A. Antworten zu den Aufgaben

Aufgabe 1 Was ist ein Computer?

Ein Computer ist ein universell programmierbarer Rechner, mit dem sich unterschiedlichste Aufgaben lösen lassen.

Aufgabe 2 Wie hieß der erste Computer?

Je nach Sichtweise wird der Zuse Z3 (1941) oder der ENIAC (1944) als erster Computer angesehen.

Aufgabe 3 Forschen Sie nach, warum der Z1 üblicherweise noch nicht als Computer gilt.

Der Z1 arbeitete mechanisch und hatte ein Problem mit der Zuverlässigkeit: Es verhakte sich konstruktionsbedingt die Mechanik, so dass er nicht wirklich funktionsfähig war.

Aufgabe 4 Vergleichen Sie genannten Beispiele nach Preis, Hauptspeicher- und Festplattenausstattung mit heutigen Angeboten! Um welchen Faktor unterscheiden sich die genannten Größen? Wie groß ist demnach der Verdoppelungszeitraum aus dem MOOREschen Gesetz?

Es ist schwer abzuschätzen, welche Parameter ein „typischer“ PC mitbringt: Zu groß ist die Bandbreite dessen, was aktuell angeboten wird und auch, was man zu einem bestimmten Preis bekommt. Daher kann es sich nachfolgend nur um grobe Schätzungen handeln, die stark variieren können. Ebenso sind Angaben von Nachkommastellen sehr differenziert zu betrachten.

Es gilt für den Faktor f, um den sich eine Größe im betrachteten Zeitraum verbessert hat:

Faktor f = Wert(2016)/Wert(1983)

Anzahl v der Verdopplungen im betrachteten Zeitraum:

v = ld f = ln f/ln 2, wobei ld der Logarithmus zur Basis 2 ist (Logarithmus Dualis).

Der Verdopplungszeitraum (Zeitraum, den *eine* Verdopplung benötigt) ergibt sich aus der gesamten Zeitspanne von T=33 Jahren (von Mitte 1983 bis Mitte 2016 gerechnet) und der Anzahl der Verdopplungen währenddessen:

t = T/|v|

Beim Preis ergibt sich keine Zunahme, sondern Abnahme. Das resultiert in einem negativen Vorzeichen bei v, das aber beim Berechnen des Verdopplungszeitraums ignoriert wird. Deswegen verwenden wir dafür den Betrag von v.

Bei typischen „Informatik"-Größen, z.B. Speichergrößen, nimmt man Zweierpotenzen anstelle der Zehnerpotenzen. So ist 1 kByte nicht 1000 Bytes, sondern 1024 Bytes. Das wurde bei den entsprechenden Zahlenwerten in der Tabelle berücksichtigt.

Wir führen noch einige zusätzliche Schätzungen durch. Betrachtet man die Rechenleistung R, dann kann man diese ungefähr als Produkt

R = Anzahl Prozessorkerne · Wortbreite · Taktfrequenz · Optimierungsfaktor

abschätzen. Bei dieser Schätzung ist berücksichtigt, dass die interne Architektur im Laufe der Jahre erheblich optimiert wurde, was die Rechenleistung zusätzlich anwachsen ließ. So benötigen z.B. manche Befehle, die früher mehrere Takte brauchten, nur noch einen Takt. Auch können heutige Prozessoren in jedem Prozessorkern mehrere Operationen gleichzeitig ausführen. Der Optimierungsfaktor wurde eher konservativ als 10 angenommen.

Tab. A.1: Zunahme der Leistungsfähigkeit von Computern

	1983	2016	Faktor f	v	t [Jahre] bzw. [Monate]
Anzahl Prozessorkerne	1	8	8	3	11 Jahre
Wortbreite	16 Bit	64 Bit	4	2	16 Jahre
Taktfrequenz	4,77 MHz	3,5 GHz	734	9,52	3,47 Jahre
Hauptspeicher	256 kB	8 GB	32.768	15	2,20 Jahre
Festplatte	10 MB	3 TB	349.525	18,4	1,79 Jahre
Preis	30.000.- DM	1.000.- Euro	1/15	-3,91	8,43 Jahre
Rechenleistung			96.000	16,55	1,99 Jahre
Rechenleistung/Preis			1.440.000	20,46	1,61 Jahre

Man erkennt, dass der Verdopplungszeitraum stark von der betrachteten Größe abhängt. In der Literatur findet man Zeiträume von 18 bis 24 Monaten für eine Verdopplung, also 1,5 bis 2 Jahre. Das kann durch unsere Schätzung für Festplatten und Rechenleistung sowie für Rechenleistung/Preis bestätigt werden.

Allerdings sollte man berücksichtigen, dass manche Größen derzeit an Grenzen stoßen, z.B. die Festplattengröße, die in den vergangenen Jahren deutlich langsamer angestiegen ist als zuvor. Ferner ist sowohl bei der Taktfrequenz als auch beim Preis für einen gut ausgestatteten PC eine gewisse Stabilisierung eingetreten, so dass eine Näherung durch eine Exponentialkurve nicht mehr angemessen scheint.

Aufgabe 5 Finden Sie heraus, welche Arten von PCs derzeit auf dem Markt beworben werden und wie sie sich unterscheiden!

Eine Auswahl:

- Desktop-PC: Der „klassische“ PC mit Tower-, Midi-Tower, und anderen Gehäusen
- Workstation: Ein besonders leistungsfähiger PC. Der Begriff ist auch bei High-End-Notebooks zu finden.
- Notebook (Laptop): Tragbarer Rechner mit Akku und eingebautem Flachbildschirm
- Netbook: besonders handlicher, billiger Notebook-ähnlicher Rechner mit geringer Ausstattung und Rechenleistung
- Tablet PC: Notebook-ähnlicher Rechner mit Touch Screen, der per Berührung oder Stifteingabe bedient wird. Eine Tastatur kann physisch vorhanden sein, oder sie wird als virtuelle Tastatur auf dem Bildschirm abgebildet.

Aufgabe 6 Schrauben Sie einen PC auf und sehen Sie nach, welche Komponenten eingebaut sind und wie sie miteinander verbunden sind! (Für etwaige Garantieverluste wird keine Gewähr übernommen ...)

Diese Aufgabe müssen Sie schon selber lösen ...

Aufgabe 7 Informieren Sie sich über das Aussehen eines Serverraumes. Wenn möglich, besichtigen Sie einen. Welche Geräte und Vorrichtungen finden sich außer den Servern typischerweise noch darin? Wie wird die Datensicherung (Erstellung von Sicherheitskopien) durchgeführt?

Man findet dort z.B. auch noch

- Speichermedien wie SAN
- Netzwerkkomponenten wie Switches, Router, Firewall
- Netzwerkanbindungen wie Netzwerk-Backbone, Glasfaser nach draußen
- Geräte zur Datensicherung wie Tape Library: In dieser befindet sich eine Menge von Backup-Bändern, die von der Library eigenständig verwaltet wird. In gewissen Zeitabständen werden die Bänder mit neuen Daten überschrieben, verschlissene Bänder werden ausgesondert.
- Tresor für Backup-Medien; dieser sollte idealerweise in einem anderen Raum stehen
- Brand- und Wassermeldesensoren
- Feuerlöscher oder Löschanlage
- Alarmanlage

Aufgabe 8 Welche Hauptvorteile von Mainframes vermuten Sie?

Der Client bzw. das Terminal, an dem der Benutzer arbeitet, braucht keine leistungsfähige Hardware zu besitzen. Häufig reicht bereits, wenn es ein vom Mainframe geliefertes Bild darstellen und Benutzereingaben an den Mainframe weiterleiten kann. Festplatten und optische Laufwerke sind nicht immer vorhanden. Das führt zu geringen Kosten und geringem Verwaltungsaufwand pro Benutzer. Datenspeicherung und somit auch Datensicherung kön-

nen sehr gut zentral erfolgen. Sensible Informationen können in besonders geschützten Räumen untergebracht werden.

Aufgabe 9 Welche Art von Clients eignet sich besonders gut in Verbindung mit Mainframes?

Thin Clients eignen sich besonders gut, siehe auch Frage 8.

Aufgabe 10 Informieren Sie sich auf http://www.top500.org/ über die derzeit schnellsten Supercomputer. Welche Rechenleistung haben sie? Welche Informationen lassen sich zu Hauptspeicher, Anzahl der Prozessorkerne und Betriebssystem finden?

Gemäß Liste vom Juni 2016 war der schnellste Rechner der chinesische Sunway TaihuLight, der im National Supercomputing Center in Wuxi steht. Seine Rechenleistung beträgt 93 Petaflop/s beim Linpack Benchmark. Er verfügt über 40.960 Knoten mit je 260 Kernen, was insgesamt 10.649.600 Kerne ergibt. In der derzeitigen Ausbaustufe besitzt er 1,31 Petabyte RAM. Betriebssystem ist Sunway RaiseOS 2.0.5, eine Linux-Variante.

Die nötige elektrische Leistung beträgt 15,37 MW. Die Anschaffungskosten werden mit umgerechnet 273 Millionen US-Dollar angegeben. Der Computer wird u.a. eingesetzt zur Erdölsuche, für Life Sciences und zur Wettervorhersage.

Gemäß Sublistengenerator auf der Website werden sämtliche der 500 schnellsten Rechner mit Unixoiden Betriebssystemen (u.a. Linux, Unix) betrieben. Dabei entfallen 497 auf Linux, die restlichen 3 auf Unix. Windows und andere Betriebssysteme sind dort nicht vertreten.

Aufgabe 11 Finden Sie 10 Embedded Systems in Ihrem Zuhause und in Ihrer unmittelbaren Umgebung. Sammeln Sie so viele Informationen wie möglich darüber (Zweck, Typ des Mikrocontrollers, Speichergröße, etc.)

Z.B. EC-Karte, Mensa-Karte, Mitarbeiterausweis, elektronischer Personalausweis und Reisepass, Handy, Waschmaschine, DVD-Player, Fernseher, Festplattenrecorder, PC-Tastatur, Soundkarte im PC, etc. Die weiteren Informationen sind stark von der Art des Mikrocontrollers abhängig, z.B. 8-Bit-Mikrocontroller auf 8051-Basis und 16 KB RAM, 32 kB ROM und 64 kB Flash-Speicher.

Aufgabe 12 Welche Grundvoraussetzung müssen die Adressen erfüllen, die von Speicher- und E/A-Bausteinen belegt werden?

Sie müssen eindeutig sein und dürfen sich nicht überschneiden. Einige Prozessoren verfügen allerdings über einen „Umschalter“ zwischen Speicher- und E/A-Bereich, der doch erlaubt, dass beide dieselben Adressen belegen.

Zusammenhängend brauchen die Adressbereiche nicht notwendigerweise zu sein. Es gibt bei vielen Rechnern unbelegte Adressen, die keinem Baustein zugeordnet sind. Das ist der Fall, wenn nicht die maximal mögliche Zahl von Speicher- und E/A-Bausteinen vorhanden ist.

Aufgabe 13 Bei jeder Adresse fühlt sich MAXIMAL ein Baustein angesprochen. Unter welchen Gegebenheiten könnte es vorkommen, dass GAR KEINER angesprochen wird? Was könnte in so einem Fall passieren? Wie lässt es sich vermeiden?

Nicht zu jedem Zeitpunkt wird vom Prozessor ein Speicher- oder E/A-Baustein angesprochen. Daher kommt es sehr häufig vor, dass kein Baustein ausgewählt wird. Hier ist allerdings eher gemeint, dass der Prozessor Daten von einer Adresse lesen oder dorthin schreiben will, diese Adresse aber keinem Baustein zugeordnet ist.

Ein Lese- oder Schreibvorgang würde also entweder ignoriert werden oder er würde eine Fehlermeldung („Speicherzugriffsverletzung“) hervorrufen.

Das Ansprechen nicht vorhandener Adressen könnte z.B. ein Programmierfehler sein. Der Programmierer sollte also sorgfältig Grenzen von Arrays beachten und nicht z.B. i[5] ansprechen, wenn es nur maximal i[4] gibt.

Aufgabe 14 Welchen Einfluss der Wortbreite auf die Geschwindigkeit des Computers vermuten Sie?

Wenn man die Wortbreite verdoppelt, können in einem Zyklus doppelt so viele Daten übertragen werden wie zuvor. Die Performance des Computer erhöht sich also deutlich.

Weil man nicht immer die volle Wortbreite benötigt und nicht ununterbrochen Daten über den Bus schickt, wird sich die Performance aber weniger als verdoppeln.

Aufgabe 15 Welchen Vorteil könnte es haben, wenn Befehle auf Vorrat in den Predecode Cache geholt werden? Unter welchen Umständen bringt es nichts?

Im Predecode Cache wird eine vordecodierte Form der Befehle abgelegt. Das erspart erneutes Holen und Decodieren, wenn Befehle mehrfach ausgeführt werden, z.B. bei Programmschleifen.

Wenn Befehle nicht mehrfach benötigt werden, bringt der Predecode Cache keinen Vorteil.

Aufgabe 16 Ein PC habe einen Vierkernprozessor. Wie ist er nach der FLYNNschen Taxonomie zu klassifizieren?

Jeder Kern kann unabhängig von den anderen Befehle abarbeiten. Zu einem gegebenen Zeitpunkt werden also mehrere Befehle (MI) ausgeführt, jeder mit unterschiedlichen Daten (MD). Der Vierkernprozessor fällt also in die Kategorie MIMD.

Aufgabe 17 In welche Klasse der FLYNNschen Taxonomie fällt ein moderner Grafikprozessor?

Eine typische Anwendung von Grafikprozessoren ist die gleichzeitige Bearbeitung zahlreicher Teile eines Bildes (MD). Dazu ist eine zentrale Steuerung (SI) sinnvoll, weil jeder Teil auf dieselbe Weise bearbeitet wird, z.B. um Kanten zu extrahieren oder eine Filterung durchzuführen. Es handelt sich dabei somit um eine SIMD-Architektur. Sie ist nach wie vor weit verbreitet und besitzt den Vorteil eines relativ einfachen und kostengünstigen Aufbaus.

Immer häufiger werden Grafikprozessoren aber für universellere Aufgaben eingesetzt, z.B. für wissenschaftliche Berechnungen und das Lösen von Gleichungssystemen. Daher ist ein Trend zu MIMD zu beobachten, wobei die Grafikkerne zwar gleichartige Operationen durchführen, also dasselbe Programm abarbeiten. Das erfolgt aber nicht mehr unter streng zentraler Kontrolle, sondern jeder Kern kann zu einem gegebenen Zeitpunkt an einer anderen Stelle des Programmes sein. Dafür wurde der Begriff ***SPMD*** (single program, multiple data) geprägt. SPMD ist ein Spezialfall von MIMD.

Aufgabe 18 Wie stellt man Multiprozessorsysteme im ECS dar?

Multiprozessorsysteme haben mehrere Prozessoren, somit auch mehrere Steuerwerke und mehrere Rechenwerke. Davon unabhängig ist die Wortbreite. Beim Tripel (k, d, w) sind also k und d jeweils größer 1.

Aufgabe 19 Wie stellt man Feldrechner im ECS dar?

Ein Feldrechner zeichnet sich dadurch aus, dass ein Feld von Verarbeitungselementen, also Rechenwerken, zentral gesteuert wird. Es gilt also k=1, d>1.

Aufgabe 20 Woran könnte es liegen, dass zahlreiche Benchmark-Programme keine Grafik- und E/A-Befehle messen?

Grafik und E/A befinden sich außerhalb der CPU. Ein Benchmark-Programm, das die CPU-Leistung misst, wird diese Bereiche also unberücksichtigt lassen.

Benchmark-Programme, die das gesamte System einbeziehen, ermitteln Grafikleistung und E/A-Leistung häufig als separate Werte.

Aufgabe 21 Warum ist eine statische Einteilung der Computer nach Rechenleistung nicht sinnvoll? Z.B. „Ab Rechenleistung x handelt es sich um einen Supercomputer“.

Die Rechenleistung ist eine sich äußerst dynamisch ändernde Größe. Ein Supercomputer, der zu den weltschnellsten zählt, kommt einige Jahre später womöglich nicht einmal mehr unter die Top 500. Für solche dynamischen Größen ist eine Angabe statischer Größen wenig sinnvoll. Besser sind „mitwachsende“ Definitionen, die z.B. einen Supercomputer in Relation zu Nicht-Supercomputern setzen.

Aufgabe 22 In den Medien wird von einem Supercomputer berichtet, der z.B. 25 PFLOPS (PetaFLOPS) schnell ist. Wieviele MFLOPS sind das?

25 PFLOPS = 25 · 10^{15} FLOPS = 25 · $10^{15}/10^{6}$ MFLOPS = 25 · 10^{9} MFLOPS = = 25 000 000 000 MFLOPS

Aufgabe 23 Wie lange braucht der 25 PFLOPS-Supercomputer für eine Berechnung, die bei 1 MFLOPS Rechenleistung 10 Jahre dauert?

