Systemnahe Programmierung: Übungszettel 2

DHBW Stuttgart
Prof. Dr. Matthias Drüppel

Aufgabe 1

Folgendes Programmfragment soll in MIPS Assemblersprache übersetzt werden:

```
int a[10] = { 5, 2, 1, 1, 2, 2, 4, 3, 9, 1};
int tmp = 0;
int i = 0;
while ((tmp <= 5) && (i < 9))
{
      tmp = (a[i] + a[i+1]) / 2;
      i = i + 1;
}
printf("Erster Mittelwert > 5 ist %d an Position %d\n", tmp, i);
```

Hinweis: Legen Sie *a* in den statischen Daten an über:

```
.align 2
a: .word 5, 2, 1, …
```

Aufgabe 2

1. Geben Sie eine Sequenz aus MIPS Instruktionen an, die testet, ob bei der Addition zweier vorzeichenloser positiver Zahlen mit `addu $s0,$s1,$s2` ein Überlauf auftritt. *Hinweis*: Die Überlaufbedingung $a + b > 2n - 1$ bei der Addition zweier vorzeichenloser n-Bit Zahlen tritt ein, wenn gilt: $b > (2n - 1) - a$. Sie brauchen in der gesamten Aufgabe immer nur die vorzeichenlose Addition/Subtraktion.
2. Wie können unter Verwendung des Codefragments aus 1. zwei vorzeichenlose 64-Bit Zahlen auf einer MIPS-32 Architektur addiert werden? Geben Sie eine entsprechende Sequenz aus MIPS Instruktionen an.

Aufgabe 3

Implementieren und testen Sie in MIPS Assembler die folgende Funktion, die das größte Element im Array `a` zurückgibt. Die Übergabe der Startadresse des Arrays soll im Register `$a0`, die Übergabe der Anzahl der Elemente im Register `$a1` erfolgen.

```
int max(int[] a, int length)
{
      int tmp = a[0];
      for (int i=1; i<length; i++)
      {
            if (a[i] > tmp)
            {
                  tmp = a[i];
            }
      }
      return tmp;
}
```