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).

```
-- Bestimmung der Summe der ersten n Zahlen (iterativ)

summe_iterativ:
SEG
                JUMP einstieg

solange:        SUB W I 1, R0, R0
                CMP W R0, I 0
                JEQ ist_fertig

                ADD W I 1, R4
                ADD W R2, R4,R2

                JUMP solange

ist_null:       MOVE W I 0, R5
                JUMP abschluss

ist_eins:       MOVE W I 1, R5
                JUMP abschluss

-- Das Ergebnis sollte 28 sein.
ist_fertig:     MOVE W R2, R5
                JUMP abschluss

einstieg:       MOVE W I 1, R2
                MOVE W I 1, R4
                MOVE W n, R0
                CMP W R0, I 0
                JEQ ist_null
                CMP W R0, I 1
                JEQ ist_eins

abschluss:      HALT
-- int n = 7;
n:              DD W 7
END
```

```
public class SummeIterativ {

  public static int summe(int n) {
    int erg = 0;
    for (int i = 1; i <= n; i++) {
      erg = i + erg;
    }
    return erg;
  }

  public static void main(String[] args) {
    System.out.println(summe(7));
  }

```

```
}
```

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

```
-- Bestimmung der n-ten Fibonaccizahl (iterativ)

fibonacci_iterativ:
SEG
                JUMP einstieg

n:              DD W 10

einstieg:       MOVE W I 1, R2
                MOVE W I 0, R4
                MOVE W n, R0
                CMP W R0, I 0
                JEQ Null
                CMP W R0, I 1
                JEQ Eins

While:          SUB W I 1, R0, R0
                CMP W R0, I 0
                JEQ istEins
                ADD W R2, R4,R6
                MOVE W R4,R2
                MOVE W R6,R4
                JUMP While

istEins:        MOVE W R6, R5
                JUMP abschluss

Null:           MOVE W R4, R5
                JUMP abschluss
Eins:           MOVE W R2, R5

abschluss:      HALT
END
```

```
public class FibonacciIterativ {

}
```

(b) Folie 57/1,2

(i) zur Multiplikation zweier Zahlen unter Verwendung eines Unterprogramms

```
-- Programm zur Multiplikation zweier Zahlen unter Verwendung
↪  eines Unterprogramms

multiplikation:
SEG
                MOVE W I H'0000FFFF', SP
                JUMP einstieg

mult:           PUSHR
                -- a * b
                MULT W 64+!SP, 68+!SP, 72+!SP
                POPR
                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
5   }
```

(ii) Summe der ersten n Zahlen (rekursiv)

```
1   -- Summe der ersten n Zahlen (rekursiv)
2
3   summe_rekursiv:
4   SEG
5                   MOVE W I H'10000', SP
6                   JUMP einstieg
7
8   summe:          PUSHR
9                   MOVE W 64+!SP, R0
10                  CMP W R0, I 1
11                  JEQ ist_eins
12                  MOVE W I -1, -!SP
13                  SUB W I 1, R0, -!SP
14                  CALL summe
15                  ADD W I 4, SP
16                  ADD W !SP+, R0
17                  JUMP rueckgabe
18
19  ist_eins:       MOVE W I 1, R0
20
21  rueckgabe:      MOVE W R0, 68+!SP
22                  POPR
23                  RET
24
25  einstieg:       MOVE W I -1, -!SP
26                  MOVE W n, -!SP
27                  CALL summe
28                  ADD W I 4, SP
29                  -- Das Ergebnis sollte 28 sein.
30                  MOVE W !SP+, R5
31                  HALT
32
33  -- int n = 7;
34  n:              DD W 7
35  END
```

```
3  public class SummeRekursiv {
4
5  }
```