Maschinenorientierte Programmierung Klausur - Zug 2 Gruppe B, 18.07.2016, SoSe 2016	Name: [redacted]
	Matrikelnr.: [redacted]

Hinweis

Als Hilfsmittel sind erlaubt:

- MOP-Skript und Folien
- Befehlsliste für den Mikrocontroller 8051
- eigene Mitschriften aus der Vorlesung
- eigene Lösungen von Laboraufgaben
- Taschenrechner (nicht programmierbar)

Das Verwenden weiterer Hilfsmittel, sowie die Angabe falscher Voraussetzungen (siehe "Erklärung") gelten als Täuschung und führen zur Nichtanerkennung der Klausur.

Es sind alle Arbeits- und Ableitungsschritte zu dokumentieren. Lösungen ohne erkennbaren Lösungsweg oder kurze Begründung, sowie durchgestrichene oder nicht lesbare Lösungen werden nicht gewertet. Bei zwei angegebenen Lösungen wird keine berücksichtigt. Bitte benutzen Sie keine roten oder grünen Stifte. Mobiltelefone sind abzuschalten und in der Tasche zu verstauen. Der gegenseitige Austausch von Unterlagen, sowie Unterhaltungen während der Klausur sind nicht gestattet. Jeder Täuschungsversuch, sowie das Anfertigen von Kopien führt zum sofortigen Ausschluss und somit zum Nichtbestehen der Prüfung.

Zum Bestehen der Klausur sind 50 von insgesamt 120 (100) Punkten hinreichend. Es stehen mehr Aufgaben für die Bearbeitung zur Verfügung als zum Bestehen erforderlich.

Bitte versehen Sie jedes Blatt mit Namen und Matrikelnummer und nummerieren Sie alle Blätter durch.

Bearbeitungszeit: 90 min.

Erklärung

Ich bestätige mit meiner Unterschrift, dass ich prüfungsfähig bin und bei Beginn der Klausurarbeit die vollständigen Unterlagen erhalten habe. Ferner erkläre ich, dass ich zu dieser Prüfung angemeldet bin und alle Voraussetzungen zur Zulassung durch eigenständige Bearbeitung aller Laboraufgaben und Abgabe eines selbst angefertigten Protokolls zu jedem Termin erfüllt habe.

[redacted]

(Unterschrift)

Ab hier bitte keine Eintragungen vornehmen!

Aufgabe:	1	2	3	4	5	6	**Summe**
Punkt(e):	24	22	24	12	24	14	120
Erreicht:	21	17	22.5	9	15	13	97,5

Note: 1.0

Gratulation

Prof. Dr. Sven-Hendrik Voß

Fachbereich VI

Aufgabe 1 (24 Punkte)

Multiple Choice

Bitte kreuzen Sie alle richtigen Antwortmöglichkeiten an. Es gibt immer (mindestens) eine korrekte Antwort.

Die Zahl 0F52h...

○ entspricht 392d ○ passt in 8-Bit Register ⊗ passt nicht in 8-Bit Register ⊗ entspricht 3922d

Der interne Speicherbereich zwischen 80h und FFh beim 8051 ist belegt mit... −1

○ den vier Registersätzen ⊗ SFRs ○ bitadressierbarem RAM ⊗ frei verfügbarem byte-adressierbarem RAM

Der 8051 unterstützt sowohl direkte als auch indirekte Adressierung. Welche Aussage ist richtig?

○ Programmspeicher ist nur direkt adressierbar ○ alle 32 Register in den Registerbänken dienen indirekter Adressierung ⊗ Data Pointer dient zur indirekten Adressierung ○ externe Speicherbereiche sind über Akku indirekt adressierbar

Welches Register liegt nicht im SFR-Bereich?

○ Stackpointer-Register ○ Datapointer-Register ⊗ R0-Register ○ Akkumulator-Register

Bei einem Zugriff auf den Stack wird beim 8051...

⊗ der Stackpointer vor dem Schreibzugriff inkrementiert ○ der Stackpointer nach dem Schreibzugriff inkrementiert ○ der Stackpointer vor dem Lesezugriff dekrementiert ⊗ der Stackpointer nach dem Lesezugriff dekrementiert

Über wieviele Interruptquellen (extern und intern) verfügt der (klassische) 8051?

