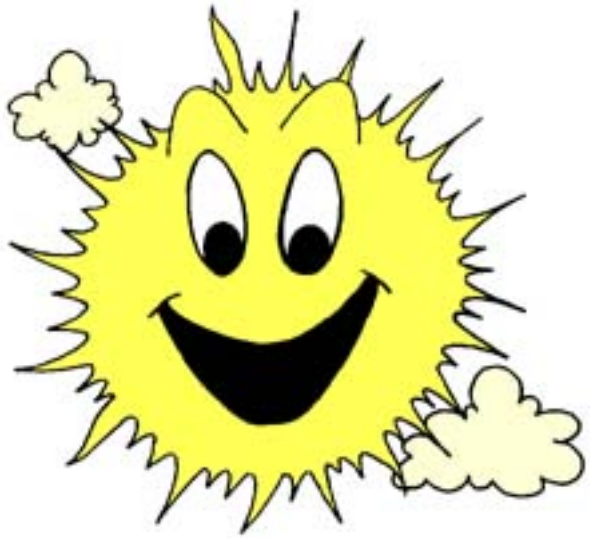
Wiederholung der 6. Vorlesung



- Entropie
- Redundanz
- Wirkungsgrad
- Huffman-Codierung

4.3 Entropie

- Haben mögliche Ereignisse oder Zustände eines Systems verschiedene Wahrscheinlichkeiten, wird aus diesen der mittlere Informationsgehalt bestimmt. ➔ Entropie (aus Thermodynamik)
- Entropie (gr., „sich zu etwas hinwenden“)
- natürliche Systeme streben Gleichgewicht an.
- je wahrscheinlicher ein Zustand desto größer dessen Entropie (= Maß für Unordnung)

Informatik

Der **mittlere Informationsgehalt** einer Informationsquelle nennt man **Entropie H**

$$H = \sum_i I(N_i)\, p_i = \sum_i p_i\, ld(1/p_i)$$

$$= -\sum_i p_i\, ld(p_i)\ ;\ i = 1, 2, ..., n$$

wenn ferner gilt $\sum_i p_i = p_1 + p_2 + ... + p_n = 1$

- **Entropie** ist **maximal**, wenn alle Zustände (Nachrichten) gleich wahrscheinlich sind,
- wenn gilt: $\mathbf{p_i = 1/n}$ für alle i = 1, ..., n
- dann gilt $H_{max}(n) = H_0(n)$

4.4 Redundanz

- Redundanz ist die Differenz aus maximalem und tatsächlichem Informationsgehalt.
- Redundanz bedeutet „Weitschweifigkeit“, „Überfluss“ (lat. redundare)
- Redundanz kann nützlich sein. Störungen können ggf. kompensiert werden:
  - Vorlesung am Mittwoch, den 24.11.04 im 2. Block von 10:15 – 11:45 ...
  - %orle{%ung am M[@~woch, den 2?.11.04 im 2. Bl!“k von 1-:ß5 – |1:45 ...

4.4 Redundanz (2)

- Die Maße **Redundanz** und **Wirkungsgrad** charakterisieren eine Informationsquelle
- Redundanz: $\mathbf{R = |H(n) - H_0(n)|}$
- Wirkungsgrad: $\eta = \mathbf{H(n) / H_0(n)}$

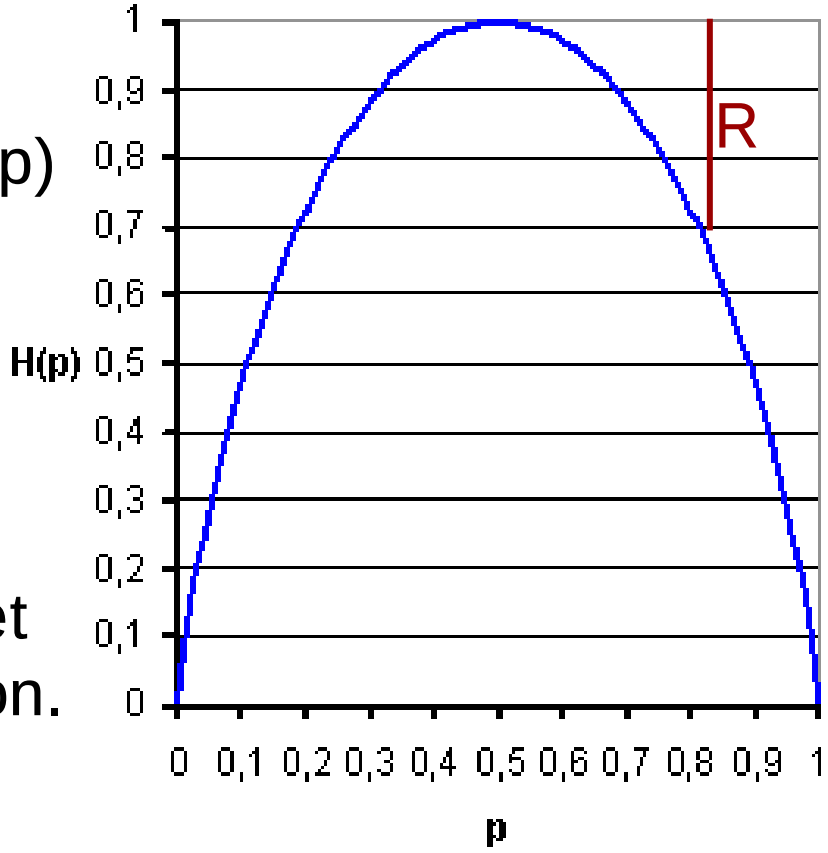
Z	p_i	$-\mathrm{ld}\,p_i$	$\lceil -\mathrm{ld}\,p_i \rceil$
a	0,4	1,322	2
b	0,2	2,322	3
c	0,2	2,322	3
d	0,1	3,322	4
e	0,1	3,322	4

$H_0(5) = 2{,}322$
$H(5) = 2{,}122$
Redundanz:
$R = 0{,}200$
Wirkungsgrad:
$h = 0{,}914$

4.4 Shannon-Funktion

Sendet eine Nachrichtenquelle zwei Signale (0 und 1) aus, so tritt 0 mit der Wahrscheinlichkeit p und 1 mit der Wahrscheinlichkeit $q = 1-p$ auf. H hängt also von p ab.

$$H(p) = -(p \text{ ld } p + q \text{ ld } q)$$
$$= -\text{ p ld p} + (1-\text{p}) \text{ ld } (1-\text{p})$$



Die Funktion H(p) bezeichnet man als die Shannon-Funktion.

