Zürcher Hochschule
für Angewandte Wissenschaften

zh aw School of Engineering

Modul Informatik-II

Kurs Informatik-3: Aufgabenserie-3b

www.engineering.zhaw.ch/de/engineering/studium/bachelor/informatik/studium-zurich.html

Prof. Dr. Olaf Stern
Leiter Studiengang Informatik (Standort Zürich)
+41 58 934 48 51
olaf.stern@zhaw.ch

Aufgabenserie 3b

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering

Punkteskala

In der Aufgabe sind maximal 19 Punkte möglich – die Punkteskala wird dabei wie folgt angewandt:

18 - 19:	**4 Punkte**
15 - 17:	**3 Punkte**
11 - 14:	**2 Punkte**
8 - 10:	**1 Punkt**
übrige:	**0 Punkte**

Für diese Aufgabe haben Sie bis zum Ende Oktober Zeit!

Aufgabenserie 3b

Aufgabe „*Mini-Power-PC*“

Schreiben Sie ein Programm in einer beliebigen Programmiersprache / Umgebung, das den in der Vorlesung spezifizierten „*Mini-Power-PC*“ emuliert – d. h. alle Befehle des Befehlssatzes (siehe Anlage **Befehlssatz** „*Mini-Power-PC*“) **ausführt.**

a) Als **Eingabe** soll gelesen werden: (2 Punkte)

- Ein **beliebiges** mit dem **Befehlssatz** geschriebenes **Programm**, das ab **Speicher *100*** in den **Speicher** eingelesen wird (in **Maschinencode**, als **Binärprogramm**)
- Die **Parameter** des **Programms** (Eingabewerte) ab **Speicher *500***

  (Der **Op-Code** und die **Parameter** des Programms können als ***Binär-***, ***Dezimal-*** oder ***Hex-Zahlen*** eingelesen werden - Ihre Wahl)

Aufgabe „*Mini-Power-PC*“ (Forts.)

Schreiben Sie ein Programm in einer beliebigen Programmiersprache / Umgebung, das den in der Vorlesung spezifizierten „*Mini-Power-PC*“ emuliert – d. h. alle Befehle des Befehlssatzes (siehe Anlage **Befehlssatz „*Mini-Power-PC*“**) **ausführt.**

b) Als **Ausgabe** wird folgendes erwartet: (3 + 2 Punkte)

- Die **aktuellen Zustände** der **Register:**
  - ***Befehlszähler***, ***Befehlsregister***
  - ***Akkumulator***, ***Carry-Bit***
  - ***Reg-1***, ***Reg-2***, ***Reg-3***
  - **Optional**: Alle Werte auch als **Binär-Werte** (*16* Bit)
- Der **aktuelle decodierte Befehl** aus dem **Befehlsregister** (als **Mnemonics**) (1 Punkt)

Aufgabenserie 3b

Aufgabe „*Mini-Power-PC*“ (Forts.)

Schreiben Sie ein Programm in einer beliebigen Programmiersprache / Umgebung, das den in der Vorlesung spezifizierten „*Mini-Power-PC*“ emuliert – d. h. alle Befehle des Befehlssatzes (siehe Anlage **Befehlssatz „*Mini-Power-PC*“**) **ausführt.**

b) (Forts.) Als **Ausgabe** wird folgendes erwartet: (5 + 2 Punkte)

- …
- Der **aktuelle Zustand** des **Speichers:**
  - **5 Befehle vor** bis **10 Befehle nach** dem **aktuellen Befehl**
  - Der **Inhalt** der **Speicherzellen *500*** bis ***529*** (wortweise)
  - Optional: alle Werte auch als **Binär-Werte** (*16* Bit)
- Die **Anzahl** der **durchgeführten Befehle** (zum Programmstart *0*)

Aufgabenserie 3b

Zürcher Hochschule für Angewandte Wissenschaften
zh aw School of Engineering

Aufgabe „*Mini-Power-PC*“ (Forts.)

Schreiben Sie ein Programm in einer beliebigen Programmiersprache / Umgebung, das den in der Vorlesung spezifizierten „*Mini-Power-PC*“ emuliert – d. h. alle Befehle des Befehlssatzes (siehe Anlage **Befehlssatz** „*Mini-Power-PC*“) **ausführt.**

c) **Implementieren** Sie einen „***schnellen***“ und einen „***slow***“ ***Modus*** sowie optional einen „***Step-Modus***“: (2 + 2 Punkte)

Schneller Modus: Während des **Programmablaufs** erfolgt **keine Ausgabe** (keine Aktualisierung); diese erfolgt erst am **Programmende**.

