

Grundbegriffe der Informatik

Tutorium 33

Lukas Bach, lukas.bach@student.kit.edu | 1.12.2016



Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

■ Was ist sind die folgenden Mengen?

MIMA

Maschinenbefehle

Aufgaben

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

■ Was ist sind die folgenden Mengen?

■ \mathbb{N}

MIMA

Maschinenbefehle

Aufgaben

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

- Was ist sind die folgenden Mengen?
 - \mathbb{N} = Menge der natürlichen Zahlen (1, 2, 3, ...)

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

- Was ist sind die folgenden Mengen?
 - \mathbb{N} = Menge der natürlichen Zahlen (1, 2, 3, ...)
 - \mathbb{N}_0

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

- Was ist sind die folgenden Mengen?
 - \mathbb{N} = Menge der natürlichen Zahlen (1, 2, 3, ...)
 - $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

- Was ist sind die folgenden Mengen?
 - \mathbb{N} = Menge der natürlichen Zahlen (1, 2, 3, ...)
 - $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$
 - \mathbb{R}

- Was ist sind die folgenden Mengen?
 - \mathbb{N} = Menge der natürlichen Zahlen (1, 2, 3, ...)
 - $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$
 - \mathbb{R} = Menge der Reellen Zahlen

■ Was ist sind die folgenden Mengen?

- \mathbb{N} = Menge der natürlichen Zahlen (1, 2, 3, ...)
- $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$
- \mathbb{R} = Menge der Reellen Zahlen
- \mathbb{R}^+

■ Was ist sind die folgenden Mengen?

- \mathbb{N} = Menge der natürlichen Zahlen (1, 2, 3, ...)
- $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$
- \mathbb{R} = Menge der Reellen Zahlen
- \mathbb{R}^+ = Menge der positiven reellen Zahlen

■ Was ist sind die folgenden Mengen?

- \mathbb{N} = Menge der natürlichen Zahlen (1, 2, 3, ...)
- $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$
- \mathbb{R} = Menge der Reellen Zahlen
- \mathbb{R}^+ = Menge der positiven reellen Zahlen
- \mathbb{R}_0

■ Was ist sind die folgenden Mengen?

- \mathbb{N} = Menge der natürlichen Zahlen (1, 2, 3, ...)
- $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$
- \mathbb{R} = Menge der Reellen Zahlen
- \mathbb{R}^+ = Menge der positiven reellen Zahlen
- \mathbb{R}_0 gibt es nicht! 0 ist auch so schon in \mathbb{R}

■ Was ist sind die folgenden Mengen?

- \mathbb{N} = Menge der natürlichen Zahlen (1, 2, 3, ...)
- $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$
- \mathbb{R} = Menge der Reellen Zahlen
- \mathbb{R}^+ = Menge der positiven reellen Zahlen
- \mathbb{R}_0 gibt es nicht! 0 ist auch so schon in \mathbb{R}
- \mathbb{R}_0^+ genauso nicht!

■ Was ist sind die folgenden Mengen?

- \mathbb{N} = Menge der natürlichen Zahlen (1, 2, 3, ...)
- $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$
- \mathbb{R} = Menge der Reellen Zahlen
- \mathbb{R}^+ = Menge der positiven reellen Zahlen
- \mathbb{R}_0 gibt es nicht! 0 ist auch so schon in \mathbb{R}
- \mathbb{R}_0^+ genauso nicht!

■ Aufgabe: $R : A^* \rightarrow A^*$

- $R(\varepsilon) = \varepsilon$
- $\forall x \in A : R(x) = x$
- $\forall w \in A^* \forall x \in A \forall y \in A^* : R(xwy) = yR(w)x$
- Zeige: $\forall n \in \mathbb{N}_0 : \forall w \in A^n : |R(w)| = |w|$

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

■ Theoretischer, idealisierter Prozessor

MIMA

Maschinenbefehle

Aufgaben

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

- Theoretischer, idealisierter Prozessor
- Funktioniert wie ein echter Prozessor, ist aber simpler

Maschinenbefehle

Aufgaben

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

- Theoretischer, idealisierter Prozessor
- Funktioniert wie ein echter Prozessor, ist aber simpler
- Nah an Technischer Informatik

- Theoretischer, idealisierter Prozessor
- Funktioniert wie ein echter Prozessor, ist aber simpler
- Nah an Technischer Informatik

Grundaufbau:

- Theoretischer, idealisierter Prozessor
- Funktioniert wie ein echter Prozessor, ist aber simpler
- Nah an Technischer Informatik

Grundaufbau:

- Adressen als 20*bit* Datenwort

- Theoretischer, idealisierter Prozessor
- Funktioniert wie ein echter Prozessor, ist aber simpler
- Nah an Technischer Informatik

Grundaufbau:

- Adressen als 20*bit* Datenwort
- Speicherworte als 24*bit* Datenwort

Was ist die MIMA?

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

- Theoretischer, idealisierter Prozessor
- Funktioniert wie ein echter Prozessor, ist aber simpler
- Nah an Technischer Informatik

Grundaufbau:

- Adressen als *20bit* Datenwort
- Speicherworte als *24bit* Datenwort
- Maschinenbefehle als...
 - *4bit* Befehl und *20bit* Adresse
 - oder *8bit* Befehl und unwichtigem Rest

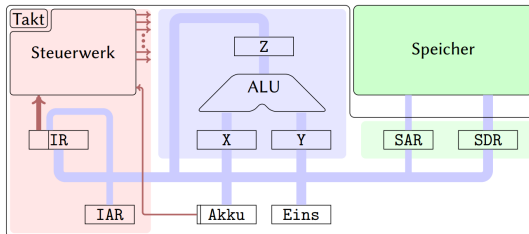
Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



