Grundlagen Rechnertechnik

1. Inhaltsverzeichnis

2. Grundlagen

2.1. Prinzipieller Aufbau

Jeder heute übliche Digitalrechner besteht aus den Komponenten:

CPU	– zentrale Verarbeitungseinheit – übernimmt sowohl die eigentliche Datenverarbeitung als auch die Steuerung – Koordination aller rechnerinternen Abläufe
Speicher	– nimmt die zu verarbeitenden Daten und das Programm auf, sowie weitere Informationen – alle Informationen werden in einem gemeinsamen Speicher untergebracht
IOP	– I/O Prozessoren → programmgesteuerte Baugruppen die eigenständig I/O-Aktivitäten durchführen – erhalten von CPU entsprechenden Auftrag und melden der CPU den Abschluss zurück
I/O-Geräte	– dienen als Schnittstelle zur Außenwelt, Anzahl und Ausstattung hängen stark vom jeweiligen Verwendungszweck des Rechners ab
BUS	– realisiert die Datenübertragung zwischen den einzelnen Baugruppen – Übertragung erfolgt über gemeinsam genutzte Datenpfade, an welche alle Komponenten gleichzeitig angeschlossen sind – Koordinierung von Übertragungsrichtung und zeitlichem Ablauf, sowie Auswahl von Sender und Empfänger erfolgt über besondere Steuerleitungen – 2 verschiedene Konzepte kommen hier zum Einsatz: – **Master/Slave** o gesamter rechnerinterner Ablauf unterliegt einem Schema, die CPU steuert als Master aktiv alle Abläufe + Datentransporte o Speicher und periphere Komponenten arbeiten als Slave → sind stets passiv und werden nur auf Anforderung der CPU tätig – **Interrupt-Konzept** o Erweiterung des Master/Slave Konzeptes o periphere Slave-Einheiten stellen Bedienanforderung (interupt request) an die CPU o CPU entscheidet, über Berücksichtigung und Zeitpunkt der Bearbeitung der Anforderung

Eine CPU besteht im Wesentlichen aus:

Recheneinheit	– nimmt eigentliche Datenverarbeitung vor – beinhaltet Registersatz (kleine Speicher für Zwischenspeicherung von Werten) – beinhaltet ALU (Arithmetic Logical Unit)
Steuereinheit	– kontrolliert und steuert die Aktivitäten der Recheneinheit als auch die übrigen Rechnerbestandteile – besteht im wesentlichen aus Befehlsdekodierer und einem weiteren Prozessor internen Register, den so genannten Programm-Counter / Intruction Pointer / Befehlszähler

2.2. Technische Grundlagen

Die Weiterentwicklung auf dem Gebiet der Mikroelektronik führte im Laufe der Zeit zu einer Spezialisierung der Prozessoren, folgende Einteilung ergibt sich heute:

Standardprozessoren	– auf nichts spezialisiert, können alles aber nichts besonders gut
Hochleitungsprozessoren	– spezialisiert, hohe Taktraten, hoher Datendurchsatz
Mikrocontroller	– kleine Prozessoren für Embedded Anwendungen, Gerätesteuerung – vollständige Mikrocomputersysteme auf einem Chip – neben CPU und Speicher sind auch E/A-Komponenten auf dem Chip integriert
DSP Digitale-Signal-Prozessoren	– sehr spezialisiert auf bestimmte Aufgaben, dadurch hohe Geschwindigkeit, z.B. für Spracherkenndung, Mustererkennung – Spezialprozessoren für die schnelle Verarbeitung mathematischer Befehle zur Abarbeitung komplexer Algorithmen der analogen Signalverarbeitung

Allen bisher betrachteten Rechnersystemen liegt die Annahme zugrunde:
Alle zu verarbeitenden Signale/Daten liegen als Binärwert vor!

Diese Beschränkung auf 2 Zeichen, hat in der Praxis den großen Vorteil, das sich aus einer kleinen Menge logischer Grundfunktionen mit „etwas“ Schaltalgebra alle komplexen Funktionen herleiten lassen.

Die logischen Grundfunktionen:
- Negation (NOT) $y = \overline{y}$
- Konjungtion (UND) $y = x \wedge y$
- Disjunktion (OR) $y = x \vee y$

Die Schaltalgebra ist die mathematische Grundlage der Digitaltechnik, Ihre Aufgaben sind:
- Beschreibung von Signalverknüpfungen in digitalen Schaltungen
- Schaltungsentwurf bei vorgegebener Funktion
- Minimierung des technischen Aufwands

2.3. Schaltkreissysteme

Schaltnetze	=	speicherfreie Schaltungen
Schaltwerke	=	speicherhaltigen Schaltungen

2.4. Zahlensysteme

Alle modernen Zahlensysteme sind Stellenwertsysteme, d.h. der Wert einer Ziffer innerhalb einer Zahl hängt von Ihrer Position ab.
Bsp.: 11234_{10} 11234_{16}

Beispiel für nicht stellenwertbasiertes System, sind die römischen Zahlen.

