Zürcher Hochschule
für Angewandte Wissenschaften

Eigene Programmiersprache erster Teil

Aufgabe 1 Code Generierung für arithmetische Ausdrücke

Studieren Sie zuerst die beiden Java Klassen im Anhang der Vorlesung. Compilieren Sie diese und führen Sie diese aus. Es sollten entsprechende WASM Files erzeugt werden, die mittels den beigefügten Werkzeugen oder dem Browser ausführen können.

Den Instruktionssatz entnehmen Sie den Links aus den Folien. Zu bemerken ist, dass JWebAssembly nicht den vollen Instruktionssatz anbietet und z.B. für die Werte-Umwandlungsroutinen eigene Mnemonics (D2I, etc.) verwendet.

InstructionList (il)
In der Instruktionsliste werden die Instruktionsobjekte hinzugefügt. Die Instruktionsobjekte und die Reihenfolge, in der Sie in der Instruktionsliste abgelegt wurden definieren die Funktion der Methode.

Aufgabe: Generieren Sie Code für die Berechnung von arithmetischen Ausdrücken (Expressions).

Hinweise:

- Ihr Parser muss in der JWebAssembly-Methode übergebenen emit-Methode aufgerufen werden (siehe Beispiele in den Folien)
- Das Interface das als erstes Argument mitgegeben wird, definiert die Funktion die implementiert wird und deren Signatur (Parameter).

Statt den Wert direkt zu berechnen, soll Code für den Ausdruck generiert werden und dann nachträglich berechnet werden (durch Aufruf der generierten Methode). Das Gerüst für den Aufruf sieht folgendermassen aus:

```
public static void main(String[] args) throws Exception
  Scanner.init("4.2 + 3.2*2");
  Scanner.scan();
  JWebAssembly.emitCode(ICalculator.class, new Calculator ());
}
```

Wobei die Calculator Klasse das Emitter Interface implementiert. In der `emit` Methode wird dann einfach `expr()`aufgerufen, das den Parser startet. Beim Parsen wird dann der Code generiert und nicht mehr der Ausdruck ausgewertet.

Aufgabe 2 Code Generierung mit Variablen

Jetzt sollen Sie einen Schritt weiter gehen und Code für Ausdrücke mit Variablen generieren. Erstellen Sie dafür eine Klasse Programm und folgendes Interface;

```
public interface IProgram {
      double main(double arg);
}
```

Anbei die Grammatik unserer (ersten) Programmiersprache.

```
expr = term {("+" | "-”) term}
term = factor {("*" | "/") factor}
factor = number | ident | "(" expr ")"

assignement = ident "=" expr ";"
statement = assignment

statementSequence = statement {statement}
program = statementSequence
```

Assignement: einer lokalen Variablen wird ein Ausdruck zugewiesen, z.B. a = 6+2. Zusätzlich sollen in Ausdrücken (expr) ebenfalls Variablen zugelassen werden, z.B. i = i - 1

Statement: Eine Anweisung ist ein Assignment (wird später noch erweitert).

Program: besteht aus einer oder mehreren Anweisungen.

Eine Zuweisung besteht aus einem Variablennamen (dem ein Speicherplatz zugewiesen wurde) und einem Ausdruck. Damit der Parser weiss, welcher Speicherplatz für welche Variabel reserviert wurde, wird eine Liste geführt.

Beispiel-Programm

```
x = $arg0;
a = 1;
b = 2;
c = 3;
value = a*x*x + b*x + c;
```

Hinweis:

- Die Methode ***local*** der Klasse JWebAssembly liefert die Nummer des Slots der lokalen Variablen oder generiert bei Bedarf eine neue Slot-Nummer; diese können für die entsprechenden Load und Store Operation verwendet werden.
- Z.B. local(ValueType.f64,"value") liefert die Slot Nummer für die entsprechende Variable.
- Mit den vordefinierten Namen **$arg0, $arg1, …** kann auf die Parameter der Methode zugegriffen werden.
- Vergessen Sie nicht, dass am Schluss der Methode ein RETURN stehen und der berechnete Wert (der Variablen "value") auf dem Stack abgelegt sein muss.