6 Lösung der Aufgabensammlung

6.1 Einleitung

Ausgehend von Verständnisfragen zu den Begriffen und Methoden der Informatik folgen Übungsaufgaben zur selbständigen Bearbeitung. Der Schwierigkeitsgrad der Aufgaben ist jeweils mit Sternen entsprechend dem nachfolgenden Schema gekennzeichnet:

* - einfach

** - mittel

*** - schwer

6.2 Allgemeine Fragen

Über die eigentliche Antwort hinaus wird oft eine vertiefende Beschreibung der Lösung in kursiv angefügt. Sie dient nur einem besseren Verständnis der Lösung.

Aufgabe 1: Information, Daten und Algorithmen

a. Daten sind eine strukturierte Darstellung von Information. Algorithmen beschreiben die strukturierte Verarbeitung von Information. Informationen haben einen Neuigkeitsgehalt und weisen eine Verwendbarkeit auf.

b. „E“: Eingabe, „V“: Verarbeitung, „A“: Ausgabe

c. Der Algorithmus ist eine Verarbeitungsvorschrift. Ein Prozess beschreibt den Ablauf und die Ausführung eines konkreten Programms.

d. Alle polynomielle Ordnungen (inklusive der linearen, logarithmischen und konstanten).

e. Zur Lösung von Problemen mit exponentieller Ordnung

Aufgabe 2: Datendarstellung

a. Vorteile: direkte Darstellung dezimaler Ziffern; zur Speicherung von buchhalterisch exakten Zahlen mit fixer Stellenzahl geeignet.
Nachteile: „Verschwendung“ von Speicher, da 6 Symbole ungenutzt; komplizierte Weiterverarbeitung verglichen mit Zweierkomplement.

b. Vorteile: Einfache Multiplikation und Division; einfache Negation.
Nachteile: Doppelte Nulldarstellung; aufwendige Addition *(negative Zahlen!)* bzw. Subtraktion.

c. Die Zahl Null. *Bei einer normalisierten Mantisse bleibt eine Eins als einzige Vorkommastelle übrig. Da das Vorhandensein der Eins bei der Kodierung und Dekodierung bekannt ist, kann die Eins weggelassen werden. Dieses Bit wird Hidden-Bit bezeichnet. Auf der anderen Seite bedeutet eine Eins vor dem Komma, dass die Zahl aufgrund eines negativen Exponenten zwar sehr klein - aber niemals Null werden kann.*

d. Für eine Dezimalstelle = 10 Zustände, werden ld(10) = 3.322 Bits benötigt. Ein Bit erhöht folglich die Genauigkeit um 1/3.322 ≅ 0.3 Dezimalstellen.

e. Gefahren: Vergleich bzw. Fehlerfortpflanzung *(da Zahlen i.d.R. nur Näherungslösungen sind)*, Rechnen über den Bereich der signifikanten Stellen hinaus.

f. Die Datenstruktur, welche Hierarchie direkt abbildet, ist der Baum.
Auch Graphen können über gewichtete Kanten Hierarchie-Beziehungen darstellen.

g. Verkettete Liste. Vorteil: einfaches Einfügen und Entfernen von Objekten. Nachteil: Keine Indizierung, also kein schneller Zugriff auf dedizierte Objekte. Ein weiterer Vorteil ist der dynamische Speicherbedarf gegenüber einem festen Speicherbedarf; ein weiterer Nachteil ist das aufwendigere Verwalten der Listen (Anlegen, Löschen).

h. LIFO: engl. *last in, first out*, dt. der als letztes herein kommt, kommt als erstes wieder heraus. Einsatzgebiet: Stapelspeicher; Speicher für Zwischenergebnisse.
FIFO: engl. *first in, first out*, dt. der als erstes herein kommt, kommt als erstes wieder heraus. Einsatzgebiet: Warteschlange, Pufferspeicher.

i. Das Feld.

Aufgabe 3: Programmerzeugung

a. Eine Programmiersprache ist eine formale Sprache. *Eine natürliche Sprache unterliegt keinen so strengen formalen Richtlinien wie Programmiersprachen. Durch den Kontext wird bei natürlichen Sprachen häufig die Bedeutung einzelner Worte verändert, während Programmiersprachen immer eindeutig in ihrer Definition sein müssen.*

b. Die EBNF wird zu Definition formaler Sprachen, wie z.B. Programmiersprachen verwendet. Andere Anwendungsfelder sind die Beschreibungen von Daten bzw. Dateiformaten, Netzwerkprotokollen etc.

c. Terminalsymbole entsprechen den Vokabeln einer formalen Sprache (*diese können nicht mehr weiter unterteilt werden*). Ein Nicht-Terminalsymbol wird mit Hilfe von Produktionen aus Terminalsymbolen definiert *(„Substitution“). In der EBNF findet man Terminalsymbole nur auf der rechten Seite vom Gleichheitszeichen.*

d. Alphabet, Syntax (=Vokabeln und Grammatik) und Semantik.

e. Endlosschleife, abweisende Schleife, nicht abweisende Schleife, Zählerschleife.

f. Vorteile:

- Reduktion des Programmcodes (Strukturierung, Abstraktion).
- Auslagern von mehrfach oder wieder verwendbarem Programmcode.
- Aufbau von modularen Programmen
(Funktionen können separat entwickelt und getestet werden).

g. Man spricht von Methoden, wenn diese Daten innerhalb eines Objektes manipulieren, und damit zu einer Klasse gehören. Funktionen arbeiten dagegen mit Daten, welche als Parameter übergeben werden.

h. Eine Klasse besteht aus der allgemeinen Beschreibung einer Datenstruktur und den dazugehörigen Methoden, wie diese Daten verarbeitet werden können. Ein Objekt ist dagegen eine konkrete Instanz einer Klasse. Es enthält über die abstrakte Beschreibung der Datenstruktur hinaus auch konkrete Inhalte, welche sich zur Laufzeit ändern können. *Anders ausgedrückt benötigt ein Objekt zur Laufzeit Speicher, um die individuellen Daten aufnehmen zu können, während eine Klasse nur die Strukturbeschreibung enthält und damit keinen Speicher für Daten benötigt.*

i. Alternativanweisungen stellen ein zentrales Element der imperativen Programmierung dar, weil sie es ermöglichen, dass Programme auf Ereignisse von Außen reagieren. Ohne Alternativanweisungen wären alle Programme starr in der Ausführung (*z.B. Waschmaschinensteuerung durch Programmrad*). *Ebenso wären keine rekursiven Algorithmen möglich*.

j. Objektcode zusammenbinden, Bibliotheken hinzufügen, Programmimage (lauffähiges Programm) erzeugen. *Ein Linker stellt aus einer Reihe von Code-Quellen und Bibliotheken ein lauffähiges Programm zusammen. Er beschleunigt die Softwareentwicklung, da damit der Vorgang des Übersetzens einzelner Module und der Vorgang des Generierens eines Programmimages separiert werden. Damit ist es nur noch erforderlich, Module zu übersetzen, wenn diese Änderungen aufweisen; bereits fertige Module und Bibliotheken müssen nur dazu gelinkt werden. Außerdem ermöglicht der Linker das Zusammenbinden von Quellen aus verschiedenen Programmiersprachen.*

k. Ein Compiler erzeugt ein eigenständig lauffähiges Programm, während ein Interpreter ein Programm zur Laufzeit abarbeitet. *Typische Anwendungsgebiete für Interpreter sind: sich häufig ändernde und/oder kurze Programme oder Skripte, wie z.B. Shell-Skripte, Skripte für mathematische Berechnungen (Matlab), einfache Funktionalität für Webauftritte, einfache Formularfelder sowie Skripte zum Testen von Software. Ein Compiler kommt immer dann zum Einsatz, wenn ein kommerzielles Produkt erzeugt werden soll, welches ohne zusätzliche Hilfsmittel lauffähig ist; wenn größere Programme entwickelt werden müssen; wenn die Geschwindigkeitsanforderungen hoch sind.*

Aufgabe 4: Digitaltechnik und Speicher

a. UND, ODER, NICHT

b. DRAM: Ein Kondensator wird zur Speicherung verwendet.
SRAM: Der Inhalt wird durch rückgekoppelte Transistoren (Flip-Flop) gespeichert.

c. Leseverstärker für Refresh.

d. Flash-Speicher können nur blockweise gelöscht werden. Ein wahlfreier Zugriff ist damit nicht gegeben. *Außerdem sind Flash-Speicher erheblich langsamer beim Schreibvorgang als DRAMs oder SRAMs.*

e. Festwertspeicher *(erlaubt die Realisierung jedes Schaltnetzes). Einfachste Realisierung erfolgt über einen 256 Byte großen Festwertspeicher, welcher als Eingänge die Adressleitungen 0-3 sowie 4-7 und als Ausgang die Datenleitungen 0-7 aufweist (bei einer 4-Bit-Multiplikation weist das Ergebnis bis zu 8 Bit auf!).*

f. Kurzeitige ungültige Ergebnisse von digitalen Schaltnetzen, welche durch unterschiedliche Gatterlaufzeiten entstehen.

g. Durch Kaskadierung *(hintereinander schalten)* mehrerer Flip-Flops *entsprechend der Speichertiefe. Die Ausgänge werden dabei mit den Eingängen der nachfolgenden Flip-Flops verschaltet. Um ein direktes Durchreichen der Information durch alle Flip-Flops zu vermeiden, muss eine flankengetriggerte Variante eingesetzt werden. Eine solche Schaltung wird auch Schieberegister bezeichnet. Oft sind Ein- bzw. Ausgänge der Flip-Flops nach außen geführt, um eine Wandlung von seriellen zu parallelen Datenströmen und umgekehrt zu realisieren.*

h. Der Durchgriff vom Eingangs- auf das Ausgangsschaltnetz beim Mealy-Automaten.

i. Oder. *Die Oder-Verknüpfung eines beliebigen Bitwertes mit einer 1 ergibt am Ausgang immer eine 1*.

