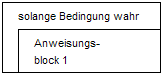
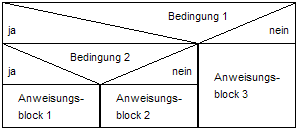
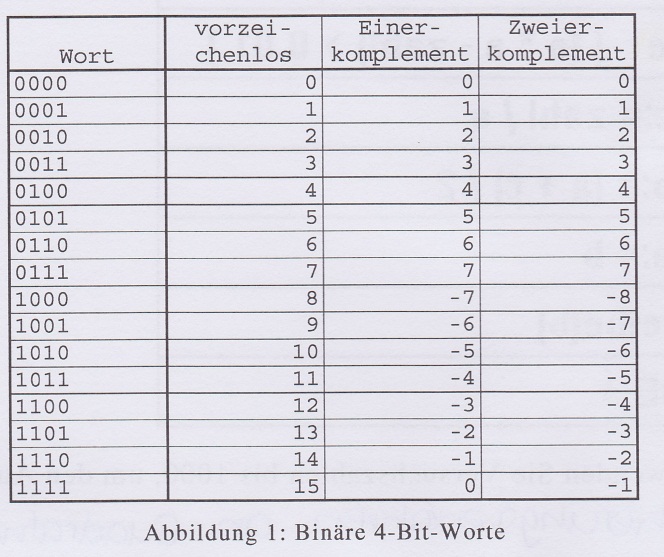
# Nassi Schneidermann



# Einerkomplement

**Einerkomplement bilden:**

Zahl10 negieren

****

**Umrechnen Zahl10 -> Zahl2**

*Feststellen ob Zahl10 negativ oder positiv*

*Wenn positiv:*

1) Zahl2 ganz normal umrechnen

*Wenn Negativ :*

1) Zahl2 ganz normal umrechnen

2) Zahl negieren!

3) Erhaltene Resultat hinschreiben

**Umrechnen Zahl2 -> Zahl10**

*Wenn MSB = 0*

1. Ganz normal umrechnen

*Wenn MSB = 1*

1. Zahl2 negieren
2. Umrechnen und Resultat hinschreiben mit dem negativem Vorzeichen **minus**

# Zweierkomplement

**Zweierkomplement: (10011010)2**

|  |  |  |
| --- | --- | --- |
| **signed** | Wenn MSB 1, dann ist die Zahl negativ   1. Negieren (10011010) 2. (01100101)2   Danach wird 1 dazu rechnen...   1. (01100101)2  (00000001)2  **(01100110)** | **-102** |
| **unsigned** | *ganz normal im TR ausrechnen --> (10011010)2* | **154** |

# Gleitpunkzahlen und das IEEE – 754 Format

So rechnet der Computer Gleitpunktzahlen (**floats**).

Beispiel Maschinenzahl in Dezimal umrechnen:

1100 0000 1110 0000 0000 0000 0000 00002

1. BIAS nachschauen gemäss Tabelle:

|  |  |  |
| --- | --- | --- |
|  | **FLOAT** | **DOUBLE** |
| Vorzeichen-Bits | 1 | 1 |
| Exponenten-Bits | 8 | 11 |
| Mantissen-Bits | 23 | 52 |
| Bits insgesamt | 32 | 64 |
| **BIAS** | **127** | **1023** |
| Exponentenbereich | [-127,127] | [-1022, 1023] |

1. Maschinenzahl Interpretieren

1100 0000 1110 0000 0000 0000 0000 00002

Negativ 129 – **127**

-1 \* 2 \* 1.75 = **-7**

Die 1.75 ergeben sich aus den restlichen Bits + 1. ½ ¼ 1/8 -> 0.75 + 1 = 1.75

1 1 0

*Beispiel Dezimalzahl in Maschinenzahl umrechnen:*

*1, da die Zahl geschoben wird bis zum letzten 1-er Bit wird die Zahl vergessen und muss sie jetzt zusätzlich addieren*

