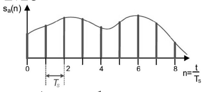
Technische Grundlagen Medieninformatik

2. "Analog" und "Digital"

2.8
Analog: kontinuierliche (stufenlose) Werte
Digital: diskrete (bestimmte) Werte

2.11
Musik: Zeitlicher Verlauf des Schalldrucks
Bilder: Örtlicher Verlauf der Helligkeit

2.13
Abtastung: Zeitliche Quantisierung

2.15


$n = \frac{t}{T_s}, f_s = \frac{1}{T_s}$Hz
$S_a(n) = s(n \cdot T_s) = s(\frac{n}{f_s})$
(analog)

2.16 s_q = Quantisiertes und abgetastetes Signal

2.17 Codierung: Anzahl der Bits $n = ldw, daw = 2^n$
"polyadisches Stellenwertsystem"

2.18 Polyadisch: LSB → MSB
Zweierkompliment (Gray): 1 Stellenwechsel

2.21

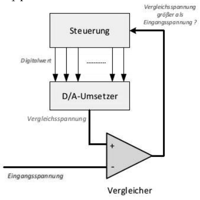
Standard	"Null"	"Eins"
CMOS 5V	$\leq 1,5$	$\geq 3,5$
LVTTL 3,3V	$\leq 0,8$	$\geq 2,0$
CMOS 2.5V	$\leq 0,7$	$\geq 1,7$

2.22 Spannungsintervalle für Störsicherheit
Bits sind wie Schalter/Transistoren

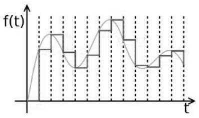
2.25
Mikrofon: Bewegung eines Drahtes in einem Magnetfeld erzeugt elektrische Spannung
Lautsprecher: Stromfluss durch Draht im Magnetfeld erzeugt Bewegung

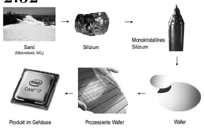
2.26 D/A-Umsetzer = Widerstandsnetzwerk

2.28 Sukzessive Approximation



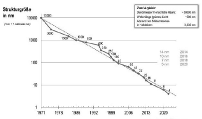
2.29
Abstand-Halte-Glied ("Sample and Hold")
Eine elektronische Schaltung, die analoge Werte "halten" kann



2.32


2.35
1965: "Moores Law"
"The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. Certainly over the short term this rate can be expected to continue, if not to increase."

2.36
Halbleiter-Strukturgrößen



2.37-2.38 Moderne Prozessoren haben eine hohe Verlustleistung. Diese ist etwa genauso groß, wie die Leistung einer Herdplatte.

2.42-2.44
Ausbeutungsmodelle
Ausbeute (Yield Y), Defektdichte (D)
The Poisson Model: $Y = e^{-AD}$
The Murphy Model:
$Y = \left[\frac{1-e^{-AD}}{AD}\right]^2$
The Seeds Model:
$Y_{Seeds} = \frac{1}{1+AD}$
The Moore Model: $Y = e^{-\sqrt{AD}}$

3. Signalverarbeitung

3.6
Einfache Schwingung (Feder und Gewicht)
$s(t) \quad sin(\sqrt{\frac{D}{m}} \cdot t)$

3.7
$t = T = 2\pi$ (1. Periode),
$f = \frac{1}{T} = 1\pi$ Hz (Frequenz),
$s(t) = sin(2\pi ft) = sin(t)$
(Zeitlicher Verlauf)

3.8
Einfache (harmonische) Schwingungen: sinus/cosinus-förmig
Komplexe Schwingungen: Mehrere Sinus-/Cosinus-Terme, Frequenz (f) und Amplitude (A)
$x(t) = \sum_{i=1}^{n} = A_{Si} \cdot sin(2\pi f_1 \cdot t) + A_{Ci} \cdot cos(2\pi f_1 \cdot t)$

3.9
Jean Baptiste Josephe Fourier (1768-1830)
Fourier-Transform/Reihe für kontinuierliche bzw. periodisch kontinuierliche Funktionen

3.10-3.13 (Vereinfachte) Diskrete Fourier-Transform (DFT)
Relevante Perioden für N Perioden: $sin < \frac{N}{2}, cos \leq \frac{N}{2}$
x beliebige (kontinuierliche) Werte,
t = Zeit (kontinuierlich),
n = ganzzahlige (diskrete) Werte
(Folge von Abtastwerten),
$f(x) = sin(i \cdot \frac{2\pi}{N}x)$

3.14-3.19
x = zeitkontinuierlich, n = zeitdiskret
$f(x) = sin(i \cdot \frac{2\pi}{N}x) \rightarrow f(n) = sin(i \cdot \frac{2\pi}{N}n)$

3.21 Frequenzanalyse
$A_c(k) = \sum_{n=0}^{N-1} f(n) \cdot cos(\frac{2\pi}{N}kn)$
$A_s(k) = \sum_{n=0}^{N-1} f(n) \cdot sin(\frac{2\pi}{N}kn)$
+ Korrekturfaktoren