Die elektronische Darstellung von Zahlen geschieht im Rechner als sog. Dual-/Binärzahl, d.h. als Zahl zur Basis 2. Prinzipiell macht es keinen Unterschied, ob eine Zahl zur Basis 2 oder 10 dargestellt wird.
Der nutzbare Zahlenbereich im Binärsystem liegt von:

$$0 \; bis \; (2^{n-1}-1) \quad (positive \; Zahlen)$$

$$(-2^{n-1}) \; bis \; -1 \quad (negative \; Zahlen)$$

Das 2-er Komplement gewinnt man in 2 Schritten:

1. Bildung des Einer-Komplements (alle Bits invertieren)
2. Addition von 1

Beispiel: gesucht Zweierkomplement von -3

3_{10}	0011_2
Einerkomplement →	1100
Addition von 1:	0001
	$1101 \rightarrow -3$

Rechnerinterne Darstellung von Dezimalzahlen bereitet allerdings etwas Schwierigkeiten, da ein Digitalrechner nur n-verschiedene Stellen (Bitbreite) hat. Damit können genau 2^n unterschiedliche Zahlen dargestellt werden. Je nach Aufgabenstellung kommen verschiedene Zahlencodierungen zur Anwendung:

- Integer (ohne Vorzeichen)
- Zweierkomplement (Integer mit Vorzeichen)
- Fließkommazahlen (Floating Point)

Integerdarstellung erlaubt bei n Stellen 2^n verschiedene Zahlen, z.B. bei n=4 Zahlenbereich von 0…15.

Die Zweierkomplementdarstellung erlaubt auch die Nutzung negativer Zahlen, das höchstwertigste Bit dient als Vorzeichen, Beispiel:

Dez.	**1er Komplement**	**2er Komplement**
-2	1101	1110
$+3$	0011	0011
$+1$	10000	10001

2.4.1. Floating Point Darstellung

- Entspricht der gewohnten wissenschaftlichen Darstellung einer Zahl mit Mantisse und Exponent, allerdings zur Basis 2

Bsp.: $768 = 7,68 \cdot 10^2$

$768_{10} = 1,5 \cdot 2^9$

Jede Floating Point Zahl besteht immer aus 2 Teilen, Mantisse und Exponent.

Die Umwandlung einer Dezimal in eine FP-Zahl geschieht durch wiederholtes Teilen mit 2, bis das Ergebnis zwischen 1 und 2 liegt:

Beispiel:	$45_{10} \rightarrow 22,5 \cdot 2^1$ $11,25 \cdot 2^2$ $5,625,5 \cdot 2^3$ $2,8125 \cdot 2^4$ $1,40625 \cdot 2^5$ → Komma um 5 Stellen nach rechts verschieben	$45_{10} = 101101$ 1,01101 $45_{10} = 1,011101 \cdot 2^5$ Für die Mantisse stehen allerdings nur 4 Bit zur Verfügung: $45_{10} \hat{=} 1,1011 \cdot 2^5$

Mit dieser Darstellung kann man nun einen größeren Wertebereich abdecken, als mit 2-Komplement-Zahlen:

$|x| = Mantisse \cdot 2^{Exponent}$

$Mantisse \in \{1...1,999999\}$

$Exponent \in \{2^{-8}...2^0...2^7\}$

$positiv\ (größte)\ Zahl \approx 2 \cdot 2^7 = 2 \cdot 128 = 256$

$positiv\ (kleinste)\ Zahl \approx 1 \cdot 2^{-8} = 1/256 = 0,003906$

$negativ\ (größte)\ Zahl \approx -1 \cdot 2^{-8} = -1/256 = -0,003906$

$negativ\ (kleinste)\ Zahl \approx -2 \cdot 2^7 = -2 \cdot 128 = -256$

Diese Art der Darstellung birgt jedoch Rechenungenauigkeiten, z.B.:

$45_{10} \cdot 18_{10} = 810_{10}$

$45_{10} \hat{=} 1{,}011 \cdot 2^5$

$18_{10} \hat{=} 12_{16} \hat{=} 10010_2 \hat{=} 1{,}001 \cdot 2^4$

1.) Exponenten addieren	**2.) Exponenten addieren**
1,011*1,001 101100 1011 1,100011 → neue Mantisse: 1,100	0101 0100 1001→9_{10}
Gesamtergebnis: $1{,}100 \cdot 2^9 \hat{=} 1{,}5 \cdot 512 = 768_{10}$	**Vergleich:** Ungenauigkeiten beim Rechnen mit FP-Zahlen!