Aufgabe 5: Rechnertechnik

a. Es unterscheidet die Datenströme, welche gleichzeitig durch einen Befehl verarbeitet werden können. SISD (single instruction single data) entspricht der sequentiellen Abarbeitung eines einzelnen Datenstroms. SIMD (single instruction multiple data) entspricht

der sequentiellen Abarbeitung mehrerer Datenströme. Dabei werden die gleichen Befehle auf die unterschiedlichen Daten angewandt.

b. Eingebettete Systeme realisieren die Funktionalität eines Geräts (oder einer Teilkomponente) und repräsentieren dabei nicht universelle (also beliebig nutzbare, programmierbare) Rechenleistung. Sie verfügen über keine Benutzerverwaltung (Benutzer = Gerät).

c. Die Von-Neumann-Architektur verwendet einen gemeinsamen Arbeitsspeicher für Programm und Daten.

d. Systeme mit unterschiedlicher Repräsentation von Daten und Programm, wie z.B. Signalprozessoren, RISC-Mikrocontroller.

e. Die ALU (*arithmetic and logical unit*) führt arithmetische Operationen und logische Verknüpfung auf ihre Eingangsoperanden (*Register*) durch und liefert Statusinformationen. Das Steuerwerk setzt zur Auswahl der Operation *(z.B. über ein Mikroprogramm)* entsprechende Steuerleitungen an der ALU.

f. Das Bussystem.

g. Ein Mikroprogramm stellt den Kontrollfluss des Steuerwerks einer CPU als eine Folge von Signalen zum Auswählen, Ausgeben und Laden von Registern, zum Auswählen von ALU Operationen, zur Aus- und Eingabe über den externen Bus etc. dar.

h. Die Ausführung des Mikroprogramms wird durch das Maschinenwort (OpCode) und das Status-Register (Flags) ausgewählt.

i. Die Verwendung mehrerer paralleler interner Busse, da typische Operationen, wie z.B. das Erhöhen des Adresszählers parallel zu anderen Operationen durchgeführt werden können („Instruction Prefetch“).

j. Das Feld (indizierter Zugriff auf Daten) und der Stapelspeicher.

k. Der Stackzeiger darf seinen zugewiesenen Bereich nicht über- bzw. unterschreiten (Überlauf, Unterlauf).

l. Kleiner Befehlssatz, welcher schnell ausgeführt werden kann. Compiler optimiert Programme. Pipeline-Verarbeitung. Gleiches Format und gleiche Ausführgeschwindigkeit aller Maschinenbefehle.

m. Control-Hazards (Kontrollfluss bedingte Probleme durch Sprunganweisungen), Structural-Hazards (Ressourcenkonflikt) und Data-Hazards (Datenabhängigkeit der Operanden).

n. *Es genügt eine Antwort*:

- Compiler-Optimierung der Maschinenbefehle (gegen Data-Hazards)
- Out-of-order-Execution (gegen Data-Hazards)
- Früherkennung von Sprunganweisungen (gegen Control-Hazards)
- Branch Prediction *„Sprungvorhersage“* (gegen Control-Hazards)
- Separater Code/Daten-Cache (gegen Structural-Hazards)
- Mehrfache Hardware-/Bussysteme auf der CPU (gegen Structural-Hazards)

o. zeitliche und örtliche Lokalität, Geschwindigkeit des Caches verglichen mit dem Arbeitsspeicher.

p. Paritätsbit, Kodierung (z.B. 8 Bit durch 10 Bit).

q. Produktformel: $512\ \text{Bytes} \cdot n_s \cdot n_c \cdot n_h$

r. Es sind bis zu vier primäre Partitionen möglich *(wobei nicht jedes Betriebssystem dies unterstützt).*

s. Verwaltung der Betriebsmittel und Abstraktion der Hardware.

t. Vorteil: einfache Realisierbarkeit *(keine besondere Hardware für die Speicherverwaltung erforderlich, sondern als reine Software-Lösung realisierbar)*, Nachteil: langsam (*da bei jedem Prozesswechsel der geswappte Speicher wieder geladen werden muss*).

Aufgabe 6: Rechnernetze und Protokolle

a. Durch die OSI-Schicht. *HTML (Darstellungsschicht) wird über HTTP (Sitzungsschicht) übertragen.*

b. Adressierung *(Vermittlung der IP-Pakete)* und Fragmentierung *(Aufteilung größerer Pakete in kleinere, wenn z.B. das Netzwerk nur den Transport von kleineren Paketen unterstützt).*

c. TCP ist **verbindungsorientiert**, UDP ist **verbindungslos**. *Bei TCP wird die Datenübertragung durch Empfangsbestätigungen überwacht. Verlorene Datenpakete werden erneut gesendet. UDP ist dagegen verbindungslos; es können zwar Daten übertragen werden, eine Überwachung erfolgt nicht (müsste durch eine höhere Schicht realisiert werden).*

d. Sie wird länger. Das Verfahren ist erforderlich, da SMTP nur einen 7 Bit Zeichensatz unterstützt. *Es werden immer 3 Bytes durch 4 ASCII-Symbole übertragen, d.h. die Nachrichtenlänge wird um den Faktor 1,33 länger (33 %). Ein Verfahren ist notwendig, da SMTP die Verwendung des 7-Bit ASCII-Codes spezifiziert; binäre (Daten mit mehr als 7 Bit) müssen folglich kodiert werden. Theoretisch wäre auch ein base128 Verfahren (Abbildung von 7 Bytes auf 8 ASCII-Symbole) möglich, allerdings stellt das base64-Verfahren nur Anforderung an 64 definierte Zeichen, welche auch auf Rechnern verwendet werden, welche nicht den ASCII-Zeichensatz unterstützen, wie z.B. Buchstaben, Zahlen und wenige Symbole.*

e. MIME (engl. *multipurpose internet mail extensions*) dienen zur Kodierung beliebiger Dateninhalte für die Übertragung per E-Mail.

f. Systemverfügbarkeit (Schutz vor Ausfall), Datenintegrität (Schutz vor Manipulation) und Datenvertraulichkeit (Schutz vor Abhören).

g. Ein Computervirus braucht ein Wirtsprogramm, um sich zu reproduzieren bzw. um Schaden anzurichten. Die Verbreitung erfolgt durch Weitergabe des infizierten Wirtsprogramms. Ein Computerwurm dagegen versucht, sich selbständig über Netzwerkdienste zu verbreiten.

6.3 Algorithmen

Aufgabe 7: Sortieren *

a. *1. Durchgang*: Tauschen: Eva, Daniel; Gerhard, Franz; Klaus, Inge; Rudi, Anton.

b. Bernd, Carla, Daniel, Eva, Franz, Gerhard, Inge, Klaus, Anton, Rudi.

c. 9. *Es sind soviele Durchgänge erforderlich, bis Anton an erster Stelle steht. Man zählt die Anzahl von Anton bis Bernd. Im ungünstigsten Fall - dito - sind bei n Datensätze n-1 Durchgänge erforderlich.*

d. 5.
1. Durchgang: 3317,3318,3319,3320,3316,3322,3323,3333,3343,3399; alles bis auf 3316 ist damit bereits sortiert, folglich müssen nur noch die Durchgänge bis 3316 an erster Stelle ist, gezählt werden.

e. $O(n^2)$. *Eigentlich $(n-1)\cdot(n-1)$, da n-1 Vergleiche pro Durchlauf nötig sind bei maximal n-1 Durchgängen. Da bei der Angabe der Ordnung nur das höchste Polynom angegeben wird, folgt aus n^2-2n+1: $O(n^2)$.*

Aufgabe 8: Fakultät **

a.
```
= 7 · Fakultät von 6
= 7 · 6 · Fakultät von 5
= 7 · 6 · 5 · Fakultät von 4
= 7 · 6 · 5 · 4 · Fakultät von 3
= 7 · 6 · 5 · 4 · 3 · Fakultät von 2
= 7 · 6 · 5 · 4 · 3 · 2 · Fakultät von 1 = 5040
```

b. **Ja**, da der Algorithmus selbst zur Lösung herangezogen wird.

c. **Ja**, da er aus einzelnen, unabhängigen Schritten besteht.

d. **statisch finit**, da endlich in der Notation.
dynamisch finit, nur in den Fällen, bei denen der Algorithmus terminiert.

e. **Nein**, negative Werte von n werden nicht berücksichtigt.

f. **Nein**, bei negativen Werten von n tritt die Abbruchbedingung nie ein.

g. **Ja**, *da deterministisch.*

h. **Ja**, weil Operationen und Operanden nicht zufällig sind.