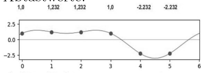
3.22 Korrekturfaktoren
Für $f(n) = sin/cos(\frac{2\pi}{N}kn)$ soll gelten $A_{c/s} = 1 \rightarrow$
$C_c(0) = C_c(\frac{N}{2}) = \frac{1}{N}$
$C_c(k) = C_s(k) = \frac{2}{N}$

3.24 Synthese
$f(n) \quad \sum_{k=0}^{N/2} \left[A_c(k) \cdot cos(\frac{2\pi}{N}kn) + A_s(k) \cdot sin(\frac{2\pi}{N}kn)\right]$
Periodendauer: $\frac{N}{k}$
Frequenz: $\frac{k \cdot f_s}{N}$ Hz

3.25
$t = n \cdot T_s = \frac{n}{f_s}$
Einsetzen in
$c(n,k) = cos(\frac{2\pi k}{N}n)$ (diskret)
$\rightarrow c(t,k) \quad cos(\frac{2\pi f_s k}{N}t)$
(kontinuierlich)
k = Frequenz (digital), f = Frequenz (analog)
$f = \frac{f_s k}{N}$

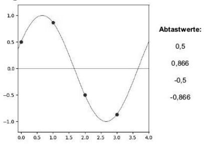
3.28-3.31 Beispiel für N=6

n=	0	1	2	3	4	5
$cos(0 \cdot \frac{2\pi n}{6})$	1	1	1	1	1	1
$cos(1 \cdot \frac{2\pi n}{6})$	1	0,5	-0,5	-1	-0,5	0,5
$cos(2 \cdot \frac{2\pi n}{6})$	1	-0,5	-0,5	1	-0,5	-0,5
$sin(1 \cdot \frac{2\pi n}{6})$	0	$\sqrt{3}/2$	$\sqrt{3}/2$	0	$-\sqrt{3}/2$	$-\sqrt{3}/2$
$sin(2 \cdot \frac{2\pi n}{6})$	0	$\sqrt{3}/2$	$-\sqrt{3}/2$	0	$\sqrt{3}/2$	$-\sqrt{3}/2$

Abtastwerte:


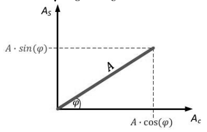
$A_c(0) = \frac{1}{6} \cdot (1 \cdot 1 + 1 \cdot 1,232 + 1 \cdot 1,232 + 1 \cdot 1 - 1 \cdot 2,232 + 1 \cdot 2,232) = 0$
$A_c(1) = \frac{1}{3} \cdot (1 \cdot 1 + 0,5 \cdot 1,232 - 0,5 \cdot 1,232 - 1 \cdot 1 + 0,5 \cdot 2,232 - 0,5 \cdot 2,232) = 0$
$A_c(2) = \frac{1}{3} \cdot (1 \cdot 1 - 0,5 \cdot 1,232 - 0,5 \cdot 1,232 - 1 \cdot 1 + 0,5 \cdot 2,232 + 0,5 \cdot 2,232) = 1$
$A_s(1) = \frac{1}{3} \cdot (0 \cdot 1 + \frac{\sqrt{3}}{2} \cdot 1,232 + \frac{\sqrt{3}}{2} \cdot 1,232 + 0 \cdot 1 + \frac{\sqrt{3}}{2} \cdot 1,232 + \frac{\sqrt{3}}{2} \cdot 1,232) = 2$
$A_s(2) = \frac{1}{3} \cdot (0 \cdot 1 + \frac{\sqrt{3}}{2} \cdot 1,232 - \frac{\sqrt{3}}{2} \cdot 1,232 + 0 \cdot 1 - \frac{\sqrt{3}}{2} \cdot 1,232 + \frac{\sqrt{3}}{2} \cdot 1,232) = 0$

3.32-3.33 60° ($\frac{\pi}{3}$)verschobener Cosinus, N=4



$cos(x - \varphi) = cosx \cdot cos\varphi + sinx \cdot \varphi$
$A_c(0) = 0, A_c(1) = 0,5, A_s(1) = 8,66 = \frac{\pi}{3}$
Oder $A(1) = 1, \varphi(1) = \frac{\pi}{3} \sim 60°$ in der Betrag/Phase-Darstellung

3.34-37
Betrag/Phasen-Darstellung
Von zwei Schwingungen zu einer Schwingung:
$A_s \cdot sin(x) + A_c \cdot cos(x) = A \cdot cos(x - \varphi)$
Additionstheorem:
$sin(\varphi) \cdot sin(x) + cos(\varphi) \cdot cos(x) = cos(x - \varphi)$
$A_s \cdot sin(x) + A_c \cdot cos(x) = A \cdot cos(\varphi) \cdot cos(x) + A \cdot sin(\varphi) \cdot sin(x) \rightarrow$
$A_s \cdot sin(x) = A \cdot sin(\varphi) \cdot sin(x) \rightarrow A_s = A \cdot sin(\varphi)$,
$A_c \cdot cos(x) = A \cdot cos(\varphi) \cdot cos(x) \rightarrow A_c = A \cdot cos(\varphi)$
Phasenverschiebung:
$sin(\varphi) = -\frac{A_s}{A}, cos(\varphi) = -\frac{A_c}{A}$
$\frac{sin(\varphi)}{cos(\varphi)} \quad tan(\varphi) \quad -\frac{A_s}{A_c}$
$\rightarrow \varphi = -arctan(\frac{A_s}{A_c})$ Betrag:
Für $x = -\varphi$:
$A \quad A_s \cdot sin(-\varphi) + A_c \cdot cos(-\varphi)$
$\rightarrow A = \sqrt{A_c^2 + A_s^2}$



