Vorname: _____ **Name:** _____

Matr.-Nr.: _____ **Note:**

19.09.2005 - 14^{00} Uhr bis 16^{00} Uhr

UNIVERSITÄT KARLSRUHE
Institut für Industrielle Informationstechnik
- Prof. Dr.-Ing. habil. K. Dostert -

Vordiplomprüfung im Fach

Mikrorechnertechnik

Die Prüfung umfasst **10 Aufgaben auf 22 Seiten**.

Bitte schreiben Sie auf **alle** Lösungsblätter Ihren Namen und Ihre Matrikelnummer.

Geben Sie bei allen Aufgaben den kompletten Lösungsweg an! Die alleinige Nennung des Endresultats ist zur Erlangung der vollen Punktzahl einer Aufgabe nicht ausreichend.

Die Verwendung eigenen Papiers ist nicht erlaubt.
Bei Bedarf kann zusätzliches Schreibpapier bei der Aufsicht angefordert werden.

Als Hilfsmittel sind Schreib- und Zeichenzeug sowie Taschenrechner mit zu Beginn der Klausur gelöschtem Speicher zugelassen.

Aufgabe:	1	2	3	4	5	6	7	8	9	10	gesamt
Punkte:											
Erreichbare Punktzahl:	9	10	10	11	10	9	10	11	10	10	100

Aufgabe 1: Speicher und speicherbasierte Anwendungen — 9 Punkte

a) Was ist ein EEPROM? Erläutern Sie den Unterschied zwischen einem EEPROM und einem RAM! Warum werden EEPROMs nicht als RAM verwendet?

Eine spezielle Anwendung von Speichern ist die schnelle speicherbasierte Multiplikation. Gegeben sei ein speicherbasierter Multiplizierer gemäß Bild 1.1 zur Multiplikation zweier vorzeichenloser Zahlen.

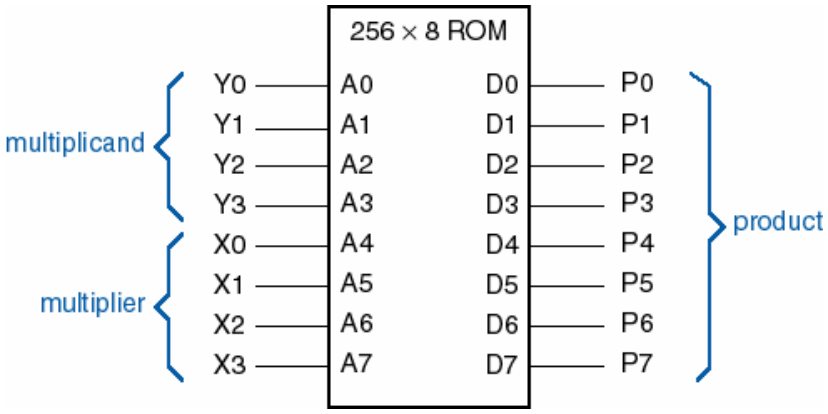

Bild 1.1: 256-Byte-ROM als Multiplizierer

Fortsetzung der 1. Aufgabe:

b) In Tabelle 1.1 ist in der Spalte Inhalt das Ergebnis der Multiplikation gegeben. Vervollständigen Sie Tabelle 1.1 indem Sie eine der möglichen Adressen angeben, die zur Berechnung des jeweiligen Ergebnisses führen!

Adresse	Inhalt
00000000	00000000
	00001010
	00010010
	00011011
	00111000
	01100010
	01111000
	10000000
	10001111
	10101000
	11100001

Tabelle 1.1: Adresse und Speicherinhalt

c) Warum ist die speicherbasierte Multiplikation auf kleine Wortbreiten beschränkt? Begründen Sie Ihre Antwort, indem Sie die Größe des ROMs für einen 16x16bit-Mulitplizierer berechnen!
(**Hinweis:** Geben Sie das Ergebnis in GByte an!)

Aufgabe 2: Zahlendarstellung in Mikrorechnerprogrammen 10 Punkte

Zur Zahlendarstellung in Mikrorechnerprogrammen finden verschiedene Zahlenformate Verwendung. Zunächst wird die nachstehende Bitfolge betrachtet:

b = 1000 0101 1010 0000 0111 0000 0000 0000

(**Hinweis:** Es handelt sich um eine Bitfolge mit einer Länge von 32 bit. Die Bitfolge wurde zur besseren Übersicht in Blöcke mit jeweils 4 bit aufgeteilt!)

Geben Sie den jeweiligen Dezimalwert dieser Bitfolge an, wenn man sie wie folgt deutet:

a) Zahl zur Basis 2 im 2er-Komplement

b) BCD-Zahl

c) Fraktalzahl

d) Gleitkommazahl gemäß IEEE 754. Geben Sie die Gleitkommazahl als gekürzten Bruch und nicht als gerundete Dezimalzahl an!