*-0.62510*

1. Den Betrag der Zahl, d.h. 0.625, in Binär umrechnen:

0.625 \* 2 = 1.25 -> 1

0.25 \* 2 = 0.5 -> 0

0.5 \* 2 = 1 -> 1

0 \* 2 = 0 -> 0

1. Kommaverschieben (Dezimalpunkt hinter die erste signifikante Zahl):

0,10102 \* 20 ->1,0102 \* 2-1

1. BIAS + Exponent

127 + (-1) = 126

1. Die Zahl von Schritt 3 in Dual umrechnen

12610 -> 0111 11102

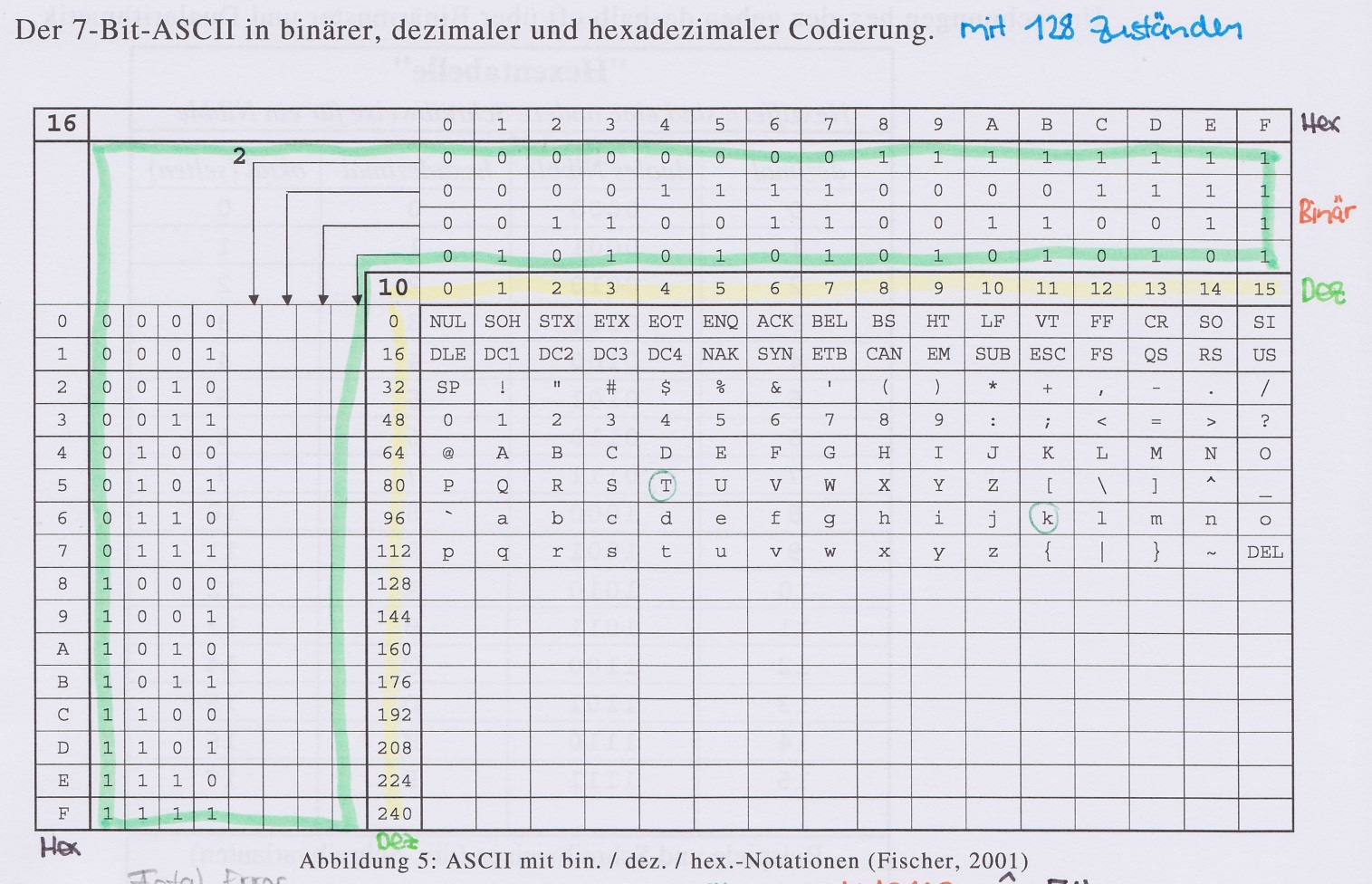
1. Nun die Maschinenzahl mit den errechneten Daten zusammenstellen:

* 1 am Anfang da Minus Zahl
* Anschliessend die 8 Bits von 126 aus Schritt 4
* Dann die Nachkommstellen aus Schritt 2

1011 1111 0010 0000 0000 0000 0000 0000

Negativ 126 Nachkommastellen von 0.625 aus Schritt 2

# ASCII und Unicode UTF



# UTF-Berchnung

**UTF-8 Berechnen**

|  |  |
| --- | --- |
| Char. number range  (hexadecimal) | UTF-8 octet sequence (binary) |
| 0000 0000-0000 007F  0000 0080-0000 07FF  0000 0800-0000 FFFF  0001 0000-0010 FFFF | 0xxxxxxx  **110**xxxxx **10**xxxxxx  **1110**xxxx **10**xxxxxx **10**xxxxxx  **11110**xxx **10**xxxxxx **10**xxxxxx **10**xxxxxx |

## Bsp. mit = ❼ = U+277C

1. Bereich bestimmen in welcher die Zahl passt (aus der obigen Tabelle entnehmen)
2. Vorgegebene Oktett Sequenz notieren  
   **1110**xxxx **10**xxxxxx **10**xxxxxx
3. Dezimale Zahl in Binäre umwandeln
4. Nun müssen die X des Vorgegebenen Oktetts durch die Binärzahl von hinten nach vorne eingeschoben werden. Falls Xe übrig blieben diese durch 0 ersetzen.  
    **1110**0010 **10**011101 **10**111100

**UTF-16 Berechnen**

Bsp. mit

1. Dezimale Zahl in Binäre umwandeln
2. UTF-16 verfügt über 16Bit. Daher muss nun lediglich von vorne aufgefüllt werden bis 16Bit vorhanden sind:  
   **00**100111 01111100

**UTF-32 Berechnen**

Bsp. mit

1. Dezimale Zahl in Binäre umwandeln

# BCD Binary Coded Decimal, BIT-Prüfung

|  |  |
| --- | --- |
| Dez | BCD |
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |
| Pseudotetraden | 1010 |
| 1011 |
| 1100 |
| 1101 |
| 1110 |
| 1111 |

|  |  |  |
| --- | --- | --- |
| Dezimal | Dual | BCD |
| 294 | 100100110 | 0010.1001.0100 2 9 4 |

# Diskretisierung / Digitalisierung

|  |  |
| --- | --- |
| **Bsp. einer Diskretisierung eines kontinuierlichen Signals** | **Mathematische Formeln** |
|  |  |

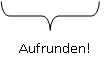
**Aufgabe 1**

*Ein Messsystem, z.B. ein Druckmessung, liefert unaufhörlich und regelmässig Messergebnisse. Diese Werden auf* ***25*** *Stufen abgetastet, sofort diskretisiert, digitalisiert und übertragen.*

* *Das digitale Wort hat die kleinste, theoretische notwendige Breite*
* *Jedem digitalen Wort wird noch* ***1*** *Paritätsbit beigefügt*
* *Zur Übermittlung des digitalen Signals steht Ihnen eine Leitung mit* ***2400*** *bps zur Verfügung*

*Mit welcher Abtastfrequenz in HZ kann höchstens gearbeitet werden?*

Aufrunden!



*Wie viele Signaldrähte hätte ein Parallelkabel, welches die digitalen Ergebnisse „abführt“?*

6 (besser wäre: 6+1, für Erde)

*Es sollte nun die Anzahl diskreter Stufen erhöht werden. Die Qualität der Signalleitung bleibt unverändert. Bis auf welche Anzahl diskreter Stufen kann ohne Folgen erhöht werden?*

*Unter den Bedingungen von oben müssen nun noch mehr diskrete Stufen realisiert werden. Mit welchen Massnahmen ist dies möglich? (zwei Möglichkeiten, eine davon riskanter)*

