

Vorlesungsaufgaben

Geben Sie die Lösungen zu den Aufgaben aus der Assembler-Vorlesung ab. Bearbeiten Sie erst danach die folgenden Aufgaben auf diesem Übungsblatt.

(a) Folie 37/3,4

(i) Bestimmung der Summe der ersten n Zahlen (iterativ).

```
1  -- Bestimmung der Summe der ersten n Zahlen (iterativ)
2
3  summe_iterativ:
4  SEG
5
6          JUMP einstieg
7
8  -- erg R5
9  -- n R4
10
11 -- while (n > 0)
12 solange:  CMP W R4, I 0
13           JEQ abschluss
14           -- erg = n + erg;
15           ADD W R4, R5
16           -- n--;
17           SUB W I 1, R4
18           JUMP solange
19
20 einstieg:  MOVE W n, R4
21           -- int erg = 0;
22           MOVE W I 0, R5
23           JUMP solange
24
25 -- Das Ergebnis sollte 28 sein, siehe R5.
26 abschluss:  HALT
27
28 -- int n = 7;
29 n:          DD W 7
30
31 -- n:          DD W 0 -- 0
32 -- n:          DD W 1 -- 1
33 -- n:          DD W 2 -- 3
34 -- n:          DD W 3 -- 6
35 -- n:          DD W 4 -- 10
36 -- n:          DD W 5 -- 15
37 -- n:          DD W 6 -- 21
38 -- n:          DD W 7 -- 28
39 -- n:          DD W 8 -- 36
40 -- n:          DD W 9 -- 45
41 -- n:          DD W 10 -- 55
42
43 END
44
45 public class SummeIterativ {
46
47     public static int summe(int n) {
48         int erg = 0;
49         while (n > 0) {
50             erg = n + erg;
51             n--;
52         }
53         return erg;
54     }
55 }
```

```

13
14 public static void main(String[] args) {
15     int n = 7;
16     System.out.println(summe(n)); // 28
17 }
18 }

```

(ii) Bestimmung der n -ten Fibonaccizahl (iterativ).

```

1  -- Bestimmung der n-ten Fibonaccizahl (iterativ)
2
3  -- n          R2
4  -- vorletzte  R3
5  -- letzte     R4
6  -- erg        R5
7
8  fibonacci_iterativ:
9  SEG
10
11      JUMP einstieg
12
13  solange:  -- while (n > 1)
14             CMP W R2, I 1
15             JLE abschluss
16             -- erg = letzte + vorletzte;
17             ADD W R3, R4, R5
18             -- vorletzte = letzte;
19             MOVE W R4, R3
20             -- letzte = erg;
21             MOVE W R5, R4
22             -- n--;
23             SUB W I 1, R2
24             JUMP solange
25
26  kl_gleich_eins: MOVE W R2, R5
27                  JUMP abschluss
28
29  einstieg:  MOVE W n, R2
30             -- if (n <= 1) return n;
31             CMP W R2, I 1
32             JLE kl_gleich_eins
33
34             -- int vorletzte = 0;
35             MOVE W I 0, R3
36             -- int letzte = 1;
37             MOVE W I 1, R4
38             -- int erg = 0;
39             MOVE W I 0, R5
40             JUMP solange
41
42  abschluss:  HALT
43
44  n:          DD W 7
45
46  -- n:       DD W 0 -- 0
47  -- n:       DD W 1 -- 1
48  -- n:       DD W 2 -- 1
49  -- n:       DD W 3 -- 2
50  -- n:       DD W 4 -- 3
51  -- n:       DD W 5 -- 5
52  -- n:       DD W 6 -- 8
53  -- n:       DD W 7 -- 13

