

# MPMP-01 Einführung Offline Version während der COVID-19 Pandemie

## Modul C004 [INB8034]

- Hardwarebeschreibungssprachen für kombinatorische und sequenzielle Systeme
- Automaten, Mikroprogrammierung und Mikroprogrammsteuerwerke
- Mikroprogrammsteuerwerk und Hardwaresteuerwerk im Vergleich:
  - Verschiedene Automatentypen
  - Minimierung des Aufwandes für den Mikroprogrammspeicher
  - Ein mikroprogrammierbarer Rechner
- Mikroprozessoren und Mikrorechner:
  - Zeitverhalten, Adressierungsarten, Befehlsausführung, Interruptsystem, Periphere
  - Systembauelemente

## Modul C004 [INB8034]

- Die Studentinnen und Studenten sind in der Lage, die verschiedenen Architekturprinzipien mikroelektronischer Systeme zu charakterisieren und typische Anwendungen mit den hierfür geeigneten Hard- und SoftwareWerkzeugen zu implementieren.
- Die StudentenX beherrschen verschiedene Kontrollstrukturen von den Zustandsfolgen endlicher Automaten bis zum Timesharing in Interruptsystemen.
- Sie können damit Aufgabenstellungen in verteilten und zeitlich parallelen Anwendungen implementieren. Insbesondere sind die Voraussetzungen geschaffen, sich mit Kernel- und Treiberprogrammierung auseinanderzusetzen.

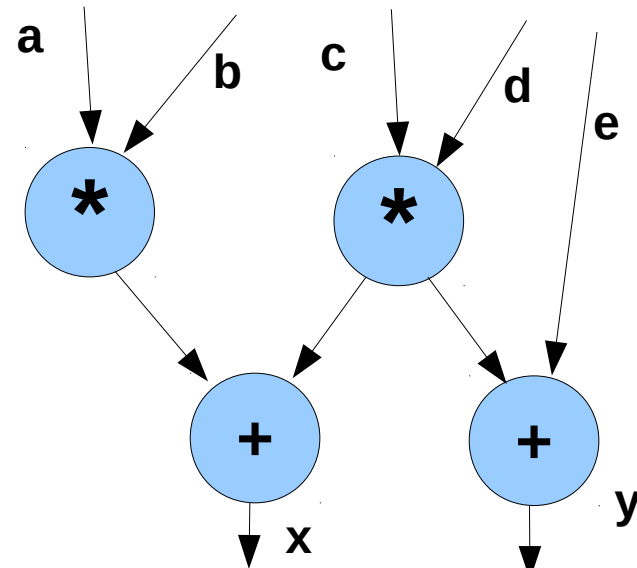
- Sichten auf eine Informationsverarbeitungseinheit:
  1. Physikalische Schicht
  2. Transistorebene
  3. Gatterebene
  4. **Mikrosystemebene**
  5. **Registertransferebene**
  6. Hochsprachenebene
  7. Höhere Abstraktionsebenen
- Die Technische Informatik beschäftigt sich hauptsächlich mit den Ebenen 3, 4 und 5
- Im zweiten Teil der Vorlesungsreihe wollen wir uns von der Gatterebene lösen und in Richtung Hochsprachenebene bewegen

- Datenflussmaschine
  - Abarbeiten von Operationen, ähnlich funktionaler Sprachen entlang der Kanten eines Datenflussgraphen (DFG)
  - Hardware extrem aufwändig
  - Parallelität einfach zu realisieren
  - Interaktivität kompliziert
  - Gesamtauslastung niedrig
- Prozessor (CPU)
  - Abarbeiten einer Anweisungsliste entlang der Kanten eines Kontrollflussgraphen (CFG)
  - Hardware ist einfach
  - Ohne Erweiterungen wird immer nur eine Instruktion gleichzeitig ausgeführt
  - Gesamtauslastung sehr hoch

Prozessoren dominieren den Markt für elektronisches Rechnen,  
Datenflussmaschinen sind nur in Nischen zu finden

- Datenflussgraph (DFG) wird durch Verdrahtung von Funktionseinheiten realisiert
- $DFG = (O, V)$   
Der DFG wird definiert durch die Menge der Operationen (O) und die Datenflussskanten, die als Variablen (V) diese Operationen verbinden
- Variablen werden im mathematischen Sinne definiert, sie verkörpern eine Verdrahtung, keine Speicherzelle

