KSP Aufgabe 4

Jetzt wird's ernst... ;-) Mit der Lösung dieser Aufgabe steht Ihnen das komplette Ninja (mit Ausnahme der Array- und Record-Objekte) zur Verfügung. Sie können damit z.B. die folgenden Programme ausführen: a) ggT b) n! rekursiv c) n! iterativ

Aber schön der Reihe nach...

Neue Instruktionen der VM

Table 1. VM Instruktionen

Instruktion	Opcode	Stack Layout
`call <target>`	`26`	`... -> ... ra`
`ret`	`27`	`... ra -> ...`
`drop <n>`	`28`	`... a0 a1...an-1 -> ...`
`pushr`	`29`	`... -> ... rv`
`popr`	`30`	`... rv -> ...`
`dup`	`31`	`... n -> ... n n`

1. Hier ist eine Diskussion der Instruktionen zum Unterprogrammaufruf und in VM Instruktionen eine Liste aller Instruktionen der VM dieser Version.
2. Realisieren Sie Unterprogrammsprünge und -rücksprünge mit den Instruktionen `call` und `ret`. Hier ist ein Testprogramm ohne Argumente und ohne Rückgabewert. Sie sollten die Ausführung mit Hilfe Ihres Debuggers im Einzelschrittverfahren genau verfolgen können.
3. Implementieren Sie die Instruktion `drop` und testen Sie den Zugriff auf die Argumente einer Prozedur mit diesem Testprogramm.
4. Fügen Sie das Rückgaberegister zur Maschinenarchitektur hinzu und implementieren Sie die Instruktionen `pushr` und `popr`. Sie können die Rückgabe eines Wertes mit diesem Testprogramm überprüfen.
5. Dann sollte schließlich dieses Testprogramm ohne weitere Arbeit funktionieren.
6. Implementieren Sie die Instruktion `dup`, die den obersten Stackeintrag dupliziert: `⋯ n → ⋯ n n`
7. Der zu dieser Aufgabenstellung gehörende Assembler ist natürlich auch verfügbar: nja
8. Zu guter Letzt hier wieder die Referenzimplementierung: https://git.thm.de/arin07/KSP_public/-/blob/master/aufgaben/a4/njvm

Der Ninja-Compiler

Nun gibt's eine erste Version des Ninja-Compilers - noch ohne Arrays, Records und anderen Schnickschnack, aber mit Prozeduren und Funktionen, Kontrollstrukturen, Zuweisungen und Ausdrücken. Die Grammatik zusammen mit den Tokens und den vordefinierten Bezeichnern beschreibt, wie ein korrektes Ninja-Programm (in unserem noch etwas reduzierten Sprachumfang)

aussieht.

1. Übersetzen Sie die drei ganz oben genannten Testprogramme mit Hilfe des Ninja-Compilers. Der Output ist Ninja-Assembler und kann z.B. mit einem Texteditor angeschaut werden. Versuchen Sie, jedes vom Compiler generierte Assembler-Statement einer Ninja-Quelltext-Anweisung zuzuordnen. Wenn Sie Fragen dazu haben, diskutieren Sie diese bitte mit den Betreuern im Praktikum!
2. Assemblieren Sie die drei Testprogramme und lassen Sie sie durch Ihre VM ausführen. Verfolgen Sie die Ausführung mit dem Debugger Ihrer VM.
3. Schreiben Sie mindestens fünf weitere Testprogramme, entweder in Ninja oder in Assembler, um sicherzugehen, dass die VM korrekt funktioniert. Loten Sie dabei auch Konstruktionen aus, die bisher nicht Bestandteil der Tests waren. Ein Beispiel: Kann eine Funktion in ihrem Return-Statement eine andere Funktion aufrufen? Geht das auch geschachtelt?
4. Welche Sequenz von Instruktionen erzeugt der Compiler für das logische `und` und das logische `oder`? Erklären Sie genau, wie damit die Kurzschlussauswertung implementiert wird!