

Rechnerorganisation im WS 2020/21

### 3. Übungsblatt

Abgabetermin: 07. Dezember, 13:15 Uhr

Prof. Dr. Jörg Henkel  
Dr.-Ing. Lars Bauer  
Roman Lehmann, M. Sc.  
Haid-und-Neu-Str. 7,  
Geb. 07.21 (Technologiefabrik)  
Email: [roman.lehmann@kit.edu](mailto:roman.lehmann@kit.edu)

#### Aufgabe 1

(10 Punkte)

Implementieren Sie die Caesar-Verschlüsselung (<http://de.wikipedia.org/wiki/Caesar-Verschl%C3%BCsslung>). Im Ilias finden Sie hierzu das Programmskelett `caesar.c`.

Es sollen nur Groß- und Kleinbuchstaben (ohne Umlaute) kodiert und alle anderen Zeichen durch die Kodierung nicht verändert werden.

1. Implementieren Sie den Rumpf der Funktion `stringLength(char* text)`, die die Länge des nullterminierten Strings `text` zurückgeben soll. Bereits bestehende Funktionen wie `strlen()` dürfen hierfür nicht benutzt werden. 2 P.

Bsp.: `stringLength("hallo") == 5`

2. Implementieren Sie den Rumpf der Funktion `rotateChar(char c, int n)`, die `c` um `n` Stellen rotiert, falls `c` ein Buchstabe ist. 3 P.

Bsp.: `rotateChar('A', 1) == 'B'` und `rotateChar('!', 3) == '!'`

3. Implementieren Sie nun die Funktionen zur Ver- und Entschlüsselung `encrypt(char* text, int key)` und `decrypt(char* crypted, int key)`. 4 P.

Sie können die bereits implementierte Funktion `createBuffer` verwenden, um einen Puffer der benötigten Größe für den Rückgabewert zu allokalieren.

Testen Sie Ihre Implementierung ausgiebig!

4. Geben Sie Ihren Namen verschlüsselt mit Ihrer Matrikelnummer an. 1 P.

Bsp.: `encrypt("Peter Müller", 1234567) == "Yncna Vüuuna"`

Es gelten folgende Regeln für die Abgabe der Programmieraufgabe:

- keine Gruppenabgabe
- benennen Sie die Programmdatei nicht um
- erstellen Sie ein Zip-Archiv zusammen mit der restlichen Abgabe und laden Sie es ins Ilias hoch
- nicht übersetzbare/ausführbare Programme werden mit 0 Punkten bewertet
- für nicht ausgefüllte Header in der Programmdatei (Name, Matrikelnummer, ...) werden 2 Punkte abgezogen

Aufgabe 2

(8 Punkte)

1. Welche Ausgabe hat der Author der folgenden Zeilen C-Code vermutlich als Ausgabe erwartet? Übersetzen Sie das Programm, notieren Sie die Ausgabe und erklären Sie, warum die Ausgabe anders lautet. 3 P.

```
#include <stdio.h>
int main(int argc, char* argv[])
{
    printf("%.10f\n", 17541.233f); // %.10f gibt 10 Nachkommastellen aus
    return 0;
}
```

2. Erläutern Sie die Ausgabe des folgenden C-Programms. Weicht die Ausgabe von Ihrer Erwartung ab? 2 P.

```
#include <stdio.h>

int* f()
{
    int i = 5;
    return &i;
}

void g()
{
    int j = 25;
}

int main()
{
    int* x = f();
    g();
    printf("x = %d\n", *x);
    return 0;
}
```

3. Das folgende Programm, soll überprüfen, ob das Array a sortiert ist. Welcher Fehler steckt im Programm, wie äußert er sich und wie kann er korrigiert werden?