2.4.2. Interne Speicherung einer Floating-Point-Zahl

- Mantisse bei FP-Zahl ist immer zwischen 1 und 2 → Vorkommateil immer 1
- Vorkommateil muss daher nicht abgespeichert werden
- Ein übliches Format für die FP-Darstellung besteht aus 33 bit:

VZ	Exponent	Mantisse
1 bit	8 bit	(m_{23})… 24 bit ……..(m_3) (m_2) (m_1) (m_0)

(m_{23}) wird nicht abgespeichert

Folgender Wertebereich ergibt sich bei dieser Darstellung:

$|x|_{max} = 2 \cdot 2^{127} = 2^{128} = 3{,}4 \cdot 10^{38}$

$|x|_{min} = 1 \cdot 2^{127} = 2^{-128} = 5{,}88 \cdot 10^{-39}$

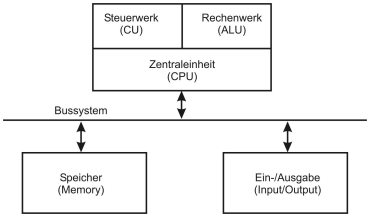
Der Wertebereich des Exponenten geht nur von +127 bis -127. Der Exponent -128 wird benötigt um die Zahl 0 darzustellen.

3. Mikroprozessorarchitekturen

- Ziel einer Mikroprozessorarchitektur ist es ein beliebiges Anwenderprogramm möglichst schnell auszuführen
- Architekturbeschreibung liefert Anhaltspunkte, aber keine quantitative Aussage über die Leistungsfähigkeit eines Mikroprozessors

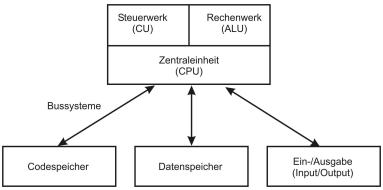
3.1. Von-Neumann-Architektur

- Charakterisiert durch 3 Grundbestandteile:
  - Zentraleinheit mit Steuerwerk / Rechenwerk
  - Speicher
  - E/A



Die Verbindung der 3 Komponenten erfolgt über ein Bussystem, weitere wichtige Merkmale der Architektur sind:

- Unabhängigkeit der Rechnerstruktur von speziell zu bearbeitenden Aufgabenstellungen: für jedes Problem (Aufgabe) wird ein eigenes Programm geladen → programmgesteuerter Universalrechner
- Programme und Daten stehen im selben Speicher
- Speicher verfügt über Speicherplätze mit gleicher Wortlänge
- Jede Speicherstelle hat ihre eigene Adresse, über welche sie angesprochen wird
- Alle Befehle werden sequentiell abgearbeitet
- Wichtige Variante der Von-Neumann-Architektur stellt die Harvard-Architektur dar, deren Merkmale sind:
  - Es gibt **einen** Datenspeicher und **einen** Programmspeicher
  - Verbindung zu Daten-/Programmspeicher u. E/A-Geräten erfolgt über eigene Bussysteme
- Heutzutage werden insbesondere Spezialprozessoren nach einem etwas modifizierten Hardvard-Konzept gebaut, Ziel ist die Erhöhung des Datendurchsatzes



3.2. Klassifizierungsschema

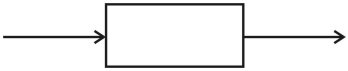
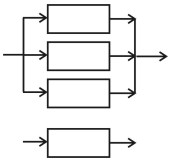
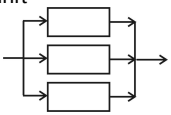
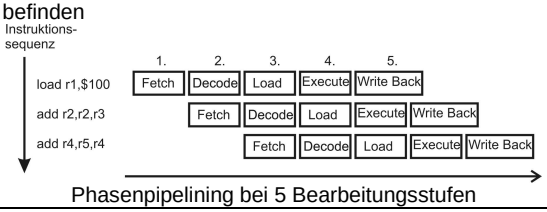
Klassifizierung erfolgt nach Gleichzeitigkeit von Daten und Befehlen:

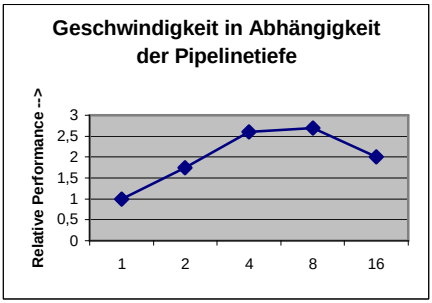
- SISD
- MISD
- SIMD
- MIMD

- SI Single instruction = ein Befehl zu einem Zeitpunkt
- SD Single data = ein Datenwert wird zu einem Zeitpunkt bearbeitet
- MI Multiple instruction = mehrere Befehle zu einem Zeitpunkt
- MD Multiplipla data = es werden mehrere Datenworte zu einem Zeitpunkt bearbeitet

