Maschinenorientierte Programmierung Klausur, 01.02.2013, WS 2012	Name:
	Matrikelnr.:

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.

_____

(Unterschrift)

Ab hier bitte keine Eintragungen vornehmen!

Aufgabe:	1	2	3	4	5	6	**Summe**
Punkt(e):	24	22	12	24	12	26	120
Erreicht:	21	13,5	12	18	12	17	93,5

Note: 1,0

Glückwunsch!

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 101h...

○ entspricht 175d ☒ entspricht 257d ○ passt in 8-Bit Register ☒ passt nicht in 8-Bit Register

Die Bitbreite des internen Datenbusses beim 8051 beträgt...

○ 4 Bit ☒ 8 Bit ○ 16 Bit ○ 32 Bit

Wieviele Registersätze mit jeweils 8 Registern existieren beim 8051?

☒ 4 ○ 8 ○ 128 ○ 256

Die Ausführung von Multiplikations- und Divisionsbefehlen beim 8051 ist erlaubt...

○ für alle Register ○ nur für das Akkumulator-Register ○ für R0 bis R7 der ersten Registerbank
☒ für Akkumulator- und B-Register

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

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

7 (-2)

Default-mäßig ist welche Registerbank als aktive Registerbank voreingestellt?

☒ Registerbank 0 ○ Registerbank 1 ○ Registerbank 2 ○ Registerbank 3

Das Interrupt-System des (Original) 8051 ist gekennzeichnet durch...

○ vier Interruptquellen und zwei Prioritätsebenen ☒ fünf Interruptquellen und zwei Prioritätsebenen
○ fünf Interruptquellen ohne Prioritätsvergabe ○ sechs Interruptquellen und beliebige Prioritäten

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

○ Registerbank 0 ☒ Registerbank 1 ○ Registerbank 2 ○ Registerbank 3

Zu den peripheren Einheiten des 8051 gehört auch der Timer. Welche Aussage ist richtig?

○ Timer kann nur dann laufen, wenn Interrupts aktiviert sind ○ Timer läuft immer ☒ Timer kann laufen, wenn Prozessor Befehle ausführt ○ Timer läuft nur, wenn Anweisung an die zugehörige Interruptvektoradresse geschrieben wurde

Damit Timer / Counter 0 als Counter aktiviert wird, muß bei gesetztem Gate Bit...

☒ TR0 auf 1 sein und INT0 auf High-Pegel liegen ○ TR0 auf 0 sein und T0 auf H-Pegel liegen ○ TR1 auf 0 sein und T1 auf Low-Pegel liegen ○ TR0 auf 1 sein und INT0 auf Low-Pegel liegen

Was konfiguriert das TCON Register?

(-1)

☒ Timer ○ Baudrate ○ serielle Schnittstelle ☒ Interrupts

Die ORG- und END-Direktiven werden bei der Assemblierung...

○ übersetzt als OPCODES ○ ignoriert ☒ verwendet zur Steuerung des Assemblierungsvorgangs
○ nicht übersetzt, da es fehlerhafte Anweisungen sind

Aufgabe 2 (22 Punkte)

Umgang mit dem 8051

Geben Sie für die folgenden Assemblerbefehle an, aus welchem Speicherbereich die Quelldaten kommen und in welchen Speicherbereich die Zieldaten gespeichert werden. Als Kürzel verwenden Sie ID für den "internen Datenspeicher", ED für den "externen Datenspeicher" und PS für den "Programmspeicher",
Beispiel $ED \rightarrow ID$

a) MOV 40h, R1 ~~ID -> ID~~ PS -> ID ✓ ID => ID
b) MOVC A, @A+PC ~~ID -> ID PS~~ PS -> PS ✓ (0,5) PS => ID
c) MOV A, R2 ~~ID -> PS~~ PS -> PS ✓ ID => ID
d) MOVX A, @R5 ED -> PS ✓ (0,5) ED => ID
e) MOV DPTR, #06h PS -> PS PS => ID

Benennen Sie für die obenstehende Aufgabe jeweils die Adressierungsarten und schreiben Sie diese dahinter.