5.2 „EINER IST BESSER ALS KEINER“

- Der zu komprimierende Text wird komplett durchsucht
- Jeder vorkommende Buchstabe wird gezählt und bekommt ein Gewicht entsprechend seiner Häufigkeit. (Unterstrich _ steht für Leerzeichen)

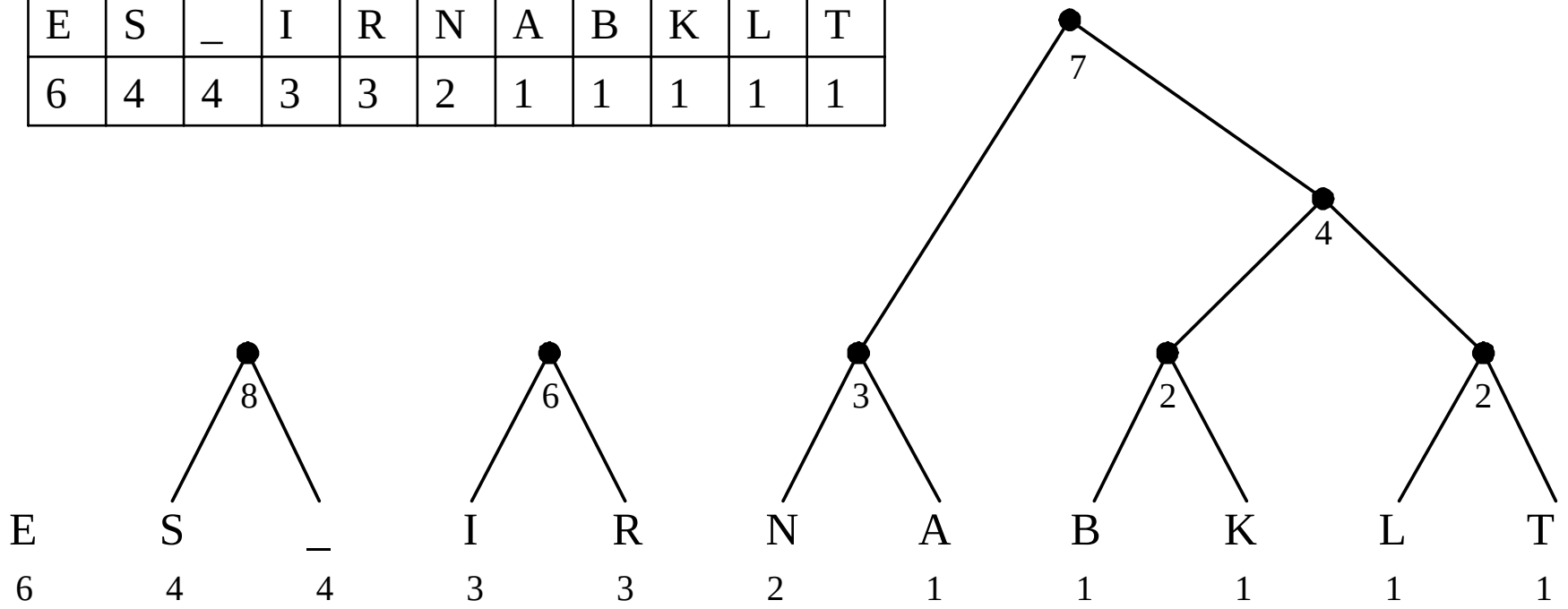
E	S	_	I	R	N	A	B	K	L	T
6	4	4	3	3	2	1	1	1	1	1

Es folgen jetzt noch weitere Schritte ...

5.2 Huffmann – 1. Schritt

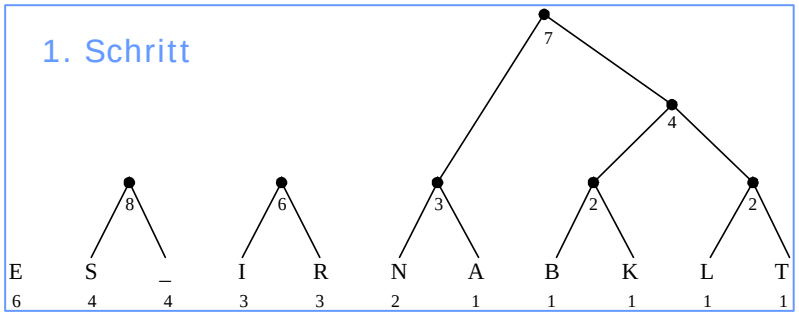
Es wird ein Baum erstellt. Jeweils **2 Knoten mit der geringsten Gewichtung** werden zu einem neuen Knoten zusammengefasst und erhält die **Summe der einzelnen Gewichtungen** der Knoten, aus dem er hervorgeht. Zusammengefasste Knoten werden **als verwendet markiert** und nicht weiter benutzt.

E	S	_	I	R	N	A	B	K	L	T
6	4	4	3	3	2	1	1	1	1	1



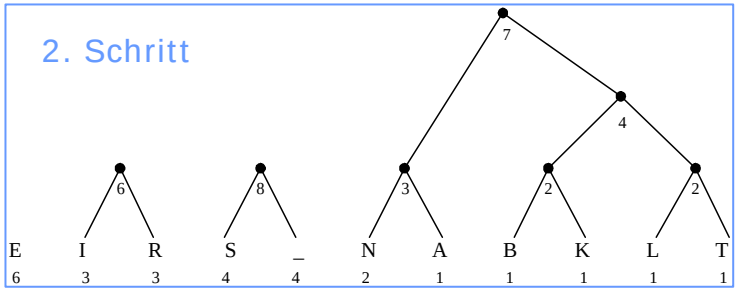
5.2 Huffmann – 2. Schritt

Als nächsten müssen die beiden Knoten mit der Gewichtung 6 zusammengefasst werden. Damit sich die Ästen in den darauf folgenden Schritten nicht überschneiden, wird (nur der Übersicht halber) umsortiert. Das „E“ wird mit dem Zweig rechts daneben vertauscht



