Vorlesungsaufgaben

(*Vorlesungsaufgaben*)

Stichwörter: Mehr-Adress-Befehl-Assembler

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).

Lösungsvorschlag

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

-- public static int summe(int n) {
--   int erg = 0;
--   while (n > 0) {
--     erg = n + erg;
--     n--;
--   }
--   return erg;
-- }

summeIterativ:
SEG
                JUMP einstieg

-- erg R5
-- n R4

-- while (n > 0)
solange:        CMP W R4, I 0
                JEQ abschluss
                -- erg = n + erg;
                ADD W R4, R5
                -- n--;
                SUB W I 1, R4
                JUMP solange

einstieg:       MOVE W n, R4
                -- int erg = 0;
                MOVE W I 0, R5
                JUMP solange

-- Das Ergebnis sollte 28 sein, siehe R5.
abschluss:      HALT

-- int n = 7;
n:              DD W 7

-- Tests

-- n:              DD W 0 -- 0
-- n:              DD W 1 -- 1
```

```
-- n:              DD W 2 -- 3
-- n:              DD W 3 -- 6
-- n:              DD W 4 -- 10
-- n:              DD W 5 -- 15
-- n:              DD W 6 -- 21
-- n:              DD W 7 -- 28
-- n:              DD W 8 -- 36
-- n:              DD W 9 -- 45
-- n:              DD W 10 -- 55
END
```

```
public class SummeIterativ {

  public static int summe(int n) {
    int erg = 0;
    while (n > 0) {
      erg = n + erg;
      n--;
    }
    return erg;
  }

  public static void main(String[] args) {
    int n = 7;
    System.out.println(summe(n)); // 28

    System.out.println(summe(0)); // 0
    System.out.println(summe(1)); // 1
    System.out.println(summe(2)); // 3
    System.out.println(summe(3)); // 6
    System.out.println(summe(4)); // 10
    System.out.println(summe(5)); // 15
    System.out.println(summe(6)); // 21
    System.out.println(summe(7)); // 28
    System.out.println(summe(8)); // 36
    System.out.println(summe(9)); // 45
    System.out.println(summe(10)); // 55
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/tech_info/assembler/mehr_adress/SummeIterativ.java`

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

Lösungsvorschlag

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

-- public static int fib(int n) {
--   if (n <= 1)
--     return n;
--
--   int vorletzte = 0;
--   int letzte = 1;
--   int erg = 0;
--
--   while (n > 1) {
```

```
--     erg = letzte + vorletzte;
--     vorletzte = letzte;
--     letzte = erg;
--     n--;
--   }
--   return erg;
-- }

-- n           R2
-- vorletzte   R3
-- letzte      R4
-- erg         R5

fibonacciIterativ:
SEG
                JUMP einstieg

solange:        -- while (n > 1)
                CMP W R2, I 1
                JLE abschluss
                -- erg = letzte + vorletzte;
                ADD W R3, R4, R5
                -- vorletzte = letzte;
                MOVE W R4, R3
                -- letzte = erg;
                MOVE W R5, R4
                -- n--;
                SUB W I 1, R2
                JUMP solange

klGleichEins:   MOVE W R2, R5
                JUMP abschluss

einstieg:       MOVE W n, R2
                -- if (n <= 1) return n;
                CMP W R2, I 1
                JLE klGleichEins

                -- int vorletzte = 0;
                MOVE W I 0, R3
                -- int letzte = 1;
                MOVE W I 1, R4
                -- int erg = 0;
                MOVE W I 0, R5
                JUMP solange

abschluss:      HALT

n:              DD W 7

-- n:           DD W 0 -- 0
-- n:           DD W 1 -- 1
-- n:           DD W 2 -- 1
-- n:           DD W 3 -- 2
```

```
-- n:           DD W 4 -- 3
-- n:           DD W 5 -- 5
-- n:           DD W 6 -- 8
-- n:           DD W 7 -- 13
-- n:           DD W 8 -- 21
-- n:           DD W 9 -- 34
-- n:           DD W 10 -- 55
END
```

```
public class FibonacciIterativ {

  public static int fib(int n) {
    if (n <= 1)
      return n;

    int vorletzte = 0;
    int letzte = 1;
    int erg = 0;

    while (n > 1) {
      erg = letzte + vorletzte;
      vorletzte = letzte;
      letzte = erg;
      n--;
    }
    return erg;
  }

  public static void main(String[] args) {
    int n = 7;
    System.out.println(fib(n)); // 13