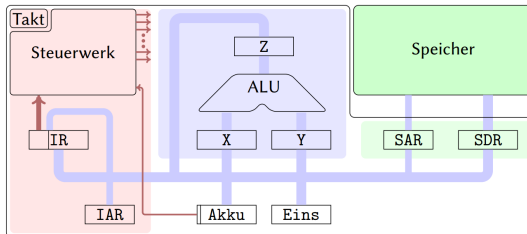
Steuerwerk

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Steuerwerk

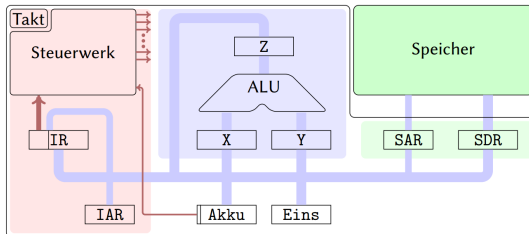
- Instruction Register (IR) enthält den nächsten auszuführenden Befehl

Zum Übungsblatt

MIMA

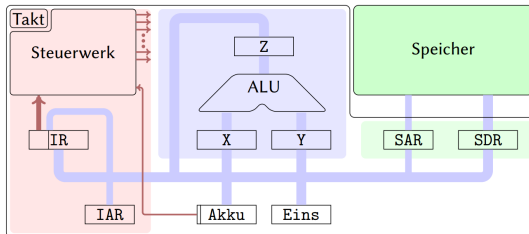
Maschinenbefehle

Aufgaben



Steuerwerk

- Instruction Register (IR) enthält den nächsten auszuführenden Befehl
- Instruction Address Register (IAR) enthält die Adresse des nächsten Befehls



Steuerwerk

- Instruction Register (IR) enthält den nächsten auszuführenden Befehl
- Instruction Adress Register (IAR) enthält die Adresse des nächsten Befehls
- Takt bestimmt die "Tickrate", also die Geschwindigkeit



Steuerwerk

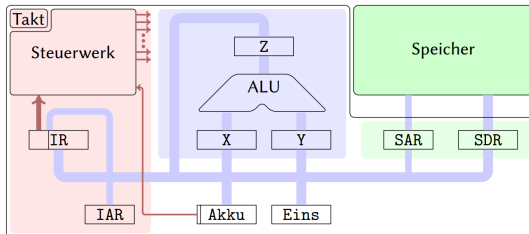
- Instruction Register (IR) enthält den nächsten auszuführenden Befehl
- Instruction Adress Register (IAR) enthält die Adresse des nächsten Befehls
- Takt bestimmt die “Tickrate”, also die Geschwindigkeit
- Steuerwerk interpretiert alle Befehle und führt sie aus

Zum Übungsblatt

MIMA

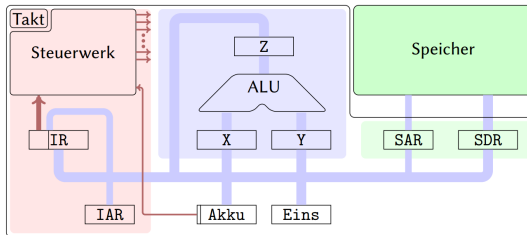
Maschinenbefehle

Aufgaben



Steuerwerk

- Instruction Register (IR) enthält den nächsten auszuführenden Befehl
- Instruction Adress Register (IAR) enthält die Adresse des nächsten Befehls
- Takt bestimmt die “Tickrate”, also die Geschwindigkeit
- Steuerwerk interpretiert alle Befehle und führt sie aus
- Welche Befehle es gibt: Siehe später



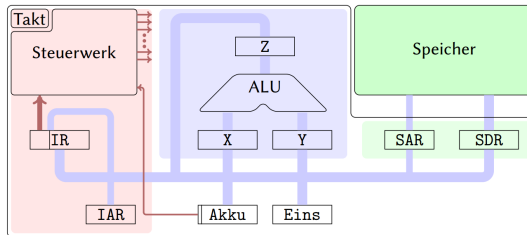
Aufbau der MIMA: Akku und Eins

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Akku und Eins

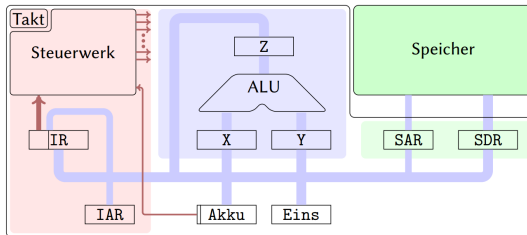
Aufbau der MIMA: Akku und Eins

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Akku und Eins

- Akku dient als Zwischenspeicher für Datenworte

Aufbau der MIMA: Akku und Eins

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Akku und Eins

- Akku dient als Zwischenspeicher für Datenworte
- Hält maximal ein Wort

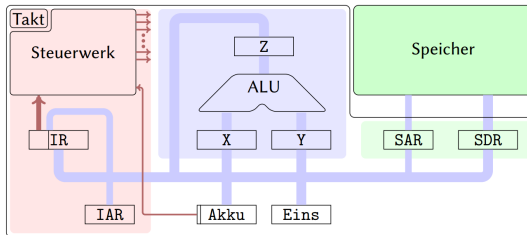
Aufbau der MIMA: Akku und Eins

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Akku und Eins

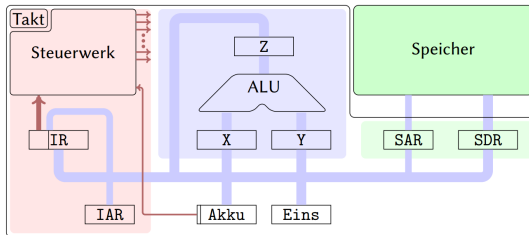
- Akku dient als Zwischenspeicher für Datenworte
- Hält maximal ein Wort
- Eins liefert die Konstante 1, hält also Strom

