

10. Übungsblatt

Aufgabe 1 Virtuelle Maschine: Objekt und Arrayzugriff

- a) Gegeben sei die (vereinfachte) Jack-Implementierung der Klasse `Fraction` aus der Vorlesung, von der wir den folgenden Teil betrachten wollen:

```
class Fraction {
    field int numerator, denominator;

    constructor Fraction new (int a, int b) {
        let numerator = a;
        let denominator = b;
        return this;
    }
    // weitere Methoden/Funktionen
}
```

Implementieren Sie den Konstruktor dieser Klasse in der Sprache der virtuellen Maschine (d.h., übernehmen Sie die Rolle des Übersetzer (*compilers*))!

- b) Gegeben sei die folgende Klasse in der Jack-Programmiersprache:

```
class Dummy {
    function int[] makearray (int n, int i) {
        var int[] a;
        a = Array.new(n);
        a[i] = 3;
        return a;
    }
    // weitere Funktionen/Methoden
}
```

Implementieren Sie die Funktion `makearray(n,i)` dieser Klasse in der Sprache der virtuellen Maschine (d.h., übernehmen Sie die Rolle des Übersetzer (*compilers*))!

In beiden Funktionen können Sie zur Anforderung von Speicher für die zu erzeugende Objektinstanz bzw. den anzulegenden Array die Funktion `Memory.alloc(n)` aus der Standardbibliothek benutzen, die `n` Speicherzellen auf dem Heapspeicher anfordert und die (Anfangs-)Adresse des zugeordneten Speicherbereichs auf dem Stapel ablegt.

Aufgabe 2 Virtuelle Maschine: Übersetzung in Assemblersprache

Übersetzen Sie die folgenden Anweisungen aus der Sprache der virtuellen Maschine in die Hack-Assemblersprache (d.h., übernehmen Sie die Rolle der virtuellen Maschine):

```
push argument 0
push static 0
sub
pop local 1
```

Aufgabe 3 Virtuelle Maschine: Funktionsaufrufe

Wir betrachten die folgende rekursive Funktion zum Berechnen der Summe der Zahlen von 1 bis n in der Sprache der virtuellen Maschine:

```
function Main.sum 0
  push argument 0
  push constant 0
  gt
  if-goto RECURSE
  push constant 0
  return
label RECURSE
  push argument 0
  push constant 1
  sub
  call Main.sum 1
  push argument 0
  add
  return
```

Welche Zustände durchläuft der Stapel bei der Abarbeitung eines Aufrufs der obigen Funktion aus dem Hauptprogramm mit

```
function Main.main 0
  push constant 2
  call Main.sum 1
  return
```

Gehen Sie davon aus, daß der Stapelzeiger `SP` am Anfang der Funktion `Main.main` (also direkt vor `push constant 2`) auf die Speicherzelle 300 zeigt und die virtuellen Register `R1`, `R2`, `R3` und `R4` die Werte 300, 295, 4000 bzw. 4001 enthalten. (Hinweis: Welche anderen Symbole sind in der Symboltabelle des Assemblers mit den gleichen Werten wie `R1`, `R2`, `R3` bzw. `R4` definiert? Welchen Bezug haben diese Symbole zur virtuellen Maschine?)