* Reduktion Abtastfrequenz
* Elimination Paritätsbit

**Aufgabe 2**

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



Aufrunden

***Wie viele Bits brauchen wir?***

**Aufgabe 3**

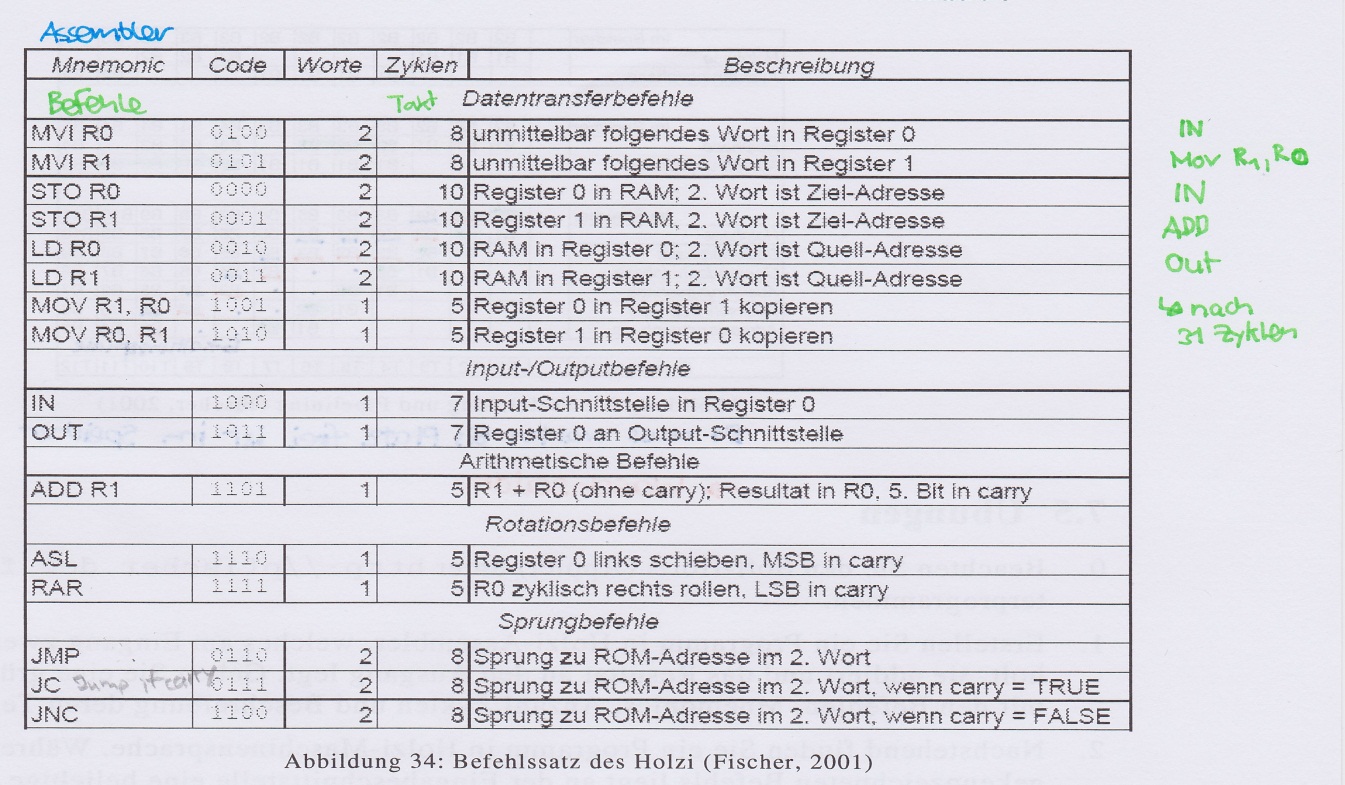
*Eine CD-ROM* ***650Mb*** *wird mit* ***44.1 kHz*** *auf* ***65‘356 Stufen*** *und in* ***zwei*** *getrennten Kanälen abgetastet. Wie viele Minuten haben Platz?*

Mbits

Anzahl Kanäle:  
Mono: 1  
Stereo: 2

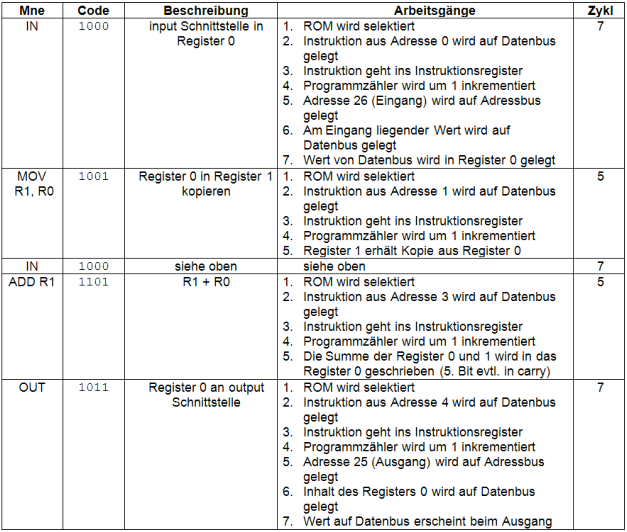
Bit Konsum

# Assembler



**Aufgabe 1**

*Erstellen Sie ein Programm in Holzi-Assembler, welches am Eingang zwei sSummanden abholt, sie addiert und das Resultat an den Ausgang legt.*



**Aufgabe 2**

*Entwickeln Sie ein Holziprogramm, welches den (ganzzahligen) Mittelwert M zweier einzugebender Zahlen a und b ermittelt: M = (a + b) div 2*

|  |  |  |
| --- | --- | --- |
| IN | 1000 | Inputschnittstelle in Register 0 |
| MOV R1,R0 | 1001 | Register 0 in Register 1 verschieben |
| IN | 1000 | Inputschnittstelle in Register 0 |
| ADD R1 | 1101 | R1 mit R0 addieren |
| RAR | 1111 | zyklisches Rechtsrollen |
| OUT | 1011 | Register 0 an output Schnittstelle versetzen |