○ 2 ○ 4 ⊗ 5 ○ keine

Über wieviele Timer Modi verfügt der 8051?

○ 1 ○ 2 ○ 3 ⊗ 4

Wie groß ist der Maximalwert, den das Timer-Zählregister speichern kann?

⊗ 65536 ⊗ 65535 ○ 28800 ○ 256 −2

Wie wird die Zieladresse eines relativen Sprungs beim 8051 gebildet?

○ relativ zum aktuellen Program Counter Wert im Bereich -64 bis +63 ○ absolut durch Überschreiben der unteren 11 Bits des Program Counter ⊗ relativ zum aktuellen Program Counter Wert im Bereich -128 bis +127 ○ absolut durch komplettes Überschreiben des Program Counter Wertes

Nach Unterscheidung zwischen Primär- und Sekundärmodus haben...

○ alle Ports des 8051 eine alternative Funktion ⊗ einige Ports des 8051 eine alternative Funktion ○ keine Ports des 8051 eine alternative Funktion ○ die Ports entgegengesetzte Polarität

Default-mäßig ist welche Registerbank als aktive Registerbank voreingestellt?

⊗ Registerbank 0 ○ Registerbank 1 ○ Registerbank 2 ○ Registerbank 3

Im PSW befindet sich der Wert 88h. Auf welcher Registerbank arbeitet der Prozessor?

○ Registerbank 0 ⊗ Registerbank 1 ○ Registerbank 2 ○ Registerbank 3

Aufgabe 2 (22 Punkte)

Umgang mit dem 8051

Gegeben sei folgendes Assemblerprogramm.

```
MOV 81h, #19h
MOV R2, #3Dh
MOV R3, #56h
MOV A, R2
SWAP A
ANL A, #Ø42h
PUSH ØEØh
ADD A, R3
PUSH ØEØh
POP Ø2
POP Ø3
END
```

1. Das Programm nutzt offensichtlich den Stack (PUSH, POP). In welchem Speicherbereich (welcher Speicher, welche Adressen) wird der Stack angelegt?
2. Wozu dienen normalerweise die unter 1. ermittelten Adressen? Gibt es in jedem Fall einen Konflikt?
3. Was wird auf dem Stack abgelegt?
4. Welche Adressierungsart wird für den Stack-Zugriff genutzt?
5. Was steht letztlich in den Registern R2 und R3?

Das folgende Programm benutzt die relative Adressierung für einen bedingten bzw. unbedingten Sprungbefehl. Nehmen Sie an, dass am Port P2 Schalter als Eingabemöglichkeit angeschlossen sind.

```
ØØØØ|   MOV R4, P2
ØØØ2|   CLR C
ØØØ3|   MOV A, #6Eh
ØØØ5|   SUBB A, Ø4h
ØØØ7|>  JZ ________
ØØØ9|   INC Ø4h
ØØØB|   SJMP ________
ØØØD|   NOP
        END
```

1. Vom Befehl JZ soll zum Befehl NOP verzweigt werden. Der SJMP Befehl soll einen Sprung an die Stelle CLR C bewirken. Tragen Sie in die gekennzeichneten Felder den resultierenden relativen Adress-Offset für die relative Adressierung ein.
2. Worauf bezieht sich der Befehl JZ?
3. Wozu ist es (im Kontext des konkreten Programms) nötig, das Carry-Bit zu löschen (Zeile 2)?
4. Nachdem Sie nun das vollständige Programm analysiert haben, beschreiben Sie kurz was das Programm tut. Welche Funktion hat es?

Aufgabe 3 (24 Punkte)

Programmierung in Assembler

Ein 8051 Mikrocontroller soll dazu verwendet werden ein 100 Hz Rechtecksignal an Port P3.6 zu erzeugen, solange an Port 1 nicht 00h anliegt. Dazu soll mit Hilfe des Timers 0 in regelmäßigen Abständen ein Interrupt erzeugt werden. In der zugehörigen Interrupt Service Routine soll der Ausgabe-Pin abwechselnd ein- bzw. ausgeschaltet werden, sofern die notwendige Bedingung erfüllt ist.

