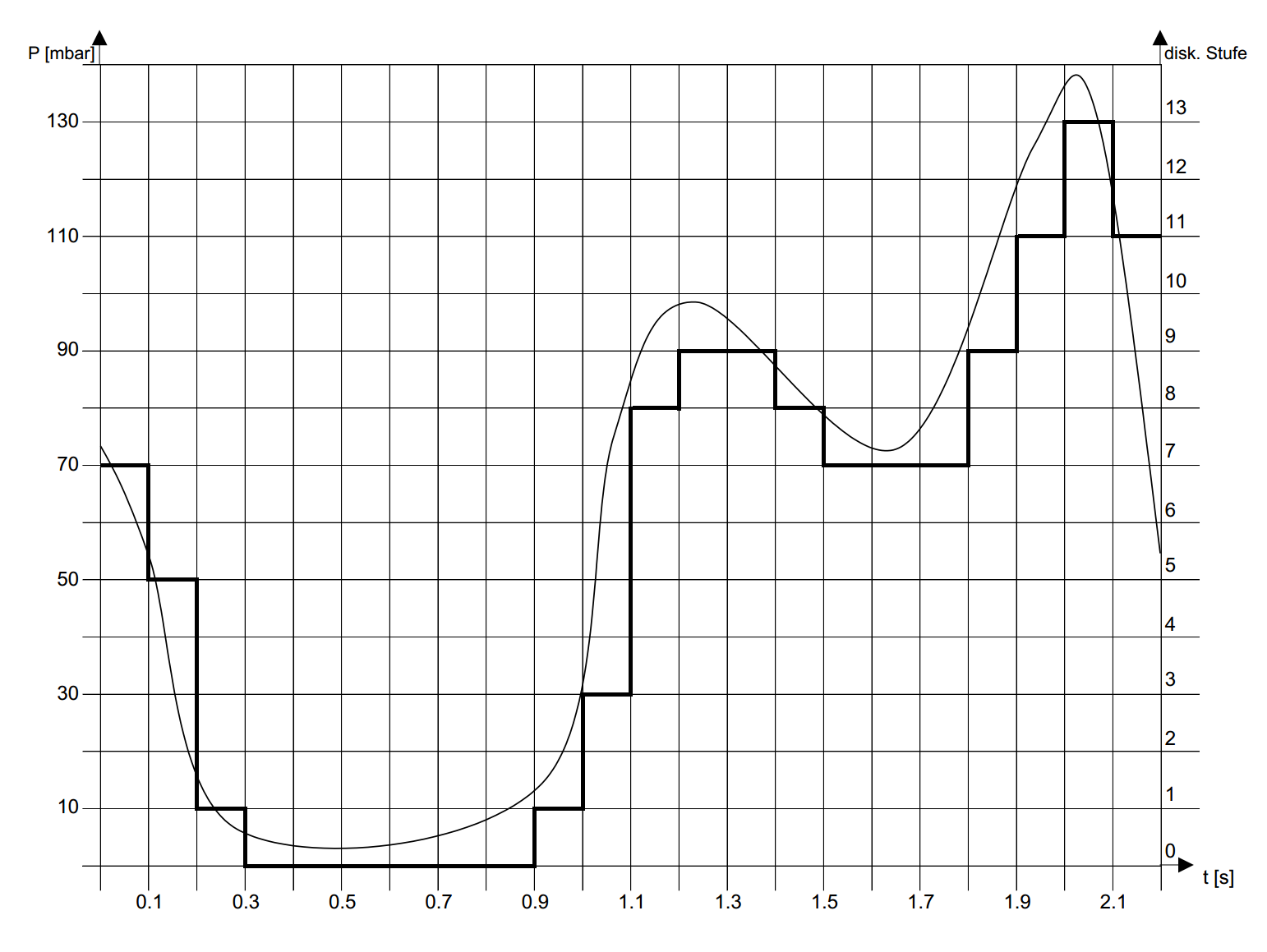
Digitalisieren Sie das kontinuierliche Signal wie folgt:



auf diskrete Werte abrunden

* die Abtastung beginnt zum Zeitpunkt 0
* das kontinuierliche Signal ist ein Schalldruck in mbar (0, 10, 20, ...)
* die Abtastfrequenz beträgt 10 Hz
* beschriften Sie die Tabelle möglichst weitgehend (Zeitachse, kont. sowie disk. Achse).

Wie viele Zustände kennt unsere diskrete Signalamplitude? Welche minimale Wortbreite in Bits hat ein Speicher für Ihre Signale theoretisch? Und welche dann praktisch?



* 14 (15 ?) Zustände: 4 Bits (konsequent aufrunden!!!); beim PC wären es 8 Bits, ein Byte
* Beachten Sie beim Zeichnen die absteigenden Flanken! Lesen Sie diesen Hinweis zweimal :-)
* Hz = Messung/Sekunde 🡪 Hier in diesem Beispiel heisst dies, dass jeder X-Achsenabschnitt 0.1 sec sind

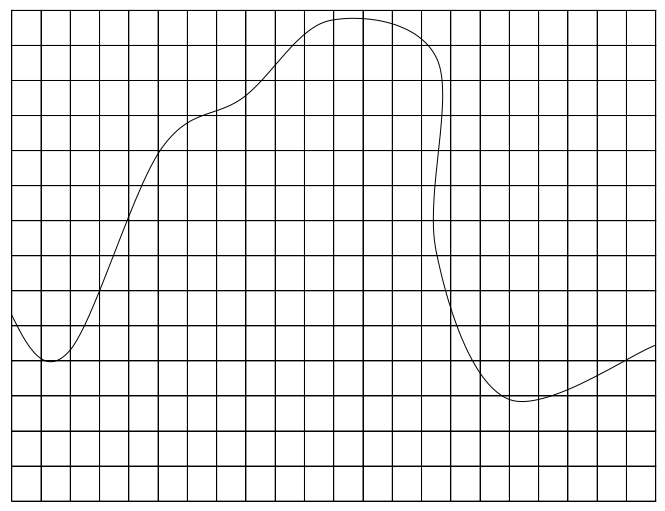
## Frage 2: Digital-analog-Wandlung

Ihr Dozent präsentiert Ihnen eine Serie binärer Werte oder eine diskrete Kurve mit der Aufforderung, das kontinuierliche Signal zu ermitteln, welches ursprünglich vorhanden war. Wie viele Lösungen sind möglich.

* Unendlich viele!

## Frage 3: Digital-analog-Wandlung

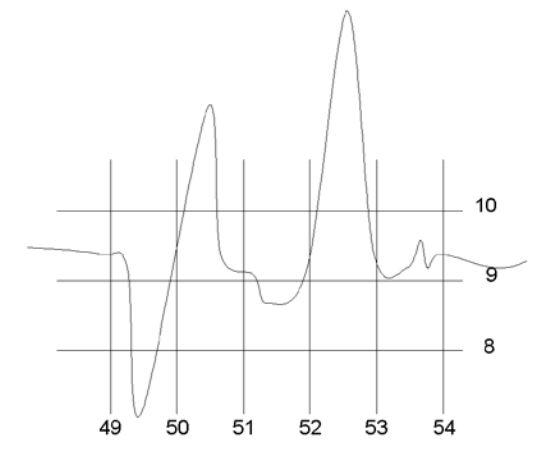
Student Halbgar bekommt genau eine solche Aufgabe. Am Ende präsentiert er folgende kontinuierliche Kurve. Wozu gratulieren Sie Halbgar voller Ehrfurcht?



* Bei Halbgar läuft die Zeit rückwärts!

