Informationssysteme
Semesterwoche 4

1. Beilage PC Architektur

Lesen Sie die Beilage PC Architektur und beantworten Sie anschliessend die nachstehenden Fragen.
Beantworten Sie folgende Fragen
1. Gibt es verschiedenen Busse? Wenn ja, beschreiben Sie die jeweilige Aufgabe in ein paar Stichworten.
2. Nennen Sie zwei Benchmarkverfahren.
3. Aus welchen Komponenten besteht eine CPU?
4. Welche Schnittstellen weist ein PC-Computer heute typischerweise auf?
Welche Schnittstellen wird es in Zukunft möglicherweise immer weniger geben?
5. Was versteht man unter Chipsatz?
6. Was versteht man unter North-Bridge?
7. * Welche vier Messdaten sind bei Festplatten relevant?
8. * Wozu dient eine FPU?
9. * Was versteht man unter BIOS? Welche Aufgaben erfüllt das BIOS?
10. Was versteht man unter cache Speicher? Was ist ein L1, was ein L2 cache?

2. MikroSim

Die Aufgaben stehen auf der letzten Seite. Schauen Sie sich zuerst kurz die Aufgaben an. Lesen Sie dann den ganzen Text durch. Lösen Sie dann die Aufgaben unter Zuhilfenahme des Simulators MikroSim.

Fetch-Decode-Execute Zyklus
Die Aufgabe einer CPU (Central Processing Unit) kann mit Pseudocode wie folgt beschrieben werden:

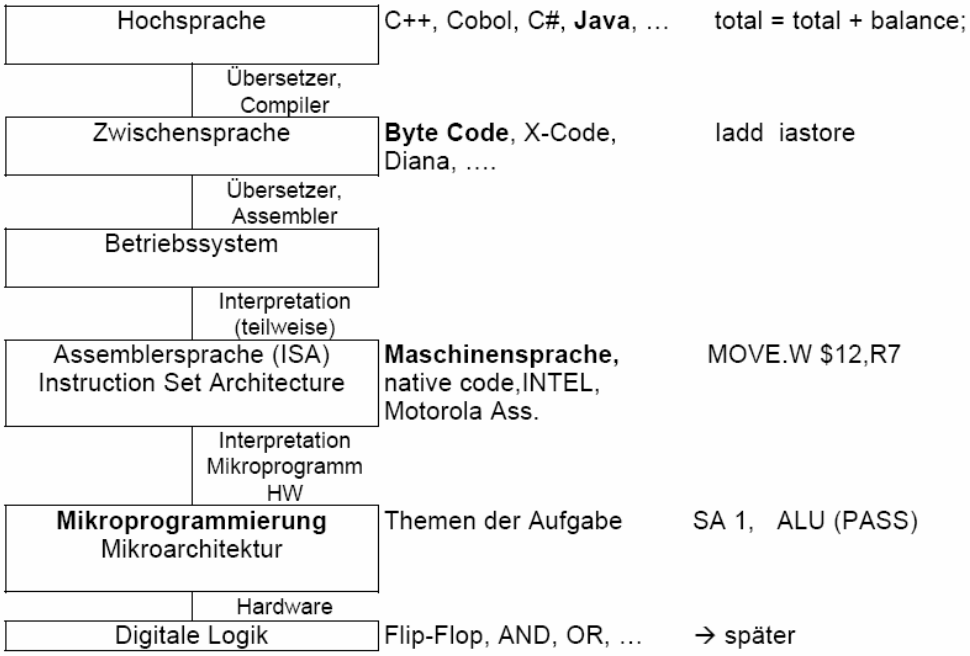
```
do forever
      1. Fetch next instruction from memory into instruction
      register
      2. Change program counter to point to next instruction
      3. Determine type of instruction just fetched
      4. If instructions uses word in memory, determine where it
      is.
      5. Fetch word, if needed, into CPU register
      6. Execute the instruction
reverof od
```

Erklären Sie den Zweck des Schritts 2.
→ Schritt 2 inkrementiert den Program Counter (PC). D.h. der PC zeigt nun auf den nächsten auszuführenden Befehl.

Was würde geschehen, wenn Schritt 2 weggelassen würde?
→ Die CPU würde ständig/unendlich denselben Befehl ausführen.

3. CPU Simulator

Der Fokus in dieser Aufgabe ist Mikroprogrammierung und Maschinensprache. Ein Schichtenmodell hilft, grosse Aufgaben in kleinere Teilaufgaben aufzuteilen.



→ Assemblerprogramm und Daten stehen im gleichen Speicher.
→ Das Mikroprogramm ist das Bindeglied zwischen Assemblerbefehlen und der Hardware.

