

Grundbegriffe der Informatik

Tutorium 38

Speicher, MIMA

Patrick Fetzner, uxkln@student.kit.edu | 22.11.2018



Hinweise

Speicher

MIMA

Maschinenbefehle

Aufgaben

- Reguläre Ausdrücke:
- Nur $\{, \}, (,), *, \cup$ und \cdot dürfen verwendet werden
- $\{a\} \cdot \{b\} = \{ab\}$
- Unterschied zwischen $\{a\} * \cup \{b\}^*$ und $\{a\} * \cdot \{b\}^*$
- Keine äußeren Klammern
- Berechnungen mit Num, Repr haben keinen Rechenweg verlangt
- $|\varepsilon| < 8$

Patrick Fetzer,
urkln@student.kit.edu

Hinweise

Speicher

MIMA

Maschinenbefehle

Aufgaben

- Ein **Bit** ist Zeichen aus $A = \{0, 1\}$
- Ein **Byte** ist ein Wort aus acht Bits
- Abkürzungen
 - Für Bit: `bit`
 - Für Byte: `B`

Hinweise

Speicher

MIMA

Maschinenbefehle

Aufgaben

Dezimal

10^{-3}	10^{-6}	10^{-9}	10^{-12}	10^{-15}	10^{-18}
1000^{-1}	1000^{-2}	1000^{-3}	1000^{-4}	1000^{-5}	1000^{-6}
milli	mikro	nano	pico	femto	atto
m	μ	n	p	f	a
10^3	10^6	10^9	10^{12}	10^{15}	10^{18}
1000^1	1000^2	1000^3	1000^4	1000^5	1000^6
kilo	mega	giga	tera	peta	exa
k	M	G	T	P	E

Binär

2^{10}	2^{20}	2^{30}	2^{40}	2^{50}	2^{60}
1024^1	1024^2	1024^3	1024^4	1024^5	1024^6
kibi	mebi	gibi	tebi	pebi	exbi
Ki	Mi	Gi	Ti	Pi	Ei

Gesamtzustand eines Speichers

Hinweise

Speicher

MIMA

Maschinenbefehle

Aufgaben

Zu jedem Zeitpunkt ist

- für jede **Adresse** festgelegt, welcher **Wert** dort ist
- beides meist Bitfolgen

Vorstellung: Tabelle mit zwei Spalten

Adresse	Wert
Adresse 1	Wert 1
Adresse 2	Wert 2
Adresse 3	Wert 3
...	...
Adresse n	Wert n

Definition des Speicherzustandes

Sei Adr die Menge aller Adressen und Val die Menge aller Werte.
Dann ist

$$m : Adr \rightarrow Val$$

der aktuelle Zustand des Speichers. Dabei ist $m(a)$ der aktuelle Wert an der Adresse a .

Mem

Menge aller möglichen Speicherzustände, also Menge aller Abbildungen von *Adr* nach *Val*

$$Mem := Val^{Adr}$$

Anmerkung: Für zwei Mengen A, B gilt: $A^B := \{f : B \rightarrow A\}$.

memread

$$memread : Mem \times Adr \rightarrow Val \text{ mit } (m, a) \mapsto m(a)$$

memwrite

$$memwrite : Mem \times Adr \times Val \rightarrow Mem \text{ mit } (m, a, v) \mapsto m'$$

Für m' wird folgendes gefordert:

$$m(a') := \begin{cases} v & \text{falls } a' = a \\ m(a') & \text{falls } a' \neq a \end{cases}$$

Eigenschaften von *memread* und *memwrite*

Hinweise

Speicher

MIMA

Maschinenbefehle

Aufgaben

Eigenschaften (“Invarianten”)

- $\text{memread}(\text{memwrite}(m, a, v), a) = v$ (Also: An a einen Wert v zu schreiben und danach bei a zu lesen gibt den Wert v zurück \Rightarrow Konsistente Datenhaltung)
- $\text{memread}(\text{memwrite}(m, a', v'), a) = \text{memread}(m, a)$ (Also: Auslesen einer Speicherstelle ist unabhängig davon, was vorher an eine andere Adresse geschrieben wurde \Rightarrow Unabhängige Datenhaltung)

Aufgaben

Aktueller Speicherzustand:

Adresse	Wert
00000	01110
00001	00100
00010	00111
00011	00000
...	...

Was ist?