1. Skizzieren Sie den Programmablauf zu dieser Aufgabe.
2. Welche Zeitspanne muss Timer 0 für die Rechtecksignalgenerierung erzeugen?
3. Welcher Timer Modus ist dafür am besten geeignet?
4. Welche ISR-Einsprungadresse ist dem Timer 0 zugeordnet?
5. Geben Sie alle Maßnahmen an, damit der Timer 0 überhaupt einen Interrupt auslösen und an die soeben ermittelte Adresse verzweigt werden kann.
6. Geben Sie die Programmanweisungen in Assemblersyntax an, um den Timer für den in 3. ermittelten Modus zu konfigurieren. Hierzu gehört auch das Setzen der richtigen Zählregisterwerte.
7. Muss der Timer während des Neusetzens der Zählregisterwerte in der ISR angehalten werden? Begründen Sie in jedem Fall ihre Aussage.
8. Welche Zeitspanne kann in dem von Ihnen gewählten Modus maximal vom Timer generiert werden? Zeigen Sie die Rechnung.
9. Schreiben Sie die ISR für den Timer 0 als Assemblercode (<u>nicht</u> das ganze Programm!)

Aufgabe 4

(12 Punkte)

Speicherverwaltung des 8051 Mikrocontroller

Es liegt ein Mikrocontrollersystem vor bestehend aus einem 8051, einem externen Programmspeicher, einem externen Datenspeicher und Latches. Programm- und Datenspeicher umfassen jeweils 64 kByte.

1. Welche Ports des 8051 werden für das gemäß obiger Beschreibung realisierte System benötigt?

   P0 → AD-Signal

   P2 → A1-Signal

   (3)

2. Es sollen nun Daten aus dem externen Datenspeicher gelesen werden, die sich an der Adresse 01B3h befinden. Die Daten sollen in den Akku geladen werden. Geben Sie eine geeignete Befehlsfolge an.

   1. MOV ~~DPRT~~ DPTR, #01B3h
   2. MOV A, #00h
   3. MOVX A, @DPTR

   (3)

3. Nennen Sie die Adressierungart, die hier zur Anwendung kommt.

   1./2. Unmittelbare Adressierung

   3. Register-indirekte Adressierung

   (3)

4. Mit dem Oszilloskop messen Sie für das ALE-Signal eine Frequenz von 250 kHz. Mit welcher Frequenz wird der 8051 betrieben?

Aufgabe 5 (24 Punkte)

Arbeiten mit der seriellen Schnittstelle

Folgendes Programm für die Kommunikation über die serielle Schnittstelle sei gegeben. Die Oszillatorfrequenz betrage 12 MHz.

```
ORG ØØh
MOV TMOD,#2Øh
MOV TH1,#-12
MOV SCON,#5Øh
SETB TR1
warte: JNB RI, warte
MOV A,SBUF
MOV P1,A
CLR RI
SJMP warte
```

1. Analysieren Sie das Programm und beschreiben Sie kurz die Funktionalität. Was tut das Programm?
2. Geben Sie an, in welchem Modus die serielle Schnittstelle und der Timer betrieben wird.
3. Welche Baudrate wird hier generiert? Entspricht die Baudrate einer Standard-Baudrate? Falls nicht, geben Sie bitte die prozentuale Abweichung zu nächstgelegenen Standard-Baudrate an.
4. Modifizieren Sie das Programm in der Hichsicht, dass es 9-Bit Daten verarbeiten kann. Gehen Sie dann davon aus, dass bei allen empfangenen Daten an der 9. Stelle das Parity-Bit zur Fehlererkennung mitgesendet wird. Implementieren Sie darauf basierend eine Fehlererkennung, die an Port P2.7 einen Fehler durch einen High-Pegel anzeigt.

