Praktikum zu Rechnerarchitektur

Aufgabe 1

Formulieren Sie das folgende einfache C-Programm in MIPS-Assembler. Die Funktion ruft keine andere Funktion auf, so dass sie $ra nicht speichern müssen. Außerdem benötigt sie keinen Zwischenspeicher. Daher kann sie ohne jegliche Stapelmanipulation geschrieben werden. Die Befehlsfolge zur main-Funktion ist mit einem syscall-Aufruf mit Parameter 10 (exit) zu beenden.

```
int a = 10;
int b = 20;
int c;

int f(int x, int y) {
   return (x + y);
}

void main () {
   c = f(a, b);
}
```

Aufgabe 2

Erweitern Sie Ihr MIPS-Programm aus Aufgabe 1 um eine MIPS-Assembler-Prozedur zu folgender C-Funktion.

Sie müssen die Anzahl der von Ihrer neuen MIPS-Prozedur verwendeten Register minimieren: Verwenden Sie das $s0-Register, um alle temporären Werte für die von der MIPS-Prozedur durchgeführten Berechnungen zu speichern. Sie müssen den Stack verwenden, um den Inhalt aller CPU-Register zu speichern und wiederherzustellen, die vom Aufrufer in Übereinstimmung mit den etablierten MIPS-Konventionen verwendet werden. Sie müssen auch die MIPS-Registerverwendungskonventionen befolgen, um Argumente und Rückgabewerte zu übergeben und zu empfangen.

```
int a = -10;
int b = 20;
int c;

int g(int x, int y) {
   int t;
   int result;
   if (x >= 0) {
      t = x;
   } else {
      t = -x;
   }
   t = f(t, y);
   result = x + t;
   return result;
}

void main () {
   c = g(a, b);
   c = g(c, a);
}
```

MIPS-Konventionen zur Verwendung der Register:

Name	Nummer	Verwendung
$zero	0	Enthält den Wert 0, kann nicht verändert werden.
$at	1	temporäres Assemblerregister. (Nutzung durch Assembler)
$v0	2	Funktionsergebnisse 1 und 2 auch für Zwischenergebnisse
$v1	3	
$a0	4	Argumente 1 bis 4 für den Prozeduraufruf
$a1	5	
$a2	6	
$a3	7	
$t0,…,$t7	8-15	temporäre Variablen 1-8. Können von aufgerufenen Prozeduren verändert werden.

Name	Nummer	Verwendung
$s0,…, $s7	16 … 23	langlebige Variablen 1-8. Dürfen von aufgerufenen Prozeduren nicht verändert werden.
$t8,$t9	24,25	temporäre Variablen 9 und 10. Können von aufgerufenen Prozeduren verändert werden.
$k0,k1	26,27	Kernel-Register 1 und 2. Reserviert für Betriebssystem, wird bei Unterbrechungen verwendet.
$gp	28	Global Pointer: Zeiger auf Datensegment
$sp	29	Stackpointer Zeigt auf das erste freie Element des Stacks.
$fp	30	Framepointer, Zeiger auf den Prozedurrahmen
$ra	31	Return Adresse