## Frage 4

Folgende Illustration stellt einen völlig willkürlich gewählten Ausschnitt aus irgendeiner Diskretisierung dar. Erklären Sie einem Aussenstehenden anhand dieser Skizze in verständlichen Worten, weshalb die analog-digital-analog-Wandlung ausnahmslos mit Informationsverlusten verbunden ist!



* Die Diskretisierung dieser Kurve ergäbe eine Gerade! Alle Ausschläge zwischen den Abtastzeitpunkten werden ignoriert, gehen verloren und können nicht wieder gewonnen werden. Das Problem würde behoben mit einer höheren Auflösung: grössere Abtastfrequenz und/oder mehr diskrete Stufen. Der Preis für beide Massnahmen: mehr zu speichernde und transportierende Bits oder Bytes.

## Frage 5

Ein kontinuierliches Signal wird alle 125 µs (Mikrosekunden, Millionstelssekunden) über 200 Stufen abgetastet, digitalisiert und ohne Overhead übertragen. Reicht ein Nutzkanal des ISDN-Basisanschlusses mit 64 kbps? Welche Übertragungskapazität wäre nötig, wenn jedem Datenwort noch ein Startbit (Protokoll) und ein Paritätsbit (Fehlererkennung) beigefügt würden?

* f = 1 /T (f: Frequenz, T: Periodendauer), deshalb auch:

T = 1 / f

Bitstrom = 1 / (125 \* 10-6) \* (int(ld(200)) + 1) = 8'000 s-1 \* 8 Bits = 64'000 Bits / s == 64 kbps.

🡪Ja, es reicht exakt.

* 8'000 \* (8 + 2) = 80'000 bps == 80 kbps .

## Frage 6

Bei der Produktion einer Audio CD-ROM wird mit 44.1 kHz auf 65'536 Stufen und in zwei getrennten Kanälen abgetastet. Die klassische CD-ROM hat eine Kapazität von 650 MBytes. Wie viele Minuten (auf zwei Kommastellen!) ununterbrochene Musik haben Platz?

* Unter www.fischerpeter.info finden Sie ein Merkblatt zur Bedeutung der Vorsätze k/M/G/T bzw. K/M/G/T im Zusammenhang mit physikalischen Massen bzw. Speicherkapazitäten. Dies ist zu berücksichtigen.
* Die Berechnung ignoriert die zusätzlichen, redundanten Bits für die fehlertolerante Codierung. Deshalb wird die reale Musikdauer etwas geringer sein.
* t = 650 \* 220 \* 8 / (2 \* 44'100 \* ld(65'536)) = 3863 s = 1.07 h

## Frage 7

Mit einem Adresswort von 36 Bits sind 68'719'476'736 Speicherplätze adressierbar. Wie viele Bits mehr braucht es zur Verdoppelung der Adressen? 1 Bit / 2 Bits / 36 Bits?

* 1 Bit mehr.

## Frage 8

Hexadezimale Adressierung: An welcher Speicheradresse (binär) ist ein fataler Fehler passiert:

0a9cdd61 ?

* 0000 1010 1001 1100 1101 1101 0110 0001 (Die Lücken sind rein optischer Natur.)

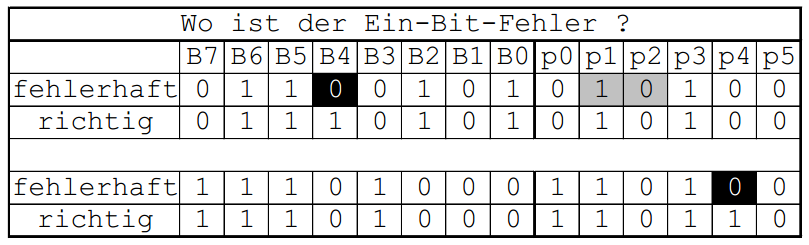
# Fehlertolerante Codierung

|  |  |  |
| --- | --- | --- |
| **Buch** | **Skript** | **Folien / Zusammenfassung** |
| 733 ff | 41 ff |  |

## Frage 1

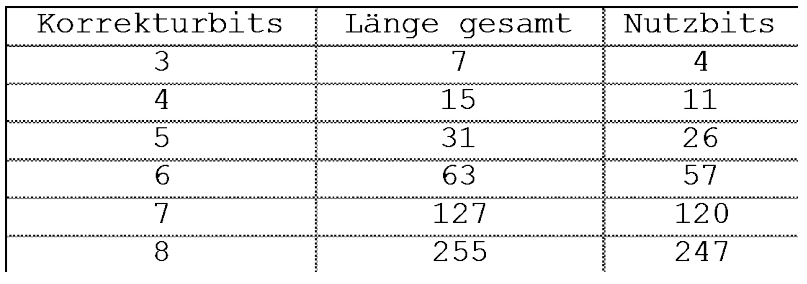
Finden und korrigieren Sie den 1-Bit-Fehler in folgenden zwei Codewörtern (es gilt der ECC

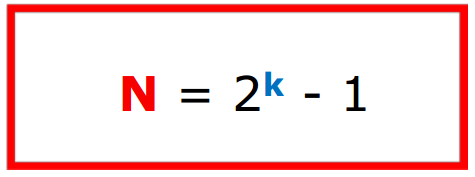
Ansatz aus der Unterricht):

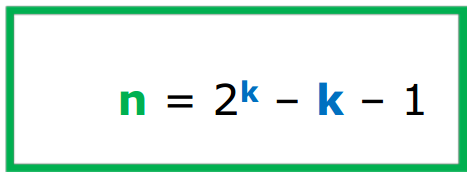


## Frage 2

Hamming Codes gibt es nicht in beliebiger Länge! Im Buch finden Sie eine Formel über real existierende Hamming Codes. Berechnen Sie einige dieser produktiven Hamming Codes und füllen Sie diese in folgende Tabelle:





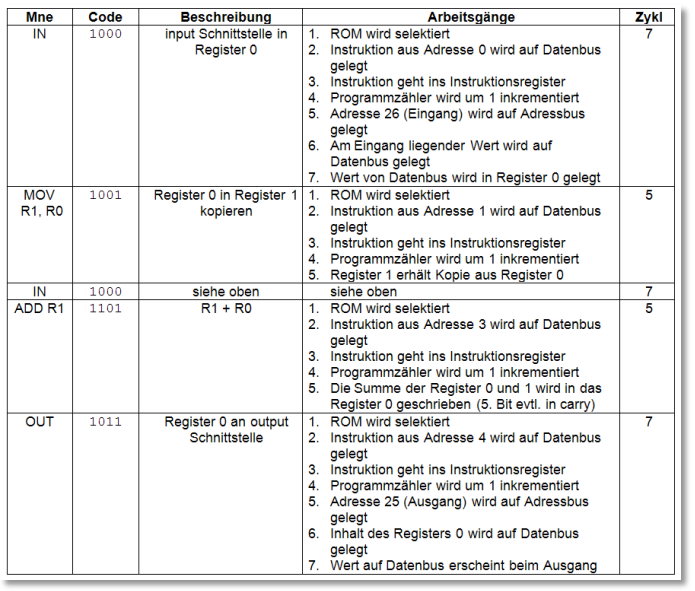


# Holzi

|  |  |  |
| --- | --- | --- |
| **Buch** | **Skript** | **Folien** |
|  |  |  |

## Frage 1:

Erstellen Sie ein Programm in Holzi-Assembler, welches am Eingang zwei Summanden abholt, sie addiert und das Resultat an den Ausgang legt. Geben Sie hier tabellarisch eine gründliche Lösung mit den Befehlen, Mnemonics, Anzahl Zyklen und Beschreibung deren Teile.