$A \cdot cos(x - \varphi) = A \cdot sin(\varphi) + A \cdot cos(\varphi) \cdot cos(x)$

3.42-3.43
Aliasing: Frequenzen über $f_s/2$ werden gespiegelt dargestellt (ref 2.48)

3.46 Zufällig / Fehlerhaftes Ergebnis der Abtastung bei Frequenzen gleich $f_s/2$

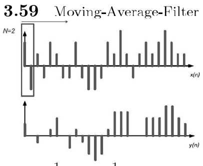
3.50 Nyquist-Shannon-Abtasttheorem / Nyquist-Kriterium / Abtasttheorem:
$f_{max} - f_{min} < \frac{f_s}{2}$ oder
$f_{max} < \frac{f_s}{2}$ falls $f_{min} = 0$

3.52 Tiefpass-Filter
Ein Tiefpass-Filter lässt nur niedrige Frequenzen durch

3.54
Es treten Fehler bei der Frequenzanalyse auf, falls die Signalperiode nicht der Transformationslänge entspricht

3.56 Fensterfunktion
Vermindert die in 2.54 auftretenden Fehler durch Multipizieren der Abtastwerte mit dieser Funktion. Dies wird benötigt, da die beiden Werte fast nie gleich sind

3.59 Moving-Average-Filter



$y(n) = \frac{1}{2}x(n) + \frac{1}{2}x(n-1)$

3.60
Differenzfilter
$y(n) \quad \frac{1}{2}x(n) - \frac{1}{2}x(n-1)$
Mittelwert
$y(n) = \sum_{i=0}^{N-1} \frac{1}{N}x(n-1)$

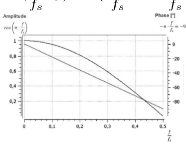
3.61 FIR-Filter
FIR = Finite Impulse Response (Endliche Impulsantwort)
Modularer Filter ie die (Filter-)Koeffizenten (h) sind frei wählbar
$y(n) = \sum_{i \ 0}^{N-1} h(i) \cdot x(n-i)$

3.63
FIR-Tiefpass $(\frac{1}{2}, \frac{1}{2})$,
FIR-Hochpass $(\frac{1}{2}, -\frac{1}{2})$

3.64 Graphendarstellung des Frequenzgangs
$x(i) = 1 : y(n) = \sum_{i=0}^{N-1} h(i)$
$x(i) = (i\%2) * 2 - 1 : y(n) = \sum_{i=0}^{N-1} ()h(2i) - h(2i+1))$

3.65 Beliebige Frequenzen
$x(t) = cos(2\pi f \cdot)$
(zeitkontinuierlich)
$\rightarrow x(n) = cos(2\pi f \cdot \frac{n}{f_s})$
(zeitdiskret)
Antwort des Filters: $\frac{y(n)}{x(n)}$

3.66 Beispiel:
$y(n) \quad \frac{1}{2}x(n) + \frac{1}{2}x(n-1)$
$x(n) = cos(\frac{2\pi f}{f_s} \cdot n)$
$\xrightarrow{einsetzen}$
$x(n) = \frac{1}{2}cos(\frac{2\pi f}{f_s} \cdot n) + \frac{1}{2}cos(\frac{2\pi f}{f_s} \cdot (n-1))$
$\xrightarrow{umformen}$
$x(n) = cos(\frac{\pi}{f_s} \cdot f) \cdot cos(\frac{2\pi f}{f_s} \cdot n - \frac{\pi}{f_s} \cdot f)$



3.68-3.70 Frequenzgang über die Transformation
Der Frequenzgang entspricht der Transformierten von h
Amplitudenverlauf für
$y(n) = \frac{1}{2}x(n) + \frac{1}{2}x(n-1)$,
$h(m) = \frac{1}{2}$ für $m = 0, 1, 0$ sonst
$k = \frac{N}{f_s}f$ (N fällt auf Grund der Periodizität von Sinus und Cosinus weg)
$A_c'(k) = \sum_{m=0}^{a-1} h(m) \cdot cos(2\pi km)$
$\rightarrow A_c'(k) = \frac{1}{2} + \frac{1}{2}cos(2\pi k)$
$\rightarrow A_c'(f) = \frac{1}{2}(1 + cos(\frac{2\pi f}{f_s}))$
$A_s'(k) = \sum_{m=0}^{a-1} h(m) \cdot sin(2\pi km)$
$\rightarrow A_s'(k) = \frac{1}{2} + \frac{1}{2}sin(2\pi k)$
$\rightarrow A_s'(f) = \frac{1}{2}(1 + sin(\frac{2\pi f}{f_s}))$
Amplitudenverlauf:
$A'(f) = \sqrt{A_c'^2(f) + A_s'^2(f)}$
$\xrightarrow{einsetzen}$
$A'(f) = \frac{1}{2}\sqrt{(1 + cos(\frac{2\pi}{f_s}f))^2 + sin(\frac{2\pi}{f_s}f)^2}$
$\xrightarrow{umformen}$
$A'(f) = |cos(\frac{\pi a}{f_s}f)|$
$\varphi'(f) = -arctan\left(\frac{sin(\frac{2\pi}{f_s}f)}{1 + cos(\frac{2\pi}{f_s}f)}\right)$
$\xrightarrow{umformen}$
$\varphi'(f) = -\frac{\pi}{f_s}f$