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Akku und Eins

- Akku dient als Zwischenspeicher für Datenwörter
- Hält maximal ein Wort
- Eins liefert die Konstante 1, hält also Strom
- z.B. erhöhen des IAR

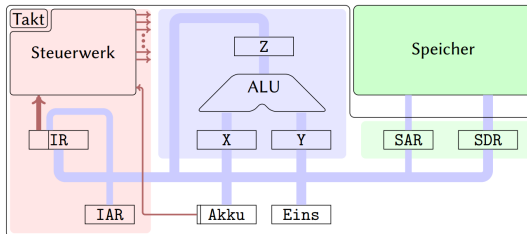
Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



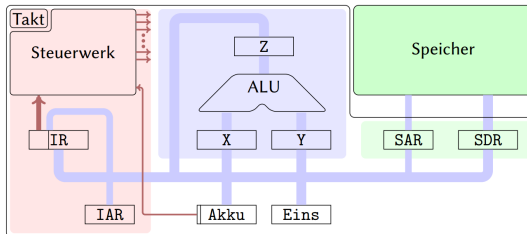
Aufbau der MIMA: ALU

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Arithmetic Logic Unit (ALU) / Rechenwerk

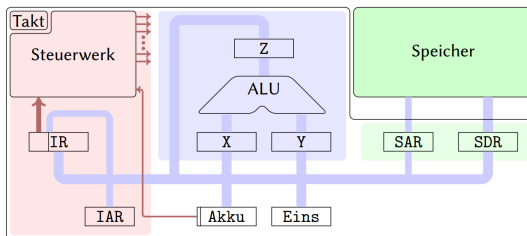
Aufbau der MIMA: ALU

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Arithmetic Logic Unit (ALU) / Rechenwerk

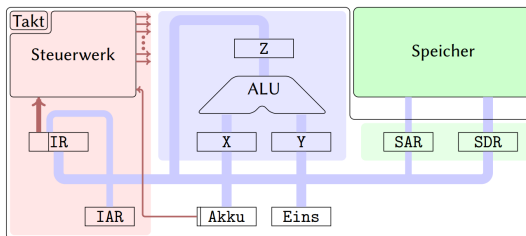
- Durchführt arithmetische Operationen

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Arithmetic Logic Unit (ALU) / Rechenwerk

- Durchführt arithmetische Operationen
- **mod** , **div** , $+$, $-$, ... , bitweises OR/AND/...

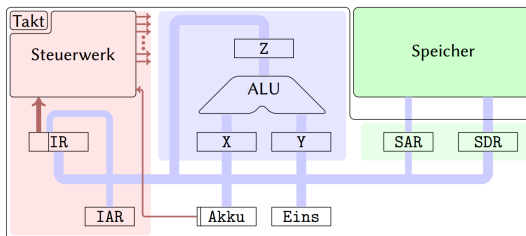
Aufbau der MIMA: ALU

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Arithmetic Logic Unit (ALU) / Rechenwerk

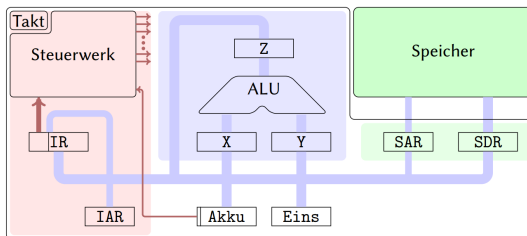
- Durchführt arithmetische Operationen
- **mod** , **div** , +, −, ..., bitweises OR/AND/...
- X und Y sind Eingaberegister

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Arithmetic Logic Unit (ALU) / Rechenwerk

- Durchführt arithmetische Operationen
- **mod** , **div** , +, −, ..., bitweises OR/AND/...
- X und Y sind Eingaberegister
- Z ist Ausgaberegister

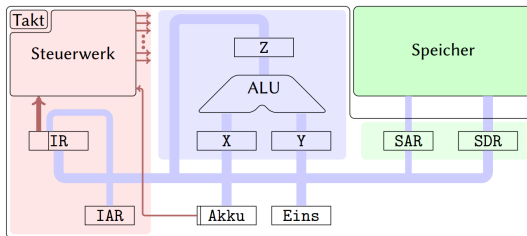
Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Aufbau der MIMA: ALU

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Speicher(werk)

Speicher selbst speichert Befehle und Daten.

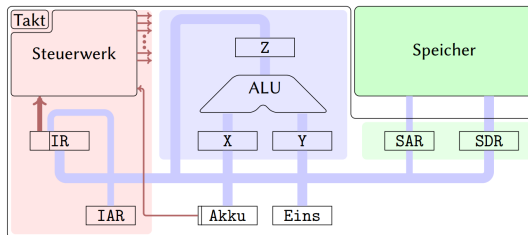
Aufbau der MIMA: ALU

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Speicher(werk)

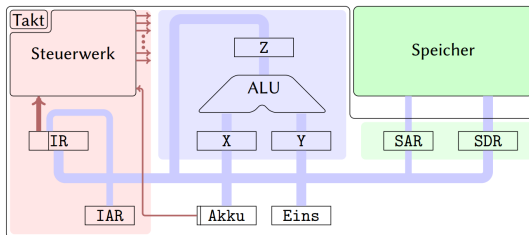
Speicher selbst speichert Befehle und Daten. Speicherwerk besteht aus:

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Speicher(werk)

Speicher selbst speichert Befehle und Daten. Speicherwerk besteht aus:

- Speicheradressregister (SAR)
ist die Adresse, bei der im
Speicher gespeichert/gelesen
werden soll

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Speicher(werk)

Speicher selbst speichert Befehle und Daten. Speicherwerk besteht aus:

- Speicheradressregister (SAR) ist die Adresse, bei der im Speicher gespeichert/gelesen werden soll
- Speicherdatenregister (SDR) Datum, das bei der Adresse gespeichert werden soll/gelesen wurde.