Beispiel eines Datenflussgraphen

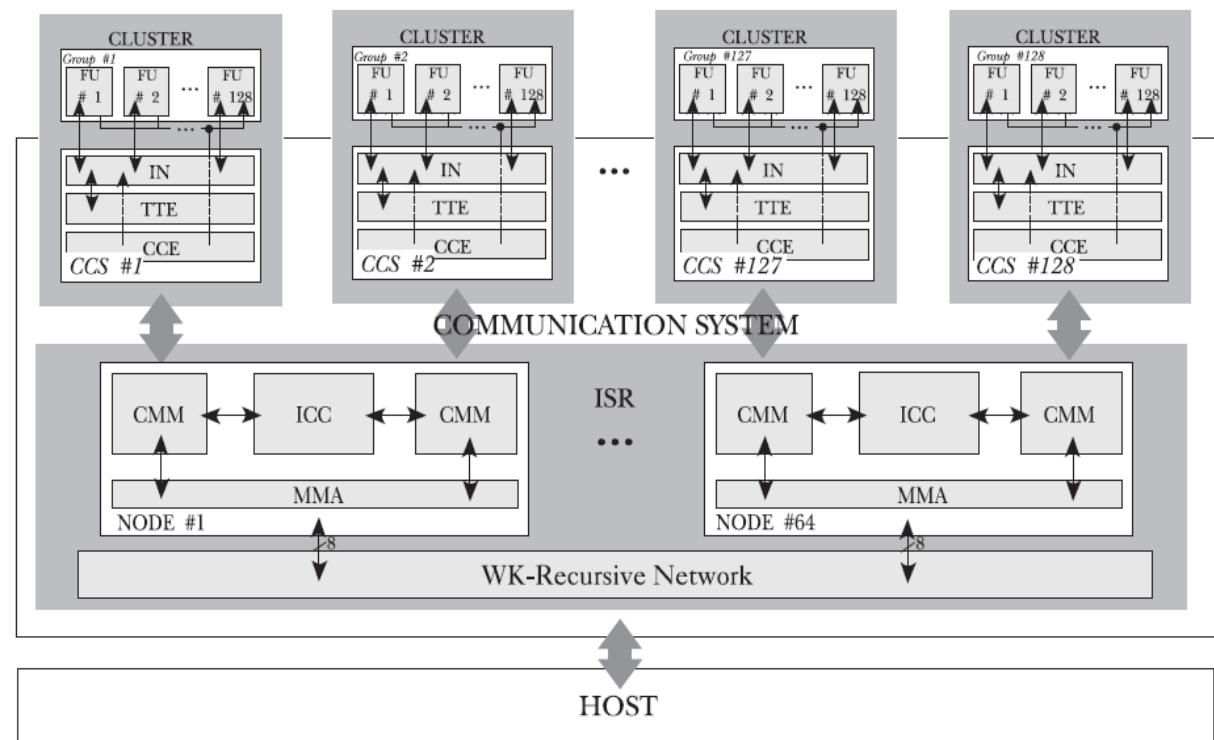


$$x = ab + cd \quad y = cd + e$$

- Bereitstellen von Eingaben und Abholen von Ergebnissen über Nachrichtensysteme

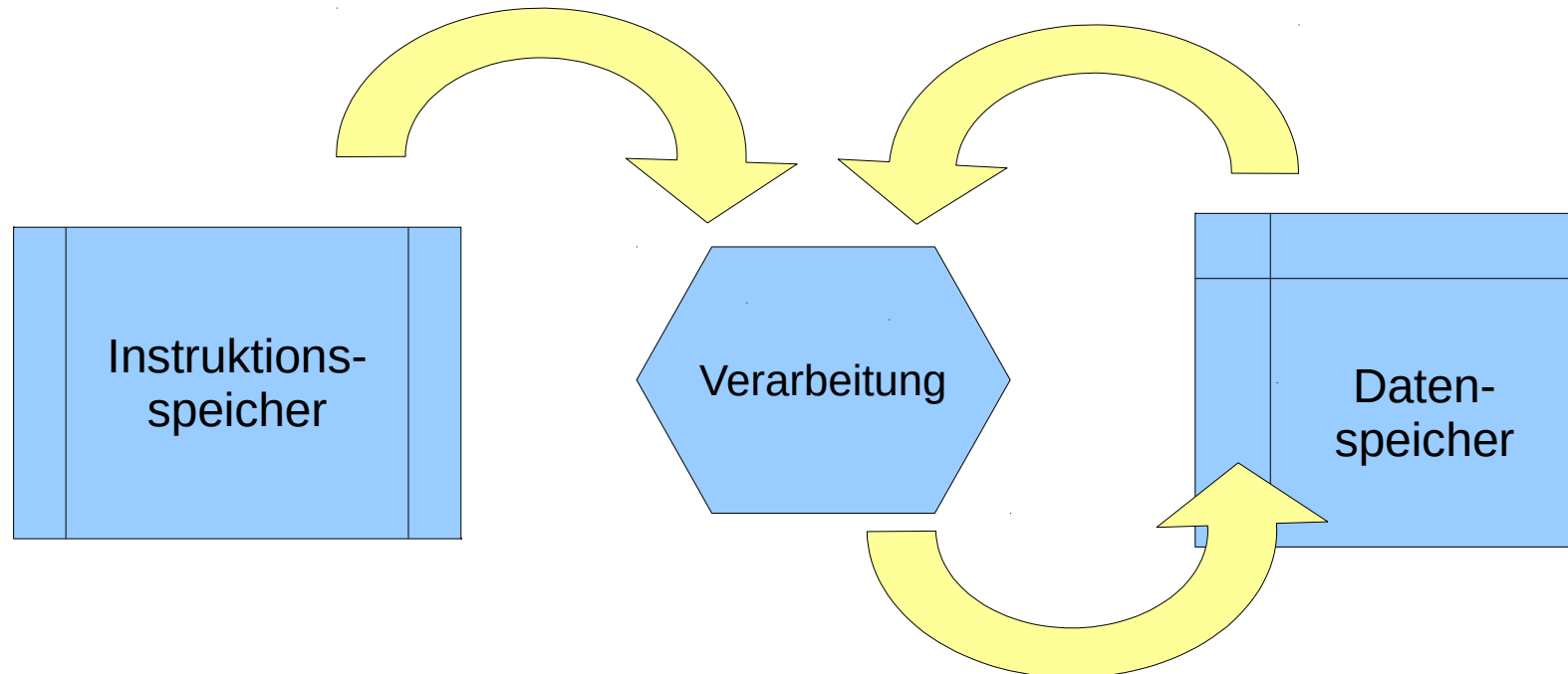
# Beispielimplementierung

- Lorenzo Verdoscia: „ALFA fine grain dataflow machine“, in International Programming I, Ed. by M.A. Orgun, pp. 110-134, 1996, <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.752>