3 P.
------

```
#include <stdio.h>

int main(int argc, char* argv[])
{
    const int length = 10;
    const int a[10] = {1, 4, 7, 10, 11, 14, 18, 21, 25, 28};
    unsigned int i;
    /* Prüfe, ob das Array sortiert ist */
    for (i = length - 1; i >= 0; --i) {
        if (a[i-1] >= a[i]) {
            printf("Array ist nicht sortiert!\n");
            return 0;
        }
    }
    printf("Array ist sortiert!\n");
    return 0;
}
```

### Aufgabe 3

(3 Punkte)

Die MIMA-Architektur soll um einen weiteren Maschinenbefehl erweitert werden. Der TRI-Befehl soll den Inhalt des Akku-Registers mit 3 multiplizieren und das Ergebnis wieder im Akku speichern:

TRI :  $3 * \text{Akku} \rightarrow \text{Akku}$

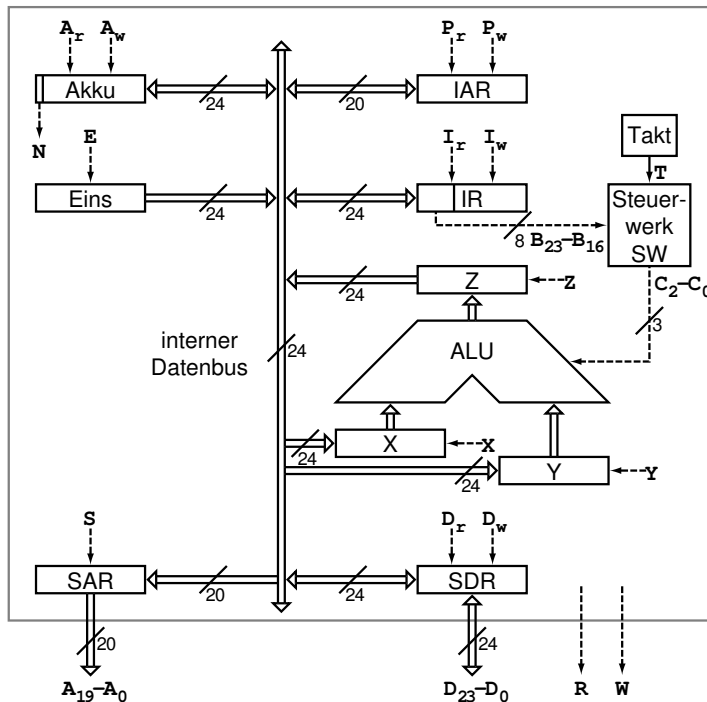
Schreiben Sie ein Mikroprogramm, das die Ausführungsphase des TRI-Befehls realisiert. Geben Sie dieses Mikroprogramm in Register-Transfer-Schreibweise an.

Aufgabe 4

(8 Punkte)

Beantworten Sie folgende Fragen zur MIMA-Architektur (siehe Beiblatt).

1. Wieso ist es wichtig, für den LDC-Befehl den Opcode 0x0 zu wählen? 2 P.  
Welche praktischen Probleme würde die Verwendung eines anderen Opcodes für LDC nach sich ziehen?
2. Wie viele Maschinenbefehle implementiert die MIMA in der Ihnen bekannten Version? 2 P.  
Wie viele verschiedene Maschinenbefehle könnte die MIMA aufgrund ihrer Architektur theoretisch in ihrem Befehlssatz zur Verfügung stellen?  
(Tipp: Betrachten Sie den Aufbau des MIMA-Befehlsformats!)
3. Wofür stehen die Abkürzungen der folgenden MIMA-Register und welche Aufgabe haben diese Register: 2 P.
  - IAR
  - IR
  - SAR
  - SDR
4. Drei der vier zuvor genannten Register werden nach der Lesephase (*fetch phase*) nicht zur Ausführung aller Maschinenbefehle benötigt. Welche Register sind dies und welcher Befehl benötigt diese beispielsweise nicht? 2 P.