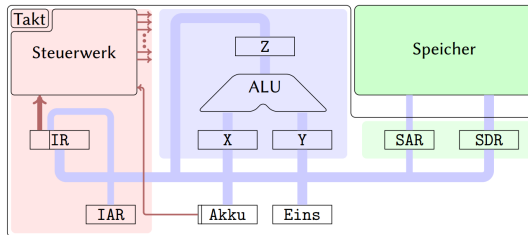
Lukas Bach, lu-
kas.bach@student.kit.edu

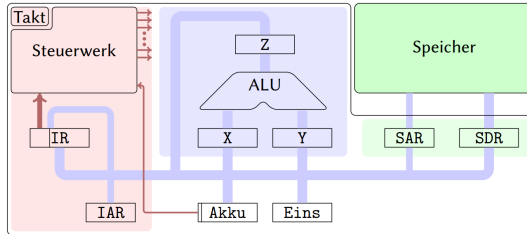
Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben





Busse

Aufbau der MIMA: ALU

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Busse

- “Kabel” zwischen den Verbindungen

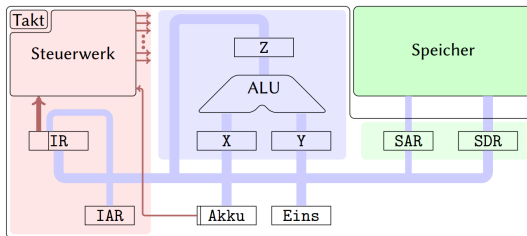
Aufbau der MIMA: ALU

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Busse

- “Kabel” zwischen den Verbindungen
- Ein kompletter Bus überträgt entweder 1, 0, oder nichts

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



Busse

- “Kabel” zwischen den Verbindungen
- Ein kompletter Bus überträgt entweder 1, 0, oder nichts
- Kann nur eine einzige Information auf einmal übertragen

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Um MIMA Programme und dazugehörige Definitionen verständlicher zu machen, vereinbaren wir folgende Konventionen:

Maschinenbefehle

Aufgaben

Um MIMA Programme und dazugehörige Definitionen verständlicher zu machen, vereinbaren wir folgende Konventionen:

- Befehle (eigentlich Bitfolge) schreiben wir als Befehlname und Adresse

Um MIMA Programme und dazugehörige Definitionen verständlicher zu machen, vereinbaren wir folgende Konventionen:

- Befehle (eigentlich Bitfolge) schreiben wir als Befehlname und Adresse
 - `0010000000000000000101010` \equiv *STV 42*

Um MIMA Programme und dazugehörige Definitionen verständlicher zu machen, vereinbaren wir folgende Konventionen:

- Befehle (eigentlich Bitfolge) schreiben wir als Befehlsname und Adresse
 - $0010000000000000000101010 \equiv STV\ 42$
- $X \leftarrow Y \equiv$ "Der Variable X wird der Wert Y zugewiesen"

MIMA Befehle

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

| Befehlssyntax | Formel | Bedeutung |
|---------------|--------|-----------|
|---------------|--------|-----------|

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

MIMA Befehle

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

| Befehlssyntax | Formel | Bedeutung |
|------------------|---------------------------------------|--|
| <i>LDC const</i> | <i>Akku</i> \leftarrow <i>const</i> | Lade eine Konstante <i>const</i> in den Akku |

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

MIMA Befehle

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

| Befehlssyntax | Formel | Bedeutung |
|------------------|--------------------------|---|
| <i>LDC const</i> | $Akku \leftarrow const$ | Lade eine Konstante <i>const</i> in den Akku |
| <i>LDV adr</i> | $Akku \leftarrow M(adr)$ | Lade einen Wert vom Speicher bei Adresse <i>adr</i> in den Akku |

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

MIMA Befehle

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

| Befehlssyntax | Formel | Bedeutung |
|------------------|--------------------------|---|
| <i>LDC const</i> | $Akku \leftarrow const$ | Lade eine Konstante <i>const</i> in den Akku |
| <i>LDV adr</i> | $Akku \leftarrow M(adr)$ | Lade einen Wert vom Speicher bei Adresse <i>adr</i> in den Akku |
| <i>STV adr</i> | $M(adr) \leftarrow Akku$ | Lade Speichere den Wert aus dem Akku im Speicher bei Adresse <i>adr</i> |

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

MIMA Befehle

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

| Befehlssyntax | Formel | Bedeutung |
|------------------|-----------------------------|---|
| <i>LDC const</i> | $Akku \leftarrow const$ | Lade eine Konstante <i>const</i> in den Akku |
| <i>LDV adr</i> | $Akku \leftarrow M(adr)$ | Lade einen Wert vom Speicher bei Adresse <i>adr</i> in den Akku |
| <i>STV adr</i> | $M(adr) \leftarrow Akku$ | Lade Speichere den Wert aus dem Akku im Speicher bei Adresse <i>adr</i> |
| <i>LDIV adr</i> | $Akku \leftarrow M(M(adr))$ | Lade einen Wert vom Speicher bei der Adresse, die bei <i>adr</i> gespeichert ist, und lade den Wert in den Akku |

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

MIMA Befehle

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

| Befehlssyntax | Formel | Bedeutung |
|------------------|-----------------------------|---|
| <i>LDC const</i> | $Akku \leftarrow const$ | Lade eine Konstante <i>const</i> in den Akku |
| <i>LDV adr</i> | $Akku \leftarrow M(adr)$ | Lade einen Wert vom Speicher bei Adresse <i>adr</i> in den Akku |
| <i>STV adr</i> | $M(adr) \leftarrow Akku$ | Lade Speichere den Wert aus dem Akku im Speicher bei Adresse <i>adr</i> |
| <i>LDIV adr</i> | $Akku \leftarrow M(M(adr))$ | Lade einen Wert vom Speicher bei der Adresse, die bei <i>adr</i> gespeichert ist, und lade den Wert in den Akku |
| <i>STIV adr</i> | $M(M(adr)) \leftarrow Akku$ | Speichere den Wert im Akku bei der Adresse, die in <i>adr</i> gespeichert ist. |