Aufgabe 9: Monte-Carlo-Methode ***

a. **Nein**, da Zufallszahlen verwendet werden.

b. **exponentiell**.
Für das Berechnen einer *Dezimalstelle werden mindestens 10 Stichproben benötigt. Für die nächste Dezimalstelle 100, dann 1000 usw. Die Ordnung beträgt folglich $O(10^n)$ = exponentiell. Da die Zahl π irrational ist, lässt sie sich durch den Bruch $i/n\cdot 4$ nie exakt darstellen, folglich kann auch nicht eine zufällige Kombination von i und n zu einem schnelleren Ergebnis führen.*

c. **Nein**, aufgrund exponentieller Ordnung.

d. **Nein**. *Auch wenn sich A und S auf 1000 Nachkommastellen nicht unterscheiden, so könnte sich jedesmal ein anderes 1000-stelliges S bzw. A ergeben. Aufgrund der Zufallszahlen ergibt sich jedesmal eine andere Näherungslösung.*

Aufgabe 10: Das Rucksackproblem **

a. Die Anzahl der Auswahlmöglichkeiten steigt mit 2^n, d.h. die Problemklasse ist **exponentiell**. *Um eine optimale Lösung zu finden, könnten Sie eine Baumstruktur zeichnen: Ausgehend vom LCD-Fernseher (Wurzel) überprüfen sie die beiden Zweige „einpacken ja/nein". auf der nächsten Ebene prüfen Sie die Zweige „Laptop einpacken ja/nein". Es ergibt sich mit jedem Produkt eine neue Ebene, so dass Sie in diesem Beispiel auf 2^9 zu untersuchenden Kombinationen kommen. Es handelt sich um ein sog. NP-Problem, welches deterministisch (zumindest für größere n) nicht gelöst werden kann.*

b. **Nein**! Zwar ist eine Abbruchbedingung vorhanden, sollten ausnahmsweise jedoch Waren mit weniger als 11 Kg angeboten werden, so würde die erste Schleife nicht verlassen werden!
*Wir untersuchen Algorithmen immer **unabhängig** von den Eingabedaten. In diesem Beispiel bedeutet es auch, dass der Algorithmus **nicht vollständig** ist, da er nicht prüft, ob die Voraussetzung Gesamtgewicht >= 11 Kg erfüllt ist.*

c. **Nein**! Keine Zufallswerte, Ablauf ist eindeutig.

d. **Ja**, da deterministisch!

e. **Nein**, es wird nicht das Optimum gefunden:
Greedy: LCD-TV, Laptop, DVD-Recorder = € 5.500,
Optimum: alles außer Laptop, Kamera-Set = €5.800.
Begründung: Greedy tastet nur das letzte Objekt im Rucksack an, folglich wird der mögliche Lösungsraum stark reduziert. Gleichzeitig wird dadurch die Ordnung erheblich vereinfacht.

f. **Nicht vollständig!** Die Voraussetzung mindestens einer Produktmenge mit einem Gesamtgewichts = 11 Kg wird nicht überprüft.

g. **Nein**, da nicht vollständig, s.o. *Die zusätzliche Bedingung von "exakt 11 Kg" erschweren das Terminieren noch weiter! Theoretisch (da nicht vollständig) könnte ein Artikel, welcher mehr als 11 Kg wiegt, immer im Rucksack verweilen!*

h. **Nein**, bei beliebigen Eingabedaten ließen sich unterschiedliche Eingabemengen finden, welche exakt 11 Kg wiegen und damit als Ergebnis ausgewählt werden könnten. *Das gilt z.B. auch für die vorgegebenen Daten. Da die Ergebnismenge daraus zufällig gewählt wird, ist der Algorithmus nicht determiniert.*

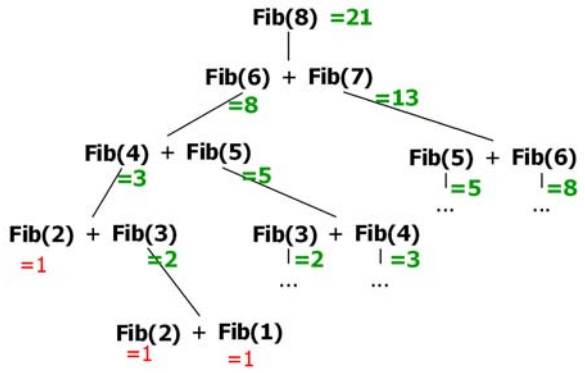
i. Das optimale Auswahlkriterium ist der **Preis pro Kilogramm.** *Der Greedy-Algorithmus weist eine lineare Ordnung auf und ist - im Vergleich zur Problemklasse - sehr schnell durchzuführen! Wenden Sie den Greedy-Algorithmus auf diese Weise an, so erhalten Sie bei diesem Beispiel das Optimum. Allerdings gilt, dass - aufgrund der Quantisierung der Ware (sie dürfen nicht 200g vom DVD-Player mitnehmen) und der Eigenschaft dieses Greedy-Algorithmus, nur ein Element wieder herausnehmen zu dürfen - Sie nicht immer das Optimum erhalten. Dafür ist die Ordnung nicht exponentiell!*

Aufgabe 11: Fibonacci-Zahlen **

a.
```
Fibonacci-Zahl von 8=
= Fibonacci-Zahl von 6 + Fibonacci-Zahl von 7
= (Fibonacci-Zahl von 4 + Fibonacci-Zahl von 5)
 +(Fibonacci-Zahl von 5 + Fibonacci-Zahl von 6)
=((Fibonacci-Zahl von 2 + Fibonacci-Zahl von 3) +
  +(Fibonacci-Zahl von 3 + Fibonacci-Zahl von 4))
 +((Fibonacci-Zahl von 3 + Fibonacci-Zahl von 4)
  +(Fibonacci-Zahl von 4 + Fibonacci-Zahl von 5))
```

```
= ((1 + ( 1 + 1 ))
  +((1 + 1) +(1 + Fibonacci-Zahl von 3)))
 +(((1 + 1)
   +(1 + Fibonacci-Zahl von 3))
  +((1 + Fibonacci-Zahl von 3)
  +(Fibonacci-Zahl von 3 + Fibonacci-Zahl von 4)))
= (3 + (2 + 3))
 +((2 + 3) + (3 + (2 + (Fibonacci-Zahl von 3 + 1))))
= 8 + (5 + (3 + (2 + ((1 + 1)+1))))
= 21
```

Alternativ als Baumdarstellung:



b. Der Algorithmus ist **rekursiv**.

c. **Statisch finit,** da die Beschreibung nur wenige Zeilen umfasst.
Nicht dynamisch finit, da für x<=0 der Algorithmus unendlich oft erneut aufgerufen wird und somit unendlich viele Ressourcen benötigt.
Nicht vollständig, da Spezialfälle nicht erfasst werden *(Werte für x <= 0 werden nicht abgefangen)*.
Deterministisch, da eindeutig im Ablauf und kein Zufall in Operanden & Operationen Anwendung findet.

d. Umsetzung mit einer Programmiersprache (Implementierung) und Ausführung des übersetzten Programms *(Prozess ist die Instanz eines Programms während der Ausführung).*

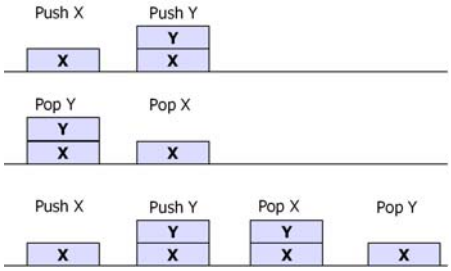
e. Die Laufzeit ist **exponentiell**. *Sie erzeugen mit jedem Aufruf des Algorithmus (ausgenommen für x=1 u. x=2) zweimal rekursive Aufrufe des gleichen Algorithmus, siehe Aufgabe a). Daraus ergibt sich ein $O(2^n)$. Die schlechte Effizienz geht insbesondere daraus hervor, dass der Algorithmus immer wieder die gleichen Folgen (nur um die letzten Elemente reduziert) berechnet. Ein iterativer Algorithmus (mit Schleife) könnte die gleiche Aufgabe mit linearer Ordnung realisieren. Die Ordnung des Algorithmus ist exponentiell, die Ordnung der Problemstellung (Erzeugen von Fibonacci-Zahlen) ist linear!*