## Frage 2:

Nachstehend finden Sie ein Programm in Holzi-Maschinensprache. Während des mit (a) gekennzeichneten Befehls liegt an der Eingabeschnittstelle eine beliebige Zahl, z.B. 0011, an. Während des mit (b) gekennzeichneten Befehls liegt an der Eingabeschnittstelle eine beliebige Zahl, z.B. 1001, an. Das Programm terminiert nach dem letzten Befehl. Geben Sie an, was dieses Programm “macht”:

1000 (a) / 0000 / 0001 / 1000 (b) / 1001 / 0010 / 0001 / 1011 / 1010 / 1011.

* Das Programm liest zwei Werte von Input / Wert (a) in RAM-Zelle 17 / Wert (b) in R1 / Werte in gleicher Reihenfolge an Output.

## Frage 3:

Entwickeln Sie ein Holzi-Programm, welches den (ganzzahligen) Mittelwert M zweier einzugebender Zahlen a und b ermittelt: M = (a + b) div 2.

* 1000 / 1001 / 1000 / 1101 / 1111 / 1011
* Merken Sie sich die beiden Ganzzahl-Operationen DIV und MOD.

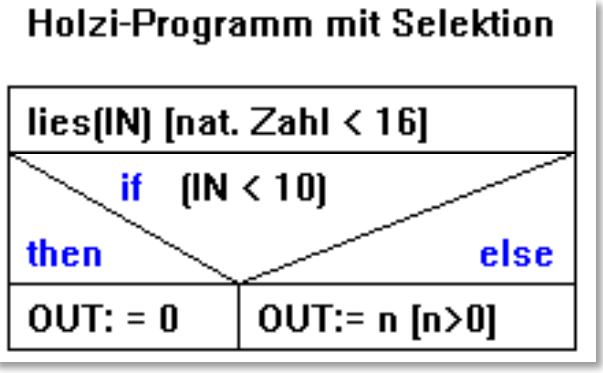
## Frage 4:

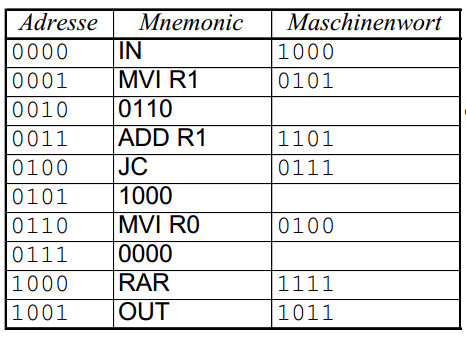
Erstellen Sie ein Programm in Holzi-Assembler, welches bei einer an Input liegenden Zahl untersucht, ob diese gerade sei oder ungerade. Ist die Zahl ungerade, muss an OUT und in RAM 16 der Wert 1111 aufleuchten, sonst je 0000.Das Programm muss im vorhandenen Programmspeicher Platz haben.

* 1000 / 1111 / 1100 / 0110 / 0101 / 1111 / 1010 / 1011.
* Die restlichen Befehle sind 0000 / 0000 - deshalb die Belegung von RAM 16 ...

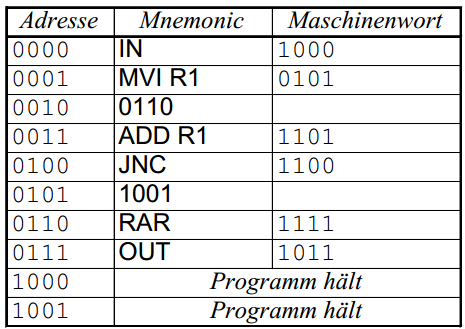
## Frage 5:

Programmieren Sie folgenden Algorithmus (tief durchatmen):





* Summand 6 bewirkt für jeden möglichen und gültigen Operanden ein Carry, weil die Summe >= 16 ist.
* RAR in Schritt 8 ist notwendig, "holt" das Carry "herein" und stellt so eine Zahl <> 0 dar.
* Im Fall JNC "saust" das Programm bei Schritt 6 weiter, doch das RAR "beschädigt" die dort eingefüllte 0000 nicht.
* Die freche Lösung geht von der frechen Annahme aus, dass das Programm bei Schritt 9 einfach hält und gar nichts an Output schreibt, sichtbar ist dies also eine 0000 !

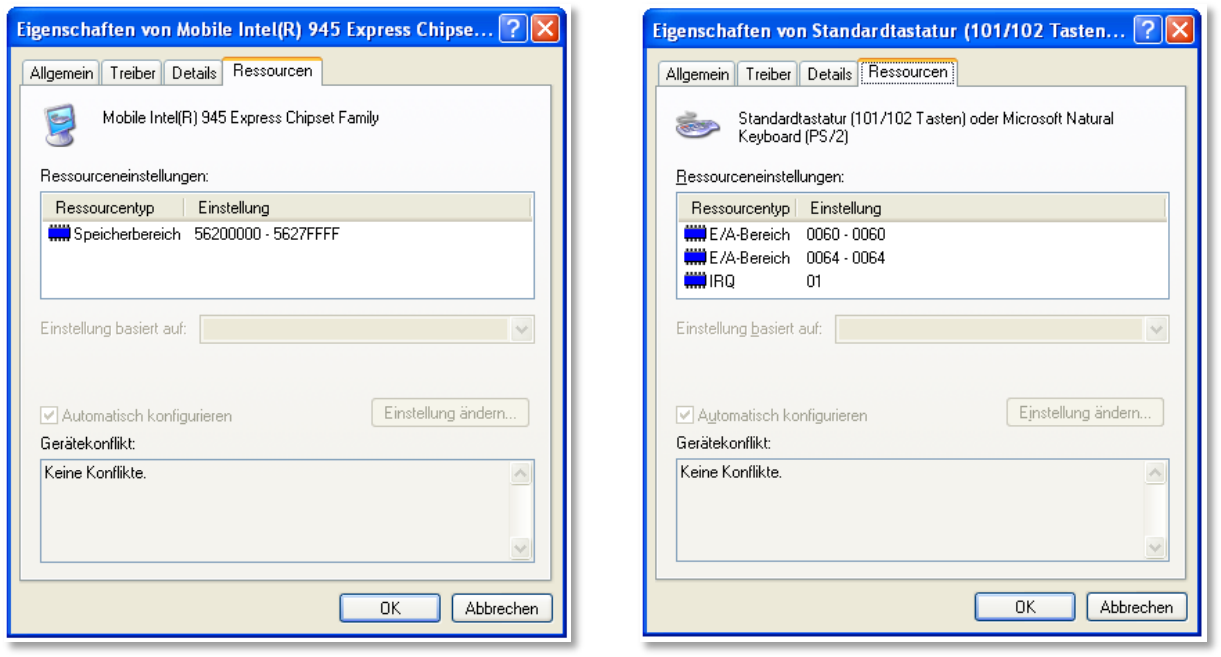


# Hardware

|  |  |  |
| --- | --- | --- |
| **Buch** | **Skript** | **Folien** |
|  |  |  |

## Frage 1

Der Gerätemanager zeigt für zwei Geräte folgende "Ressourcen"-Eigenschaften:



Ziehen Sie möglichst viele Schlüsse daraus:

* Wie umfangreich sind die Adressenräume?
* Welches Gerät hat speicherbezogene, welches isolierte Adressierung?
* Weshalb braucht eine reine Input-Peripherie Tastatur zwei Adressenräume?
* Weshalb braucht die Tastatur einen IRQ mit tiefer Nummer, also hoher Priorität?
* Die Grafikkarte verfügt über einen Adressenraum von 0x56200000 - 0x5637FFFF = 0x7FFFF entsprechend 524'287 Bytes!
* Das ist wohl der Grafikspeicher oder ein Teil davon, der ins RAM delegiert wird. Die Adressierung ist memory mapped - also Teil des globalen Adressenraums.
* Die Grafikkarte ist nicht "old fashioned" unterbrechungsgesteuert. (Die Steuerung ist direkt in den Controller verlagert.)
* Die Tastatur verfügt über zwei i/o-Adressen zu je einem Byte in einem separaten 16-Bit-Adressenraum. (Diese typische, hergebrachte, oft als legacy bezeichnete PC-Architektur heisst isolierte Adressierung - es hat(te) also ein separates "Speicherhochhaus" für Peripheriegeräte.
* Eine Adresse dürfte Input, die andere Output sein, denn die Tastatur empfängt auch Signale von der Zentraleinheit.
* Die Tastatur funktioniert unterbrechungsgesteuert.
* Die Anfragenummer ist 01 - dies bedeutet sehr hohe Priorität.
* (Anmerkung: noch höhere Priorität hat der Timer, der allerwichtigste Unterbrecher - siehe Lexikon.)