**Architektur der MIMA****Register**

Akku: Akkumulator  
 X: 1. ALU Operand  
 Y: 2. ALU Operand  
 Z: ALU Ergebnis  
 Eins: Konstante 1  
 IAR: Instruktionsadreßregister  
 IR: Instruktionsregister  
 SAR: Speicheradreßregister  
 SDR: Speicherdatenregister

**Steuersignale vom SW**

– für den internen Datenbus

$A_r$ : Akku liest  
 $A_w$ : Akku schreibt  
 $x$ : X-Register liest  
 $y$ : Y-Register liest  
 $z$ : Z-Register schreibt  
 $E$ : Eins-Register schreibt  
 $P_r$ : IAR liest  
 $P_w$ : IAR schreibt  
 $I_r$ : IR liest  
 $I_w$ : IR schreibt  
 $D_r$ : SDR liest  
 $D_w$ : SDR schreibt  
 $s$ : SAR liest

– für die ALU

$c_2-c_0$ : Operation auswählen

– für den Speicher

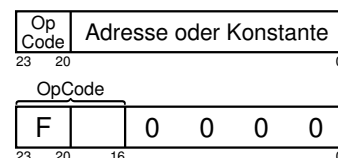
$R$ : Leseanforderung  
 $W$ : Schreibsanforderung

**Meldesignale zum SW**

$T$ : Takteingang  
 $N$ : Vorzeichen des Akku  
 $B_{23}-B_{16}$ : OpCode-Feld im IR

$c_2 c_1 c_0$	ALU Operation
0 0 0	tue nichts ( d.h. $Z \rightarrow Z$ )
0 0 1	$X + Y \rightarrow Z$
0 1 0	rotiere $X$ nach rechts $\rightarrow Z$
0 1 1	$X \text{ AND } Y \rightarrow Z$
1 0 0	$X \text{ OR } Y \rightarrow Z$
1 0 1	$X \text{ XOR } Y \rightarrow Z$
1 1 0	Eins-Komplement von $X \rightarrow Z$
1 1 1	falls $X = Y$ , $-1 \rightarrow Z$ , sonst $0 \rightarrow Z$

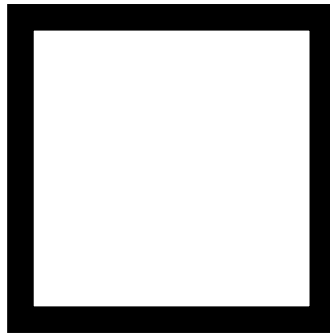
OpCode	Mnemonic	Beschreibung
0	LDC c	$c \rightarrow \text{Akku}$
1	LDV a	$\langle a \rangle \rightarrow \text{Akku}$
2	STV a	$\text{Akku} \rightarrow \langle a \rangle$
3	ADD a	$\text{Akku} + \langle a \rangle \rightarrow \text{Akku}$
4	AND a	$\text{Akku AND } \langle a \rangle \rightarrow \text{Akku}$
5	OR a	$\text{Akku OR } \langle a \rangle \rightarrow \text{Akku}$
6	XOR a	$\text{Akku XOR } \langle a \rangle \rightarrow \text{Akku}$
7	EQL a	falls $\text{Akku} = \langle a \rangle$ : $-1 \rightarrow \text{Akku}$ sonst: $0 \rightarrow \text{Akku}$
8	JMP a	$a \rightarrow \text{IAR}$
9	JMN a	falls $\text{Akku} < 0$ : $a \rightarrow \text{IAR}$
F0	HALT	stoppt die MIMA
F1	NOT	bilde Eins-Komplement von Akku $\rightarrow \text{Akku}$
F2	RAR	rotiere Akku eins nach rechts $\rightarrow \text{Akku}$

**Befehlsformate**

# Vorlesung Rechnerorganisation Wintersemester 2020/21

## - Übungsblatt 3 -

Tutoriumsnummer



Name, Vorname: \_\_\_\_\_

Matrikelnummer: \_\_\_\_\_

Studiengang: \_\_\_\_\_

Name des Tutors: \_\_\_\_\_

/29 Punkte