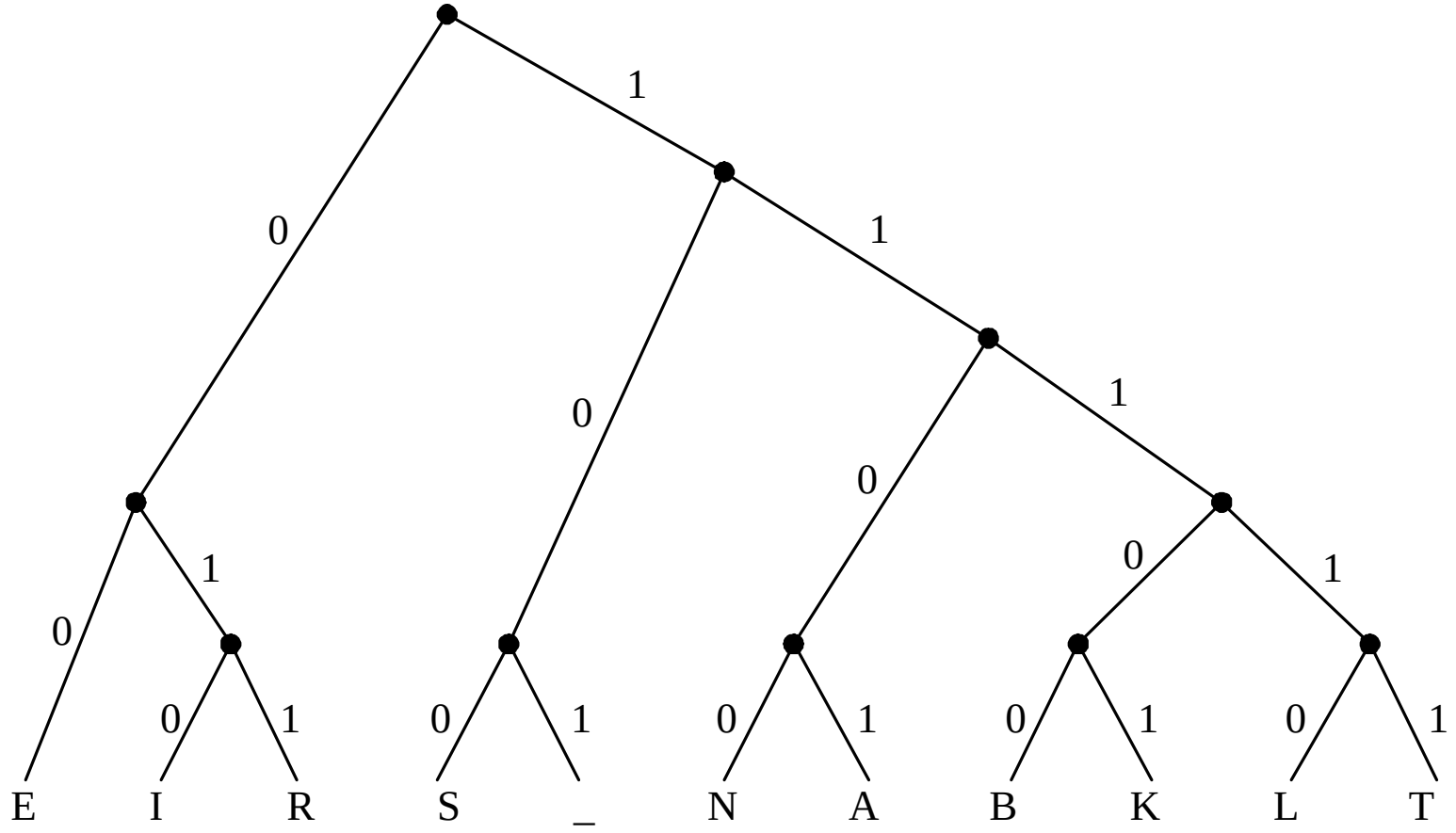
5.2 Huffmann – 3. Schritt

Fertigstellung des Baumes



5.2 Huffmann – 4. Schritt

Nun wählt man frei für je eine Richtung (links oder rechts) eines Astes des Baumes eine Binärzahl. Hier ist die Null für links und die Eins für rechts gewählt. Der fertige Baum sieht dann so aus:



5.2 Huffmann – Dekomprimierung

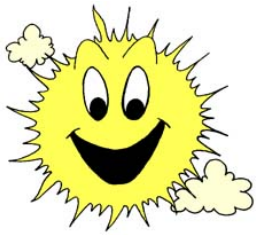
Beim Dekomprimieren muss der Baum bekannt sein. (Dieser wird in der Archivdatei mit abgespeichert.) Zur Rückgewinnung der ursprüngliche Daten wird für jedes Zeichen oben am Baum angefangen. Man durchläuft dann je nach Binärzahl solange die Verzweigungen, bis man unten angekommen ist. :

E	⇨	00	S	⇨	100
I	⇨	010	N	⇨	1100
R	⇨	011	K	⇨	11101

Komprimierung	**E**	**I**	**N**	**E**	**R**	**Sum**
ohne	01000101	01001001	01001110	01000101	01010010	40 Bit
mit	00	010	1100	00	011	14 Bit

Buchstaben, die häufig vorkommen, haben eine kürzere Sequenz. Binärzahlen müssen keine einheitliche Länge haben

Ende der Wiederholung



6. Rechnerarchitekturen

1. von-Neumann-Rechner
2. Harvard
3. CISC
4. RISC
5. Architekturen nach Befehls- und Datenstromstruktur

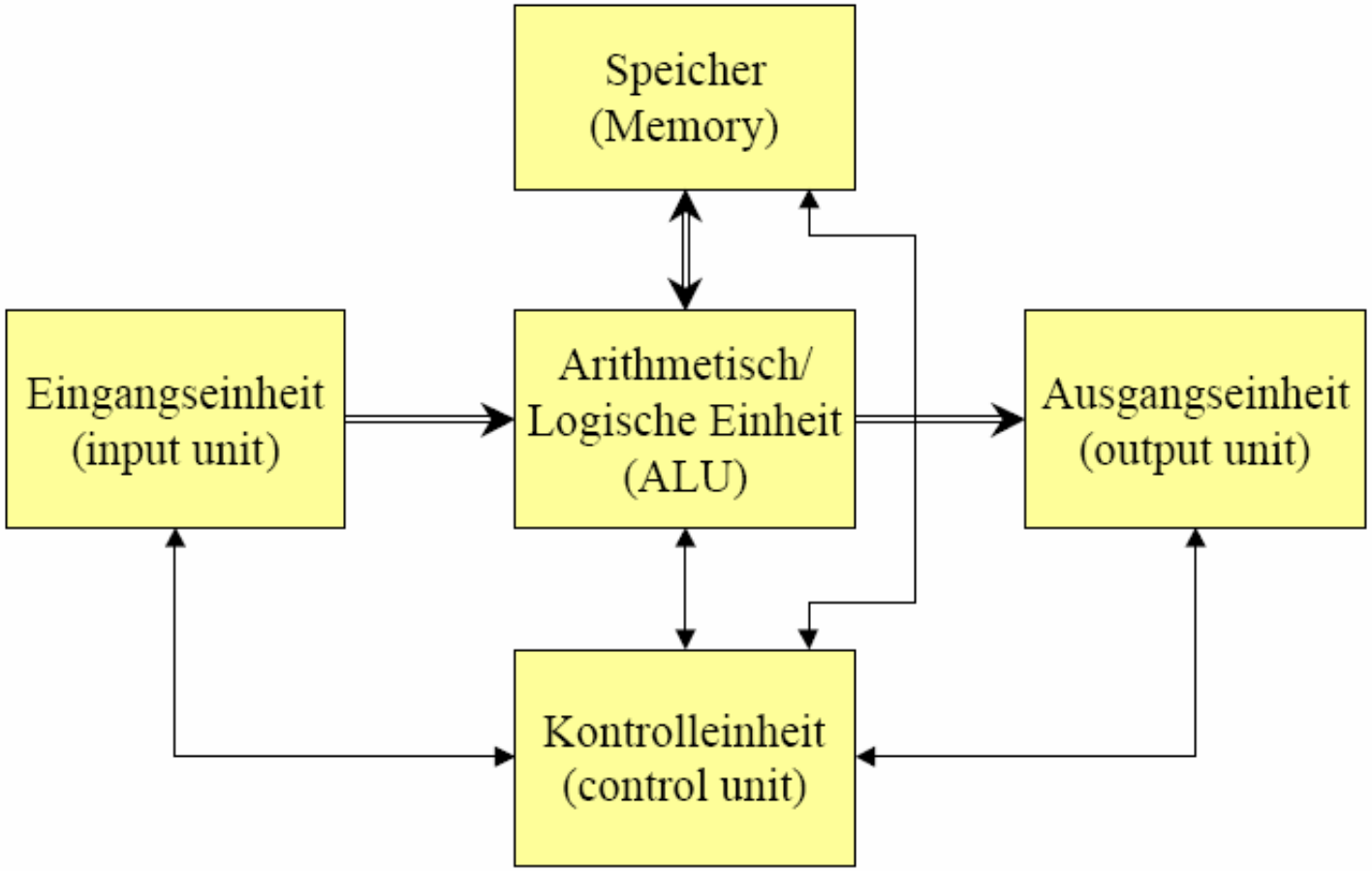
6. Rechnerarchitekturen

Eine Rechnerarchitektur beschreibt die Eigenschaften eines Systems, d.h. die Struktur des Entwurfs und das funktionelle Verhalten.

