Computerarchitektur

Chris Weber, Kantonsschule Limmattal

EF Informatik 2025/26

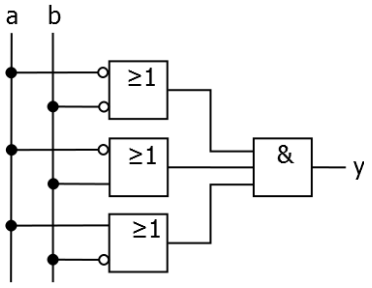
1 Logische Gatter (*logic gates*)

Übersicht über die logischen Gatter sowie Halb-/ Volladdierer: s. Helm/ Gantert, Praktikum Digitalelektronik. Wichtig sind die Gatter NOT, AND, OR, XOR und NAND (jeweils nur die eckigen IEC-Symbole, die erste Zeile der Spalte „Funktion"´ sowie die Wahrheits-/ Schalttabelle).

Ebenfalls nützlich sind DeMorgan'schen Regeln:

$$\overline{a \wedge b} = \overline{a} \vee \overline{b} \qquad\qquad \overline{a \vee b} = \overline{a} \wedge \overline{b}$$

1. a) Stelle einen Schaltplan für den folgenden Ausdruck auf: $y = (a \vee \overline{a}) \wedge (\overline{\overline{b}} \vee b)$
   b) Vereinfache den Ausdruck.
2. Stelle die Schalttabelle für einen Volladdierer auf.
3. a) Ermittle für die nebenstehende Schaltung die zugehörige Schaltgleichung.
   b) Vereinfache diese anschliessend weitestmöglich.
   c) Zusatz: Versuche die vereinfachte Schaltgleichung so umzuformen, dass nur eine Sorte Logikgatter nötig ist.



4. a) Ermittle aus der rechts stehenden Schalttabelle eine Schaltgleichung. Falls sie mehr als zwei Operationen enthält, vereinfache diese Schaltgleichung rechnerisch mithilfe der Schaltalgebra.
   b) Gib zu der vereinfachten Schaltgleichung eine Schaltung an.

a	b	c	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

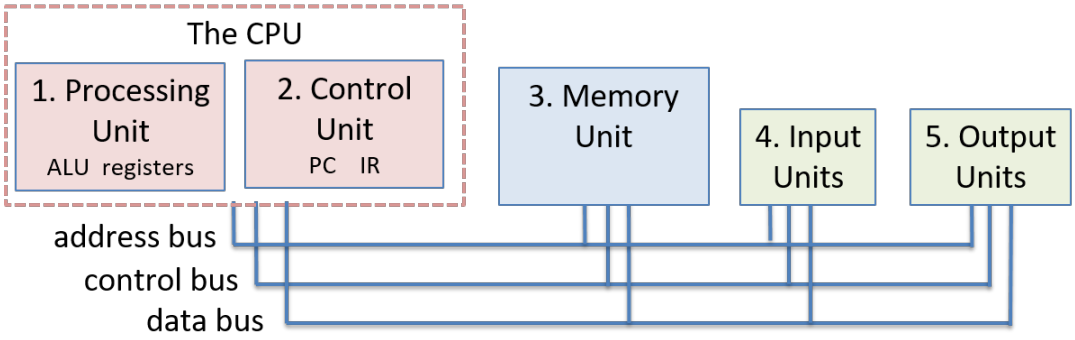
5. Drücke nur mit NAND-Operationen ($\overline{\wedge}$) aus:

   a) $y = \text{NOT } a$ b) $y = a \text{ AND } b$ c) $y = a \text{ OR } b$!d) $y = a \text{ XOR } b$

(Quellen: https://de.serlo.org/informatik/109346/09.-schaltnetze-%C3%BCbungen-zum-entwickeln-und-vereinfachen) sowie https://en.wikipedia.org/wiki/NAND_logic)

2 von Neumann-Architektur

Ein Computer gemäss der von-Neumann-Architektur ist wie im folgenden Schema aufgebaut. Dabei sind:
CPU (central processing unit)
ALU (arithmetic-logical unit:) führt logische und arithmetische Operationen aus.
Register: kurzfristige Speichermöglichkeit in der CPU.
IR (instruction register): Register, das den Befehl enthält, der gerade ausgeführt wird.
PC (program counter): Register, das die Adresse des nächsten Befehls enthält.
Busse: Kommunikationskanäle zwischen den Komponenten