zu a) direkte Adr. ✓
zu b) indizierte Adr. ✓
zu c) Register-Adr. ✓
zu d) Register-indirekte Adr. ✓ (5)
zu e) unmittelbare Adr. ✓

Ermitteln Sie die Anzahl der ausgeführten Schleifendurchläufe für folgenden Code.

```
MOV R2,#4
abc:
MOV R1,#2Ø
def:
MOV RØ,#7
ghi:
DJNZ RØ,ghi
DJNZ R1,def
DJNZ R2,abc
```

$7 + 7 \cdot 20 + 4 \cdot 7 \cdot 20 = 707$ (✓) (1)

Der Inhalt des Stack-Pointers ist 30h. Dann wird das folgende Programmstück abgearbeitet.

```
SETB RS1
CLR RSØ
MOV RØ,#ØA6h
MOV R1,#ØB1h
MOV A,R1
XRL A,RØ
ACALL Marke

Marke:
PUSH ACC
PUSH PSW
PUSH Ø
PUSH 1
RET
```

10110001 R1
10100110 XRL RØ
00010111

32h
~~31h~~ 33h
~~32h~~ 34h
~~33h~~ 35h
~~34h~~ 36h
34h -> SP am Ende (2)

36h 01h ←
35h 00h
34h 10h (5)
33h 17h
32h Rücksprungadr. lowByte
31h Rücksprungadr. highByte
~~30h~~

Geben Sie den Inhalt das Stack-Speicherbereichs an mit jeweiliger Datenadresse (soweit bekannt). Welchen Inhalt hat jetzt der Stack-Pointer?

Aufgabe 3 (12 Punkte)

Fragen zu Wissen und Anwendung

1. Vergleicht man C und Assembler, welche der beiden Sprachen ist effizienter im Hinblick auf die Code-Generierung (d.h. die Menge an für das Programm verwendeten ROM-Speicher)?

   Assembler ist effizienter, da Assemblercode für das Programm optimiert ist. Bei C generiert der Compiler Code, der durch allgemeine Übersetzungen nicht auf den Verwendungszweck optimiert sein muss. (4)

2. Berechnen Sie die gesamte Anzahl von übertragenen Bits, wenn 100 Seiten Textzeichen gesendet werden unter Verwendung der asynchronen seriellen Datenübertragung. Als Konfiguration sei eine Datengröße von 8 Bit, die Verwendung von 1 Stop Bit und keine Parität angenommen. Jede Seite hat 80 · 50 Textzeichen (ASCII-Daten).

   80 · 50 = 4000 Textzeichen, 4000 · 100 Seiten = 400k Tz.
   pro Zeichen: 1 Startbit, 8 Datenbits, 1 Stoppbit → 10 Bit/Tz
   ↔ 400k Tz · 10 Bit/Tz = 4M Bit ✓ (4)

3. Wie lange wird die Datenübertragung dauern, wenn die Baudrate 4800 ist?

   $\frac{4M\ Bit}{x} = \frac{4800\ Bit}{s}$ $\quad x = \frac{4M\ Bit\ s}{4800\ Bit} = \frac{833,\overline{3}\ s}{}$ ✓ (4)

Aufgabe 4 (24 Punkte)

Programmierung in Assembler

Ein Programm soll durch eine negative Flanke an Port 3, Bit 2 unterbrochen werden. Nach Auftreten der Flanke soll an die Adresse 60h verzweigt werden.

1. Geben Sie stichpunktartig alle Maßnahmen an, damit diese Unterbrechung wie beschrieben durchgeführt wird.
2. Wäre es möglich das Programm statt durch eine Flanke durch ein andere Ereignis am selben Pin zu unterbrechen? Wenn ja: Wie wird dies realisiert?
3. Welche Befehlsfolge muß ab Adresse 60h mindestens stehen, damit das unterbrochene Programm wieder fortgesetzt werden kann?
4. Entwickeln Sie nun den Assembler-Code für das Programm und kommentieren Sie Ihren Code. Das zu unterbrechende Programm kann durch eine Dauerschleife dargestellt werden.
5. Angenommen es kommen mehrere Flanken am genannten Portpin an. Wie schnell dürfen die Flanken erscheinen, damit sie noch fehlerfrei erkannt werden können? Die Oszillatorfrequenz betrage 11,0592 MHz.