Die Organisation des Daten- und Kontrollflusses, der logische Entwurf und die physikalische Realisierung stellen die Umsetzung der Architektur dar.

Heutige konventionelle Rechner basieren auf der „von Neumann“-Struktur (Princeton). Es ist das am weitesten verbreitete Operationsprinzip von Rechnern.

6.1 von Neumann Architektur



Datenleitungen

Kontrollleitungen

6.1 von Neumann

Informatik

von Neumann-Rechner besteht aus 5 Funktionen:

1. Eingangseinheit

   oder Eingabewerk dient zur Aufnahme von Daten in den Rechner

2. Ausgabeeinheit

   oder Ausgabewerk dient der Ausgabe der vom Rechner ermittelten Ergebnisse

3. Speicher
4. Rechenwerk/ALU
5. Kontrolleinheit

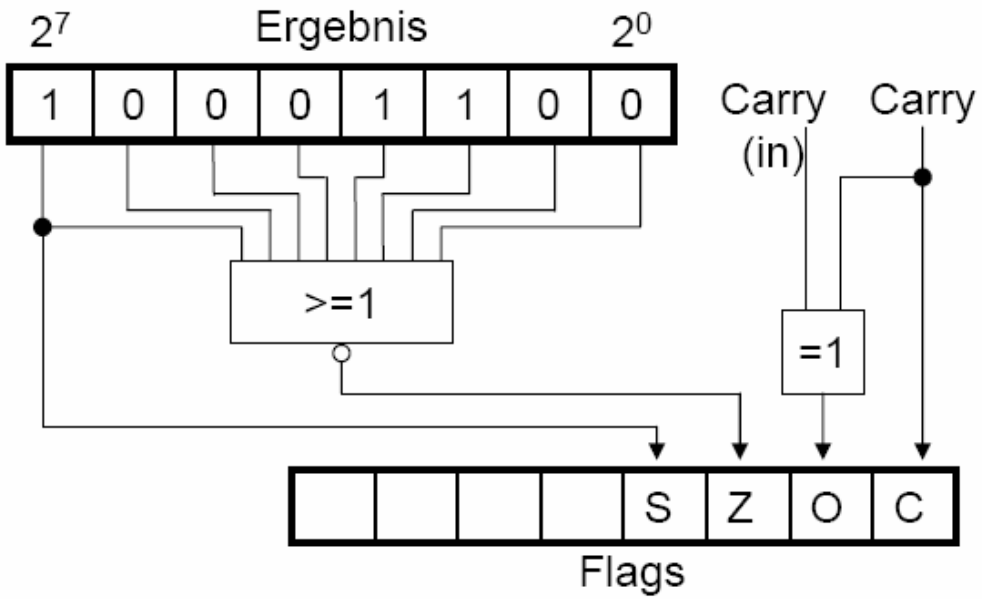
6.1.3 Speicher

- Im Speicher werden Daten abgelegt. Der Speicher hat eine Informationsstruktur, die auf „**Worten**“ basiert. Ein Wort ist eine Binärzahl fester Länge.
- Die Anzahl der Bits eines Speicherworts wird als **Speicherwortbreite** bezeichnet. Typische Wortbreiten sind 8, 16, 32 oder 64 Bit.
- Im Speicher wird keinerlei Unterscheidung zwischen verschiedenen Datentypen vorgenommen. So werden z.B. **Instruktionen** und **Programmdaten** in der gleichen Informationsstruktur (Wort) des Speichers abgelegt.

6.1.4 Rechenwerk / ALU

- Diese Einheit führt alle Berechnungen des Programms durch. Dazu beinhaltet diese Einheit arithmetische und logische Verarbeitungseinheiten zur Durchführung von arithmetischen und logischen Operationen sowie von Schiebeoperationen.
  (**Arithmetic Logical Unit**)
- Nach Durchführung einer Operation stehen sowohl das Ergebnis als auch sogenannte Bedingungs-Bits (Flags) zur Verfügung.
- Die Bedingungs-Bits beschreiben das Ergebnis der Operation (z.B. Null, Negativ, Überlauf, Carry).

6.1.4 Rechenwerk / ALU



(Statusregister, Processor Status Word PSW)

C: Carry-Flag	Übertrag, abhängig von der arithmetischen Operation
O: Overflow-Flag	Wird gesetzt, wenn ein Überlauf des Zahlenbereichs in Zweierkomplementdarstellung erfolgt
Z: Zero-Flag	Wird gesetzt, wenn der Wert im Ergebnis 0 ist.
S: Sign-Flag	(Negativ-Flag, Vorzeichen-Flag) ist eine Kopie des Vorzeichenbits des Ergebnisses

6.1.4 Rechenwerk / ALU

- ALU verarbeitet ganze Datenworte in einem Schritt.
- Ist Schaltnetz, hat nur Gatter, keine Speicher (außer dem Ergebnisregister)
- ALU ist kaskadierbar,
  z.B. 2x4-bit-ALU → 8-bit-ALU (ca. 200 Gatter)
- Rechenoperationen zurückführbar auf
  - Addition
  - Komplementbildung
  - Schieben (Shift)
- Logische Operationen zurückführbar auf
  - AND, OR, NOT
- Informationen, die für Nachfolgeoperationen wichtig sein können wie z.B. Flags

6.1.4 ALU für einen Beispielrechner

Informatik

Der folgend beschriebene Beispielrechner benötigt zur Realisierung von Operationen eine ALU für elementare Rechenoperationen und Adressberechnungen. Rechenoperationen sollen mit Wortbreite 16 bit, Adressberechnungen mit 24 bit durchgeführt werden. ALU muss also beide Wortbreiten unterstützen.

(adressierbarer Speicherbereich: 2^{24} = 16.7 Mbit)

6.1.4 ALU für einen Beispielrechner

Informatik

Die ALU des Beispielrechners soll folgende Funktionsgruppen beinhalten:

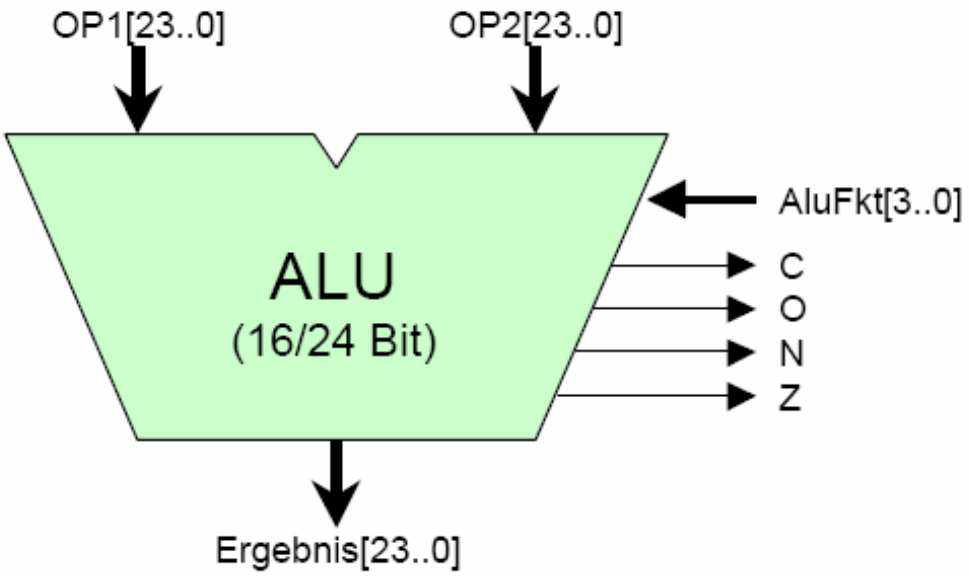
- 1. Erzeugung von Konstanten (16/24 Bit)
- 2. Logikfunktionen (16 Bit)
- 3. Arithmetische Operationen (16/24 Bit)
- 4. Schiebeoperationen (16 Bit)
- 5. Durchreichen von Operanden durch die ALU (16/24 Bit)

Die ALU verknüpft zwei Operandenworte OP1 und OP2 zu einem Ergebniswort ‚Ergebnis'.

