Informatik Rechnerinterne Vorgänge: Programmstrukt. (Lsg.)

Gierhardt

1. Die Zahlen von 1 bis 10 sollen ausgegeben werden

   (a) absteigend mit einer `do while`-Schleife

```
1         JMP Anfang
2  k      DEF 0              ; int k
3  zehn   DEF 10             ; int zehn = 10
4  Anfang
5          LDA zehn
6          STA k             ; k = 10
7  Schleife                  ; do {
8          LDA k
9          OUT k             ;       print(k)
10         DEC
11         STA k             ;       k = k-1
12         JNP Schleife      ; } while (k > 0)
13         END
```

   (b) aufsteigend mit einer `do while`-Schleife

```
1         JMP Anfang
2  k      DEF 00             ; int k
3  eins   DEF 01             ; int eins = 1
4  zehn   DEF 10             ; int zehn = 10
5  Anfang
6          LDA eins
7          STA k             ; k = 1
8  Schleife                  ; do {
9          LDA k
10         OUT k             ;       print(k)
11         INC
12         STA k             ;       k = k+1
13         SUB zehn
14         JNP Schleife      ; } while (k - 10 <= 0)
15         END
```

   (c) absteigend mit einer `while`-Schleife

```
1         JMP Anfang
2  k      DEF 0                ; int k
3  zehn   DEF 10               ; int zehn = 10
4  Anfang
5          LDA zehn
6          STA k               ; k = 10
7  SchleifenAnfang             ; while (k > 0)
8          JNP SchleifenEnde   ; {
9          LDA k
10         OUT k               ;       print(k)
11         DEC
12         STA k               ;       k = k-1
13         JMP SchleifenAnfang ;
14 SchleifenEnde               ; }
15         END
```

(d) aufsteigend mit einer `while`-Schleife

```
1         JMP  Anfang
2  k      DEF  0                  ; int k
3  eins   DEF  1                  ; int 1 = 1
4  zehn   DEF  10                 ; int zehn = 10
5  Anfang
6          LDA  eins
7          STA  k                 ; k = 1
8  SchleifenAnfang                ; while (k-10 <= 0)
9          SUB  zehn
10         JPL  SchleifenEnde     ; {
11         LDA  k
12         OUT  k                 ;       print(k)
13         INC
14         STA  k                 ;       k = k+1
15         JMP  SchleifenAnfang   ;
16 SchleifenEnde                  ; }
17         END
```

2. Die Zahlen von 1 bis zu einer einzulesenden Zahl sollen aufaddiert und die Summe ausgegeben werden.

```
1         JMP  Anfang
2  null   DEF  000                ; int null = 0
3  summe  DEF  000                ; int summe
4  k      DEF  0                  ; int k
5  Anfang
6          LDA  null              ;
7          STA  summe             ; summe = 0
8          INM  k                 ; k = in.readInt();
9  WhileAnfang
10         LDA  k                 ;
11         JNP  WhileEnde         ; while (k > 0)
12         LDA  summe             ; {
13         ADD  k                 ;
14         STA  summe             ;       summe = summe + k
15         LDA  k                 ;
16         DEC                    ;
17         STA  k                 ;       k = k - 1
18         JMP  WhileAnfang       ;
19 WhileEnde                      ; } //end while
20         OUT  summe
21         END
```

```
1         JMP  Anfang
2  null   DEF  000                ; int null = 0
3  summe  DEF  000                ; int summe
4  k      DEF  0                  ; int k
5  Anfang
6          INM  k                 ; void main() { k = in.readInt()
7          JSR  Addiere           ;     addiere()
8          OUT  summe             ;     print(summe)
9          END                    ; }
10 addiere                        ; void addiere()
11         LDA  null              ; { summe = 0
12         STA  summe             ;
13 WhileAnfang
14         LDA  k                 ;
15         JNP  WhileEnde         ;   while (k > 0)
16         LDA  summe             ;   {
17         ADD  k                 ;
18         STA  summe             ;       summe = summe + k
19         LDA  k                 ;
20         DEC                    ;
21         STA  k                 ;       k = k - 1
22         JMP  WhileAnfang       ;
23 WhileEnde                      ;   } end while
24         RTN                    ; } end addiere
```

3. Die Struktur von `if else` ist in möglichst allgemeiner Form in eine Kombination von Jump-Befehlen zu übersetzen.

```
1       LDA x           ; if (x > 0)
2       JNP else        ;      {
3       OUT x           ;        print(x)
4       JMP endif       ;      }
5  else                 ; else {
6       INC             ;
7       STA x           ;        x++
8       OUT x           ;        print(x)
9  endif                ;      }
10      ...             ;
```

4. Eine Zahl wird eingelesen und festgestellt, ob sie gerade oder ungerade ist. Bei einer geraden Zahl soll 0, sonst 1 ausgegeben werden. Die Ermittlung des Restes bei der Division durch 2 soll in einem Unterprogramm gemacht werden.

```
1      JMP Anfang
2  zahl DEF 0
3  zwei DEF 2
4  rest DEF 0
5
6  Anfang
7      INM zahl
8      JSR Rest_bilden
9      OUT rest
10     END
11
12 Rest_bilden
13     LDA zahl
14     STA rest
15 Schleife
16     SUB zwei
17     STA rest
18     JPL Schleife    ; solange Rest > 0
19                     ; jetzt ist Rest 0 oder -1
20     NEG             ; jetzt ist Rest 0 oder 1
21     STA rest
22     RTN
```