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

| Befehlssyntax | Formel | Bedeutung |
|---------------|--------|-----------|
|---------------|--------|-----------|

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

| Befehlssyntax | Formel | Bedeutung |
|----------------|---------------------------------|--|
| <i>ADD adr</i> | $Akku \leftarrow Akku + M(adr)$ | Addiere den Wert bei <i>adr</i> zum Akku dazu. |

MIMA

Maschinenbefehle

Aufgaben

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

| Befehlssyntax | Formel | Bedeutung |
|---------------|---------------------------------|--|
| $ADD\ adr$ | $Akku \leftarrow Akku + M(adr)$ | Addiere den Wert bei adr zum Akku dazu. |
| $"OP" adr$ | $Akku "OP" M(adr)$ | Wende bitweise Operation auf Akku mit Wert bei adr an. $Op \in \{AND, OR, XOR\}$. |

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

| Befehlssyntax | Bedeutung |
|---------------|-----------|
|---------------|-----------|

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

| Befehlssyntax | Bedeutung |
|---------------|---|
| <i>NOT</i> | Bitweise Invertierung aller Bits des Akku-Datenwortes |

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

| Befehlssyntax | Bedeutung |
|---------------|---|
| <i>NOT</i> | Bitweise Invertierung aller Bits des Akku-Datenwortes |
| <i>RAR</i> | Rotiere alle Akku-Bits eins nach rechts |

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

| Befehlssyntax | Bedeutung |
|----------------|---|
| <i>NOT</i> | Bitweise Invertierung aller Bits des Akku-Datenwortes |
| <i>RAR</i> | Rotiere alle Akku-Bits eins nach rechts |
| <i>EQL adr</i> | Setze Akku auf 11 ··· 11, falls Wert bei <i>adr</i> gleich Akku-Wert, setze Akku auf 00 ··· 00 sonst. |

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

| Befehlssyntax | Bedeutung |
|----------------|---|
| <i>NOT</i> | Bitweise Invertierung aller Bits des Akku-Datenwortes |
| <i>RAR</i> | Rotiere alle Akku-Bits eins nach rechts |
| <i>EQL adr</i> | Setze Akku auf 11 · · · 11, falls Wert bei <i>adr</i> gleich Akku-Wert, setze Akku auf 00 · · · 00 sonst. |
| <i>JMP adr</i> | Springe zu Befehlsadresse <i>adr</i> |

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

| Befehlssyntax | Bedeutung |
|----------------|---|
| <i>NOT</i> | Bitweise Invertierung aller Bits des Akku-Datenwortes |
| <i>RAR</i> | Rotiere alle Akku-Bits eins nach rechts |
| <i>EQL adr</i> | Setze Akku auf 11 ··· 11, falls Wert bei <i>adr</i> gleich Akku-Wert, setze Akku auf 00 ··· 00 sonst. |
| <i>JMP adr</i> | Springe zu Befehlsadresse <i>adr</i> |
| <i>JMN adr</i> | Springe zu Befehlsadresse <i>adr</i> , falls Akku negativ (also erstes Bit = 1), sonst fahre normal fort. |

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

Lukas Bach, lu-
kas.bach@student.kit.edu

- Befehle zum laden und Speichern in den Speicher

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

MIMA Befehle: Sichern und Laden

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

- Befehle zum laden und Speichern in den Speicher
- LDV um Daten vom Speicher zu laden, STV um Daten in den Speicher zu schreiben

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

- Befehle zum laden und Speichern in den Speicher
- LDV um Daten vom Speicher zu laden, STV um Daten in den Speicher zu schreiben
- LDC um eine Konstante zu laden

- Befehle zum laden und Speichern in den Speicher
- LDV um Daten vom Speicher zu laden, STV um Daten in den Speicher zu schreiben
- LDC um eine Konstante zu laden
- Daten werden in einem Zwischenspeicher gelagert, der nur ein Datenwort hält

- Befehle zum laden und Speichern in den Speicher
- LDV um Daten vom Speicher zu laden, STV um Daten in den Speicher zu schreiben
- LDC um eine Konstante zu laden
- Daten werden in einem Zwischenspeicher gelagert, der nur ein Datenwort hält: Akku.

- Befehle zum laden und Speichern in den Speicher
- LDV um Daten vom Speicher zu laden, STV um Daten in den Speicher zu schreiben
- LDC um eine Konstante zu laden
- Daten werden in einem Zwischenspeicher gelagert, der nur ein Datenwort hält: Akku.

Beispiele:

- Befehle zum laden und Speichern in den Speicher
- LDV um Daten vom Speicher zu laden, STV um Daten in den Speicher zu schreiben
- LDC um eine Konstante zu laden
- Daten werden in einem Zwischenspeicher gelagert, der nur ein Datenwort hält: Akku.

Beispiele:

- *LDV 9* lädt das Datum, das im Speicher bei Adresse 9 liegt, in den Akku.

- Befehle zum laden und Speichern in den Speicher
- LDV um Daten vom Speicher zu laden, STV um Daten in den Speicher zu schreiben
- LDC um eine Konstante zu laden
- Daten werden in einem Zwischenspeicher gelagert, der nur ein Datenwort hält: Akku.

Beispiele:

- *LDV 9* lädt das Datum, das im Speicher bei Adresse 9 liegt, in den Akku.
- *STV 9* speichert das Datum, das im Akku liegt, in den Speicher an Adresse 9.