■ `memread(memwrite(m, memread(m, 00011), 01010), 00000)`

→ 01010

Hinweise

Speicher

MIMA

Maschinenbefehle

Aufgaben

Patrick Fetzer,
urkln@student.kit.edu

Hinweise

- Theoretischer, idealisierter Prozessor

Speicher

- Funktioniert wie ein echter Prozessor, ist aber simpler

MIMA

- Nah an Technischer Informatik

Maschinenbefehle

Grundaufbau:

Aufgaben

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

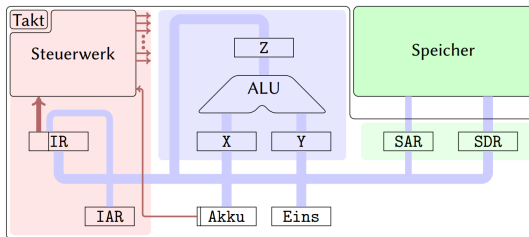
Hinweise

Speicher

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

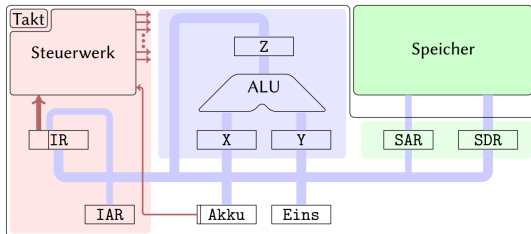
Hinweise

Speicher

MIMA

Maschinenbefehle

Aufgaben



Akku und Eins

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

- Patrick Fetzner,
uxkln@student.kit.edu

Aufbau der MIMA: ALU

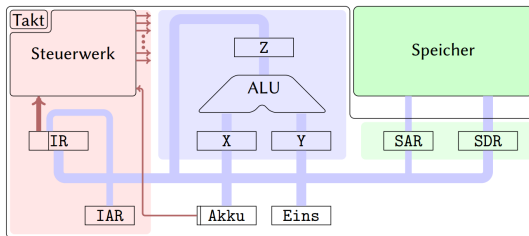
Hinweise

Speicher

MIMA

Maschinenbefehle

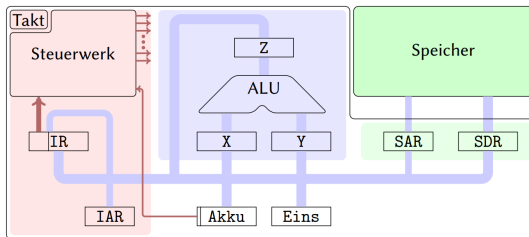
Aufgaben



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.



Busse

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

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

- Befehle (eigentlich Bitfolge) schreiben wir als Befehlsname und Adresse
 - $00100000000000000000101010 \equiv STV\ 42$
- $X \leftarrow Y \equiv$ "Der Variable X wird der Wert Y zugewiesen"

MIMA Befehle

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

Befehlssyntax	Formel	Bedeutung
<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>	$M(adr) \leftarrow Akku$	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.

Hinweise

Speicher

MIMA

Maschinenbefehle

Aufgaben

Hinweise

Eine MIMA-Maschine beherrscht folgende Maschinenbefehle:

Speicher

MIMA

Maschinenbefehle

Aufgaben

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
<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.

Hinweise

Speicher

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
- 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).

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.

Beispielprogramm mit initialem Speicherabbild

LDIV 4
STV 5
LDIV 5
STIV 4
HALT

Adresse	Wert
4	6
5	0
6	7
7	2

- 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.

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>	$Akku "OP" M(adr)$	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

Hinweise

Speicher

MIMA

Maschinenbefehle

Aufgaben

- *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 *addr*.
 - 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.

Beispielprogramm mit initialem Speicherabbild

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

Hinweise

Speicher

MIMA

Maschinenbefehle

Aufgaben

- 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 definierten Befehlen 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

Hinweise

Speicher

MIMA

Maschinenbefehle

Aufgaben

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 .

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 .

Lösung:

```
LDC 1 // 00000000000000000000000001
```

```
AND  $a_1$ 
```

```
STV  $a_1$ 
```

```
HALT
```

MIMA-Programm schreiben

Schreibe ein MIMA-Programm:

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

Lösung:

LDC 1

NOT

AND a_1 // Setze "rechtestes" Bit auf 0

RAR

STV a_1

HALT

Grundbegriffe der Informatik

Patrick Fetzer,
uxkln@student.kit.edu

Hinweise

Speicher

MIMA

Maschinenbefehle

Aufgaben



That's all Folks!