	Einfacher Datenstrom **(SI)**	Mehrfacher Befehlstrom **(MI)**
Einfacher Datenstrom **(SD)**	SISD *Von-Neumann-Architektur*	MISD *Nicht verwendet in Praxis*
Mehrfacher Datenstrom **(MD)**	SIMD *Vektorrechner*	MIMD *Mehrprozessorsysteme*

3.2.1. Wichtige Begriffe zum Thema

Serialität	▪ verschiedene Aktionen können nur nacheinander ausgeführt werden 
Parallität	▪ wenn zu einem Zeitpunkt mehr als nur eine Aktion durchgeführt werden kann 
Nebenläufigkeit	▪ hier werden alle möglichen parallelen Aktionen tatsächlich gleichzeitig ausgeführt 
Pipelining	▪ eine bestimmte Aktion lässt sich in n-Teilschritte zerlegen, die alle nacheinander durchlaufen werden müssen ▪ Pipelining liegt dann vor, wenn sich 2 (oder mehr) Aktionen zu einem bestimmten Zeitpunkt in unterschiedlichen Teilschritten befinden  Phasenpipelining bei 5 Bearbeitungsstufen



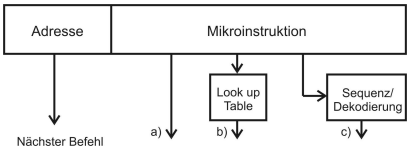
3.3. CISC

Complex
Instruction
Set
Computer

Gründe für die Entwicklung von CISC-Prozessoren:
- der Geschwindigkeitsunterschied zwischen Hauptspeicher und Prozessor, hier ist es sinnvoll möglichst viele Aktionen in einen Maschinenbefehl zu „packen“
- die kompakte Programmierung spart Speicherplatz
- Mikroprogrammierung: ist für komplexe Befehle unabdingbar; ermöglicht außerdem eine Abwärtskompatibilität innerhalb von Prozessorfamilien

Was ist ein Mikroprogramm?
- Darstellung eines Algorithmus, der von den Hardwarekomponenten eines Prozessors ausgeführt wird.
- kann *fest verdrahtet oder programmierbar* sein.
- besteht aus der sequentiellen Abfolge von Mikroprogramm-Operationen die sich ihrerseits aus Steuersignalen (Picobefehle) für die Prozessorhardware zusammensetzen.

3 Arten der Mikroprogrammierung werden unterschieden:
- *a) horizontal:* Picobefehle liegen uncodiert im Mikroprogrammbefehl vor
- *b) quasi-horizontal:* der Picocode ist in Tabellen abgelegt, der Mikroprogrammbefehl dient als Pointer in diese Tabelle
- *c) Mikroprogrammcode durchläuft einen Decoder*, der die erforderlichen Picobefehle generiert



Vorteil: Flexibilität, nahezu jede Instruktion ist realisierbar
Nachteil: Fehleranfälligkeit, Programmierung auf Mikroprogrammebene ist sehr komplex

Kennzeichen eines CISC-Prozessors:
- Universalprozessoren mit Mikroprogrammierung
- Komplexe Befehle ermöglichen mehrfache Operationen auf wenige Daten oder auf umfangreiche Datenfelder
- Komplexe arithmetische Befehle werden in einem Befehl zusammengefasst
- umfangreicher Befehlssatz
- kompakter Programmcode
- Mikroprogrammcode benötigt viel Fläche auf dem Prozessorchip

Derzeitige Grenzen des CISC-Konzeptes:
- Geschwindigkeitsunterschied zwischen Prozessor und Hauptspeicher ist durch neue Speichertechnologien reduziert
- Verwendungshäufigkeit der Befehle
  - Nutzungsuntersuchung an CISC-Prozessor mit ca. 200 Befehlen ergaben:
    ca 80% des Programmcodes nutzt 10 Befehle
    ca. 95% des Programmcodes nutzt 21 Befehle
    ca. 99% des Programmcodes nutzt 30 Befehle

3.4. RISC-Prozessoren

Reduced
Instruction
Set
Computer

Entwurfsmerkmale:
- 1-Zyklus Befehle, pro Takt soll ein Maschinenbefehl ausgeführt werden
- keine Mikroprogrammierung um den Dekodier- und Steueraufwand im Prozessor möglichst gering zu halten
- einheitliches Befehlsformat um den Dekodieraufwand für alle Befehle zu reduzieren
- Load/Store Architektur: Datenverarbeitung wird nur auf Prozessorinternen Registern ausgeführt, der Zugriff auf den Hauptspeicher erfolgt nur über ein Schnittstellenregister