- Befehle zum laden und Speichern in den Speicher
- LDV um Daten vom Speicher zu laden, STV um Daten in den Speicher zu schreiben
- LDC um eine Konstante zu laden
- Daten werden in einem Zwischenspeicher gelagert, der nur ein Datenwort hält: Akku.

Beispiele:

- *LDV 9* lädt das Datum, das im Speicher bei Adresse 9 liegt, in den Akku.
- *STV 9* speichert das Datum, das im Akku liegt, in den Speicher an Adresse 9.
- *LDC 4* lädt die Zahl 4 in den Akku (also kein Speicherzugriff).

MIMA Befehle: Sichern und Laden

| Befehlssyntax | Formel | Bedeutung |
|------------------|--------------------------|---|
| <i>LDC const</i> | $Akku \leftarrow const$ | Lade eine Konstante <i>const</i> in den Akku |
| <i>LDV adr</i> | $Akku \leftarrow M(adr)$ | Lade einen Wert vom Speicher bei Adresse <i>adr</i> in den Akku |
| <i>STV adr</i> | $M(adr) \leftarrow Akku$ | Lade Speichere den Wert aus dem Akku im Speicher bei Adresse <i>adr</i> |

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

MIMA Befehle: Sichern und Laden

| Befehlssyntax | Formel | Bedeutung |
|------------------|--------------------------|---|
| <i>LDC const</i> | $Akku \leftarrow const$ | Lade eine Konstante <i>const</i> in den Akku |
| <i>LDV adr</i> | $Akku \leftarrow M(adr)$ | Lade einen Wert vom Speicher bei Adresse <i>adr</i> in den Akku |
| <i>STV adr</i> | $M(adr) \leftarrow Akku$ | Lade Speichere den Wert aus dem Akku im Speicher bei Adresse <i>adr</i> |

Beispielprogramm mit initialem Speicherabbild

LDC 5

STV a_1

LDC 7

STV a_2

:

:

LDV a_1

STV a_3

HALT

| Adresse | Wert |
|---------|------|
| a_1 | 0 |
| a_2 | 0 |
| a_3 | 0 |

MIMA Befehle: Indirektes Sichern und Laden

| Befehlssyntax | Formel | Bedeutung |
|-----------------|-----------------------------|---|
| <i>LDIV adr</i> | $Akku \leftarrow M(M(adr))$ | Lade einen Wert vom Speicher bei der Adresse, die bei <i>adr</i> gespeichert ist, und lade den Wert in den Akku |
| <i>STIV adr</i> | $M(M(adr)) \leftarrow Akku$ | Speichere den Wert im Akku bei der Adresse, die in <i>adr</i> gespeichert ist. |

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

MIMA Befehle: Indirektes Sichern und Laden

| Befehlssyntax | Formel | Bedeutung |
|-----------------|-----------------------------|---|
| <i>LDIV adr</i> | $Akku \leftarrow M(M(adr))$ | Lade einen Wert vom Speicher bei der Adresse, die bei <i>adr</i> gespeichert ist, und lade den Wert in den Akku |
| <i>STIV adr</i> | $M(M(adr)) \leftarrow Akku$ | Speichere den Wert im Akku bei der Adresse, die in <i>adr</i> gespeichert ist. |

Beispielprogramm mit initialem Speicherabbild

LDIV 4
STV 5
LDIV 5
STIV 4
HALT

| Adresse | Wert |
|---------|------|
| 4 | 6 |
| 5 | 0 |
| 6 | 7 |
| 7 | 2 |

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

■ Befehle zu arithmetischen Operationen

- Befehle zu arithmetischen Operationen
- Eine ALU-Operation, angewandt auf dem Wert des Akkus und dem Wert an gegebener Adresse

- Befehle zu arithmetischen Operationen
- Eine ALU-Operation, angewandt auf dem Wert des Akkus und dem Wert an gegebener Adresse
- Beispiele:

- Befehle zu arithmetischen Operationen
- Eine ALU-Operation, angewandt auf dem Wert des Akkus und dem Wert an gegebener Adresse
- Beispiele:
 - *ADD 4* addiert den Wert im Akku mit dem Wert aus dem Speicher an Adresse 4 und legt das Resultat im Akku ab

- Befehle zu arithmetischen Operationen
- Eine ALU-Operation, angewandt auf dem Wert des Akkus und dem Wert an gegebener Adresse
- Beispiele:
 - *ADD 4* addiert den Wert im Akku mit dem Wert aus dem Speicher an Adresse 4 und legt das Resultat im Akku ab. Achtung: Addition nicht mit dem Wert 4!

- Befehle zu arithmetischen Operationen
- Eine ALU-Operation, angewandt auf dem Wert des Akkus und dem Wert an gegebener Adresse
- Beispiele:
 - *ADD* 4 addiert den Wert im Akku mit dem Wert aus dem Speicher an Adresse 4 und legt das Resultat im Akku ab. Achtung: Addition nicht mit dem Wert 4!
 - *AND* 3 führt bitweise Verundung zwischen dem Wert im Akku und dem Wert aus dem Speicher an Adresse 4 durch und legt das Resultat im Akku ab.

MIMA Befehle: Eins plus Eins

| Befehlssyntax | Formel | Bedeutung |
|-----------------|------------------------------------|---|
| <i>ADD adr</i> | $Akku \leftarrow Akku + M(adr)$ | Addiere den Wert bei <i>adr</i> zum Akku dazu. |
| <i>"OP" adr</i> | $Akku \leftarrow OP(Akku, M(adr))$ | Wende bitweise Operation auf Akku mit Wert bei <i>adr</i> an. $Op \in \{AND, OR, XOR\}$. |

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

MIMA Befehle: Eins plus Eins

| Befehlssyntax | Formel | Bedeutung |
|-----------------|---------------------------------|---|
| <i>ADD adr</i> | $Akku \leftarrow Akku + M(adr)$ | Addiere den Wert bei <i>adr</i> zum Akku dazu. |
| <i>"OP" adr</i> | $Akku \text{ "OP" } M(adr)$ | Wende bitweise Operation auf Akku mit Wert bei <i>adr</i> an. $Op \in \{AND, OR, XOR\}$. |

Beispielprogramm mit initialem Speicherabbild

LDC 5
ADD 3
AND 4
STV 5
LDC 12
XOR 5
HALT

| Adresse | Wert |
|---------|------|
| 3 | 3 |
| 4 | 8 |
| 5 | 17 |

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

- *NOT* invertiert alle Bits des Datums im Akku.