T = 10 [a]/(25 · 10^{9}) = 10 · 365 · 24 · 60 · 60 [s]/(25 · 10^{9}) = 0,0126144 [s] = 12,6144 [ms]

Oder mit anderen Worten: Der Supercomputer könnte jede Sekunde 79 solche Berechnungen durchführen, die bei dem Beispielcomputer jeweils 10 Jahre dauern würden.

Aufgabe 24 Führt Caching in jedem Falle zu einem schnelleren Zugriff auf Speicherstellen oder sind auch Fälle denkbar, wo Zugriffe langsamer als ohne Caching erfolgen?

Das Caching an sich benötigt etwas Zeit: Der Cache-Controller muss nachsehen, ob sich die Daten bereits im Cache befinden. Das wäre ohne Caching nicht nötig.

Andererseits bringt das Caching keinen Nutzen, wenn Daten nur ein einziges Mal benötigt werden. In diesem Fall bleibt also nur der Nachteil des höheren Zeitaufwandes, so dass die Zugriffe ohne Caching schneller erfolgen würden.

Aufgabe 25 Für den Cache benötigt man ohnehin schnellen Speicher. Was könnte der Grund dafür sein, dass man nicht den gesamten Hauptspeicher aus diesen Bausteinen aufbaut? Dann bräuchte man kein Caching.

Cache für den Hauptspeicher ist üblicherweise aus sehr schnellem und damit auch extrem teurem statischen RAM aufgebaut. Neben dem Preisnachteil ist statisches RAM auch nicht so hoch integrierbar wie das für den Hauptspeicher eingesetzte dynamische RAM. Weil man den Cache viel kleiner als den Hauptspeicher wählt, bekommt man einen guten Kompromiss aus Mehrkosten und Nutzen.

Aufgabe 26 Welche Auswirkung der Cache-Größe auf die Performance ist zu erwarten?

Je größer der Cache, desto mehr Code bzw. Daten profitieren vom Caching. Der Zusammenhang ist allerdings nicht proportional, sondern eher exponentiell. Die ersten paar Kilobytes Cache bringen am meisten, denn Schleifen, die häufig durchlaufen werden, sind meist nicht besonders groß und passen idealerweise vollständig in den Cache. Ähnliches gilt für die am meisten verwendeten Datenstrukturen. Man kann mit wenigen Kilobytes Cache schon recht hohe Trefferraten und somit eine gute Performanceverbesserung erzielen.

Um die Performance spürbar weiter zu verbessern, muss man den Cache-Speicher z.B. verdoppeln oder vervierfachen. Jede weitere Performancesteigerung benötigt um ein Mehrfaches größeren Cache.

Aufgabe 27 Durch welche Art von Ereignis könnte das Zurückschreiben der Daten in den Hauptspeicher ausgelöst werden?

Das Zurückschreiben erfolgt, sobald die Cache Line anderweitig benötigt wird. Das könnte z.B. der Fall sein, wenn neue Anwendungen gestartet werden oder bisher nicht verwendete Bereiche einer Anwendung durchlaufen werden. Auch das Öffnen einer Datei, aus der größere Datenmengen gelesen und im Cache zwischengespeichert werden, könnte das Zurückschreiben auslösen.

Aufgabe 28 Was könnte passieren, wenn man den Hauptspeicher aufrüstet, um einen Rechner schneller zu machen?

Falls man mehr Hauptspeicher einbaut als das Cacheable Area umfasst, dann wird ein Teil des Hauptspeichers nicht mehr vom Caching profitieren. Es kann sein, dass ein Teil des Betriebssystems oder anderer Code bzw. Daten, die häufig benötigt werden, in diesen langsamen Teil des Hauptspeichers gelangen und somit den Rechner herunterbremsen.

Die Problematik wird dadurch verschärft, dass Betriebssysteme dazu neigen, den gesamten Hauptspeicher mit Code oder Daten zu belegen. Der Hauptspeicher ist nämlich deutlich schneller als eine Festplatte, und wenn Daten schon im Hauptspeicher stehen und nicht erst von der Festplatte geladen werden müssen, geht das erheblich schneller. So wird der Hauptspeicher teilweise „vorsorglich“ mit Daten gefüllt, für den Fall dass man diese einmal brauchen sollte. Dabei wird kein Unterschied zwischen dem Bereich innerhalb und außerhalb des Cacheable Area gemacht.

Anmerkung: Unabhängig vom Caching kann ein Prozessor im 32-Bit-Modus nur maximal 4 GByte ansprechen, von denen je nach Betriebssystem z.B. nur 3 GByte genutzt werden können. Baut man mehr Hauptspeicher ein, verwendet aber kein 64-Bit-Betriebssystem, dann hat man keinen Vorteil von dem über diese Grenzen hinausgehenden Hauptspeicher.

Aufgabe 29 Warum kann man die Cache-Inhalte im Gegensatz zu Büchern einfach wegwerfen?

Bücher hat man üblicherweise nur einmal zur Verfügung und man wird sich nicht jedes Mal eine Kopie davon machen, wenn man es aus dem Regal holt. Somit existiert nur das Original, das seinen Ort wechselt.

Im Cache wird dagegen jeweils eine Kopie des Originals angelegt. Somit kann man die Kopie problemlos löschen, wenn man sie nicht mehr benötigt, denn das Original bleibt an der ursprünglichen Stelle erhalten.

Aufgabe 30 Welchen Einfluss hat die Pipelinelänge auf die Performance? (beispielhaft)

Hat die Pipeline nur wenige Stufen, dann werden nur wenige Anweisungen parallel abgearbeitet. Die Parallelisierung ist somit gering, der Nutzen ebenfalls.

Ist die Pipeline sehr lang, dann benötigt man sehr viele passende Operationen hintereinander, damit die Pipeline nicht leerläuft bzw. das Risiko des Leerlaufens steigt mit der Pipelinelänge.

Eine optimale Pipeline darf somit weder zu kurz noch zu lang sein. Für viele Zwecke ist eine Pipelinelänge von 8 bis 9 Stufen angemessen.

Aufgabe 31 Alles in allem scheint die Harvard-Architektur das bessere Konzept zu sein. Was mögen Gründe sein, warum die VON-NEUMANN-ARCHITEKTUR im PC-Bereich trotzdem eine so dominierende Rolle besitzt?

Ein wesentlicher Grund ist, dass PCs von je her eine VON-NEUMANN-ARCHITEKTUR besitzen. Die gesamte Software, sowohl Betriebssysteme als auch Anwendungen, sind darauf ausgelegt. Ein grundlegender Architekturwechsel würde es erfordern, auch die Software grundlegend zu überarbeiten. Das würde immense Kosten verursachen.

Aufgabe 32 Welchen Unterschied macht es, ob man von einem UND-Gatter oder von einer UND-Verknüpfung redet?

Die UND-Verknüpfung ist eine Operation der BOOLEschen Algebra, die mit Nullen und Einsen arbeitet und ein abstraktes Denkmodell ist. Dagegen ist das UND-Gatter ein Stück Hardware, das die UND-Verknüpfung umsetzt.

UND-Gatter können je nach verwendeter Repräsentation von Nullen und Einsen sowie abhängig von der Technologie ganz unterschiedlich aufgebaut sein. Dagegen funktioniert eine UND-Verknüpfung im Sinne der BOOLEschen Algebra immer auf dieselbe Weise.

Aufgabe 33 Geben Sie die Wahrheitstabelle eines NAND-Gatters mit drei Eingängen an!

Tab. A.2: Wahrheitstabelle eines NAND-Gatters mit drei Eingängen

x_2	x_1	x_0	y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Aufgabe 34 Geben Sie die Wahrheitstabelle eines UND-Gatters mit zwei Eingängen an, bei dem der Eingang x_1 invertiert ist!

Tab. A.3: Wahrheitstabelle eines UND-Gatters mit zwei Eingängen und invertiertem x1

x_1	x_0	$\neg x_1$	$y=(\neg x_1)x_0$
0	0	1	0
0	1	1	1
1	0	0	0
1	1	0	0

Wie man sieht, unterscheidet sich die Spalte y von der der NAND-Verknüpfung. Es spielt also eine Rolle, ob man am Ausgang oder an einem Eingang invertiert.

Aufgabe 35 Geben Sie die Wahrheitstabelle eines Multiplexers 4:1 an!

Tab. A.4: Wahrheitstabelle eines Multiplexers 4:1

s_1	s_0	y
0	0	x_0
0	1	x_1
1	0	x_2
1	1	x_3

Der Multiplexer benötigt zwei Steuersignale s_0 und s_1, um damit einen von $4 = 2^2$ Eingängen $x_0 \ldots x_3$ auswählen zu können. Der ausgewählte Eingang wird zum Ausgang y durchgeschaltet, dieser nimmt also den Wert des Eingangs an.

Aufgabe 36 Beschalten Sie einen 8:1-Multiplexer so, dass folgende Wahrheitstabelle implementiert wird:

x_2	x_1	x_0	y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Hinweis: „Beschalten“ heißt, dass an vorhandene Eingänge eine feste 0 oder feste 1 angelegt wird, dass Ein- und Ausgänge mit Formelbuchstaben aus der Aufgabenstellung versehen werden und – falls erlaubt – dass Gatter an die Ein- und Ausgänge geschaltet werden. Eine Änderung am internen Aufbau eines vorgegebenen Bausteins ist nicht gestattet.

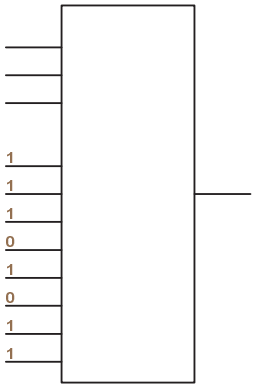

Abb. A.1: Beschaltung eines MUX 8:1 zur Umsetzung einer Wahrheitstabelle

In einer Zeile n der Wahrheitstabelle soll am Ausgang y ein bestimmter Wert erscheinen. Legt man diesen Wert an den Eingang e_n des Multiplexers, dann wird dieser zum Ausgang y durchgeschaltet, wenn die Steuersignale x_0 … x_2 den Wert n als Dualzahl annehmen. Wir können also mit x_0 … x_2 jeden Wert der Wahrheitstabelle an y erzeugen.

Aufgabe 37 Geben Sie die Wahrheitstabelle eines Demultiplexers 1:4 an!

Tab. A.5: Wahrheitstabelle eines Demultiplexers 1:4

s_1	s_0	y_0	y_1	y_2	y_3
0	0	x	0	0	0
0	1	0	x	0	0
1	0	0	0	x	0
1	1	0	0	0	x

Der Eingang x wird zu demjenigen Ausgang y_i weitergeleitet, dessen Nummer i mittels s_1 und s_0 als Dualzahl ausgewählt wurde.

Aufgabe 38 Beschalten Sie einen 1:8-Demultiplexer so, dass er als 2-Bit-Adressdecoder arbeitet.

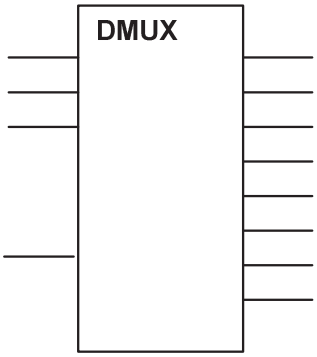

Abb. A.2: DMUX 1:8

In Abb. A.2 ist das Schaltzeichen des unbeschalteten DMUX 1:8 zu sehen. Es sind zwei Teilaufgaben zu lösen:

1. Aus dem Demultiplexer soll ein Adressdecoder gemacht werden.
   Ein Adressdecoder legt einen ausgewählten Ausgang auf 1, während ein Demultiplexer einen Eingang e dorthin durchschaltet. Legen wir beim Demultiplexer somit e auf 1, haben wir dieselbe Arbeitsweise wie beim Adressdecoder.
2. Der 1:8-Demultiplexer arbeitet mit 3 Steuersignalen, der 2-Bit-Adressdecoder nur mit zweien.
   Wir nutzen also nur den halben Demultiplexer und lassen die andere Hälfte „brach liegen“. Wir können dazu s_0 dauerhaft auf 0 legen oder dauerhaft auf 1 legen. Aber genauso gut könnte man das mit s_1 oder s_2 machen. Daher gibt es sechs mögliche Lösungen. Eine davon ist in Abb. A.3 zu sehen.

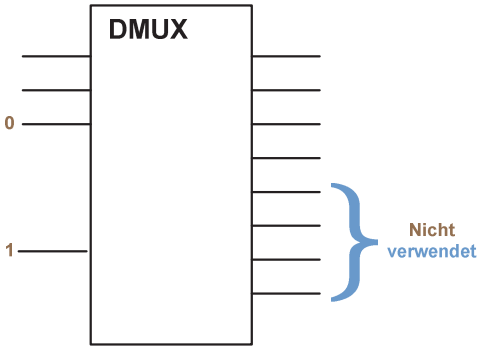

Abb. A.3: Beschaltung eines DMUX 1:8

Wenn man s_2 konstant auf 0 legt, kann $(s_2s_1s_0)_2$ nur noch Werte von $(000)_2$ bis $(011)_2$ annehmen. Es können also nur noch die Ausgänge y_0 bis y_3 selektiert werden. y_4 bis y_7 liegen somit brach.

Aufgabe 39 Wie könnte man mit weniger Steuersignalen auskommen?

Man könnte beispielsweise beim Sender und Empfänger die momentane Position speichern und mit einem einzigen Steuersignal einen Zähltakt übertragen, sobald auf den nächsthöheren Kanal geschaltet werden soll.

Nachteilig daran ist, dass man Sender und Empfänger dennoch synchronisieren können muss, damit z.B. beim Verbindungsaufbau beide mit demselben Kanal beginnen. Auch kann man die Kanäle auf diese Weise nur in einer festgelegten Reihenfolge durchschalten und nicht mehr beliebig.

Aufgabe 40 Wie viele Leitungen benötigt man mit dem beschriebenen Verfahren zur Übertragung von 128 Kanälen?

Es sind 1 + ld 128 = 1 + 7 = 8 Leitungen erforderlich.

Aufgabe 41 Skizzieren Sie die Schaltung eines 2×2:2-Multiplexers, der aus 2:1-Multiplexern zu bilden ist. Geben Sie das Schaltzeichen an.

Ein Multiplexer 2×2:2 hat zwei Eingangsbündel A und B, die jeweils aus 2 Signalen bestehen. Eines davon wird ausgewählt und zum Ausgang weitergeleitet, der somit ebenfalls ein Bündel aus 2 Signalen sein muss.

Um eines von 2 Bündeln auszuwählen, genügt ein einziges Steuersignal, da $2^1 = 2$. Somit kommt man auf folgendes Schaltzeichen:

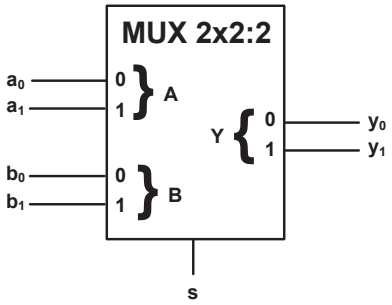

Abb. A.4: MUX $2 \times 2{:}2$

Möchte man diesen Multiplexer aus 2:1-Multiplexern bilden, dann muss man diese mit einem gemeinsamen s versehen, weil das Steuersignal nun nicht nur eines sondern 2 Datensignale gleichzeitig zum Ausgang weiterleiten soll:

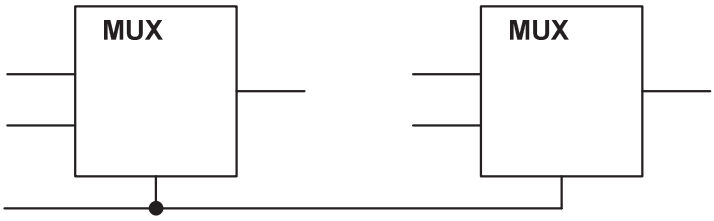

Abb. A.5: MUX $2 \times 2{:}2$ aus MUX 2:1

Aufgabe 42 Bilden Sie einen Multiplexer $2 \times 4{:}4$ aus 2:1-Multiplexern.

Man geht hier genauso wie bei der vorigen Aufgabe vor, nur dass man jetzt Bündel mit 4 Signalen hat und somit noch zwei weitere MUX 2:1 ergänzen muss:

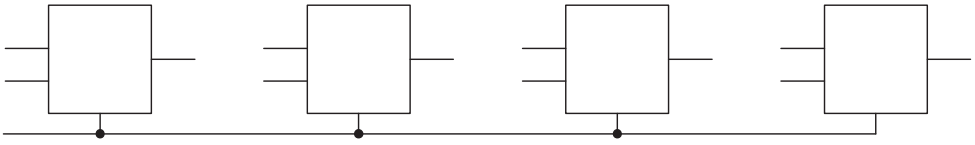

Abb. A.6: MUX $2 \times 4{:}4$ aus MUX 2:1

Aufgabe 43 Skizzieren Sie, wie man einen Multiplexer einsetzen könnte, um wahlweise 16 Testsignale oder 16 Datensignale an Anschlusspins eines Bausteins zu leiten.