Forderung nach 1-Zyklus-Befehlen erforderte neue Designkonzepte:
- interne Taktvervielfachung
- Phasenpipelining: jeder Befehl lässt sich in unterschiedliche immer wiederkehrende, gleiche Bearbeitungsphasen zerlegen

Eine solche Aufteilung kann z.B. sein:
- FETCH (Befehl hereinholen)
- DECODE (Was ist zu tun?)
- LOAD OPERANDS (Operanden laden)
- EXECUTE (Ausführung des Befehls)
- WRITE BACK (Aufräumen, Ergebnisse zurückschreiben)

verschiedene Hardware jeweils an Operation beteiligt ↓

In der Praxis muss die Zerlegung in einzelne Bearbeitungsphasen so gewählt werden, dass alle Phasen möglichst gleichlang dauern. Außerdem müssen die pro Phase benutzten HW-Komponenten exklusiv dieser Phase zuordnungsfähig sein.

Heutzutage werden Prozessoren mit mehr als 10 Stufen als „Super-Pipelining-Prozessoren“ bezeichnet. Für das Pipelining gibt es zwei begrenzende Faktoren:
- Buszugriffe sind nicht mehr weiter unterteilbar, da die Leiterbahnen während des Speicherzugriffs exklusiv genutzt werden
- Daten- und Kontrollflussabhängigkeiten: wenn nachfolgende Befehle auf den Ergebnissen oder Zuständen des vohergehendes Befehls aufbauen

Kennzeichen für RISC-Prozessoren:
- wenige Maschinenbefehle (< 150)
- wenige Befehlsformate (< 4)
- ein Befehl pro Taktzyklus
- wenige Adressierungsarten (< 4)
- viele Prozessorregister
- Speicherzugriffe nur über Load-Store-I/F-Register

3.5. Superskalare Architekturen

Prozessoren die mit Hilfe interner Einheiten mehr als einen Befehl pro Takt verarbeiten, werden als superskalare Prozessoren bezeichnet. Dies kann auf 2 Wegen geschehen:
- Hardwarevervielfachung im Prozessor

- Zusammenspiel von Compiler und Prozessor: Compiler fasst geeignete Befehle zusammen und macht daraus einen langen Befehl (VILW → very long instruction word)

Die superskalare Ausführung eines Programms bedeutet, dass die Instruktion zur gleichen Zeit oder in veränderter Reihenfolge ausgeführt wird.

Um weiterhin die bekannten Programmiersprachen und Methoden, insbesondere das Prinzip der sequentiellen Befehlsverarbeitung anwenden zu können, sind zusätzliche Hardwareeinheiten zur Kontrolle und Steuerung der korrekten Befehlsreihenfolge erforderlich.

Das Problem der Programmverzweigungen versucht man durch zwei Maßnahmen in seinen Auswirkungen zu mildern:

- ***Branch-Prediction*** (spekulative Befehlsausführung / dynamische Sprungvorhersage)
  - Mit Hilfe eines internen Speichers für das letzte Sprungziel (Branch-Target-Cache) und entsprechender Algorithmen zur Sprungvorhersage ist es heutzutage möglich, dass die Wahrscheinlichkeit für eine Unterbrechung des Arbeitsablaufes < 5% liegt.
- ***beide Alternativen werden dekodiert*** / bearbeitet
  - erfordert die mindestens doppelte Auslegung aller wichtigen Funktionen im Prozessor

Schwieriger zu lösen sind Probleme durch Datenabhängigkeiten: $(a+b)\cdot(c-d)$ .Hier
Hier hilft meist nur geeignetes Scheduling der Informationen und die Out-of-Order-Execution.

3.6. Signalprozessoren

DSPs sind für den Bereich der digitalen Signalverarbeitung optimiert, unter „digitaler Signalverarbeitung“ versteht man die numerische Verarbeitung abgetasteter Signale.
Hauptmerkmale dieser Prozessoren: Große Datenmengen sowohl bei der Eingabe als auch bei der Ausgabe.

Für den hohen Datendurchsatz kommt heute meist das modifizierte Harvard-Konzept zum Einsatz, es besitzt mehrere voneinander unabhängige Busse für Daten und Programme.
Die eigentliche Modifikation: große Teile des Speichers sind auf dem Prozessorchip integriert.
DSP weisen aufwändige und schnelle Adress-ALUs auf. Die Daten ALUs verfügen im allgemeinen über spezielle Multiplikationsfähigkeiten (MAC-Befehle); entsprechend verfügt der Befehlssatz über angepasste Befehle.