Aufgabe 12: Stapelspeicher **

a. Push X
Push Y

b. Pop Y
Pop X

c. Push X
Push Y
Pop X
Pop Y



d. Z = X
X = Y
Y = Z

Aufgabe 13: Primzahlentest **

a. 2, 3, 5, 7, 11, 13.

b. **Ja**, der Sonderfall M<2 wird abgefangen.

c. **Ja**, Abbruchbedingungen werden immer erreicht.

d. **statisch finit**: Ja, da die Beschreibung endlich ist.
dynamisch finit: Ja, maximal wird ein Speicher in der Größe von M+2 benötigt.

e. *Eine mögliche Anwort:*

- Nach der Elimination der Vielfachen von 2 brauchen nur noch ungerade Werte von *i* berücksichtigt werden. *Damit würde gelten: i = 3, 5, 7, 9, 11, 13, 15 usw. Der Algorithmus würde etwa doppelt so schnell ablaufen.*
- Für i werden nur Zahlen berücksichtigt, die bisher schon als prim gelten. *Damit würde gelten: i=2, 3, 5, 7, 11, 13, 17, 19, 23 usw. Damit werden alle bereits gefundenen Vielfachen ausgeschlossen. Dies entspricht dem original Algorithmus nach Eratosthenes.*

f. **Quadratisch**, aufgrund der Verschachtelung der Schleifen i und j.

6.4 Zahlendarstellung

Aufgabe 14: Umrechnung von Hexadezimalzahlen *

a. `#0E8C3A = 0000 1110 1000 1100 0011 1010b`
`#E3170D = 1110 0011 0001 0111 0000 1101b`
`#191970 = 0001 1001 0001 1001 0111 0000b`
`#0276FD = 0000 0010 0111 0110 1111 1101b`
Big Endian: von links (mathematische Schreibweise).

b. #0276FD : Rot #02 = 2 , Grün #76 = $7 \cdot 16 + 6 = 118$, Blau #FD = $15 \cdot 16 + 13 = 253$.

c. #E3170D : Rot (#E3): 227/255 = 89 % , Grün (#17): 23/255 = 9 %,
Blau (#0D): 13/255 = 5% .

d. $0{,}75 \cdot 255 = 191.25$ (Ohne Komma weiter gerechnet): 191 = 1011 1111b = #BF

Aufgabe 15: Hexadezimalzahlen *

a. `12` *(3 Hexziffern à 4 Bit)*

b. $BEE = 3054, $ACE = 2766, $FEE = 4078

Aufgabe 16: Vierer-System *

a.
Symboltabelle *(Abbildung von Binärsymbolen auf das Vierersystem)*:

```
00 ↔ 0
01 ↔ 1
10 ↔ 2
11 ↔ 3
```

b. Damit lässt sich die Kodierung mit 2er Gruppenbildung einfach durchführen:

1031_4, 3212_4, 0123_4, 2121_4

Aufgabe 17: Wertebereiche *

Bits / Wertebereiche	Vorzeichen-Betrag	Einerkomplement	Zweierkomplement
5	-15 ... 15	-15 ... 15	-16 ... 15
8	-127 ... 127	-127 ... 127	-128 ... 127
10	-511 ... 511	-511 ... 511	-512 ... 511
13	-4095 ... 4095	-4095 ... 4095	-4096 ... 4095

Aufgabe 18: Zweierkomplement **

a. 12 = 00001100b *(Komplement: 11110011b + 1b)* => -12 = 11110100b

b. 11001101b = ? *(Komplement: 00110010b + 1b = 00110011b)* ==>11001101b = - 51

c.
```
   00111010b   =  58
+  00001101b   = +13
=  01000111b   =  71, korrekt, da kein Überlauf innerhalb des Wertebereichs
```

d. *Differenz = Addition mit negiertem Subtrahenden*
Subtrahend 00111010b *(Komplement:* 11000101b *+ 1b):* 11000110b *(= -58)*

```
     01001101b
+    11000110b
 (1) 00010011b        = 19 (Gegenrechnung: 77 - 58 = 19),
```

Ergebnis ist korrekt, da das Ergebnis im Wertebereich liegt.
Der Überlauf erfolgt im stetigen Bereich des Zahlenkreises.

e. *Zweierkomplement ist ungünstige Darstellung zur Berechnung des Produkts, d.h. die Faktoren müssen zunächst in die Vorzeichen-/Betragsdarstellung überführt werden.*

Beträge:
`|1001b| = (`*Komplement*`: 0110b + 1b) = 0111b = 7, V=1` *Vorzeichen*`!`
`|0101b| = 5 (kein Vorzeichen)`

`Berechnung des Betrags:`

```
0111b · 0101b
      0111b
 +     0111b
 =    100011b  = 35 = | -7 · 5 |
```

Ergebnis ist wegen des Vorzeichens negativ ==> Zweierkomplementbildung; 35 ist mit 4 Bit nicht darstellbar ==> Ergebnis mit 8-Bit-Zweierkomplement:

`-00100011b = (`*Komplement*`: 11011100b + 1b) = 11011101b = -35`

Alternative: Erweitern auf 8-Bit-Zweierkomplement mit anschließender direkter Multiplikation der Faktoren im Zweierkomplement:

```
11111001b · 00000101b
          11111001b
      +    11111001b
 =       100110111101b  Ergebnis in den letzten 8 Bit:  11011101b
```

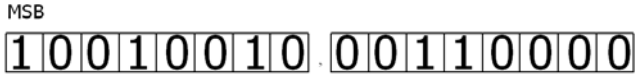
Achtung: Diese Art der Multiplikation funktioniert nur innerhalb der angewandten Bitlänge der Operanden (hier 8 Bit). Eine Multiplikation zweier 8-Bit-Zahlen würde 16-Bit-Operanden erfordern, damit die unteren 16 Bit das korrekte Ergebnis liefern.

f. Für die binäre Darstellung der Zahl 35 sind 6 Bits nötig; zzgl. Vorzeichenbit ergibt eine 7-Bit-Darstellung, sinnvoll ist hier eine 8-Bit-Darstellungsform, siehe e). *Allgemein gilt, dass bei einer Multiplikation von n · n Bits, das Ergebnis bis zu 2 · n Bits aufweisen kann.*

Aufgabe 19: Zahlendarstellung **

a. *Kleinste negative Zahl:* `11111111,11111111` *= -127,99609375*
größte positive Zahl: `01111111,11111111` *= +127,99609375*
- 128 ... 128

b.



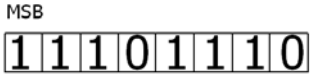
c. 3,322 Bit

d. 4 Bit

e. Der negative Zahlenbereich wird um eins erhöht, da die negative Darstellung der Zahl Null wegfällt.

f. - 129 ... 128

g. *-00010010b = (Komplement: 11101101b + 1b) = 11101110b = -18.*



h. *Achtung, wir müssen zurück zur Vorzeichen und Betragsrechnung wechseln!*

Z = -18,1875 = `10010,0011b`

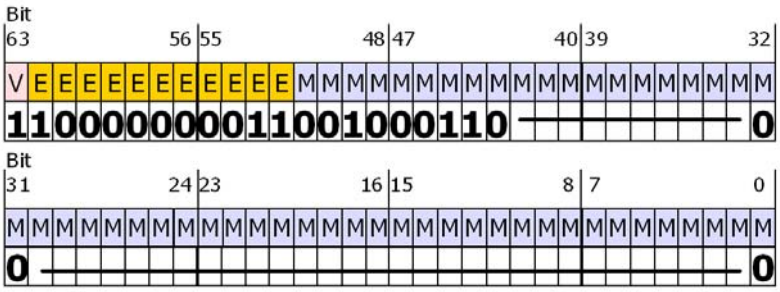
Komma um 4 Stellen nach links schieben ergibt:

Z = $1,00100011b \cdot 2^4$

Mantisse (8-Bit): `00100011b`

i. E = 4 + 1023 = 1027 = `10000000011b`

j.



k. $C0, $32, $30, $00, $00, $00, $00, $00

l. 15-16 Stellen *(die 53 Mantissen-Bits erlauben Zahlen von* 0 ... 2^{53}-1 ≅ $9.0 \cdot 10^{15}$*).*

m. -2^{1024} ... 2^{1024}
aufgerundet gemäß der exponentiellen Darstellung (E = 2046 - 1023 = 2^{1023})
-1,999999999999999... · 2^{1023} bis +1,999999999999999... · 2^{1023}