Aufgabe 6

(14 Punkte)

Debugging eines Assembler-Programms

Ein Neuling auf dem Gebiet der Assembler-Programmierung hat folgendes Programm entworden. Zur Übergabe der aktuellen Werte von Akku und PSW möchte er den Stack nutzen.

```
        ORG ØØh
ØØØØ|   MOV A, #Ø9h
ØØØ2|   MOV B, #Ø2h
        weiter:
ØØØ5|   MUL AB
ØØØ6|   PUSH ACC
ØØØ8|   PUSH PSW
ØØØA|   ACALL foo
ØØØC|   INC B
ØØØE|   SJMP weiter

        ORG 44h
        foo:
ØØ44|   POP PSW
ØØ46|   POP ACC
ØØ48|   JB ACC.Ø, back
ØØ4B|   INC ACC
        back:
ØØ4D|   RETI
```

1. Analysieren Sie das Programm. Worin besteht der elementare Fehler in dem Programm?
2. Welche Auswirkungen hat der Fehler? Was passiert fälschlicherweise?
3. Wie kann der Fehler behoben werden? Welcher andere Lösungsansatz ist möglich, damit das Programm funktioniert?
4. Skizzieren Sie den Stack-Inhalt zum Zeitpunkt des RETI-Aufrufs und beschriften Sie ihn vollständig mit allen bekannten Informationen (Start-Adresse, Füllstand)

2) a)

1. im Daten-Speicher
an der Adresse 19h +1 (1)

2. Es gibt nur Konflikte, wenn die 3. Register-Bank benutzt wird und auch nur dann, wenn R1 benutzt wird. R2 (1.5)

3. Der Wert an der Adresse 0E0h, was dem Akku-Inhalt entspricht, ~~0E0~~ mit den Zwischenergebnissen (2)

4. Für den Stack kann nur die direkte Adressierungsart verwendet werden. (2)

5. $\begin{pmatrix} R2 = 3Dh \\ R3 = 56h \end{pmatrix}$

A = 3Dh

SWAP = D3h

ANL
$$\begin{array}{r} 1101\ 0011 \\ 0100\ 0010 \\ \hline 0100\ 0010 \end{array}$$

PUSH 42h

ADD 42h + 56h = 98h

push 98h

$$\boxed{\begin{array}{l} R2 = 98h \\ R3 = 42h \end{array}}$$ ✓ (3)

1

b)

1.

PC = 0007h

000Dh

JZ 05h -4

PC = 000Bh

0002h

-9h

00001001
11110110
11110111
F7h -2 (3,5)

JMP F7h

2. Auf den Inhalt des Akkus (2)
er springt, wenn der Akku leer ist.

~~3. 0110 1110~~
~~- 0000 0100 → 1111 1011 +1~~
~~1111 1100~~
~~Das Comp-Bit~~

~~0110 1110~~
~~+ 1111 1100~~
~~$\boxed{1}$ 1110 1010~~

2

3. ~~[illegible]~~

Der SUBB-Befehl zieht nicht nur R4 von Akku ab, sondern auch noch das Carry-Bit.

Da das Carry hier aber nicht weiter benutzt werden soll, da es die Rechnung verfälschen würde muss es gelöscht werden, sonst endet das Programm 1 zu früh.

(2)

