Theoretische Informatik - Übungsblatt 8

Jan Hendrik Dithmar - Matrikelnummer 2031259
Dirk Heine - Matrikelnummer 2030941
Di, 9-11 Uhr, Andreas Augustin

Aufgabe 1

bearbeitet von Jan Hendrik Dithmar
Es muss nur der Teil $x_i = x_j \circ x_k$ betrachtet werden, da $x_i = x_j \circ c$ bei beiden Registermaschinen gleich ist. Man muss also $x_i = x_j \circ x_k$ durch $x_i = x_j \circ c$ darstellen.
Der Einfachheit halber wird in den folgenden Programmen $\neq$ als `!=` und $\dot{-}$ als `-`.

$x_i = x_j + x_k$:

```
x_i = x_j + 0
WHILE (x_k != 0) DO
  x_i = x_i + 1
  x_k = x_k - 1
OD
```

$x_i = x_j \dot{-} x_k$:

```
x_i = x_j - 0
WHILE (x_k != 0) DO
  x_i = x_i - 1
  x_k = x_k - 1
OD
```

$x_i = x_j \times x_k$:

```
x_i = x_j * 1
WHILE (x_k != 0) DO
  x_i = x_i + x_j
  x_k = x_k - 1
OD
```

$x_i = x_j \text{ div } x_k$:

```
WHILE (x_j != 0) DO
   x_k = x_i + 0
WHILE (x_k != 0) DO
   x_l = x_l - 1
   x_k = x_k - 1
OD
   x_j = x_j - 1
OD
x_i = x_l + 0
```

$x_i = x_j \bmod x_k$:

```
x_k = x_i + 0
x_l = x_j + 0
IF (x_l != 0) THEN
  GOTO program
ELSE
   GOTO end
FI

program:
x_m = x_k + 0
WHILE (x_l != 0) DO
   IF (x_k != 0) THEN
    x_l = x_l - 1
    x_k = x_k - 1
  ELSE
    x_j = x_m + 0
    GOTO end
  FI
OD
x_l = x_j + 0
GOTO program
end:
```

Aufgabe 3

bearbeitet von Dirk Heine

```
; Globale Variablen:
; x1 : i // Rückgabewert des Programms
; x2 : n
; x3 : Stackpointer
; x4 : Freie Verwendung

; Lokale Variablen:
; x[x3+0] : retval
; x[x3+1] : i
; x[x3+2] : n
; x[x3+8] : Wenn != 0, Rücksprung nach back1
; x[x3+9] : Wenn != 0, Rücksprung nach back2

; Beginn des Hauptprogramms
x3 = 10 ; Stackpointer initialisieren
x[x3+1] = x1 ; push i
x[x3+2] = x2 ; push n
x[x3+8] = 0 ; Nicht nach back1 springen
x[x3+9] = 0 ; Nicht nach back2 springsn
GOTO ackermann

main: ; hierhin wird zurück gesprungen
x1 = x[x3+0] ; Ergebnis der Ackermann-Funktion in x1
GOTO end
; Ende des Hauptprogramms

; Beginn der Ackermann-Funktion
ackermann:
  case1:
    IF (x[x3+2] != 0) THEN
      GOTO case2
    ELSE ; n == 0
      x[x3+0] = 1 ; A(i, 0) = 1
      GOTO return
    FI

  case2: ; n != 0
    IF (x[x3+1] != 0) THEN
```

```
      GOTO case4
    ELSE ; i == 0
      x4 = x[x3+1] - 1
      IF (x4 != 0) THEN
        GOTO case3
      ELSE ; n == 1
        x[x3+0] = 2 ; A(0, 1) = 2
        GOTO return
      FI

  case3: ; n > 1
      x[x3+0] = x[x3+2] + 2 ; A(0, n) = n+2
      GOTO return
    FI

  case4: ; (1) A(i, n-1) ausrechnen
    x4 = x3 + 10 ; Stackpointer f¨ur Funktionsaufruf
    x[x4+1] = x[x3+1] ; push i
    x[x4+2] = x[x3+2] - 1 ; push n-1
    x[x4+8] = 1 ; Rücksprung nach back1
    x[x4+9] = 0 ; nicht nach back2
    x3 = x3 + 10 ; Stackpointer setzen
    GOTO ackermann

back1: ; (2) A(i-1, [Ergebnis von (1)])
  x4 = x3 ; Stackpointer der aufgerufenen Funktion
  x3 = x3 - 10 ; Stackpointer der aktuellen Funktion
  x[x4+1] = x[x3+1] - 1 ; push i-1
  x[x4+2] = x[x4+0] ; push [Ergebnis von (1)]
  x[x4+8] = 0 ; nicht nach back1
  x[x4+9] = 1 ; Rücksprung nach back2
  x3 = x3 + 10 ; Stackpointer setzen
  GOTO ackermann

back2:
  x4 = x3 ; Stackpointer der aufgerufenen Funktion
  x3 = x3 - 10 ; Stackpointer der aktuellen Funktion
  x[x3+0] = x[x4+0] ; A(i, n) = A(i-1, A(i, n-1))

return: ; Bestimmung der passenden Rücksprungadresse
IF (x[x3+9] != 0) THEN
  GOTO back2
ELSE
```

```
  IF (x[x3+8] != 0) THEN
    GOTO back1
  ELSE
    GOTO main
  FI
FI

end:
```