Stellen Sie die Zahl –8,75 wie folgt dar:

e) Als Gleitkommazahl gemäß dem IEEE-P754 Single-Precision-Standard!

f) Als 2er-Komplement Festkommazahl mit 16 Stellen vor dem Komma und 16 Nachkommastellen!

Aufgabe 3: MOS-Technologie — 10 Punkte

a) Zeichnen Sie das Schaltsymbol und den Querschnitt eines n-Kanal- sowie eines p-Kanal-Transistors in MOS-Technologie! Beschriften Sie im Schaltbild und im Querschnitt Gate, Source und Drain, sowie alle Dotierungen im Querschnitt.
(**Hinweis:** Kennzeichnen Sie eindeutig, bei welchem der Transistoren es sich um einen n-Kanal- bzw. einen p-Kanal-Typen handelt!)

b) Zeichnen Sie die Kennlinie des Drainstroms I_{DS} über der Gatespannung U_{GS} für beide Transistortypen (n- und p-Kanal)! Kennzeichnen Sie jeweils die Schwellspannung U_{th}!

c) Erläutern Sie in Stichworten die Funktionsweise eines n-Kanal-Transistors in MOS-Technologie!

Aufgabe 4: CMOS-Transfergates 11 Punkte

Die CMOS-Technologie ermöglicht den Aufbau besonderer Schaltungsstrukturen mit Hilfe von Transfergates.

a) In Bild 4.1 ist eine mit Transfergates aufgebaute Schaltung abgebildet. Füllen Sie die Wahrheitstabelle (Tabelle 4.1) aus! Welche Funktion wird realisiert?

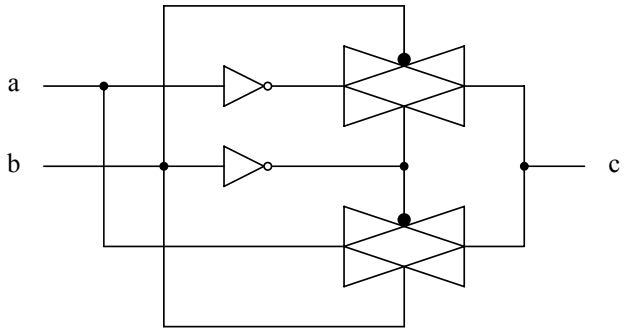

Bild 4.1: Schaltung mit Transfergates

a	b	c

Tabelle 4.1: Zustandsbelegung

Bild 4.2 zeigt ein flankengetriggertes D-Flip-Flop mit einem Takt T, einem negierten Takt /T, einem Dateneingang D, einem Ausgang Q, sowie dem negierten Ausgang /Q.

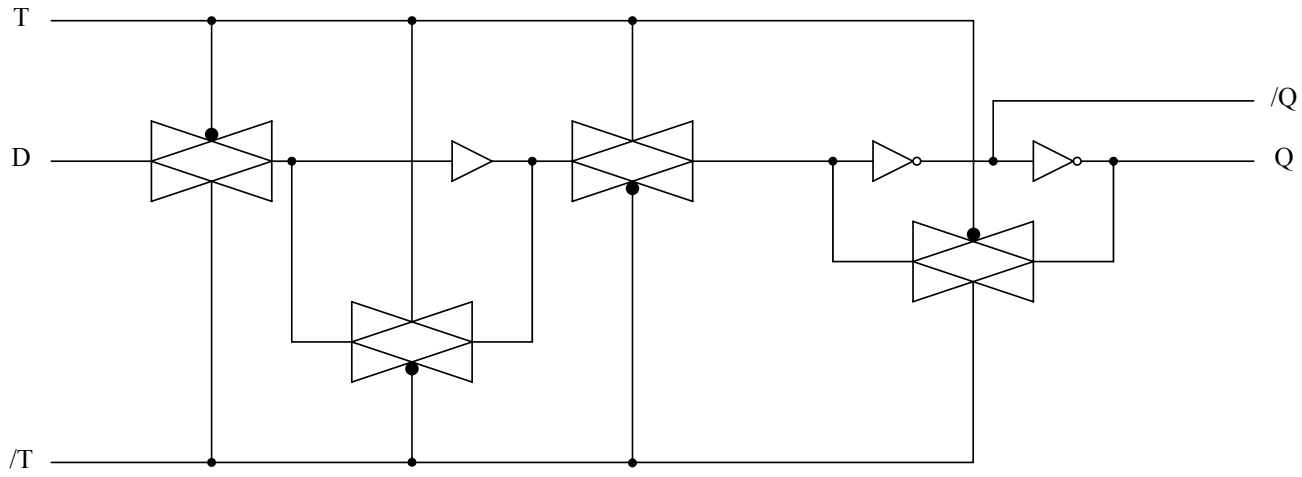

Bild 4.2: Flankengetriggertes D-Flip-Flop

Fortsetzung der 4. Aufgabe:

b) Nennen Sie die Zustände der 4 Transfergates (offen, gesperrt) für den Fall T=0.
(**Hinweis:** Bezeichnen Sie dabei die Transfergates beginnend von links mit den Namen TG1 bis TG4!)

c) Am Eingang des flankengetriggerten D-Flip-Flops liegt nun ein Takt T sowie das Datensignal D gemäß Bild 4.3 an. Vervollständigen Sie das Timingdiagramm in Bild 4.3, indem Sie die resultierenden Signale Q und /Q einzeichnen!