Slow-Modus: Während des **Programmablaufs** wird nach **Bearbeitung eines jeden Befehls** die **Ausgabe aktualisiert**.

Step-Modus: Wie der ***Slow-Modus***, jedoch wird das **Programm** nach **Bearbeitung eines jeden Befehls unterbrochen** und wird erst nach einer **Bestätigung** durch den **User** (z. B. Drücken einer Taste) wieder **fortgesetzt**.

Aufgabenserie 3b

Aufgabe „*Mini-Power-PC*“ (Forts.)

Befehlssatz für das Prozessormodell: "Mini-Power-PC"

Maschinen-Code (Op-Code)	Mnemonics ("*Assembler*")	Kurzbeschreibung	Beschreibung
0 0 0 0 x x 0 1 <not used>	CLR *Rnr*	«*Rxx*» := 0	Lösche das Register «*Rxx*» (alle Bit auf 0 setzten) und das Carry-Flag (*00* bis *11* für: Akku, R-1, R-2 bzw. R-3).
0 0 0 0 x x 1 1 <not used>	ADD *Rnr*	*Akku* := *Akku* + «*Rxx*»	Addition zweier 16-Bit-Zahlen (Zahl im Akku und Zahl im Register «*Rxx*»; *00* bis *11* für Akku, R-1, R-2 bzw. R-3) im *2er*-Komplement; bei Überlauf wird das Carry-Flag gesetzt (= 1), sonst auf den Wert *0*.
1 <- - - - Zahl - - - - ->	ADDD *#Zahl*	*Akku* := *Akku* + *#Zahl*	Addition der 16-Bit-Zahl im Akku mit der 15-Bit-Zahl als direkten Operanden im *2er*-Komplement; bei Überlauf wird das Carry-Flag gesetzt (= 1), sonst auf den Wert *0*. Vor der Addition wird die 15-Bit-Zahl des Operanden auf 16 Bit erweitert (mit MSb des MSB auf *1* wenn negativ, sonst auf *0*).
0 0 0 0 0 0 0 1 not used	INC	*Akku* := *Akku* + *1*	Der Akku (16-Bit-Zahl im *2er*-Komplement) wird um den Wert *1* inkrementiert; bei Überlauf wird das Carry-Flag gesetzt (= 1), sonst auf den Wert 0.
0 0 0 0 0 1 0 0 not used	DEC	*Akku* := *Akku* - *1*	Der Akku (16-Bit-Zahl im *2er*-Komplement) wird um den Wert *1* dekrementiert; bei Überlauf wird das Carry-Flag gesetzt (= 1), sonst auf den Wert 0.
0 1 0 - x x <Adresse->	LWDD *Rnr* , *#Adr*	«*Rnr*» := Inhalt Speicher(*Adr*)	In das Register mit der Nummer *xx* (*00* bis *11* für Akku, R-1, R-2 bzw. R-3) wird der Inhalt der Speicherzellen *Adr* und *Adr* + 1 (1 Wort = 2 Byte) geladen. Mit 10 Bit können 1KiB Speicher adressiert werden.
0 1 1 - x x <Adresse->	SWDD *Rnr* , *#Adr*	Inhalt Speicher(*Adr*) := «*Rnr*»	In die über *Adr* und *Adr* + 1 adressierten Speicherzellen wird der Inhalt des Registers *xx* (*00* bis *11* für Akku, R-1, R-2 bzw. R-3) geschrieben. Mit 10 Bit können 1KiB Speicher adressiert werden.
0 0 0 0 0 1 0 1 not used	SRA	*Akku* := *Akku* / 2	Schieben *arithmetisch* nach rechts: der Inhalt des Akkus wird um eine Stelle nach rechts geschoben; der Inhalt vom LSb des LSB (das rechte Bit des Wortes) wird als Carry-Flag gesetzt. Dabei bleibt das MSb des MSB (Vorzeichenbit) und das 2. Bit des MSB erhalten.
0 0 0 0 1 0 0 0 not used	SLA	*Akku* := *Akku* x 2	Schieben *arithmetisch* nach links: der Inhalt des Akkus wird um eine Stelle nach links geschoben; der Inhalt vom 2. Bit des MSB (das 2. Bit des Wortes) wird als Carry-Flag gesetzt. Dabei bleibt das MSb des MSB (Vorzeichenbit) erhalten. In das LSb des LSB (das letzte Bit des Wortes) wird eine *0* geschrieben.