Aufgabe 20: Gleitkommazahlen **

a. **-11.75**
Binär: `1011.1100b` + Vorzeichen

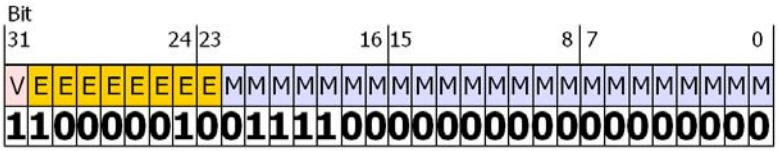
b. `M = 01111000b`

Normalisieren - Komma verschieben: `1.0111100b` · 2^3
Hidden-Bit löschen.

c. 2^3, Bias = 127:

`Exponent mit Bias: E = 3+127 = 130 = 10000010b`

d. *Übertragen in die Maske - Vorzeichen nicht vergessen!*



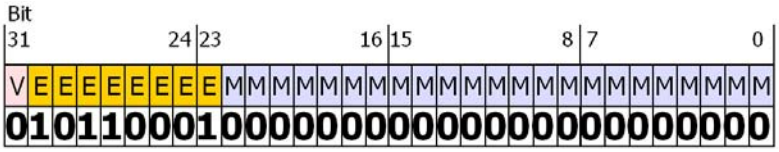
e. **$1.126 \cdot 10^{15}$**
Suchen des optimalen Exponenten $10^{15} \cong 2^{50}$
(Exponent mal 3.322: $15 \cdot 3.322 = 49{,}83$)
$2^{50} \cong 1.125899991 \cdot 10^{15}$

```
Exponent mit Bias: E = 50+127 = 177 = 10110001b
```

f. $2^{50} \cong 1.125899991 \cdot 10^{15}$, d.h. für 4 signifikante Stellen ausreichend; die binäre Darstellung ergibt sich zu $1.0 \cdot 2^{50}$.

g. Darstellung aus Teilaufgabe f) ist bereits normiert!

h.



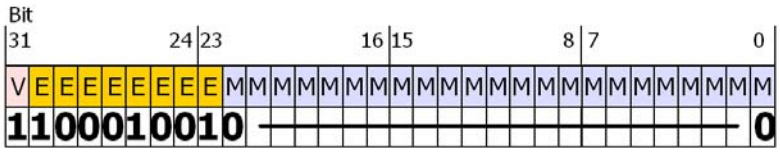
Aufgabe 21: Weitere Gleitkommazahlen **

a. $f_0 = 0$ und $f_1 = 1.00 \cdot 2^{-126}$

b. Obwohl f_1 näherungsweise null ist, sind f_0 und f_1 unterschiedliche Zahlen (Vergleich nur mit Näherung möglich).

c. $-1.0 \cdot 2^{10}$

d.
```
E = 10+127 = 137 = 128+9 = 128+8+1 = 1000 1001b
M = 0, V = 1
```



e. $C4, $80, $00, $00

6.5 Zeichendarstellung

Aufgabe 22: Kodierung von Zeichenfolgen *

a. 6 Bit. *(26+3)*2+2 = 60 Zeichen gesamt.*

b. ld(60) = 5.91 Bit.

Aufgabe 23: Null-Terminierte Strings *

a. $48, $61, $6C, $6C, $6F, $20, $47, $44, $49, $00.

b. 10. *9 für den Text, 1 für die Terminierung.*

c. 20. *18 für den Text, 2 für die Terminierung.*

d. Vorteil: Zeichenfolgen sind nicht in der Länge begrenzt. *Bei einem String mit Längenangabe ist die maximale Länge vorgeschrieben, z.B. bei einer Längenangabe mit 8-Bit: 255 Zeichen, bei 16 Bit 65535 Zeichen.*
Nachteil: Die Längeninformation der Zeichenfolge ist nicht vorhanden. *Um die tatsächliche Länge der Zeichenfolge zu ermitteln, muss der gesamte String nach dem Null-Zeichen abgesucht werden.*

Aufgabe 24: UTF- und UCS-Kodierung **

a. Für ’ä’ (0xE4 = 11100100) wird eine 2-Byte Kodierung erforderlich:
110x xxxx 10xx xxxx (11 Bit). 0xE4 = 00011100100 auf 11 Bit erweitert, eingefügt in die Kodierung folgt:

UTF-8: 1100 0011 1010 0100

b. 0xC3, 0xA4

c. *Mit ’B’ = 0x42 und das ’r’ = 0x72 ergibt sich*:

0x42, 0xC3, 0xA4, 0x72

d. 2 Bytes (33 %) werden gespart. *Der Speicherbedarf entspricht 4 Bytes gegenüber 6 Bytes nach UCS-2.*

e. 4 Bytes. *Das Schema lässt sich aus Bild 2.23 ableiten:*

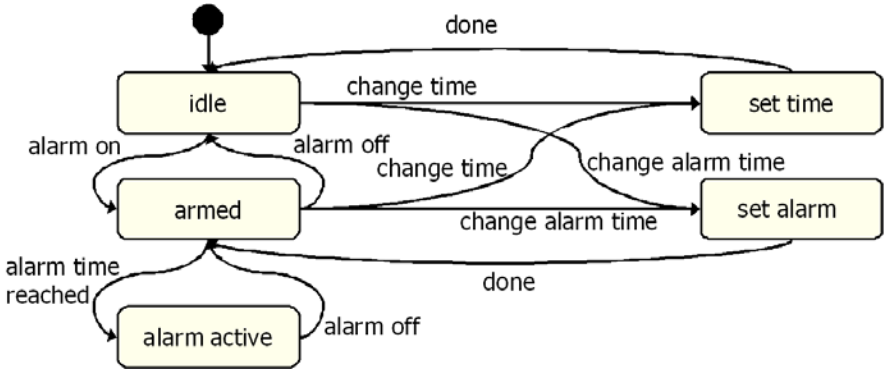
```
UTF-8:
0xxx xxxx = 7 Bit
110x xxxx 10xx xxxx = 11 Bit
1110 xxxx 10xx xxxx 10xx xxxx = 16 Bit
1111 0xxx 10xx xxxx 10xx xxxx 10xx xxxx = 21 Bit
```

f. UCS-4 = 32 Bit: max. 2^{32} = 4294967296
(der geltende Standard unterstützt aber nur 10FFFFh Zeichen)

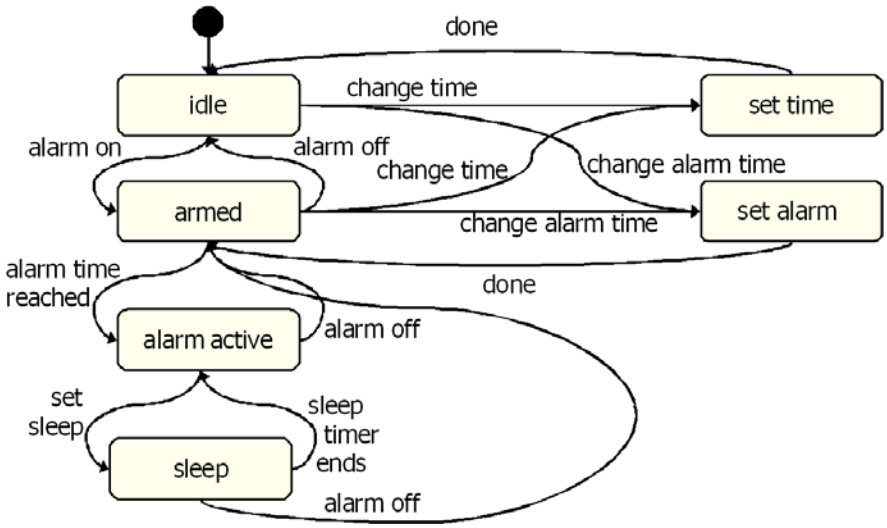
6.6 Modellierung und Formale Sprachen

Aufgabe 25: Wecker **

a.



b.



c. *Der Trick: Ausgehend von einem 1-Tasten-Gerät, sucht man Zustände, welche für die Zustandsübergange mehr als eine Taste benötigen:*

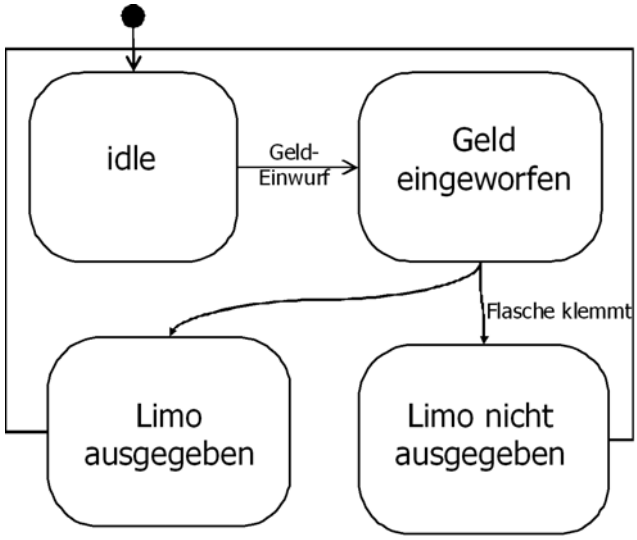
alarm active: alarm off und set sleep ==> man braucht min. 2 Tasten
idle: alarm on, change time, change alarm time ==> 3 Tasten
armed: alarm off, change time, change alarm time ==> 3 Tasten

Folglich werden drei Tasten benötigt. Eine mögliche Zuordnung:

Taste 1: alarm off, alarm on, done
Taste 2: change time, set sleep
Taste 3: change alarm time.