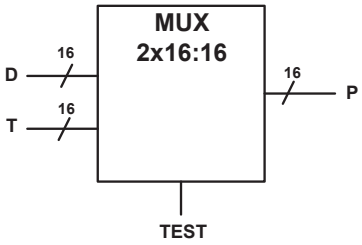

Abb. A.7: Gemultiplextes Herausführen zweier Busse auf Anschluss-Pins

Sowohl Datensignale D als auch Testsignale T besitzen eine Wortbreite von 16 Bits. Ebenso gibt es 16 Pins, auf die das Bündel P geführt wird. Weil lediglich zwischen zwei Bündeln D und T umgeschaltet werden soll, reicht zur Auswahl ein einziges Steuersignal TEST. Für TEST=0 wird D zu den Pins geleitet, für TEST=1 T.

Aufgabe 44 Wie könnte man durch Beschaltung mit geeigneten Gattern aus einem „normalen“ RS-Flipflop ein solches mit vorrangigem S-Eingang bekommen?

Die Beschaltung muss dafür sorgen, dass im Fall S’=R’=1 am Flipflop S=1 und R=0 entsteht. Andererseits dürfen andere Eingangsbelegungen nicht verändert werden.

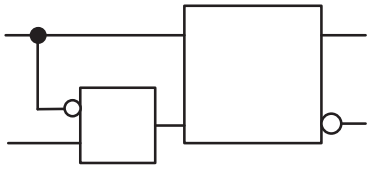

Abb. A.8: Beschaltung eines RS-Flipflops für vorrangigen S-Eingang

Das kann durch ein Und-Gatter mit negiertem Eingang erreicht werden. Wenn S’ = 1 ist, dann wird durch die Negation eine Null daraus. Ein Und-Gatter mit einer 0 an einem Eingang kann am Ausgang nichts anderes als eine 0 hervorbringen. Somit wird aus S’ = R’ = 1 tatsächlich am Flipflop S = 1 und R = 0.

S’ wird nicht verändert. Also gilt immer S = S’. Wie man überprüfen kann, ergeben sich somit abhängig von S’ und R’ folgende Werte für S und R:

Tab. A.6: Umsetzung der Eingangswerte durch Gatterbeschaltung beim RS-Flipflop

S'	R'	S	R
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	0

Aufgabe 45 Konstruieren Sie einen Zähler, der bis 12 zählt und dann wieder bei 0 beginnt. Verwenden Sie dafür einen CTR16.

Die Zählfolge soll also die folgende sein:

Tab. A.7: Zählfolge eines Modulo-12-Zählers

dez.	y_3	y_2	y_1	y_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
(13	1	1	0	1)

Der Zählerstand 13 kann verwendet werden, um den Zähler auf 0 zurückzusetzen: sobald dieser Wert erscheint, wird ein Impuls auf einen asynchronen Rücksetzeingang gegeben, und der Zähler startet wieder bei 0.

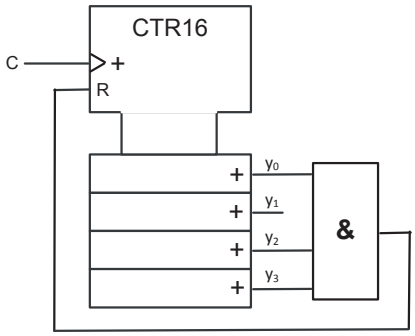

Abb. A.9: Modulo-12-Zähler

Den Zählerstand 13 kann man mit Hilfe eines Und-Gatters erkennen. Eigentlich müsste man ein Und-Gatter mit 4 Eingängen nehmen, wobei y_1 zu invertieren wäre, weil es beim Zählerstand 13 Null ist. Wenn wir aber die Tab. A.7 genauer ansehen, dann stellen wir fest, dass 13 der erste Zählerstand ist, bei dem $y_0 = y_2 = y_3$ gilt. Es reicht also, nur diese drei Eingänge zu testen. So kommt man auf die Schaltung in Abb. A.9.

Aufgabe 46 Konstruieren Sie für eine Abpackmaschine einen Zähler, der beim Zählerstand 12 stehen bleibt. Verwenden Sie dafür einen CTR16.

Diesmal soll der Zähler bei 12 stehen bleiben. Die 13 wird also gar nicht mehr erreicht. Daher decodieren wir die 12. Sobald sie erkannt wird, also am Ausgang des rechten Und-Gatters eine 1 erscheint, werden durch die Negation am linken Und-Gatter weitere Zähltakte blockiert, denn am Ausgang des rechten Und-Gatters kann nur noch eine Null entstehen.

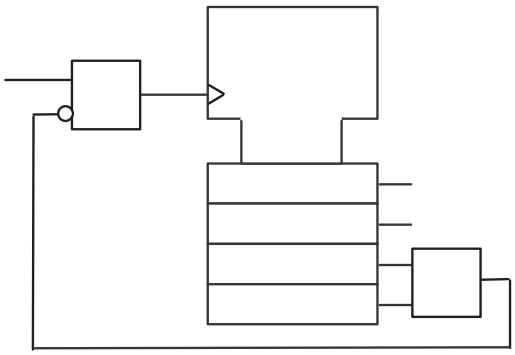

Abb. A.10: Zähler, der bei 12 stehenbleibt

Der Zähler sollte auch die Möglichkeit bieten, wieder rückgesetzt zu werden und von vorn zu beginnen. Daher wäre es sinnvoll, einen asynchronen Rücksetzeingang beim Zähler vorzusehen, der zum Rücksetzen mit einem Taster kurzzeitig mit 1 verbunden werden kann.

Aufgabe 47 Welcher Wertebereich ergibt sich für 16- und 32-Bit-Zahlen in der Vorzeichen-Betrags-Darstellung?

Bei n Bits (incl. Vorzeichen) lassen sich Zahlen von $-2^{n-1} + 1$ bis $+2^{n-1} - 1$ darstellen.

Für n=16 ergibt sich $-2^{15} + 1$ .. $+2^{15} - 1$ bzw. –32767 .. +32767.

Für n=32 bekommt man $-2^{31} + 1$ .. $+2^{31} - 1$ bzw. –2 147 483 647 .. +2 147 483 647.

Aufgabe 48 Stellen Sie die Zahl $(-23)_{10}$ als Dualzahl in der Vorzeichen-Betrags-Darstellung mit 7 Bit und mit 9 Bit Wortbreite dar!

$(23)_{10}=(10111)_2$

Dieses Umwandlungsergebnis (z.B. aus dem Taschenrechner) ist vorzeichenlos.

Wir wollen zunächst eine vorzeichenbehaftete Zahl mit 7 Bits Wortbreite. Ein Bit wird für das Vorzeichen verwendet. Das oben stehende vorzeichenlos Umwandlungsergebnis benötigt 5 Bits. Es ist also noch 1 Bit aufzufüllen. Das geschieht durch Einfügen von Nullen nach dem Vorzeichenbit. somit bekommt man:

$(-23)_{10}=(1\ 0\ 10111)_{2\ VZB}$

Vorzeichenbit und aufgefülltes Bit sind mit Abständen gekennzeichnet.

Eine 9-Bit-Zahl bekommt man, wenn man weitere Nullen einfügt:

$(-23)_{10}=(1\ 000\ 10111)_{2\ VZB}$

Aufgabe 49 Stellen Sie die Zahl $(-23)_{10}$ als Dualzahl in der Einerkomplement-Darstellung mit 7 Bit und mit 9 Bit Wortbreite dar!

$(23)_{10} = (10111)_2$

Wir wollen eine 7-Bit-Zahl im Einerkomplement. Für die Umwandlung gibt es zwei Alternativen:

a) Wir wandeln zuerst ins Einerkomplement um und füllen dann mit Einsen auf 7 Bits auf:

   $(23)_{10} = (10111)_2$ ist vorzeichenlos! Wenn wir das direkt invertieren würden, also zu $(01000)_{2\ EK}$ umwandeln würden, dann würde uns ein Bit für das Vorzeichen fehlen. Das Umwandlungsergebnis hätte eine führende Null, wäre also positiv. Daran könnte man erkennen, dass etwas schief gegangen sein muss.

   Wir müssen also vor der Umwandlung (mindestens) eine führende Null ergänzen:

   $(23)_{10} = (010111)_2$

   $(-23)_{10} = (101000)_{2\ EK}$

   Zum Auffüllen auf die gewünschte Wortbreite werden Einsen zwischen MSB und nachfolgenden Bits eingefügt. In unserem Fall ist es genau eine Eins, was mit Abständen gekennzeichnet ist:

   $(-23)_{10} = (1\ 01000)_{2\ EK} = (1\ 1\ 01000)_{2\ EK}$

   Insgesamt ist das eher umständlich und fehlerträchtig. Einfacher ist folgende Alternative:

b) Wir füllen die vorzeichenlose Zahl zuerst mit Nullen auf und wandeln sie dann ins Einerkomplement um

   $(23)_{10} = (10111)_2 = (0010111)_2$

   $(-23)_{10} = (1101000)_{2\ EK}$

   Eine 9-Bit-Zahl bekommt man, wenn man zwei weitere Einsen einfügt, nach dem MSB also insgesamt drei:

   $(-23)_{10}= (1\ 111\ 01000)_{2\ EK}$

Aufgabe 50 Bilden Sie das Einerkomplement von $(CACA0)_{16}$ mit einer Wortbreite von 32 Bit.

32 Bit sind 8 Nibbles (Halbbytes) zu je 4 Bits. Jedes davon entspricht einer Hexadezimalziffer.

Wir ergänzen $(CACA0)_{16}$ somit auf 8 Stellen: $(000CACA0)_{16}$.

Dann invertieren wir: $(FFF3535F)_{16\ EK}$

Aufgabe 51 Stellen Sie die Zahl $(–23)_{10}$ als Dualzahl in der Zweierkomplement-Darstellung mit 7 Bit und mit 9 Bit Wortbreite dar!

$(–23)_{10} = (1101000)_{2\ EK} = (1101001)_{2\ ZK}$

bzw. mit 9 Bit Wortbreite:

$(–23)_{10} = (1\ 111\ 01000)_{2\ EK} = (1\ 111\ 01001)_{2\ ZK}$

Aufgabe 52 Bilden Sie das Zweierkomplement von $(CACA0)_{16}$ mit einer Wortbreite von 32 Bit.

$(FFF3535F)_{16\ EK} = (FFF35360)_{16\ ZK}$

Aufgabe 53 Kontrollieren Sie den angegebenen Wertebereich für den Datentyp long long.

Bei n Bits (incl. Vorzeichen) lassen sich unter Verwendung des Zweierkomplements Zahlen von $–2^{n–1}$ bis $+2^{n–1} – 1$ darstellen.

Für n=64 ergibt sich $–2^{63}$ .. $+2^{63} – 1$ bzw.

–9 223 372 036 854 775 808 .. +9 223 372 036 854 775 807.

Aufgabe 54 Zeigen Sie, wie man die Bits 4 und 7 eines 8-Bit-Registers löscht, die Bits 0, 3 und 6 setzt und ferner die Bits 1, 2 und 5 invertiert.

Man führt Verknüpfungen mit folgenden Bitmasken durch:

AND	01101111
OR	01001001
XOR	00100110

Aufgabe 55 Führen Sie folgende Rechnungen mit vorzeichenlosen Zahlen durch:

a) $(0100)_2 + (1000)_2$

Wertigkeit		3	2	1	0
Operand 1:		0	1	0	0
Operand 2:	+	1	0	0	0
Ergebnis:		1	1	0	0

b) $(1101)_2 + (10001)_2$

Wertigkeit		4	3	2	1	0
Operand 1:			1	1	0	1
Operand 2:	+	1_1	0_1	0_1	0_1	1
		1				
Ergebnis:		1	0	1	0	1	0

c) $(10110100)_2 + (00010111)_2$

Wertigkeit		7	6	5	4	3	2	1	0
Operand 1:		1	0	1	1	0	1	0	0
Operand 2:	+	0	0_1	0_1	1	0_1	1	1	1
Ergebnis:		1	1	0	0	1	0	1	1

d) $(10011101)_2 + (11011011)_2 + (00101010)_2$

Wertigkeit		7	6	5	4	3	2	1	0
Operand 1:		1	0	0	1	1	1	0	1
Operand 2:		1	1	0	1	1	0	1	1
Operand 3:	+	0_1	0_1	1_{1+1}	0	1_1	0_1	1_1	0
		1							
Ergebnis:		1	1	0	1	0	0	0	1	0

e) $(BEEF)_{16} + (1CF7)_{16}$

Wertigkeit		3	2	1	0
Operand 1:		B	E	E	F
Operand 2:	+	1_1	C_1	F_1	7
Ergebnis:		D	B	E	6

f) $(A03FD)_{16} + (46EA)_{16}$

Wertigkeit		4	3	2	1	0
Operand 1:		A	0	3	F	D
Operand 2:	+		4	6_1	E_1	A
Ergebnis:		A	4	A	E	**7**

Aufgabe 56 Führen Sie folgende Rechnungen mit vorzeichenlosen Zahlen durch:

a) $(1101)_2 - (1010)_2$

Wertigkeit		3	2	1	0
Operand 1:		1	1	0	1
Operand 2:	–	1	0_1	1	0
Ergebnis:		0	0	1	1

b) $(1000)_2 - (0111)_2$

Wertigkeit		3	2	1	0
Operand 1:		1	0	0	0
Operand 2:	–	0_1	1_1	1_1	1
Ergebnis:		0	0	0	1

c) $(10011101)_2 - (00011011)_2 - (00100010)_2$

Wertigkeit		7	6	5	4	3	2	1	0
Operand 1:		1	0	0	1	1	1	0	1
Operand 2:	–	0	0	0	1	1	0	1	1
Operand 3:	–	0	0_1	1	0	0	0_1	1	0
Ergebnis:		0	1	1	0	0	0	0	0

d) $(4CF5)_{16} - (30EB)_{16}$

Wertigkeit		3	2	1	0
Operand 1:		4	C	F	5
Operand 2:	–	3	0	E_1	B
Ergebnis:		1	C	0	**A**

Aufgabe 57 Wie groß wäre das Multiplikationsergebnis allgemein, wenn man den letzten Schiebeschritt vergessen würde?

Weil der letzte Schiebeschritt ein Rechtsschieben ist, wäre das Ergebnis doppelt so groß wie der korrekte Wert.

Aufgabe 58 Führen Sie die Multiplikation $(14)_{10} \cdot (10)_{10}$ mit vorzeichenlosen 4-Bit-Zahlen im Dualsystem mit beiden Algorithmen durch.

Verfahren 1:

		Op1									Op2			
Wertigkeit											3	2	1	0
						1	1	1	0	·	1	0	1	0
ZE		0	0	0	0	0	0	0	0					↘ keine Addition
	+				1	1	1	0	**0**					↘ ADD ZE, Op1'
ZE_{neu}		0	0	0	1	1	1	0	0					↘ keine Addition
	+		1_1	1_1	1	0	**0**	**0**	**0**					↘ ADD ZE, Op1'''
		1	0	0	0	1	1	0	0					

Verfahren 2:

		Op1								Op2			
Wertigkeit										3	2	1	0
		1	1	1	0					·	1	0	1	0
ZE		0	0	0	0								↘ SHR ZE, 1
ZE_{neu}		0	0	0	0	0							↘ ADD ZE, Op1
	+	1	1	1	0								
ZE'		1	1	1	0	0							SHR ZE, 1
ZE_{neu}		0	1	1	1	0	0						↘ SHR ZE, 1
ZE_{neu}		0	0	1	1	1	0	0					↘ ADD ZE, Op1
	+	1_1	1	1	0								
			1										
ZE'	1	0	0	0	1	1	0	0					↘ SHR ZE, 1
ZE_{neu}		1	0	0	0	1	1	0	0				

Aufgabe 59 Führen Sie die Ganzzahl-Division $(129)_{10} : (14)_{10}$ mit vorzeichenlosen Dualzahlen (8 bzw. 4 Bit) durch. Woran bemerkt man, dass die Division nicht aufgeht?

$(129)_{10} = (10000001)_2$

$(14)_{10} = (1110)_2$

		C	Akku				MQ				
Start		0	1	0	0	0	0	0	0	1	SHL (AK, MQ), 1
ZE		1	0	0	0	0	0	0	1	0	10000 ≥ 1110
Op2	–	1	1_1	1_1	1	0					Op2 subtrahieren
ZE'		0	0	0	1	0	0	0	1	**1**	MQ_0 setzen
ZE		0	0	1	0	0	0	1	1	0	SHL (AK, MQ), 1
ZE		0	0	1	0	0	0	1	1	**0**	00100 < 1110, 0 bleibt
ZE		0	1	0	0	0	1	1	0	0	SHL (AK, MQ), 1
ZE		0	1	0	0	0	1	1	0	**0**	01000 < 1110, 0 bleibt
ZE		1	0	0	0	1	1	0	0	0	SHL (AK, MQ), 1, 10001 ≥ 1110
Op2	–	1	1_1	1_1	1	0					Op2 subtrahieren
ZE		0	0	0	1	1	1	0	0	**1**	MQ_0 setzen
Ergebnis		0	1	0	0	1	0	0	0	0	MQ nach AK schieben, C löschen

Wir haben eine Division mit m = 4 Bit Divisor. Daher ist die Division nach 4 Schritten beendet. Wenn die Division aufgeht, ist der Akku zu diesem Zeitpunkt Null. In unserem Falle steht dort aber $(0011)_2 = 3$. Das ist der Divisionsrest, denn $9 \cdot 14 = 126$. 129 : 14 ist somit 9 Rest 3.