3.71
Verzögerung eines Signals um die Zeit T:
$cos(2\pi f(t - T))$
Äquivalente Phasenverschiebung:
$\varphi = -2\pi fT$ bzw $T = -\frac{\varphi}{2\pi f}$

3.74 Annäherung des Frequenzgangs
Gesucht: 2M+1 Koeffizienten eines Filters mit dem Frequenzgang A
Ausgangspunkt M+1 Abtastwerde des gewünschten Frequenzgangs A für die Frequenzen
$f = (0, \frac{2}{2M+1}, \frac{4}{2M+1}, ..., \frac{2M}{2M+1})\frac{f_s}{2}$
Filter-Koeffizienten:
$h(n) = \frac{1}{2M+1}(A(0) + 2\sum_{i=1} A(f) \cdot cos(\frac{2\pi i}{2M+1}n))$

3.75 Beispiel: Tiefpass
$M = 3, f_g = 0,25f_s$

$\frac{2f}{f_s}$	0	$\frac{2}{7}$	$\frac{4}{7}$	$\frac{6}{7}$
$A(f)$	1	1	0	0

$h(n) = \frac{1}{7}(1 + 2 \cdot cos(\frac{2\pi}{7}n))$

n	0	±1	±2	±3
h(n)	0,43	0,32	0,08	-0,11

Koeffizienten n¡0 verschieben:
$h'(n) \quad h(n - M)$

3.79
Audio: Schalldruck, Abtastung über die Zeit (t), Eindimensional, 16 bit pro Sample, 1/2 Kanäle
Bilder: Helligkeit, Abtastung über den Ort (x,y), Zweidimensional, 8 bit pro Sample, 1/3 Kanäle

3.84 Filterung von Bildern
$q(x,y) = \sum_{j=0}^{M-1} \sum_{i=0}^{N-1} h(i,j) \cdot p(x-i, y-j)$

3.85 Anwendungsbeispiele
Rauschreduktion, Bandbreitenbegrenzung, Kantenerkennung, Verbesserung der (subjektiven) Bildschärfe, Grundfunktion von Neuralen Netzen (künstliche Intelligenz)

3.87
Einige Filter können seperat, also in zwei Schritten angewendet werden

$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$

3.91 Beispiele
Glättungsfilter (Weichzeichner):
$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \cdot \frac{1}{9}$
Schärfungsfilter:
$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$
Kantenfilter:
$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$
Relieffilter:
$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$

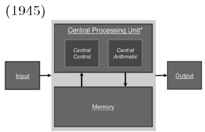
3.99 Kleine Formelsammlung
Symmetrie
$sin(x) = sin(x + m \cdot 2\pi), sin(x) = -sin(-x)$
$cos(x) = cos(x + m \cdot 2\pi), cos(x) = -cos(-x)$
Additionstheoreme
$sin(x \pm y) = sin(x)cos(y) \pm cos(X)sin(y)$
$cos(x \pm y) = cos(x)cos(y) \pm sin(x)sin(y)$
Umrechnungen / weitere Beziehungen
$sin(arctan(x)) = \frac{x}{\sqrt{x^2+1}}$
$cos(arctan(x)) = \frac{1}{\sqrt{x^2+1}}$
$1 + cos(2x) = 2 \cdot cos^2(x)$
$sin(2x) = 2 \cdot sin(x) \cdot cos(x)$
$cos^2(x) + sin^2(x) = 1$

3.100
$sin(k \cdot \frac{2\pi n}{N})/cos(k \cdot \frac{2\pi n}{N})$
Sinus/Cosinus mit der Periode von N Abtastwerten besitzen eine Symmetrie und deshalb brauchen wir nur die Werte von $k = 0..\frac{N}{2} - 1$

4. Grundprinzipien eines Rechners

4.2-4.4 Historische Rechenhilfen Abakus (2500vC), Rechenschieber (1600), Erste mechanische Rechenmaschine (1623), Z3 (1941. Deutschland), Collossus (1943, Großbritannien), ASCC / Mark1 (1944)

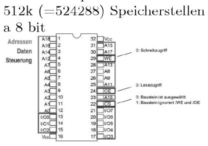
4.6-4.7
Von-Neumann-Architektur (1945)