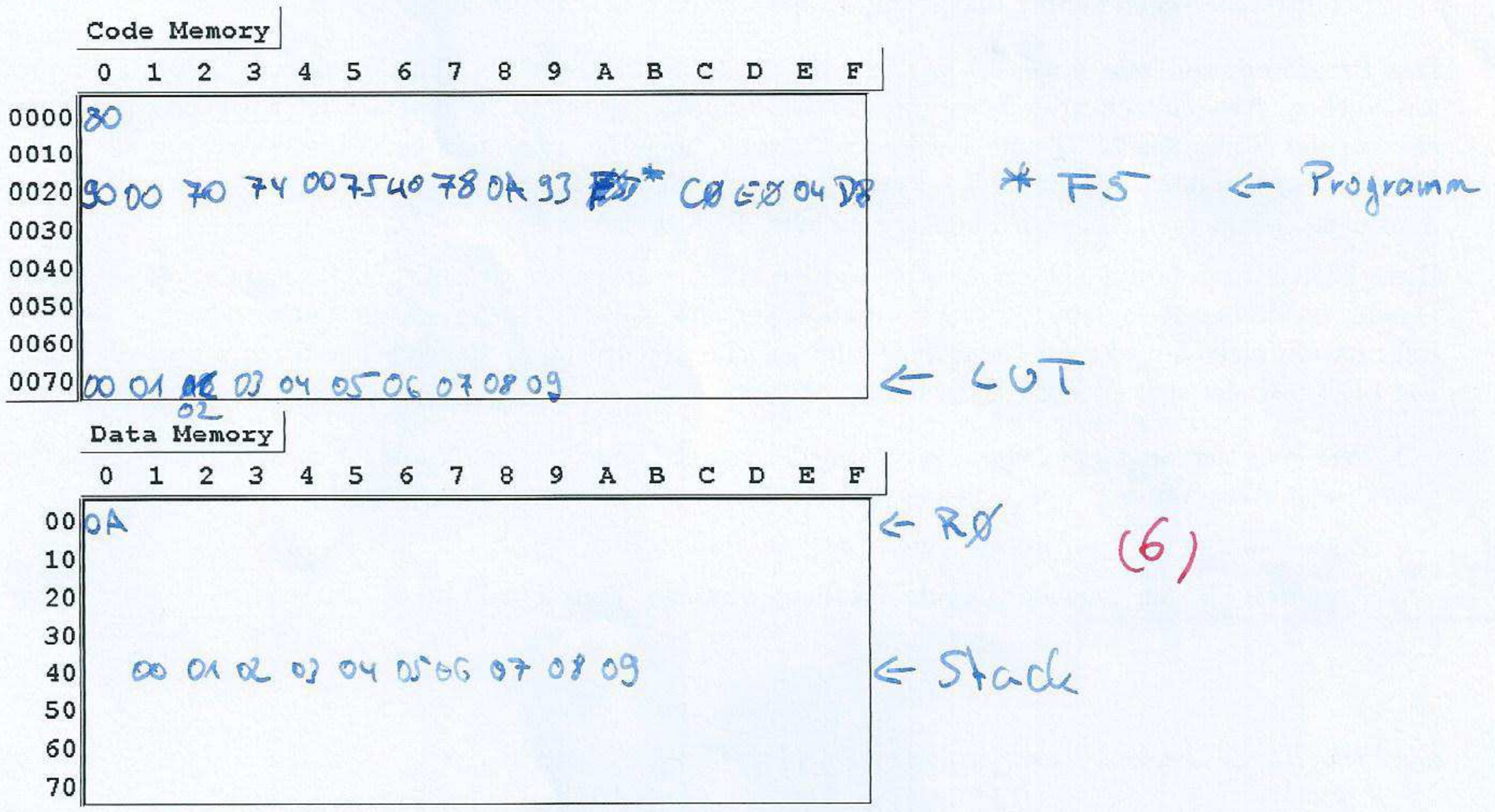
Aufgabe 5 (12 Punkte)