Aufgabe 26: Getränkeautomat *

a.



b. **Nein**, da der Zustandsübergang „Flasche klemmt“ zufällig (nicht vorherbestimmt) auftritt. *Zumindest ist der normale Automatenkunde nicht in der Lage, das Klemmen der Flasche aufgrund verschiedener physikalischer Parameter (Ausgangslage der Flasche, temperaturbedingte Ausdehnung der Transportwege etc.) vorherzusagen.*

Aufgabe 27: Analyseautomat für ein Vierer-System **

a. *idle*, 0, 1, 00, 01, 10, 11.

b.

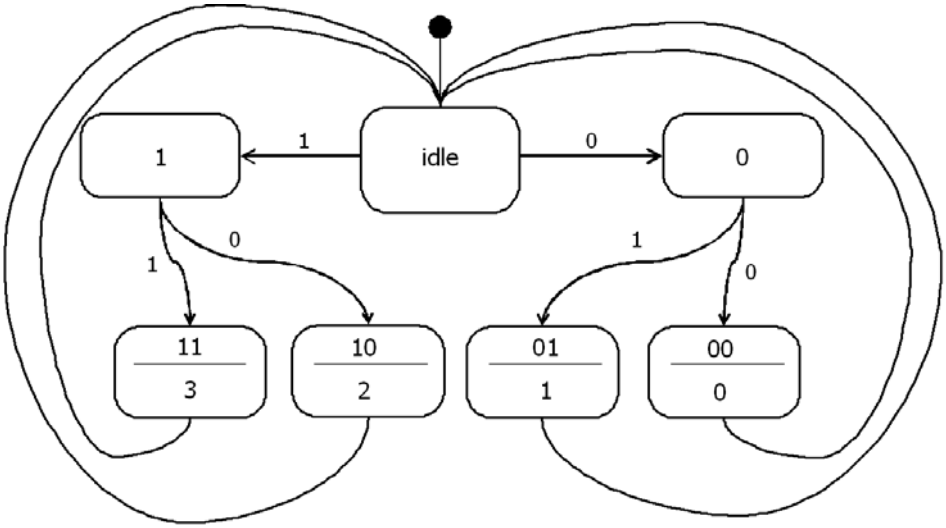
	Z_1 Z_2 Z_3	Ausgabe
idle	0 0 0	-
0	0 1 0	-
1	0 1 1	-
00	1 0 0	0
01	1 0 1	1
10	1 1 0	2
11	1 1 1	3

mögliche Lösung als Wahrheitstabelle

Zustand	Ausgabe
idle	-
0	-
1	-
00	0
01	1
10	2
11	3

Symboltabelle

c.

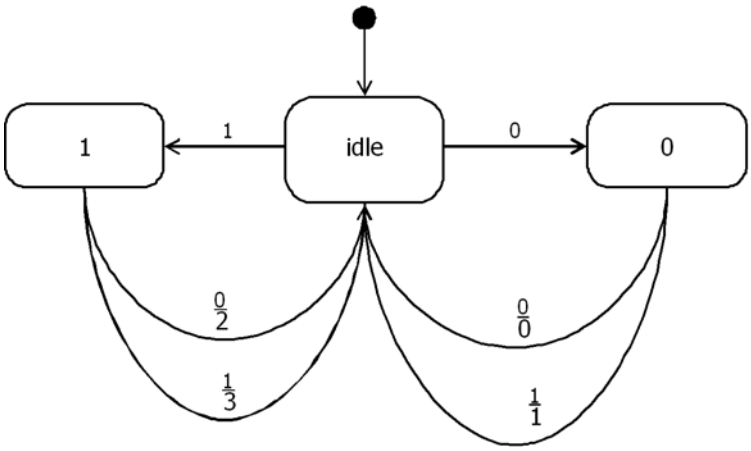


d. Die Ausgänge werden durch die Zustände bestimmt.

e. 00, 01, 10, 11.

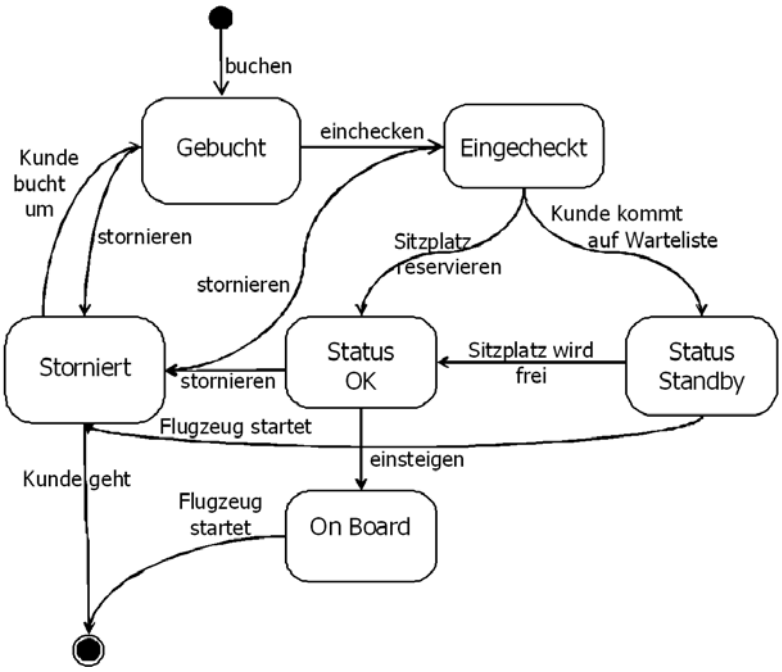
f. 0 auf idle, 1 auf idle, je nach Eingang 0 oder 1

g.



Aufgabe 28: Zustandsdiagramm Buchungssystem ***

a.

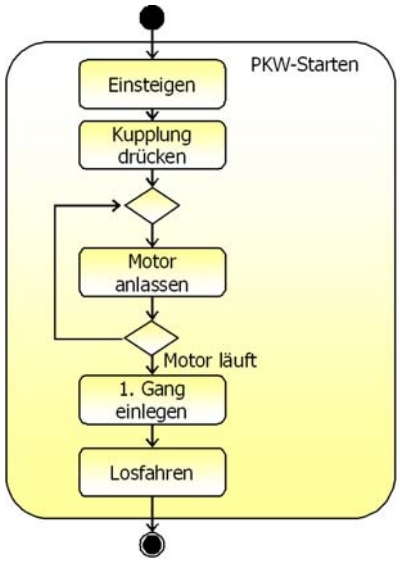


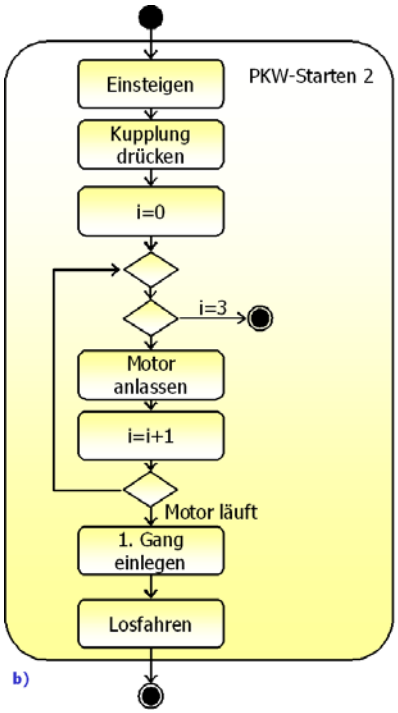
b. Der Zustandsübergang „stornieren“ von *Status-OK* auf *Storniert*! *Nur wenn ein Kunde seinen Platz wieder frei gibt (storniert), kann ein anderer Kunde vom* Status Standby *auf* Status Ok *wechseln. Zwar stellt das Zustandsdiagramm nur die Sichtweise eines einzelnen Kunden dar; in der Praxis existiert für jeden Kunden ein eigenständiges Zustandsdiagramm.*

Aufgabe 29: Modellierung **

a.

b. *Einführen einer Zählervariablen i.*



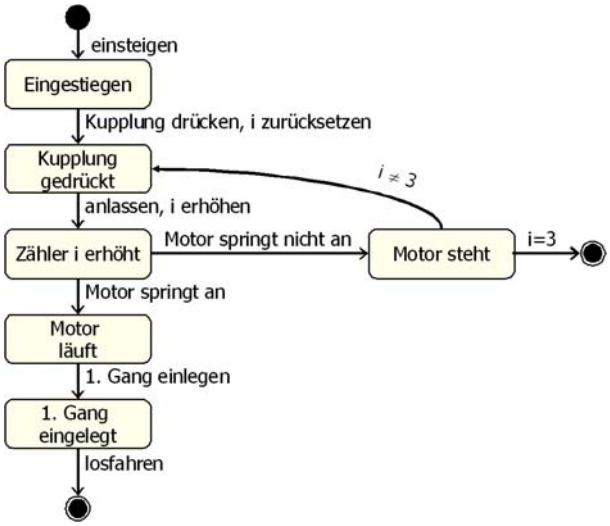


c. Zustände: Eingestiegen, Kupplung gedrückt, Motor läuft, Motor steht 1. Gang eingelegt, Zähler erhöht.