- 128 Cluster mit je 128 Verarbeitungseinheiten
- Einheitliches I/O
- Routersystem (ISR)
- Host-Rechner zur Interaktion
- Paper beschreibt Funktion, leider keine Benchmarks



© L. Verdoscia, 2006

# Grundprinzip eines Prozessors



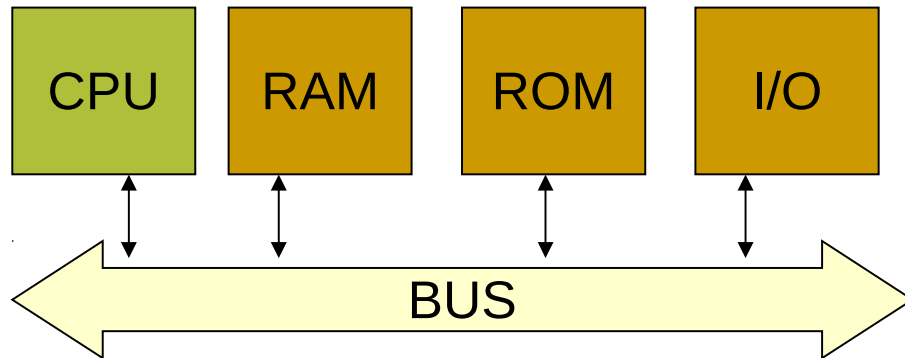
- Instruktionen werden sequentiell abgearbeitet
- Dabei werden Daten gelesen, manipuliert und zurückgeschrieben
- Nur ein Verarbeitungsschritt gleichzeitig