## Frage 2

Was ist ein NMI? Welche Gründe sprechen für solche, welche dagegen?

* Non Maskable Interrupt - nicht-maskierbare Unterbrechung; ohne "Maske" wird die Unterbrechung vorbehaltlos auf den Prozessor "losgelassen" und von ihm bearbeitet. Dies räumt ihm höchste Priorität ein. Die Massnahme ist geeignet für Unterbrechungen, die unmittelbar bearbeitet werden, die also einem "Echtzeit"-Kriterium genügen müssen. Andererseits können solche NMI andere, wichtige Unterbrechungen dadurch behindern. Eine schlüssige Empfehlung kann nur aus dem Zweck des Systems und der betreffenden NMI abgeleitet werden.

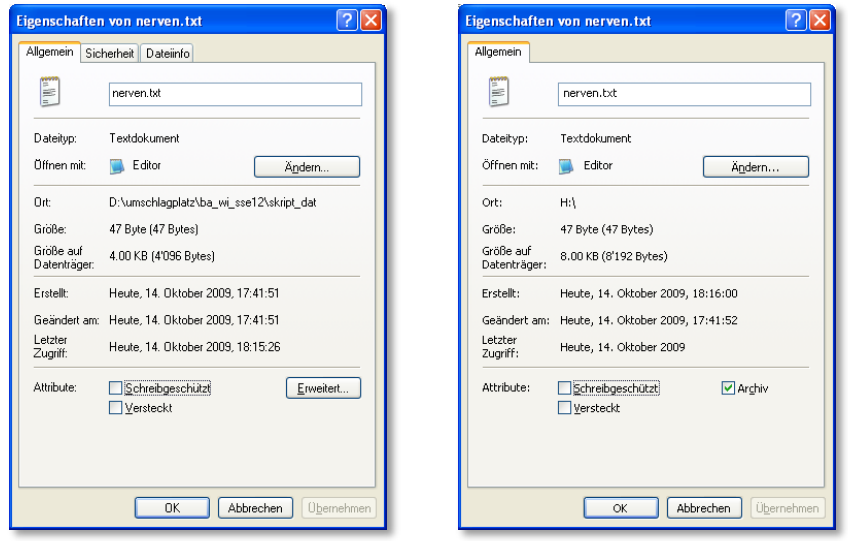
# Sekundärspeicher

|  |  |  |
| --- | --- | --- |
| **Buch** | **Skript** | **Folien** |
|  |  |  |

## Frage 1

Sie schreiben mit einem reinen ASCII-Editor, bzw. mit Word folgenden, dem harten Alltag abgerungenen Text: Computer können gelegentlich ganz schön nerven.

Der ASCII-Text umfasst (inkl. Leerschläge) 47 Bytes. Die Datei wird nun auf zwei Festplatten gespeichert. Sie wissen, dass ein Sektor auf Ihren Festplatten 4096 Bytes umfasst. Die Dateieigenschaften zeigen folgendes Bild:

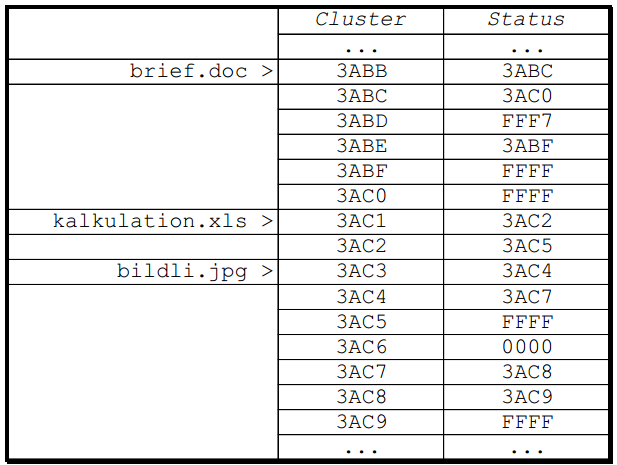


Beantworten Sie folgende Fragen (Platte links = Platte A):

* Wie kann eine Datei früher geändert werden als sie erstellt wurde (siehe B)?
* Weshalb wird für die ASCII-Mini-Datei in beiden Fällen so viel Speicherplatz "verzehrt"?
* Wie gross ist ein Cluster (C) auf Ihrer Platte A (in Bytes / in Vielfachen eines Sektors (S))?
* Wie gross ist ein C auf Ihrer Platte B (in Bytes / in Vielfachen eines S)?
* Welches minimale Einheit kann das Betriebssystem adressieren (C oder S)?
* Welche minimale Einheit kann die Hardware adressieren (C oder S)?
* Wie kommt es, dass 8 KB = 8192 Bytes sind (genaue Herleitung)?
* Weshalb belegt eine Word-Datei mit genau dem gleichen Text viel mehr Platz?
* Welches ist wohl die kleinere, welches die grössere Platte?
* Welche Platte ist wohl mit einem 16- (z.B. FAT16) und welche mit einem 32-Bit-Dateisystem (z.B. FAT32 oder NTFS) formatiert?
* Weshalb bietet mir das eine Fenster wohl mehr Optionen als das andere?
* Durch das Kopieren von einem Datenträger auf den andern. Wahrscheinlich wird das Erstellungsdatum durch das Dateisystem verwaltet und ist im Index eingetragen. Das Änderungsdatum wird wohl durch die Anwendung verwaltet und ist folglich in die Dateistruktur des Dokuments eingetragen
* weil ein ganzer Cluster gebraucht wird (interne Fragmentierung)
* 4096 Bytes, 1 Sektor
* 8192 Bytes, 2 Sektoren
* C
* S
* 8 \* 210 Bytes
* wegen der vielen Formatierungsangaben und Metadaten, die das Erzeugerprogramm dem Dokument mitgibt
* kann zuverlässig erst beantwortet werden, wenn ich das Dateisystem kenne, mit dem die Platte formatiert wurde: vorausgesetzt, es wurden beide Platten mit dem gleichen Dateisystem formatiert, dann ist Platte B grösser, denn das Dateisystem sah sich genötigt zu "clustern"
* wurden die Platten mit unterschiedlichen Dateisystemen formatiert, ist keine zuverlässige Aussage möglich
* es gibt zwei Hinweise darauf, dass Platte A mit NTFS formatiert wurde und Platte B mit FAT 16 (wie auf vielen Memory Sticks üblich)
* 1. Platte A wurde nur gerungfügig geclustert, also steht wohl ein 32-Bit-Adressenraum zur Verfügung
* 2. Platte A hat ein Register "Sicherheit", in welchem Zugriffsrechte verwaltet werden; dies kennen nur jüngere Dateisysteme - im Fall Windows ist dies NTFS
* siehe vorangehende Antwort

