

# Tutorium 17, #5

Max Göckel– [uzkns@student.kit.edu](mailto:uzkns@student.kit.edu)

Institut für Theoretische Informatik - Grundbegriffe der Informatik

Man kann sich den Speicher als eine Art Tabelle vorstellen. Die linke Spalte sind die Adressen, rechte Spalte die Daten an der jeweiligen Adresse.

Bei einer Tabelle wird immer nur ein Zustand gespeichert, nicht eine "Referenz" auf die Tabelle.

	Stelle	Wert
	00	010
Bsp: Speicher $m_0$	01	000
	10	111
	11	100

Schreiben in den Speicher mit:

## Definition

- $memwrite : Val^{Adr} \times Adr \times Val \rightarrow Val^{Adr}$ .
- $(m, a, v) \mapsto m'$

Dies schreibt den Wert **v** an Adresse **a** von Speichertabellenzustand **m** und gibt den veränderten Zustand **m'** zurück.

Lesen aus dem Speicher mit:

## Definition

- $memread : Val^{Adr} \times Adr \rightarrow Val$
- $(m, a) \mapsto m(a)$

Dies liest aus der Adresse **a** von Zustand **m** aus und gibt diesen Wert zurück.

Da die Operationen einen Rückgabewert haben, können wir sie Anstelle von konkreten Werten in anderen Operationen einsetzen.

Beispiel: Wert aus Stelle 00 auslesen und in 01 einsetzen.

	Stelle	Wert
$m_1$ :	00	010
	01	000
	10	111
	11	100

Wie sieht  $m_3$  nach der folgenden verketteten Operation auf Tabelle  $m_2$  aus?

■  $m_3 = \text{memwrite}(m_2, 11, \text{memread}(\text{memwrite}(m_2, 10, 000), 00))$

	Stelle	Wert
$m_2$ :	00	010
	01	000
	10	111
	11	100

Wie sieht  $m_3$  nach der folgenden verketteten Operation auf Tabelle  $m_2$  aus?

■  $m_3 = \text{memwrite}(m_2, 11, \text{memread}(\text{memwrite}(m_2, 10, 000), 00))$

	Stelle	Wert
$m_3$ :	00	010
	01	000
	10	111
	11	010

- Steuerwerk
  - Holt Befehle aus dem Speicher, decodiert sie und steuert die Ausführung
- Rechenwerk
  - Führt arithmetische/logische Operationen aus
- Speicher
  - Speichert Daten an Adressen, besteht aus Programm- und Datenspeicher
  - Adressen 20Bit, Werte 24Bit
- Register
  - Speichern je ein Wort oder eine Adresse (20/24Bit)
- Bus
  - Verbindet die Komponenten



Führt die vom Steuerwerk decodierten Befehle aus (logisch oder arithmetisch).

- X, Y: Eingabewerte
- Z: Ausgabe der verarbeiteten Werte

Bitweise Operationen der ALU:

Name	Operation in Worten
ADD	Zahl 1 + Zahl 2
AND	1 $\Leftrightarrow$ beide Eingabwerte 1 sind (logische $\wedge$ )
OR	1, wenn min. 1 wert 1 ist (logisches $\vee$ )
NOT	Jeder wert wird invertiert (Negierung)
XOR	1 wenn genau ein Wert 1 ist (exclusive or)

- Akku: Akkumulator (Zwischenspeicher)
- X, Y: ALU-Operanden
- Z: ALU-Ergebnis
- EINS: Die Konstante 1
- IAR: Instruktionsadressreg., Speichert den Speicherwert des nächsten Befehls
- IR: Instruktionsreg., Speichert die derzeitige Instruktion
- SAR: Speicheradressreg., (write-only) ruft im Speicher seinen wert auf
- SDR: Speicherdatenreg., Wert→Speicher oder Speicher→Wert

- Maschinenbefehle/Assemblersprache: Rechen- und Sprungbefehle
- Mikrobefehle: Bitfolgen, die an das Steuerwerk gesendet werden

Die Mikrobefehle werden in 7+ Takten abgearbeitet. Wie lange ein Takt (bzw. wann der nächste Takt beginnt) steuert ein Taktquarz im Steuerwerk.

Auf einer Extra-PDF im ILIAS.

- Die Befehle werden aufsteigend durchnummeriert und dann abgearbeitet
- Alle Werte immer in binär aufschreiben!
- Es ist nicht möglich mehr als 20Bit lange Worte zu laden und es ist auch nicht möglich negative Zahlen zu laden
- Kommentare werden im Format *;<Kommentar>* geschrieben
- Jedes Assemblerprogramm endet mit einem *HALT* (Punktabzug!)

Was tut das folgende Programm?

```
LDC 1
STV 011
LDV 010
NOT
ADD 011
ADD 001
JMN here
LDC 1
STV 111 ;result
JMP end
here: LDC 0
      STV 111 ;result
end: HALT
```

# Die MIMA: Aufgabe

Schreibe ein Programm, dass den Wert an Adresse 001 verdreifacht.

Schreibe ein Programm, dass den Wert an Adresse 001 verdreifacht.

```
LDV 001  
ADD 001  
ADD 001  
STV 001  
HALT.
```



Schreibe ein Programm dass ein ganzzahliges  $x$  aus der Stelle 111 lädt,  $-|x|$  bildet und das Ergebnis wieder in 11 speichert.

Schreibe ein Programm dass ein ganzzahliges  $x$  aus der Stelle 111 lädt,  $-|x|$  bildet und das Ergebnis wieder in 11 speichert.

```
LDV 001
JMN m1
NOT
ADD EINS
STV 001
m1: HALT
```

Realisiert die Funktion:

$$\max(a, b) = \begin{cases} b & b \geq a \\ a & b < a \end{cases}$$

Wobei  $a$  in 001 steht,  $b$  in 010 und das Ergebnis in beide Speicherplätze (001,010) geschrieben werden soll.

# Die MIMA: Lösung

```
LDV 001
NOT
ADD EINS
ADD 010
JMN m1
LDV 010
STV 001
JMP m2
m1: LDV 001
    STV 010
m2: HALT.
```