- *NOT* invertiert alle Bits des Datums im Akku. Beispiel *NOT* mit 5 im Akku, angenommen der Akku speichert bis zu 8 bits

- *NOT* invertiert alle Bits des Datums im Akku. Beispiel *NOT* mit 5 im Akku, angenommen der Akku speichert bis zu 8 bits:
 $5_{10} = 00000101_2$, nach der Invertierung: 11111010_2 .

- *NOT* invertiert alle Bits des Datums im Akku. Beispiel *NOT* mit 5 im Akku, angenommen der Akku speichert bis zu 8 bits:
 $5_{10} = 00000101_2$, nach der Invertierung: 11111010_2 .
- *RAR* rotiert alle Bits des Datums im Akku um eine Stelle nach rechts.

- *NOT* invertiert alle Bits des Datums im Akku. Beispiel *NOT* mit 5 im Akku, angenommen der Akku speichert bis zu 8 bits:
 $5_{10} = 00000101_2$, nach der Invertierung: 11111010_2 .
- *RAR* rotiert alle Bits des Datums im Akku um eine Stelle nach rechts. Beispiel mit 5 im Akku: 00000101_2 wird zu 10000010_2 .

- *NOT* invertiert alle Bits des Datums im Akku. Beispiel *NOT* mit 5 im Akku, angenommen der Akku speichert bis zu 8 bits:
 $5_{10} = 00000101_2$, nach der Invertierung: 11111010_2 .
- *RAR* rotiert alle Bits des Datums im Akku um eine Stelle nach rechts. Beispiel mit 5 im Akku: 00000101_2 wird zu 10000010_2 .
- *EQL adr* vergleicht den Wert im Akku mit dem Wert bei *adr*.

- *NOT* invertiert alle Bits des Datums im Akku. Beispiel *NOT* mit 5 im Akku, angenommen der Akku speichert bis zu 8 bits:
 $5_{10} = 00000101_2$, nach der Invertierung: 11111010_2 .
- *RAR* rotiert alle Bits des Datums im Akku um eine Stelle nach rechts. Beispiel mit 5 im Akku: 00000101_2 wird zu 10000010_2 .
- *EQL adr* vergleicht den Wert im Akku mit dem Wert bei *adr*.
 - Setzt Akku = 11 \dots 11 falls Werte gleich sind.

- *NOT* invertiert alle Bits des Datums im Akku. Beispiel *NOT* mit 5 im Akku, angenommen der Akku speichert bis zu 8 bits:
 $5_{10} = 00000101_2$, nach der Invertierung: 11111010_2 .
- *RAR* rotiert alle Bits des Datums im Akku um eine Stelle nach rechts. Beispiel mit 5 im Akku: 00000101_2 wird zu 10000010_2 .
- *EQL adr* vergleicht den Wert im Akku mit dem Wert bei *adr*.
 - Setzt Akku = 11 ... 11 falls Werte gleich sind.
 - Setzt Akku = 00 ... 00 falls Werte nicht gleich sind.

MIMA Befehle: Bits und Bytes

| Befehlssyntax | Bedeutung |
|----------------|---|
| <i>NOT</i> | Bitweise Invertierung aller Bits des Akku-Datenwortes |
| <i>RAR</i> | Rotiere alle Akku-Bits eins nach rechts |
| <i>EQL adr</i> | Setze Akku auf 11 ··· 11, falls Wert bei <i>adr</i> gleich Akku-Wert, setze Akku auf 00 ··· 00 sonst. |

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

| Befehlssyntax | Bedeutung |
|----------------|---|
| <i>NOT</i> | Bitweise Invertierung aller Bits des Akku-Datenwortes |
| <i>RAR</i> | Rotiere alle Akku-Bits eins nach rechts |
| <i>EQL adr</i> | Setze Akku auf 11 ··· 11, falls Wert bei <i>adr</i> gleich Akku-Wert, setze Akku auf 00 ··· 00 sonst. |

Beispielprogramm mit initialem Speicherabbild

| | |
|-------|--------|
| LDC 5 | : |
| NOT | : |
| RAR | RAR |
| NOT | EQL 15 |
| RAR | EQL 0 |
| : | HALT |

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

- Normalerweise wird die Instruktionsadresse nach jedem Befehl um eins erhöht

- Normalerweise wird die Instruktionsadresse nach jedem Befehl um eins erhöht
- Also Befehle werden von oben nach unten abgearbeitet

- Normalerweise wird die Instruktionsadresse nach jedem Befehl um eins erhöht
- Also Befehle werden von oben nach unten abgearbeitet
- Mit Sprüngen kann man die MIMA zwingen, zu definiertem Befehl zu springen und damit die Vorgehensreihenfolge zu beeinflussen

- Normalerweise wird die Instruktionsadresse nach jedem Befehl um eins erhöht
- Also Befehle werden von oben nach unten abgearbeitet
- Mit Sprüngen kann man die MIMA zwingen, zu definiertem Befehl zu springen und damit die Vorgehensreihenfolge zu beeinflussen
- *JMP adr* führt als nächsten Befehl den an Adresse *adr* aus.