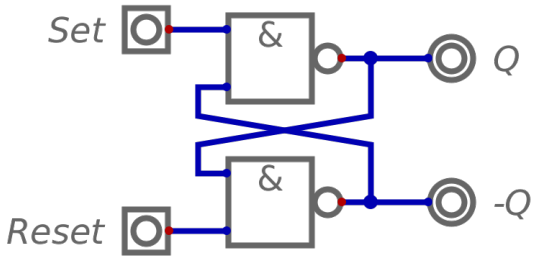


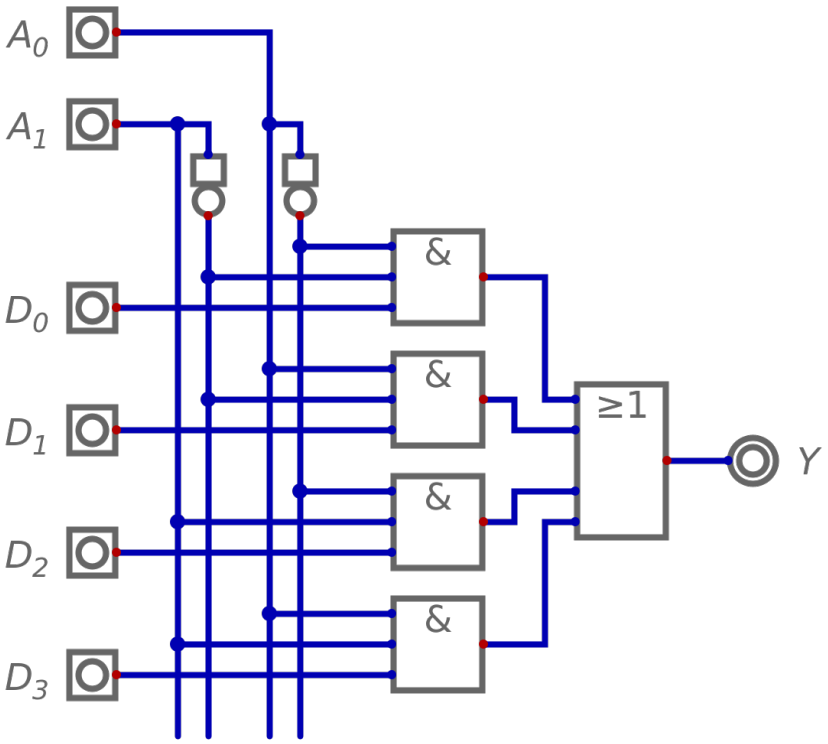
Ein von-Neumann-Computer arbeitet Befehle nach dem folgenden **Zyklus** ab:

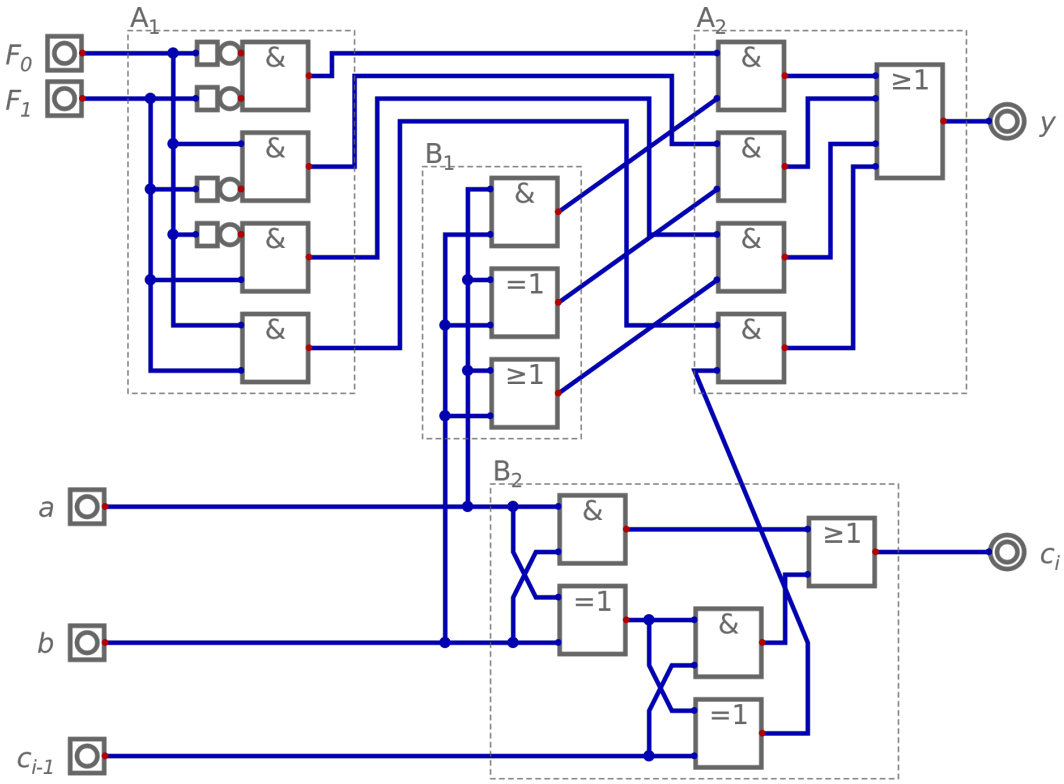
1. **fetch:** Die Kontrolleinheit holt den nächsten auszuführenden Befehl aus dem Speicher.
2. **decode:** Die Kontrolleinheit entschlüsselt den Befehl, gibt ihn an die ALU weiter und lädt allfällige Argumente aus dem Speicher in die Register der Processing Unit.
3. **execute:** Die Processing Unit führt den Befehl aus.
4. **store:** Die Control Unit speichert das Resultat im Speicher.

Mehr Infos: https://diveintosystems.org/book/C5-Arch/von.html

3 Einige Komponenten eines Rechners







Zum Ausprobieren und Rumspielen: https://github.com/hneemann/Digital