TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Lehrstuhl II — *Sommersemester 2007*
Prof. Helmut Seidl — *Andrea Flexeder*

Compilerbau & Virtuelle Maschinen

3. Übungsblatt

Abgabe: 7. Mai 2007, bis 16 Uhr in der Vorlesung oder Raum 02.07.59

Aufgabe 7: *Code-Erzeugung* — *7 Punkte*

Für diese Aufgabe sollen Sie die CMa des Visualisierungs-Systems VAM verwenden, welche unter `http://wwwseidl.in.tum.de/~ziewer/vam/` heruntergeladen werden kann.
Die Dateiendung für CMa-Programme lautet `.cma`.

```
int result;

int ackermann(int m, int n) {
  int result;
  int tmp;

  if (m == 0)
    result = n+1;
  else{
    if ((m > 0) & (n == 0))
       result = ackermann(m-1, 1);
    else{
       tmp = ackermann(m, n-1);
       result =ackermann(m-1, tmp);
    }
  }
  return result;
}

int main(){
  int m;
  int n;

  m = 1;
  n = 2;
  result = ackermann(m,n);
  return 0;
}
```

a) Berechnen Sie eine Adressumgebung für die Variablen (globale,lokale,die der Funktion).
b) Erzeugen Sie CMa-Code für das angegebene Programm.
c) Verfizieren Sie Ihren CMa-Code mit Hilfe des VAM-Systems.

Aufgabe 8: *Referenz-Parameter* *8 Punkte*

C++ bietet neben Parameter-Übergabe “by-value” auch Referenz-Parameter an. Wird z.B. im Rumpf der Funktion *f* der formale Parameter *x* verwendet, ist die Variable gemeint, mit der *f* als aktuellem Parameter aufgerufen wurde.

```
int a;
void f(int &x) {
  x=7;
}
main() {
  f(a);
}
```

Nach Ausführung von *f(a)* sollte also die Variable a den Wert *7* enhalten.

a) Modifizieren Sie das Übersetzungsschema für Parameter-Übergabe
b) und für Variablenzugriffe so, dass Referenz-Parameter korrekt behandelt werden.
c) Übersetzen Sie das Programm und überprüfen Sie die Korrektheit mit dem VAM-System.

Aufgabe 9: *Exception Handling* *6 Punkte*

C++ stellt ein Konzept zur Handhabung von Exceptions bereit. Die syntaktische Erweiterung besteht aus den Sprachkonstrukten einer “throw”-Anweisung und eines “try-catch”-Blocks. Die Semantik eines “throw”-Konstrukts entspricht dem Abbruch der aktuellen Programmausführung und führt zur Fortsetzung der Programmausführung am Beginn des zugehörigen “catch”-Konstrukts.

```
try{
  //...
  throw i-5;
}
catch(struct s1 struct_exception){
  //...
}
catch(int int_exception){
  printf("␣int-exception␣%i␣trat␣auf␣\n",int_exception);
}
```

Überlegen Sie sich Anpassungen für das Übersetzungsschema und sinnvolle CMa-Befehle, um Exceptions handzuhaben.

Tipp: Es wäre u.U. sinnvoll eine Exception-Umgebung χ zu verwalten, in welcher eine Zuordung von Exception-Typen zu Ganzzahlen vorgenommen wird. Hierbei können Sie davon ausgehen, dass zur Compilezeit der komplette Programmcode vorliegt.
Für das oben angegebene Beispiel ergäbe sich:

$$\chi = \{int \mapsto 0,\ s1 \mapsto 1\}$$