Zustandsübergänge: einsteigen, Kupplung drücken, anlassen, Motor springt an, Motor springt nicht an, 1. Gang einlegen, losfahren, falls Zähler = 3, falls Zähler ≠ 3, Zähler zurücksetzen, Zähler erhöhen.

Die Tätigkeiten (Aktivitäten) bewirken Änderungen der Systemzustände und sind somit wie Ereignisse als Zustandsübergänge einzuzeichnen. Für die Zählung benötigen wir die zusätzlichen Zustände „Zähler erhöht“ und „Motor steht“, mit dessen Zustandsübergängen wir den Zähler auf drei überprüfen. Ohne diesen Zustand würde ein nicht deterministischer Automat entstehen!

d.

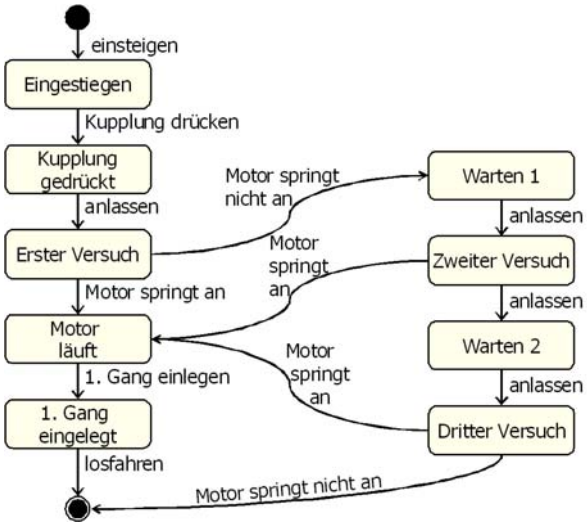


*Alternative Lösung **ohne** Zähler (der Zähler wird durch Zustände realisiert):*

c. Zustände: Eingestiegen, Kupplung gedrückt, Motor läuft, 1. Gang eingelegt, Erster Versuch, Zweiter Versuch, Dritter Versuch, Warten 1 und Warten 2

Zustandsübergänge: einsteigen, Kupplung drücken, anlassen, Motor springt an, Motor springt nicht an, 1. Gang einlegen, losfahren,

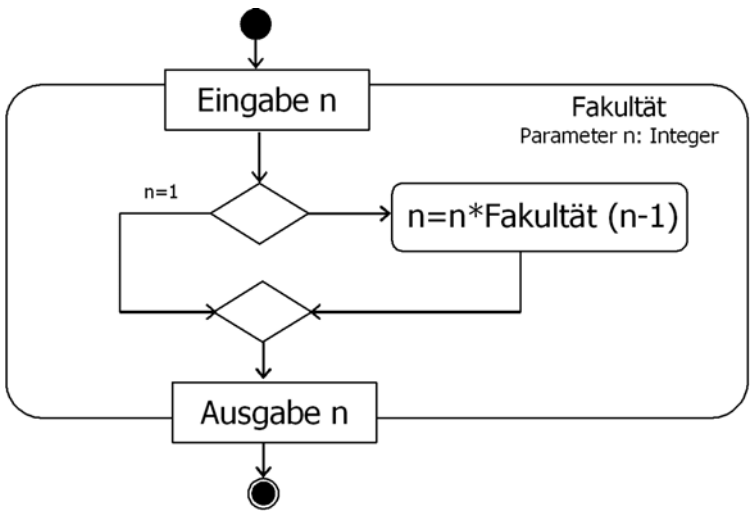
d.



***Anmerkung**: Das Aktivitätsdiagramm ist für diese Aufgabenstellung sicherlich vorzuziehen. Beim Zustandsdiagramm werden Aktivitäten in Zustände (eigentlich: Zustandsübergänge) übergeführt, was sich negativ auf das Verständnis des Modells auswirkt. Die Aufgabe zeigt einerseits, dass beide Darstellungen ineinander überführbar sind. Andererseits ist es wichtig, die richtige Darstellungsform zu wählen: Aktivitätsdiagramme stellen den Ablauf und die Interaktion in den Vordergrund, wohingegen Zustandsdiagramme besser die Reaktion auf Ereignisse darstellen.*

Aufgabe 30: Aktivitätsdiagramm **

a.

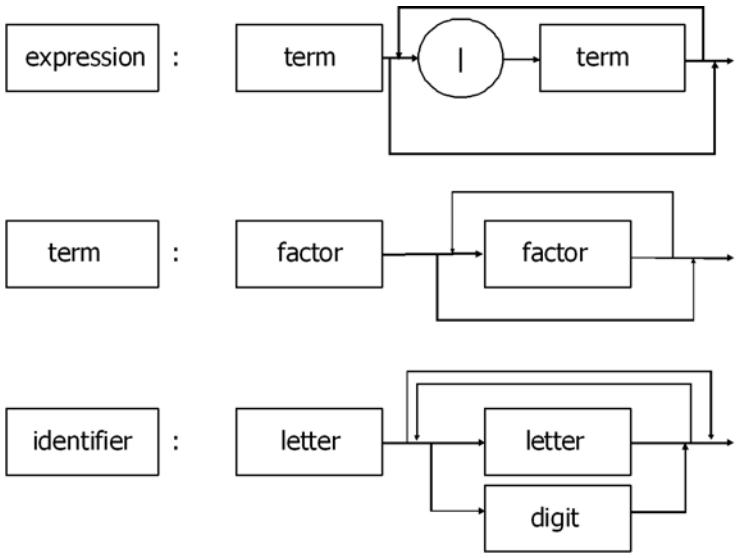


Anmerkung: In dem Lösungsbeispiel wird die Variable n sowohl für die Eingabe als auch für die Ausgabe des Ergebnisses verwendet. Dies wird erreicht, indem beide Zweige (n=1, sonst) ihr Ergebnis in n zurückliefern. Selbstverständlich wäre auch die Verwendung einer zweiten Variablen für die Aufnahme des Ergebnisses richtig.

b. Nein, da eine sequentielle Abhängigkeit durch die Rekursion gegeben ist.

Aufgabe 31: EBNF und Syntaxgraph **

a.



Es ist nur eine mögliche Lösung dargestellt!

b. Nein! Die Darstellungen sind semantisch ineinander überführbar!

Aufgabe 32: EBNF-Syntaxdefinition ***

a.
```
variable = letter { digit | letter }.
letter = "a" | ... | "z" | "A" | ... | "Z".
digit = "0" | "1" | ... "9".
```

b.
```
number = ["-"] digit { digit }.
```

c.
```
syntax = variable "=" term [("+"|"-") term] ";".
term = variable | number.
```

d. Ja, wegen vollständiger Substituierbarkeit *(rechte Seite enthält nur Terminalsymbole). Alternative Begründung: Es tritt keine Rekursion auf.*

Beispiel vollständige Substitution:

```
syntax = ("a" | ... | "z" | "A" | ... | "Z")
     { ("0" | ... | "9") | ("a" | ... | "z" | "A" | ... | "Z" ) }
     "=" ((("a" | ... | "z" | "A" | ... | "Z")
     { ("0" | ... | "9") | ("a" | ... | "z" | "A" | ... | "Z" ) })
     | ("0" | ... | "9") { "0" | ... | "9"} )
     [("+"|"-") ((("a"|...|"z"|"A"|...|"Z")
     { ("0" | ... |"9") | ("a" | ... | "z" | "A" | ... | "Z" ) })
     | ("0" | ... | "9") { "0" | ... | "9"})] ";".
```

6.7 Digitaltechnik

Aufgabe 33: Grundschaltungen *

a.

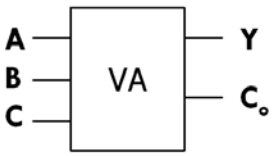
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

b.

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Aufgabe 34: Volladdierer ***

a.



