1 Information, Nachrichten

- Informatik als Wissenschaft von der maschinelle Verarbeitung von Information

  geeignete Darstellung von Information in einem Rechner

  Eine abstrakte Information wird durch einen konkrete Nachrichten übermittelt. Einen Nachricht ist eine Sequenz von eichen aus einem vorgegebenen Zeichenvorrat.

- Eine Nachricht kann verschieden Informationen enthalten und/oder unterschiedlich interpretiert werden, Sie kann an einen und mehrere Empfänger geschickt werden.

- Es gibt eine Informationsvorschrift f, die mit dem Absender vereinbart ist

- Nachrichtenträger:

  Frequenz, Amplitude, Stromstärke, elektrische Spannung

- der zeitliche Verlauf einer physikalischen Größe die eine Nachricht überträgt heißt Signal, die Kenngröße heißt Signalparameter

  wenn der Signalparameter nur endlich viele Werte annehmen kann heißt das Signal digital

- Ein Zeichen wird definiert als Element einer endlichen Menge unterscheidbarer "Dinge". Diese Menge wird als Zeichenvorrat bezeichnet

  Ein Zeichenvorrad, der eine Reihenfolge hat wird als Alphabet bezeichnet

2 Codierungen

- Eine Vorschrift zur Abbildung eines Zeichenvorrats in einen andere Zeichenvorrat heißt Code

- in technischen Codes werden Buchstaben, Ziffern und ggf. andere Zeichen fast immer durch Binärworte codiert

  bei Computern werden meistens Erweiterungen von ASCII-Code verwendet (Alle ASCII Zeichen sind kompatibel)

3 Betriebssysteme

- Sammlung von Programmen zur Verwaltung der Hardware

- Schichtmodell:

  Schicht nutzt ausschließlich Dienste(Schnittstellen) der direkt darunter liegender Schicht, kein bereichsübergreifender Zugriff

- Aufgaben

  - Verwaltung des Hauptspeichers

- Verwaltung des Zugriffs auf der Peripherie
- Steuerung des Ablaufs von Programmen
- Zugriffsschutz
- Abrechnungsinformation

- Im Detail

  Verwaltung einer Festplatte

  - Blockweise Übertragung von externem Speicher vom/zum Rechner
  - Dateien belegen i.d.R. mehrere Blöcke auf der Festplatte (meist nicht konsekutiv)
  - Zusätzlich zum Inhalt wird für jede Datei weitere Information verwaltet
  - Anlegen Wiederfinden, Ändern, Löschen einer Datei
  - Verwalten der Dateiblöcke
  - Koordination der Plattenzugriffe (Minimale Bewegungen des Schreib-/Lesekopfs)

  Prozessverwaltung:

  - Prozess: Eingenständer Ablauf eines Programms auf einem Rechner
  - Mehrprozesssysteme (Mehren Programme von einem oder mehreren Benutzen laufen Parallel auf dem selben Rechner)
  - Zuteilung von Ressourcen an Prozesse
  - Schutz der Prozesse voreinander vor gegenseitiger Störung
  - Zuteilung von Rechenzeit an Prozesse am Beispiel von Zeitscheiben (time slicing)
  - Prinzip:

    Prozess bekomme den Prozessor exklusiv für einen bestimmten Zeitraum, wird der Prozessor einem anderen Prozess zugeteilt

    quasi parallele Abläufe mehrere Prozesse auf einem Prozessor

- Speicherverwaltung:

  - Mögliche Probleme: Prozesse greifen auf die selbe Speicherstelle zu und überschreiben gegenseitig Information
  - Jeder Prozess bekommt seinen eigenen Speicher, der physikalisch von den Speichern andere Prozesse getrennt wird(jeder Prozess hat den Eindruck den Hauptschuldig exklusiv zu besitzen)
  - Speicher wird in Pages fester Größe aufgeteilt
  - In einer Tabelle wird jeder Seite eines Prozesses eine Seite im physikalischen Speicher(Kachel)
  - Jeder Prozess kennt nur eine logischen Adressen

- Betriebssystem ordnet logische Adresse zu physikalischen Adressen zu
- Speicher logisch größer als physikalisch
- Seitengröße=Blockgröße der Festplatte
- Jeder Seite des virtuellen Speichers wird ein Block der Festplatte zugeordnet
- Seitentabelle gibt an ob und ggf. in welcher Kachel des phys. Speichers die Seite liegt, oder in welchem Block der Festplatte
- benötigte Seite nicht im phys. Speicher $\Leftarrow$ Holen von Platte, Auslagern einer nicht benötigten Seite auf Platte und Laden der benötigten in die frei gewordenen Kachel