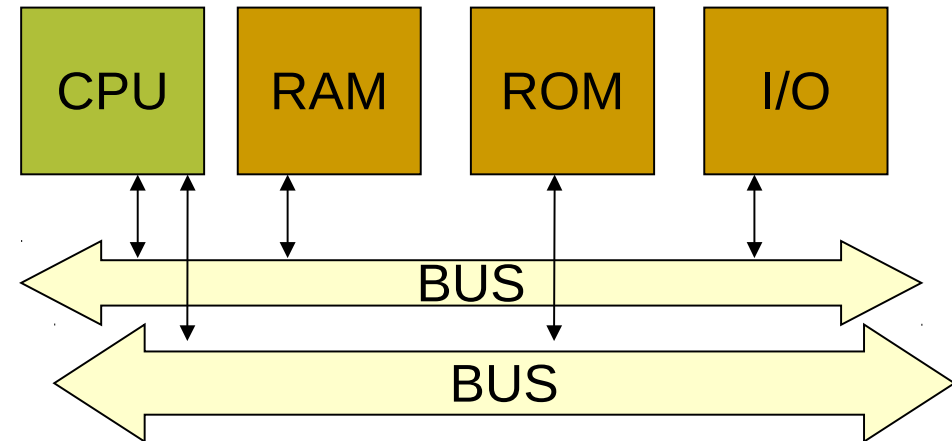
- Ursprüngliche Idee: CISC (Complex Instruction Set Computer)
  - Bestehende Aufgaben in Rechenschritte zerlegen
  - Jeder Teilschritt entspricht einem Maschinenbefehl
- Folge: Komplizierte Maschinenbefehle, unterschiedlich lange Ausführungszeiten (Addition, Division), unterschiedlich breite Kodierung, wenige spezialisierte Register
- Maschinenoperationen interagieren oftmals mit dem Hauptspeicher
- Maschinenbefehle werden in Mikrobefehle zerlegt, die dann direkt verdrahtet sind
- CPU wird kompliziert und groß, hoher Optimierungsbedarf
- CISC-Rechner haben oft von Neumann-Architektur

- Optimiert auf Geschwindigkeit:  
RISC (Reduced Instruction Set Computer)
  - Einfache, direkt verdrahtete Maschinenoperationen
  - Aufgabenlösung muss aus diesen zusammengesetzt werden
  - Höherer Bedarf an Programmspeicher, RAM, Caches
- Alle Befehle gleich breit kodiert mit gleicher Ausführungszeit
- Einfache, homogene CPU, viele Register
- Nur wenige Befehle greifen auf (langsamen) Hauptspeicher zu
- Kleine, schnelle, energieeffiziente CPU
- RISC Rechner benutzen oft Harvard-Architektur
- Viele Probleme werden in den Compiler verschoben