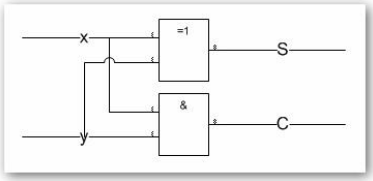
4. Aufbau einer Zentraleinheit

Grundsätzlicher Funktionsablauf in jeder CPU:

- Befehl adressieren
- Befehl holen
- Befehl dekodieren || Befehlszähler inkrementieren
- Befehl ausführen

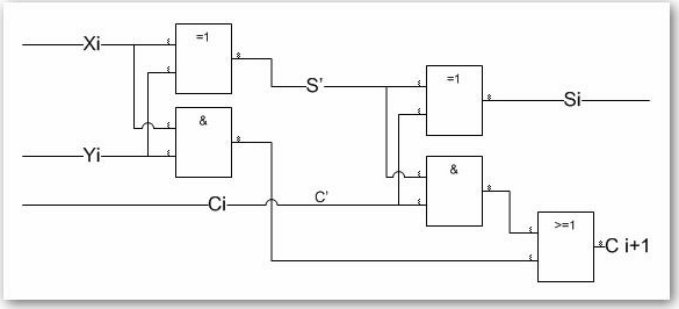
4.1. Addition

4.1.1. Halbaddierer



x	y	S=x+y	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

4.1.2. Volladdierer



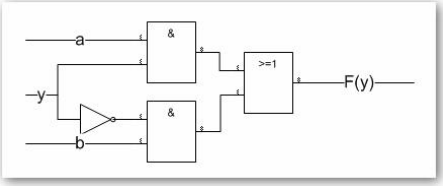
x_i	y_i	c_i	**Wunschfunktion** $s_i=x_i+y_i+c_i$	c_i+1	$s'=x_i+y_i$	$c'=x_i \wedge y_i$	$c''=s' \wedge c_i$	$c_{i+1}=c' \vee c''$	$s_i=s'+c_i$
0	0	0	**0**	**0**	0	0	0	0	0
0	1	0	**1**	**0**	1	0	0	0	1
1	0	0	**1**	**0**	1	0	0	0	1
1	1	0	**0**	**1**	0	1	0	1	0
0	0	1	**1**	**0**	0	0	0	0	1
0	1	1	**0**	**1**	1	0	1	1	0
1	0	1	**0**	**1**	1	0	1	1	0
1	1	1	**1**	**1**	0	1	0	1	1

4.2. Subtraktion

7 – 3 = 7 + (-3)

Darstellungsmöglichkeiten negativer Zahlen:
Einerkomplement $-x \hat{=} \overline{x}$
Zweierkomplement $-x \hat{=} \overline{x}+1$

→ Subtraktion kann durchgeführt werden als Addition des invertierten Subtrahenden.

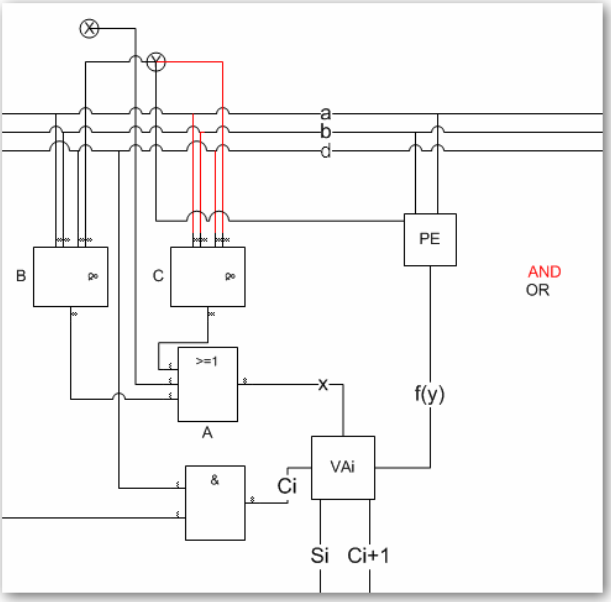


Gewünschte Übertragungsfunktion:

y				
f(y)	0	y	$\overline{y}$	1

a	b	$b \wedge y$	$b \wedge \overline{y}$	or
0	0	0	0	0
0	1	0	$\overline{y}$	$\overline{y}$
1	0	y	0	y
1	1	y	$\overline{y}$	1

Volladdierer mit parametrischer Eingangsschaltung:



- mit Carry-Bit Übertragung = arith. Operationen
- ohne Carray-Bit Übertragung = log. Operationen

→ Steuersignal (**d**)

Wahrheitstabelle

a	b	c_0	f(y)	s = x + f(y) + c0	
0	0	0	0	x	NOP
0	0	1	0	$x+1$	Inkrement
0	1	0	$\overline{y}$	$x+\overline{y}=$ Subtr. im 1er-Kompl.	(Subtraktion im 1er-Komplement)
0	1	1	$\overline{y}$	$x+\overline{y}+1=$ $x+(\overline{y}+1)=$ $x-y=$Subtraktion	Subtraktion
1	0	0	y	$x+y$	Addition
1	0	1	y	$x+y+1$	(Addition + 1)
1	1	0	1	$x+(-1)=x-1$	Dekrement
1	1	1	1	$x+(-1)+1=x-1+1=x$	NOP