Arbeitsspeicher	
Adresse	Inhalt
$00	LDA $10
$01	ADD $11
$02	STA $12
$03	00000000
$04	00000000
$05	00000000
$06	00000000
$07	00000000
$08	00000000
$09	00000000
$0A	00000000
$0B	00000000
$0C	00000000
$0D	00000000
$0E	00000000
$0F	00000000
$10	23.
$11	24.
$12	0.
$13	00000000
$14	00000000

```
START
SA 0
TI
SWITCH Opcode OF
CASE 0: LDA
SA 1
TO
ALU (PASS)
TA
SP 0
CASE 1: STA
SA 1
TW
SP 0
CASE 2: ADD
SA 1
TO
ALU (ADD)
TA
SP 0
CASE 3: AND
SA 1
TO
ALU (AND)
TA
SP 0
CASE 4: XOR
SA 1
TO
ALU (XOR)
TA
SP 0
```

	CASE 5: ROL ALU (ROL) TA SP 0 CASE 6: JNZ IF Z=0 SP 1 ELSE Z=1 SP 0 CASE 7: JMP SP 1 END SWITCH TP END

Assembler Befehle:
LDA mem ladet den Wert aus mem in den Akkumulator
STA mem speicher den Wert aus dem Akkumulator in mem
ADD mem addiert den Wert aus mem zum Akkumulator
AND führt AND Verknüpfung zwischen Akku und mem aus
XOR führt EXOR Verknüpfung zwischen Akku und mem aus
ROL rotiert das Bitmuster nach links
JNZ springt an die angegebene Adresse bei Nicht-Null im Zero-Flag
JMP springt bedingungslos an die angegebene Adresse

Mikroprogramm Befehle:
SA = Schalter Adressen Input
TO = Trigger Operanden Register
(Trigger = Auslöser, auslösendes Ereignis)
TA = Trigger Akkumulator
TI = Trigger Instruktionsregister
TP = Trigger Programmzähler
TW = Trigger Arbeitsspeicher
SP = Schalter PC Input (PC = Program Counter)
ALU = ALU-Funktion
ZFLAG = Zero-Flag Test
Flag = Flagge, die etwas signalisiert

Fragen und Antworten:
1. Was tut das Programm im Arbeitsspeicher?
➔ Wert aus Adresse $10 laden.
➔ Wert aus Adresse $11 dazu addieren.
➔ Resultat in Adresse $12 speichern.

Mikroprogramm	Erklärung
START	
SA 0	➔ Schalter A=0: Adresse $10 wird aus Program Counter ausgelesen.
TI	➔ Trigger TI teilt dem Instr.Register mit, dass ein Befehl auf dem Bus liegt.
SWITCH Opcode OF	
CASE 0: LDA	➔ Case 0, falls ein LDA Befehl vorliegt.
SA 1	➔ Schalter A=1: LDA $10 lädt den Wert (="23") aus der RAM-Adresse $10.
TO	➔ Trigger TO teilt dem Operationsregister "OP Reg" mit, dass ein Befehl/Wert auf dem Bus liegt.
ALU (PASS)	PASS teilt der ALU mit, dass sie den Wert nicht beachten muss.
TA	Trigger TA teilt dem Akkumulator mit, dass ein Wert bereit liegt.
SP 0	Schalter P=0: Nächster Befehl kommt von Program Counter.

CASE 1: STA	→ Case 1, falls ein STA Befehl vorliegt.
SA 1	
TW	
SP 0	
CASE 2: ADD	→ Case 2, falls ein ADD Befehl vorliegt.
SA 1	
TO	
ALU (ADD)	
TA	
SP 0	
CASE 3: AND	→ Case 3, falls ein AND Befehl vorliegt.
SA 1	
TO	
ALU (AND)	
TA	
SP 0	
CASE 4: XOR	→ Case 4, falls ein XOR Befehl vorliegt.
SA 1	
TO	
ALU (XOR)	
TA	
SP 0	
CASE 5: ROL	→ Case 5, falls ein ROL Befehl vorliegt.
ALU (ROL)	
TA	
SP 0	
CASE 6: JNZ	→ Case 6, falls ein JNZ Befehl vorliegt.
IF Z=0	
SP 1	
ELSE Z=1	
SP 0	
CASE 7: JMP	→ Case 7, falls ein JMP Befehl vorliegt.
SP 1	
END SWITCH	
TP	Trigger P: Program Counter wird inkrementiert.
END	

2. Erweitern Sie das Programm derart, dass es fortlaufend den Wert 2 addiert.

```
00    LDA $10
01    ADD $10
02    STA $11
03    JMP $01
04    00000000
05    00000000
06    00000000
07    00000000
08    00000000
09    00000000
10    2.
11    00000000
```

3. Erweitern Sie das Programm derart, dass es einmalig alle ungeraden Zahlen zwischen 3 und 17 erzeugt.
4. Studieren Sie den detaillierten Ablauf einzelner Mikrocodes.
5. Implementieren Sie ein Mikroprogramm, das statt der Addition eine Division ausführen kann.
* Welche spezielle Situation sollte bei (5) auch beachtet werden? Ergänzen Sie Ihr Programm entsprechend.
6. Wozu dient der Schalter "SA"?
→ Schalter A entscheidet, ob eine Adresse aus dem Instruktionsregister oder aus dem Program Counter gelesen wird.