Fachhochschule - Isny **26.11.2001**

Mikroprozessortechnik **Dozent: Dipl. -Inf Gerum**

Klausur 1 von 2

NAME:

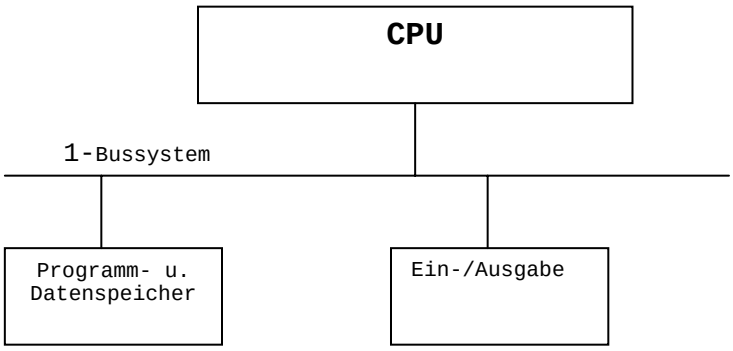
Zeit : 60 min Max. Punktzahl : 36

Zugelassene Hilfsmittel: Vorlesungsskript und Fachliteratur
Nicht zugelassen sind Rechner jeglicher Art !
Bitte verwenden Sie keinen roten Farbstift !

Bearbeiten Sie 3 der 4 Aufgaben. Werden mehr als 3 Aufgaben bearbeitet, wird die Aufgabe mit den meisten Punkten nicht gewertet.

Aufgabe 1.1 (6 Punkte)
Zur Unterscheidung von Rechnerarchitekturen gibt es verschiedene Kriterien u.a. dient der Zugriff auf Programme und Daten als Unterscheidungsmerkmal.

a)



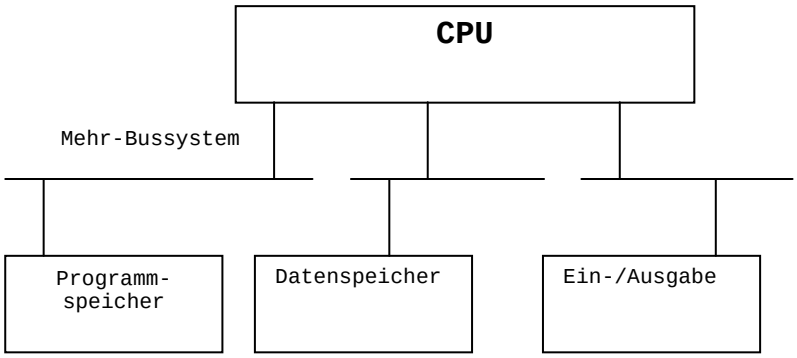
a)

Von Neumann – Architektur

Vorteil:
einfach und flexibel

Nachteil:
nicht besonders schnell

b)



b)

Harvard – Architektur

Vorteil:
gleichzeitiger Zugriff auf Programme und Daten

Nachteil:
viele Anschlusspins

Fragen:
a) Wie lautet die Bezeichnung für beide Architekturen ?
b) Beschreiben Sie Vor- und Nachteile der beiden Architekturen in Stichworten.

Aufgabe 1.2 (6 Punkte)

a) Wie ist bei ganzen positiven Zahlen (Betragszahlen) der Überlauf definiert ?

Carry Flag = 1 ~> [Wertebereich überschritten]

b) Wie ist bei ganzen Zahlen mit Vorzeichen (Zweierkomplement) der Überlauf definiert ?

Carry Flag n XOR Carry Flag n-1 muss erfüllt sein

c) Geben Sie je ein Beispiel für eine 8-Bit-Zahl. Berechnen Sie beide Beispiele und wenden Sie die Regeln aus a) und b) an.