Speicherverwaltung des 8051 Mikrocontroller

Betrachten Sie folgendes Assemblerprogramm.

```
ORG 00h
sjmp start

ORG 20h
start:
MOV DPTR,#70h
MOV A, #00h
MOV SP, #40h
MOV R0, #0Ah

loop:
MOVC A, @A+DPTR
MOV P1, A
PUSH ACC
INC A
DJNZ R0, loop

ORG 70h
table:
DB 0,1,2,3,4,5,6,7,8,9

END
```

1. Analysieren Sie das Programm und beschreiben Sie kurz die Funktionalität. Was tut das Programm?
2. Ließe sich statt dem MOVC-Befehl auch der MOV-Befehl verwenden? Ließe sich statt dem Befehl PUSH ACC auch PUSH A schreiben? Ließe sich statt dem Befehl INC A auch INC ACC schreiben? Wenn ja, warum? Wenn nein, warum nicht? Begründen Sie jede Ihrer Antworten plausibel.
3. Geben Sie nun an wo genau die einzelnen Segmente (Programmcode, Stack, Lookup-Table, etc.) im Speicherabbild zu finden sind und markieren Sie dies eindeutig in den unten gegebenen Speicherschemata.



Aufgabe 6

(26 Punkte)

Arbeiten mit der seriellen Schnittstelle

Folgendes Programm für die Kommunikation über die serielle Schnittstelle ist gegeben. Die Oszillatorfrequenz beträgt 11,0592 MHz.

```
ORG 00h
MOV SCON, #50h
ORL PCON, #00h

MOV TMOD, #20h
MOV TH1, #0F4h
MOV TL1, #0F4h
SETB TR1

Marke1:
JNB RI, Marke1
CLR RI
MOV A, SBUF
MOV SBUF, A

Marke2:
JNB TI, Marke2
CLR TI
JMP Marke1
END
```

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 betrieben wird, wie der Timer konfiguriert ist und welche Baudrate sich letztlich einstellt.
3. Wie verhält sich im eingestellten Modus die Netto- zur Bruttodatenrate?
4. Statt mit Polling könnte man das Programm auch mit einer Interrupt Service Routine realisieren. Bestimmen Sie, ob die Interrupt Service Routine direkt in dem Interrupt-Vektorraum für die serielle Schnittstelle liegen könnte und erläutern Sie warum bzw. warum nicht.

Das Programm soll nun erweitert werden um die Funktionalität der Paritätskontrolle. Dazu sei davon auszugehen, dass eintreffende Daten mit der Einstellung "'Even Parity"' versendet wurden. Sie enthalten also an der Stelle des MSB eine Parity-Information über die unteren 7 Bit. Dies setzt voraus, dass die Nutzinformation nur 7 Bit umfasst, beispielsweise bei ASCII-Zeichenübertragung. (Hinweis: Die Betriebsart wird nicht geändert, d.h. es wird kein zusätzliches Bit übertragen!)

Diese Parity-Information soll ausgewertet werden und verglichen werden mit der Parität der angekommenen Daten. Ist die mitgesendete Parity-Information korrekt, soll das Programm normal weiterarbeiten. Ist die Information nicht korrekt, soll eine LED zum Leuchten gebracht werden und das Programm soll stoppen. Die LED befindet sich an Port 0, Bit 0 (Low Active).

5. Wie lässt sich am einfachsten die Parität der eingetroffenen Daten ermitteln? Dies müssen Sie tun, um diesen Wert mit dem übertragenen zu vergleichen.
6. Entwerfen Sie für die Aufgabe einen Programmablaufplan.
7. Ergänzen Sie den gegebenen Code um die gewünschte Funktionalität.

Klausur

01.02.13

Aufgabe 4)

1.