4.2.1. Logische Operatoren

a	b	c	d	f(y)	$S=x_i+f(y_i)$
0	0	-		0	$x_i+0=x$
0	1	-		$\overline{y}$	$x_i+\overline{y}_i=Gleichheit\ (A)$
1	0	-		y	$x_i+y_i=XOR(B)$
1	1	-		1	$x_i+1\hat{=}\overline{x}=NOT\ (C)$

x	y	A $\overline{y}$	B $x+\overline{y}$	C $x+1$
0	0	1	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	0	0

Mit den gezeigten Schaltungsergänzungen sind 4 logische Operationen möglich geworden:

- Identität (x=x)
- Gleicheit (x=y)
- XOR (x xor y)
- NOT ($x \rightarrow \overline{x}$)

Die Funktionen für AND und OR werden mittels einer Schaltungsergänzung im x-/y-Eingang des Rechenwerks realisiert.

a	b	c	d	Funktion
0	0	-	0	~~Identität~~ OR
0	1	-	0	~~Gleichheit~~ AND
1	0	-	0	EXOR
1	1	-	0	NOT
0	0	0	1	
0	1	0	1	
1	0	0	1	

…

Wahrheitstabelle für **AND** / OR

AND: $x \wedge y\bar{a}b\bar{d}$

OR: $x \vee y\overline{\overline{abd}}$

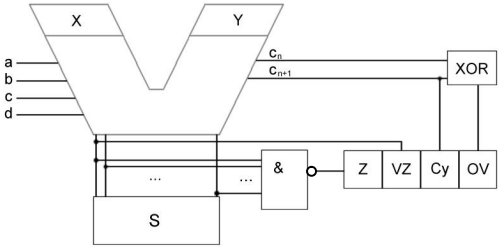
a	b	d	A	B	VA
0	0	0	$x \vee y$	y	$(x \vee y) + 0$
0	1	0	x	0	
1	0	0	x	0	
1	1	0	x	0	

a	b	d	C	A	VA
0	0	0	0	$x \vee y$	$x \vee y$
0	1	0	$\bar{y}$	$x \vee \bar{y}$	$(x \vee \bar{y}) + \bar{y}$
					$= x \wedge y$
1	0	0	0	x	
1	1	0	0	x	

a	b	d	$\overline{\overline{abd}}$	B (…)y	$\bar{a}\bar{b}\bar{d}$	C (…)y	A = x v B v C
0	0	0	1	y	0	0	$x \vee \bar{y} \vee 0 = x \vee y$
0	0	1	0	0			$x \vee 0 \vee 0 = x$
0	1	0			1	$\bar{y}$	$x \vee 0 \vee \bar{y} = x \vee \bar{y}$
0	1	1			0	0	$x \vee 0 \vee 0 = x$
1	0	0					
1	0	1					
1	1	0					
1	1	1					

4.3. Statusregister

Beim Rechnen mit diesem Rechenwerk sind Flags nötig, die den Zustand des Rechenwerks zum Leitwerk mitteilen.
Vereinfachtes ALU-Blockschaltbild mit Statusregister:



c_n	c_{n+1}	Beispiel-aufgabe	OV
0	0	3 - 8	0
0	1	4 + 5	1
1	0	-2 – 7	1
1	1	-2 + 5	0

4.4. Zusammenfassende Beschreibung eines einfachen Rechenwerks

- setzt sich aus n Volladdierern zusammen
- es hat 2 Steuerleitungen zur Operationsauswahl, eine Steuerleitung (d zur Umschaltung Logik / Artithmetik
- Steuerleitung (d zum setzen des Eingangscarrybits
- verfügt über ein Schaltwerk im y-Eingang (Subtraktion
- ein Schaltwerk im x-Eingang (AND, OR)
- Statusflagregister

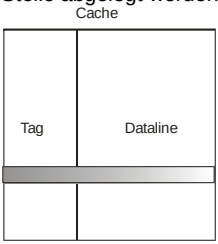
Durchführbare Operationen:

a	b	c	d	Operation
1	0	0 / 1	1	Addition (mit / ohne Eingangscarry)
0	1	1	1	Subtraktion (Zweierkomplement)
0	1	0	1	Subtraktion (Einerkomplement)
0	0	1	1	Inkrement
1	1	0	1	Dekrement
0	0	-	0	OR
0	1	-	0	UND
1	0	-	0	EXOR
1	1	-	0	Einerkomplement (NOT)
0	0	0	1	NOP
1	1	1	1	

<TDO>

Es gibt 3 mögliche Cache-Organisationsformen:

1. voll assoziativ → jeder Hauptspeichereintrag kann an jeder beliebigen Cache-Stelle abgelegt werden



2. direkt abbildend (Direct Mapped Cache) → jeder Hauptspeichereintrag kann nur an einer ganz bestimmten Stelle im Cache abgelegt werden. Grund: Teile der Originaladresse werden auch zur Adressierung des Caches genutzt.
   Bsp.: Hauptspeicher = 65536 * 1 Byte
   Cache = 1024 * 4 Byte
   Bei 1024 Datalines im Cache konkurrieren 16 verschiedene Hauptspeicherstellen um die gleiche Cache-Line.
3. n-Wege assoziativ → Mischung aus beiden Verfahren: hier hat eine Hauptspeicherstelle **n** verschiedene Möglichkeiten, im Cache abgelegt zu werden.

Wichtige Randbedingungen beim Einsatz von Cache-Speichern:
Cache-Einsatz in EIN-Prozessorsystemen:

- veränderte Cache-Einträge müssen erst dann im Hauptspeicher nachvollzogen werden, wenn die betreffende Cache-Line durch andere Inhalte ersetzt werden soll.

Cache-Einsatz in MEHR-Prozessorsystemen:

- es muss sichergestellt sein, dass alle Komponenten, die auf den Hauptspeicher zugreifen dürfen (2. CPU, DMA …), dort auch die aktuell gültige Information vorfinden.
- In Mehrprozessorsystemen muss ein Datenkohärenz-Protokoll laufen, welches dafür sorgt, dass jede Veränderung an gemeinsam genutzten Daten den anderen Nutzern mitgeteilt wird.

Dieses Kohärenzprotokoll überwacht und verändert die Statusinformationen im Tag-Bereich der angeschlossenen Caches. Häufig benutzt: **MESI-Protkoll**

Modified	Daten nur in einem Cache, aber bereits verändert
Exclusive	Daten nur in einem Cache
Shared	Eintrag in mehreren Caches vorhanden
Invalid	invalid

H/W-Voraussetzung für ein solches Protokoll: „Snooping Logic“ = Abhören der Adressleitungen.

Es gibt 2 verschiedene Möglichkeiten das MESI-Protokoll zu implementieren:
- WRITE INVALIDATE
- WRTE BROADCAST

5. Hauptspeicher

Der Hauptspeicher spielt in der Speicherhierarchie die zentrale Rolle: Er beinhaltet die „Original"-Daten und Programme, auf die sich alle beteiligten Komponenten abstützen. Üblicherweise wird heutzutage der Hauptspeicher mit DRAM-Chips (Dynamic RAM) aufgebaut.

DRAM-Eigenschaften:
- billig
- große Kapazitäten auf kleinem Raum
- Speicherung mittels von Kondensatoren, dadurch hoher Aufwand (Refresh der Speicherzellen notwendig), aber wenig Komplexität beim Aufbau (1 Kondensator pro Bit).

SRAM (Static RAM):
- teuer
- groß
- Speicherung durch FlipFlops, Information bleibt bei angelegter Spannung erhalten, aber Aufbau eines FlipFlops hat eine höhere Komplexität.

DRAMs haben den großen Nachteil, dass sie nach einer gewissen Zeit die gespeicherte Information „vergessen" haben, außerdem müssen sie nach dem Lesezugriff neu beschrieben werden. Daher unterscheidet man bei DRAMs die sogenannte Zugriffszeit und die sogenannte Zykluszeit.

Zugriffszeit: nach der der Dateninhalt an die CPU zurückgeliefert wird.
*Zykluszeit:*das Zeitintervall, nachdem erneut auf den Speicherbaustein zugegriffen werden kann.

Bedingt durch diese Eigenschaften sind Standard-DRAMs etwa 10x langsamer als Standard SRAMs. Umgekehrt benötigt der komplexe Zellenaufbau von SRAMs etwa 4 – 8 mal soviel Chipfläche.
Der große Geschwindigkeitsnachteil von DRAMs wird heutzutage weitgehend kompensiert durch verbesserte / optimierte Zugriffsarten.

DRAM-Speicher sind intern als rechteckige Gitter aus Zeilen (Rows) und Spalten (Columns) organisiert. Dafür werden sog. RAS-Signale (Row Access Signal) und CAS-Signale (Column Access Signal) benutzt.

Die Adresse wird in 2 Teilen hintereinander an den Chip übertragen:
- Nibble-Modus: liefert für jedes angelegte RAS-Signal sofort die 3 nächsten Bit mit, ohne dass die RAS nochmals gesetzt werden muss.
-