Martin-Luther-Universität Halle-Wittenberg
Fachbereich Mathematik und Informatik
Lehrstuhl für Technische Informatik
Prof. Dr. P. Molitor, S. Wefel

Telefon: ☎ 03 45 / 5 52 47 25
E-Mail: wefel@informatik.uni-halle.de

Halle, 13. November 2003

Rechnerarchitektur und Rechnerorganisation (WS 2003/04)

Übungsserie 5

Aufgabe 5.1. (10 Punkte)

Erweitern/Ändern Sie das Programm zur Zahlenumwandlung aus der Serie 4, um folgende Funktionalitäten zu erreichen:

Das Programm soll innerhalb einer Schleife jeweils eine Zeichenkette (!!!) mit maximal 10 Zeichen vom Nutzer erfragen. Die Eingabe ist mittels der Eingabetaste abzuschließen.

Die Zeichenkette ist nun zu interpretieren, wobei folgende Eingaben zulässig sind:

- Zahlen (ohne Vorzeichen) mit 1 bis 10 Stellen
- oder Operationssysmbole `+`, `-`, `*` oder `/`
- oder das Zeichen 'q'

Eine gemischte Eingabe von Operationssymbol, Zahl oder Zeichen ist bei einer Eingabe nicht erlaubt.

Anschließend soll das Programm entsprechend der Eingabe folgende Aktion ausführen:

- Wird eine Zahl erkannt, so ist diese auf dem Stack abzulegen.
- Wird ein Operationssysmbol erkannt, so ist die entsprechende Operation auszuführen. Da es sich hierbei um binäre Operationen handelt, werden zwei Operanden benötigt. Diese beiden Operanden sind vom Stack zu holen. Das Ergebnis ist auf dem Stack abzulegen.
- Wird das Zeichen 'q' erkannt, so ist der letzte Wert vom Stack zu holen, auf der Konsole auszugeben und anschließend die Schleife zu verlassen, woraufhin auch das Programm beendet werden soll.
- Bei Fehleingabe ist eine entsprechende Fehlermeldung auszugeben, aber das Programm wird nicht beendet!

Anschließend ist mit dem nächsten Schleifendurchlauf fortzusetzen.

Überlegen Sie sich auch, wie sie einen Stack-Underflow erkennen und mittels entsprechender Fehlermeldung darauf reagieren können.

Aufgabe 5.2. (8 Punkte)

In der Vorlesung wurde die Funktionsweise der Pipeline zur Performancesteigerung anhand der MIPS-Prozessorpipeline vorgestellt. Ein Problem der Pipelinearchitektur sind `Branch`-Befehle, bei denen die Ausführung der folgenden Befehle von der Entscheidung der Verzweigung abhängt. Man spricht von sogenannten Control-Hazards.

Eine sehr ineffektive Lösung wäre das Einfügen von `NOP`-Befehlen unmittelbar nach dem `Branch`-Befehl, so dass bei ausgeführten Sprung die nachfolgenden Befehle, die sich ja bereits in der Pipeline befinden, keine Wirkung mehr haben, also weder Register noch Speicherinhalte verändern. Eine andere, ebenfalls ineffektive Lösung wäre das Anhalten der nachfolgenden Pipelinestufen nach einem dekodierten `Branch`-Befehl, bis die Durchführung des Verzweigung entschieden wurde.

Werden jedoch derartige Lösungen nicht in Betracht gezogen, so muß durch eine geeignete Logik dafür gesorgt werden, dass die dem `Branch` folgenden Befehle im Falle der Verzweigung keine Wirkung haben.

- Wie könnte eine derartige Logik aussehen und wo müßte Sie im Prozessor integriert werden? Begründen Sie dies möglichst ausführlich und verständlich?

Hinweis: Verwenden Sie zur Lösung die Skizze “Hardware mit Kontrolllogik” aus der Vorlesung MIPS-Pipelining(1), “Beschleunigung durch Pipelining”. Sollten Sie zur Untermauerung ihrer Erklärung eine Grafik benötigen, so verwenden Sie bitte diese Folie als Grundlage, zeichnen Sie Ihre Erweiterungen ein und senden Sie diese im GIF-Format ein.