CPU: Central Processing Unit, Steuerung und Arithmetik
Arithmetik: Berechnungen wie Addition, Subtraktion, Multiplikation, Division, Wurzel, Sinus, Cosinus, ...
Steuerung: Befehele auswerten, Operanden holen, Ergebnisse Speichern
Speicher: Speichert Daten und Programme

4.10 Anschlüsse eines Speichers
Adresse: Auf welche Speicherstelle soll zugegriffen werden
Daten: Inhalt der Speicherstelle
Steuerung: Sollen die Daten im Speicher abgelegt werden oder sollen gespeicherte Werte ausgegeben werden

4.11 Beispiel: 628512
512k (=524288) Speicherstellen a 8 bit



4.12 Wortbreiten
1 Bit, 8 Bit = 1 Byte
16 Bit 1 Halfword
32 Bit = 1 Word
64 Bit = 1 Doubleword
128 Bit = 1 Quadwo rd

4.13 Byteadressierung
Der Speicher besteht aus einer Speicherwortbreite und Speicherstellen.
Die Bytes können unabhängig von der Speicherwortbreite adressiert werden.

4.17 Little und Big Endian
Die Byteadressen eines Wortes sind umgedreht

Byte-Adresse	Little	Big
0x0	D	A
0x1	C	B
0x2	B	C
0x3	A	D

4.19 Abstraktionsebenen eines Programms
Hochsprache $\xrightarrow{Compiler}$ Assembler $\xrightarrow{Assembler}$ Maschinencode

4.20
Es gibt maschinen-unabhängige Sprachen, die auf dem ausführenden Rechner "interpretiert" oder vor der Ausführung in Maschinencode übersetzt werden

4.21
"+" wird zu "add", "if" gibt es nicht

4.22 Befehlsabarbeitung
1. Nächsten Befehl aus dem Speicher holen
2. Operanden aus dem Speicher holen
3. Operation ausführen
4. Ergebnis in Speicher schreiben

4.23 Program Counter
Spezielle Speicherstelle die die aktuelle Befehlsadresse enthält
Nächste Adresse = Aktuelle Adresse + Befehlsadresse

4.24 Speicheradressierung
Zum Reduzieren der Befehlslänge gibt es extra Operanden- und Ergebnisadressen

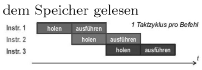
4.25 Akkumulator
Ein Speicher für Zwischenergebnisse

4.28
CISC: Complex Instruction Set Computer
RISC: Reduced Instruction Set Computer

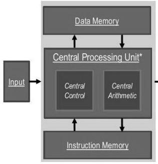
4.29 CISC
Kostenersparnisse durch kleinen Programmcode und wenige Speicherzugriffe
Steuerung innerhalb der CPU durch ein "Mikroprogrammsteuerwerk"
"Jeder Befehl startet ein kleines Programm innerhalb der CPU"

4.30 RISC
Häufige Befehle schneller machen, da Speicher günstiger ist
Komplexe Befehle werden selten benötigt und machen das Design komplexer, was die CPU langsamer macht
Nur häufig benutzte Befehle werden in Hardware implementiert
Speicherzugriffe sind nur durch Load- und Storebefehle erlaubt (einfache Adressierung)
Hohe Anzahl an Registern (meist mindestens 32 Register)
Kein Mikroprogrammsteuerwerk
Pipelining und Harvard-Architektur zur Erhöhung des Befehlsdurchsatzes

4.32 Pipelining
Während eine Instruktion verarbeitet wird, wird die nächste Instruktion bereits aus dem Speicher gelesen



4.33 Harvard-Architektur
Zugriffe auf Instruktionen und Daten trennen



5. Zahlendarstellung in Rechnern

5.3-5.4 Stellenwertsysteme
Jeder Position einer Ziffer wird eine Wertigkeit (Stellengewicht) zugeordnet
Die Summe der Produkte der einzelnen aus Zifferngewicht und Stellengewicht ergibt den Wert den dargestellten Zahl
Ziffernfolge $A = (a_{n-1}a_{n-2}...a_1a_0)$, Zifferngewicht a_i, Stellengewicht g_i
$A = \sum_{i=0}^{n-1} a_i \cdot g_i$

5.5 Polyadische Stellenwertsysteme
Um ganze Zahlen lückenlos darstellen zu können, muss das Stellengewicht g_i in Abhängigkeit der Anzahl der zur Verfügung stehenden Ziffern n_z gewählt werden
$g_i = n_z^i$
n_z wird üblicherweise als Radix R oder Basis B bezeichnet
$A = \sum_{i=0}^{n-1} a_i \cdot B^i$

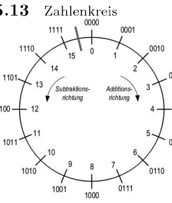
5.6-5.7
Wichtige Basen 2, 10 und 16

5.8 Rechnen in Hexadezimal
Aus dem Zehner-Übergang wird ein Sechszehner-Übergang

5.9
Kennzeichnung der Hexdezimaldarstellung mit $(...)_16$, $...h$ oder $0x...$

5.12 Überlaufe
Ist das korrekte Ergebnis größer als die verfügbare Wortbreite tritt ein Überlauf auf und das ausgegebene Ergebnis falsch. Das Auftreten eines Überlaufes kann Signalisiert werden