## Frage 2

Folgende Illustration lehnt an unseren Modellindex an. Darin gehören vier Cluster nicht zu den drei genannten Dateien: Deuten Sie in Worten diese vier Cluster:



Beantworten Sie die Fragen:

* Ist auch eine Rückwärtsverkettung möglich?
* Wie kommt ein Bad Cluster zustande?
* 3ABD: bad / 3ABE und 3ABF: gehören zu einer anderen, intakten oder "gelöschten" Datei (wenn gelöscht, sind davon evtl. nur noch Bruchstück-Ketten vorhanden, so genannte "lost clusters") / 3AC6: frei // ja, ist selbstverständlich möglich / durch Abnutzungseffekte (Gasströme zw. Oberfläche und Schreib-/Lesekopf)

## Frage 3

Ein Memory Stick wurde mit dem FAT Dateisystem (auch etwa FAT16 genannt) formatiert. Obige Datei würde "Grösse auf Datenträger 8.00 KB" angeben. Welche (adressierbare) Kapazität hat der Memory Stick im Maximum? Was brächte ein Wechsel zum System FAT32 oder NTFS: Clustergrösse, max. Anzahl Dateien, interne Fragmentierung, externe Fragmentierung usw.?

* Das Dateisystem hat "geclustert": 216 \* 8 \* 1024 Bytes = 536'870'912 Bytes / ein Wechsel brächte ein Schrumpfen der Clustergrösse auf die Sektorgrösse mit sich, da viel mehr Adressen zur Verfügung stehen: Anzahl Cluster = Anzahl Sektoren = max. Anzahl (kleine!) Dateien (ich müsste die Sektorgrösse kennen, um eine genaue Angabe machen zu können) / die interne Fragmentierung wäre entschärft, externe Fragmentierung tendenziell grösser, da grössere Dateien sich auf mehr Cluster verteilen.

## Frage 4

Zählen Sie mindestens drei Nachteile der Platten-Fragmentierung auf - Gesichtspunkte können sein: Mechanik, Performanz, Datensicherheit, ...

* Mehr mechanische Bewegungen beim Lesen und Schreiben / längere Transferzeiten (Latenzen wegen Mechanik) / stärkere Erwärmung / erhöhte Wahrscheinlichkeit des Datenverlustes durch Verkettungsfehler.

## Frage 5

Weshalb ordnen Defragmentier-Programme die Daten in den äussersten Zylindern an? Welche Nachteile brächte eine übertrieben häufige Defragmentierung mit sich (angenommen, die Platte würde dazwischen tatsächlich immer wieder ziemlich stark fragmentiert)?

* Erstens: Aus Sicherheitsgründen, weil die Daten aussen weniger dicht gepackt sind (heutige Platten gleichen dies - herstellerspezifisch aus; siehe Lexikon: Zone Bit Recording.) Zweitens: Um die mechanischen Armbewegungen gering zu halten: Sprünge von der Dateitabelle zu den Daten.
* Die Platte wird mechanisch nie so beansprucht, wie beim Defragmentieren selber. Die defragmentierten Dateien werden aussen abgelegt, wo sich die Platte deshalb mehr abnutzt. Gefahr des Datenverlusts beim Defragmentieren (deshalb zuerst: Backup). Arbeitsbehinderungen während der Defragmentierung.

## Frage 6

Angenommen, Sie hätten ausnahmslos Dateien mit einer kleineren Grösse als die Grösse der Clusters. Wie gross wäre die maximale interne Fragmentierung (verlorener Platz in % zum gesamten Plattenplatz)? Angenommen, Sie hätten ausnahmslos Dateien mit einer die Grösse der Clusters überschreitenden Grösse. Wie ist es dann?

* < 100% / < 50% (bei max. zwei Clusters)
* Bei kleinen Dateien ist die Fragmentierung also tendenziell gross bis sehr gross)

## Frage 7

Antworten Sie je mit "intern" oder "extern" - gemeint ist die Fragmentierung. Platz wird verschwendet / Zeit wird verschwendet / sieht das Betriebssystem gar nicht / gibt es nur im letzten Cluster einer Dateikette / nimmt mit dem Gebrauch der Platte zu / kann behoben werden.

* intern / extern / intern / intern / extern / extern

## Frage 8

Betrachten Sie unseren Modellindex und beantworten Sie folgende Fragen (insgesamt vier Antworten lauten eindeutig: "wissen wir nicht"):

* Wie viele Cluster kann das Dateisystem maximal verwalten?
* Wie viele Sektoren kann die Hardware maximal verwalten?
* An welcher dezimalen und binären Plattenadresse beginnt die Datei calc.xls ?
* An welcher dezimalen und binären RAM-Adresse beginnt die Datei calc.xls ?
* Eine Datei sei inhaltlich 854 Bytes gross. Wie viel Platz belegt sie auf dem Datenträger?

Der Datenträger sei ein Memory Stick mit 16 GBytes, die Sektorgrösse 4 KBytes.

* Wie viele Sektoren umfasst ein Cluster?
* Wie viele Bytes (auf ein Byte genau) umfasst ein Cluster?
* Eine Datei sei inhaltlich 854 Bytes gross. Wie viel Platz belegt sie auf dem Datenträger?
* Eine Datei sei inhaltlich 270'339 Bytes gross. Wie viel Platz belegt sie auf dem Datenträger?
* Ist diese Datei intern fragmentiert?
* Ist diese Datei extern fragmentiert?

Erster Teil:

* 216 = 65'536 Clusters (= max. Dateien)
* wissen wir nicht: Controller
* 15'039 / 00111010'10111111
* wissen wir nicht: abhängig von virtueller Speicherverwaltung, von RAM, MMU usw.
* wissen wir nicht, da Clustergrösse unbekannt

Zweiter Teil:

* Vorrechnung: Kapazität: 16 \* 230 = 17'179'869'184 Bytes

Clustergrösse in Bytes: 17'179'869'184 / 216 = 262'144 Bytes

Clustergrösse in Sektoren: 262'144 / 4096 = 64 Sektoren

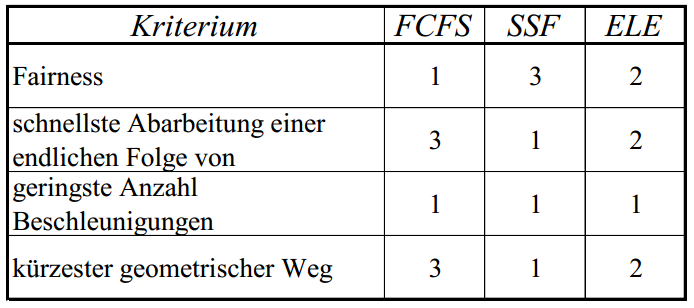
* 854 Byte-Datei belegt: 262'144
* 270'339 Byte-Datei belegt: (270'339 DIV 262'144 = 1 voller Cluster) + 1 intern fragmentierter Cluster = total 2 Cluster
* ja, siehe vorhergehende Antwort
* wissen wir nicht

## Frage 9

Die Industrie forscht nach wie vor an der Optimierung der mechanischen Bewegung von Plattenarmen (Arme mit den Schreib/Leseköpfen - siehe Abbildungen). Wir nehmen an, für den Plattenarm seien nacheinander Sprung-Anforderungen von ganz unterschiedlichen Spurnummern eingetroffen - so, als hätte bei einem Personenlift nacheinander auf mehreren Stockwerken jemand den Knopf gedrückt. Dann sind die möglichen Grund-Strategien, diese Spuren anzuspringen (Lexikon):

* first come - first served (FCFS)
* shortest seek first (SSF)
* Elevator.