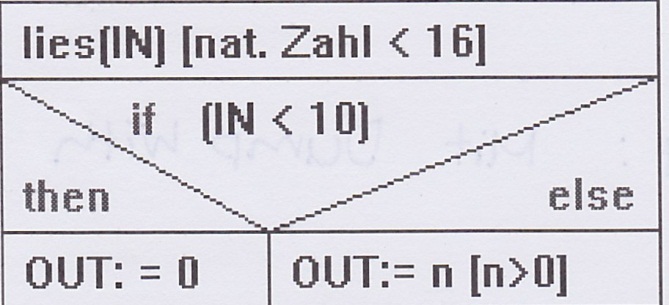
**Aufgabe 3**

*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*

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | IN | 1000 | Inputschnittstelle in Register 0 |
| 1 | RAR | 1111 | R0 zyklisch nach rechts rollen, LSB in Carry |
| 2 | JNC | 1100 | 1. Sprung zu ROM-Adresse im , wenn Carry = False = 0  2. Sonst Spring zur ROM-Adresse auf dem nächsten Befehl |
| 3 | ………………..gehört zu JNC………………. | 0110 | Binär: 6 |
| 4 | MVI R1 | 0101 | unmittelbar folgendes Wort in R1 |
| 5 | …………………gehört zu MVI R1…………. | 1111 | Binär: 16 |
| 6 | MOV R0, R1 | 1010 | Register 1 in Register 0 kopieren |
| 7 | OUT | 1011 | Register 0 an Outputschnittstelle |

**Aufgabe 4**

*Folgendes Programm im Holzi darstellen:*

****

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | IN | 1000 | Inputschnittstelle in Register 0 |
| 1 | MVI R1 | 0101 | unmittelbar folgendes Wort in R1 |
| 2 | 0110 | 0110 | Binär 6 |
| 3 | ADD R1 | 1101 | R1 mit R0 addieren |
| 4 | JC | 0111 | 1. Sprung zu ROM-Adresse im , wenn Carry = True = 1  2. Sonst Spring zur ROM-Adresse auf dem nächsten Befehl |
| 5 | ……….gehört zu JC… | 1000 | Binär: 8 |
| 6 | MVI R0 | 0100 | unmittelbar folgendes Wort in R0 |
| 7 | ……….gehört zu MVI R0……….. | 0000 | Binär: 0 |
| 8 | RAR | 1111 | R0 zyklisch nach rechts rollen, LSB in Carry |
|  | OUT | 1011 | Register 0 an Outputschnittstelle |

# Parity Prüfung

**Hamming-Code**

Hamming-Code = 1-fehler korrigierender Code mit einem Hammingabstand von *d* = 3.  
Der einfachste Hamming-Code ist ein (7,4)-Code der eine Läge von 7 Bits hat, wovon allerdings nur 4 Bits Nutzinformationen sind und die restlichen 3 Bits der Fehlerkorrektur dienen.

**Formel gesamte Länge:   
Formel Nutzinformationen:**r = Anz. Korrekturbits

Jedes Parity Bit überprüft eine bestimmte Stelle:

* Parity-Bit an Stelle 1, überprüft die Stellen 1,3,5,7 (Daher 🡪 Alle Dualzahlen die an letzter Stelle eine 1 besitzen: 001, 011, 101, 111)
* Parity-Bit an Stelle 2, überprüft die Stellen 2,3,6,7 (Daher 🡪 Alle Dualzahlen die an vorletzter Stelle eine 1 besitzen: 010, 011, 110, 111)
* Parity-Bit an Stelle 4, überprüft die Stellen 4,5,6,7 (Daher 🡪 Alle Dualzahlen die an drittletzter Stelle eine 1 besitzen: 100, 101, 110, 111)

