Karlsruhe Institute of Technology
Institut für Anthropomatik und Robotik (IAR)

Rechnerorganisation im WS 2017/18

# Musterlösungen zum 3. Übungsblatt

Prof. Dr. Wolfgang Karl
Haid-und-Neu-Str. 7

Dr.-Ing. Ömer Terlemez
Adenauerring 2, Geb. 50.20

Email: ti@ira.uka.de
Web: http://ti.ira.uka.de

## Lösung 1 (3 Punkte)

```
1. - 6. Hol- und Ausfuehrungsphase wie bekannt
7. Akku -> X, Akku -> Y
8. ALU: Addition (c2 = 0, c1 = 0, c0 = 1)
9. Z -> Y
10. ALU: Addition (c2 = 0, c1 = 0, c0 = 1)
11. Z -> Akku
```

## Lösung 2 (6 Punkte)

Bei den Mikroprogrammen handelt es sich um folgende MIMA-Befehle:

1. EQL — 2 P.
2. JMP — 1 P.
3. LDV — 1 P.
4. LDC — 1 P.
5. STV — 1 P.

## Lösung 3 (4 Punkte)

Die Befehlssequenz bildet das Einerkomplement aus dem Inhalt an Speicherstelle `0x2014` und speichert es an Speicherstelle `0x2015`. Danach wird das Zweierkomplement gebildet (+1) und an der Speicherstelle `0x2016` gespeichert.

Kodierte Befehlsfolge:

```
0x102014 (0001 0000 0010 0000 0001 0100)
0xF10000 (1111 0001 0000 0000 0000 0000)
0x202015 (0010 0000 0010 0000 0001 0101)
```

Lösung 4 (8 Punkte)

1. Konstanten werden geladen, indem der Inhalt des Instruktionsregister in das Akku-Register kopiert wird. Beide Register haben eine Länge von 24 Bit (wie auch der interne Datenbus). Bei einem anderen Opcode als $0x0$ könnten allerdings Konstanten wie 0, 1, 2, 3 nicht mehr geladen werden, da diese mit im Binärformat mit `0000` beginnen. 2 P.

2. Das Befehlsformat der MIMA sieht die Verwendung der vier höchstwertigen Bit als Opcode vor. Sind die vier höchstwertigen Bit alle gesetzt ($0xF$), so können über die vier nachfolgenden Bit weitere 16 Maschinenbefehle unterschieden werden. Insgesamt sind mit diesem Befehlsformat $15 + 16 = 31$ Maschinenbefehle realisierbar, von denen allerdings nur 13 Befehle implementiert sind. Nicht belegte Opcodes sind ungültig und versetzen die MIMA in einen Ausnahmezustand. 2 P.

3. Registernamen und Bedeutung: 2 P.

   - `IAR` (InstruktionsAdressRegister):
     Speichert die Adresse des aktuell auszuführenden Befehlsworts zu Beginn der Lese-Phase *(fetch phase)*. Nach der Lese-Phase enthält es die Adresse des im nächsten Zyklus auszuführenden Maschinenbefehls.
   - `IR` (InstruktionsRegister):
     Speichert das aktuell auszuführende Befehlswort.
   - `SAR` (SpeicherAdressRegister):
     Enthält die Speicheradresse auf die bei der nächsten Lese- ($R = 1$)/Schreibanfrage ($W = 1$) zugegriffen wird.
   - `SDR` (SpeicherDatenRegister):
     Enthält bei einer Schreiboperation ($W = 1$) das zu schreibenden Datenwort.
     Nach einer Leseoperation ($R = 1$) enthält es das gelesene Datenwort.

4. Das Speicheradress- (`SAR`) und das Speicherdatenregister (`SDR`) werden nur für Befehle benötigt, die auf den Speicher zugreifen. Daher werden sie zum Beispiel für den Befehl `LDC` oder `JMN` nicht benötigt. Das InstruktionsAdressRegister (`IAR`) wird beispielsweise für den `LDC`-Befehl nicht benötigt. 2 P.

Lösung 5 (6 Punkte)

Das Speicherlayout des Programms sieht folgendermaßen aus:

| | | | | | |
|---|---|---|---|---|---|
| 0x00000 | 0 0 0 0 0 3 | OP1 | DS | 3 | ; Zahl 1 |
| 0x00001 | 0 0 0 0 0 4 | OP2 | DS | 4 | ; Zahl 2 |
| 0x00002 | 0 0 0 0 0 0 | RES | DS | 0 | ; Ergebnis |
| 0x00003 | 0 0 0 0 0 1 | EINS | DS | 1 | ; Konstante 1 |
| 0x00004 | 0 0 0 0 0 3 | DREI | DS | 3 | ; Konstante 3 |
| 0x00005 | 0 0 0 0 0 4 | VIER | DS | 4 | ; Konstante 4 |
| | ⋮ | | | | |
| 0x00100 | 1 0 0 0 0 0 | START | LDV | OP1 | ; Lade OP1 |
| 0x00101 | 3 0 0 0 0 1 | | ADD | OP2 | ; Addiere OP2 |
| 0x00102 | 2 0 0 0 0 2 | | STV | RES | ; Ergebnis → RES |
| 0x00103 | 4 0 0 0 0 3 | | AND | EINS | ; Letztes Bit: RES & EINS |
| 0x00104 | 7 0 0 0 0 2 | | EQL | EINS | ; AKKU = -1 falls RES ungerade |
| 0x00105 | 9 0 0 1 0 A | | JMN | 0x10A | ; Zahl ungerade? → weiter bei 0x10A |
| 0x00106 | 1 0 0 0 0 2 | | LDV | RES | ; Zahl gerade: Lade RES |
| 0x00107 | 3 0 0 0 0 4 | | ADD | DREI | ; Addiere 3 |
| 0x00108 | 2 0 0 0 0 2 | | STV | RES | ; Speichere Ergebnis in RES |
| 0x00109 | F 0 0 0 0 0 | | HALT | | ; Ausführung beenden |
| 0x0010A | 1 0 0 0 0 2 | | LDV | RES | ; Ungerade Zahl: Lade RES |
| 0x0010B | 3 0 0 0 0 5 | | ADD | VIER | ; Addiere 4 |
| 0x0010C | 2 0 0 0 0 2 | | STV | RES | ; Speichere Ergebnis in RES |
| 0x0010D | F 0 0 0 0 0 | | HALT | | ; Ausführung beenden |