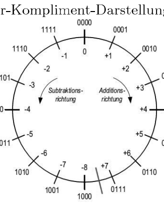
5.13 Zahlenkreis



5.14
Vorzeichen-Betrag-Darstellung
Negative Zahlen werden mit einer extra Stelle für das Vorzeichen dargestellt (1 Negativ)

5.16
Radix-Kompliment-Darstellung
Ziel: Einfache Rechenregeln sollten ohne Fallunterscheidung gelten und es soll nur eine Null geben →

5.17
2er-Kompliment-Darstellung



5.18 Zusammenfassung
Vorzeichenlose Darstellung:
$A = \sum_{i=0}^{n-1} a_i \cdot 2^i$
Vorzeichen-Betrag-Darstellung:
$A = (-1)^{a_{n-1}} \cdot \sum_{i=0}^{n-2} a_i \cdot 2^i$
2er-Komplement-Darstellung:
$A = -a_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} a_i \cdot 2^i$

5.19-5.21
Vorzeichenerweiterung
Vorzeichenlose Zahlen: Nach MSB eine 0 einfügen
2er-Kompliment-Zahlen: Nach MSB den Wert des vorherigen MSB einfügen

5.22-5.26
Überlauferkennung
Addition:

Höchstwertiges Bit			Überlauf ?	
Operand1	Operand2	Ergebnis	vorzeichen-los	vorzeichen-behaftet
0	0	0	✓	✓
0	0	1	✓	×
0	1	0	×	✓
0	1	1	✓	✓
1	0	0	×	✓
1	0	1	✓	✓
1	1	0	×	×
1	1	1	×	✓

✓ Ergebnis ok, kein Überlauf × Ergebnis falsch, Überlauf

Subtraktion:

Höchstwertiges Bit			Überlauf ?	
Operand1	Operand2	Ergebnis	vorzeichen-los	vorzeichen-behaftet
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

5.28-5.29
Festkommadarstellung
Positive Zahlen: Vorkommastellen m, Nachkommastellen k
$A = \sum_{i=-k}^{m-1} a_i \cdot 2^i$
Positive und Negative Zahlen: Vorkommastellen l, Nachkommastellen k
$A = -a_{l-1} \cdot 2^{l-1} + \sum_{i=-k}^{l-2} a_i \cdot 2^i$

5.30 Gleitkommadarstellung
Mantisse m, Exponent e (vorzeichenbehaftet):
$A = (-1)^s \cdot m \cdot 2^e$
Gebräuchliche Format:
Einfache Genauigkeit M: 23 bi, E: 8 bit
Doppelte Genauigkeit M: 52 bi, E: 11 bit

6. Beispielprozessor "MIPS"

6.2 Aufgaben eines Prozessors
Daten im System transportieren und (arithmetische/logische) Berechnungen durchführen

6.3 Modelvorstellung
Rechnersystem bestehend aus Prozessor und Speicher (jeder Eintrag 32 bit breit)
Der Rechner versteht nur Assembly
"Wortschatz des Rechners" = Instruktionssatz

6.6 Speicherzugriffe und Rechnerleistung
Regel 1: "Speicher sind eigentlich immer zu langsam"
Regel 2: "Je größer ein Speicher ist, desto langsamer ist er" → Zwischenspeicher für temporäre Variablen anlegen

6.7
Moderne Prozessoren haben mehrere Register (Speicherstellen/Akkumulatoren)

6.8
Registernamen beginnen mit \$.. (zb \$0, \$1, ...)
Unser Prozessor hat 32 Register

6.9
\$0 ist eine Konstante mit dem Wert 0

6.11
Unser CPU hat 2 gleichzeitige Lesezugriffe und einen Schreibzugriff

6.13 Performancesteigerung
Load/Store-Architektur: Daten in Registern halten, keine arithmetischen Instruktionen mit Speicheroperanden zulassen

6.17 Multiplikationen
Das Ergebnis (der maximalen Breite von 2n) wird in 2 Registern gespeichert (HI und LO)

6.18 Division
Das Ergebnis wird in 2 Registern gespeichert
(LO = Quotient, HI = Rest)

6.19 Zugriff auf LO und HI
Nur über die Befehle mflo/mfhi (laden) und mtlo/mthi (speichern)

6.20 Arithmetische / logische Verknüpfung mit Konstanten
Statt eines dritten Registers kann auch eine Konstante verwendet werden (Befehl + i (immediate))

6.21 Wortbreite bei Konstanten
Konstanten sind immer vorzeichenbehaftet und 16 bit breit (sie werden auf 32 bit erweitert 0xFFFFxxxx)

6.22 32-Bit-Konstante laden
Zum Laden eines kompletten Registers benötigt man 2 Schritte:
1. lui: Setzt die oberen 16 Bits auf den Wert der Konstanten, setzt die unteren 16 Bits auf 0
2. ori: Setzt die unteren 16 Bits