Zahl mit Vorzeichen (Zweierkomplement)

64 + 65 = 129

Stelle		7	6	5	4	3	2	1	0	
		128	64	32	16	8	4	2	1	
		0	1	0	0	0	0	0	0	= 64
+		0	1	0	0	0	0	0	1	= 65
Carry	0	1	0	0	0	0	0	0		
		1	0	0	0	0	0	0	1	129

Overflow - Wertebereich überschritten

Zahl ohne Vorzeichen (Betragszahl)

129 + 131 = 260

Stelle		7	6	5	4	3	2	1	0	
		128	64	32	16	8	4	2	1	
		1	0	0	0	0	0	0	1	= 129
+		1	0	0	0	0	0	1	1	= 131
Carry	1	0	0	0	0	0	1	1		
		0	0	0	0	0	1	0	0	260

Overflow - Wertebereich überschritten

d) Mit welchen Assemblerbefehlen (bedingte Sprünge) können Sie die Überläufe überprüfen ?

Zweierkomplement:

JO, JNO [Jump Overflow, Jump Not Overflow]

Betragszahl:

JC, JNC [Jump Carry, Jump Not Carry]

Aufgabe 2 (12 Punkte)
In einem 1-Bus-Rechenwerk soll der Inhalt von Register R4 auf eine Speicherzelle im Datenspeicher verschoben werden, die von R3 adressiert wird. Der entsprechende Befehl lautet : **MOV [R3],R4**

a) Bei dieser Anordnung werden Daten und Adressen auf einem gemeinsamen Bus transportiert. Wie nennt man dieses Bussystem ? Wo liegen die Vor- und Nachteile dieses Bussystems ?

> Multiplexbus (gemeinsam für Daten und Adressen)
> Vorteil: weniger Hardwareaufwand
> Nachteil: langsamer

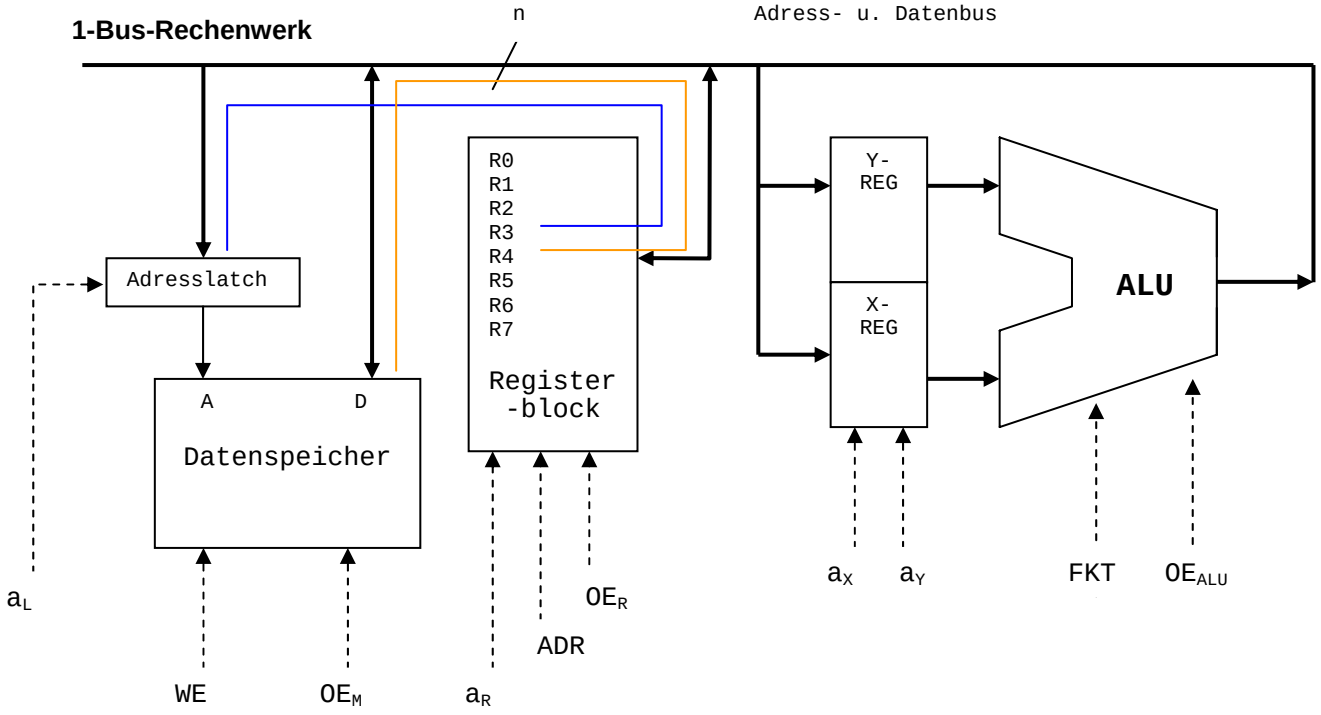
b) Wie wirkt sich ein 1-Bus-Rechenwerk auf die Komplexität der Mikroprogrammschritte bzw. auf die Mikroprogrammlänge aus ?