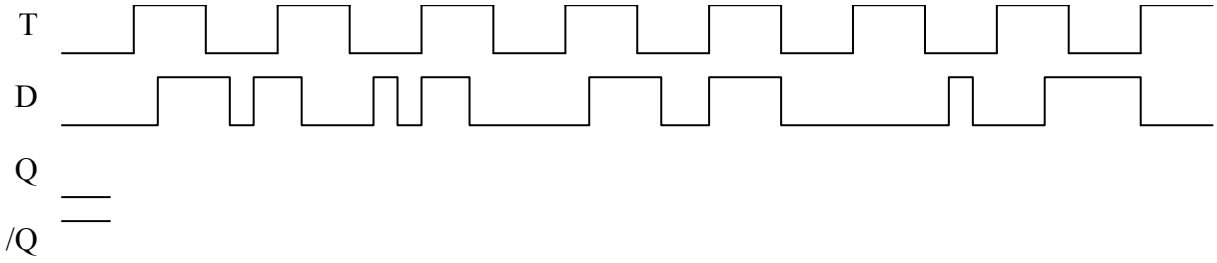

Bild 4.3: Timingdiagramm des flankengetriggerten D-Flip-Flops

d) Handelt es sich bei dem D-Flip-Flop um ein vorderflankengetriggertes oder ein rückflankengetriggertes D-Flip-Flop?

e) In Bild 4.4 auf der folgenden Seite sind zwei „Schleifen“ des D-Flip-Flops schraffiert markiert. In welcher der beiden Schleifen wird für den Fall T=0 die über D eingeschriebene Information (1 bit) gespeichert? Begründen Sie Ihre Antwort!
(**Hinweis:** Die linke Schleife werde mit S1, die rechte Schleife mit S2 bezeichnet.)

Fortsetzung der 4. Aufgabe:

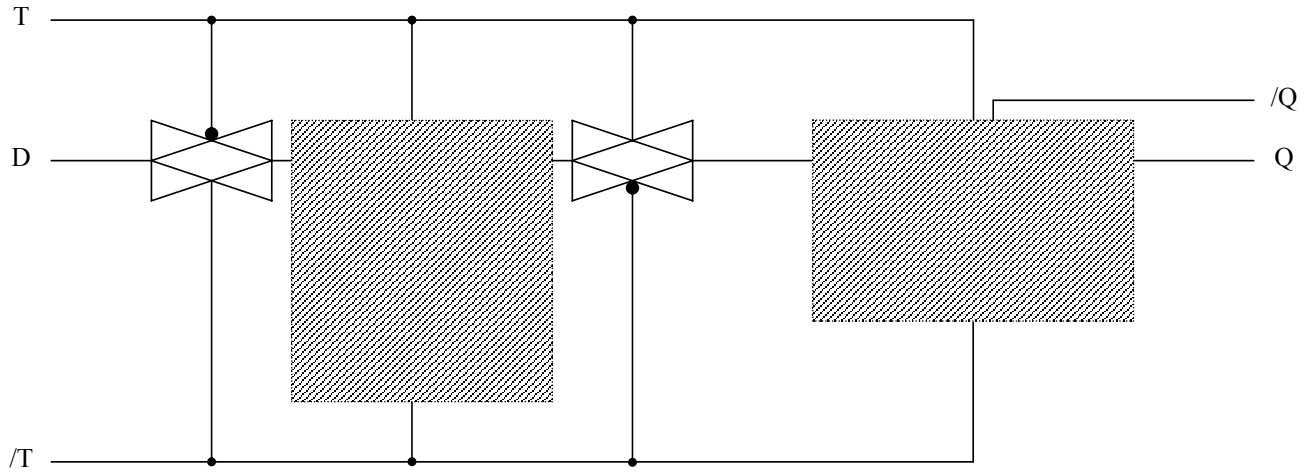

Bild 4.4: Flankengetriggertes D-Flip-Flop

Aufgabe 5: Arithmetik-Schaltungen 10 Punkte

Zunächst wird die in Bild 5.1 dargestellte Schaltung mit den Eingängen *a*, *b* und *c* sowie den Ausgängen *d* und *e* betrachtet:

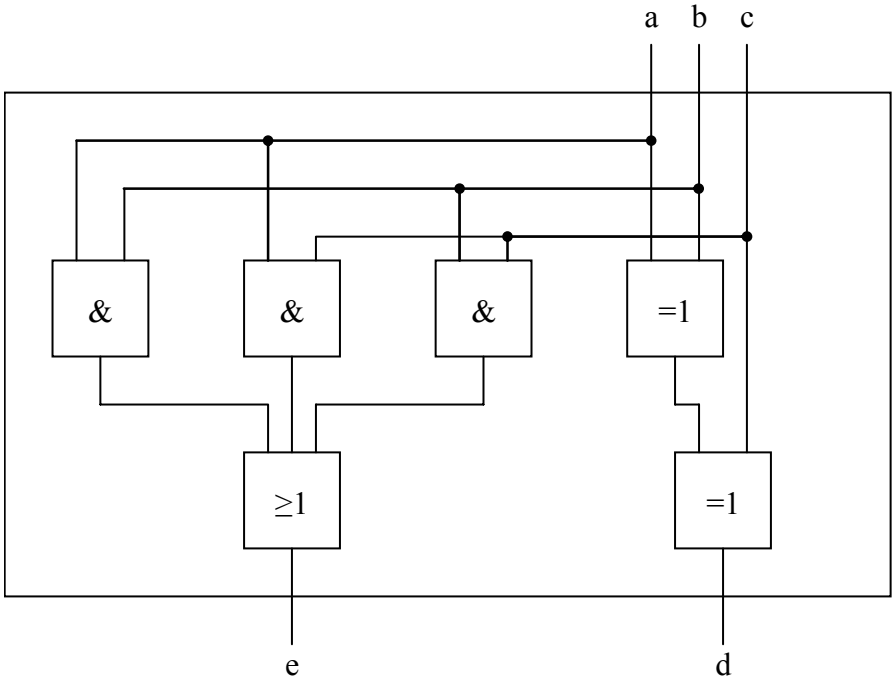

Bild 5.1: Arithmetik-Schaltung 1

a) Vervollständigen Sie entsprechend der Funktion der Schaltung nach Bild 5.1 die folgende Tabelle 5.1!
(**Hinweis:** Die mit „=1“ bezeichneten Gatter realisieren die Antivalenzfunktion. Die Antivalenzfunktion entspricht einer XOR-Verknüpfung, also $(\bar{a} \wedge b) \vee (a \wedge \bar{b})$!)

a_i	b_i	c_i	d_i	e_i
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Tabelle 5.1: Zustandstabelle der Arithmetik-Schaltung 1

Fortsetzung der 5. Aufgabe:

b) Welche Funktion wird mit dem Ausgang *d* in Bild 5.1 beschrieben, welche Funktion wird mit dem Ausgang *e* in Bild 5.1 beschrieben und welche arithmetische Operation wird durch die gesamte Schaltung in Bild 5.1 realisiert?

Im Folgenden wird ein „Halbsubtrahierer“ betrachtet, der für zwei einstellige Dualzahlen *a* und *b*, die positive Differenz $d = a\text{-}b$ sowie das „Borgen“ *e* berechnet.

c) Vervollständigen Sie die folgende Tabelle 5.2, welche die Funktionsweise des obigen Halbsubtrahierers beschreibt!

a_i	b_i	d_i	e_i
0	0		
0	1		
1	0		
1	1		

Tabelle 5.2: Zustandstabelle eines Halbsubtrahierers

d) Durch welche Booleschen Ausdrücke können die Differenz *d* sowie das Borgen *e* beschrieben werden?

e) Der Halbsubtrahierer soll nun in Hardware umgesetzt werden. Ergänzen Sie hierzu das Schaltnetz in Bild 5.2 durch Ausfüllen der Schaltungsblöcke derart, dass ein Halbsubtrahierer entsteht!

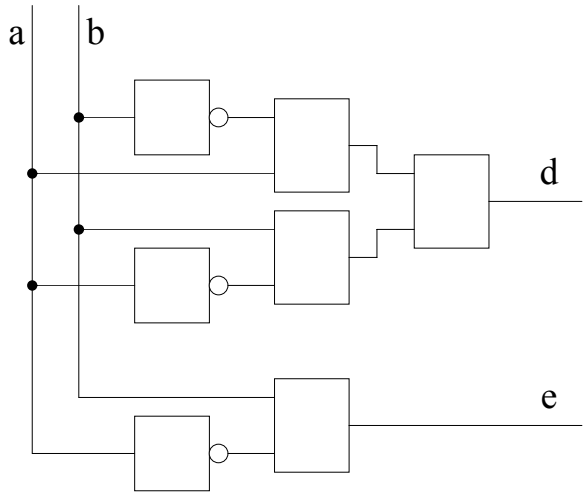

Bild 5.2: Schaltnetz eines Halbsubtrahierers

Aufgabe 6: Automaten, FSM — 9 Punkte

In Bild 6.1 ist das Zustandsdiagramm eines endlichen Automaten abgebildet.