4 Rechnerarchitektur

4.1 Neumann Architektur

- abstrakte, allgemein einsetzbare Rechnerarchitektur
- wird von heutigen Rechner umgesetzt
- abstrakte Sichtweise
- dient dem grundlegenden Verständnis der Funktionsweise eines Computers und der Abarbeitung von Software
- zur Lösung eines Problems muss von außen ein Programm eingegeben und im Speicher abgelegt werden
- Programme, Daten inkl. Zwischen- und Endergebnisse nutzen den Selben Speicher
- Speicher inkl. in gleich große Zellen unterteilt, die fortlaufend nummeriert sind
- über die Adresse eine Speicherzelle kann deren Inhalt gelesen oder geschrieben werden
- Ausführung eines Programms wird durch das Steuerwerk gesteuert
- Aufeinander folgende Befehle eines Programms werden in aufeinanderfolgenden Speicherzellen abgelegt
- das Ansprechen des nächtens Befehls geschieht vom Steuerwerk durch erhöhen der Befehlsadressen um eins
- durch Sprungbefehl kann von der Bearbeitung der befehle in der gespeicherten Reihenfolge
- Es gibt mindestens:
  - arithmetische Befehle wie Vergleiche, logisches und, oder usw.
  - Transportbefehle, z. B. von Speicher zum Rechenwert und für die Ein-/Ausgabe

  - bedingte Sprünge

- Der Speicher:
  - für den Zugriff auf den Speicher existieren zwei Leitungen(Busse): Adressbus und Datenbus
  - Jede Speicherzelle umfasst "Wortbreite" Bits
  - Auf dem Adressbus wird die Adresse "Nummer" der Speicherstelle übermittelt, auf die zugegriffen werden soll
  - soll im Speicher geschrieben werden, wird die adressierte Speicherzelle mit dem Wert auf dem Datenbus überschrieben
  - soll im Speicher gelesen werden, wird der wert der adressierten Speicherstelle auf den Datenbus geschriebenen

- Das Steuerwerk:
  - steuert die Ausführung eines Programms
  - hold Befehle nacheinander aus dem Speicher(fetch) in das Befehlsregister
  - Jeder Programmbefehle besteht aus zwei Teilen:
    Der Operationsteil legt fest, was gemacht werden soll
    der Adressteil bestimmt, auf welche Daten der Befehl anzuwenden ist
  - jeder Programmbefehl wird decodiert (decode)
  - danach ausgeführt (execute) in dem entsprechen Funktionseinheiten(z.B Rechenwerk , Ein-/Ausgabe aktiviert werden)
  - Das Befehlszählerregister erhöht die Speicheradresse des aktuell ausgeführten Befehls
  - Nach Ausführung eines Befehls wird das Befehlsregister geändert und der nächsten Befehl wird geholt
    Befehlsregister +1(falls kein Sprungbefehl)
    Sprungziel sonst
  - Sind Daten für eine Befehlsausführung im Speicher abgelegt, so müssen diese vor Ausführung geholt bzw nach Ausführung gespeichert werden:
    zuerst wird bestimmt auf welche Speicherstelle zugegriffen werden muss (Adressbrechung: Adresse)
    dann wird der Speicherzugriff(read/write durchgeführt)
  - Vollständiger Befehlsfluss
    $fetch \Leftarrow decode \Leftarrow address \Leftarrow read \Leftarrow execute \Leftarrow write$

- Das Rechenwerk:
  - Akkumulator dient als Speicher für Zwischenergebnisse
  - Beim klassischen Von-Neumann-Modell enthält der Akkumulator immer den ersten Operanden (evlt. vorher aus dem Speicher laden!) und das Ergebnis
  - Die Rechenlogik führt arithmetischen Operationen aus

4.2 Reale Rechnerarchitekturen

- Laden größerer Speicherbereiche in speziellen Speicher (Cache)
- Bearbeitung mehrere Befehle gleichzeitig (Pipelining: Parallelität innerhalb der Befehlsausführung durch Überlagerung von Phasen aufeinanderfolgender Befehle)
- Befehlszyklus ohne Pipelining:

  $fetch \Leftarrow decode \Leftarrow address \Leftarrow read \Leftarrow execute \Leftarrow write$
- Befehlszyklus mit Pipelining:

  gleichzeitiges Nutzen "disjunkter" Teile des Rechenwerks (Ganzzahlarithmetik/Gleitkor
- Speicherhierarchien

  sehr schnelle Cache Speicher

  "langsamere" Hauptspeicher

  "Langsamer" Plattenspeicher
- Parallele Prozessoren mit gemeinsamen Speicher (shared memory)
- Parallele Prozessoren mit verteilen Speicher (distributed memory)

5 Zahlensysteme

- Dezimal(Basis 10)
  - $d = d_{n-1} * 10^{n-1} + \cdots + d_0$
    $d_i \in \{0, \ldots, 9\}$
- Basis n
  -
  - $z = z_{n-1} * b^{n-1} + \cdots + z_0$
    $z_i \in \{0, \ldots, b-1\}$; $b \geq 2$
- Praktisch relevante Basen:

  b = 10 (Dezimal) z_{10}

  b = 2 (Binär-/Dual) z_2

  b = 16 (Hexadezimal) z_{16}

  b = 8 (Oktal) z_8

  b = 1 (Unäre) z_1 (Strichliste)

5.1 Binärzahlen

- Rechen mit Binärzahlen
  - Addition
    $0+0=1; 0+1=1; 1+1=10$
  - Subtraktion
    $0-0=0; 1-0=1; 1-1=0; 0-1=10$(Übertrag)
- Umwandlung
  - $Hexadezimal \Leftarrow$ Binär
    Eine Hexadezimalstelle kann durch 4 Binärstellen ausgedrückt werden
  - $Oktal \Leftarrow$ Binär
    Eine Oktalstelle kann durch 3 Binärstellen
  - BCD(Binäry coded Dezimal)
    jeder Dezimalziffer wird mit 4 Binärziffer dargestellt
- Negative Zahlen
  - Explizite Speicherung des Vorzeichens(+ durch 0, - durch 1)
  - nur bei vorzeichenbehafteten Größen:
    Letzter Bit(Binär) steht für das Vorzeichen (+ durch 0, - durch 1)
  - Komplementbildung
    Einerkomplementdarstellung (Invertieren Aller Bits) (-0(negative Null) addieren)
    Zweierkomlementdarfstellung (Invertieren aller Bits und 1 addieren)
- Ganzzahlarithmetik
  - Schiebe Operatoren(shifts):
    - $<<$ Linksshift, $>>$ Rechtsshift
    - Linksshift ziehen immer einen "0" nach, Rechtsshift auf unsinged Größen ebenfalls
    - Rechtsshift auf auf singed Größen ziehen das Vorzeichenbit nach
    - Multiplikation bzw. Division durch 2er Potenzen (bei Laufzeitkritischen Anwendungen)
- Fixed Point

  PRO: Rechnung schnell

  CONTRA: Rundungsfehler
- Floating Point ($Gz := (< Vorzeichen >) < Mantisse > * Basis^{<Vorzeichen>Exponent}$)

  Eine Gleitkommazahl heißt normiert,wenn die Mantisse < 1 und die erste Nachkommastelle ungleich Null ist

6 Aussagenlogik

- Eine Aussage ist ein Objekt, dem sich eindeutig ein Wahrheitswert (true,T)/(false,F) zuordnen lässt
- Operatoren
  - Konjunktion $\wedge$
    Gesamtaussage ist wahr, wenn A und B wahr sind
  - Disjunktion $\vee$
    Gesamtaussage ist wahr, wenn mindestens eine Aussage wahr ist
  - Negation $\neg$
    Ergebnis falsch, wenn A wahr und umgekehrt
  - Implikation $\Longrightarrow$
    Ergebnis whar, wenn entweder A falsch oder A und B wahr
  - Äquivalenz $\Longleftrightarrow$
    Ergebnis wahr, wenn entweder A und B Falsch oder A und B wahr
  - Exklusives oder *xor*
    Ergebnis war, wenn A und B ungleich
  - negiertes oder *nor*
    Ergebnis wahr, wenn A und B unwahr
  - negiertes und *nand*
    Ergebnis wahr, wenn A oder B unwahr
- Eine Formel ist erfüllbar, wenn mindestens eine Belegung gibt, deren Auswertung den Wert wahr liefert
- Eine Formel heißt Tautologie, wenn jeder Belegung den Wert wahr liefert
- Eine Formel heißt unerfüllbar (Kontradiktion) wenn es keine Belegung gibt, die die Formel erfüllt
- Zwei Formeln heißen äquivalent($\tilde{=}$) wenn jede Belegung der Formlen zum gleichen Wert führt
- Äquivalenzen:
  - Kommutativität
    $F_1 \wedge F_2 \tilde{=} F_2 \wedge F_1$ oder $F_2 \wedge F_1 \tilde{=} F_1 \wedge F_2$
  - Assoziativität
    $(F_1 \wedge F_2) \wedge F_3 \tilde{=} F_1 \wedge (F_2 \wedge F_3)$ oder $(F_1 \vee F_2) \vee F_3 \vee = F_1 \vee (F_2 \vee F_3)$
  - Distributivität
    $(F_1 \wedge F_2) \vee F_3 \tilde{=} (F_1 \vee F_3) \wedge (F_2 \vee F_3)$ oder $(F_1 \vee F_2) \wedge F_3 \tilde{=} (F_1 \wedge F_3) \vee (F_2 \wedge F_3)$