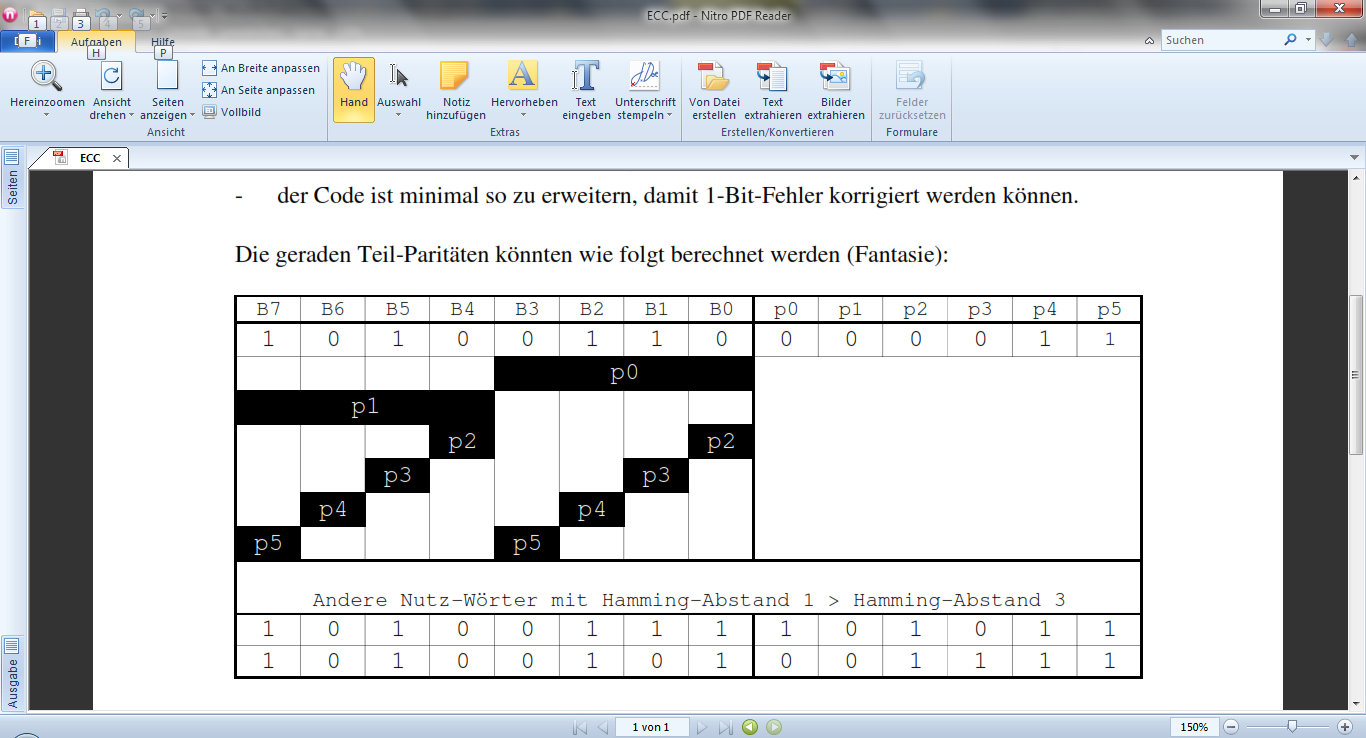
Tritt nun ein Fehler auf, ergibt die Binärfolge der Auswertung der Parity-Bits die Position an welcher der Fehler aufgetreten ist

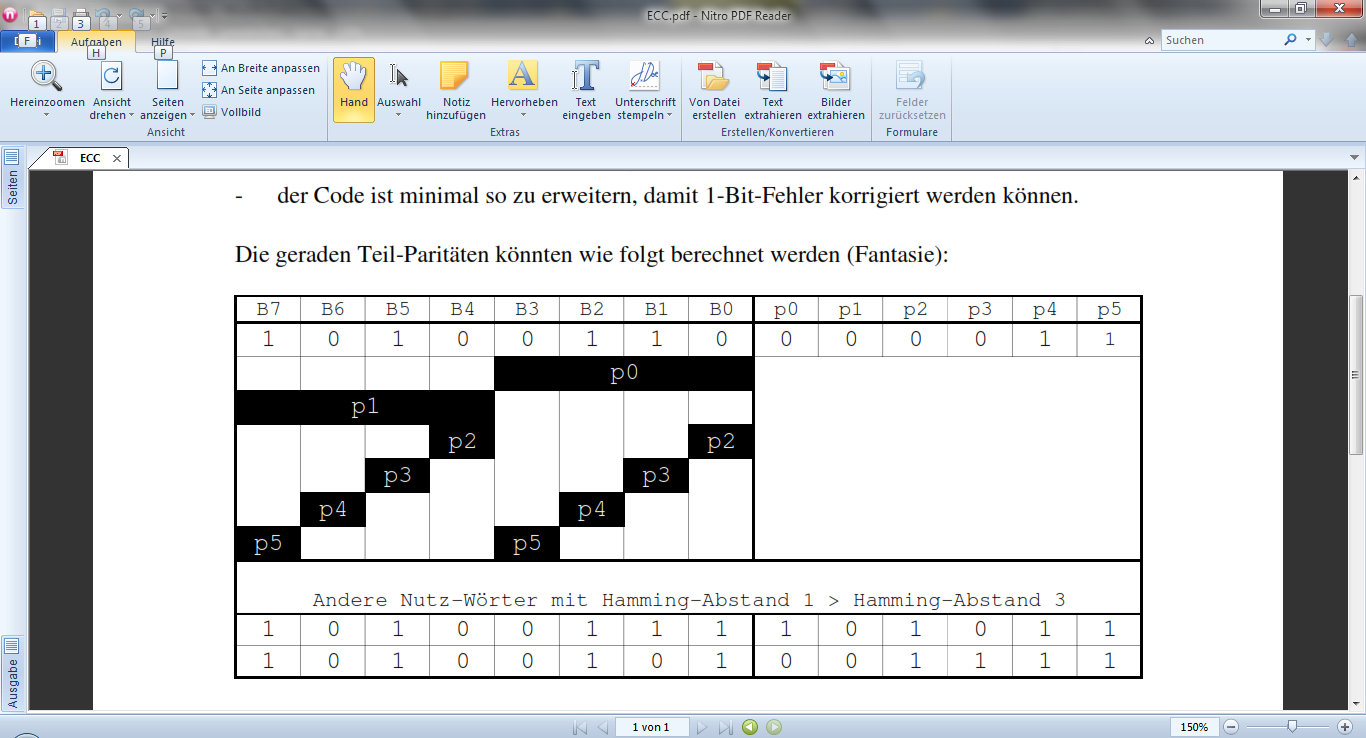
**Bsp.** Es soll die folgende Bitfolge übertragen werden 1100110. Bei der Übertragung gibt es einen Fehler dabei kippt das fünfte Bit 🡪 11**1**0110