6.25 Adressierung des Speichers beim MIPS
Speicheradresse = Registerwert + Konstante
Beispiel: lw \$9, 4(\$7)

6.26 Vorzeichenerweiterung beim Laden von Werten
Mit Vorzeichenerweiterung: lh, lb
Auffüllen mit Nullen: lhu, lbu

6.38 Sprungbefehle
Ein Programm ohne Sprünge / Verzweigungen / Unterprogrammaufrufe ist nicht denkbar
Unbedingte Sprünge: Das Programm wird auf jeden Fall an einer anderen Stelle ausgeführt
Bedingte Sprünge: Der Sprung wird nur ausgeführt, wenn eine bestimmte Bedingung erfüllt ist

6.39 Unbedingte Sprünge
Befehl j
Sprungziele werden durch "Labels" (...:) markiert oder Programmadressen (0x) angegeben

6.41 Sprünge und Pipelining
Branch Delay Slot: Die Instruktion nach einem Sprungbefehl wird vor dem Sprung ausgeführt (deswegen muss nach dem Sprung ein "nop" (No Operation) folgen)

6.43 Befehl für Vergleiche: SLT
set if less than vergleicht zwei Register und setzt das Zielregister auf 1 falls a¡b gilt, sonst wird das Zielregister auf 0 gesetzt

6.44 Unterprogramme
jal: "Springe in ein Unterprogramm und merke dir die Rücksprungadresse in \$31"
jr: Setze den Programm Counter auf den Registerwert

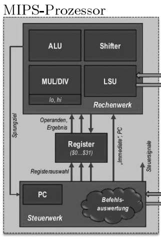
6.50-6.51 Der Stack - Ein LIFO-Speicher
Stapelspeicher, LIFO: Last-In-First-Out
Man benötigt einen Speicherbereich und einen Zeiger zur Adressierung des Speichers (Stackpointer)
Stackpointer: \$29 / \$sp zeigt immer auf den "obersten" Eintrag auf dem Stack
"Der Stack wächst nach unten"

6.52 Arbeiten mit dem Stack
Platz reservieren (\$sp erniedrigen) mit "addi \$sp,\$sp,-4", Daten ablegen mit "sw \$x,0(\$sp)", Daten zurückholen mit "lw \$x,0(\$sp)" und Speicherplatz freigeben (\$sp erhöhen) mit "addi \$sp, \$sp,4"

6.53 Unterprogramme und der Stack
Die Rücksprungadresse wird in \$ra abgelegt, soll im Unterprogramm ein anderes Unterprogramm aufgerufen werden, muss dieser Wert erstmal gespeichert werden

6.54 Übergabe von Parametern
Genutzt werden können Register, gemeinsame Speicherstellen (globale Variablen) oder der Stack

6.66 Modell eines MIPS-Prozessor



7. Speicherhierarchie

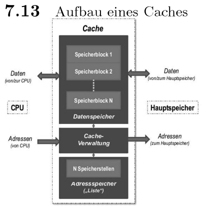
7.2
Register: kleine Speicherkapazität, schneller Zugriff
Hauptspeicher: große Speicherkapazität, langsamer Zugriff

7.5 Speicherarchitektur mit "Zwischenspeicher" (Cache)
Register ↔ Cache ↔ Hauptspeicher

7.6 Lokalitätsprinzip
Temporale Lokalität: "Wenn ein Zugriff auf ein Speicherelement erfolgt, ist die Wahrscheinlichkeit hoch, dass bald wieder ein Zugriff auf dieses Element erfolgt"
Räumliche Lokalität: "Nach einem Zugriff auf ein Element, ist es wahrscheinlich, dass bald ein Zugriff auf ein Element in der Nähe erfolgt"

7.10 Prinzip eines Caches
Der Cache enthält eine Kopie der Daten aus dem Hauptspeicher
Eine spezielle Hardwarekomponente verwaltet eine Liste, in der die Adressen der Hauptspeicherkopien zu finden sind
Für jeden Speicherblock ist ein Eintrag in der Liste vorhanden

7.13 Aufbau eines Caches



7.14-7.18 Cache-Beispiel
Adresswortbreite: 16 bit
Datenwortbreite: 8 bit
Anzahl der Cache-Blöcke: 1
Größe eines Cache-Blocks: 256 Byte
Beim Abrufen einer Adresse wird geprüft ob die Adresse in der Liste ist
Markierung in der Liste "tag", Treffer "hit"

7.19-7.20 Mehrere Hauptspeicherkopien verwalten
Für jeden Speicherblock muss ein Listeneintrag "Cache line" existieren

7.21 Listeneinträge
"Tag": Liegt die Adresse als Kopie vor
"Valid": Wurde der Cache schon geladen
Evtl. "Dirty": Wurde der Cacheinhalt modifiziert

7.22 Write-Through / Write Back
Write-Through: Schreibt sowohl auf den Hauptspeicher als auf den Cache
Write-Back: Falls der Block im Cache modifiziert wurde, schreibe ihn beim Cache neuladen in den Hauptspeicher zurück