- Idempotenz

  $F_1 \wedge F_1 \tilde{=} F_1$ oder $F_1 \vee F_1 \tilde{=} F_1$

- De Morgan'sche Regel

  $\neg(F_1 \wedge F_2) \tilde{=} \neg F_1 \vee \neg F_2$ oder $\neg\neg(F_1 \vee F_2) \tilde{=} \neg F_1 \wedge \neg F_2$

- Implitaktionsregel

  $F_1 \iff F_2 \tilde{=} \not{F}_1 \vee F_2$

- Konjunktive Normalform

  Eine Formel F ist in konjunktiver Normalform (KNF) , wenn sie eine Konjunktion von Disjunktionen von Literalen ist

  Ein Literal ist dabei eine atomare Formel oder die Negation eine atomaren Formel (positives bzw. negatives Literal)

  $F = \wedge_{i=1}^{k}(\vee_{j=1}^{mk} L_i j)$

- Disjunktive Normalform

  Eine Formel F ist in disjunktiver Normalform (DNF), wenn sie eine Disjunktion von Konjunktionen von Literalen ist

  $F = \vee_{i=1}^{k}(\wedge_{j=1}^{mk} L_i j)$

- Für jeder Formel F gilt:

  zu F gibt es eine äquivalente Formel in konjunktiver Normalform und eine äquivalente in Disjunktiver Normalform

- deMorgan'sche Verknüpfungsbase n$\{\vee, \neg\};\{\wedge, \neg\}$

- Boole'sche Verknopfungsbasis $\{\wedge, \vee, \neg\}$

- Frege-Basis $\{\neg, \iff\}$

- nand-Basis $\{nand\}$

- nor-basis $\{nor\}$

7 Prädikatenlogik

- Sei M eine menge (Gegenstandsbereich, INdividuenbereich). Dann heißt einen Abblunding $M^n \leftarrow \{wahr, falsch\}$ ein n-stelliges Prädikat über der Menge M

- Logische Operatoren

  - Konjunktion $\wedge$

    Gesamtaussage ist wahr, wenn A und B wahr sind

  - Disjunktion $\vee$

    Gesamtaussage ist wahr, wenn mindestens eine Aussage wahr ist

- Negation $\neg$

  Ergebnis falsch, wenn A wahr und umgekehrt
- Implikation $\Longrightarrow$

  Ergebnis wahr, wenn entweder A falsch oder A und B wahr
- Äquivalenz $\Longleftrightarrow$

  Ergebnis wahr, wenn entweder A und B Falsch oder A und B wahr

- Quantoren:

  $\forall$ heißt Allquantor (für alle)

  $\exists$ heißt Existenzquantor (existiert)

- Variablen, die an Quantoren gebunden sind, heißen gebundene Variablen, alle anderen heißen freie Variablen

- Eine Formel ohne konstante Prädikate, die für jede Wahl der variablen Prädikate und dann fpr jede mögliche Wahl der Individuenvariablen immer den Wert T ergibt, heißt Gesetz der Prädikatenlogik

8 Zustandsautomaten

- Mealy-Automat
  - Ein Mealy-Automat ist durch (Z, z_0, E, A, T) gegeben

    Z ist eine endliche nichtleere Menge von Zuständen

    $z_0, z_0 \in Z$ ist der Anfangszustand

    E ist die endliche Menge der möglichen Eingaben

    A ist die endliche Menge der möglichen Ausgaben

    T ist die Menge der Transitionen(Zustandsübergänge)
  - Jeder Transition $t \in T$ ordnen einem Ausgangszustand $z_a \in Z$ und einer Eingabe $e \in E$ einen Folgezustand $z_f \in Z$ und einen Ausgabe $a \in A$ zu:$(z_a, e) \rightarrow (z_f, a)$