- Normalerweise wird die Instruktionsadresse nach jedem Befehl um eins erhöht
- Also Befehle werden von oben nach unten abgearbeitet
- Mit Sprüngen kann man die MIMA zwingen, zu definiertem Befehl zu springen und damit die Vorgehensreihenfolge zu beeinflussen
- *JMP adr* führt als nächsten Befehl den an Adresse *adr* aus.
- *JMN adr* führt als nächsten Befehl den an Adresse *adr* aus, falls der Akku negativ ist.

- Normalerweise wird die Instruktionsadresse nach jedem Befehl um eins erhöht
- Also Befehle werden von oben nach unten abgearbeitet
- Mit Sprüngen kann man die MIMA zwingen, zu definiertem Befehl zu springen und damit die Vorgehensreihenfolge zu beeinflussen
- *JMP adr* führt als nächsten Befehl den an Adresse *adr* aus.
- *JMN adr* führt als nächsten Befehl den an Adresse *adr* aus, falls der Akku negativ ist.
 - Also wenn das erste Bit im Akku negativ ist.

- Normalerweise wird die Instruktionsadresse nach jedem Befehl um eins erhöht
- Also Befehle werden von oben nach unten abgearbeitet
- Mit Sprüngen kann man die MIMA zwingen, zu definiertem Befehl zu springen und damit die Vorgehensreihenfolge zu beeinflussen
- *JMP adr* führt als nächsten Befehl den an Adresse *adr* aus.
- *JMN adr* führt als nächsten Befehl den an Adresse *adr* aus, falls der Akku negativ ist.
 - Also wenn das erste Bit im Akku negativ ist.
 - Wenn vorher ein *EQL* erfolgreich verglichen hat, wird also gesprungen.

- Normalerweise wird die Instruktionsadresse nach jedem Befehl um eins erhöht
- Also Befehle werden von oben nach unten abgearbeitet
- Mit Sprüngen kann man die MIMA zwingen, zu definiertem Befehl zu springen und damit die Vorgehensreihenfolge zu beeinflussen
- *JMP adr* führt als nächsten Befehl den an Adresse *adr* aus.
- *JMN adr* führt als nächsten Befehl den an Adresse *adr* aus, falls der Akku negativ ist.
 - Also wenn das erste Bit im Akku negativ ist.
 - Wenn vorher ein *EQL* erfolgreich verglichen hat, wird also gesprungen.
 - Wenn der Akku positiv ist, werden die Befehle nach *JMN* normal weiter abgearbeitet.

MIMA Befehle: Springen

| Befehlssyntax | Bedeutung |
|----------------|--|
| <i>EQL adr</i> | Setze Akku auf $11 \dots 11$, falls Wert bei <i>adr</i> gleich Akku-Wert, setze Akku auf $00 \dots 00$ sonst. |
| <i>JMP adr</i> | Springe zu Befehlsadresse <i>adr</i> |
| <i>JMN adr</i> | Springe zu Befehlsadresse <i>adr</i> , falls Akku negativ (also erstes Bit = 1), sonst fahre normal fort. |

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben

MIMA Befehle: Springen

| Befehlssyntax | Bedeutung |
|----------------|--|
| <i>EQL adr</i> | Setze Akku auf $11 \dots 11$, falls Wert bei <i>adr</i> gleich Akku-Wert, setze Akku auf $00 \dots 00$ sonst. |
| <i>JMP adr</i> | Springe zu Befehlsadresse <i>adr</i> |
| <i>JMN adr</i> | Springe zu Befehlsadresse <i>adr</i> , falls Akku negativ (also erstes Bit = 1), sonst fahre normal fort. |

Beispielprogramm mit initialem Speicherabbild

| | | | |
|-------------------|-------------------|---------|------|
| LDC 5 | : | | |
| a_1 : JMN a_2 | NOT | Adresse | Wert |
| EQL 1 | a_2 : JMP a_3 | 1 | 5 |
| JMN a_1 | NOT | | |
| : | a_3 : HALT | | |

MIMA-Programm schreiben

Schreibe ein MIMA-Programm:

- Eingabe: Adresse a_1 einer positiven Zahl x .
- Ausgabe: Speichert $3 \cdot x$ in a_1 .

MIMA-Programm schreiben

Schreibe ein MIMA-Programm:

- Eingabe: Adresse a_1 einer positiven Zahl x .
- Ausgabe: Speichert $3 \cdot x$ in a_1 .

Lösung:

LDV a_1

ADD a_1

ADD a_1

STV a_1

HALT

MIMA-Programm schreiben

Schreibe ein MIMA-Programm:

- Eingabe: Adresse a_1 einer positiven Zahl x .
- Ausgabe: Speichert $x \bmod 2$ in a_1 .

MIMA-Programm schreiben

Schreibe ein MIMA-Programm:

- Eingabe: Adresse a_1 einer positiven Zahl x .
- Ausgabe: Speichert $x \bmod 2$ in a_1 .

Lösung:

```
LDC 1    // 0000000000000000000000000001
```

```
AND  $a_1$ 
```

```
STV  $a_1$ 
```

```
HALT
```


MIMA-Programm schreiben

Schreibe ein MIMA-Programm:

- Eingabe: Adresse a_1 einer positiven Zahl x .
- Ausgabe: Speichert $x \mathbf{div} 2$ in a_1 .

MIMA-Programm schreiben

Schreibe ein MIMA-Programm:

- Eingabe: Adresse a_1 einer positiven Zahl x .
- Ausgabe: Speichert $x \mathbf{div} 2$ in a_1 .

Lösung:

LDC 1

NOT

AND a_1 // Setze "rechtestes" Bit auf 0

RAR

STV a_1

HALT

Grundbegriffe der Informatik

Lukas Bach, lu-
kas.bach@student.kit.edu

Zum Übungsblatt

MIMA

Maschinenbefehle

Aufgaben



That's all Folks!