7.24 Write-Through mit Write-Buffer
Der Schreibzugriff wird zunächst zwischengespeichert und im Hintergrund auf den Hauptspeicher kopiert

7.25 Zuordnung von Cache-Lines
Direct Mapped: Eindeutige Zuordnung zwischen Hauptspeicherbereich und Cache-Line
Fully Associative: Jeder Bereich kann in einer beliebigen Cache-Line abgelegt werden (dynamisch)
N-way Set Associative: Jeder Bereich kann in N verschiedene Cache-Lines abgelegt werden (dynamisch)

7.28 Cacheverwaltung
Direct Mapped: Beim Speicherzugriff muss die Adresse mit genau 1 Tag verglichen werden. Blockadresse % Anzahl der Cache Lines
Associative: Die Adresse muss mit n Tags verglichen werden. Blockadresse % Anzahl der Sets

7.33 Ersetzungsstrategien
Random: Der Zufallsgenerator entscheidet
LRU (least recently used): Die Cacheline, auf die am längsten nicht zugegriffen wurde wird ersetzt

7.34-7.36 Cache.Hierachie
Um die Anzahl der Cache-Misses zu reduzieren werden mehrere Ebenen an Caches verbaut (klein (CPU) $\rightarrow groß(Hauptspeicher)$)

7.43 Modell des Cache-Verhaltens
Ausführungszeit des Programms: T_{prg}, Anzahl der Taktzyklen zur Ausführung des Programmes: N_{exec}, Anzahl der Wartezyklen auf Grund von Cache-Misses N_{stall}, Taktfrequenz des Systems f_{clk}, $T_{prg} = (N_{exec} + N_{stall})/f_{clk}$

7.44
Speicherstillstands-Zyklen
Anzahl der Speicherzugriffe des Programms N_{mem}, Cache-Miss-Rate für Lesezugriffe R_{miss}, Anzahl der Wartezyklen auf bei Auftreten eines Cache-Miss N_{miss},
$N_{stall} = N_{mem} \cdot R_{miss} \cdot N_{miss}$

8. Parallelverarbeitung

8.4 Aufteilung der Verarbeitungsschritte
"Alle machen alles" oder "Spezialisierung der Verarbeitung"

8.6 Amdahls Law
Entweder es ist keine Parallelverarbeitung im Programm möglich oder es kann für einen Anteil eine ideale Beschleunigung erreicht werden (P)
Parallelität der Hardware n,
$S = \frac{1}{(1-P) + \frac{P}{n}}$

8.13 5-stufige Pipeline
Die Anzahl der Pipeline-Stufe einer CPU ist im Prinzip beliebig wählbar, es sollten sich aber sinnvolle Teilfunktionen ergeben
Fetch: Instruktion lesen
Decode: Instruktion dekodieren und Register lesen
Execute: Operation ausführen
Memory: Speicherzugriff (falls Load/Store-Instruktion)
Write-Back: Schreiben des Ergebnisses in das Ziel-CPU-Register

8.14-8.17 Datenharzard / Kontrollharzard
Ergebnisse einer Instruktion liegen noch nicht vor, wenn sie in einer nachfolgenden Instruktion benötigt werden
Entweder man hält die Pipeline für einen Moment an oder man nutzt das Ergebnis bevor alle Stufen durchlaufen sind

8.18-8.19 Struktueller Harzard
Zwei Instruktionen nutzen gemeinsame Resourcen (zb. die gleiche Speicheradresse)
Hier kann man die Pipeline nur kurzzeitig anhalten

8.20 Zusammenfassung
Mit Pipelining lässt sich die Taktfrequenz einer CPU erhöhen. Damit erhöht sich der Instruktionsdurchsatz.
Idealerweise steigt dieser proportional zu der Anzahl der Pipelinestufen
Abhängigkeiten zwischen Instruktionen führen zu Pipeline-Harzards und reduzieren den Instruktionsdurchsatz
Pipelinestufen n, x% "abhängie Instruktionen, Speed-Up gegenüber nicht gepipelinter CPU:
$S = (1.0 - x) \cdot n + x$

8.22-8.23 Klassifikation nach Flynn
Einfache, griffige Klassifikation von Computer-Architekturen
Einteilung nach Parallelität der Instruktionströme und Datenströme ("Single" oder "Multiple")

8.24 SISD
Skalare (nicht-parallele) Architektur

8.25 MISD
Pipeline-Struktur für ganze Programmteile (Makropipeline)

8.26-8.27 SIMD
Alle Verarbeitungseinheiten führen die gleiche Instruktion aus
Parallelverarbeitung auf "Datenebene"
Es wird ein Compiler mit "Auto-Vectorization" benötigt

8.28 MIMD
Mehrere unabhängig voneinander arbeitende Verarbeitungseinheiten (Multicore Systeme)

8.29-8.32 Multithreading
Möglichkeiten
Coarse-Grained: Umschaltung von Threads bei Stillständen
Fine-Grained: Umschaltung von Threads nach kurzen Zeitabständen
Simultaneous Multithreading (SMT): Die Issue Slots der CPU können gleichzeitig von mehreren Threads belegt werden

/ DOMENIK KRANKE /