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | Paritätsbit | wird gewertet als |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | - | 1 | - | 1 | - | 0 | falsch | 1 |
| 1 | 1 | - | - | 1 | 1 | - | richtig | 0 |
| 1 | 1 | 1 | 0 | - | - | - | falsch | 1 |

Schreibt man nun die Bitkombination aus der letzten Spalte von unten nach oben, erhält man die Dualzahl 101 = Dezimal 5. Und genau da im 5. Bit ist der Fehler aufgetreten.

**1 Bit Prüfung**





Für B4 wird Kontrolle gemacht mit:

# Datenträger

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Masseinheit für Bytes** | | | | |
| **Masseinheit** |  | **Anzahl von Bytes** | **KBytes** | **MBytes** |
| Byte |  | 1 |  |  |
| Kilobyte |  | 1024 | 1 |  |
| Megabyte |  | 1‘048‘576 | 1‘024 | 1 |
| Gigabyte |  | 1‘073‘741‘824 | 1‘048‘576 | 1‘024 |
| Terabyte |  | 1‘099‘511‘627‘776 | 1‘073‘741‘824 | 1‘048‘576 |
| Petabyte |  | 1‘125‘899‘906‘842‘624 | 1‘099‘511‘627‘776 | 1‘073‘741‘824 |
| Exabyte |  | 1‘152‘921‘504‘606‘846‘976 | 1‘125‘899‘906‘842‘624 | 1‘099‘511‘627‘776 |

MB ≠ MByte:

**intern fragmentiert** = im Cluster ist noch Platz vorhanden (verlorener Speicher)

**extern fragmentiert** = der Cluster ist zu klein. Es muss ein weiterer Cluster beansprucht werden. Dieser ist nicht zwingend nach der Reihe!

**Aufgabe 1**

*Ihr Memory Stick fasst* ***4******GBytes*** *und kann mit* ***FAT16*** *oder FAT32 formatiert werden. Berechnen Sie die minimale Clustergrösse in exakten Vielfachen von 1 Sektor, wenn die Formatierung die ganze Kapazität des Sticks verfügbar macht. Die Sektorengrösse ist* ***1 KByte****.*

***FAT16***

***FAT32***

1 Cluster muss maximal 1 Sektor gross sein also *1 Byte*

FAT 16 = 2hoch 16

**Aufgabe 2**

*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 FAT 32 oder NTFS: Clustergrössse, max. Anzahl Dateien interne Fragmentierung, externe Fragmentierung?*

* Ein Wechsel bräche eine 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 Sektorengrösse kennen, um eine genaue Angabe machen zu können)
* interne Fragmentierung entschärft,
* externe Fragmentierung tendenziell grösser, da grössere Dateien sich auf mehr Cluster verteilen.

