Organisation

- Vorlesungsevaluation

Bedingungen

- Gleichheit (==, <, <=, > und >=)
- Keine Booleans (If … == True)

Rechenblöcke

- Eine oder Mehrere Rechenoperationen
  - Unär oder binäre Operationen
- Zuweisung: <*Variable*> := <*Wert*>
  - $R_3 := R_1 + R_2$
  - $R_1 := 42$
- Laden/Speichern
  - Laden/Speichern eines Wertes an Adresse *x* im Speicher

Toy-Prozessor

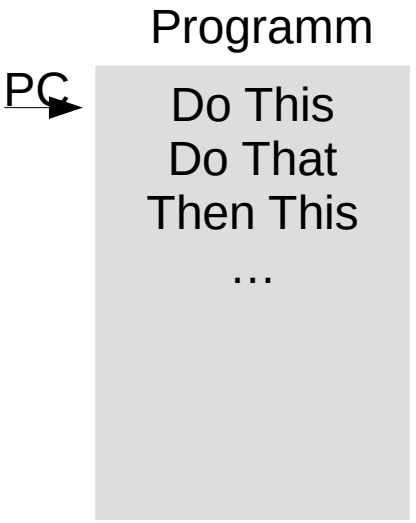
- CPU = Prozessor
- Register R_i
  - (Wenige) Mehrbit Speicherzellen
- Arithmetische Logische Einheit (ALU)
  - Schaltungen für arithmetische Operationen
  - Weitere Schaltungen zur Programmsteuerung
- Speicher *S[i]*
  - Viele Mehrbit Speicherzellen

Register-Transfer-Ebene

- Simple „Assemblerprogramme“
- Wenige Operationen
  - Laden/Speichern
  - Arithmetische/Boolsche Operationen
  - Ablaufsteuerung
- Prozessornah > Verständlichkeit
- Schwer abzufragen, daher gut auf Lücke zu lernen

Programm Counter + Jumps

- PC gibt die Zeilennr. des Programms an
  - Bzw. der aktuelle/nächste Befehl
  - NICHT die Zeile im Dokument
  - Nach Befehl autom. erhöht
- Jumps setzten PC um
  - Jumpen, wenn ACCU == 0



ALU, Akkumulator, Speicher

- ALU stellt Logik
  - Speicherzugriff
  - Programmsteuerung
  - Rechnen
- Akkumulator ein Eingang der ALU UND Ausgang
- Zweiter ALU Eingang direkt aus Speicher

Toy-Prozessor Assembler

- PC beginnt bei 0
- Initiale Werte MÜSSEN definiert sein
  - **:<*Speicheraddresse*>:<*Wert*>**
  - Default-Wert: 0
- Endlos Loop am Ende
  - **ZRO;**
  - **BRZ _____;**
- Nach Semikolon wird ignoriert → **Kommentare!!!!!**

Toy-Prozessor Assembler

- Good vs Bad Code

Wichtig: Prozessor/Assembler

- Blockformen im Ablaufdiagramm
- Kommentare: Was und Wozu?
  - Keine Kommentare = Abzüge
- Assembler Programme testen
- Anzahl der Register beachten
- Speichern/Laden des ACCs, falls neue Werte geladen werden sollen
- Doku des ToyPorzessors und Beispiele anschauen