Prof. Dr. Christoph Scholl
Dr. Tim Welschehold
Alexander Konrad
Niklas Wetzel

Freiburg, 25.11.2022

Betriebssysteme
Übungsblatt 6

Aufgabe 1 (6 Punkte)

Betrachten Sie folgendes Pico-C Programm mit Deklarations- und Anweisungsteil.

```
void main()
{
   //Deklarationsteil
   int x;
   int y;
   const int z = 2;

   //Anweisungsteil
   y = 3;
   x = 15;

   while(x >= z * y)
   {
      x = x - 3;
   }
}
```

Erstellen Sie die Einträge der Symboltabelle für den Deklarationsteil (nehmen Sie an, dass *bds*= 128) und übersetzen sie obiges Programm in eine Befehlsfolge der erweiterten ReTI. Werten Sie die Ausdrücke und Anweisungsfolgen aus, wie Sie es in der Vorlesung gelernt haben. Versehen Sie Ihr Programm mit Kommentaren, die erklären, was Sie mit den entsprechenden Programmzeilen erzielen wollen. Nehmen Sie weiter an, dass *eds*= 256 (dezimal). Gehen Sie dabei wie in Kapitel 3 der Vorlesung der Einfachheit halber davon aus, dass an die ReTI keine anderen Speicher als der Hauptspeicher angeschlossen sind und die Adressen des Hauptspeichers bei 0 beginnen. DS ist mit 0^{10} vorbelegt. Wenn mehrere Speicher angeschlossen wären und das Memory Mapping wäre wie in Kapitel 2.3 angegeben, dann müssten die Binärdarstellungen von *bds* und *eds* natürlich mit "10 . . ." beginnen.

Aufgabe 2 (6 Punkte)

In der Vorlesung wurde Pico-C unter anderem um Felder (Arrays) erweitert.

Betrachten Sie den allgemeinen Fall eines Feldes das mit $t\ ab[s_1][s_2]\dots[s_n]$ deklariert wurde. Der Zugriff auf ein Element erfolgt durch $ab[e_1]\dots[e_n]$ mit arithmetischen Ausdrücken $e_1, \dots, e_n$. Schreiben Sie ein Programm mit Befehlen der erweiterten RETI, das die e_i zum jeweiligen Wert $val(e_i)$ auswertet und die Adresse $a + \sum_{i=1}^{n} l_i \cdot val(e_i)$ des indizierten Elements berechnet und in IN1 abspeichert. Hierbei sei $l_i = \prod_{j=i+1}^{n} s_j$ wie in der Vorlesung. Die l_i ergeben sich also aus dem Programmcode und können somit zur Übersetzungszeit als Konstanten angenommen werden. Sie dürfen annehmen, dass der Typ des Feldes t in eine Speicherzelle passt, also $size(t) = 1$ gilt. Sie dürfen außerdem $code^{aa}(e_i)$ (so wie dieser Befehl in der Vorlesung definiert wurde) als abkürzende Schreibweise für die Auswertung der e_i benutzen. **Gehen Sie für Ihr Programm von den konkreten Werten $n = 3, s_1 = 4, s_2 = 2, s_3 = 3$ aus.** Versehen Sie Ihr Programm mit Kommentaren, die erklären, was Sie mit den entsprechenden Programmzeilen erzielen wollen.

Aufgabe 3 (3+6+1 Punkte)

Betrachten Sie folgendes Pico-C-Programm.

```
#include <stdlib.h> //Stellt Bibliotheksfunktionen malloc() und free() bereit

void main()
{
   struct point
   {
      int x;
      int y;
   };

   // Annahme Symboltabelleneintrag st(p1) = (var, struct point*, 8)
   struct point *p1;
   // Annahme Symboltabelleneintrag st(p3) = (var, struct point*, 9)
   struct point *p3;
   // Annahme Symboltabelleneintrag st(a) = (var, int*, 10)
   int* a;

   //Annahme Symboltabelleneintrag:
   //st(p2) = (struct, x -> (int,0), y -> (int,1), 15)
   struct point p2;

   a = &(p2.x);
   p2.x = 7;
   p2.y = 4;

   /*** MARKE 1 ***/

   //Annahme: reserviert zusammenhaengenden Bereich auf dem Heap ab Adresse 33
   p1 = (struct point *) malloc(sizeof(struct point));

   (*p1).y = *a;

   p3 = p1;
   p1 = &p2;
```

```
   /*** MARKE 2 ***/

   if((*p1).y > 5)
   {
      *a = 42;
   }
   else
   {
      *a = 1;
   };

   /*** MARKE 3 ***/

   free(p3);
};
```

a) Zählen Sie alle Speicheradressen auf, die beim Ablauf dieses Programms gelesen bzw. geschrieben werden (beachten Sie die Kommentare im Programmcode). Begründen Sie Ihre Antwort.

b) Erstellen sie einen Speicherabzug mit den relevanten Speicheradressen nach Erreichen der Marken 1, 2 und 3, indem Sie den Inhalt der Speicherzellen aus a) benennen. Undefinierte Speicherzellen können Sie entsprechend kennzeichnen. Begründen Sie Ihre Antwort.

c) Ist die letzte Zeile `free(p3);` zulässig? Falls ja, welcher Speicherbereich wird hier freigegeben? Falls nein, weshalb?

Hinweis: Nehmen Sie wie in der Vorlesung an, dass Code für die RETI erzeugt wird, die 4-Byte-Worte adressiert und nicht einzelne Bytes. Ein `int` passe in eine Speicherzelle.

Abgabe: als PDF im Übungsportal bis 02.12.2022 um 12:00