von-Neumann-Architektur



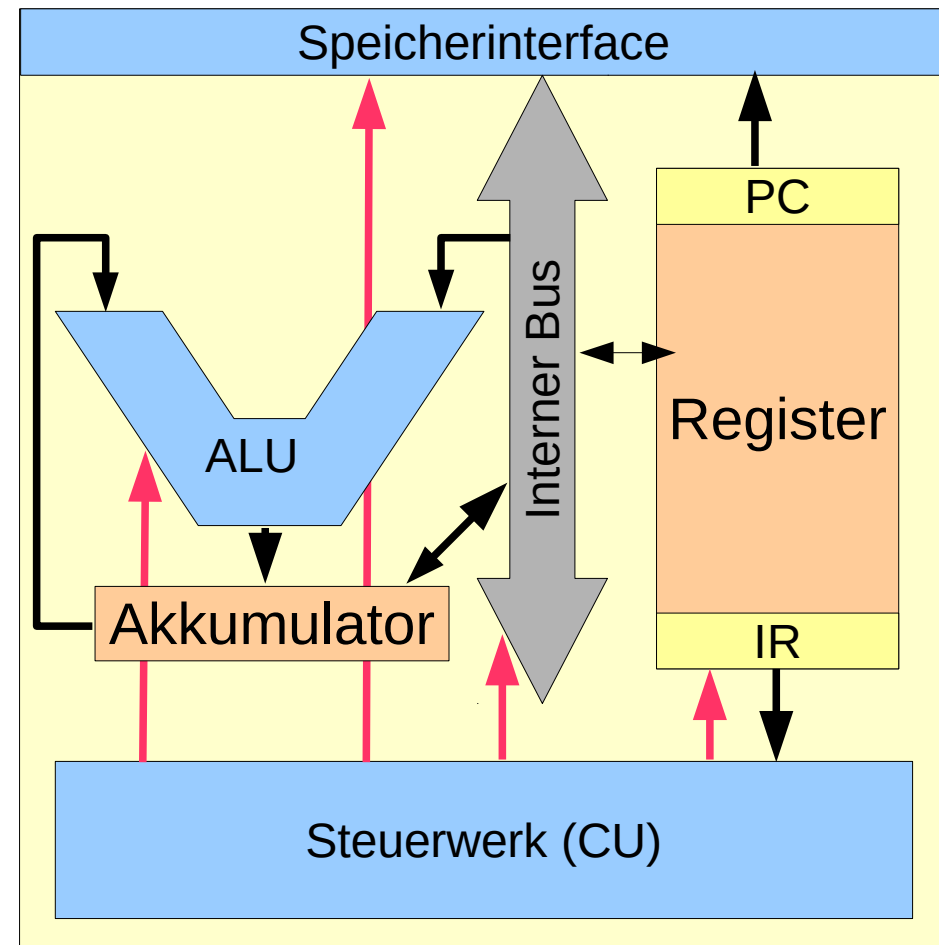
Harvard-Architektur



- Von-Neumann Architektur: Daten und Befehlscode liegen im gleichen Speicher und werden in aufeinander folgenden Buszyklen gelesen. (Johann von Neumann gibt diesem System zu unrecht seinen Namen, 8 Jahre zuvor von Konrad Zuse patentiert)  
Nachteile: **langsam, Abhängigkeiten, besonders Wortbreite**
- Harvard-Architektur: **Mehrere optimierte Bussysteme/Speicher** (benannt nach dem Harvard-Mark I Computer)  
Bei DSPs auch ein Bus je Operand, **sehr teuer, sehr schnell**

# Innere Sicht – CISC CPU

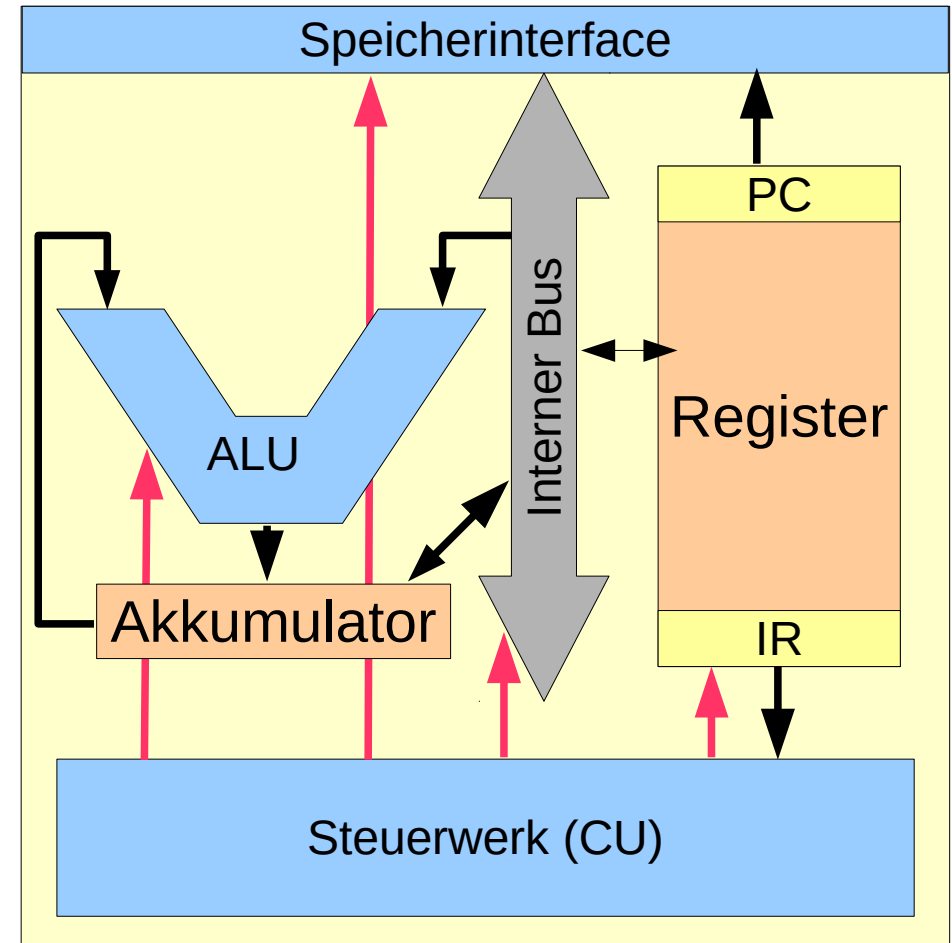
- Speicherinterface
- Arithmetik-Logik-Einheit (ALU), Operationen in Breite der Register
- Steuerwerk (CU), Zustands-maschine, führt Mikrocodes aus
- Register, schneller Speicher in der CPU
  - Programmzähler (PC), Adresse des nächsten Befehls
  - Instruktionsregister (IR), aktueller Befehl
  - Akkumulator (Accu), Ergebnis und erster Operand der ALU



Einfache Beispiel-CPU (CISC)