```

```

53  -- n:          DD W 8 -- 21
54  -- n:          DD W 9 -- 34
55  -- n:          DD W 10 -- 55
56  END

3  public class FibonacciIterativ {
4
5      static int fib(int n) {
6          if (n <= 1)
7              return n;
8
9          int vorletzte = 0;
10         int letzte = 1;
11         int erg = 0;
12
13         while (n > 1) {
14             erg = letzte + vorletzte;
15             vorletzte = letzte;
16             letzte = erg;
17             n--;
18         }
19         return erg;
20     }
21
22     public static void main(String[] args) {
23         int n = 7;
24         System.out.println(fib(n)); // 13
25
26         System.out.println(fib(0)); // 0
27         System.out.println(fib(1)); // 1
28         System.out.println(fib(2)); // 1
29         System.out.println(fib(3)); // 2
30         System.out.println(fib(4)); // 3
31         System.out.println(fib(5)); // 5
32         System.out.println(fib(6)); // 8
33         System.out.println(fib(7)); // 13
34         System.out.println(fib(8)); // 21
35         System.out.println(fib(9)); // 34
36         System.out.println(fib(10)); // 55
37     }
38 }

```

(b) Folie 57/1,2

- (i) zur Multiplikation zweier Zahlen unter Verwendung eines Unterprogramms

```

1  -- Programm zur Multiplikation zweier Zahlen unter Verwendung
   ↳ eines Unterprogramms
2
3  multiplikation:
4  SEG
5              MOVE W I H'0000FFFF', SP
6              JUMP einstieg
7
8  mult:        PUSHR
9              -- a * b
10             MULT W 64+!SP, 68+!SP, 72+!SP
11             POPR
12             RET
13

```

```

14  Einstieg:      MOVE W I -1, -!SP
15                MOVE W a, -!SP
16                MOVE W b, -!SP
17                CALL mult
18                ADD W I 4, SP
19                ADD W I 4, SP
20                -- Das Ergebnis sollte 49 sein.
21                MOVE W !SP+, R5
22                HALT
23
24  -- int a = 7;
25  a:             DD W 7
26  -- int b = 7;
27  b:             DD W 7
28  END

```

```

3  public class MultiplikationUnterprogramm {
4      public static int mult(int a, int b) {
5          return a * b;
6      }
7
8      public static void main(String[] args) {
9          int a = 7;
10         int b = 7;
11         System.out.println(mult(a, b)); // 49
12     }
13 }

```

(ii) Summe der ersten n Zahlen (rekursiv)

```

1  -- Summe der ersten n Zahlen (rekursiv)
2
3  summe_rekursiv:
4  SEG
5
6                MOVE W I H10000, SP
7                JUMP Einstieg
8
9  -- n R4
10 -- erg R5
11 summe:         PUSHR
12                MOVE W 64, !SP, R4
13                CMP W R4, I 0
14                JEQ ist_null
15                MOVE W I -1, -!SP
16                -- n - 1
17                SUB W I 1, R4, -!SP
18                CALL summe
19                ADD W I 4, SP
20                -- n + summe(n-1);
21                ADD W !SP+, R4
22                JUMP rueckgabe
23
24 ist_null:      MOVE W I 0, R4
25
26 rueckgabe:     MOVE W R4, 68, !SP
27                POPR
28                RET
29
30 Einstieg:     MOVE W I -1, -!SP
31                MOVE W n, -!SP
32                CALL summe

```

```

32      ADD W I 4, SP
33      -- Das Ergebnis sollte 28 sein.
34      MOVE W !SP, R5
35      HALT
36
37      int n = 7;
38      n:      DD W 7 -- 28
39
40      -- n:      DD W 0 -- 0
41      -- n:      DD W 1 -- 1
42      -- n:      DD W 2 -- 3
43      -- n:      DD W 3 -- 6
44      -- n:      DD W 4 -- 10
45      -- n:      DD W 5 -- 15
46      -- n:      DD W 6 -- 21
47      -- n:      DD W 7 -- 28
48      -- n:      DD W 8 -- 36
49      -- n:      DD W 9 -- 45
50      -- n:      DD W 10 -- 55
51      END
3      public class SummeRekursiv {
4          public static int summe(int n) {
5              if (n > 0) return n + summe(n - 1);
6              else return 0;
7          }
8
9          public static void main(String[] args) {
10             int n = 7;
11             System.out.println(summe(n)); // 28
12         }
13     }

```