Aufgabe 60 Führen Sie folgende Berechnungen in der Vorzeichen-Betrags-Darstellung und im Zweierkomplement durch. Verwenden Sie dabei das Dualsystem und 8-Bit-Zahlen (incl. Vorzeichen). Kontrollieren Sie jeweils das Ergebnis, wo sinnvoll.

Zunächst wandeln wir die verwendeten Zahlen ins Dualsystem, diejenigen, die auch mit negativem Vorzeichen vorkommen, außerdem ins Zweierkomplement.

$(18)_{10} = (00010010)_2$, $(-18)_{10} = (11101101)_{2\,EK} = (11101110)_{2\,ZK}$
$(27)_{10} = (00011011)_2$
$(51)_{10} = (00110011)_2$, $(-51)_{10} = (11001100)_{2\,EK} = (11001101)_{2\,ZK}$
$(112)_{10} = (01110000)_2$, $(-112)_{10} = (10001111)_{2\,EK} = (10010000)_{2\,ZK}$

a) $-51 + 18 = ?$

Vorzeichen-Betragsdarstellung:

Z_1 ist negativ, Z_2 ist positiv, also dritter Fall der Tabelle 11.1.

$|E| = \big|\, |Z_1| - |Z_2| \,\big|$

$\|Z_1\|$:		0	1	1	0	0	1	1	$= (51)_{10}$
$\|Z_2\|$:	–	0	0	1	0	0	1	0	$= (18)_{10}$
$\|E\|$:		0	1	0	0	0	0	1	$= (33)_{10}$

Negativer Operand ist der betragsmäßig größere von beiden, also Ergebnisvorzeichen negativ. Das Ergebnis lautet $E = (1\ 0100001)_2 = (-33)_{10}$.

Zweierkomplement:

Z_1:		1	1	0	0	1	1	0	1	$= (-51)_{10}$
Z_2:	+	0	0	0	1	0	0	1	0	$= (+18)_{10}$
E_{ZK}:		1	1	0	1	1	1	1	1	
E_{EK}		1	1	0	1	1	1	1	0	
$\|E\|$		0	0	1	0	0	0	0	1	$= (+33)_{10}$

$\Rightarrow E = (-33)_{10}$

b) $18 + 27 = ?$

Addition zweier positiver Zahlen, kein Unterschied zwischen VZB, ZK und vorzeichenloser Darstellung!

Z_1:		0	0	0	1	0	0	1	0	$= (18)_{10}$
Z_2:	+	0	0	0_1	1	1	0_1	1	1	$= (27)_{10}$
E:		0	0	1	0	1	1	0	1	$= (45)_{10}$

c) $27 - 18 = ?$

Vorzeichen-Betragsdarstellung:

Z_1 ist positiv, Z_2 ist negativ, also zweiter Fall der Tabelle 11.1.

$|E| = \big|\, |Z_1| - |Z_2| \,\big|$

$\|Z_1\|$:		0	0	1	1	0	1	1	$= (27)_{10}$
$\|Z_2\|$:	–	0	0	1	0	0	1	0	$= (18)_{10}$
$\|E\|$:		0	0	0	1	0	0	1	$= (9)_{10}$

Positiver Operand ist der betragsmäßig größere von beiden, also Ergebnisvorzeichen positiv.

Das Ergebnis lautet $E = (0\ 0001001)_2 = (+9)_{10}$.

Zweierkomplement:

Z_1:		0	0	0	1	1	0	1	1	$= (+27)_{10}$
Z_2:	$+_1$	1_1	1_1	1_1	0_1	1_1	1_1	1	0	$= (-18)_{10}$
E_{ZK}:	~~+~~	0	0	0	0	1	0	0	1	$= (+9)_{10}$

d) $-112 - 18 = ?$

Vorzeichen-Betragsdarstellung:

Z_1 ist negativ, Z_2 ist negativ, also vierter Fall der Tab. 11.1.

$|E| = |Z_1| + |Z_2|$

$\|Z_1\|$:		1	1	1	0	0	0	0	$= (112)_{10}$
$\|Z_2\|$:	+	0_1	0_1	1	0	0	1	0	$= (18)_{10}$
$\|E\|$:	1	0	0	0	0	0	1	0	$= (130)_{10}$

Überlauf! Ergebnis ungültig!

Zweierkomplement:

Z_1:		1	0	0	1	0	0	0	0	$= (-112)_{10}$
Z_2:	$+_1$	1	1	1	0	1	1	1	0	$= (-18)_{10}$
E_{ZK}:	1	0	1	1	1	1	1	1	0	$= (+9)_{10}$

Überlauf! Ergebnis ungültig!

Aufgabe 61 Führen Sie die Multiplikation $10 \cdot 6 = 60$ im Dualsystem mit Hilfe des Algorithmus aus Kapitel 10.2 (Verfahren 2) durch.

	Op1								Op2			
Wertigkeit									3	2	1	0
	1	0	1	0					· 0	1	1	0
ZE	0	0	0	0								↘ SHR ZE, 1
ZE_{neu}	0	0	0	0	0						↘ ADD ZE, Op1	
	+ 1	0	1	0								
ZE'	1	0	1	0	0						SHR ZE, 1	
ZE_{neu}	0	1	0	1	0	0				↘ ADD ZE, Op1		
	+ 1	0	1	0								
ZE'	1	1	1	1	0	0				SHR ZE, 1		
ZE_{neu}	0	1	1	1	1	0	0		↘ SHR ZE, 1			
ZE_{neu}	0	0	1	1	1	1	0	0				

Aufgabe 62 Welche Werte dürfen die Exponenten annehmen, wenn man die Charakteristik 8 Bits groß macht und die Konstante $K_0 = 127$ nimmt?

$Ch := E + K_0 \Rightarrow E = Ch - K_0$

8 Bit Charakteristik $\Rightarrow 0 \le Ch \le 255$

$\Rightarrow 0 - K_0 \le E \le 255 - K_0$

$\Rightarrow -127 \le E \le +128$

Aufgabe 63 Welchen darstellbaren Zahlenbereich erhält man für 8 Bit Charakteristik und $K_0 = 127$, falls die Basis B=2 ist und die Zahlen normalisiert sind?

Bei normalisierten Zahlen bewegt sich der Betrag der Mantisse zwischen 0 und 1, weil der Vorkommateil Null ist: $0 < |M| < 1$.

Die 0.0 ist zwar definitionsgemäß nicht normalisiert. Sie fällt aber in den darstellbaren Zahlenbereich. Somit gilt $0 \le |M| < 1$.

Der darstellbare Zahlenbereich wird also durch den Exponenten bestimmt. Wie aus Aufgabe 16 hervorgeht, gilt:

$-127 \le E \le +128$.

Wegen der Basis B=2 gilt

$2^{-127} \le 2^E \le 2^{+128}$.

2^E ist der Faktor, mit dem die Mantisse multipliziert wird, um die Gleitkommazahl $Z = M \cdot B^E$ zu erhalten. Dieser Faktor liegt zwischen 2^{-127}, also ungefähr Null, und $2^{+128} \approx 3{,}403 \cdot 10^{38}$. Der Mantissenbetrag liegt zwischen 0 und 1, die Mantisse somit zwischen –1 und +1.

Also liegt Z zwischen $-3{,}403 \cdot 10^{38}$ und $+3{,}403 \cdot 10^{38}$. Das ist auch der darstellbare Zahlenbereich.

Aufgabe 64 Wieviele gültige (Nachkomma-)Stellen bezogen auf das Dezimalsystem besitzt eine float-Variable maximal, und wieviele eine double-Variable? Welche Auswirkungen kann das in der Praxis haben?

Mit m Mantissenbits lassen sich 2^m verschiedene Werte darstellen. Dazu benötigt man $n = \lg(2^m)$ Dezimalstellen. Es gilt:

$n = \lg(2^m) = m \lg 2$

Bei float beträgt m=23, somit $n \approx 6{,}92$ oder abgerundet 6 gültige Stellen.

Double verwendet m=52 Mantissenbits, was $n \approx 15{,}65$ oder 15 gültige Stellen ergibt.

Möchte man beispielsweise mit Geldbeträgen im Bereich von 10.000,00 Euro und mehr rechnen, ergeben sich bei Verwendung von float-Variablen Ungenauigkeiten. Stattdessen sollte man double-Variablen einsetzen.

Entsprechendes gilt, wenn die dargestellten Zahlen zwar keine allzu hohe Genauigkeit benötigen, aber z.B. viele tausend Male in einer Schleife miteinander verrechnet werden. Hier können sich die Ungenauigkeiten aufaddieren.

Aufgabe 65 Wandeln Sie folgende Zahlen in Dualbrüche mit 8 Bit Mantissenbetrag und 4 Bit Charakteristik, kein Hidden Bit, um. Wählen Sie einen sinnvollen Wert für K_0. Machen Sie die Probe und geben Sie den absoluten Umwandlungsfehler an.

a) 65,625

$(65)_{10} = (1000001)_2$

0,625 $\cdot 2 = 1 + 0{,}25$
0,25 $\cdot 2 = 0 + 0{,}5$
0,5 $\cdot 2 = 1 + 0{,}0$

An dieser Stelle können wir mit der Umwandlung aufhören, weil Multiplikation von 0,0 nur 0,0 ergeben kann. Somit würden wir lediglich Nullen ans Ende des Nachkommateils anhängen. Wir lesen die Bits von oben nach unten und normalisieren:

$x_W = (65{,}625)_{10} = (1000001{,}101)_2 = (0{,}1000001\mathbf{01})_2 \cdot 2^7$

Weil das Komma 7 Stellen nach links verschoben wird, müssen wir mit einem Faktor 2^7 ausgleichen. Nun schneiden wir nach der achten Nachkommastelle ab, so dass die oben fettgedruckte 0 und 1 wegfallen:

$x = (0{,}10000011)_2 \cdot 2^7$

Die 4-Bit-Charakteristik läuft von 0 ... 15. Es wird somit sinnvollerweise $K_0 := 7$ gewählt. Damit ergibt sich für die Charakteristik:

$Ch = K_0 + E = 7 + 7 = 14 = (1110)_2$

Die Zahl steht also so im Gleitpunktregister:

0	10000011	1110
Vz(M)	|M|	Ch

Wir machen nun die Probe:

$x = (0{,}10000011)_2 \cdot 2^{14-7} = (2^{-1} + 2^{-7} + 2^{-8}) \cdot 2^7 = 2^6 + 2^0 + 2^{-1} = 65{,}5$

$\Delta x = x - x_W = 65{,}5 - 65{,}625 = -0{,}125 = -2^{-3}$

Das Ergebnis ist also um ein Achtel zu klein. Das ist genau der Wert der Eins, die nicht mehr in den Mantissenbetrag passte.

b) 65,675

$(65)_{10} = (1000001)_2$

0,675 $\cdot 2 = 1 + 0{,}35$
0,35 $\cdot 2 = 0 + 0{,}7$
0,7 $\cdot 2 = 1 + 0{,}4$
0,4 $\cdot 2 = \mathbf{0 + 0{,}8}$
0,8 $\cdot 2 = \mathbf{1 + 0{,}6}$
0,6 $\cdot 2 = \mathbf{1 + 0{,}2}$
0,2 $\cdot 2 = \mathbf{0 + 0{,}4}$
0,4 $\cdot 2 = 0 + 0{,}8$
...

Wir stellen fest, dass sich der fett gedruckte Teil wiederholt. Sowohl ganzzahliger Anteil des Ergebnisses als auch Nachkommateil, nämlich 0 + 0,8, tauchen erneut auf. also müssen sich auch die nachfolgenden Zeilen der Umwandlung wiederholen. Es handelt sich also um einen ***periodischen*** Dualbruch. Das Umwandlungsergebnis lautet somit:

$$x_w = (65{,}675)_{10} = (1000001{,}101\ \overline{0110})_2 = (0{,}1000001101\ \overline{0110})_2 \cdot 2^7$$

Zur Normalisierung verschieben wir das Komma wieder um 7 Stellen nach links und gleichen mit dem Faktor 2^7 aus.

Bei einem periodischen Dualbruch kann die Mantisse beliebig lang sein, trotzdem werden unendlich viele Nachkommastellen abgeschnitten. Es entsteht also *immer* ein Umwandlungsfehler, schon bevor mit der Zahl gerechnet wird. Das liegt nicht am Dualsystem, denn in jedem Zahlensystem gibt es periodische Brüche mit unendlich vielen Stellen, denen endliche Brüche in anderen Zahlensystemen entsprechen können. Wir schneiden nun ab und erhalten:

$x = (0{,}10000011)_2 \cdot 2^7$

Wir stellen fest, dass dieses x den gleichen Wert hat wie das aus Aufgabe a). Mehr zu den Konsequenzen siehe Aufgabe d).

Entsprechend stimmen auch Vorzeichen und Charakteristik überein. Wieder steht also im Gleitkommaregister:

0	10000011	1110
Vz(M)	|M|	Ch

Wieder machen wir die Probe:

$x = 65{,}5$ (wie oben)

$\Delta x = x - x_W = 65{,}5 - 65{,}675 = -0{,}175$

Obwohl wir unendlich viele Bits abgeschnitten haben, also auch unendlich viele Einsen, ist der entstehende Fehler dennoch endlich und vergleichsweise klein.

c) 0,025

Diesmal haben wir als Vorkommateil Null, können uns also dessen Umwandlung sparen. Wir wandeln den Nachkommateil um:

0,025 · 2 = 0 + 0,05
0,05 · 2 = 0 + 0,1
0,1 · 2 = 0 + 0,2
0,2 · 2 = **0 + 0,4**
0,4 · 2 = **0 + 0,8**
0,8 · 2 = **1 + 0,6**
0,6 · 2 = **1 + 0,2**
0,2 · 2 = 0 + 0,4
...

Auch hier erkennen wir eine Periodizität. Es gilt:

$$x_w = (0{,}025)_{10} = (0{,}000\overline{0011})_2$$

Beim Normalisieren stoßen wir auf ein Problem: Das Komma müsste um 5 Stellen nach rechts verschoben werden, unter die Periode. Wie kann man mit so etwas umgehen? Ein kleiner Trick ist, eine der Perioden explizit hinzuschreiben. Die Periode wiederholt sich trotzdem noch unendlich oft. Dann können wir das Komma verschieben:

$$x_w = (0{,}025)_{10} = (0{,}000\overline{0011})_2 = (0{,}0000011\overline{0011})_2 = (0{,}11\overline{0011})_2 \cdot 2^{-5}$$

Zum Ausgleich für das fünfmalige Rechtsschieben, das die Mantisse um den Faktor 2^5 vergrößert, benötigen wir einen Faktor 2^{-5}, der das wieder ausgleicht.

Nun können wir die Nachkommastellen abschneiden. Wieder haben wir das Problem, dass das inmitten der Periode geschehen müsste. Erneut lösen wir das Problem dadurch, dass wir die Periode ausschreiben. Wir müssen sie allerdings zweimal ausschreiben:

$$x_w = (0{,}11\overline{0011})_2 \cdot 2^{-5} = (0{,}1100110011\overline{0011})_2 \cdot 2^{-5}$$

$\Rightarrow$ x = $(0{,}11001100)_2 \cdot 2^{-5}$

Wir berechnen die Charakteristik:

Ch = K_0 + E = 7 + (–5) = 2 = $(0010)_2$

Die Zahl steht also so im Gleitpunktregister:

0	11001100	0010
Vz(M)	|M|	Ch

Wir machen nun die Probe:

x = $(0{,}11001100)_2 \cdot 2^{2-7} = (2^{-1} + 2^{-2} + 2^{-5} + 2^{-6}) \cdot 2^{-5} = 2^{-6} + 2^{-7} + 2^{-10} + 2^{-11}$ = 0,02490234375

$\Delta x = x - x_W = -9{,}765625 \cdot 10^{-5}$ (exakt)

Man beachte: Bei der Berechnung von x sollte man möglichst alle Stellen angeben, weil sonst der Wert für Δx viel zu ungenau wird!

d) Vergleichen Sie die Ergebnisse aus Aufgabe a) und b) miteinander. Welche Probleme könnten sich daraus bei der Softwareentwicklung ergeben?

Wir können aus Aufgabe a) und b) entnehmen, dass für die hier betrachtete FPU gilt:

65,675 = 65,625 = 65,5, denn alle drei Werte werden auf denselben Gleitkommaregister-Inhalt abgebildet:

0	10000011	1110
Vz(M)	|M|	Ch

Das ist aus mathematischer Sicht offenkundig falsch. Es gibt unendlich viele Zahlen, die auf diesen Registerinhalt abgebildet werden und zwischen denen die FPU somit nicht unterscheiden kann. Bei m Mantissenbits gibt es 2^m unterschiedliche Mantissen. Nur die Zahlenwerte, die diesen entsprechen, werden genau dargestellt. Alle anderen Zahlen werden auf diese 2^m Mantissen abgebildet, wodurch ein Fehler entsteht. Man kann daher sagen, die meisten, nämlich unendlich viele Gleitkommazahlen, werden verkehrt dargestellt!