A	B	C	Y	C_o
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

b. Daraus lassen sich die Booleschen Gleichungen ablesen:

$C_0 = A \cdot B \cdot \overline{C} + (A+B) \cdot C$

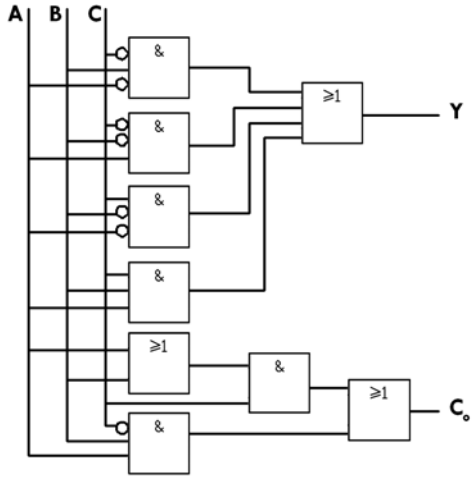
$Y = \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C + A \cdot B \cdot C$

Alternativ könnte in der Schaltung statt UND/ODER auch XOR eingesetzt werden:

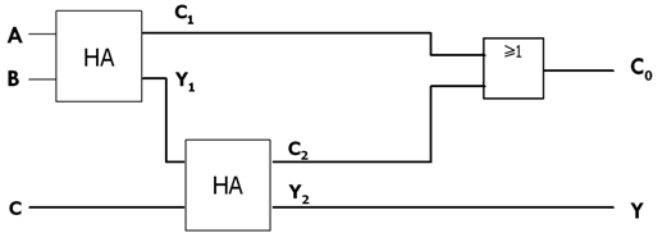
$Y = A \oplus B \cdot \overline{C} + \overline{A \oplus B} \cdot C$

Hinweis: die Addition ist assoziativ, d.h. die Reihenfolge von A,B und C kann beliebig vertauscht werden. Das führt zu äquivalenten Gleichungssystemen!

c.



d. Reihenschaltung zweier HA:



e. Vorteil: Einfacheres Schaltungslayout *(HA kann als „fertiges" Modul angesehen werden)*
Nachteil: größere Gatterlaufzeit. *(Zur Berechnung von Y sind 4 Gatter anstelle von zwei Gattern erforderlich).*

Aufgabe 35: Logikschaltung *

a.

A	B	C	D	Y
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

b. $Y = \overline{A} \cdot B + A \cdot \overline{B} = A \oplus B$

c. XOR

d. Sie ist symmetrisch.

Aufgabe 36: Ganzzahlen-Division **

a. 3.*(Division durch 2 entspricht einer Subtraktion des Exponenten um eins* $2^4/2 = 2^{4-1}$. *Folglich genügt es, die untere Datenleitung wegzulassen).*

b.

D_0	D_1	D_2	D_3	Y_0	Y_1	Y_2
X	0	X	X	0	X	X
X	1	X	X	1	X	X
X	X	0	X	X	0	X
X	X	1	X	X	1	X
X	X	X	0	X	X	0
X	X	X	1	X	X	1

oder alternativ als vollständige Wahrheitstabelle (alle Kombinationen von D_x*)*

D_3	D_2	D_1	D_0	Y_2	Y_1	Y_0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	0	1	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	1	0	1
1	0	1	1	1	0	1
1	1	0	0	1	1	0
1	1	0	1	1	1	0
1	1	1	0	1	1	1
1	1	1	1	1	1	1

c. Y0 = D1, Y1 = D2, Y2=D3

d. Schieben um 1 Bit nach rechts.

Aufgabe 37: Bitweise Logik ***

a. b1 = 9, b2 = 13, b3 = 5

b. b1 = 12, b2 = 13, b3 = 5

c. b1 = 8, b2 = 17, b3 = 7

6.8 Rechnertechnik

Aufgabe 38: ALU **

a.

Register A				Register B				Ergebnis					
A	A	A	A	B	B	B	B	A	A	A	A	C	Z
0	1	0	0	1	1	1	1	0	1	0	0	0	0

b.

Register A				Register B				Ergebnis				C	Z
A	A	A	A	B	B	B	B	A	A	A	A	C	Z
0	1	1	1	1	0	0	1	0	0	0	0	1	1

c.

Register A				Register B				Ergebnis				C	Z
A	A	A	A	B	B	B	B	A	A	A	A	C	Z
0	0	0	0	0	0	0	1	0	0	0	1	0	0

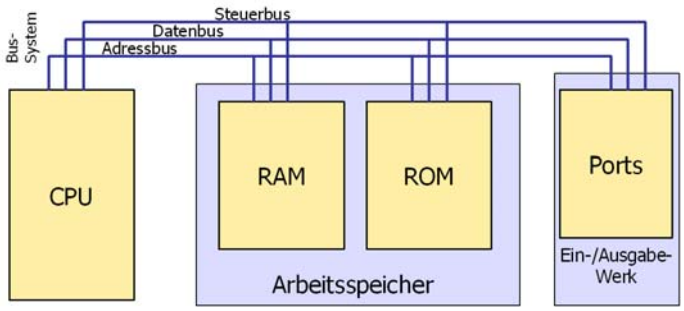
d.

Operation	Register A				Register B				Ergebnis				C	Z	
	A	A	A	A	B	B	B	B	A	A	A	A	C	Z	
NOT A	0	1	0	1	1	0	1	1	1	0	1	0	0	0	
INC A	1	0	1	0	1	0	1	1	1	0	1	1	0	0	
ADD A,B	1	0	1	1	1	0	1	1	0	1	1	0	1	0	= 6 = 11-5

Aufgabe 39: 8-Bit-Rechner *

a. Arbeitsspeicher (RAM/ROM), Bussystem, Ein-/Ausgabeports, CPU

b.



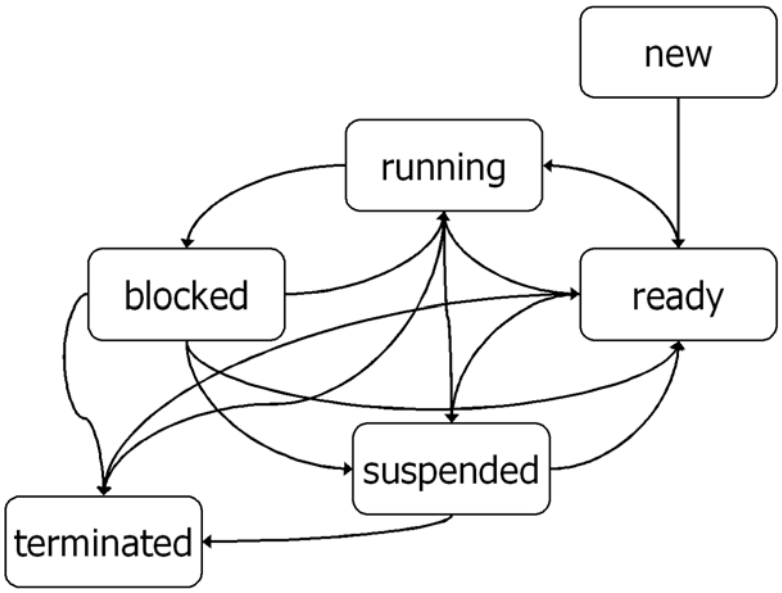
c. Nein, der Pipeline-Betrieb hat keine Auswirkung auf die Verschaltung.

Aufgabe 40: Prozessmanagement **

a. ready (rechenbereit), running (rechnend), blocked (blockiert).

b. Weiter sinnvolle Zustände: new (neu erzeugt), suspended (auf bestimmte Zeit angehalten), terminated (beendet).

c.



Aufgabe 41: Cache-Speicher **

a. Die Segmentgröße des VA-Caches ist 16 Bit - 5 Bit (32 Datenbytes pro Block) = 11 Bit, d.h. bei einem VA-Cache müssen für alle 64 Blöcke je 11 Adressbits verglichen werden. Insgesamt werden 11·64 = 704 Komparatoren benötigt.

b. Beim Direkt Mapped Cache wird nur der Adressbereich (Tag) des ausgewählten (direkt abgebildeten) Cache-Blocks verglichen. Damit reduziert sich die Anzahl der notwendigen Adressbits um weitere 6 (64 Blöcke werden durch 6 Leitungen adressiert). Der Tag enthält nur:
16-5-6=5 Bits, also 5 Komparatoren.

c. *Zwei Antworten genügen:*

- LRU *(least recently used; der am längsten ungenutzte Block wird entfernt).*
- LFU *(least frequently used; der am wenigsten häufig genutzte Block wird entfernt).*
- Zufällige Auswahl*(Pseudo-Random; zufällig werden Blöcke entfernt).*

Aufgabe 42: Virtueller Speicher **

a. 2GB / 4K = 524288 Seiten

b. 524288 · 4 Bytes = 2 MByte.

c. Vorteil der 1-MByte-Speicherseite: weniger Verwaltungsaufwand
Nachteil: Jeder Prozess benötigt mindestens 1 MByte physikalischen Speicher *(auch wenn der Prozess selbst vielleicht nur 4 KByte benötigen würde).*

d. Vorteile virtueller Speicher: Prozesse können mehr Speicher verwenden als physikalisch vorhanden ist. Solange physikalischer Speicher vorhanden ist, muss nicht ausgelagert werden.
Vorteil Swapping: erheblich einfacher zu realisieren.