**Aufgabe 3**

*Modellindex das FAT16 formatiert ist mit einer calc.xls-Datei deren erste Speicheradresse 3AC1 ist:*

*Wie viele Cluster kann das Dateisystem maximal verwalten?*

An welcher dezimalen und binären Plattenadresse beginnt die Datei calc.xls

hex = 3AC1 dez = 15‘039 binär = 00111010‘10111111

*Wie viele Sektoren kann die Hardware maximal verwalten? ---> wissen wir nicht*

*An welcher dezimalen und binären RAM-Adresse beginnt die Datei calc.xls* ----> wissen wir nicht

*Eine Datei sei inhaltlich 854 Bytes gross. Wie viel Platz belegt sie auf dem Datenträger*? ---> Clustergrösse nicht bekannt, wissen wir nicht

**Aufgabe 4**

*Der Datenträger sei ein Memory Stick mit* ***16 GBytes****, die Sektorengrösse* ***4 Kbytes,******FAT16***

*Wie viele Sektoren umfasst ein Cluster, Wieviele Bytes umfasst ein Cluster?*

Kapazität des Datenträgers:

Clustergrösse in Bytes:

Clustergrösse in Sektoren:

*Eine Datei sei inhaltlich 854 Bytes gross. Wie viel Platz belegt sie auf dem Datenträger?*

854 Byte-Datei belegt: 262'144 (ein Cluster)

*Deine Datei sei inhaltlich 270‘339 Bytes gross. Wie viel Platz belegt sie auf dem Datenträger?*

270'339 Byte-Datei belegt: (270'339 DIV 262'144 = 1 voller Cluster) + 1 intern

fragmentierter Cluster = total 2 Cluster

# RAID

|  |  |
| --- | --- |
|  | **RAID 0: Striping – Beschleunigung ohne Redundanz**  RAID 0 bietet gesteigerte Transferraten, indem die beteiligten Festplatten in zusammenhängende Blöcke gleicher Größe aufgeteilt werden, wobei diese Blöcke quasi im Reißverschlussverfahren zu einer großen Festplatte angeordnet werden. Somit können Zugriffe auf allen Platten parallel durchgeführt werden (engl. striping, was „in Streifen zerlegen“ bedeutet, abgeleitet von stripe, der „Streifen“). Die Daten können Parallel und so schneller abgerufen werden. Die Größe der Datenblöcke wird als **Striping-Granularität** (auch chunk size oder interlace size) bezeichnet.  **Vorteil**: **Nachteil:**  Bitet gesteigerte Transfarerate Wenn eine Platte Futsch-> beide Platten futsch |
|  | **RAID 1: Mirroring – Spiegelung**  RAID 1 ist der Verbund von mindestens zwei Festplatten. Ein RAID 1 speichert auf allen Festplatten die gleichen Daten (Spiegelung) und bietet somit volle Redundanz. Die Kapazität des Arrays ist hierbei höchstens so groß wie die kleinste beteiligte Festplatte  **Vorteil Nachteil:**  Einfachheit, da sie identisch sind Speicherkapazität  Jede Platte könnte in andere Systeme laufen  Totalsicherheit da vollkommene Redundanz  Spiegelung nicht gleich Datensicherung: Wegen Stromausfall, Viren, usw. |
|  | **RAID 3: Byte-Level Striping mit Paritätsinformationen auf separater Festplatte**  Der wesentliche Gedanke bei RAID 3 ist die Daten-Sicherung. Im RAID 3 werden die eigentlichen Nutzdaten normal auf einer oder mehreren Daten-Platten gespeichert. Außerdem wird eine Summen-Information auf einer zusätzlichen Paritäts-Platte gespeichert. Die Paritäts-Platte bewirkt die Datensicherung: Für die Paritäts-Platte werden die Bits der Datenplatten zusammengezählt und die errechnete Summe wird darauf untersucht, ob sie eine gerade oder eine ungerade Summe darstellt; eine gerade Summe wird auf der Paritäts-Platte mit dem Bit-Wert 0 gekennzeichnet; eine ungerade Summe wird mit dem Bit-Wert 1 gekennzeichnet. Die Datenplatten enthalten also normale Nutz-Daten, während die Paritäts-Platte nur die Summen-Informationen enthält. |
|  | **RAID 5: Leistung + Parität, Block-Level Striping mit verteilter Paritätsinformation**  Hier werden sowohl die Daten wie auch die Paritäten auf alle Disks verteilt; der Flaschenhals Paritätsplatte entfällt; beim Ausfall einer Platte kann diese, so hot pluggable während dem Serverbetrieb, ausgewechselt werden. |
|  | **RAID-01-Verbund**  Manchmal auch RAID 10 genannt, vereinigt Mirrorring und Striping. Zwei striped Sets werden gespiegelt wobei sich die hohe Sicherheit von RAID 1 mit der Performance von RAID 0 vereinigt.. |