Symbol der ALU → nächste Folie.

6.1.4 ALU für einen Beispielrechner

Bei Ausführung der ALU-Operationen werden Bedingungsbits gesetzt, welche Informationen über die Berechnung und über das Ergebnis liefern. Diese sind:



C (Carry): Überlauf bei der Addition positiver Zahlen (nur arithmetische Operationen).

O (Overflow): Überlauf bei der 2er-Komplement-Addition (nur arithmetische Operationen).

Z (Zero): ALU-Ergebnis ist Null.

N (Negative): ALU-Ergebnis ist negativ.

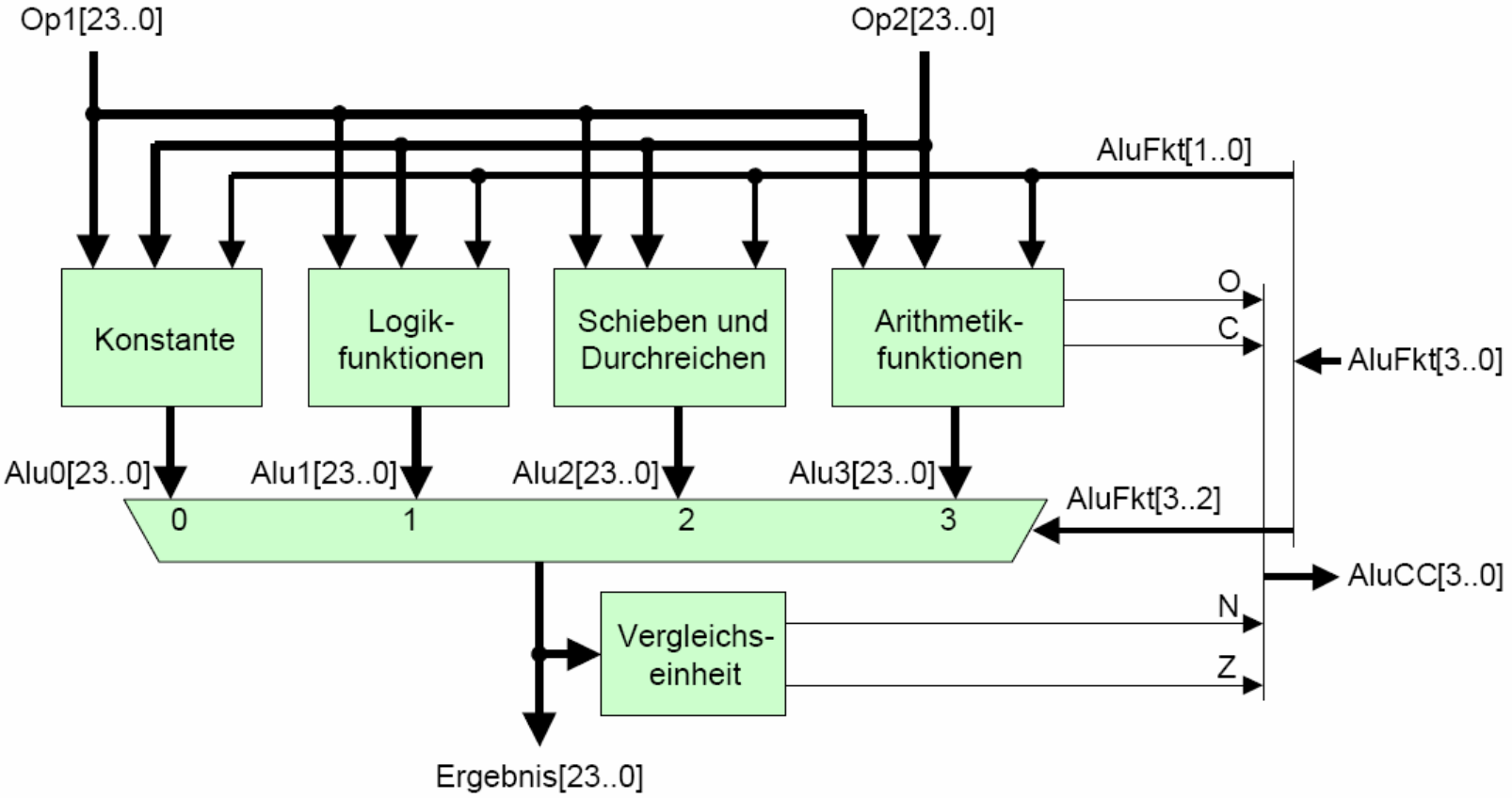
Die Bedingungsbits werden aus den unteren 16 Bit der Operatoren generiert, da nur bei arithmeti-schen und logischen Operationen von Interesse. ist.

Bei Adressberechnungen nicht nötig.