Das hat beispielsweise Auswirkungen auf Abbruchbedingungen. Man sollte Gleitkommazahlen nie auf Gleichheit testen, denn zum einen kann eine Gleichheit erkannt werden, die gar nicht gegeben ist, wie im beschriebenen Fall. Eine Anweisung

```
Falls a = = 65,5 dann ...
```

wäre auch erfüllt, wenn a den Wert 65,675 oder 65,625 hat.

Aber auch das umgekehrte kann passieren: Das Umwandlungsergebnis ist mal dicht drüber, mal dicht unter dem Vergleichswert, aber nie identisch. Daher:

> Man sollte nie auf Gleichheit von Gleitkommazahlen prüfen!

Aber auch Größenvergleiche sind kritisch. Bei folgender Abfrage

```
Falls 65,675 > 65,5 dann ...
```

wäre die Bedingung **nicht** erfüllt, obwohl sie mathematisch gesehen erfüllt wäre!

Man sollte ferner im Sinn behalten, dass es unendlich viele Zahlen gibt, die sich auf keinem üblichen Prozessor exakt darstellen lassen. Denn dazu bräuchte man wegen der Periodizität unendlich viele Mantissenbits, und das ist nicht machbar.

Denkbar wären zwar Rechenwerke, die die Darstellung von Perioden erlauben, aber auch das hätte seine Grenzen, denn es lassen sich keine Perioden beliebiger Länge in Hardware darstellen, sondern eine Obergrenze wäre vorgegeben. Außerdem wären Berechnungen damit sehr umständlich. Und es bleiben Zahlen wie π oder e, die unendlich viele Nachkommastellen haben, aber nicht periodisch sind.

Aufgabe 66 Gleitkommaaddition

Gegeben sei ein einfaches Gleitkommarechenwerk mit 1 Bit Vorzeichen Vz, 4 Bit Charakteristik Ch und 6 Bit Mantissenbetrag |M|. K_0 sei der Einfachheit halber gleich Null. Addieren Sie zum Akku den Op1 unter Verwendung folgender Inhalte:

	Vz	|M|	Ch
Ak:	0	001001	1001
Op1:	0	101011	0111

a) ohne vorher den Akkumulator zu normalisieren.

1) Charakteristikangleichung laut Tab. 14.3:

Ch(Ak) > Ch(Op1) ⇒ Ch(Op1)++, SHR |M(Op1)|

	Vz	|M|	Ch
Ak:	0	001001	1001
Op1:	0	010101	1000

Hier wird ein Bit von Op1 hinaus geschoben!

Ch(Ak) > Ch(Op1) ⇒ Ch(Op1)++, SHR |M(Op1)|

	Vz	|M|	Ch
Ak:	0	001001	1001
Op1:	0	001010	1001

Ein weiteres Bit von Op1 verschwindet!

Ch(Ak) = = Ch(Op1) ⇒ Charakteristikangleichung fertig

2) Mantissenaddition

	Vz	|M|	Ch
Ak:	0	001001	1001
+ Op1:	0	001010	1001
Ak_{neu}:	0	**0**10011	1001

3) Vorzeichenermittlung: Beide Operanden positiv, Ergebnis ebenso

4) Normalisierung

MSB(M(Ak)) = = 0 ⇒ SHL M(Ak), Ch(Ak) – –

	Vz	|M|	Ch
Ak:	0	**1**00110	1000

MSB(M(Ak)) = = 1 ⇒ Normalisierung beendet.

Probe:

Die Rechnung sollte eigentlich folgendermaßen lauten:
$Ak = (001001)_2 \cdot 2^9 = (2^{-3} + 2^{-6})_2 \cdot 2^9 = 2^6 + 2^3 = 72$
$Op1 = (101011)_2 \cdot 2^7 = (2^{-1} + 2^{-3} + 2^{-5} + 2^{-6})_2 \cdot 2^7 = 2^6 + 2^4 + 2^2 + 2^1 = 86$
$\Rightarrow$ Summe $x_W = (158)_{10} = (10011110)_2$

Wir haben erhalten:
$x = Ak = (100110)_2 \cdot 2^8 = (2^{-1} + 2^{-4} + 2^{-5})_2 \cdot 2^8 = 2^7 + 2^4 + 2^3 = 152$

Es gilt also:
$\Delta x = x - x_W = 152 - 158 = -6$

Das entspricht genau den beiden Bits von Op1, die bei der Charakteristikangleichung hinausgeschoben wurden. Sie hatten die Wertigkeiten –5 und –6 und wurden mit Charakteristik 7 multipliziert, also hatten sie den Wert $(2^{-5} + 2^{-6}) \cdot 2^7 = 2^2 + 2^1 = 4+2 = 6$.

b) mit vorheriger Normalisierung.

	Vz	|M|	Ch
Ak:	0	**0**01001	1001
Op1:	0	**1**01011	0111

1) Normalisierung: Der Kürze halber normalisieren wir beide Operanden gleichzeitig:
MSB(M(Ak)) = = 0 $\Rightarrow$ SHL M(Ak), Ch(Ak) – –
MSB(M(Op1)) = = 1 $\Rightarrow$ Op1 ist bereits normalisiert.

	Vz	|M|	Ch
Ak:	0	**0**10010	1000
Op1:	0	101011	0111

MSB(M(Ak)) = = 0 $\Rightarrow$ SHL M(Ak), Ch(Ak) – –

	Vz	|M|	Ch
Ak:	0	**1**00100	0111
Op1:	0	101011	0111

MSB(M(Ak)) = = 1 $\Rightarrow$ Normalisierung beendet.

2) Charakteristikangleichung laut Tab. 14.3:
Ch(Ak) = = Ch(Op1) $\Rightarrow$ Die Charakteristiken sind bereits angeglichen.

3) Mantissenaddition

	Vz	|M|	Ch
Ak:	0	100100	0111
+ Op1:	0	101011	0111
Ak_{neu}:	0	**1**001111	0111

Es tritt ein Übertrag auf. Sinnvollerweise wird dieser als zusätzliches Mantissenbit aufgefasst. Somit wird die Mantisse rechtsgeschoben und die Charakteristik erhöht, damit das Übertragsbit in die Mantisse passt:

	Vz	|M|	Ch
Ak:	0	**1**00111	1000

4) Vorzeichenermittlung: Beide Operanden positiv, Ergebnis ebenso

5) Normalisierung
MSB(M(Ak)) = = 1 $\Rightarrow$ Der Akku ist bereits normalisiert. Das Endergebnis lautet somit:

	Vz	|M|	Ch
Ak:	0	100111	1000

Probe:

$x = Ak = (100111)_2 \cdot 2^8 = (2^{-1} + 2^{-4} + 2^{-5} + 2^{-6})_2 \cdot 2^8 = 2^7 + 2^4 + 2^3 + 2^2 = 156$

Es gilt also:

$\Delta x = x - x_W = 156 - 158 = -2$

c) Vergleichen Sie die Ergebnisse von a) und b) im Hinblick auf ihre Genauigkeit!

Das Ergebnis von b) ist genauer als das Ergebnis von a). Bei a) wurden 2 Bits von Op1 hinausgeschoben, bei a) nur ein Bit von Ak, weil der Übertrag auftrat.

Die vorherige Normalisierung kann also das *unnötige* Hinausschieben von Bits verhindern. Normalisierung kann aber nicht in jedem Einzelfall verhindern, dass Bits verloren gehen, denn bei b) hätte man 7 Mantissenbits gebraucht, damit alle Einsen hineinpassen. Daran ändert auch vorherige Normalisierung nichts.

d) Variante: Nehmen Sie nun folgende Inhalte und rechnen Sie die Aufgabe nochmal durch. Vergleichen Sie die Auswirkungen.

	Vz	|M|	Ch
Ak:	0	001001	1001
Op1:	0	001011	0111

Wir führen die Rechnung zunächst wieder durch, ohne vorher den Akkumulator zu normalisieren.

1) Charakteristikangleichung laut Tab. 14.3:
Ch(Ak) > Ch(Op1) ⇒ Ch(Op1)++, SHR |M(Op1)|

	Vz	|M|	Ch
Ak:	0	001001	1001
Op1:	0	000101	1000

Wieder wird ein Bit von Op1 hinaus geschoben!

Ch(Ak) > Ch(Op1) ⇒ Ch(Op1)++, SHR |M(Op1)|

	Vz	|M|	Ch
Ak:	0	001001	1001
Op1:	0	000010	1001

Ein weiteres Bit von Op1 verschwindet!
Ch(Ak) = = Ch(Op1) ⇒ Charakteristikangleichung fertig

2) Mantissenaddition

	Vz	|M|	Ch
Ak:	0	001001	1001
+ Op1:	0	000010	1001
Ak_{neu}:	0	**0**01011	1001

3) Vorzeichenermittlung: Beide Operanden positiv, Ergebnis ebenso
4) Normalisierung
MSB(M(Ak)) = = 0 ⇒ SHL M(Ak), Ch(Ak) – –

	Vz	|M|	Ch
Ak:	0	**0**10110	1000

MSB(M(Ak)) = = 0 ⇒ SHL M(Ak), Ch(Ak) – –

	Vz	|M|	Ch
Ak:	0	**1**01100	0111

MSB(M(Ak)) = = 1 ⇒ Normalisierung beendet.

Probe:

Die Rechnung sollte eigentlich folgendermaßen lauten:
$Ak = (001001)_2 \cdot 2^9 = (2^{-3} + 2^{-6})_2 \cdot 2^9 = 2^6 + 2^3 = 72{,}0$
$Op1 = (001011)_2 \cdot 2^7 = (2^{-3} + 2^{-5} + 2^{-6})_2 \cdot 2^7 = 2^4 + 2^2 + 2^1 = 22{,}0$
$\Rightarrow$ Summe $x_W = (94)_{10} = (1011110)_2$

Wir haben erhalten:
$x = Ak = (101100)_2 \cdot 2^7 = (2^{-1} + 2^{-3} + 2^{-4})_2 \cdot 2^7 = 2^6 + 2^4 + 2^3 = 88$

Es gilt also:
$\Delta x = x - x_W = 88 - 94 = -6$

Das entspricht genau den beiden Bits von Op1, die bei der Charakteristikangleichung hinausgeschoben wurden. Sie hatten die Wertigkeiten –5 und –6 und wurden mit Charakteristik 7 multipliziert, also hatten sie den Wert $(2^{-5} + 2^{-6}) \cdot 2^7 = 2^2 + 2^1 = 4+2 = 6$.

Bei dem Schritt, der vorher der Aufgabe b) entsprach, wird nun zuerst normalisiert, bevor wir die Rechnung durchführen:

	Vz	|M|	Ch
Ak:	0	**0**01001	1001
Op1:	0	001011	0111

1) Wir normalisieren zunächst nur den Akku.
MSB(M(Ak)) = = 0 ⇒ SHL M(Ak), Ch(Ak) – –

	Vz	|M|	Ch
Ak:	0	**0**10010	1000
Op1:	0	001011	0111

MSB(M(Ak)) = = 0 ⇒ SHL M(Ak), Ch(Ak) – –

	Vz	|M|	Ch
Ak:	0	**1**00100	0111
Op1:	0	001011	0111

MSB(M(Ak)) = = 1 ⇒ Die Normalisierung des Akkus ist nun beendet. Wir werden gleich sehen, dass es nichts bringen würde, den Op1 auch noch zu normalisieren.

2) Charakteristikangleichung
Es gilt: Ch(Ak) = = Ch(Op1), die Charakteristiken sind also bereits angeglichen.
Den Op1 im vorigen Schritt auch noch zu normalisieren, hätte also nichts gebracht, denn dadurch wären die Charakteristiken wieder ungleich geworden. Die dann nötige Charakteristikangleichung hätte dies einfach wieder rückgängig gemacht.

3) Mantissenaddition

	Vz	|M|	Ch
Ak:	0	100100	0111
+ Op1:	0	001011	0111
Ak_{neu}:	0	**1**01111	0111

4) Vorzeichenermittlung: Beide Operanden positiv, Ergebnis ebenso
5) Normalisierung
MSB(M(Ak)) = = 1 $\Rightarrow$ Der Akku ist bereits normalisiert. Das Endergebnis lautet somit:

	Vz	|M|	Ch
Ak:	0	101111	0111

Probe:

$x = Ak = (101111)_2 \cdot 2^7 = (2^{-1} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6})_2 \cdot 2^7 = 2^6 + 2^4 + 2^3 + 2^2 + 2^1 = 94$

Das ist das exakte Ergebnis! Wir hatten ja auch keine Bits hinaus geschoben. Es gilt also:

$\Delta x = x - x_W = 94 - 94 = 0$

Vergleichen Sie die Ergebnisse im Hinblick auf ihre Genauigkeit!

Diesmal konnten wir durch die Normalisierung sogar verhindern, dass überhaupt ein Fehler aufgetreten ist. Das ist der Idealfall.

Aufgabe 67 Installieren Sie einen Disassembler mit HexDump-Funktionalität. Sehen Sie sich eine Binärdatei damit an.
Anmerkung: Bitte beachten Sie, dass manche Software-Lizenzvereinbarungen das Disassemblieren der Software verbieten. Analysieren Sie also am besten eine selbst erzeugte Binärdatei.

Beispiele für geeignete Disassembler sind IDA Pro – Free, DDD, Kdbg, objdump, etc.

Aufgabe 68 Ein Maschinenbefehl, der zuvor 5 Takte bzw. Mikrobefehle benötigt hat, wird durch eine Optimierung auf 3 Takte verbessert. Um wieviel Prozent verbessert sich dadurch seine Performance?
Die Taktfrequenz des Prozessors betrage 2,6 GHz. Welche Taktfrequenz würde man benötigen, wenn man durch Erhöhung der Taktfrequenz denselben Effekt erzielen möchte?

Die Performance verbessert sich um (5–3)/5 = 40%.

Man müsste die Taktfrequenz um ebenfalls 40% erhöhen, also um 0,4 · 2,6 GHz = 1,04 GHz auf 3,64 GHz.

Weil sich eine Optimierung eines einzelnen Maschinenbefehls nur anteilig auswirkt, ist diese Angabe aber nur sinnvoll, wenn alle Befehle des Befehlssatzes entsprechend verbessert werden.

Aufgabe 69 Welchen Inhalt muss also das Register in diesem Moment noch haben?

Der Adressmultiplexer leitet den Opcode zum Ausgang weiter. Also muss der Multiplexer an seinem Eingang A/~~B~~ eine Null haben, damit B zum Ausgang durchgeschaltet wird.

Ein OR-Gatter hat genau dann eine 0 am Ausgang, wenn alle Eingänge 0 sind. Somit muss eine $(0000)_2$ im Register befinden. Sie stellt die Adresse des letzten Mikrobefehls des vorigen Maschinenbefehls dar, welche bei unserer Mikroprogrammsteuerung immer Null sein muss.

Aufgabe 70 Warum teilen sich alle Mikroprogramme die Adresse 0?

Weil ausschließlich die Adresse 0 als Endekennung zum Umschalten des Multiplexers verwendet wird. Weil jedes Mikroprogramm endet, wird schließlich irgendwann von jedem Mikroprogramm die Adresse 0 erreicht.

Aufgabe 71 Könnte man mehrere unterschiedliche Adressen als Endekennung verwenden? Wenn ja, wie müsste man die Mikroprogrammsteuerung modifizieren, um das zu erreichen?

Ja. Man könnte z.B. die Adressen 0 und 1 als Endekennungen nehmen, mit jeweils unterschiedlichen aktivierten Steuersignalen.

Man bräuchte lediglich das niederwertigste Bit des Registers nicht auf das OR-Gatter zu führen. Dann würde es sowohl bei $(0000)_2$ als auch bei $(0001)_2$ auf den neuen Opcode umschalten. Würde man beide niederwertigsten Bits vom OR-Gatter trennen, hätte man 4 Endekennungen, nämlich die Adressen $(0000)_2$ bis $(0011)_2$.

Aufgabe 72 Welche Maschinenbefehle sind außer dem betrachteten in der obigen Mikroprogrammsteuerung enthalten? Geben Sie die Opcodes, die der Reihe nach jeweils durchlaufenen Mikroprogrammadressen und die aktivierten Steuersignale an!

Am besten schaut man nach bisher nicht verwendeten Zeilen mit der Folgeadresse 0. Das könnten letzte Zeilen eines Mikroprogrammes sein, allerdings auch unbelegte Zeilen.

Dann arbeitet man sich „rückwärts“ bis zur Startadresse des Mikroprogrammes vor. Die ist der Opcode des Maschinenbefehls. Man kann sie daran erkennen, dass kein Mikrobefehl sie als Folgeadresse hat.

