Übungen zu
Grundlagen der Informatik für Studierende des Maschinenbaus
im Sommersemester 2001
(Dipl.-Inf. Jörg Barner)

Aufgabe 7: Assemblersprache
Übersetzen Sie das folgende einfache C-Programm in Assemblersprache!

```
/* Programm zu Aufgabe 7 */
extern void ausgabe(int);

void add(int *a, int b)
{
   *a+=b;
}

void main(void)
{
   int sum,i;

   sum = 0;
   for(i=1; i<10; i++) {
      if ( i/2*2 == i )
         add(&sum,i);
   }
   ausgabe(sum);
}
```

Das Unterprogramm `ausgabe` Sei an anderer Stelle definiert. Verwenden Sie zur Parameterübergabe den Kellerspeicher (Stack).
Geben Sie den Inhalt des Kellerspeichers zu dem Zeitpunkt an, an dem das Unterprogramm `add` zum ersten Male betreten wird.

Lösung:

```
; extern void ausgabe(int a)

            lad, #0           ;
            spi, @sum         ; sum=0

; for(i=1;i<10;i++) {

            lad, #1           ;
            spi, @i           ; i=1

loop:       lad, @i           ; i laden
            sub, #10          ;
            jgz, @end         ; i>=10, dann Ende
```

```
            lad, @i            ; i laden
            div, #2            ; i/2
            mul, #2            ; i/2*2
            sub, @i
            jez, @act          ; i/2*2 ist gleich i, also Action
            jmp, @nact         ; i/2*2 ist ungleich i, also keine Action

act:        push, #sum         ; &sum auf Stack
            push, @i           ; i auf Stack
            jsr, @add          ; add(&sum,i)

nact:       lad, @i            ;
            add, #1            ;
            spi, @i            ; i++
            jmp, @loop         ;

; }

end:        push, @sum         ;
            jsr, @ausgabe      ;

            hlt                ;

; void add(int *a, int b)
; {

add:        pop, @radr         ; Ruecksprungadresse sichern
            pop, @b            ; Parameter b (int) vom Stack holen
            pop, @a            ; Parameter a (Pointer auf int) vom Stack holen
            lad, (@a)          ; *a in Akku laden
            add, @b            ; b addieren
            spi, (@a)          ; Ergebnis abspeichern in *a (*a+=b)
            push, @radr        ; Ruecksprungadresse zurueck auf Stack
            ret                ;

; }

; extern void ausgabe(int a)
; {

ausgabe:    pop, @radr         ;
            pop, @a            ;
            _print, @a         ;
            push, @radr        ;
            ret                ;
; }

sum:        block, 1
i:          block, 1
a:          block, 1
b:          block, 1
radr:       block, 1
```

Aufgabe 8: Programmiersprachliche Grundbegriffe und Syntaxdiagramme

a) Erläutern Sie die folgenden Begriffe:

- Syntax
- Semantik
- Pragmatik

b) Erstellen Sie ein Syntaxdiagramm, das den korrekten Aufbau einer `for`-Anweisung in `C` beschreibt.

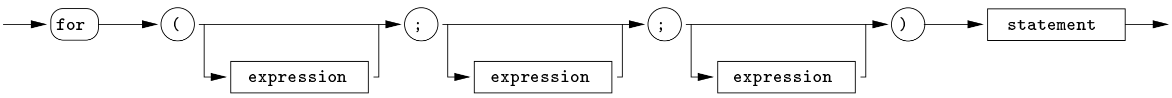
Lösung:

a) **Syntax**: Legt fest, welche Zeichenfolgen gültig sind (d.h. zur Sprache gehören, z.B. durch eine Chomsky-Grammatik.

Semantik: Regelt die Bedeutung der (syntaktisch korrekten) Zeichenfolgen, also wie sie zu interpretieren sind.

Pragmatik:

b) Syntaxdiagramm für eine `for`-Schleife in `C`:



Aufgabe 9: Grammatik
Gegeben sei die Sprache $L = \{a^n b a^n \mid n \geq 1\}$

a) Geben Sie eine Chomsky-Grammatik für die Sprache L an.

b) Stellen Sie die Sprache L in Backus-Naur-Form sowie als Syntaxdiagramm dar.

Lösung:

a) $G = (N \cup T, T, S, P)$ mit
$N = \{S, A\}$
$T = \{a, b\}$
$P = \{(S, aAa), (A, aAa), (A, b)\}$

b) Backus-Naur-Form: $S ::= a\langle A\rangle a$, $A ::= a\langle A\rangle a \mid b$.

Syntaxdiagramm: