Prof. Dr. M. Jäger
FB MNI

Compilerbau - Praktikumsaufgabe 5 „Variablen-Allokation“

Schritt 1 (SPL Laufzeitstack-Organisation studieren)

Machen Sie sich mit der Laufzeitstack-Organisation für SPL vertraut, indem Sie das dazu gehörige Informationsblatt 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)

Speichern Sie in jedem Typ seine Größe (d.h. die Anzahl Bytes, die ein Objekt dieses Typs im Speicher belegen wird). Vorschlag: Erweitern Sie die Klasse „Type“ um eine Komponente. Geben Sie beim Aufruf von „PrimitiveType“ die Größe des Typs mit. Berechnen Sie die Größe für Arrays bei der Typkonstruktion. Hinweis: Auf ECO32 und Puck VM 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 ungern tief im Code versteckt, stehen sie

a) für die Java-Implementierung unter den Namen

```
Varalloc.intByteSize,
Varalloc.boolByteSize,
Varalloc.refByteSize
```

als Klassenvariable der Klasse „Varalloc“ zur Verfügung.

b) für die C-Implementierung als Konstanten `INT_BYTE_SIZE`, `BOOL_BYTE_SIZE`, `REF_BYTE_SIZE` in „varalloc.h“.

Schritt 3 (AST-Durchgang 1: Parameter-Offsets und Variablen-Offsets)

Sehen Sie eine Speichermöglichkeit für Offsets bei Argumenten, Parametern und lokalen Variablen vor. Vorschlag: Erweitern Sie die Klassen „ParamTypeList“ und „VarEntry“ um jeweils eine Komponente (C: „ParamTypes“ und „varEntry“).

Sehen Sie Speicherplätze für die Größen der drei Bereiche vor:

- eingehende Argumente
- lokale Variable
- ausgehende Argumente.

Vorschlag: Erweitern der Klasse „procEntry“ um drei Komponenten. (für C: Erweitern der `struct procEntry`).

Ergänzen Sie das Programm aus um das Eintragen der ersten beiden Bereichsgrößen (eingehende Argumente, lokale Variable).

Schritt 4 (AST-Durchgang 2: Bereichsgröße für ausgehende Argumente)

Programmieren Sie einen zweiten Durchlauf durch die Prozedurdeklarationen, um auch die dritte Bereichsgröße (ausgehende Argumente) zu ermitteln. 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.

Vergessen Sie nicht, auch die Bibliotheksprozedur-Einträge zu behandeln, jedenfalls soweit es die Offsets der Argumente und die Größe des Bereichs der eingehenden Argumente betrifft. Sie brauchen sich NICHT um die Offsets der lokalen Variablen bzw. ausgehenden Argumente und deren Bereichsgrößen zu kümmern, denn Sie machen KEIN Layout der Stack-Frames für die Prozeduren aus der Bibliothek (das hat der Konstrukteur der Bibliothek für Sie erledigt). Sinn dieser Aktion ist es,

- zu wissen, wieviel Platz man für Argumente von Aufrufen der Bibliotheksfunktionen braucht
- Offsets zum Speichern der Argumente zu haben.

Wie immer empfiehlt es sich, Testausgaben einschalten zu können. Dazu steht die boole'sche Instanzvariable „showVarAlloc“ zur Verfügung, die genau dann den Wert true hat, wenn der Benutzer des Compilers das Kommandozeilenargument `--vars` angegeben hat.