Erstellen Sie für jedes Kriterium waagrecht eine Rangordnung 1 bis 3 (1 = am besten) für diese drei Grundstrategien. Eine Zeile lässt keine Rangierung zu.



# RAID

|  |  |  |
| --- | --- | --- |
| **Buch** | **Skript** | **Folien / Zusammenfassung** |
| 120 | 63 ff |  |

## Frage 1

Ein RAID 3 System bestehe aus acht Datenplatten (für die 8 Bits eines Bytes) und einer Paritätsplatte. Auf der Paritätsplatte werde pro Byte genau ein Bit für die gerade Parität (Parity even) gespeichert. Behauptung: Das genügt für die Fehlerkorrektur! Widerspruch zum Gelernten?

* Im Gegensatz zu Bit-Fehlern in der Elektronik "weiss" das System, welche Platte ausgefallen ist. Es kann so mit nur einem Bit Redundanz den Fehler beheben.

## Frage 2

Nehmen wir an, dass Plattenspeicher im Durchschnitt eine MTBF (Mean Time Between Failures) von 100'000 Stunden haben.

* Berechnen Sie die MTBF für ein gesamtes RAID 0 Plattenfeld aus 100 solcher Platten. (Beachten Sie nebenbei, dass bei RAID 0 der Defekt einer Platte immer auch zu einem Datenverlust führt, so dass die MTBF mit der mittleren Dauer bis zum Datenverlust (Mean Time Before / To Data Loss, MTDL) übereinstimmt.)
* Wie sieht das mit dieser MTDL bei den anderen RAID Levels aus?
* (Hoppla) Berechnen Sie die MTDL-Zeit für ein RAID 3- oder RAID 5-System bestehend aus 9 Platten (einschliesslich Paritätsplatte).
* Eine Platte läuft 100000 / 24 = 4167 Tage, bei 100 Platten ist alle 41.7 Tage eine Platte auszuwechseln.
* Der Ausfall einer Platte führt zu keinem Datenverlust.
* Lösung RAID 3 oder 5: 0 < RAID Level < 6 erträgt nur den Ausfall einer Platte. Beim Ausfall einer zweiten Platte gibt es Datenverlust. Es muss also sowohl eine Platte ausfallen, als auch während deren Wiederherstellungszeit eine der restlichen Platten! Die Wahrscheinlichkeit des Ausfalls einer Platte während einer begrenzten Zeitspanne von x Stunden ist x/100'000. Die kumulierte Wahrscheinlichkeit, dass sowohl eine erste Platte, als auch während deren Wiederherstellungszeit von x Stunden eine zweite der verbleibenden 8 Platten ausfällt ist:

P = x / 105 \* (8x / 105) = 8x2 / 1010 (Siehe Wahrscheinlichkeitsrechnung).

Beispiele:

Bei x = 20 h: P = 3.2 \* 10-7, daraus folgt in h: MTDL = 3'125'000 h = 357 a

Bei x = 0.5 h:P = 2.0 \* 10-10, daraus folgt in h: MTDL = 5 \* 109 h = 570'776 a

Beachten Sie:

* MTBF sind Wahrscheinlichkeiten, ein Ausfall kann jederzeit passieren.
* Die Ausfall-Wahrscheinlichkeit wächst mit der Gebrauchszeit der Platten.
* Die Berechnungen sind somit hypothetische Näherungen.

Keine Tränen der Selbstzweifel: Eine Berechnung in dieser Komplexität wird an der Modulprüfung nicht erwartet. Dies soll uns aber nicht daran hindern, ein solches Modell einmal durchzudenken. Oder?

# Rekursionen

|  |  |  |
| --- | --- | --- |
| **Buch** | **Skript** | **Folien / Zusammenfassung** |
| 199 ff | 75 ff |  |

## Frage 1

Ein Programm soll nun einen sortierten Vektor mit 1000 natürlichen Zahlen nach einer bestimmten Zahl binär-rekursiv so durchsuchen, wie Sie unter 1 eine Strategie entwickelt haben und die Position (Index) der Zahl ausgeben.

Versuchen Sie in einer Pseudonotation die rekursive Suchfunktion Suche(parameterliste) für das binäre Durchsuchen des Arrays anzugeben. Wenn Ihnen eine Pseudonotation schwer fällt, dann formulieren Sie verbal und prägnant, also algorithmisch, Ihr Vorgehen.

Die Funktion Suche(parameterliste) muss sich als Teil ihrer Definition enthalten, so etwa im Sinne:

Suche(zahl, li, re) =

wenn (li >= re) return

wenn (zahl ..........)

Suche(parameterliste1)

sonst

Suche(parameterliste2)

* Zu den Begriffen: Man unterscheide zwischen Suchen und Durchsuchen – siehe Lexikon! Das binäre Suchen läuft wie die "Telefonbuchsuche", die im Unterricht besprochen wird:
* stich in die Mitte des sortierten Vektors
* schau, in welcher Hälfte das gesuchte Element ist
* nimm diesen Halbvektor als neuen Vektor und stich ... (Tönt das nicht irgendwie rekursiv? Ja, es tönt – schauen Sie sich das Programm an: in der Methode Suche() wird die Methode Suche() aufgerufen!)
* Das Programm in C# realisiert das Anliegen mit 2000 sortierten Werten. Es zeigt die Position des gefundenen Elements im Vektor an:

using System;

class BinRekSucheMain

{

private static int[] vektor = new int[2000];

private static int Suche(int was, int li, int re)

{

if (li <= re)

{

int mi = (li + re) / 2;

if (was == vektor[mi]) return mi;

else

if (was < vektor[mi]) return Suche(was, li, mi - 1);

else

if (was > vektor[mi]) return Suche(was, mi + 1, re);

}

return -1;

}

public static void Main()

{

for (int i = 0; i < 1000; i++) vektor[i] = 5\*i;

Console.Write("Geben Sie Ihre natürliche Suchzahl < 5000 ein: ");

int schluessel = int.Parse(Console.ReadLine());

int position = Suche(schluessel, 0, vektor.Length - 1);

if (position >= 0)

Console.WriteLine("Das gesuchte Element steht an Position {0}", position);

else

Console.WriteLine("Element nicht gefunden");

Console.ReadKey();

}

}

* Im Zusammenhang mit den noch zu besprechenden Komplexitätsklassen, ist eine Analyse des binären Suchens sehr interessant; angenommen wir suchen ein Element unter 1 Mio sortierten Elementen, angenommen ferner, jeder Vergleich dauere 1 Sekunde:
* Lineare Suche (Klasse 0(n)) dauert im schlechtesten Fall 1 Mio Sekunden = 11.6 Tage.
* Binäre Suche (Klasse O(ld n)) dauert im schlechtesten Fall ceil(ld 1E6) = ceil(19.9) = 20 Sekunden.
* Den Logarithmus dualis von n berechnen Sie wie folgt: ln(n) / ln(2)