# Ablauf der Befehlsabarbeitung

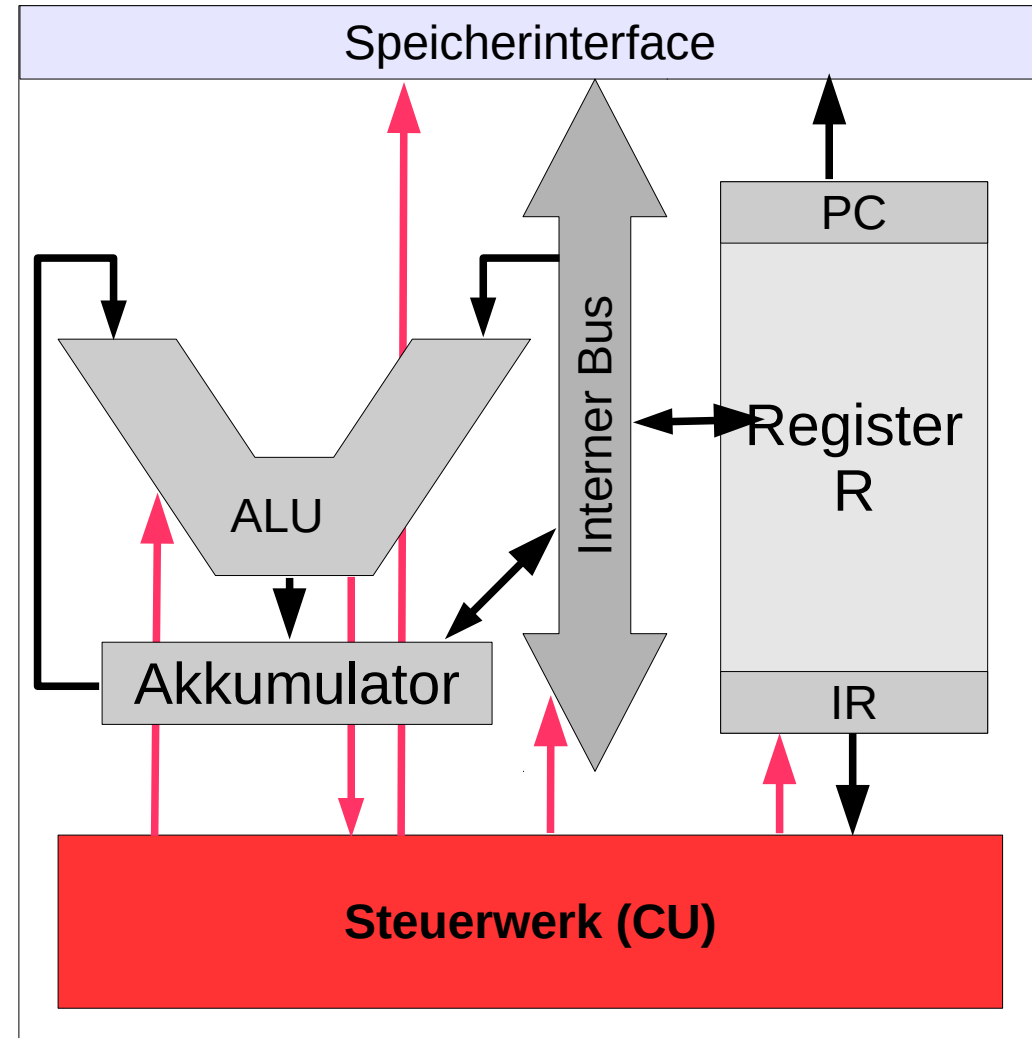
- CU legt PC auf externen Adressbus
- CU liest Speicher nach IR
- CU löst  $PC = PC + 1$  aus
- Befehl wird abgearbeitet:
  - Lesen Operanden
  - Berechnen Ergebnis (ALU)
  - Schreiben Operanden
- Abarbeitung nächster Befehl



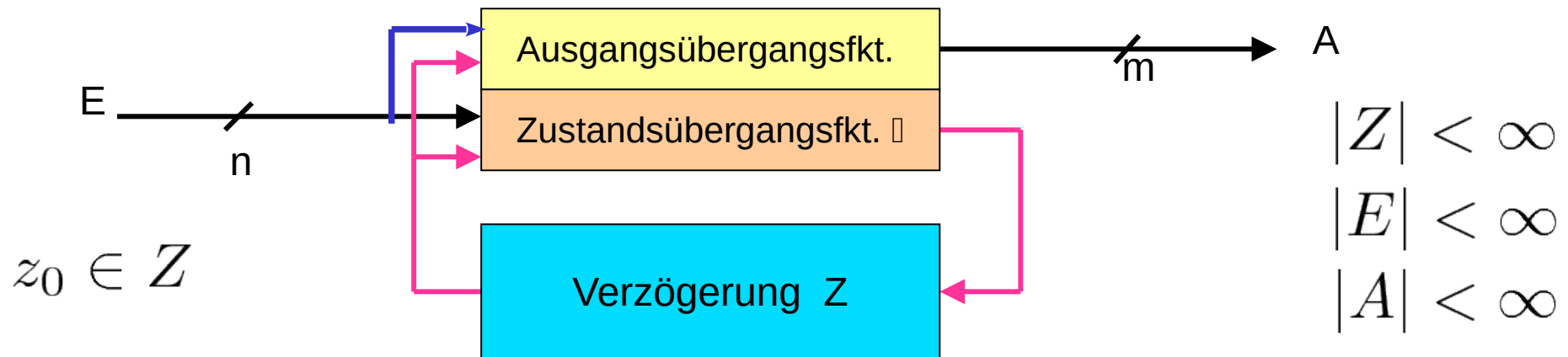
Einfache Beispiel-CPU (CISC)

