SS19

Versuch 4: MIPS-Instruktionssatz II

In diesem Versuch wird der Instruktionssatz eines MIPS-Prozessors weiter vertieft. Den Schwerpunkt bilden Befehle für den Zugriff auf den Speicher und das Arbeiten mit dem Stack.

Hinweis: Es sollen nur die aus der Vorlesung bekannten Befehle verwendet werden.

Aufgabe 1: Euklid'scher Algorithmus mit einfacher Testbench

Gegeben ist die Funktion ***ggt*** (s. Ende dieser Aufgabe), welche mit Hilfe des Euklid-Verfahrens den GGT zweier Zahlen berechnet.
Erstellen Sie im Simulator MARS die Datei **ggt.asm**. Erstellen Sie die Assemblerfunktion ***ggt***.

Die Parameter a und b sollen mit Hilfe des Stacks an die Funktion übergeben werden. Das Ergebnis wird von der Funktion ***ggt*** ebenfalls auf dem Stack abgelegt. Die lokale Variable r wird ebenfalls auf dem Stack angelegt. Sowohl für den Rückgabewert als auch für die lokale Variable werden gesonderte Speicherstellen verwendet, so dass die beiden Parameter a und b nicht überschrieben werden.

Innerhalb des Unterprogramms dürfen die Register $t0 bis $t9 zum Speichern der übergebenen Parameter und für Zwischenergebnisse verwendet werden.

Ergänzen ihren Code um eine "Testbench“. Diese soll die beiden Zahlen 44 und 12 auf dem Stack ablegen und das Ergebnis nach Aufruf der Funktion ***ggt*** vom Stack abholen und im Register $at ablegen. Die Stackbelegung vor dem Unterprogrammaufruf ist wie folgt festgelegt:

Stackbelegung Hauptprogramm	
sp+8	1. Operand (44)
sp+4	2. Operand (12)
sp	Rückgabewert

Die Testbench könnten sie beispielsweise wie folgt implementieren:

```
testbench:
        addi $sp,$sp,-12 # Platz auf Stack schaffen
        ori  $t0,$0,44   # 44 nach $t0
        sw   $t0,8($sp)  # und auf Stack ablegen
        ori  $t0,$0,12   # 12 nach $t0
        sw   $t0,4($sp)  # und auf Stack ablegen

        jal  ggt         # Sprung in den zu testenden Code
        nop              # dieser Befehl wird noch ausgeführt! (wg. Pipelining)

        lw   $at,0($sp)  # Rückgabewert vom Stack holen
        addi $sp,$sp,12  # Stackpointer wieder auf Anfangswert (Platz freigeben)

        ori  $v0,$0,10   # Funktion: Programm beenden
        syscall          # Simulatorfunktion ausführen
```

Skizzieren Sie vor der Codeeingabe die Stackbelegung nach dem Reservieren der lokalen Variable im Unterprogramm.

Stackbelegung Unterprogramm	
sp+12	
sp+8	
sp+4	
sp	

Testen Sie Ihren Code und führen Sie das lauffähige Programm Ihrem Versuchsbetreuer vor.

Tipp: Sie können – falls es Ihnen sinnvoll erscheint – als Zwischenschritt zunächst mit einer Parameterübergabe in den Registern $a0 und $a1 arbeiten (Rückgabewert in $v0). So können Sie die Kernfunktion Ihres GGT-Programms zunächst unabhängig von der Übergabe per Stack testen.

```
int ggt (int a, int b) {
    int r;
    if (a==0) {
        r = b;
    } else {
        while (b != 0) {
            if (a > b) {
                a = a – b;
            } else {
                b = b – a;
            }
        }
        r = a;
    }
    return r;
}
```

Aufgabe 2: Erweiterte Testbench

Erweitern Sie die Testbench aus der vorangegangenen Aufgabe derart, dass sie Testparameter und die zugehörigen korrekten Ergebnisse aus dem Speicher lesen kann.

Testdaten stehen in der Datei **testdata_ggt.asm** zur Verfügung. Die Datei können Sie mit der Assembler-Direktive `.include "testdata_ggt.asm"` in die Datei *testbench_ggt.asm* einbinden.

Je nach Ergebnis der Tests soll eine Erfolgsmeldung bzw. eine Fehlermeldung ausgegeben werden. Wie die Ausgabe von Texten in MARS erfolgt, zeigt das folgende Codefragment:

```
        la      $a0,my_message
        ori     $v0,$0,4
        syscall


        .data     # auf Datensegment umschalten

my_message:
        .asciiz  "Hallo"
```

Testen Sie die Testbench! Modifizieren Sie die Werte in der Datei testdata_ggt.asm, indem Sie an mindestens einer Stelle ein falsches Ergebnis eintragen. Wird nun eine Fehlermeldung von Testbench ausgegeben?

Als Hilfestellung erhalten Sie die Kommentare aus einer funktionierenden Testbench. in einer Textdatei (Vorschlag_Aufgabe2.txt). Sie können diese als Vorlage für Ihren Code nutzen. Sie können selbstverständlich auch versuchen, die Testbench ohne Berücksichtigung der Kommentare zu erstellen. Sie muss nur am Ende des Versuchs funktionieren. Der Weg zu diesem Ziel ist Ihnen freigestellt ;-)