# Algorithmenkomplexität

|  |  |  |
| --- | --- | --- |
| **Buch** | **Skript** | **Folien** |
|  |  |  |

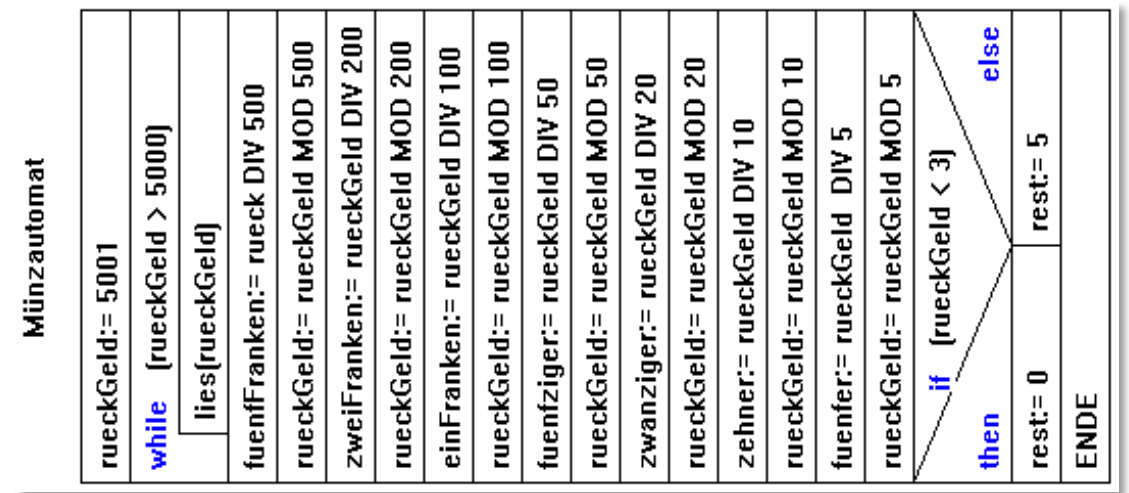
## Frage 1

Geldwechsel: Die minimale Parkgebühr in einem Parkhaus beträgt 1 Rappen, die maximale 50 Franken. Der Gebührenautomat akzeptiert max. 50 Franken in Münzen und Noten. Das Retourgeld erscheint in Münzen. Es gibt folgende Münzen: 5, 2, 1 Franken, 50, 20, 10 und 5 Rappen. Der Automat rundet mathematisch korrekt auf 5 Rappen und kennt dazu die Funktion long round(double).

Sie sehen im Folgenden das Nassi-Schneiderman Diagramm zum Münzwechsel-Algorithmus sowie den Programmcode in C#.

Fragen/Aufgaben

* Wozu dient die while Schleife am Anfang?
* Welche Komplexitätsklasse geben Sie dem Algorithmus?



using System;

class KleinGeldMain

{

static ushort rueckGeld = 5001;

public static void Main()

{

while (rueckGeld >5000)

{rueckGeld = short.Parse(Console.ReadLine());}

Console.WriteLine("Fünffränkler sind es: {0}", rueckGeld/500);

rueckGeld %= 500;

Console.WriteLine("Zweifränkler sind es: {0}", rueckGeld/200);

rueckGeld %= 200;

Console.WriteLine("Einfränkler sind es : {0}", rueckGeld/100);

rueckGeld %= 100;

Console.WriteLine("Fünfziger sind es: {0}", rueckGeld/50);

rueckGeld %= 50;

Console.WriteLine("Zwanziger sind es: {0}", rueckGeld/20);

rueckGeld %= 20;

Console.WriteLine("Zehner sind es : {0}", rueckGeld/10);

rueckGeld %= 10;

Console.WriteLine("Fünfer sind es : {0}", rueckGeld/5);

rueckGeld %= 5;

// wenn direkt mit Rappen gerechnet wird, erübrigt sich die Rundungsfunktion

// eine gängige Fünf-Rappen-Rundung ist: Round(frankenBetrag / 20.0) \* 20

Console.WriteLine("Und aus der Rundung gibt's noch: {0} Rappen.",(rueckGeld < 3) ? 0 : 5);

Console.ReadKey();

}

}

* Die Initialiserung auf 5001 sorgt dafür, dass der Algorithmus mit einem zu grossen Wert für das Rückgeld startet und deshalb in der while Schleife in die Abfrage geht. Diese Abfrage wird wiederholt, bis ein korrekter Startwert <= CHF 50 eingegeben wird, zu welchem das Wechselgeld berechnet werden soll.
* Der Algorithmus ist völlig klar von der Klasse O(n), da völlig unverschleift und linear. Die while Schleife am Anfang ändert als additive Konstante daran nichts, und die Selektion am Ende schon gar nichts, also quasi noch nichtser :-)

## Frage 2

Implementieren Sie folgenden Algorithmus rekursiv in einer realen oder Pseudosprache: Eine Folge von Buchstaben ist zu invertieren - aus SARG wird GRAS, aus KREIS wird SIERK und aus RELIEFPFEILER wird RELIEFPFLEILER (Wörter von der Art des letzteren nennt man Palindrome). Die zu verwendenden pseudosprachlichen Funktionen sind

* string invert(string) Ihre rekursive Funktion
* int length(string) Länge in Anzahl Zeichen
* string remove(string, int pos, int anzahl)
* append (string old, string appendix)

Fragen

* Welche (Zeit-) Komplexitätsklasse geben Sie dem Algorithmus für eine Zeichenkette der Länge n?
* Weshalb bedeutet in Rekursionen eine hohe Zeitkomplexität wohl auch hohe Speicherkomplexität?
* Was prognostizieren Sie einem rekursiven Algorithmus mit hoher polynomialer Komplexität für ein Laufzeitverhalten?
* (Pseudocode; der Algorithmus invertiert alle Zeichenketten, Palindrome sind nur ein Spezialfall:)

ausgabe:= ""

function invert(kette)

begin

if(length(kette) = 0)

then return(ausgabe)

else

begin

ausgabe:= append (ausgabe, kette[length(kette)])

// hinterster Buchstabe in natürlicher Zählung

invert(remove(kette, length(kette), 1))

// rekursiver Aufruf nach String-Kürzung

end

end

* Der Aufwand wächst linear mit der Länge der Strings. Der Algorithmus muss den String bis ans Ende zerlegen. Deshalb gilt O(n). Aber ...
* ... die internen Operationen sind sehr aufwändig: Stack-Verwaltung mit den hängenden Operationen (siehe Buch-Illustration zur Fakultät, die auch im Unterricht besprochen wurde). Der interne Speicherbedarf (Tiefenwachstum des Kellers) ist in jeder rekursiven Funktion sehr gross.
* Ein solcher Algorithmus wird sich mit der Zeit verlangsamen und er wird sehr schnell an die maximale Kellertiefe stossen und danach abstürzen.

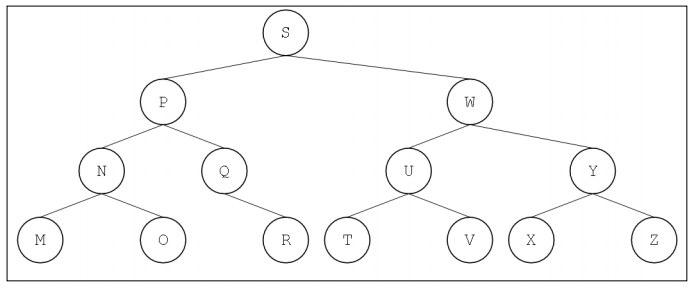
Rekursive Algorithmen sind "bequem" zu programmieren, zehren aber viele interne Ressourcen. Sie sind bei grossen n durch iterative Algorithmen zu ersetzen.

# Binärbäume

|  |  |  |
| --- | --- | --- |
| **Buch** | **Skript** | **Folien** |
|  | 96 |  |

## Frage 1

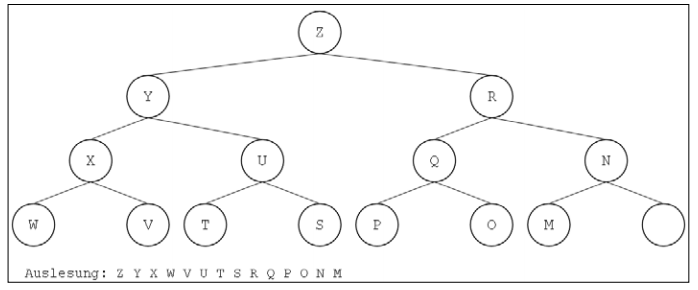
Erstellen Sie den möglichst balancierten binären Suchbaum für die Grossbuchstaben M ... Z.