Aus dem Beispiel wissen wir bereits, dass die Adressfolge 6 – 12 – 5 – 7 – 0 enthalten ist. In unserer Mikroprogrammsteuerung finden wir ferner die Adressfolgen:

2 – 1 – 3 – 8 – 0
10 – 14 – 4 – 0
13 – 0

15 – 9 – 11 – 0

Allerdings wäre es auch möglich, dass ein Mikroprogramm vollständig in einem anderen enthalten ist. Das könnte man auf diese Weise nicht entdecken. Z.B. träfe das auf einen möglicherweise existierenden Opcode 5 mit Adressfolge 5 – 7 – 0 zu.

Zu den neu gefundenen Adressfolgen gehören folgende Mikroprogramme:

a) Opcode 2:

Schritt	µPrg-Adresse	Aktivierte Steuersignale	Folgeadresse
1	2	A, H, L	1
2	1	E, K	3
3	3	D	8
4	8	D, G, J	0
5	0	X	(beliebig)

b) Opcode 10:

Schritt	µPrg-Adresse	Aktivierte Steuersignale	Folgeadresse
1	10	B, F, K	14
2	14	G	4
3	4	F, H, K	0
4	0	X	(beliebig)

c) Opcode 13:

Schritt	µPrg-Adresse	Aktivierte Steuersignale	Folgeadresse
1	13	C, E, J	0
2	0	X	(beliebig)

d) Opcode 15:

Schritt	µPrg-Adresse	Aktivierte Steuersignale	Folgeadresse
1	15	A, E, H	9
2	9	F	11
3	11	D, L	0
4	0	X	0 (beliebig)

Aufgabe 73 In den Zeilen 7 und 12 wird jeweils nur das Steuersignal I aktiviert. Beide Mikrobefehle bewirken also dasselbe. Könnte man die beiden Zeilen somit zu einer zusammenfassen?

Nein, denn die Folgeadressen stimmen nicht überein. Die Zeilen sind also nicht völlig identisch. Falls auch die Folgeadressen dieselben wären, könnte man die Zeilen zusammenfassen.

Aufgabe 74 Mikroprogrammsteuerung

Es soll eine Mikroprogrammsteuerung konzipiert werden, die beim Maschinenbefehl mit dem Opcode 2 die angegebene Folge von Steuersignalen erzeugt. Skizzieren Sie eine Mikroprogrammsteuerung mit 16 Mikroprogrammadressen, die diese Aufgabe löst.

Tab. A.8: Folge von Steuersignalen einer Mikroprogrammsteuerung

Schritt Nr.	Adresse	aktivierte Steuersignale	Folgeadresse
1	?	D, F, G	1
2	1	B, L	6
3	6	A, D, E, F	5
4	5	B, C	0
5	?	A, H, I	0 (beliebig)

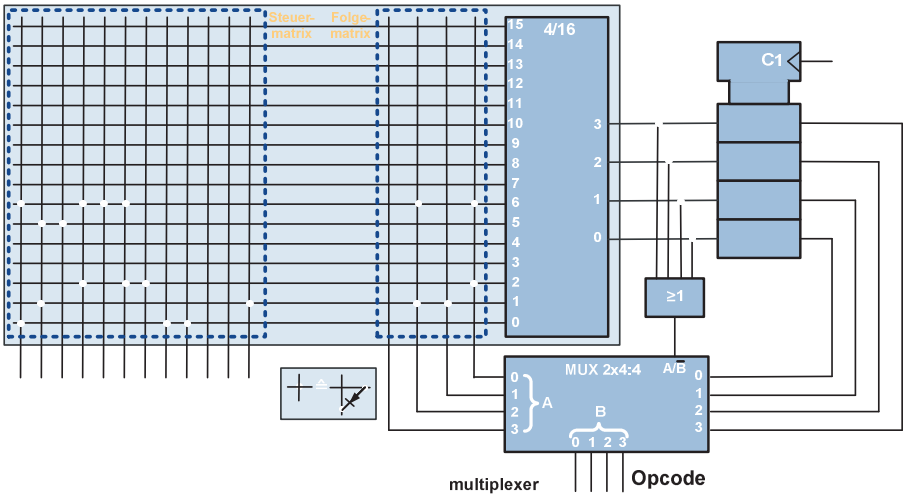

Abb. A.11: Programmierte Mikroprogrammsteuerung

Aufgabe 75 Wie viele Spalten benötigt man bei 50 000 real vorkommenden Signalkombinationen in der Steuermatrix, wenn man eine Lookup Table einsetzt? Wie viele Spalten spart man gegenüber einer Funktionalbitsteuerung mit 500 Steuersignalen ein?

Man bildet den Zweierlogarithmus, also ld 50 000, was aufgerundet 16 ergibt. Alternativ sieht man, dass 50 000 zwischen den Zweierpotenzen $2^{15} = 32768$ und $2^{16} = 65536$ liegt. Man benötigt also 16 Bits bzw. Spalten, um die Signalkombinationen durchzunummerieren.

Gegenüber den 500 Spalten von zuvor spart man 500 – 16 = 484 ein, das sind 484/500 = 96,8%.

Aufgabe 76 Wie kann man zwei Maschinenbefehle zu einer gemeinsamen Abarbeitung ihrer Mikrobefehle zusammenfassen?

In beiden Mikroprogrammen sieht man einen Mikrobefehl mit derselben Folgeadresse f vor. Ab dieser Mikrobefehlsadresse nutzen beide Mikroprogramme dieselben Zeilen der Koppelmatrix bzw. Mikrobefehlsadressen.

Aufgabe 77 Wie würde man vorgehen, wenn man vier ähnliche Maschinenbefehle teilweise zusammenfassen will, z.B. MOV AX/BX/CX/DX, []?

Das Zusammenfassen erfolgt wieder über eine gemeinsame Folgeadresse. Die Aufsplittung kann man anhand von zwei Opcode-Bits vornehmen, z.B.

Opcode-LSBs	Operanden-Ziel
00	AX
01	BX
10	CX
11	DX

Aufgabe 78 Ein Prozessor besitze 500 unterschiedliche Maschinenbefehle in seinem Befehlssatz. Wie viele Bits müsste ein Opcode umfassen? Wie könnte man vorteilhaft vorgehen, wenn der Prozessor nur Zweierpotenzen als Bitanzahl für die Opcodes verwenden kann? Welche Wortbreite müssen die Mikroprogrammadressen mindestens besitzen, wenn ein Maschinenbefehl im Schnitt 8 Mikrobefehle umfasst? Welche Wortbreite benötigt man für die im Mikroprogrammspeicher abzulegenden Daten, wenn es 30.000 verschiedene Steuersignalkombinationen gibt?

ld 500 ergibt aufgerundet 9. Ein Opcode benötigt also 9 Bits. Das passt gerade nicht mehr in ein 8-Bit-Wort. Man könnte zwar ausschließlich 16-Bit-Opcodes verwenden, aber das wäre nicht allzu effizient. Daher würde man vorteilhaft so vorgehen:

Für häufig benötigte Opcodes verwendet man 8-Bit-Worte. Damit kann man 255 Opcodes abdecken, z.B. von 1 bis 255. Wählt man als Opcode-Byte die 0, dann ist das das Zeichen für den Prozessor, dass ein weiteres Opcode-Byte folgt. Somit werden die restlichen 500 – 255 = 245 Opcodes, und nur diese, 2 Bytes lang werden. Diese verwendet man für seltener benötigte Befehle.

Das ist ein ähnliches Verfahren wie bei den Telefonnummern. Man könnte alle Vorwahlen gleich lang machen, aber besser ist es, die häufig verwendeten Vorwahlen großer Städte, z.B. München, kurz zu halten (in diesem Beispiel 089). Im Gegenzug werden die Vorwahlen kleinerer Orte, zu denen weniger oft Verbindungen aufgebaut werden, als längere Ziffernfolgen konzipiert.

Der Prozessor hat 500 Maschinenbefehle zu je 8 Mikrobefehlen im Befehlssatz. Also umfasst die Mikroprogrammsteuerung $500 \cdot 8 = 4000$ Mikrobefehle bzw. Mikroprogrammadressen. Die nächsthöhere Zweierpotenz ist $2^{12} = 4096$. Somit werden 12 Bit-Mikroprogrammadressen benötigt.

Für die 30000 Steuersignalkombinationen benötigt man 15 Bits, da die nächsthöhere Zweierpotenz $2^{15} = 32768$ ist. Außer den Steuersignalen werden im Mikroprogrammspeicher auch noch die Folgeadressen abgelegt, die wie die Mikroprogrammadressen 12 Bit lang sind. An jeder Mikroprogrammadresse muss man somit 15 + 12 = 27 Bit ablegen.

Bei dem Mikrobefehlsspeicher handelt es sich somit um einen 4 kBit x 27 – Speicher. Er hat also insgesamt 4096 Adressen (aufgerundet), an denen sich jeweils ein 27-Bit-Datenwort befindet.

Aufgabe 79 Stellen Sie die wesentlichen Vor- und Nachteile der Mikroprogrammierung zusammen.

Tab. A.9: Vor- und Nachteile der Mikroprogrammierung

Vorteile der Mikroprogrammierung	Nachteile
Einheitliche Vorgehensweise Dadurch einfacherer Entwurf von Prozessoren und geringere Entwicklungskosten Hohe Flexibilität Bei Einsatz von wiederbeschreibbaren Speichern einfache Fehlerbeseitigung.	langsam im Vergleich zum Schaltwerk

Aufgabe 80 Woher weiß der Prozessor, um welchen Wert der Befehlszeiger erhöht werden muss? Reicht einmaliges Erhöhen in jedem Falle aus? Begründung!

Anhand des Opcodes ist erkennbar, um wie viele Bytes der Befehlszeiger zu verändern ist, weil jeder Maschinenbefehl eine charakteristische Länge besitzt.

Werden Daten aus dem Hauptspeicher geholt oder Sprungbefehle ausgeführt, wird der Befehlszeiger in einer späteren Phase des Befehlszyklus nötigenfalls nochmals verändert.

Aufgabe 81 Nennen Sie Beispiele für Befehle, bei denen nicht alle Aktionen, die in einem Befehlszyklus möglich sind, benötigt werden.

Befehle, die den Prozessor in einen anderen Betriebsmodus versetzen: Z.B. BCD-Rechenmodus statt Dualzahl-Rechenmodus

a) Befehle, die nur auf Registern arbeiten, z.B. Schiebe- und Rotierbefehle, benötigen keine Adressberechnung und kein Holen von Daten aus dem Speicher

b) Der NOP-Befehl (No Operation) hat keine Ausführungsphase im eigentlichen Sinn, sondern macht lediglich 1 Takt lang nichts.

Aufgabe 82 Was könnte ein Hinderungsgrund dafür sein, das Predict Taken-Verfahren einzusetzen?

Beim Predict Taken-Verfahren muss die Zieladresse, ab der die Befehle geholt werden, erst errechnet werden. Also muss man die eigentlich erst später erfolgende Address Generation

vorziehen. Die Berechnung dauert eine gewisse Zeit, während der die Pipeline leerläuft. Außerdem kann die für die Adressberechnung nötige Komponente durch einen anderen Befehl belegt sein, so dass weitere Verzögerungen entstehen. Ein schritthaltendes Füllen der Pipeline wie beim Predict not Taken-Verfahren ist also schwer zu realisieren.

Aufgabe 83 Wie beurteilen Sie die Strategie, Pipelines sehr lang zu machen, z.B. 20 oder 30 Stufen, damit man den Prozessor höher takten kann?

Das mag aus Marketing-Sicht vorteilhaft sein, weil sich Prozessoren mit höheren Taktfrequenzen besser verkaufen lassen. Allerdings wird die positive Auswirkung der höheren Taktfrequenz auf die Performance dadurch weitgehend wieder zunichte gemacht, dass beim Leerlaufen der Pipelines viel mehr Wartezyklen nötig sind als bei optimaler Pipelinelänge.

Konkretes Beispiel: Der Intel Pentium IV Northwood hatte eine 21-stufige Integer-Pipeline, 8k L1-Cache und 512k L2-Cache. Er bestand aus 55 Millionen Transistoren und hatte eine Leistungsaufnahme von etwa 70W bei 3,2 GHz.

Um noch höhere Taktfrequenzen zu erreichen, wurde sein Nachfolger, der Intel Pentium IV Prescott, mit einer 31-stufigen Integer-Pipeline versehen. Obwohl man L1- und L2-Cache verdoppelte und unter anderem deswegen und wegen weiterer Optimierungen 125 Millionen Transistoren benötigte, hatte er bei derselben Taktfrequenz etwa dieselbe Performance wie sein Vorgänger. Allerdings brauchte er etwa 100W.

Bald darauf setzte Intel statt auf immer höhere Taktfrequenzen auf eine idealere Pipelinelänge, wie das vorher bereits beim Pentium M für Notebooks gemacht wurde. Neuere Intel-Architekturen basieren fortan auf diesem Ansatz.

Aufgabe 84 Warum enthält die BHT typischerweise doppelt so viele Einträge wie der BTB?

Die BHT enthält einen Eintrag für jeden bedingten Verzweigungsbefehl, egal ob die Verzweigung erfolgte oder nicht. Der BTB bekommt nur einen Eintrag, falls eine Verzweigung erfolgte. Weil das im statistischen Mittel bei der Hälfte der Fälle so ist, genügt es, nur halb so viele Einträge wie bei der BHT vorzusehen bzw. die BHT enthält doppelt so viele Einträge wie der BTB.

Aufgabe 85 Wie kommt man auf die 4 GB Hauptspeicher bei 32-Bit-Adressen?

2^{32} Bytes = $2^2 \cdot 2^{10} \cdot 2^{10} \cdot 2^{10}$ Bytes

Wir haben drei Faktoren von $2^{10} = 1024$, also drei Einheitensprünge: Kilo – Mega – Giga. Davor steht noch die $2^2 = 4$. Somit kommt man auf 4 GByte, die sich mit 32 Bit adressieren lassen.

Aufgabe 86 Forschen Sie nach, welche Informationen sich über aktuelle Prozessoren verschiedener Hersteller finden lassen. Wie ist deren interner Aufbau? Wie viele Ausführungseinheiten hat der Prozessor? Von welcher Art? Wie groß

sind die prozessorinternen Caches? Was lässt sich über die in diesem Kapitel behandelten Features bei dem Prozessor finden?

Die gesuchten Informationen finden sich auf den Websites der Prozessorhersteller, allerdings oft versteckt. Stichworte: Developers Guide, Datasheet.

Schneller führt oft eine Suchmaschinenanfrage zum Ziel. Stichwörter z.B. core i7 architecture. Oft enthält bereits die Wikipedia genügend Informationen.

Ferner sind in Zeitschriften wie der c't regelmäßig Informationen über neu erschienene und geplante Prozessoren und deren internen Aufbau enthalten.

Aufgabe 87 Welche Strecke legt das Licht bei einer Taktfrequenz von 3 GHz innerhalb einer Taktperiode zurück? Welche Schlussfolgerungen kann man daraus für elektrische Signale in einem Computer ziehen?

Die Vakuumlichtgeschwindigkeit beträgt etwa 300 000 km/s = $3 \cdot 10^8$ m/s.

Die Dauer einer Taktperiode bei 3 GHz beträgt $1/(3 \cdot 10^9)$ s = 0,33 ns = $3{,}3 \cdot 10^{-10}$ s.

Es gilt v = s/t. Die Strecke s beträgt somit s = v t = $3 \cdot 10^8$ m/s $\cdot$ $3{,}3 \cdot 10^{-10}$ s = 0,01m = 10 cm

Leitergebundene elektromagnetische Wellen breiten sich langsamer aus, so dass man auf Strecken in der Größenordnung von 5 bis 8 cm kommt. Das begrenzt die Abmessungen, innerhalb derer solche hohen Taktfrequenzen noch sinnvoll angewendet werden können. Innerhalb eines Prozessorchips bleibt man zwar im Rahmen dieser Distanz, aber bei Komponenten auf dem Mainboard kann es bereits problematisch werden. Außerdem sieht man, dass selbst innerhalb eines Prozessorchips kein allzu großer Spielraum für Taktfrequenzsteigerungen mehr bleibt.

Aufgabe 88 Ein Prozessorchip habe die Abmessungen 3cm X 2cm und gebe eine Leistung von 90 W ab. Welche Leistung pro Fläche erzeugt er (Angabe in Watt pro cm²)? Vergleichen Sie dies mit der Leistungsabgabe einer Schnellkochplatte mit 2kW Leistung und 20 cm Durchmesser!

Prozessor:
$P/A_P = 90\ W/6\ cm^2 = 15\ W/cm^2$

Schnellkochplatte:
$A_S = \pi d^2/4 \approx 314\ cm^2$
$P/A_S = 2000\ W/314\ cm^2 = 6{,}37\ W/cm^2$

Das bedeutet, der Prozessor heizt mehr als doppelt so gut wie eine Schnellkochplatte!

Bedenkt man, dass höhere Taktfrequenz zu größerer Verlustleistung führt, erkennt man, dass das Problem der Kühlung die mögliche Taktfrequenz begrenzt.

Aufgabe 89 Wir betrachten zwei Alternativen. Die erste ist ein System mit einem Prozessor, der mit 3 GHz getaktet wird. Die zweite Alternative ist ein System mit drei Prozessoren, die zum vorher genannten baugleich sind, aber mit jeweils 1 GHz Taktfrequenz getaktet werden. Welches ist die schnellere der beiden Alternativen?

Erstaunlicherweise meinen die meisten befragten Studierenden regelmäßig, Alternative 2 sei die schnellere. Aber wie so oft liegt die Mehrheit falsch: Alternative 1 ist schneller.

Wenn man mehrere Prozessoren einsetzt, benötigt man erheblichen Zusatzaufwand, den man bei nur einem Prozessor nicht hat. Worin der Zusatzaufwand besteht, wird im Unterkapitel „Zusatzaufwand bei der Software“ betrachtet.

Weil die Zahl der Taktzyklen pro Sekunde bei beiden Systemen insgesamt gleich ist, bleiben bei Alternative 2 weniger Taktzyklen für die Erledigung der eigentlichen Aufgaben übrig als bei Alternative 1. Somit ist Alternative 1 die schnellere von beiden.

Aufgabe 90 Welchen Grad besitzen 3D-Cube, 2D-Gitter und Stern?

3D-Cube: Grad 3

2D-Gitter: Grad 4

Stern: Grad n–1 bei n Knoten

Aufgabe 91 Wie groß ist der Durchmesser der Ring-Topologie aus Abb. 19.2?

Der Durchmesser beträgt 3 (Man betrachte zwei gegenüberliegende Knoten)

Aufgabe 92 Welche Dimensionalität besitzt ein kompletter Zusammenschluss mit 3 Knoten?

2 (Man kann genau 2 Pfade von einem beliebigen Knoten zu einem anderen finden)

Aufgabe 93 Werfen Sie einen Blick in die Handbücher von Programmiersprachen und Spracherweiterungen für Multiprozessorsysteme. Wo liegen die Unterschiede und Besonderheiten im Vergleich zu herkömmlichen Sprachen? Wozu benötigt man diese?

Siehe die im Kapitel angegebenen URLs.

Aufgabe 94 Wie wirken sich die Konsistenzprobleme auf die Effizienz von Multiprozessorsystemen im Vergleich zu Einprozessorsystemen aus?

Bei Einprozessorsystemen können strikte Konsistenz und Caching miteinander gut vereinbart werden, ohne dass großer Zusatzaufwand nötig wäre.

Bei Multiprozessorsystemen ist zur Wahrung der Konsistenz ein teils beträchtlicher Zusatzaufwand nötig, der sich aus der Synchronisation von Cache-Einträgen ergibt (Stichwort: Cache-Kohärenz-Protokolle).

Dennoch erreicht man keine strikte Konsistenz, sondern nur schwächere Formen der Konsistenz. Diese äußern sich seitens des Programmierers darin, dass an geeigneten Stellen immer wieder explizit für die Konsistenz gesorgt werden muss, was bei strikter Konsistenz oft entfallen kann.

Insgesamt erzeugen die Konsistenzprobleme somit einen deutlichen Mehraufwand für Multiprozessorsysteme im Vergleich zu Einprozessorsystemen.

Aufgabe 95 Forschen Sie nach, welche Arten von A/D-Wandlern es gibt und wie sie arbeiten. Welche zeichnen sich durch besonders hohe Geschwindigkeit aus?

Es gibt beträchtliche Unterschiede in der Geschwindigkeit, in der die Wandlung durchgeführt wird. Je höher die Auflösung, desto langsamer erfolgt in der Regel die Umsetzung. Es gibt jedoch auch sehr schnelle Wandler mit hoher Auflösung, die aber einen immensen Schaltungsaufwand sowie Präzisionsbauelemente erfordern und somit sehr teuer sind (***Flash-ADC***). Genauso gibt es langsame und trotzdem relativ teure Wandler (***Dual-Slope-Wandler***).

Einen guten Kompromiss zwischen Geschwindigkeit und Kosten bieten Wandler, die nach der ***sukzessiven Approximation*** arbeiten. Dabei wird der analoge Eingangsspannungswert V_{in} dauernd mit einer im Wandler erzeugten Spannung verglichen, die dem bisherigen Wandlungsergebnis entspricht.

Man beginnt mit einer konstanten, eingebauten Referenzspannung U_{ref} , die dem halben Maximalwert der Eingangsspannung entspricht. Befindet sich V_{in} darüber, wird das führende Bit des Ergebnisses 1, ansonsten wird es 0.

Als nächstes erhöht bzw. vermindert man U_{ref} um die Hälfte, vergleicht wieder und bestimmt so das zweithöchste Bit. Beim n-ten Durchgang wird ein 2^n-tel von U_{ref} addiert oder subtrahiert. So nähert man sich Bit für Bit dem Wert der Eingangsspannung an. Bei n Bit Auflösung benötigt man n Durchgänge.

Aufgabe 96 Was versteht man unter µ-Law? Wozu setzt man es ein?

Da die Auflösung der A/D-Wandler begrenzt ist, entsteht bei der Umwandlung einer analogen in eine digitale Größe stets ein Quantisierungsfehler. Der Quantisierungsfehler lässt sich mit gewissen Tricks subjektiv verringern: Das Gehör empfindet z.B. Lautstärken, die zehnmal so groß sind, nur als doppelt so laut. Ein Fehler, der konstant 3 mV groß ist, wirkt also am störendsten bei kleinen Signal-Amplituden. Man macht daher bei ***µ-Law*** die Quantisierungsstufen umso größer, je größer das Signal ist.

Aufgabe 97 Kann Aliasing auch dann auftreten, wenn nicht abgetastet wird? Begründung!

Nein. Aliasing setzt voraus, dass sich das Spektrum periodisch wiederholt. Das ist typisch für abgetastete Signale. Außerdem ist ohne Abtastung keine Abtastfrequenz angebbar, die einzuhalten wäre.

Aufgabe 98 Sehen Sie sich an, wie ein FFT-Algorithmus funktioniert.

z.B. Cooley–Tukey-Algorithmus

Aufgabe 99 Wie kann man bemerken, wie gut die Leitungseigenschaften bei einem DSL-Anschluss sind?

Am einfachsten kann man das anhand der Datentransferraten feststellen. Je höher der Frequenzbereich, desto größer die Gefahr, dass Leitungsdämpfungen den Datentransfer behindern. Daher wird der Downstream eher beeinträchtigt als der Upstream. Hier sind aber auch die Reserven in Form verfügbarer Kanäle am größten.

Aufgabe 100 Informieren Sie sich über Features und internen Aufbau von aktuellen DSP!

Zu finden auf den Websites der DSP-Hersteller, z.B. Texas Instruments:

http://www.ti.com/lsds/ti/dsp/platform/c6-integra/device.page

Aufgabe 101 Woran könnte es liegen, dass man RAM nicht generell durch Flash-Speicher ersetzt? Das würde das Booten überflüssig machen.

Obwohl Flash-Speicher relativ schnell ist, reicht seine Geschwindigkeit bei weitem nicht an die des RAM heran. Man nimmt daher die Zeit für das Booten in Kauf, um danach umso schneller mit dem Rechner arbeiten zu können.

Aufgabe 102 Forschen Sie nach, welche Speicherkapazitäten und Zugriffszeiten für SRAM und DRAM aktuell erreichbar sind.

Fundstellen sind z.B. Hersteller von Speicherbausteinen wie Infineon.

Aufgabe 103 Welche Adressdecoder benötigt man, wenn man bei 28 Adresssignalen eine dreidimensionale Anordnung wählt? Mit wie vielen Adressdecoder-Ausgängen kommt man nun insgesamt aus?

28/3 = 9 (abgerundet). Man könnte z.B. zwei 9:512 und einen 10:1024-Adressdecoder nehmen, um auf insgesamt 28 Eingänge zu kommen.

Insgesamt benötigt man somit 2 512 + 1024 = 2048 = 2^{11} Adressdecoder-Ausgänge. Das ist um den Faktor $2^{11}/2^{28} = 2^{-17}$ = 1/131072 weniger Bedarf als in der eindimensionalen Darstellung.

Aufgabe 104 Welche Speicherkapazität in MBit besitzt der Speicherbaustein in Abb. 21.5?

Der Speicherbaustein besitzt 28 Adresseingänge. Damit lassen sich $2^{28} = 2^{10}$ 2^{10} 2^{8} = 256M Adressen ansprechen. M steht dabei für 1024^2.

An jeder Adresse wird genau 1 Bit gespeichert. Somit besitzt der Speicherbaustein eine Speicherkapazität von 256 MBit.

Aufgabe 105 Ein Speicherbaustein wird mit DDR2-1066 CL5-5-5-15 bezeichnet. Welche Latenzzeiten in Nanosekunden kann man daraus ableiten?

DDR2 überträgt 2 Datenworte pro Taktflanke und nutzt beide Taktflanken. Um die Speicher-Taktfrequenz zu errechnen, muss man somit die 1066 durch 4 teilen. Der Speichertakt beträgt somit 1066/(2 · 2) MHz = 266 MHz. Eine Taktperiode dauert somit 1/266 MHz = 3,75 ns.

Bei DDR2 ist der I/O-Takt doppelt so hoch wie der Speichertakt, also 266 MHz · 2 = 533 MHz. Das entspricht einer Taktperiodendauer von 3,75 ns/2 = 1,875 ns. Als Vielfaches dieser Periodendauer werden die obigen Werte angegeben. 5 Takte dauern 5 · 1,875 ns = 9,375 ns. Somit beträgt

$$CL = t_{RCD} = t_{RP} = 9{,}375 \text{ ns}.$$

Die vierte Größe t_{RAS} hat eine Dauer von 15 I/O-Takten, also t_{RAS} = 15 · 1,875 ns = 28,125 ns.

Allgemein kann man eine Latenzzeit T bei DDR2-x also nach folgender Formel berechnen:

$$T = (2 \cdot n)/x \text{ [ns]},$$

wenn n die Anzahl der Takte ist, die der Latenzzeit entsprechen.

Aufgabe 106 Berechnen Sie, mit welchem Speichertakt und I/O-Takt folgende Speichermodule betrieben werden. Geben Sie ferner – wo sinnvoll – die FSBnnnn-Angabe an, falls zwei Kanäle zum Einsatz kommen.

a) PC2-6400
b) PC3-17000

Zu a):

Die Wortbreite beträgt 64 Bit = 8 Byte. Daher kommt man auf 6400 M/8 = 800 M Datentransfers pro Sekunde. Das Modul besteht also aus DDR2-800-Speicherbausteinen.

Es werden 2 Datenworte pro Taktperiode übertragen, so dass der I/O-Takt 800 MHz/2 = 400 MHz beträgt. DDR2-SDRAM liefert 4 Spalten bzw. Datenworte pro I/O-Takt, so dass der Speichertakt 800 MHz/4 = 200 MHz beträgt.

Bei zwei Kanälen verdoppelt sich der Datendurchsatz bzw. die Zahl übertragener Datenworte pro Sekunde von 800 M auf 1600 M, so dass man die Bezeichnung FSB1600 wählen würde.

Zu b):

Es werden 17000 M/8 = 2125 M Datentransfers pro Sekunde durchgeführt. Das rundet man auf den nächsten gängigen Wert, also 2133 M auf. Das Modul besteht somit aus DDR3-2133-Speicherbausteinen.

Es werden 2 Datenworte pro Taktperiode übertragen, so dass der I/O-Takt 2133 MHz/2 = 1083 MHz beträgt. Auch hier tritt wieder ein Rundungsfehler auf. Eigentlich handelt es sich um 1066 MHz.

DDR3-SDRAM liefert 8 Spalten bzw. Datenworte pro I/O-Takt, so dass der Speichertakt 2133 MHz/8 = 266 MHz beträgt.

DDR3-SDRAM wird nicht in Verbindung mit FSB eingesetzt, so dass sich eine derartige Angabe erübrigt.

Aufgabe 107 Welchen Unterschied könnte es machen, ob man eine einzelne Datei oder eine ganze Partition für die Auslagerung verwendet?

Eine Auslagerungsdatei liegt in einem konventionellen Dateisystem, das nicht auf die Auslagerung optimiert ist. Verzeichnisbäume, Zugriffsrechte und ähnliches werden für die Auslagerung nicht benötigt und können sich evtl. störend oder bremsend auswirken. Eine Auslagerungsdatei darf nicht versehentlich gelöscht oder verändert werden. Dieses Risiko ist größer, wenn sie wie alle anderen Dateien behandelt wird.

Dagegen kann eine separate Partition ein Dateisystem verwenden, das auf die Auslagerung optimiert ist und bessere Performance bietet. Auf die enthaltenen Daten kann man nicht ohne weiteres zugreifen, was einen besseren Schutz vor versehentlichen Änderungen bietet.

Aufgabe 108 Warum sollte man als Programmierer dafür sorgen, dass dynamisch angeforderter Speicher immer freigegeben wird, sobald man ihn nicht mehr benötigt?

Der Speicher würde sonst unnötigerweise Platz auf dem Heap belegen und Fragmentierungsprobleme verstärken. Außerdem würde er nach einiger Zeit der Inaktivität ausgelagert werden und Performance und Plattenplatz benötigen.

Es gibt zwar Automatismen in Compilern und Betriebssystemen, die selbsttätig für eine Freigabe sorgen, aber man kann sich nicht immer darauf verlassen. Außerdem mag es eine Weile dauern, bis eine automatische Freigabe erfolgt, z.B. bis der Prozess beendet wird. Während dieser Zeit können die genannten Probleme auftreten.

Aufgabe 109 Forschen Sie nach, wie ein Prozessor Ihrer Wahl mit Paging und Segmenting umgeht.

Als Fundstellen kommen die Websites der Prozessorhersteller in Frage.

Aufgabe 110 Forschen Sie nach, welche Alternativen es zur LRU-Strategie beim Paging gibt und welche Vor- und Nachteile sie aufweisen.

Einige Stichworte: LFU, NRU, FIFO, CLIMB

Aufgabe 111 Warum ist die RS-232-Schnittstelle immer noch so weit verbreitet?

Die RS-232-Schnittstelle ist vergleichsweise einfach aufgebaut und kann daher leicht in verschiedene Bausteine integriert werden. Außerdem sorgt gerade ihre weite Verbreitung dafür, dass sie nicht so leicht verschwindet und weit verbreitet bleibt.

Aufgabe 112 Ab welcher Frequenz dominieren Hochfrequenzeffekte, wenn man Daten über eine Leitung mit 1m Länge übertragen möchte?

Es gilt: $f = c/\lambda$. Daraus ergibt sich $f = 3 \cdot 10^8$ m/s/1m $= 3 \cdot 10^8$ Hz = 300 MHz.

Wegen leitergebundener Übertragung und Rechtecksignalen muss man diesen Wert etwa durch 20 teilen. Also liegt die Grenze bei ca. 15 MHz.

Aufgabe 113 Was fällt auf, wenn man die zeitliche Reihenfolge betrachtet, in der parallele Verfahren von seriellen abgelöst wurden? Was könnten Gründe dafür sein?

Zuerst wurden Schnittstellen abgelöst, die vergleichsweise langsam und weit vom Prozessor entfernt sind, z.B. außerhalb des PC-Gehäuses Parallelport durch USB. Es folgten immer schnellere und näher am Prozessor angesiedelte Schnittstellen: P-ATA/S-ATA innerhalb des Gehäuses, dann PCI/PCIe auf dem Mainboard, dann die Speicherschnittstellen dicht am oder gar im Prozessor.

Je länger die Leitungen, desto stärker können serielle Schnittstellen ihre Vorteile ausspielen, weil parallele Verfahren dann größere elektrische Probleme und Kostenprobleme mit sich bringen. Mit fortschreitender Entwicklung erreichen serielle Verfahren ferner immer höhere Geschwindigkeiten, während bei parallelen Verfahren die Grenzen weitgehend erreicht sind.

Aufgabe 114 Nennen Sie je zwei Beispiele für Simplex-, Halbduplex- und Vollduplexkommunikation!

Simplex: GPS, Radio, Fernsehen, Broadcast-Meldungen bei Netzwerken

Halbduplex: (Amateur-)Funk, Fax, praktisch alle Bussysteme (z.B. PCI, AGP, USB)

Vollduplex: Telefonnetze, PCIe, Gigabit-Ethernet

Aufgabe 115 Welche drei grundsätzlichen Möglichkeiten, die Datentransferrate zu erhöhen, kennen Sie?

- Verringerung der Symbolrate pro Bit
- Erhöhung der (Takt-)Frequenz, mit der Daten übertragen werden

- Verwendung mehrerer Kanäle

Aufgabe 116 Informieren Sie sich, wie das Bit Stuffing bei USB vorgenommen wird.

- USB verwendet eine Form von NRZI-Codierung, die zu der aus unserer Betrachtung invertiert ist.
- Das Bit Stuffing erfolgt demnach nicht nach einer Serie von Nullen, sondern nach einer Serie von Einsen.
- Immer nach 6 Einsen wird eine Null eingefügt. Empfängt man also 7 Einsen am Stück, dann ist ein Fehler aufgetreten.

