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).

```
Start:          SEG
                JUMP Beginn

n:              DD W 2

Beginn:         MOVE W I 1, R2
                MOVE W I 1, R4
                MOVE W n, R0
                CMP W R0, I 0
                JEQ istNull
                CMP W R0, I 1
                JEQ istEins

While:          SUB W I 1, R0, R0
                CMP W R0, I 0
                JEQ istEnde

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

                JUMP While

istNull:        MOVE W I 0, R5
                JUMP ende

istEins:        MOVE W I 1, R5
                JUMP ende

istEnde:        MOVE W  R2, R5

ende:           HALT
                END
```

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

```
Start:          SEG
                JUMP Beginn

n:              DD W 10

Beginn:         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 ende

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

ende:           HALT
                END
```

(b) Folie 57/1,2

(i) zur Multiplikation zweier Zahlen unter Verwendung eines Unterprogramms

```
                SEG
                MOVE W I H'0000FFFF' , SP
                JUMP Start

a:              DD W 5
b:              DD W 6
Multi:          PUSHR
                MULT W 64+!SP, 68+!SP, 72+!SP
                POPR
                RET

Start:          MOVE W I -1, -!SP
                MOVE W a, -!SP
                MOVE W b, -!SP
                CALL Multi
                ADD W I 4, SP
                ADD W I 4, SP
                MOVE W !SP+, R5
                HALT
                END
```

(ii) Summe der ersten n Zahlen (rekursiv)

```
Prog:           SEG

-- Initialisierung
                MOVE W I H'10000', SP    -- Stack init
                JUMP start    -- Variablendefinition
                ↪ uebersprungen
n:              DD W 4

-- Programmvorlauf
start:          MOVE W I -1, -!SP  -- Platz fuer Rueckgabewert
↪ auf dem Stack machen
                MOVE W n, -!SP   -- Startwert auf den Stack
                CALL summe   -- summe(n) aufrufen
                ADD W I 4, SP   -- Stack bereinigen
                MOVE W !SP+, R5   -- Rückgabe in Register R5
                JUMP halte
```

```
16  -- summe von n..1
17  summe:          PUSHR    -- interne Register sichern
18                  MOVE W 64+!SP, R0
19                  CMP W R0, I 1
20                  JEQ bottom
21                  MOVE W I -1, -!SP  -- Platz fuer Rueckgabewert
                    ↪  auf dem Stack machen
22                  SUB W I 1, R0, -!SP  -- n-1 auf den Stack
23                  CALL summe   -- summe(n) aufrufen
24                  ADD W I 4, SP   -- Stack bereinigen
25                  ADD W !SP+, R0
26                  JUMP fertig
27  bottom:         MOVE W I 1, R0
28
29  fertig:         MOVE W R0, 68+!SP
30                  POPR    -- Register wieder herstellen
31                  RET    -- zurueck zum Aufruf springen
32
33  halte:          HALT
34                  END
```