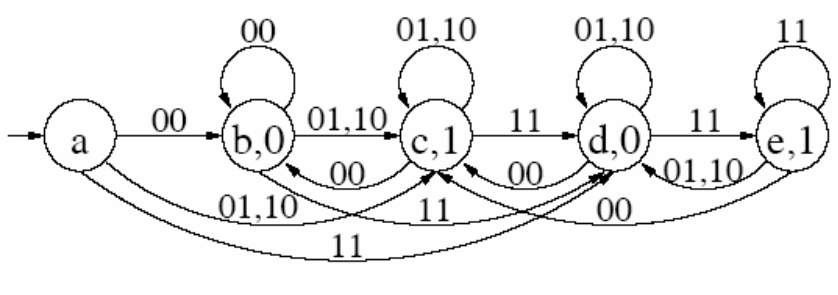

Bild 6.1: Zustandsdiagramm eines Automaten

a) Füllen sie die freien Felder der Tabelle 6.1 entsprechend den Vorgaben aus Bild 6.1 aus!

Eingabe	*Zustand*	*Ausgabe*
-	a	-
00	b	0
01		
10		
11		
11		
00		
00		
11		
10		
11		
11		

Tabelle 6.1: Zustandstabelle eines Automaten

b) Welcher Automatentyp ist in Bild 6.1 dargestellt? (Begründung!)

Fortsetzung der 6. Aufgabe:

c) Welche arithmetische Operation wird durch den Automaten realisiert und was stellen dabei die Zustände *a*, *b*, *c*, *d* und *e*, sowie die Ausgabe dar?

Ein Auszug aus dem Steuerwerk eines Mikrorechners wird mit dem in Bild 6.2 dargestellten Zustandsgraphen beschrieben.

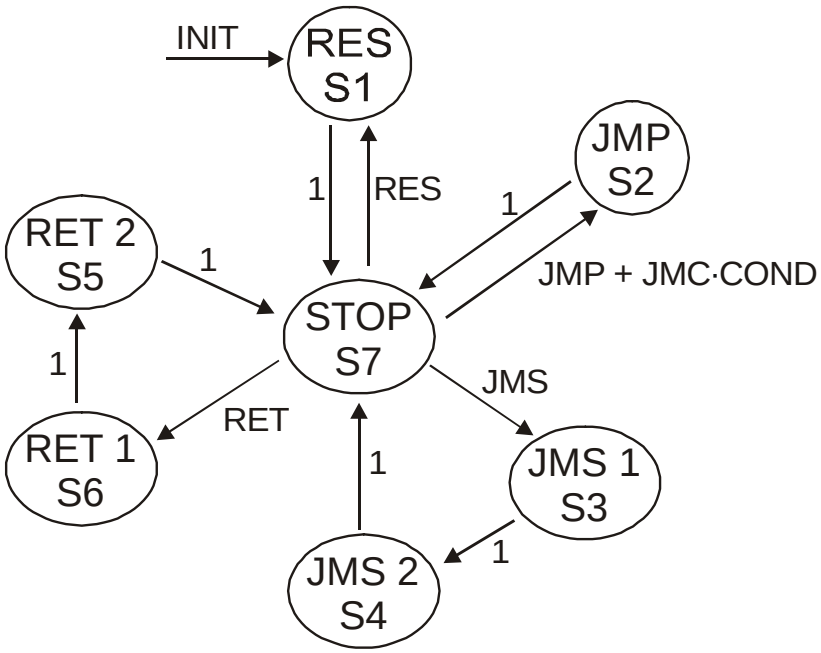

Bild 6.2: Zustandsdiagramm eines Steuerwerks

Die Realisierung des Automaten soll in Form einer „Finite State Machine“ (FSM) erfolgen, welche durch die in Aufgabenteil d) angegebene Flusstabelle (Flow-Table) definiert wird. Die Zustände sind mit s1 ... s7 bezeichnet, während f1 ... f7 jeweils die Folgezustände benennen. Der Eingangsvektor x = (OPC3, OPC2, OPC1, COND, INIT) bestimmt, ausgehend vom aktuellen Zustand s1 ... s7, jeweils den Folgezustand f1 ... f7.
Es sind die in Tabelle 6.2 angegebenen Befehle mit den zugehörigen OP-Codes zu implementieren.

OP-Code	OPC3	OPC2	OPC1	Beschreibung
RES	0	0	0	Reset
JMP	1	0	0	unbedingter Sprung
JMC	1	0	1	bedingter Sprung
JMS	1	1	0	Sprung in Unterprogramm
RET	1	1	1	Rücksprung aus Unterprogramm

Tabelle 6.2: Zu realisierende Befehle und zugehörige OP-Codes

Fortsetzung der 6. Aufgabe:

d) Vervollständigen Sie die folgende Flusstabelle (Tabelle 6.3), indem sie in jedes Kästchen jeweils einen der Werte „0“, „1“ oder „–“ (don’t care) eintragen!