> mehr Schritte ~> Mikroprogramm wird länger

c) Welche Pfade werden freigeschaltet (vgl. Tabelle) ? Tragen Sie diese Pfade in das Blockdiagramm ein.

d) Tragen Sie in die Tabelle die Mikroprogrammschritte für den Befehl "MOV [R3],R4" ein.

1-Bus-Rechenwerk



Transportzyklen

Mikroprogrammschritt	OE_M	OE_R	a_L	WE	a_R	ADR
(ohne Befehlsholphase)						
R3 => Bus => A-Latch	0	1	1	0	0	R3
R4 => Bus => [A-Latch]	0	1	0	1	0	R4

Bemerkungen:

$a_L = 1$ Information wird vom Bus in das Adress-Latch übernommen
$a_R = 1$ Information wird vom Bus in den Registerblock übernommen, wobei durch **ADR** die Auswahl des Registers erfolgt.
$OE_R = 1$ Output Enable Register
$OE_M = 1$ Output Enable Memory
ADR Adresse des Registers (R0...R7)
WE = 1 Write Enable
Die Signale a_x,a_y,FKT,OE_{ALU} sind für diesen Befehl nicht relevant.

Aufgabe 3 (12 Punkte)
In diesem Progammausschnitt ist ein Unterprogrammaufruf mit 80X86-Code realisiert.

a) Wie können Parameter an ein Unterprogramm übergeben werden ?
Welche Vorteile bietet die hier verwendete Methode ?

Register, Stack, Memory (nicht empfehlenswert)

Vorteil (Stack): beliebig viele, rekursiver Aufruf möglich

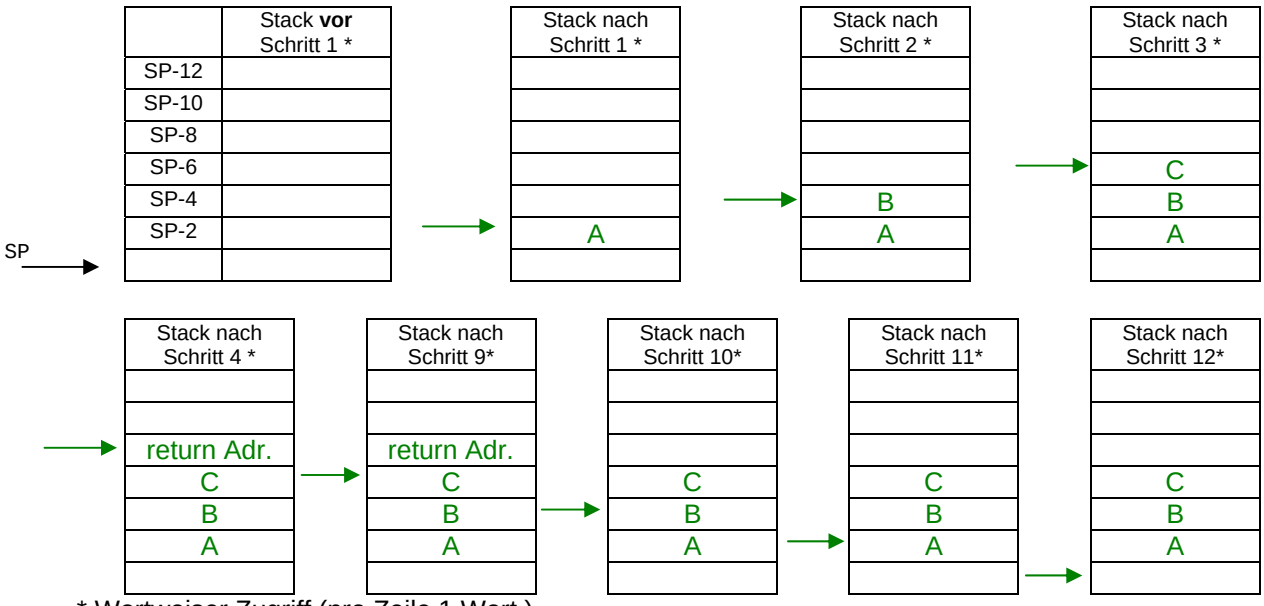
```
                    A           DW ...
                    B           DW ...
                    C           DW ...
                    RESULT      DW ...

Schritt 1           PUSH A
Schritt 2           PUSH B
Schritt 3           PUSH C
Schritt 4           CALL UP

Schritt 10          POP DX
Schritt 11          POP DX
Schritt 12          POP DX
Schritt 13          MOV RESULT, AX

            UP      PROC
Schritt 5           MOV BP,SP           ; Auf welchen Wert im Stack zeigt BP
Schritt 6           MOV AX,[BP+6] (A)            [BP ~> Basepointer ~> SP]
Schritt 7           ADD AX,[BP+4] (B)
Schritt 8           SUB AX,[BP+2] (C)
Schritt 9           RET
            UP      ENDP
```

