KSP Aufgabe 6

1. In dieser Version der VM wollen wir als Rechenobjekte beliebig große ganze Zahlen zulassen. Diese neuen Rechenobjekte ersetzen die alten 32-Bit-Zahlen und haben ihren Speicherplatz ebenfalls auf dem Heap. Auf dem Stack, in der *Static Data Area* und im Rückgabewertregister stehen wie gehabt Zeiger auf die Rechenobjekte.
2. Die Algorithmen zum Rechnen mit beliebig großen ganzen Zahlen (*Multiple Precision Arithmetic*) sind entnommen aus [D. Knuth: *The Art of Computer Programming, Vol. 2, Seminumerical Algorithms*]. Eine Implementierung in C finden Sie hier. Entpacken Sie das Paket und studieren Sie die darin enthaltene Datei README. Bauen Sie das Paket und lassen Sie die Tests laufen! Machen Sie sich mit der Struktur der Zahlendarstellung vertraut! Hinweise: Jede Ziffer der Zahl ist ein Byte, d.h. die Zahldarstellung benutzt die Basis 256. Die Ziffern werden in einem genügend großen Array gespeichert, das zusammen mit der Größenangabe in einer Struktur ("Objekt") auf dem Heap steht.
3. Die *Multiple Precision Arithmetic* braucht eine winzige Support-Bibliothek, deren Funktionalität aber in Ihrer VM schon enthalten ist. Stellen Sie die benötigten Funktionen aus Ihrer VM zur Verfügung!
4. Studieren Sie das Benutzer-Programmierinterface und passen Sie Ihre VM an. Achten Sie darauf, ALLE Operationen mit Rechenobjekten der Bibliothek zu überlassen; dazu gehören auch die Vergleiche!
5. Der Instruktionssatz ist gegenüber Aufgabe 5 unverändert; auch der Compiler und der Assembler bleiben exakt gleich (bis auf die Versionsnummern).
6. Hier wie immer die Referenzimplementierung: njvm

Aufgaben für Tests a) Schreiben Sie ein kleines Ninja-Programm zur Berechnung von 100! = 1*2*3*...*100. Fällt Ihnen am Ergebnis etwas auf? Kann das denn überhaupt richtig sein?

b) Die Fibonacci-Folge ist definiert als `F(0) = 0, F(1) = 1, und F(n) = F(n-1) + F(n-2)` für alle `n > 1`. Schreiben Sie ein kleines Ninja-Programm zur Berechnung von `F(100)`. Hinweis: Nur die iterative Version wird in annehmbarer Zeit zum Ergebnis führen!

c) Schreiben Sie ein kleines Ninja-Programm zur Berechnung der Summe aller Brüche `\frac{1}{i}` für `i = 1, \dots, 100` (die einhundertste "harmonische Zahl"). Das Ergebnis soll als exakter Bruch in gekürzter Darstellung angegeben werden. Hinweise: Halten Sie den Zähler und den Nenner des Ergebnisses in zwei verschiedenen Variablen. Wenn Sie einen neuen Term addieren wollen, müssen Sie die Brüche auf einen gemeinsamen Nenner bringen. Beim Kürzen (das Sie entweder bei jeder Rechnung oder aber einmal ganz am Ende aller Rechnungen durchführen können) leistet der größte gemeinsame Teiler gute Dienste! Können Sie die Zahl in Dezimalschreibweise mit z.B. 10 Stellen nach dem Komma ausgeben? Dazu sind nur Ganzzahloperationen nötig!