5. Zwei Zahlen werden eingelesen und die größere der beiden Zahlen wird ausgegeben.

```
1       JMP Anfang
2  X DEF 0
3  Y DEF 0
4
5  Anfang
6       INM X
7       INM Y
8
9       LDA X
10      SUB Y     ; if (x - y > 0)
11      JNP else
12      OUT X     ;      print(x)
13      JMP endif
14 else
15      OUT Y     ; else print(y)
16 endif
17      END
```

6. Ein Programm soll zwei positive Zahlen einlesen, in einem Unterprogramm `mult` das Produkt berechnen und das Ergebnis ausgeben. Zuerst ist dazu ein entsprechendes Java-Programm zu schreiben, wobei darin Multiplikation nicht erlaubt ist. Dieses Java-Programm ist dann in ein Maschinen-Programm „zu Fuß zu kompilieren“.

```
1  class Vielfaches
2  { static int zahl1,
3               zahl2,
4               ergebnis=0;
5
6    static void multipliziere()
7    {  while (zahl2>0)
8       { ergebnis = ergebnis + zahl1;
9         zahl2 = zahl2 - 1;
10      }
11   }
12
13   public static void main(String args[]) // Hauptprogramm
14   {  Out.println("Bitte ersten Faktor eingeben: ");
15      zahl1=In.readInt();
16      Out.println("Bitte zweiten Faktor eingeben: ");
17      zahl2=In.readInt();
18      multipliziere();
19      Out.println("Das Ergebnis ist "+ergebnis);
20   } // Ende von main
21 } // Ende von class Vielfaches
22
23           JMP  Anfang
24                              ;
25 Summand   DEF  04           ; int     summand
26 Ergebnis  DEF  00           ; int     ergebnis
27 WieOft    DEF  10           ; int     wieOft
28
29                              ; public void main(...)
30 Anfang                      ; {
31           INM  Summand       ;   summand = In.readInt();
32           INM  WieOft        ;   wieoft  = In.readInt();
33           JSR  Addiere       ;   addiere();
34           OUT  Ergebnis      ;   print(ergebnis)
35           END                ; }
36
37 Addiere                     ; public void  addiere()
38                              ; {
39 DO
40           LDA  Ergebnis      ;   do {
41           ADD  Summand       ;        ergebnis = ergebnis + summand;
42           STA  Ergebnis      ;
43           LDA  WieOft        ;
44           DEC                ;        wieOft = wieOft - 1;
45           STA  WieOft        ;
46           JPL  DO            ;      } while (wieOft > 0)
47           RTN                ; }
```

Schwierigere Variante: Die beiden Zahlen können negativ oder nicht-negativ sein.

7. Die Multiplikation kann auch rekursiv über Additionen erfolgen. Schreibe dazu ein Java-Programm ohne Parameterübergabe und entwirf mit Hilfe dieses Programmes ein DCL-Maschinen-Programm.

```
1  class VielfachesRekursiv
2  { static int zahl1, zahl2, ergebnis=0;
3
4    static void addiere()
5    {  ergebnis = ergebnis + zahl1;
6       zahl2--;
7       if (zahl2>0) addiere();
8    }
9
10   public static void main(String args[]) // Hauptprogramm
11   {  Out.println("Bitte ersten Faktor eingeben: "); zahl1=In.readInt();
12      Out.println("Bitte zweiten Faktor eingeben: "); zahl2=In.readInt();
13      addiere();
14      Out.println("Das Ergebnis ist "+ergebnis);
15   } // Ende von main
16 } // Ende von class VielfachesRekursiv
17
18 ; ****************************************************************
19 ; *    Rekursion mit DC (einfache Addition bzw. Multiplikation)   *
20 ; *    ohne Parameteruebergabe                                    *
21 ; ****************************************************************
22         JMP Anfang
23                          ;
24 Summand  DEF 04           ; int     summand
25 Ergebnis DEF 00           ; int     ergebnis
26 WieOft   DEF 10           ; int     wieOft
27
28                          ; public void main(...)
29 Anfang                   ; {
30         INM Summand      ;   summand = In.readInt();
31         INM WieOft       ;   wieoft  = In.readInt();
32         JSR Addiere      ;   addiere();
33         OUT Ergebnis     ;   print(ergebnis)
34         END              ; }
35
36 Addiere                  ; public void  addiere()
37                          ; {
38         LDA Ergebnis     ;
39         ADD Summand      ;      ergebnis = ergebnis + summand;
40         STA Ergebnis     ;
41         LDA WieOft       ;
42         DEC              ;      wieOft = wieOft - 1;
43         STA WieOft       ;
44         JNP EndIf        ;      if (wieOft > 0)
45         JSR Addiere      ;          addiere();
46 EndIf
47         RTN
48                          ; } (* addiere *)
```

Warum kann diese rekursive Variante der Multiplikation nicht alle im DC erlaubten Faktoren verarbeiten?

Bei zu großen Faktoren ergibt sich ein Stapelüberlauf (stack-overflow). Das kann man im DC sichtbar machen!