veritas iustitia libertas
Freie Universität Berlin

Rechnerorganisation
SS 2005
Übungsblatt Nr. 3

Prof. Dr.-Ing. Robert Tolksdorf
Freie Universität Berlin

Ausgabe am 20.05.2004 — Abgabe spätestens 03.06.2004, 12:00 Uhr

Bitte bei der Abgabe alle Namen/Matr.Nr. der Gruppe, NUMMER DER ÜBUNG/TEILAUFGABE und DATUM auf den Lösungsblättern **nicht vergessen!** Darauf achten, dass die Lösungen beim richtigen Tutor/der richtigen Tutorin abgegeben werden.
Zu spät abgegebene Lösungen werden nicht mehr berücksichtigt!

0. Hinweis

Alle Assemblerprogramme müssen ausführlich kommentiert werden. Die Abgabe eines Kompilats (d.h. Aufgabe in C o.ä. lösen und compilieren) wird natürlich nicht gewertet (und kann recht einfach erkannt werden). Sie müssen in der Lage sein, Ihr Programm detailliert zu erläutern. Weiterhin muss Ihr Programm beim Testen durch den Tutor funktionieren – also z.B. kann ein Testprogramm Ihre Funktion über y=ihre_funktion(x) aufrufen, dann sollte y das Ergebnis erhalten, welches die Funktion basierend auf x berechnet.

1. Aufgabe: Gauß-Summe (10 Punkte)

Schreiben Sie drei IA-32-Assembler-Routinen zur Berechnung der Gauß-Summe einer Zahl. Eine Routine, `gs_iter`, soll dabei die Gauß-Summe iterativ, die zweite, `gs_rec`, rekursiv berechnen. Die dritte schließlich, `gs_ohne`, soll ohne Iteration oder Rekursion auskommen. Vergleichen Sie für Ihre Lösungen jeweils den Aufwand. Geben Sie dazu die Verhältnisse der benötigten Assembler-Befehle der drei Lösungen in Abhängigkeit des Eingabeparameters *n* an (y=gs(*n*) soll berechnet werden, gs(*n*) = 1+2+…+*n*, *n*>0).

2. Aufgabe: Lucas-Zahlen (10 Punkte)

Schreiben Sie eine IA-32-Assembler-Routine zur Berechnung der Lucas-Zahlen (nach Edouard Lucas, 1842-1891). Diese sind rekursiv für natürliche Zahlen *n* wie folgt definiert: f(0)=2, f(1)=1, f(*n*+2)=f(*n*+1)+f(*n*). Der Aufruf der Routine `lucas(n)` soll entsprechend den Wert der Lucas-Zahl zurückliefern (lucas(10) hat z.B. den Wert 123). Programmieren Sie rekursiv unter Verwendung eines Stacks. (Ganz nette Infos dazu: http://www.mcs.surrey.ac.uk/Personal/R.Knott/Fibonacci/lucasNbs.html bzw. für die nahen Verwandten: http://www.mcs.surrey.ac.uk/Personal/R.Knott/Fibonacci/fib.html bzw. http://www.mcs.surrey.ac.uk/Personal/R.Knott/Fibonacci/fibnat.html)

3. Aufgabe: Vektoren (10 Punkte)

Schreiben Sie eine IA-32-Assembler-Routine zur Berechnung der Differenz zweier Vektoren und zur Bestimmung der Orthogonalität zweier Vektoren. Die Differenz zweier Vektoren ist ein Vektor, dessen Komponenten jeweils die Differenz der entsprechenden Komponenten der beiden Eingabevektoren ist. Vektoren sind dann orthogonal, wenn ihr inneres Produkt gleich Null ist. Die entsprechenden Funktionen `vorth(a, b, n)` bzw. `vdiff (a, b, r, n)` sollen als Eingabe Vektoren *a*, *b beliebiger* Länge und die Länge *n* haben. Als Ergebnis gibt `vorth` den Wert 0 zurück, falls die Vektoren orthogonal sind, ansonsten -1. `vdiff` hat zusätzlich den Ergebnisvektor *r* als Eingabe und schreibt direkt in diesen das Ergebnis der Subtraktion.
Vektoren seien (beispielhaft) wie folgt definiert:

```
const LEN = 5;
int vector [LEN] = {1,2,3,-4,2};
int *vector_ptr = vector;
```

Damit sollen die Funktionen wie folgt aufrufbar sein:

```
int vorth (int *v_a, int *v_b, int len);
void vdiff (int *v_a, int *v_b, int *res, int len);
```

Das folgende Beispielprogramm zum Testen Ihrer Routinen sollte als Ausgabe:

```
vorth: 0
vdiff:  6  -2  6  -2  3
```

liefern.

```
#include "stdafx.h"

const LEN = 5;
int vec_a [LEN] = {1,2,3,-4,2};
int *vec_a_ptr = vec_a;
int vec_b [LEN] = {-5, 4, -3, -2, -1};
int *vec_b_ptr = vec_b;
int ergebnis [LEN] = {0,0,0,0,0};
int *ergebnis_ptr = ergebnis;

int vorth (int *v_a, int *v_b, int len);
void vdiff (int *v_a, int *v_b, int *res, int len);

int vorth (int *v_a, int *v_b, int len) {
// hier Ihre Assembler-Routine
};

void vdiff (int *v_a, int *v_b, int *res, int len) {
// hier Ihre Assembler-Routine
};

int _tmain(int argc, _TCHAR* argv[]) {
      printf(" vorth: %d \n", vorth(vec_a_ptr, vec_b_ptr, LEN));

      vdiff(vec_a_ptr, vec_b_ptr, ergebnis_ptr, LEN);

      printf(" vdiff: ");
      for (int i=0; i<LEN; i++)
            printf(" %d ", ergebnis_ptr[i]);

      while (getchar()==EOF);
      return 0;
}
```