0 0 0 0 1 0 0 1 not used	SRL	*Akku* := *Akku* / 2	Schieben *logisch* nach rechts: der Inhalt des Akkus wird um eine Stelle nach rechts geschoben; der Inhalt vom LSb des LSB (das rechte Bit des Wortes) wird als Carry-Flag gesetzt. Das MSb des MSB (das 1. Bit des Wortes) wird auf *0* gesetzt.
0 0 0 0 1 1 0 0 not used	SLL	*Akku* := *Akku* x 2	Schieben *logisch* nach links: der Inhalt des Akkus wird um eine Stelle nach links geschoben; der Inhalt vom LSb des LSB (das rechte Bit des Wortes) wird mit *0* aufgefüllt, das MSb des MSB (das 1. Bit des Wortes) wird als Carry-Flag gesetzt.
0 0 0 0 x x 1 0 <not used>	AND *Rnr*	*Akku* := *Akku* AND «*Rxx*»	Akku und Register *xx* (*00* bis *11* für Akku, R-1, R-2 bzw R-3) werden bitweise logisch mit AND verknüpft.
0 0 0 0 x x 1 0 <not used>	OR *Rnr*	*Akku* := *Akku* OR «*Rxx*»	Akku und Register *xx* (*00* bis *11* für Akku, R-1, R-2 bzw R-3) werden bitweise logisch mit ODER verknüpft.
0 0 0 0 0 0 0 1 <not used>	NOT	*Akku* := NOT (*Akku*)	Alle Bit im Akku werden bitweise negiert.
0 0 0 1 x x 1 0 <not used>	BZ *Rnr*	Bedingter Sprung	Wenn der Akku *0* ist, verzweige an die durch das Register *xx* (*01* bis *11* für R-1, R-2 bzw. R-3) angegebene Speicheradresse; sonst wird der folgende Befehl normal fortgeführt.
0 0 0 1 x x 0 1 <not used>	BNZ *Rnr*	Bedingter Sprung	Wenn der Akku ≠ *0* ist, verzweige an die durch das Register *xx* (*01* bis *11* für R-1, R-2 bzw. R-3) angegebene Speicheradresse; sonst wird der folgende Befehl normal fortgeführt.
0 0 0 1 x x 1 1 <not used>	BC *Rnr*	Bedingter Sprung, Carry	Wenn das Carry-Flag gesetzt ist (= 1), verzweige an die durch das Register *xx* (*01* bis *11* für R-1, R-2 bzw. R-3) angegebene Speicheradresse, sonst wird der folgende Befehl normal fortgeführt.
0 0 0 1 x x 0 0 <not used>	B *Rnr*	Unbedingter Sprung	Verzweige an die durch das Register *xx* (*01* bis *11* für R-1, R-2 bzw. R-3) angegebene Speicheradresse.
0 0 1 1 0 - <Adresse->	BZD *#Adr*	Bedingter Sprung, direkt	Wenn der Akku 0 ist, verzweige an die durch den Operanden angegebene Speicheradresse; sonst wird das Programm mit dem folgenden Befehl forgesetzt. Mit 10 Bit können 1KiB Speicher adressiert werden.
0 0 1 0 1 - <Adresse->	BNZD *#Adr*	Bedingter Sprung, direkt	Wenn der Akku ≠ 0 ist, verzweige an die durch den Operanden angegebene Speicheradresse; sonst wird das Programm mit dem folgenden Befehl forgesetzt. Mit 10 Bit können 1KiB Speicher adressiert werden.
0 0 1 1 1 - <Adresse->	BCD *#Adr*	Bedingter Sprung, direkt	Wenn das Carry-Flag gesetzt sie, verzweige an die durch den Operanden angegebene Speicheradresse; sonst wird das Programm mit dem folgenden Befehl forgesetzt. Mit 10 Bit können 1KiB Speicher adressiert werden.
0 0 1 0 0 - <Adresse->	BD *#Adr*	Unbedingter Sprung, direkt	Verzweige an die durch den Operanden angegebene Speicheradresse. Mit 10 Bit können 1KiB Speicher adressiert werden.
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	STOP	Ende des Programms	"Hilfsbefehl" für die Realisierung: Tatsächlich arbeitet der Prozessor ja immer weiter; er muss sogesehen in eine Schleife eintreten, in der er keine Werte verändert und die er durch einen Interrupt (z. B. für einen Programmneustart) wieder verlassen kann.