b) Tragen Sie in die Tabelle ein, welche Werte auf den Stack geschrieben und wie sich die Stackpointerposition (SP) ändert.



* Wortweiser Zugriff (pro Zeile 1 Wort)

c) In welchem Zustand sollte sich der Stack nach einem Unterprogramm-Aufruf befinden

Die Stackbilanz sollte ausgeglichen sein, d.h. er sollte da stehen wo er zu Beginn auch stand.

d) Ergänzen Sie die Befehle nach dem Unterprogramm-Aufruf (Schritt 10 - 12).

e) Welche Rechenoperationen werden im Unterprogramm **UP** durchgeführt ?

A + B - C

f) Das Ergebnis der Berechnungen im Unterprogramm wird im AX-Register abgelegt. Speichern Sie in Schritt 13 das Ergebnis in der Variablen **RESULT** ab.

Aufgabe 4 (12 Punkte)

Geben Sie an, welche Werte die Variablen X,Y und Z, sowie die Register AX, BX nach Ausführung jedes einzelnen Schrittes der folgenden 80x86-Befehlssequenz haben:

```
X DB 1
Y DW 2
Z DW 05555H
...
```

		AX 0000	**BX** 0000	**X** 01H	**Y** 0002H	**Z** 5555H
1.Schritt	XOR BX, BX	0000	0000	01H	0002H	5555H
2.Schritt	NOT WORD PTR X[BX]	0000	0000	FEH	00FDH	5555H
3.Schritt	INC BYTE PTR X[BX]	0000	0000	FFH	00FDH	5555H
4.Schritt	MOV BX, OFFSET Y	0000	OFFSET Y	FFH	00FDH	5555H
5.Schritt	INC WORD PTR [BX]	0000	OFFSET Y	00H	00FEH	5555H
6.Schritt	MOV AX,[BX]	00FEH	OFFSET Y	00H	00FEH	5555H
7.Schritt	AND AX,Z	0054H	OFFSET Y	00H	00FEH	5555H
8.Schritt	MOV Z,AX	0054H	OFFSET Y	00H	00FEH	0054H

Zusatzaufgabe (war nicht Bestandteil der Klausur)

Aufgabe 5 (12 Punkte)

Ein Programm soll eingehende ASCII-Zeichen (AL-Register) nach folgendem Schema filtern:

- Kleinbuchstaben sollen in Großbuchstaben umgewandelt werden und im AL-Register abgelegt werden.
- Alle anderen Zeichen sollen ignoriert werden, d.h. der Ausgang (AL-Register) soll 0FFH sein.

Auszug aus der ASCII-Tabelle:

A	41H	a	61H
B	42H	b	62H
C	43H	c	63H
Z	5AH	z	7AH

a) Entwerfen Sie den Programmablaufplan (Flußdiagramm) für ein Assembler-Unterprogramm, daß dieses Problem löst.

b) Schreiben Sie für folgende Teilaufgabe ein Assembler-Unterprogramm :
Dieser Programmabschnitt soll Kleinbuchstaben erkennen und diese in entsprechende Großbuchstaben konvertieren. Dokumentieren Sie Ihr Programm.