# Das Steuerwerk

- Steuerwerke interpretieren den Opcode (Teil des Instruktionsregisters) und werten Statusinformationen aus (z.B. von der ALU)
- Kontrolle der anderen CPU-Komponenten
- Steuerwerke sind Automaten
  - RISC – fest verdrahtet
  - CISC – mikroprogrammiert
    - Oft auch Updates möglich
- Obwohl heute Mikroprogrammierung ganz anders eingesetzt werden wollen wir diese Anwendung aus didaktischer Sicht verfolgen



# Wiederholung: Automaten



Automaten sind endliche Zustandsmaschinen:  
Inneren Zuständen  $Z$ , Eingabealphabet  $E$ , Ausgabealphabet  $A$ :

$$EZM = (Z, E, A, \delta, \lambda, z_0)$$

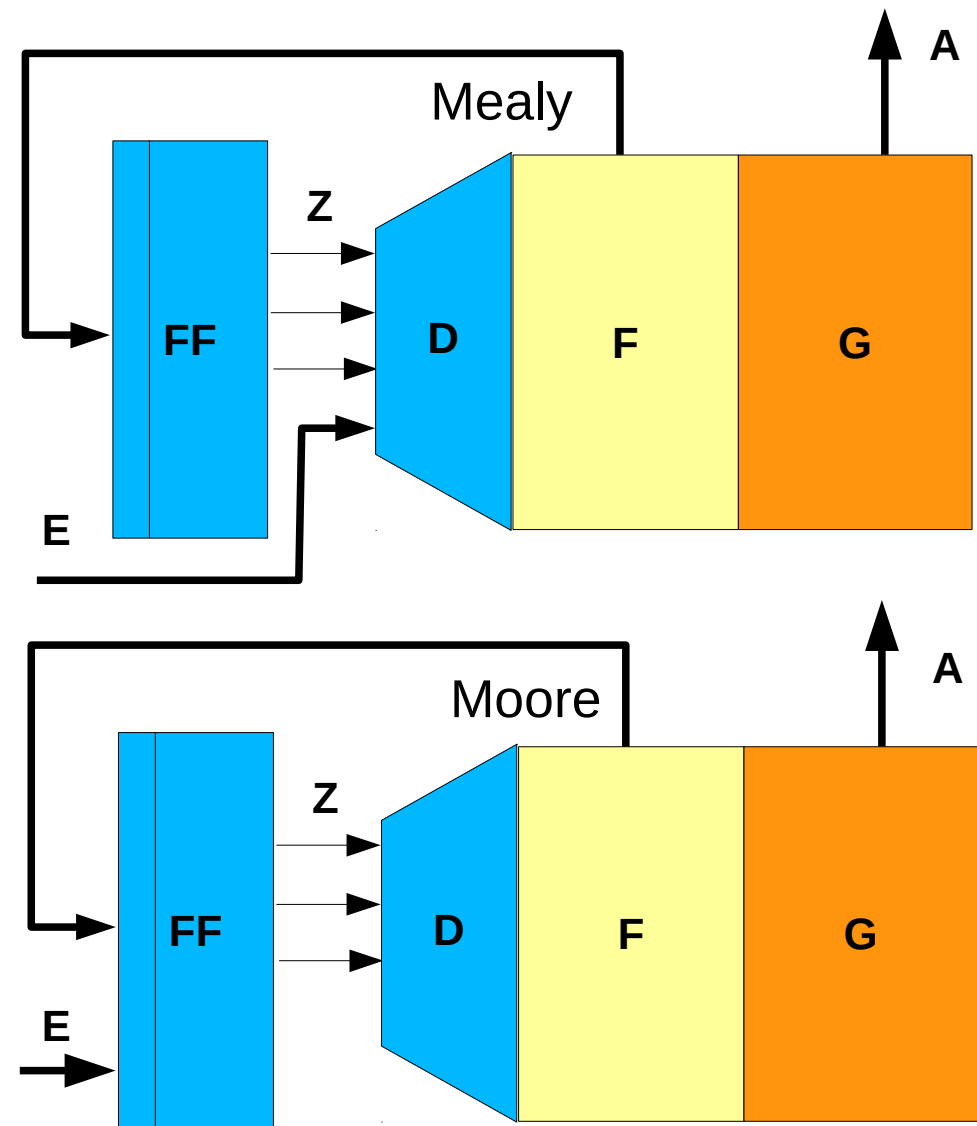
- Übergangsfunktion:  $\delta : Z \times E \rightarrow Z$
- Ausgangsfunktion:  $\lambda : Z \rightarrow A$  (Moore)  
 $\lambda : Z \times E \rightarrow A$  (Mealy)
- Alle rückgekoppelten Leitungen verzögert/getaktet (Flip/Flops)