- Graphische Darstellung:
  - Jedem Zustand wird ein Knoten (gezeichnet als Kreis) zugeordnet.
  - jeder Knoten wird mit dem Namen des zugehörigen Zustands beschriftet
  - jede Transition wird einen Gerichtete kante (gezeichnet als Pfeil) von Zustand A nach zustand F zugeordnet
  - Jeder Transition wird mit ihrer Eingabe e und Ausgabe a beschriftet

9 Modularisierung

- Zerlegung in unabhängig voneinander zu erstellende SW-Teile (Module)
- Module sind getrennt voneinander übersetzbar
- Grundlage der Zerlegung ist eine vor der Codierung definierte SW Architektur
- Die Gesamtfunktion wird durch das "Zusammenspiel" der einzelnen Module realisiert
- Elemente eines Moduls:

  Konstanten, Variablen, Funktionen und Prozeduren werden in anderen Modulen verwendet
- Prinzipien:

  per #define definierte Konstanten/Makros werden in anderen Modulen verwendet
- Realisierung in C:
  - per #define definierte Größen werden in einem Headerfile (.h) definiert, dieses diese werden, wo benötigt, eingebunden(#include)
  - Eine Konstante(const) oder Variable wird in einer Datei(.c) global definiert. Über eine externe Deklaration in einem Headerfile kann diese in anderen Modulen bekanntgemacht werden

    Selbiges gilt auch für Funktionen/Prozeduren
- Globale Namen müssen eindeutig sein
- Ein Modul muss das Hauptprogramm enthalten
- Verwendung von Funktionen und Prozeduren modulübergreifend ist OK
- globale Variablen sind zu vermeiden(Informationen möglichst über Schnittstelle austauschen)

10 Prozessmodelle

- Phasenmodelle:
  - Wasserfall-Modell
  - Prototyping, evolutionäre Entwicklung
  - Formale Entwicklung
  - Wiederverwendungsorienttierete Entwicklung (opportunistisch/strategisch)
  - Prozesswiederholende Modelle (inkrement/agile)
  - Modellbasierte Entwicklung
  - V-Modell
- Wasserfall-Modell:
  - Strikte Reihenfolge von Projektphasen
  - Korrekturzyklen werde nur auf die vorangehende Phase beschränkt
  - Vollständiges Ausführen einer Phase, Dokumentation als Abschluss
  - Nachfolgephase beginnt erst nach vollständigem Abschluss einer Phase
  - Auftraggeber und Anwender nur in der Phasenplanung und bei der Abnahme beteiligt
  - Vorteile:

    Einfaches und klares Vorgehen

    Relativ einfach umzusetzen (Planung/Steuerung)

    Erzwingt eine gewisse Disziplin bei den Entwicklern
  - Nachteile:

    Entspricht nicht der Realität

    Phasen können sich gegenseitig bedingen

    schwache Einbindung des Auftraggebers, bzw. der Anwenders

    Spätes Erkenne von Risiken, die sich aus der Implementierung ergeben

    Spätes Erkennen von Fehler aus den frühen Phasen, die erst in späteren Phasen erkennbar werden

    Dokumentation bekommt einen manchmal zu hohen Stellenwert

    Zeitverzug aus frühen Phasen schlägt voll durch auf spätere Phasen

    Im Extremfall ist das Produkt nicht brauchbar, die Entwicklung muss neu von vorne begonnen werden

    Säte Änderung und Fehlerbehebung im Code führen zu veralteter Dokumentation

- V-Modell:
  - Verbessertes Wasserfallmodell
    ergänzt um QS in allen Phasen, standardisiert
  - Ziel: Entwurfsfehler frühzeitig zu erkennen
  - Aber Nachteile des Wasserfallmodels bleiben erhalten:
    starre Reihenfolge der Phasen, Ergebnis in einem Durchlauf entwickelt
- V-Modell-XT:
  - Verbesserung der Unterstützung von Anpassbarkeit, Anwendbarkeit, Skalierbarkeit und Änder- und Erweiterbarkeit des V-Modells
  - Berücksichtigung des neusten stand der Technologie und Anpassung an aktuelle Vorschriften und Normen
  - Erweiterung des Anwendungsbereiches auf die Betrachtung des gesamten Systemlebenszyklus, im Rahmen von Entwicklungsprojekten
  - einführen eines organisationsspezifischen Vierbesserungsprozesses für Vorgehensmodelle