THM | CAMPUS GIESSEN
TECHNISCHE HOCHSCHULE MITTELHESSEN

MNI
Mathematik, Naturwissenschaften und Informatik

THM | Campus Gießen | Wiesenstraße 14 | 35390 Gießen

Prof. Dr. Uwe Meyer
University of Applied Sciences

Praktikumsaufgabe 5
Compilerbau

Schritt 1 (SPL Laufzeitstack-Organisation studieren)

Machen Sie sich mit der Laufzeitstack-Organisation für SPL vertraut, indem Sie das dazu gehörige Kapitel 7 im Praktikumsskript gründlich studieren.

Geben Sie für einen Prozeduraufruf von Hand die zugehörigen Rahmen-Layouts von Caller und Callee an. Rechnen Sie für einen Beispielwert vom FP die absoluten Adressen aller Speicherbereiche in den Rahmen nach dem Aufruf des Callee aus. Geben Sie die Adressen relativ zu SP und FP im Caller vor dem Aufruf und im Callee nach dem Aufruf an.

Schritt 2 (Speicherbedarf für Typen bestimmen)

Für jeden Typ muss seine Größe in Bytes abgespeichert werden. Dies ist im Skelett bereits für sie erledigt worden. Die Größe wird jeweils bei der Konstruktion der entsprechenden Klasse/Struktur abgespeichert. Wie wird der Platzbedarf für Arrays bestimmt? Wie für primitive Typen? Schauen sie sich dazu die Konstruktoren der entsprechenden Klassen/Strukturen an!

Hinweis: Auf ECO32 belegt ein Integer 4 Bytes und eine Adresse (wichtig für Referenz-Parameter) ebenfalls 4 Bytes. Ein boole'scher Wert kann ebenfalls mit 4 Bytes angenommen werden (obwohl man ihn niemals speichern muss - ist klar, warum?).

Da man solche Konstanten („Magic Numbers“) ungern tief im Code versteckt, sind sie bereits im Skelett definiert.

a) für die Java-Implementierung als Teil der statisch definierten Konstanten `intType` und `boolType` in der `PrimitiveType`-Klasse.
(Zugriff mit `PrimitiveType.intType.byteSize`)

b) für die C-Implementierung als Konstanten `INT_BYTE_SIZE, BOOL_BYTE_SIZE` in ”`tablebuild.c`”, sowie `REF_BYTE_SIZE` in ”`varalloc.h`“.

Schritt 3 (Speichermöglichkeiten für Offsets und Bereichsgrößen)

Für Offsets von Argumenten, Parametern und lokalen Variablen muss eine Speichermöglichkeit vorgesehen werden. Auch das ist bereits für Sie erledigt worden.

Wo könnte man diese Informationen sinnvoll speichern? Lokalisieren sie die zugehörigen Stellen im Skelett! Warum speichert man sie dort und nicht anderswo?

Für das Stacklayout einer Prozedur müssen die Größen der für folgende drei Bereiche ermittelt werden:

- eingehende Argumente
- lokale Variable
- ausgehende Argumente.

Auch Speicherorte für diese Bereichsgrößen sind bereits im Skelett vorgesehen. Lokalisieren Sie auch diese!

Schritt 4 (AST-Durchgang 1: Parameter-Offsets und Variablen-Offsets)

Ergänzen Sie das Programm aus um die ersten beiden Bereichsgrößen (eingehende Argumente, lokale Variable) sowie sämtliche Offsets zu ermitteln und einzutragen. Nutzen Sie das Visitor-Muster.

Ergänzen Sie dafür die Datei:
C: `src/phases/_05_varalloc/varalloc.c`
Java:
`src/main/java/de/thm/mni/compilerbau/phases/_05_varalloc/VarAllocator.java`

Die entsprechenden Stellen sind wie immer mit einem `TODO` markiert. Erstellen Sie sich Hilfsklassen/-funktionen, um Ihren Code ordentlich zu strukturieren!

Schritt 5 (Offsets und Bereichsgrößen von vordefinierten Prozeduren)

Welche dieser Informationen brauchen Sie die von SPl vordefinierten Prozeduren (z.B. `readi`)? Welche Informationen dieser Prozeduren brauchen Sie nicht?

Die Informationen sind im Skelett bereits vorhanden. Finden Sie heraus, wo sie berechnet und eingetragen werden, und ob Sie dafür noch Code aus dem Skelett aufrufen müssen.

Schritt 6 (AST-Durchgang 2: Bereichsgröße für ausgehende Argumente)

Programmieren Sie einen zweiten Durchlauf durch die Prozedurdeklarationen, um auch die dritte Bereichsgröße (ausgehende Argumente) zu ermitteln. Warum kann dies nicht im selben Durchgang erledigt werden?

Erinnerung: Diese berechnet sich als das Maximum über die Größen der Bereiche eingehender Argumente aller aus der betrachteten Prozedur heraus AUFGERUFENEN Prozeduren.

Hinweis: Sehen Sie dabei auch irgendeine Möglichkeit vor, zu vermerken, ob die gerade betrachtete Prozedur überhaupt eine weitere aufruft; das wird später bei der Codeerzeugung ebenfalls gebraucht. Die Bereichsgröße auf -1 zu setzen ist eine Möglichkeit, Sie können aber auch ein zusätzliches Feld einführen. Welche Lösung finden Sie eleganter?

Wie immer empfiehlt es sich, Testausgaben einschalten zu können. Dazu steht innerhalb der Klasse `VarAllocator` die boole'sche Instanzvariable `showVarAlloc` zur Verfügung. In C wird der Wert Ihrer `allocVars` Funktion übergeben. Der Wert ist genau dann `true` hat, wenn der Benutzer des Compilers das Kommandozeilenargument `--vars` angegeben hat.

Der zugehörige Code ist bereits für Sie vorgegeben. In der Java-Lösung muss dafür eine Zeile Code „entkommentiert" werden.