***flow-table**

relevant =		OPC3,	OPC2,	OPC1,	COND,	INIT;		
s7	, x	☐	☐	☐	–	–	, **f1**;	von STOP zu Reset bei RES
s7	, x	–	–	–	–	1	, **f1**;	von STOP zu Reset bei START
s1	, x	☐	☐	☐	☐	0	, **f7**;	nach Reset-Ausführung ⇒ STOP
s7	, x	☐	☐	☐	☐	☐	, **f2**;	JMP
s7	, x	☐	☐	☐	☐	☐	, **f2**;	JMC mit COND = 1
s2	, x	☐	☐	☐	☐	☐	, **f7**;	
s7	, x	☐	☐	☐	☐	☐	, **f3**;	JMS
s3	, x	☐	☐	☐	☐	☐	, **f4**;	
s4	, x	☐	☐	☐	☐	☐	, **f7**;	
s7	, x	☐	☐	☐	☐	☐	, **f6**;	RET
s6	, x	☐	☐	☐	☐	☐	, **f5**;	
s5	, x	☐	☐	☐	☐	☐	, **f7**;	
s[1..s7]	, xrest						, **f7**;	alle übrigen Eingaben bewirken STOP

Tabelle 6.3: Flusstabelle der FSM

Aufgabe 7: Mikrocontrollerprogrammierung mit dem 80C51 — 10 Punkte

Die Hardware von Mikrorechnern ist für Zahlen in reiner Binärdarstellung ausgelegt. Für die Datenübertragung und die Darstellung von Ergebnissen z.B. einer Analog-Digitalwandlung ist der „American Standard Code for Information Interchange“ – ASCII – ein weit verbreitetes Format. Ein Zeichen (z.B. ein Buchstabe oder eine Zahl) belegt in diesem Code immer ein Byte. Die Ziffern 0...9 werden im ASCII-Code ins untere Halbbyte geschrieben, während das obere immer ’0011’ ist. Die Ziffern müssen also im BCD-Format vorliegen.

Es soll der Wandelwert eines 9bit-A/D-Wandlers (000...1FFh), der in zwei Registern mit den Namen *ADH* (oberes Byte) und *ADL* (unteres Byte) zwischengespeichert ist, zunächst ins BCD-Format gewandelt werden. Dann ist das Ergebnis in drei Register mit den Namen *AD_H* (Hunderter), *AD_Z* (Zehner) und *AD_E* (Einer) im ASCII-Format zu bringen.

Das Byte aus *ADL* wird zunächst mit Hilfe des *DIV AB*-Befehls in Hunderter, Zehner und Einer zerlegt. Dann wird nach Maßgabe des Bits in *ADH* der Dezimalwert 256 addiert, wobei eine Korrektur mit dem Befehl *DA A* erfolgt.
Zunächst sollen die BCD Ziffern in Halbbytes der Register *R5* (Hunderter im unteren Halbbyte, das obere enthält Nullen) und *R4* (Zehner im oberen und Einer im unteren Hallbyte) zwischengespeichert werden.

Dann werden Hunderter, Zehner und Einer in die unteren Halbbytes von *AD_H*, *AD_Z* und *AD_E* übertragen.
Das obere Halbbyte ist dort immer ’0011’=’3d’, so dass die Ziffern im ASCII-Format vorliegen. In diesem Format können die Ziffern über eine serielle Schnittstelle z.B. an einen PC gesendet und dort unmittelbar mit einem Terminalprogramm dargestellt werden.

a) Vervollständigen Sie den folgenden Programmabschnitt an den mit **.....** markierten Stellen!

```
BIN_BCD:
MOV  R5,#00h            ;Zwischenregister leeren
MOV  R4,#00h
MOV A,ADL               ;unteres Byte (ADL) in Akku holen
MOV .................   ;Divisor nach B holen
DIV AB                  ;Inhalt von ADL durch 100 teilen
                        ;wobei das ganzzahlige Ergebnis
                        ;(Hunderter)in A, der Rest in B steht
MOV .................   ;Hunderter in Zwischenregister
                        ;abspeichern
MOV .................   ;Rest (Zehner und Einer) in Akku holen
```

Fortsetzung der 7. Aufgabe:

```
MOV B,#10
.....................  ;Rest durch 10 teilen (Zehner in A,
                       ;Einer in B)
SWAP A                 ;Zehner in oberes Halbbyte
.....................  ;Einer dazuaddieren
MOV  R4,A              ;Zehner und Einer abspeichern, jetzt
                       ;das höchstwertige Bit in ADH prüfen

;-------------------------------------------

MOV ..................  ;höherwertiges Byte holen

;falls Bit 0 gesetzt, 256 zu R5, R4 addieren mit
;Dezimalkorrektur

JNB ..................  ;falls "0", Sprung zu Abschnitt Fertig

;sonst 256 zu R5,R4 addieren

;-----------------------------------------------

MOV ..................  ;Zehner und Einer holen
ADD  A,#56h            ;Achtung: Dezimalwert als HEX-Zahl
DA   A                 ;Dezimalkorrektur
MOV  R4,A              ;korrigierten Wert abspeichern
MOV  A,R5              ;Hunderter holen
.....................  ;mit Carry aus Zehnern addieren
.....................  ;Dezimalkorrektur
MOV  R5,A              ;Hunderter abspeichern

;-----------------------------------------------
```

Fortsetzung der 7. Aufgabe:

```
FERTIG:
;ASCII-Codes erzeugen und wie folgt ablegen:
;Hunderter in AD_H
;Zehner  in AD_Z
;Einer in AD_E
MOV  A,R5               ;Hunderter holen
.....................   ;eine "3" voranstellen
MOV  AD_H,A             ;ASCII-Hunderter fertig in AD_H
MOV  A,R4               ;Zehner und Einer holen
SWAP A                  ;Zehner in unteres Halbbyte
.....................   ;oberes Halbbyte (Einer) löschen
.....................   ;eine "3" voranstellen
MOV  AD_Z,A             ;ASCII-Zehner fertig in AD_Z
MOV  A,R4               ;nochmal Zehner und Einer holen
.....................   ;oberes Halbbyte (Zehner) löschen
.....................   ;eine "3" voranstellen
MOV  AD_E,A             ;ASCII-Einer fertig in AD_E
RET
```

b) Im Folgenden soll der oben stehende Programmcode ab der Marke „*FERTIG*“ betrachtet werden. Geben Sie für jeden der auf die Marke folgenden Schritte den Inhalt des gerade modifizierten Registers an! Die Register *R4* und *R5* seien wie folgt belegt: *R5 = 01h*, *R4 = 23h*.

Aufgabe 8: Langzeitintervall-Timer eines ADuC842 — 11 Punkte

Für einen Mikrocontroller vom Typ ADuC842 soll ein Unterprogramm erstellt werden, mit dem eine Pause von 5 Tagen eingestellt wird, in welcher der Mikrocontroller z.B. auf einen externen Interrupt durch eine abgeschlossene Messwertaufnahme wartet. In dieser Wartezeit soll der Mikrocontroller in einem Energiesparmodus laufen, der durch Verringern der Taktfrequenz erreicht wird.

(**Hinweis:** In der beiliegenden Hilfsblattsammlung finden Sie eine Kurzbeschreibung der verwendeten Spezialfunktionsregister. Beachten Sie zudem, dass der Energiesparmodus nicht durch das Bit „Oszillator Power Down“ (PLLCON Bit 7) eingestellt werden soll!)

a) Geben Sie die notwendigen Einstellungen der Spezialfunktionsregister *PLLCON*, *TIMECON* und *INTVAL* in Tabelle 8.1 für die folgende Konfiguration an!

   i. energiesparsamsten Modus einstellen
   ii. Timer für Wartezeiten im Bereich von Tagen konfigurieren
   iii. Timer-Intervall-Zähler nicht freigeben
   iv. Intervall-Timer stoppen
   v. 5 Tage Wartezeit vorgeben
   vi. restliche Bits = 0 setzen.

	Bit 7							Bit 0
PLLCON								
TIMECON								
INTVAL								

Tabelle 8.1: Spezialfunktionsregister des ADuC842

b) Vervollständigen Sie den nachfolgenden Programmabschnitt, indem Sie an den gepunkteten („….“) Stellen den Befehl entsprechend kommentieren bzw. für den gegebenen Kommentar den entsprechenden Befehl schreiben!

```
;-----Intervall-Timer vorbereiten, damit er 5 Tage Pause erzeugt------
.........................    ;Intervallzähler NICHT freigeben
                             ;noch KEINEN Timer-Takt setzen
                             ;Zeitbasis für 5 Tage setzen

.........................    ;5 Tage Pause vorgeben


CALL   PAUSE

;----------------- Unterprogramm für 5 Tage PAUSE --------------------
PAUSE:
.........................    ;energiesparsamsten Modus EIN
```

Fortsetzung der 8. Aufgabe:

```
ORL    TIMECON,#00000011b     ;.....................................

                              ;.....................................

WART:
MOV    A,TIMECON              ;Bit 2 = TII in TIMECON prüfen und
JNB    ACC.2,WART             ;Sprung zu WART

.......................       ;Stopp und Reload Langzeittimer


MOV    PLLCON,#00h            ; ....................................

RET

;----------------------- END of 5 Tage PAUSE ------------------------
```

c) Warum wird der Akkumulator zur Untersuchung von *TII* des Registers *TIMECON* benutzt, d.h. warum kann nicht gleich das Bit *TIMECON.2* abgefragt werden?

d) Worauf wird bei der Abfrage *`JNB ACC.2,WART`* gewartet und was wird hier abgefragt?

