Übung zur Vorlesung Compiler 1: Grundlagen

Prof. Dr. Andreas Koch
Jens Huthmann

TECHNISCHE UNIVERSITÄT DARMSTADT

Wintersemester 12/13
Aufgabenblatt 3

Abgabemodalitäten

Die Aufgabe sind in 2er Gruppen zu bearbeiten. Ihre Lösungen reichen Sie als PDF-Datei per E-Mail bis zum 3.02.2013 um 23:59 MEZ an der Adresse oc@esa.informatik.tu-darmstadt.de ein. Die E-Mail muss als Betreff "Compiler 1 Aufgabe 3" haben. Geben Sie in Ihrem Lösungsdokument den Namen und die Matrikelnummer aller Gruppenmitglieder an.

Aufgabe 3.1 Allgemeine Fragen — 9 Punkte

a) Beschreiben Sie kurz die Aufgabe des Routinenprotokolls. Begründen Sie auch warum ein Routinenprotokoll notwendig ist.

b) Wieso ist es notwendig einen Stack und einen Heap in der Speicherverwaltung zu verwenden? Beschreiben Sie hierzu auch kurz welche Daten im Stack bzw. Heap gespeichert werden.

c) Wofür wird der dynamische bzw. statische Link innerhalb der Speicherverwaltung verwendet?

Aufgabe 3.2 Routinen als Parameter — 20 Punkte

Triangle bietet die Möglichkeit auch Funktionen oder Prozeduren als Parameter bei einem Aufruf zu verwenden. Hierzu wird ein Paar bestehende aus Static Link und der Adresse der Routine übergeben. Dieses Paar nennt man Closure.

Untersuchen Sie das folgende Programm mit Hilfe des Triangle Disassemblers. Kommentieren Sie im disassemblierten Code die Position an welcher die Clossure erzeugt wird und wie die parametrisierte Funktion aufgerufen wird. Geben Sie auch an wo die einzelnen Routinen beginnen.

```
let
  func twice(func doit(x : Integer): Integer, i : Integer): Integer ~ doit(doit(i));
  func double(d : Integer): Integer ~ d*2;
  var x: Integer
in begin
  x := twice(func double, 10);
  putint(x)
end
```

Aufgabe 3.3 Codeschablonen — 16 Punkte

Triangle soll um ein "`foreach ARRAY do C`" Kommando erweitert werden. Dieser Befehl soll auf jedes Element des Arrays das Kommando `C` ausführen. Hierbei wird das aktuelle Element von `ARRAY` innerhalb `C` durch `current` dargestellt. In `size` steht die Anzahl der Elemente von `ARRAY`. Beachten Sie hierbei, dass `C` auch ein "`begin ... end`" Block sein kann. Ein `return` beendet ganz normal die Ausführung der aktuellen Prozedur.

Bestimmen Sie für diesen Befehl die Codeschablone, welche das Kommando ausführt. Sie dürfen hierzu bestehende Schablonen verwenden.

Aufgabe 3.4 Statische Speicherverwaltung — 20 Punkte

Nachfolgend finden Sie die zur Definition eines Go-Spiels notwendigen Datentypen. Geben Sie für dieses Programm die Speicherorganisation der Variable *state* gemäß Folie 40 des 4. Vorlesungsblocks an. Ihre Lösung sollte für jeden Wort-Index des Speichers, beginnend ab der fiktiven Anfangsadresse 0, dessen benutzerdefinierten- (Player, Point,...) und Basistyp (Integer, Boolean) angeben. Markieren Sie auch welche Worte zu welchem Record bzw. Array gehören.

```
let
  type Player ~ record
    number : Integer
  end;
  type Point ~ record
    empty : Boolean,
    owner : Player
  end;
  type Board ~ record
    board : array 361 of Point
  end;
  type State ~ record
    moves : Integer
    next : Player,
    board : Board,
  end;
  var state : State
in
  ...
```

Listing 1: Beispiel zur statischen Speicherverwaltung

Aufgabe 3.5 Routinenprotokoll — 10 Punkte

Das nachfolgende Programm implementiert die notwendigen Funktionen zur Berechnung der Fibonacci-Folge. Zeichnen Sie hierfür stack frames, die Argumente und Resultate vor Eintritt in eine Funktion und nach Austritt aus einer Funktion zeigen für die Auswertung folgender Ausdrücke:

a) $mfib(4)$

```
let
        var n : Integer;

        proc mfib(n : Integer, var r : Integer) ~
        let
                var a : Integer;
                var b : Integer
        in
        if n <= 2 then r := n + 1
        else begin
                 mfib(n-1,var a);
                 mfib(n-2,var b);
                 r := a+b;
        end

in begin
        getint(var n);
        mfib(n,var n);
        putint(n);
end
```

Listing 2: Beispielprogramm zum Routinenprotokoll zu Aufgabe 3.4a

Plagiarismus

Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Zu diesen gehört auch die strikte Verfolgung von Plagiarismus. Weitere Infos unter www.informatik.tu-darmstadt.de/plagiarism