Technische Informatik - Übung

Rechnerarchitektur

1. Aufgabe

Vervollständigen Sie die folgende Code-Tabelle!

dezimal	dual	oktal	hexadezimal	ASCII-Zeichen
42				
	01101100			
		35		
			3D	
				'Z'

Technische Informatik - Übung

Rechnerarchitektur

2. Aufgabe

Wandeln Sie die Ganzzahl **`-12`** in eine 8-Bit-Binärzahl (Zweierkomplement) um. Addieren Sie anschließend die im Zweierkomplement codierten Zahlen +12 und -12 !

Technische Informatik - Übung

Rechnerarchitektur

3. Aufgabe

1) Addieren Sie die Dualzahlen **0100 1100**$_{dual}$ und **0111 0101**$_{dual}$!

2) Addieren Sie die Hexadezimalzahlen **A37B**$_{H}$ und **5B77**$_{H}$!

Technische Informatik - Übung

Rechnerarchitektur

4. Aufgabe

1) Die Binärzahl

```
11000000110100000000000000000000
```

codiert eine Gleitkommazahl nach IEEE-754. Wandeln Sie diese Zahl in Dezimaldarstellung um!

2) Wandeln Sie die Dezimalzahl ***0,15625*** in eine binäre 32-Bit-Gleitkommazahl um!

Technische Informatik - Übung

Rechnerarchitektur

5. Aufgabe

Ein Prozessor verfügt neben dem Program-Counter ***PC*** und dem Stack-Pointer ***SP*** über die General-Purpose-Register ***R1*** und ***R2***. Die Instruction-Set-Architecture (ISA) umfasst u.a. auch die folgenden Maschinenbefehle:

`NOP`		no operation
`PUSH`	`Rx`	legt `Rx` auf den Stack
`POP`	`Rx`	holt `Rx` vom Stack
`MOV`	`Rx,$addr`	kopiert Inhalt der Speicherstelle `addr` in das Register `Rx`
`MOV`	`Rx,#const`	kopiert Konstante `const` in das Register `Rx`
`MOV`	`$addr,Rx`	kopiert Register `Rx` an die Speicherstelle `addr`
`ADD`	`Rz,Rx,Ry`	addiert `Rx` und `Ry`, schreibt Ergebnis nach `Rz`
`CMP`	`Rx,Ry`	vergleicht Register `Rx` mit `Ry` und setzt ggf. Compare-Flag "not equal"
`JNE`	`addr`	Prüft Compare-Flag und springt an die Programm-Adresse `addr`, falls Flag "not equal" gesetzt ist
`JMP`	`addr`	Springt unbedingt an die Programm-Adresse `addr`

Der angeschlossene Datenspeicher wird mit 8 Bit adressiert und hat eine Größe von 256 Byte. Der 64KByte große Programmspeicher wird mit 16 Bit adressiert. Der vom Compiler erzeugte Programm-Code enthält 4 Byte für jeden auszuführenden Befehl, einschließlich der bis zu 3 Byte großen Operanden.

Die folgenden beiden, in der Programmiersprache C geschriebenen Zeilen werden vom Compiler so übersetzt, dass für die Variablen x bzw. y die Speicherstellen mit den Adressen 02_{hex} bzw. 03_{hex} vorgesehen werden.

```
1  BYTE x=0x12;
2  BYTE y=0x34;
```

Die Programmzeilen 3 bis 10 werden ebenfalls übersetzt, die nachfolgende Tabelle zeigt in der Spalte "Programm-Speicher" den erzeugten Maschinencode.

```
3  if( x == 0x10 )
4  {
5     y = y + 7;
6  }
7  else
8  {
9     y = 0x80;
10 }
```

Technische Informatik - Übung

Rechnerarchitektur

Programm-Speicher

Adresse	Instruktion
8000	---
8004	MOV R1,$
8008	MOV ,#10
800C	CMP R1,R2
8010	J #
8014	MOV R1,$03
8018	MOV R2,#07
801C	ADD R1,R1,R2
8020	MOV $03, R1
8024	J #
8028	MOV R1, #
802C	MOV $,R1
8030	NOP

Register **Datenspeicher**

Programm-schritt	PC	R1	R2	00	01	02	03
0	8004	00	00	00	00	12	34
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							

a) Vervollständigen Sie an den grau unterlegten Stellen (Spalte "Instruktion") den Assembler-Code!

b) In der Tabellenspalten "Register" und "Datenspeicher" sind die Inhalte aufgelistet, die sich nach Ausführung der Instruktion ergeben. Vervollständigen Sie die Tabelle soweit möglich!

c) Geben Sie den Verlauf der Register und Datenspeicher für den Fall an, dass im Programm

```
BYTE x=0x12;
```

definiert wurde.

Technische Informatik - Übung

Rechnerarchitektur

6. Aufgabe

Ein Prozessor verfügt neben dem Program-Counter ***PC*** und dem Stack-Pointer ***SP*** über die General-Purpose-Register ***R1*** und ***R2***. Die Datenwortbreite betrage ebenso wie der Adressraum 8-Bit.
Die Instruction-Set-Architecture (ISA) umfasst u.a. auch die folgenden Maschinenbefehle:

`NOP`	no operation
`PUSH Rx`	legt `Rx` auf den Stack
`POP Rx`	holt `Rx` vom Stack
`MOV Rx,$addr`	kopiert Inhalt der Speicherstelle `addr` in das Register `Rx`
`MOV Rx,#const`	kopiert Konstante `const` in das Register `Rx`
`MOV $addr,Rx`	kopiert Register `Rx` an die Speicherstelle `addr`
`ADD Rz,Rx,Ry`	addiert `Rx` und `Ry`, schreibt Ergebnis nach `Rz`
`CMP Rx,Ry`	vergleicht Register `Rx` mit `Ry` und setzt ggf. Compare-Flag "not equal"
`JNE addr`	Prüft Compare-Flag und springt an die Programm-Adresse `addr`, falls Flag "not equal" gesetzt ist
`JMP addr`	Springt unbedingt an die Programm-Adresse `addr`

Auf diesem Prozessor wird nun eine Programmsequenz gemäß Tabelle ausgeführt. Bestimmen Sie die Inhalte der Register ***SP***, ***R1*** und ***R2*** nach jedem Programmschritt!
Welches Speicherabbild im RAM ergibt sich abschließend?

Programm

`---`
`MOV R1,#01`
`MOV R2,#02`
`PUSH R1`
`PUSH R2`
`ADD R1,R1,R2`
`MOV $F8,R1`
`MOV R2,#04`
`MOV $F9,R2`
`POP R2`
`POP R1`

Register

Programm-schritt	SP	R1	R2
0	FF	00	00
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			

RAM

Adresse	Inhalt
F8	
F9	
FA	
FB	
FC	
FD	
FE	
FF	