4. Von 110d wird im des Register 4 Akkumulator, welches von dem User eingeben wird, als PZ.

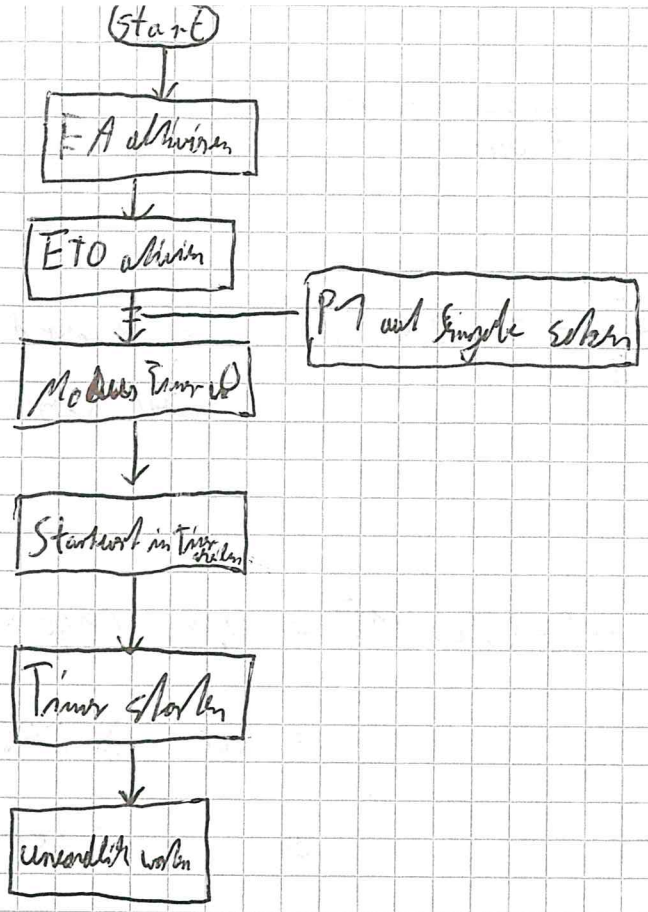
~~[illegible]~~

Die Eingaben-Zahl wird solange erhöht, bis sie dem Wert 110 entspricht und dann beendet.

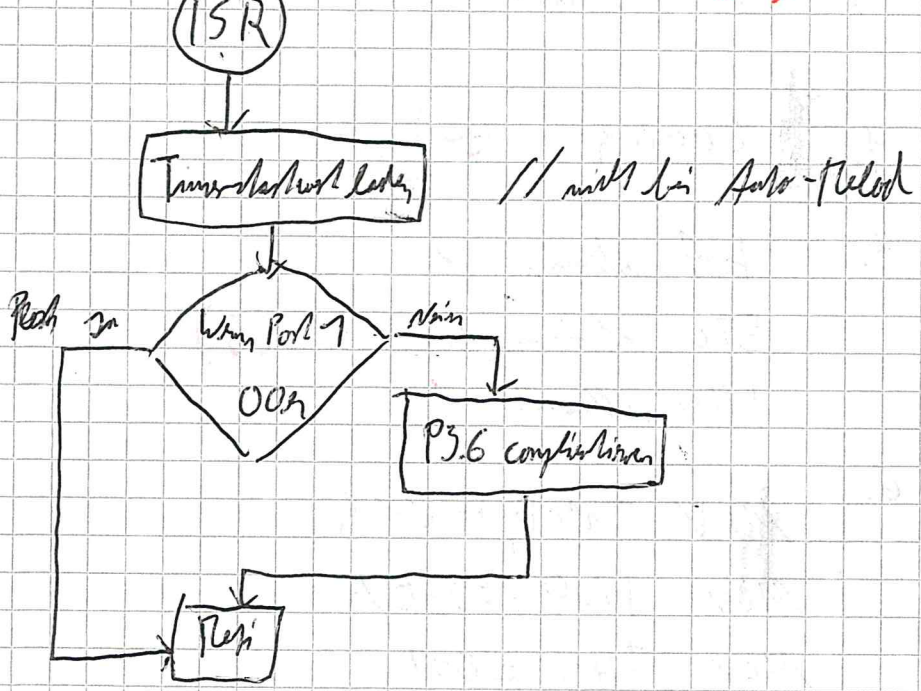
Es funktioniert ähnlich wie eine While-Schleife, die vom USER bestimmt werden kann.

(2)

3) 1.


Start
EA aktivieren
ET0 aktivieren
P1 auf Eingabe setzen
Modus Timer 0
Startwert in Timer schreiben
Timer starten
unendlich warten

(2)


ISR
Timer-Startwert laden
// nicht bei Auto-Reload
Wert Port 1
00h
Nein
Ja
P3.6 complementieren
Reti

2. $\frac{1}{100Hz} = 0{,}01\,s$

$\frac{0{,}01s}{2} = 0{,}005s$ // da nur die Hälfte der Zeit computiert werden soll

(3)

3. $+t \cdot (Losz/12) = 0{,}005_s \cdot (11{,}0592\,MHz/12)$

$= 4608 > 256$

$\Rightarrow < 65536$

Also Also Modus 1 16-Bit (2)

4. 000B h (2)