# Mikroprogrammiertes Steuerwerk

## Mealy oder Moore?

- Einfachste Implementierung zeigt kaum Unterschiede
- Moore-Automaten sind langsamer, da sie Ausgänge nur nach Taktung ändern
- Wichtigste Rückkopplung besteht mit der ALU
- Kombination Mealy/Mealy mit ALU ist gefährlich: Wettläufe (Races)
- Moore/Moore ist langsam

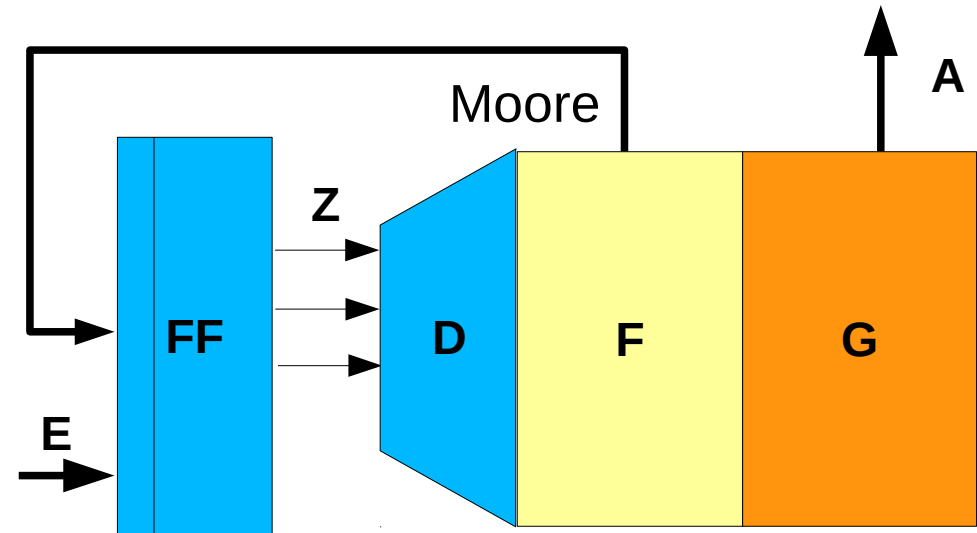
FF: Flip/Flop  
F: Folgematrix  
G: Ausgangsmatrix





- Mikromaschinenprogramm hat folgenden Aufbau:








```
while(true){  
  switch(z){  
    case 0:  
      switch(e){  
        case 0: z=..;out(a);break;  
        case 1: z= ....  
      }  
    case 1:  
      switch(e){...  
      ...  
    }  
  }  
}
```



Sie sehen: Ein solches Mikroprogramm hat keine Freiheitsgrade.

Solche naiven Steuerwerke benötigen sehr viele Speicherzellen.

Zur Übung soll jedoch ein solches Steuerwerk im Labor aufgebaut werden

$A = \{$   ,  ,  ,  ,  ,  ,   $\}$

A = {0111, 1110, 0110, 1100, 0100, 1000}

Der Würfel soll vorwärts zählen, wenn Taster 1 gedrückt ist und rückwärts zählen, wenn Taster 2 gedrückt wird. In den beiden anderen Fällen steht der Würfel.

- Im Labor erhalten Sie folgende Bauteile:
  - Einen EEPROM als Mikroprogrammspeicher
  - Zwei Taster
  - Eine Würfelanzeige
  - Ein Latch als Speicher für die inneren Zustände
- Schreiben Sie das Mikroprogramm, geben Sie die textuelle Darstellung und den hexadezimalen Speicherinhalt an.
- 2. Aufgabe: Zinken Sie den Würfel, so dass er besonders häufig Sechsen würfelt.
- Zur Lösung stehen zwei Labordoppelstunden zur Verfügung.

# Schaltplan - Würfelsteuerwerk