6.1.4 Steuerung der ALU

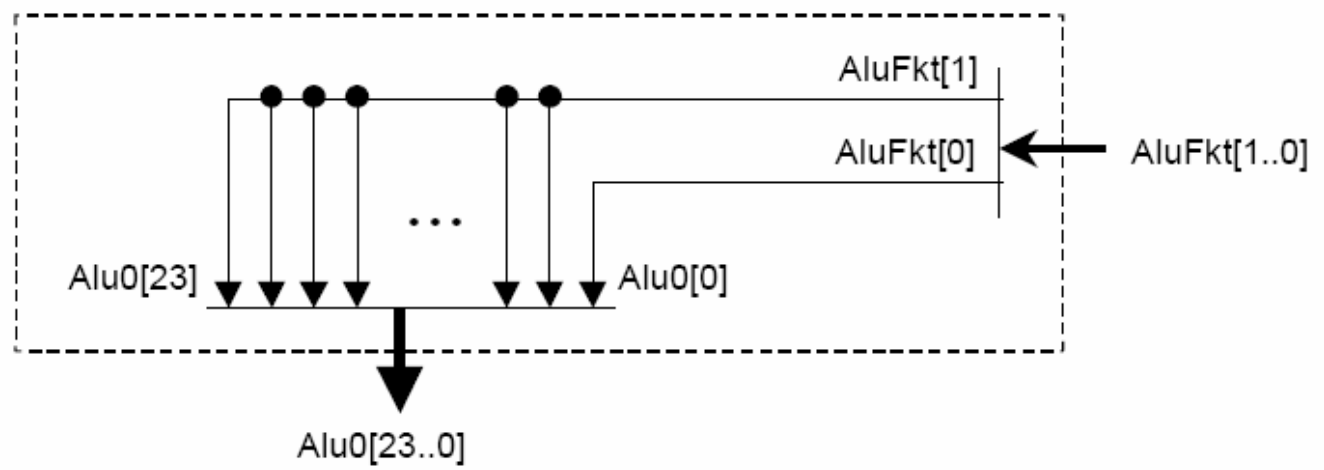
ALU Funktion	Kodierung	Einheit	Beschreibung
KONST_0	00 00	Konstante	Die Konstante 0 wird am Ausgang ausgegeben.
KONST_1	00 01		Die Konstante 1 wird am Ausgang ausgegeben.
KONST_M2	00 10		Die Konstante –2 wird am Ausgang ausgegeben.
KONST_M1	00 11		Die Konstante –1 wird am Ausgang ausgegeben.
NOT	01 00	Logik	Operand 2 wird bitweise negiert am Ausgang ausgegeben. Operand 1 wird ignoriert.
XOR	01 01		XOR-Verknüpfung von Operand 1 und Operand 2.
OR	01 10		OR-Verknüpfung von Operand 1 und Operand 2.
AND	01 11		AND-Verknüpfung von Operand 1 und Operand 2.
SHR	10 00	Schieben und Durchreichen	Operand 2 um ein Bit nach rechts schieben.
SHL	10 01		Operand 2 um ein Bit nach links schieben.
OP2	10 10		Operand 2 durchreichen, Operand 1 wird ignoriert.
OP1	10 11		Operand 1 durchreichen, Operand 2 wird ignoriert.
INC2	11 00	Arithmetik	Operand 2 wird um 2 inkrementiert und das Ergebnis am Ausgang ausgegeben. Operand 1 wird ignoriert.
DEC2	11 01		Operand 2 wird um 2 dekrementiert und das Ergebnis am Ausgang ausgegeben. Operand 1 wird ignoriert.
ADD	11 10		Operand 1 und Operand 2 werden addiert.
SUB	11 11		Operand 1 wird von Operand 2 subtrahiert.

6.1.4 Steuerung der ALU



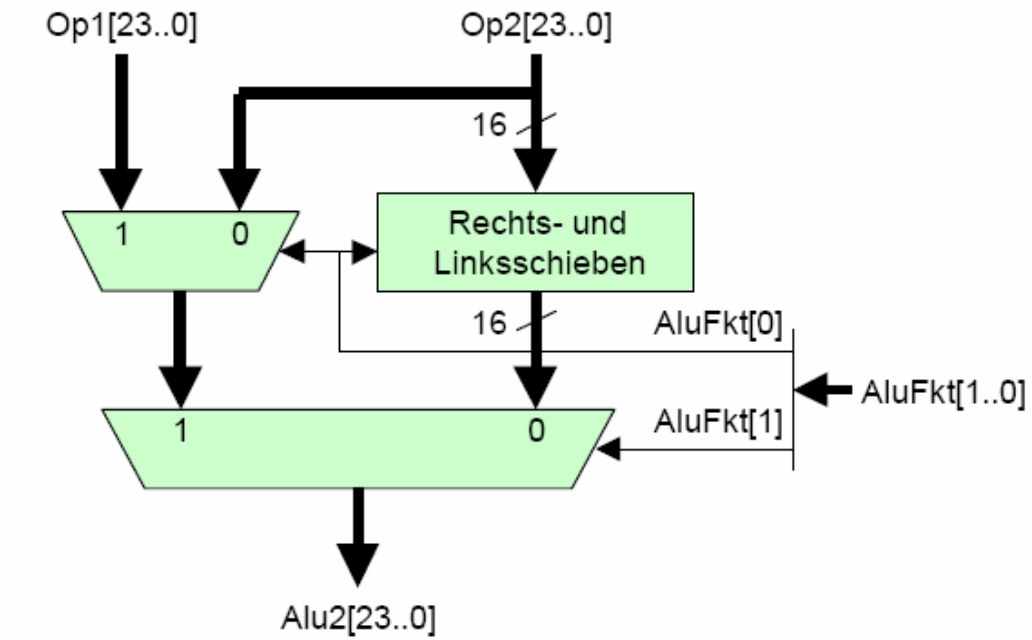
Die 4 Funktionsblöcke arbeiten parallel, durch einen nachgeschalteten Multiplexer wird das ALU-Ergebnis von der durch AluFkt[3..2] selektierten Funktionseinheit ausgewählt.

6.1.4.1 Generierung von Konstanten



Die Generierung von Konstanten erfolgt in der ALU des Beispielrechners mit einer festen Verschaltung der Funktionseingänge auf den Ausgang der ALU.