5. EA aktivieren ✓
ET0 aktivieren ✓ (3)
TR0 aktiviert ✓

6.
```
MOV TMOD, #01h
MOV TL0, #0EEh
MOV TH0, #00h
```

$65536 - 4608 = 60928\,d$ (3)
EE 00

7. Der Timer muss nach erhalt der ISR nicht angestellt werden und auch nicht, bei dem setzen des neuen Zählwerts, da so die Frequenz für die Ausgabe constant bleibt und keine Verzögerung durch weiteres Rechnen in der ISR entsteht. (1,5)

8 0006

$$2^{16} - t \cdot \left(\frac{f_{osc}}{12}\right)$$

$$2^{16} = t \cdot \left(\frac{f_{osc}}{12}\right)$$

$$\frac{2^{16}}{\left(\frac{f_{osc}}{12}\right)} = t$$

$$t = 0{,}077\,s$$ (3)

diese Zeit kann mit dem Modus 1 überbrückt werden.

9.

```
ORG 000BH
  SJMP ISR

ISR:
  MOV TL0, #EEH
  MOV TH0, #00H

  MOV A, P0
  CJNE A, #00, cont
  CPL P3.6
cont:
  RETI
```

(3)

5.) 1.

Timer 1 auf Modus 2 um Baudrate zu erzeugen

Baudrate von 2400 einstellen

Serielle Übertragung auf 8-Bit UART

12MHz !!!

Das Programm wartet auf ein Eingabe auf der serielle Schnittstelle und gibt die Daten dann auf dem P1 aus (4)

2.

Serielle Schnittstelle = Modus 1

Timer 1 = Modus 2 (4)

3. Es wird eine Standard Baudrate erzeugt von 2400 $\frac{Bit}{s}$ s.o. (3)

4.

-"- ; wie geht

MOV SCON, #00D0h

-"-

work:

MOV A, SBUF

~~MOV~~ ~~SBUF~~

MOV R0, A

MOV P0, ~~R0~~ R0B8

~~MOV R0, P~~

~~MOV A, R0~~

~~SBUF, A~~

~~SBUF, 37~~

~~jmp~~

siehe Rückseite

7

```
~~JBP, j1~~
~~JB F0, j21~~
~~CLR P2.7~~
~~j1:~~
~~JNB~~
~~JBP~~

        JB P, j1
        MOV A#00h
Play:   JB F0, j2
        MOV R0, #00h
        SJMP cmp
        ~~CJNE~~
j1:     ~~MOV R1,~~
        ~~MOV R1, #01h~~
        MOV A, #01h
        SJMP cmp:play
j2:     MOV R0, #01h

cmp:    ~~CJNE A,R1~~
        ~~CJNE,00h~~
        CJNE ~~00h~~ A, 00h, clrp
        CLR P2.7
        SJMP return
clrp:   SETB P2.7

return: CLR R1
        SJMP wait
```

work OK (4)

6.)

1.

~~Das Byte Byte sich verändert~~

Nach dem ACALL werden 2 Bytes für die Rücksprungadresse auf den Stack gelegt und der gespeicherte ACC und PSW kommen darunter und können so nicht angelegt werden. (3)

2. Anstatt des Aufrufs wird die Rücksprungadresse gezogen und nach dem RET (bzw. wäre RETI) wird die Rücksprungadresse aus dem ACC und PSW gebildet.
↳ springt ins Nichts. (4)

3. Indem die Rücksprungadresse in ein Register geladen wird und dann wieder auf den Stack gelegt, am Ende. (3)

Anstatt des ACALL ein SJMP, dann wird der Stack nicht belastet und kein RETI sondern ein weiter SJMP zu 000Ch, am besten in Code.

4.

SP	Inhalt	
07h	/	
08h	Durchgangsbereich	High-Beph
09h	Pau-11-	Low-Beph
0Ah		

SP	Inhalt
07h	/
08h	12h ✓
09h	00h ✓
~~0Ah~~	~~Durchgangsbereich~~ ~~High-Beph~~
~~0Bh~~	-11- Low-Beph
~~0A~~	warum durchgestrichen?

(3)

~~07h~~