- Interrupts aktivieren (EA = 1)
- externen Interrupt 0 aktivieren (EX0 = 1)
- ~~Interrupt auf Flankentriggerung konfigurieren (IT0 = 1)~~ (IT0 = 1)
- an Adresse 03h im Programmspeicher einen Sprungbefehl zur Adresse 60h verwenden

(4) ✓

2.

Ja wäre es und zwar durch einen ~~negt~~ negativen Pegel an P3.2.
Hierfür muss IT0 = 0 gesetzt werden und beim Auslösen des ersten Interrupts, muss der externe Interrupt 0 oder alle Interrupts deaktiviert werden, damit nicht ~~alle~~ wiederholt der Interrupt ausgelöst wird. Wenn später dieser wieder verwendet werden soll, ~~muss~~ muss er wieder entsprechend deaktiviert werden.

(4) ✓

3.

Sprungmarke: RETI ✓

(4)

4.

```
        ORG   000h
        SJMP  start
        ORG   003h
        SJMP  ISR

        ORG   030h
start:  SETB  EX0      ; ext. Int. 0 aktiv
        SETB  IT0      ; Flankentriggerung
        CLR   IE0      ; Flag clearen
        SETB  EA       ; alle Int. aktiv

main:   SJMP  main

(6)     ORG   060h
ISR:    RETI           ; zurück zum Hauptpr.
```

5.

11,0592 MHz^{-1} ≈ 90 ns

da zur Abtastung zwei mal geprüft werden muss: 90 ns / 2 = 45 ns

↳ Flanken dürfen nicht schneller als nach 45 ns kommen ✓

(0)

Flanke wird nach 2 Maschinenzyklen erkannt.

T = 1/f = 1/11,0592MHz
2T = 2,17µs

Aufgabe 5

1. Das Programm holt mittels indizierter Adressierung die Zahlen 0 bis 9 aus einer Loop-up table und gibt diese auf P1 aus. ~~Gleichzeitig~~ Außerdem wird der Index der entsprechenden Zahl auf den Stack ab der Adresse 41h gelegt. ✓ (3)

2.
- MOV statt MOVC: Nein,
denn mit MOVC wird auf den Codespeicher zugegriffen und um ~~so~~ die indizierte Adressierung zu realisieren wird MOVC benötigt

- PUSH A statt PUSH ACC: Nein,
denn der Befehl PUSH erwartet eine Adresse. A steht jedoch für den Akkumulator und Befehle mit A sind optimierte Befehle. Hinter ACC steckt jedoch die Adresse E0h der Akkumulators und kann somit für PUSH verwendet werden.

- INC ACC statt INC A: Ja,
denn INC kann sowohl optimiert für den Akku verwendet werden als auch für eine direkte Adresse (ACC entspricht E0h).
Vorteil von INC A. 1Byte Befehl
INC ACC 2Byte Befehl

✓ (3)

Aufgabe 6

1. Das Programm wartet auf gesendete Daten zum Empfangen und sendet anschließend ein Echo. Dies wird mittels Polling realisiert.

(3)

2.
- Modus serielle Schnittstelle:
  Modus 1 , 8-Bit UART
- Timer 1: Modus 2 - 8-Bit Reload
  Startwert: F4h = 244d

  <> Baudrate: 2400 ✓ (4)

3.
$$\frac{2400}{10\,Bit} = \frac{x}{8\,Bit} \qquad x = 1920 \quad \text{Netto}$$

=> Nettodatenrate entspricht 80% der Bruttodatenrate (3)

4.
```
ISR:      JB  RI  receive      3 Byte
          JB  TI  send         3 Byte
receive:  MOV A, SBUF          2 Byte
          ~~CLR RI~~ CLR RI
          ~~RETI~~
          MOV SBUF, A
send:     RETI
send:     CLR TI
          RETI
```
(4)

ISR könnte im Vektorraum liegen, da nach diesem kein weiterer Vektorraum folgt. Das Hauptprogramm muss nun entsprechend weit hinten im Speicher stehen.

5.

~~empfangene →~~

~~MOV A, SBUF~~

Paritätsbit mit übertragen (3)