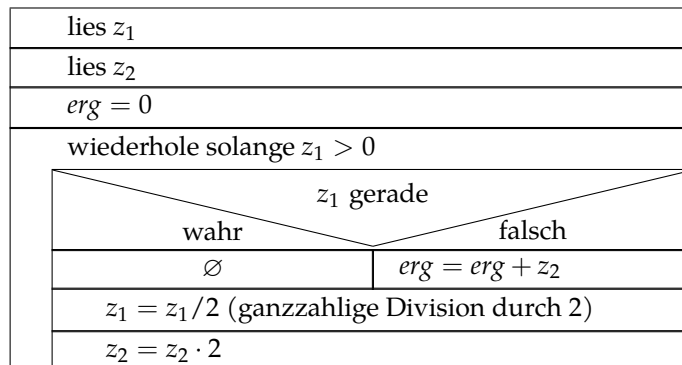


## Abitur 2015 IV

Auf dem ägyptischen *Papyrus Rhind*, der etwa auf das Jahr 1550 v. Chr. datiert wird, ist eine Möglichkeit zur Multiplikation zweier natürlicher Zahlen  $z_1$  und  $z_2$  beschrieben. Als Struktogramm lässt sich dieser Algorithmus folgendermaßen darstellen:



Das berechnete Produkt steht nach Abarbeitung des Algorithmus in der Variablen *erg*.

- (a) Berechnen Sie mithilfe der beschriebenen ägyptischen Multiplikation schrittweise das Produkt aus  $z_1 = 13$  und  $z_2 = 5$ .

$z_1$	$z_2$	$erg$
13	5	5
6	10	-
3	20	25
1	40	65
0	80	-
		65

- (b) Nennen Sie die wesentliche Idee des Speichermodells eines Rechners, der nach dem von-Neumann-Prinzip aufgebaut ist. Geben Sie einen Vor- und einen Nachteil dieses Speichermodells an.

Die 7 Grundprinzipien der *Von-Neumann-Architektur*:

- (i) Der Rechner besteht aus 4 Werken.
- (ii) Der Rechner ist programmgesteuert.
- (iii) Die Programme und Daten liegen im selben Speicher.
- (iv) Der Hauptspeicher ist in Zellen gleicher Größe aufgeteilt.
- (v) Die Programme bestehen aus Folgen von Befehlen (Sequentielle Ausführung)
- (vi) Der Programmablauf ist durch Sprünge möglich.
- (vii) Die Daten und Programme werden in Binärdarstellung gespeichert und verarbeitet.

**Vorteil:** Da in von-Neumann-Rechnern keine redundanten Komponenten verbaut werden, bleibt der Hardwareaufwand gering.

**Nachteil:** Der wichtigste Nachteil ist der sogenannte Von-Neumann-Flaschenhals. Er existiert, da das Bussystem alle Befehle und Daten streng sequentiell transportiert.

- (c) Bestätigen Sie anhand zweier Beispiele, dass mithilfe des folgenden Programmausschnitts entschieden werden kann, ob Speicherzelle 101 eine gerade oder ungerade Zahl enthält.

```
1  LOAD 101
2  SHRI 1
3  SHLI 1
4  SUB 101
```

	gerade Zahl	ungerade Zahl
LOAD 101	8 (0b1000)	9 (0b1001)
SHRI 1	4 (0b0100)	4 (0b0100)
SHLI 1	8 (0b1000)	8 (0b1000)
SUB 101	0	-1

- (d) Schreiben Sie ein Programm für die angegebene Registermaschine, das den Algorithmus des *Papyrus Rhind* umsetzt. Gehen Sie dabei davon aus, dass die beiden positiven ganzzahligen Faktoren  $z_1$  und  $z_2$  bereits in den Speicherzellen 101 und 102 stehen und dass alle weiteren nicht vom Programm belegten Speicherzellen mit dem Wert 0 vorbelegt sind.

#### Assembler

```
1  # z1 := 13;
2  # z2 := 5;
3  werte_setzen:  LOADI 13
4                  STORE 101 # wird verändert
5                  STORE 90 # z1 Eingabe, oberhalb des Ergebnisses
6                  LOADI 5
7                  STORE 102 # wird verändert
8                  STORE 91 # z2 Eingabe, rechts neben z1 Eingabe
9
10 # WHILE z1 > 0 DO
11 solange:      LOAD 101
12                JMPNP ende
13
14 # IF (z1 % 2) = 1 THEN
15 modulo:      SHRI 1
16                SHLI 1
17                SUB 101
18                CMPI 0
19                JMPZ werte_aendern
20
21 # erg := erg + z2;
22 ist_ungerade: LOAD erg
23                ADD 102
24                STORE erg
25
26 # z1 halbieren
```

```

27 # z1 := z1 / 2;
28 werte_aendern:  LOAD 101
29                  DIVI 2
30                  STORE 101
31
32 # z2 verdoppeln
33 # z2 := z2 * 2;
34                  LOAD 102
35                  MULI 2
36                  STORE 102
37                  JMP solange
38
39 # Ergebnis auf Speicherzelle 100 setzen,
40 # damit man das Ergebnis besser sieht.
41 ende:           LOAD erg
42                  STORE 100
43                  HOLD
44
45 erg:            WORD 0

```

## Minisprache

```

1 PROGRAM papyrus_rhind;
2 VAR z1, z2, erg;
3 BEGIN
4     z1 := 13;
5     z2 := 5;
6     erg := 0;
7     WHILE z1 > 0 DO
8         IF (z1 % 2) = 1 THEN
9             erg := erg + z2;
10        END;
11        z1 := z1 / 2;
12        z2 := z2 * 2;
13    END
14 END papyrus_rhind.

```

## Java (iterativ)

```

6     int ergebnis = 0;
7     while (z1 > 0) {
8         if (z1 % 2 == 1) {
9             ergebnis = ergebnis + z2;
10        }
11        z1 = z1 / 2;
12        z2 = z2 * 2;
13    }
14    return ergebnis;
15 }
16

```

## Java (rekursiv)

```

18     return String.format("%3d ", zahl);
19 }
20

```

```
21 public static int multipliziereIterativAusgabe(int z1, int z2) {
22     System.out.println(String.format("multipliziere %dx%d", z1, z2));
23     int ergebnis = 0;
24     System.out.print(formatiereZahl(z1));
25     System.out.print(formatiereZahl(z2));
26
27     while (z1 > 0) {
28         if (z1 % 2 == 1) {
29             System.out.println(formatiereZahl(z2));
30             ergebnis = ergebnis + z2;
31         } else {
```