## Frage 2

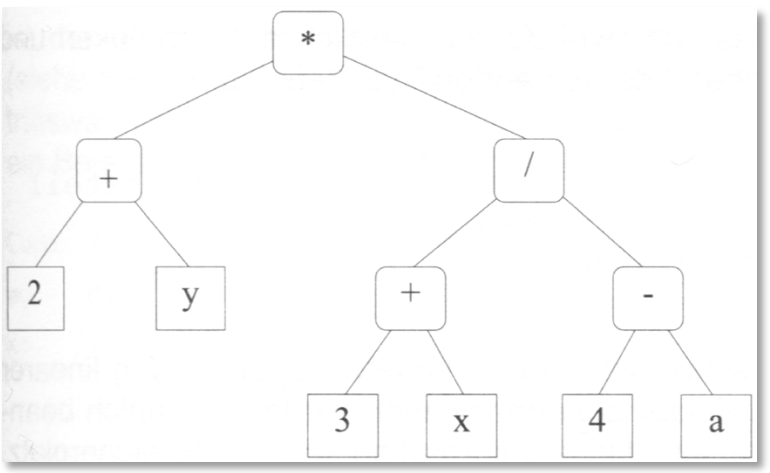
Errichten und belegen Sie den balancierten binären Baum, der preorder traversiert die Buchstaben M ... Z in absteigender Folge liefert.

* Dieser Binäre Baum ist kein Suchbaum - die Taversierung ist gemäss Aufgabe preorder.



## Frage 3

Folgender Baum liefert inorder traversiert die so genannte Infix-Notation eines algebraischen Ausdrucks (für Taschenrechner mit algebraischer Eingabe - ohne Klammern, diese stecken in der Semantik des Baums).



Wie ist das Resultat für

* x = -2
* y = 15
* a = -3.5
* (2 + y) \* (3 + x) / (4 - a) = (2 + 15) \* (3 + (-2)) / (4 - (-3.5)) = 2.2667

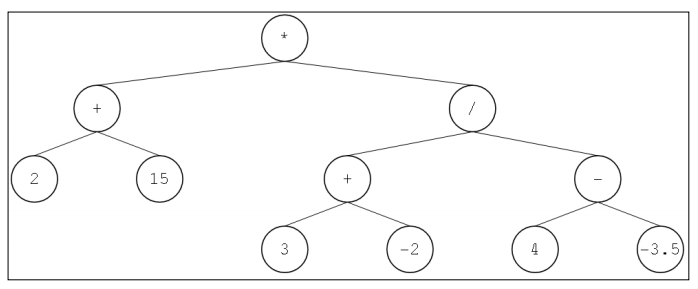
## Frage 4

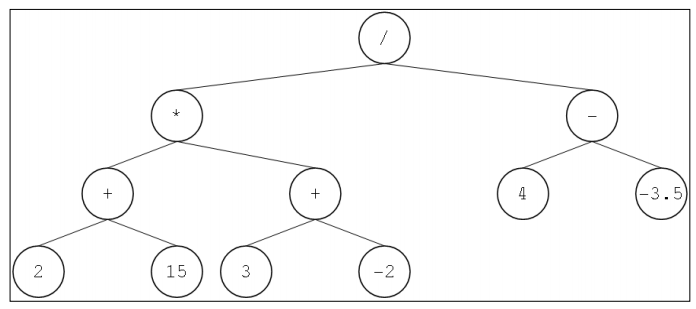
Formen Sie obigen Ausdruck um in einen Baum (es gibt zwei Varianten), der postorder traversiert die so genannte Postfix-Notation mit dem gleichen Ergebnis liefert (für Taschenrechner mit UPN-Eingabe).

* Variante nicht balanciert: bringt Tastenlogik besser zum Ausdruck
* Variante balanciert.

Merke:

* Parserbäume können inorder (Infix-Notation bei algebraischen Rechnern) oder postorder (Postfix-Notation bei UPN) ausgelesen werden.
* Klammersetzungen (Alg.) oder ENTER (UPN) ergeben sich aus den algebraischen Vorschriften, z.B. "Punkt vor Strich".
* In der Parserbaum-Traversierung kann nicht beliebig balanciert werden. Die Balance wird definitiv bestimmt durch die gegebene Reihenfolge der Operanden / Operatoren, also durch die Semantik des Baumes! Sie kann nur beschränkt "umgeknotet" werden, sofern sich dabei diese Logik nicht ändert. Der vorliegende Baum kann um die beiden gleichrangigen "Punktoperationen" Division und Multiplikation umgeknotet werden.
* Im vorliegenden Fall hat es 11 Knoten und alles binäre Operationen. Daraus folgt: Es ist kein vollständiger Binärbaum möglich, keine "totale Balance".
* Volle Binärbäume gelten als balanciert. Der vorliegende Parserbaum gilt somit als balanciert.

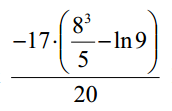


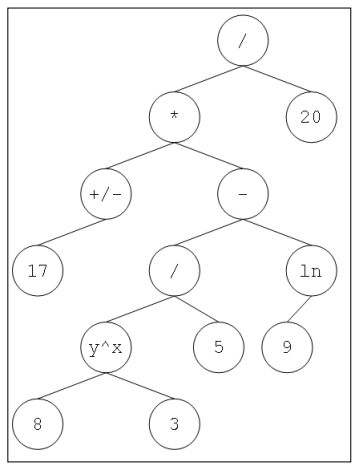


## Frage 5

Erstellen Sie den möglichst balancierten Parsing Baum, der postorder ausgelesen, die beim UPN-Beispiel (Theorie weiter oben) vermerkte Tastenfolge ergab. (Zahlen, wie 20, gelten als ein Knoten, die Taste [Enter] ist kein Knoten, die Operatoren sind sehr wohl Knoten.)

Merke:

* Der Ausdruck  liesse sich nur durch Kehrbrüche ändern, also z.B. durch eine Multiplikation mit 1/20 statt einer Division durch 20.
* Ansonsten ist keine weitere Balancierung möglich! Der Parserbaum gilt als für die Aufgabenstellung (ohne Änderung des Terms) optimal balanciert.
* Die unären Operationen +/- (Vorzeichenwechsel) und ln schränken die Äqivalenz von Inorder und Postorder etwas ein: ln muss hinter 9 gedrückt werden.



## Frage 6

Analysieren Sie die Dynamik im Keller eines UPN-Rechners und nennen Sie alle Tastendrücke, nach denen der Rechner automatisch ein push vollzieht.

Bezogen auf unser Beispiel:

* Nachfolgende Aufzählung gemäss Erscheinen im gesamten Term obiger Aufgabenstellung.
* Es sind nur die Tasten aufgeführt, deren Push-Wirkung beim Eingeben eines folgenden Operanden im Beispiel unmittelbar erkennbar ist.
* [ENTER]: der folgende Operand macht nicht eigentlich Push, er überschreibt das X-Register.
* sichtbar Push machen die binären Operatoren: [y^x] [/] [\*]
* Im Beispiel vorkommend, in der Push-Wirkung aber nicht direkt erkennbar (weil kein Operand folgt) sind: [-] [ln]
* Push-Wirkung haben generell:
* unäre Operatoren wie sqrt(), ln(), log(), 1/x, sin(), cos(), tan() usw.
* alle binären Operatoren
* weitere.