    System.out.println(fib(0)); // 0
    System.out.println(fib(1)); // 1
    System.out.println(fib(2)); // 1
    System.out.println(fib(3)); // 2
    System.out.println(fib(4)); // 3
    System.out.println(fib(5)); // 5
    System.out.println(fib(6)); // 8
    System.out.println(fib(7)); // 13
    System.out.println(fib(8)); // 21
    System.out.println(fib(9)); // 34
    System.out.println(fib(10)); // 55
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/tech_info/assembler/mehr_adress/FibonacciIterativ.java`

(b) Folie 57/1,2

(i) zur Multiplikation zweier Zahlen unter Verwendung eines Unterprogramms

Lösungsvorschlag

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

```
-- public static int mult(int a, int b) {
--   return a * b;
-- }

-- erg R5

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

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

einstieg:       MOVE W I -1, -!SP
                MOVE W a, -!SP
                MOVE W b, -!SP
                CALL mult
                ADD W I 8, SP
                -- Das Ergebnis sollte 49 sein.
                MOVE W !SP+, R5
                HALT

-- int a = 7;
a:              DD W 7
-- int b = 7;
b:              DD W 7
END
```

```
public class MultiplikationUnterprogramm {
  public static int mult(int a, int b) {
    return a * b;
  }

  public static void main(String[] args) {
    int a = 7;
    int b = 7;
    System.out.println(mult(a, b)); // 49
  }
}
```

Code-Beispiel auf Github ansehen:
src/main/java/org/bschlangaul/aufgaben/tech_info/assembler/mehr_adress/MultiplikationUnterprogramm.java

(ii) Summe der ersten n Zahlen (rekursiv)

Lösungsvorschlag

```
-- Summe der ersten n Zahlen (rekursiv)

-- public static int summe(int n) {
--   if (n > 0)
--     return n + summe(n - 1);
--   else
```

```
--     return 0;
-- }

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

-- n R4
-- erg R5
summe:          PUSHR
                MOVE W 64+!SP, R4
                -- if (n > 0)
                CMP W R4, I 0
                JEQ istNull
                MOVE W I -1, -!SP
                -- n - 1
                SUB W I 1, R4, -!SP
                CALL summe
                ADD W I 4, SP
                -- n + summe(n -1);
                ADD W !SP+, R4
                JUMP rueckgabe

istNull:        MOVE W I 0, R4

rueckgabe:      MOVE W R4, 68+!SP
                POPR
                RET

einstieg:       MOVE W I -1, -!SP
                MOVE W n, -!SP
                CALL summe
                ADD W I 4, SP
                -- Das Ergebnis sollte 28 sein.
                MOVE W !SP+, R5
                HALT

-- int n = 7;
n:              DD W 7 -- 28

-- Tests

-- n:              DD W 0 -- 0
-- n:              DD W 1 -- 1
-- n:              DD W 2 -- 3
-- n:              DD W 3 -- 6
-- n:              DD W 4 -- 10
-- n:              DD W 5 -- 15
-- n:              DD W 6 -- 21
-- n:              DD W 7 -- 28
-- n:              DD W 8 -- 36
-- n:              DD W 9 -- 45
-- n:              DD W 10 -- 55
```

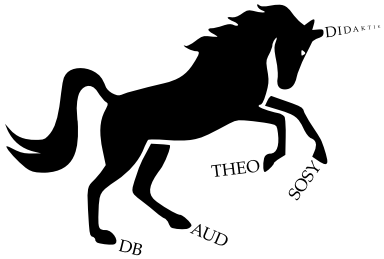
```
END
public class SummeRekursiv {

  public static int summe(int n) {
    if (n > 0)
      return n + summe(n - 1);
    else
      return 0;
  }

  public static void main(String[] args) {
    int n = 7;
    System.out.println(summe(n)); // 28

    System.out.println(summe(0)); // 0
    System.out.println(summe(1)); // 1
    System.out.println(summe(2)); // 3
    System.out.println(summe(3)); // 6
    System.out.println(summe(4)); // 10
    System.out.println(summe(5)); // 15
    System.out.println(summe(6)); // 21
    System.out.println(summe(7)); // 28
    System.out.println(summe(8)); // 36
    System.out.println(summe(9)); // 45
    System.out.println(summe(10)); // 55
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/tech_info/assembler/mehr_adress/SummeRekursiv.java`



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden: `https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/50_TECH/20_Mehr-Adress/Aufgabe_07-Vorlesungsaufgaben.tex`