Aufgabe 9: Analyse von Assembler-Code für den DSP56001 — 10 Punkte

Der unten angegebene Assembler-Code soll auf einem DSP zur Berechnung einer Korrelationsfunktion eingesetzt werden. Hierzu soll die Eingangssignalfolge $x(k)$ mit einer Referenzsignalfolge $y(k)$ wie folgt zum Ergebnis z verknüpft werden:

$$z = \sum_{k=0}^{255} x(k) \cdot y(k) \,. \tag{9.1}$$

Die 256 Abtastwerte des Eingangssignals $x(k)$ mit $k = 0 \ldots 255$ sollen zu Beginn im X-Datenspeicher ab Adresse 0 liegen, die 256 Abtastwerte des Referenzsignals $y(k)$ mit $k = 0 \ldots 255$ im Y-Datenspeicher, ebenfalls ab Adresse 0.
Das Ergebnis z soll nach dem Unterprogrammende im Register A stehen. Die Pointer R0 und R4 müssen für den nächsten Unterprogrammaufruf richtig gesetzt werden! Außerdem muss der neue Eingangswert $x(255)$ im X-Speicher für den nächsten Aufruf des Unterprogramms aktualisiert werden. Dieser Wert steht im Register X1.

```
1     MOVE #$0,R0
2     MOVE #$100,M0
3     MOVE #$10,R4
4     MOVE #$100,M4
5     CLR A          X:(R0)+,X0  Y:(R4)+,Y0
6     REP #253
7     MAC X0,Y0,A    X:(R0)+,X0  Y:(R4)+,Y0
8     MAC X0,Y0,A    X:(R0)+,X0  Y:(R4)+,Y0
9     MAC X0,Y0,B    X:X1,(R3)
10    RTS
```

Assembler-Code zur Berechnung einer Korrelationsfunktion

a) Überprüfen Sie, ob mit dem angegebenen Assembler-Code tatsächlich die gewünschte Berechnung laut Aufgabenstellung durchgeführt wird! Falls ja, erläutern Sie kurz die einzelnen Schritte des Assembler-Codes. Falls nein, geben Sie an, wie der Code verändert werden muss, damit das Ergebnis z entsprechend richtig berechnet wird!
(**Hinweis:** Die vor jeder Zeile stehenden grauen Zahlen dienen ausschließlich der Nummerierung der Zeilen und sind nicht Teil des Codes. Verwenden Sie diese Zahlen zur Kommentierung Ihrer Antwort!)

Fortsetzung der 9. Aufgabe:

Der X-Speicher eines DSP56001 sei gemäß Tabelle 9.1 belegt.

Adresse	Inhalt
$14F	$555
$150	$444
$151	$333
$152	$222
$153	$111

Tabelle 9.1: Speicherbelegung des X-Speichers eines DSP56001

b) Ergänzen Sie die freien Felder der nachfolgenden Tabelle 9.2!
(**Hinweis:** In der Tabelle bezeichnet *R0 (S)* den Startwert des Adresszeigers, *R0 (E)* den Endwert des Adresszeigers nach Abarbeitung des Befehls.)

	R0 (S)	N0	R0 (E)	A	Bezeichnung
MOVE X:-(R0),A				**$222**	
		$2	**$151**	**$111**	**indirect with postdecrement by offset**
MOVE X:(R0+N0),A			**$14F**	**$111**	**indirect by offset**

Tabelle 9.2: Adressierungsarten eines DSP56001

Aufgabe 10: Beschreibung und Analyse von Schaltungen mit VHDL 10 Punkte

Zunächst soll eine MAC-Einheit entwickelt werden. Die für die Berechnung benötigten Werte seien vom Typ *integer* und sollen nur positive Werte im Bereich von 0 bis 127 annehmen.

a) Vervollständigen Sie die nachfolgende ENTITY und die zugehörige ARCHITECTURE, so dass eine entsprechende MAC-Einheit entsteht! Die Entity soll neben den beiden Eingängen einen Takt-, sowie einen Reset-Eingang enthalten. Falls das Reset-Signal den Wert 0 annimmt, soll der Ausgang der MAC-Einheit ebenfalls auf 0 gesetzt werden. Im anderen Fall (Reset = 1) soll jeweils mit der fallenden Taktflanke des Taktes ein Rechenvorgang innerhalb der MAC-Einheit durchgeführt werden. Der berechnete Wert soll dem Port *result* übergeben werden!

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity mac is
port (




      result: out integer range 0 to 145160);
end mac;

architecture mac_arch of mac is



begin
     process(                              )
     begin









     end process;
end mac_arch;
```

Fortsetzung der 10. Aufgabe:

b) Nach wie viel Schritten tritt ein Überlauf des Ergebnisses auf, wenn die Eingangswerte immer den Maximalwert 127 annehmen?

c) Erläutern Sie in Stichworten den Unterschied zwischen Signalen und Variablen!

d) Was wird mit VHDL beschrieben, d.h. wofür wird diese Sprache benötigt? Welche Hauptunterschiede gibt es im Vergleich zu herkömmlichen Programmiersprachen wie C oder Pascal?