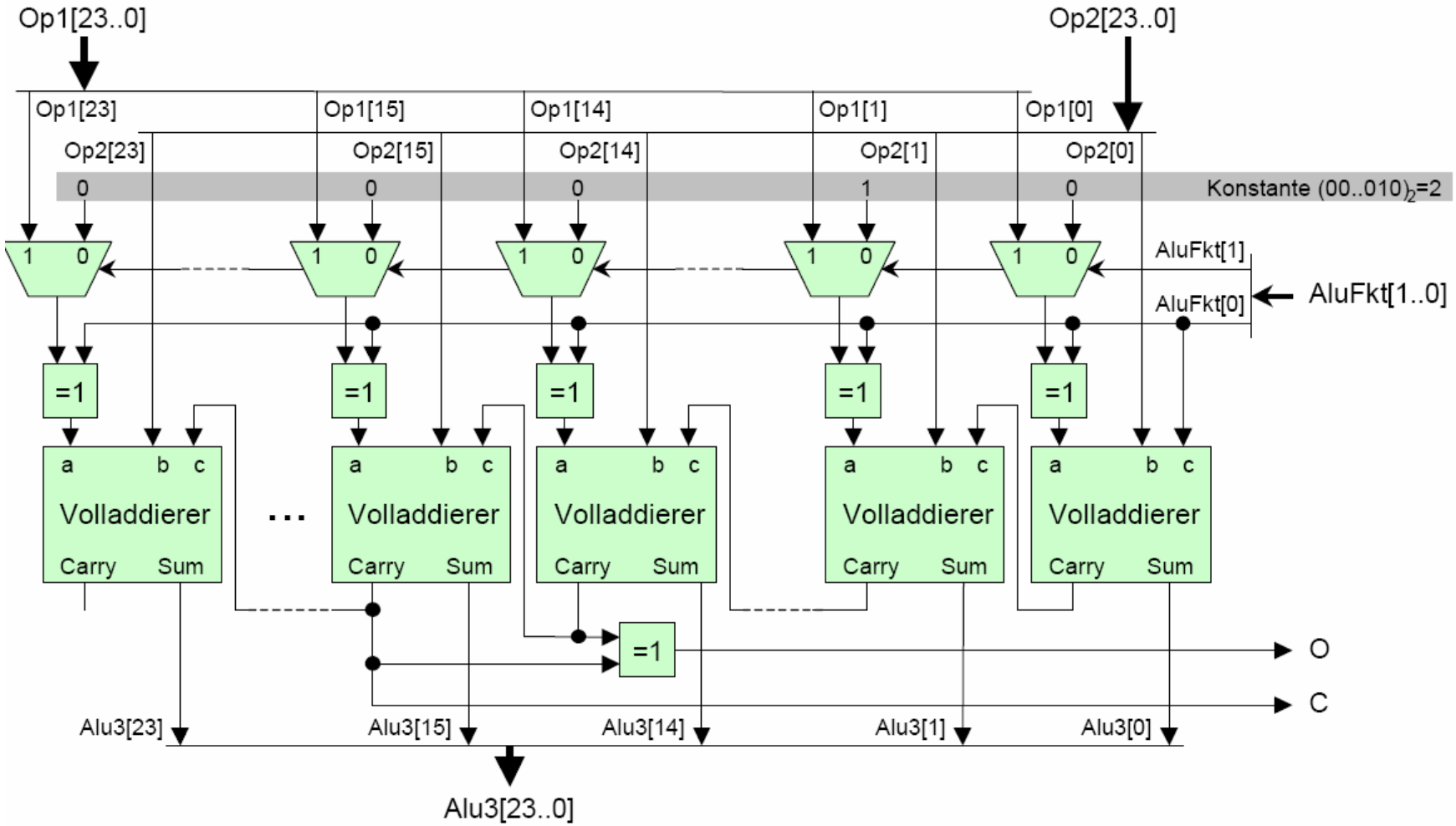
6.1.4.3 Schieben und Durchreichen



Im ALU-Block „Schieben und Durchreichen“ sind zwei Funktionalitäten kombiniert.

Zum einen kann in diesem Block der Operand 2 nach rechts und nach links geschoben werden. Dies belegt zwei Funktionen der vier Funktionen des Blocks. Zusätzlich ist in diesem Block ein Durchreichen der Operanden 1 und 2 durch die ALU realisiert.

6.1.4.4 Arithmetikfunktionen



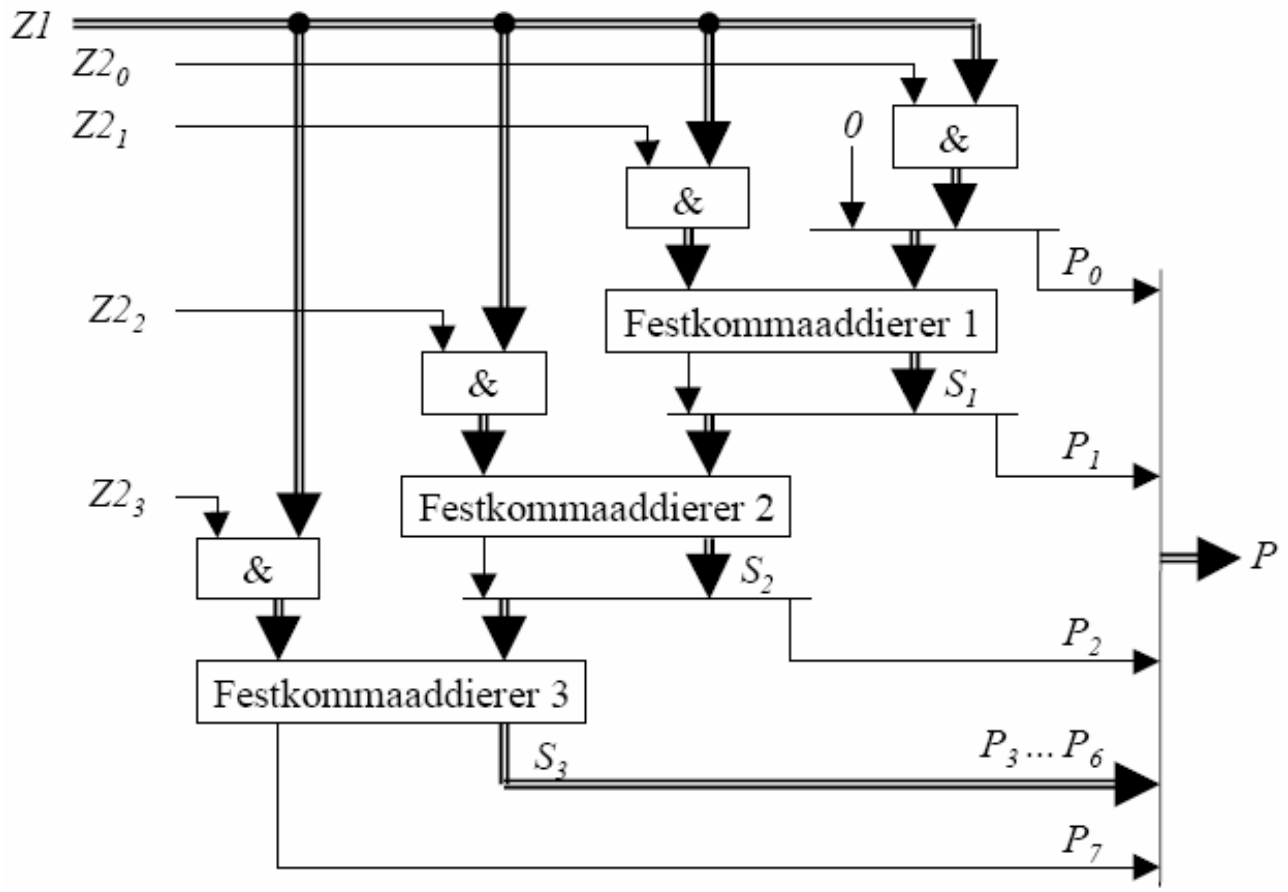
Als elementare Arithmetikoperationen werden die Addition und die Subtraktion unterstützt. Carry- und Überlaufbit (C, O) werden dabei aus den unteren 16 Bit bestimmt.

6.1.4.4 Arithmetikfunktionen

Zur Adressberechnung kann die ALU die Konstante 2 zum Operanden 2 addieren oder von diesem subtrahieren, was zur sequentiellen Veränderung von Adressen im Beispielrechner benötigt wird.

Vor dem a-Eingang ist ein Multiplexer vorgeschaltet, der entweder den Operanden 1 (Summand oder Subtrahend) oder den konstanten Wert 2 selektiert. Damit kann, wie in der Funktionstabelle der ALU angeführt, entweder Operand 1 oder der konstante Wert *2* von Operand 2 addiert bzw. subtrahiert werden.

6.1.4.4 Wiederholung Multiplizierer



Multiplizierer für 4 bit.