Aufgabe 117 Berechnen Sie die Bandbreite, die für 100-Base-TX-Ethernet nötig ist.

Wir wollen eine Datentransferrate von 100 MBit/s für die Nutzdaten erreichen. Wäre eine Signalperiode T genauso lang wie ein Zeitfenster Δt, dann würde man 100 MHz Bandbreite benötigen.

Weil sich bei MLT3-Codierung eine Signalperiode T aber über 4 Zeitfenster Δt erstreckt, reduziert sich die Bandbreite dadurch auf 100 MHz/4 = 25 MHz.

Zusätzlich benötigen wir aber die 4B/5B-Codierung. Sie vergrößert das Datenvolumen und somit auch die nötige Bandbreite um den Faktor 5/4 auf $5/4 \cdot 25$ MHz = 31,25 MHz.

Aufgabe 118 Wie lautet das Paritätsbit für 0110 1110, wenn man gerade Parität möchte? Wie bei ungerader Parität?

0110 1110 enthält 5 Einsen, also eine ungerade Zahl. Um auf eine gerade Parität zu kommen, muss man somit eine weitere 1 ergänzen. Für ungerade Parität wird eine 0 ergänzt.

Aufgabe 119 Wie lautet die Hamming-Distanz für folgenden Code:

Tab. A.10: Beispielcode C

dezimal	Code C
0	0101
1	0110
2	1001
3	1110

Man vergleicht jedes Codewort mit jedem und notiert die Zahl unterschiedlicher Bits:

- Codeworte 0/1: **2** unterschiedliche Bits
- Codeworte 0/2: **2** unterschiedliche Bits
- Codeworte 0/3: **3** unterschiedliche Bits
- Codeworte 1/2: **4** unterschiedliche Bits
- Codeworte 1/3: **1** unterschiedliche Bits
- Codeworte 2/3: **3** unterschiedliche Bits

Von diesen Werten nimmt man das Minimum. Das ist dann die Hamming-Distanz.

In unserem Falle ist das die 1, denn Codewort 1 unterscheidet sich von Codewort 3 nur in einem einzigen Bit.

Es gilt hier also: $d_{min} = 1$.

Aufgabe 120 Wie groß sind d_{min} und F_{emax} in unserem Beispiel?

Wie wir sehen, müssen alle drei Bits kippen, um von dem einen gültigen Codewort zu dem anderen zu gelangen. Es gilt also für die Hamming-Distanz $d_{min} = 3$. Somit ist die maximale Zahl erkennbarer Fehler

$F_{emax} = d_{min} - 1 = 3 - 1 = 2$

Es können also *alle* 2-Bit-Fehler erkannt werden.

Aufgabe 121 Es sei ein Code mit $d_{min} = 5$ gegeben. Wie viele Fehler sind maximal erkennbar, wie viele korrigierbar?

$F_{emax} = d_{min} - 1 = 5 - 1 = 4$

$F_{kmax} = (d_{min} - 1)/2 = 2$

Aufgabe 122 Hamming-Distanz

Tab. A.11: Beispielcodes zur Hamming-Distanz

dezimal	Code C_1	Code C_2	Code C_3	Code C_4
0	001	0100	000110	11110000
1	010	1101	101011	00001111
2	100	1001	011100	
3	111	0100		

a) Ermitteln Sie für jeden dieser Codes die Hamming-Distanz.
b) Geben Sie jeweils an, wie viele Fehler maximal erkennbar und wie viele korrigierbar sind.
c) Es wird bei C_1 101, bei C_2 1111, bei C_3 000000 und bei C_4 11000010 empfangen. Wozu könnten diese Codewörter korrigiert werden? Ist die Korrektur eindeutig? Ist sie korrekt?

Zu a)

C_1: $d_{min} = \min(2,2,2,2,2,2) = 2$
C_2: $d_{min} = \min(2,3,1,1,1,2) = 1$
C_3: $d_{min} = \min(4,3,5) = 3$
C_4: $d_{min} = 8$

Zu b)

Es gilt:

$F_{emax} = d_{min} - 1$

$F_{kmax} = \frac{d_{min}-1}{2}$ für d_{min} ungerade

$F_{kmax} = \frac{d_{min}-2}{2}$ für d_{min} gerade

C_1: $F_{emax} = 2 - 1 = 1$, $F_{kmax} = (1 - 1)/2 = 0$

C_2: $F_{emax} = 1 - 1 = 0$, Formel für Fehlerkorrektur nicht anwendbar, aber $F_{kmax} = 0$ kann angenommen werden, wenn Fehler nicht einmal erkennbar sind.

C_3: $F_{emax} = 3 - 1 = 2$, $F_{kmax} = (2 - 2)/2 = 0$

C_4: $F_{emax} = 8 - 1 = 7$, $F_{kmax} = (7 - 1)/2 = 3$

Zu c)

C_1: 101 könnte zu 001, 100 oder 111 korrigiert werden, wenn man einen 1-Bit-Fehler annimmt. Alle haben den gleichen Abstand zu 101, nämlich 1 Bit. Es ist also keine eindeutige Korrektur möglich, was das Ergebnis aus b) bestätigt.

C_2: 1111 könnte zu 1101 korrigiert werden, was als Einziges den Abstand 1 zu 1111 besitzt. Diese Korrektur wäre eindeutig.

In b) hatten wir aber festgestellt, dass Fehler nicht einmal erkennbar sind. Das ist kein Widerspruch, denn die Formeln geben nur an, dass Fehler mit der berechneten Bitzahl *immer* erkennbar oder korrigierbar sind. Hier haben wir einfach Glück gehabt, dass die Fehlerkorrektur trotzdem funktionieren würde.

C_3: 000000 kann man zu 000110 korrigieren. In diesem Fall hätte es sich um einen 2-Bit-Fehler gehandelt. Wie aus b) ersichtlich sind 2-Bit-Fehler bei diesem Code immer korrigierbar.

C_4: 11000010 hat zu 11110000 den Abstand 3 und zu 00001111 den Abstand 5. Entsprechend korrigiert man zu 11110000. Das wäre ein 3-Bit-Fehler gewesen, den man bei C_4 immer korrigieren kann.

Aufgabe 123 Hamming-Code

Ein Hamming-Code enthalte

a) m = 1 Nutzbits und k = 3 Kontrollbits
b) m = 120 Nutzbits und k = 8 Kontrollbits.

Wie groß sind jeweils Hamming-Distanz d_{min} und Redundanz r (in %)?

Anmerkung: Die Redundanz sei – für diese Aufgabe vereinfacht – das Verhältnis von Kontrollbits zur Gesamtmenge der Daten.

Zu a)

$m := 1, k := 3$

$n = m + k = 1 + 3 = 4$

$r = k/(m + k) = k/n$

$r = 3/4 = 75\ \%$

$d_{min} = ?$

$m = 2^{k-(dmin-3)} - [k - (d_{min} - 3)] - 1 = 2^{k-(dmin-3)} - k + d_{min} - 4$

1. Weg: Wir versuchen, die Gleichung nach d_{min} aufzulösen.
Substitution: $z := k - (d_{min} - 3)$
$\Rightarrow m = 2^z - z - 1$

Diese Gleichung ist nicht allgemein nach z und somit d_{min} auflösbar!

2. Weg: Wir versuchen, eine für das Einsetzen von d_{min} möglichst gut geeignete Form zu finden.

$m = 2^{k-(dmin-3)} - k + d_{min} - 4$

$\Leftrightarrow m + k - d_{min} + 4 = 2^{k-(dmin-3)}$

$\Leftrightarrow m + k - d_{min} + 4 = 2^k \cdot 2^{-dmin} \cdot 2^3$

$\Leftrightarrow m + k - d_{min} + 4 = 8 \cdot 2^k / 2^{dmin}$

$\Leftrightarrow 1/2^{dmin} = (m + k - d_{min} + 4)/(8 \cdot 2^k)$ (Formel 1)

Die Formel 1 enthält auf der linken Seite im Zähler eine 1 und im Nenner eine Zweierpotenz. Die rechte Seite muss genauso aufgebaut sein. Im Zähler der rechten Seite muss also eine Zweierpotenz stehen, sonst lässt sie sich nicht vollständig gegen den Nenner kürzen.

Setzen wir nun Zahlenwerte in Formel 1 ein:

$1/2^{dmin} = (1 + 3 - d_{min} + 4)/(8 \cdot 2^3)$

$\Leftrightarrow 1/2^{dmin} = (8 - d_{min})/2^6$ (Formel 1')

Es muss also für eine Zweierpotenz im Zähler der rechten Seite einer der folgenden Fälle vorliegen:

(1) $8 - d_{min} = 8 \Leftrightarrow d_{min} = 0$

(2) $8 - d_{min} = 4 \Leftrightarrow d_{min} = 4$

(3) $8 - d_{min} = 2 \Leftrightarrow d_{min} = 6$

(4) $8 - d_{min} = 1 \Leftrightarrow d_{min} = 7$

Fall (1) ist nicht sinnvoll, denn bei $d_{min} = 0$ gäbe es mehrdeutige Codeworte und Fehlererkennung wäre sinnlos.

Bei den Fällen (3) und (4) hätten wir ein d_{min}, das höher ist als die Zahl der Bits im Code. Das lässt sich nicht erreichen.

Es bleibt also nur Fall (2) übrig. Wir setzen $d_{min} = 4$ in unsere Formel 1' ein:

l.S.: $1/2^4$

r.S.: $(8 - 4)/2^6 = 4/2^6 = 2^2/2^6 = 1/2^4 =$ l.S.

Also ist $d_{min} = 4$.

Zu b)

$m = 120$, $k = 8$

$\Rightarrow n = m + k = 120 + 8 = 128$

$r = k/n = 8/128 = 6{,}25\%$

Wir setzen in Formel 1 ein:

$1/2^{dmin} = (132 - d_{min})/(2^{11})$

Wir probieren zunächst $d_{min} = 4$:

l.S.: $1/2^4$

r.S.: $128/2^{11} = 2^7/2^{11} = 1/2^4 =$ l.S.

Wiederum ist also $d_{min} = 4$, trotz der völlig anderen Redundanz.

Aufgabe 124 Die Codeworte eines Codes C besitzen eine Länge von 12 Bit. Es soll eine Hamming-Distanz von 3 erreicht werden. Wie viele Nutzbits können in einem Codewort von C untergebracht werden? Rechnerische Begründung!

$n := 12$, $d_{min} := 3$

$n = m + k$

$m = ?$

$m = 2^{k-(dmin-3)} - [k - (d_{min} - 3)] - 1 = 2^{k-(dmin-3)} - k + d_{min} - 4$

$\Rightarrow n = 2^{k - dmin + 3} + d_{min} - 4$

$\Leftrightarrow n - d_{min} + 4 = 2^{k - dmin + 3}$

$\Leftrightarrow ld(n - d_{min} + 4) = ld(2^{k - dmin + 3})$

$\Leftrightarrow ld(n - d_{min} + 4) = k - d_{min} + 3$

$\Leftrightarrow k = d_{min} - 3 + ld(n - d_{min} + 4)$

$m = n - k = n - d_{min} + 3 - ld(n - d_{min} + 4)$

einsetzen:

$m = 12 - 3 + 3 - ld(12 - 3 + 4) = 12 - ld13 \approx 8{,}3$

Wir müssen m abrunden, sonst erreichen wir nicht das gewünschte $d_{min} = 3$. Also wählen wir $m = 8$, $k = 4$.

Aufgabe 125 Wie groß ist die Hamming-Distanz für 8 Nutzbits und 4 Kontrollbits pro Codewort?

m = 8, k = 4 (wie in Aufgabe 24 ermittelt)

Formel 1 aus Aufgabe 23 lautete:

$1/2^{dmin} = (m + k - d_{min} + 4)/(8 \cdot 2^k)$

Wir setzen ein:

$1/2^{dmin} = (8 + 4 - d_{min} + 4)/(8 \cdot 2^4)$

$\Leftrightarrow 1/2^{dmin} = (16 - d_{min})/2^7$

In Aufgabe 24 hatten wir ausgerechnet, dass wir mit $m \approx 8{,}3$ ein $d_{min} = 3$ bekommen hätten. Nun haben wir aber nur m=8. d_{min} sollte also oberhalb von 3 liegen.

Wir prüfen $d_{min} = 3$:

l.S.: $1/2^3 = 1/8$

r.S.: $(16 - 3)/2^7 = 13/128 <$ l.S.

Wir prüfen $d_{min} = 4$:

l.S.: $1/2^4 = 1/16 = 2/32$

r.S.: $(16 - 4)/2^7 = 12/128 = 3/32 >$ l.S.

Das Ungleichheitszeichen dreht sich zwischen $d_{min} = 3$ und $d_{min} = 4$ um, also muss d_{min} zwischen 3 und 4 liegen, was nach dem Ergebnis der vorigen Aufgabe plausibel ist. Es ist also auch ein nicht-ganzzahliges d_{min} möglich.

Um d_{min} genauer zu ermitteln, könnten wir das Intervall sukzessive verkleinern, z.B. halbieren, und prüfen, ob sich das Ungleichheitszeichen immer noch umdreht. Wenn ja, ist d_{min} im untersuchten Intervall, wenn nicht in der anderen Hälfte des ursprünglichen Intervalls. So kann man d_{min} beliebig genau annähern.

Aufgabe 126 Man schließt einen USB-Stick, auf dem Videos enthalten sind, an einen Rechner an, Beim Abspielen des Videos kommt es zu Stockungen. Woran kann das liegen?

Der USB-Stick wird vermutlich als USB-Massenspeicher im Bulk Mode betrieben. Dadurch können schwankende Datentransferraten entstehen. Außerdem kann es sein, dass seine Datentransferrate zu niedrig für die Wiedergabe von Videos ist. Weitere Fehlerquellen können Betriebssystemtreiber sein.

Aufgabe 127 Ein Tool, das Systeminformationen ermittelt, zeigt für eine Festplatte mehrere Tausend Köpfe an. Woran kann das liegen?

Die Anzahl der Schreib-Leseköpfe, die nach draußen gemeldet wird, kann eine ganz andere sein, als die tatsächlich vorhandene. Davon kann man insbesondere ausgehen, wenn mehr als 16 Köpfe gemeldet werden, da kaum mehr als 8 Scheiben in einer Platte verbaut werden.

Der Grund liegt darin, dass für die Angabe von Cylinders, Heads und Sectors gewisse historisch gewachsene Maximalgrenzen bestehen, die nicht überschritten werden dürfen. Wenn die tatsächlich vorhandene Zahl von Cylinders beispielsweise über dem Maximalwert liegt, reduziert man ihn auf einen für das BIOS akzeptablen Wert. Im Gegenzug erhöht man den Wert, der als Anzahl der Köpfe angegeben wird, um insgesamt die richtige Speicherkapazität der Platte zu erhalten. Die gemeldeten Werte haben also oft nichts mehr mit dem internen Aufbau zu tun.

Aufgabe 128 Jemand kauft eine Festplatte mit doppelter Speicherkapazität als die bisherige, in der Annahme, dass sie deswegen schneller wäre. Tatsächlich ist sie das nicht. Woran könnte das liegen?

Eventuell hat sich nicht die Schreibdichte verdoppelt, sondern nur die Anzahl der Scheiben in der Platte. Üblicherweise wird immer nur von einer einzigen Scheibe gelesen, so dass sich die Datentransferrate nicht erhöht, wenn man mehr Scheiben verwendet.

Aufgabe 129 Wie viele Bits können insgesamt auf einer CD-DA mit 2352 Nutzbytes pro Sektor unerkannt kippen? Hinweis: Die Standard-Audio-CD hat eine Länge von 74 Minuten und verwendet 75 Sektoren pro Sekunde.

Die CD enthält als Nutzdaten

$n = 74 \text{ Min.} \cdot 60 \text{ s/Min.} \cdot 75 \text{ Sekt./s} \cdot 2352 \text{ Bytes} = 783\,216\,000 \text{ Bytes}$

Bei einer Bitfehlerrate von $f = 10^{-8}$ können pro CD

$n \cdot f = 10^{-8} \cdot 783\,216\,000 \text{ Bytes} = 7{,}83 \text{ Bytes} = 62 \text{ Bits}$ (gerundet) kippen, ohne dass dies erkennbar ist. Das ist auch der Grund, warum Kopien von Audio-CDs einen Qualitätsverlust erleiden.

Aufgabe 130 Wie viele CD-ROMs könnte man im Mittel komplett lesen, ohne dass ein unerkennbarer Bitfehler auftritt?

Diesmal haben wir als Nutzdaten pro CD

$n = 74 \text{ Min.} \cdot 60 \text{ s/Min.} \cdot 75 \text{ Sekt./s} \cdot 2048 \text{ Bytes} = 681\,984\,000 \text{ Bytes}$

$n \cdot f = 10^{-12} \cdot 681\,984\,000 \text{ Bytes} = 5{,}46 \cdot 10^{-3}$ Bits können pro CD unerkannt fehlerhaft sein.

$1/(n \cdot f) = 183$ (gerundet)

Man kann also im Mittel 183 CD-ROMs vollständig lesen, ohne dass ein Fehler auftritt.