6.1.5 Kontrolleinheit / Leit- / Steuerwerk

Informatik

Eines der wichtigsten allgemeinen Entwurfsprinzipien ist die **Trennung** von **Kontrolle** und der **Verarbeitung von Daten**, dementsprechend beim Hardware-Entwurf die Trennung von Steuerwerk und Operationswerk:

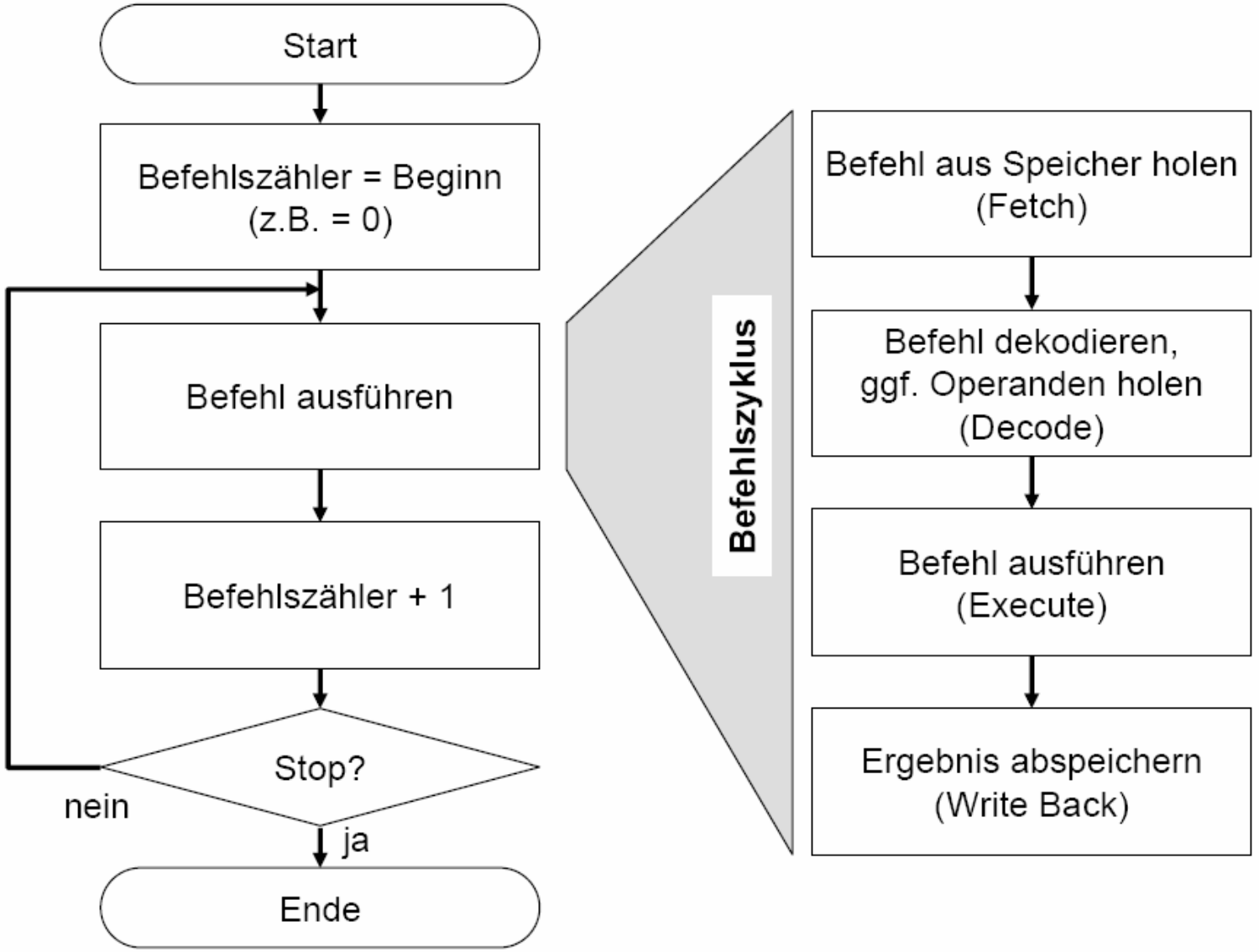
Die Kontrolleinheit (Steuerwerk, Leitwerk) interpretiert die im Speicher abgelegten Worte. Je nach Interpretation sind dies:

- Instruktionen (Befehle),
- Programmdaten oder
- Adressen

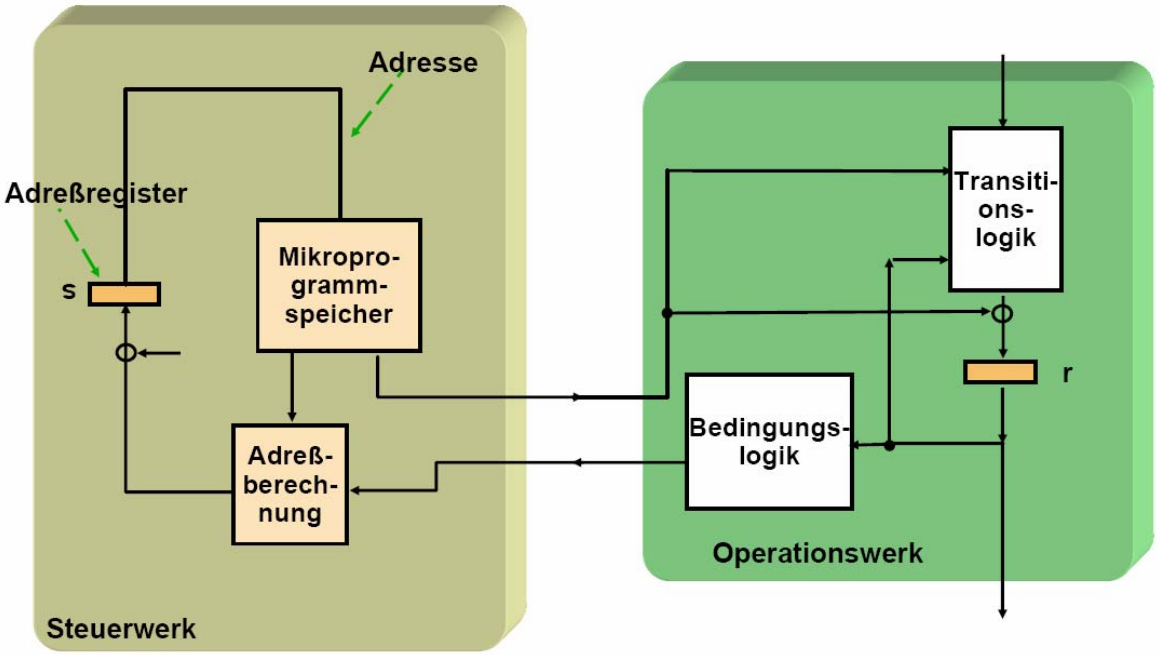
6.1.5 Kontrolleinheit / Leit- / Steuerwerk

- Aufgrund der Instruktionen werden die übrigen Einheiten gesteuert, so daß die Instruktionen korrekt ausgeführt werden. In der Kontrolleinheit wird der Zustand des Rechners gehalten, der die Speicherposition der nächsten Instruktion bestimmt.
- Ein Register in der Kontrolleinheit zeigt auf die nächste auszuführende Instruktion → **Programmzähler** (program counter, **PC**). Nach Ausführung einer Instruktion wird der Programmzähler erhöht und zeigt direkt auf das der aktuellen Instruktion nachfolgende Speicherwort.
- Sprungbefehle verändern den Programmzähler und unterbrechen damit die sequentielle Instruktionssequenz.
- Adressberechnungen werden mit einer endlichen Anzahl von Bits ausgeführt. Diese Anzahl wird als **Adresswortbreite** bezeichnet.

6.1.5 Modell eines Befehlsablaufs



6.1.5 Mikroprogrammierte Steuerwerke



Mikroprogrammierung ist eine weitverbreitete Technik der Steuerwerkrealisierung, z.B. in CISC-Rechnern. Die Speicherung der Steuersignale erfolgt in einem (i.a. read-only) Mikroprogrammspeicher (µROM).

Vorteile der Mikroprogrammierung: hohe Flexibilität während des Entwurfsprozesses; Nachbildung von Konstrukten höherer Programmiersprachen