Rheinische Friedrich-Wilhelms Universität Bonn
Institut für Informatik VI, Technische Informatik

Prof. Dr. Joachim K. Anlauf
Oguzhan Sezenlik

6. Übung für die Vorlesung

Rechnerorganisation

Sommersemester 2019

Abgabe: Donnerstag, 16.5.2019; Schicken Sie bitte den Quellcode (als lauffähige `.asm` Quelldateien) für die Programmieraufgabe zusätzlich per E-Mail an Ihren Tutor:
Alexandra Chebotareva: `s6alcheb@uni-bonn.de`
Michel Fischer: `michel-fischer@hotmail.de`

Aufgabe 1. *Fibonacci* 12 P.

1. Programmieren Sie eine **rekursive** Berechnung der Fibonacci-Zahlen in MIPS Assembler. Lesen Sie hierzu eine Zahl N von der Konsole ein, berechnen Sie $Fib(N)$, und geben Sie N und $Fib(N)$ auf der Konsole aus. Die Fibonacci-Zahlen sind wie folgt definiert:
   $Fib(N) = Fib(N-1) + Fib(N-2)$ für $n > 2$, $Fib(1) = Fib(2) = 1$.

2. Stellen Sie für $N = 4$ den Stackinhalt nach jedem Prozeduraufruf vollständig dar und kennzeichnen Sie die einzelnen Procedure-Frames.

Aufgabe 2. *ALU Overflow Detection* 8 P.

Gegeben ist die aus der Vorlesung bekannte 32-Bit ALU (Abbildung 1). Der Aufbau der jeweiligen 1-Bit ALU ist in Abbildung 2 gegeben.
Die Darstellung für die Overflow-Detection ist in Abbildung 2 nur schematisch gegeben. Entwerfen Sie ein Schaltnetz zur Überlauferkennung und füllen Sie die Black-Box mit Leben! Berücksichtigen Sie dabei die Operationen `add, addu, sub, subu`! Was gilt für die Instruktionen `addi, addiu, slt, sltu`? Welche weiteren Inputsignale benötigt die Overflow Detection? Verwenden Sie zur Implementierung des Schaltnetzes so wenig Inputsignale wie möglich. Geben Sie ein möglichst optimales Schaltnetz an.
(*Hinweis: Stellen Sie eine Wertetabelle auf und minimieren Sie die DNF mit einem Verfahren Ihrer Wahl!*)

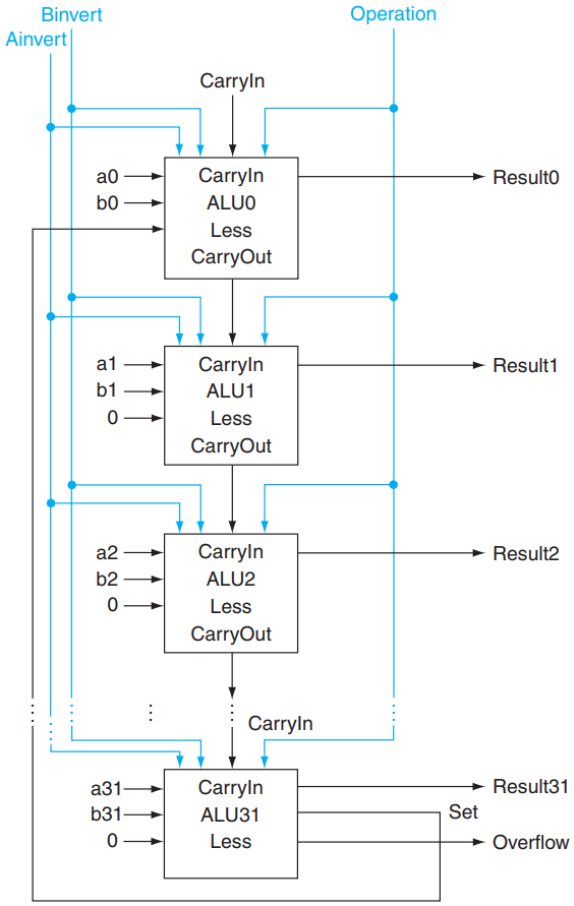

Abbildung 1: 32-Bit ALU

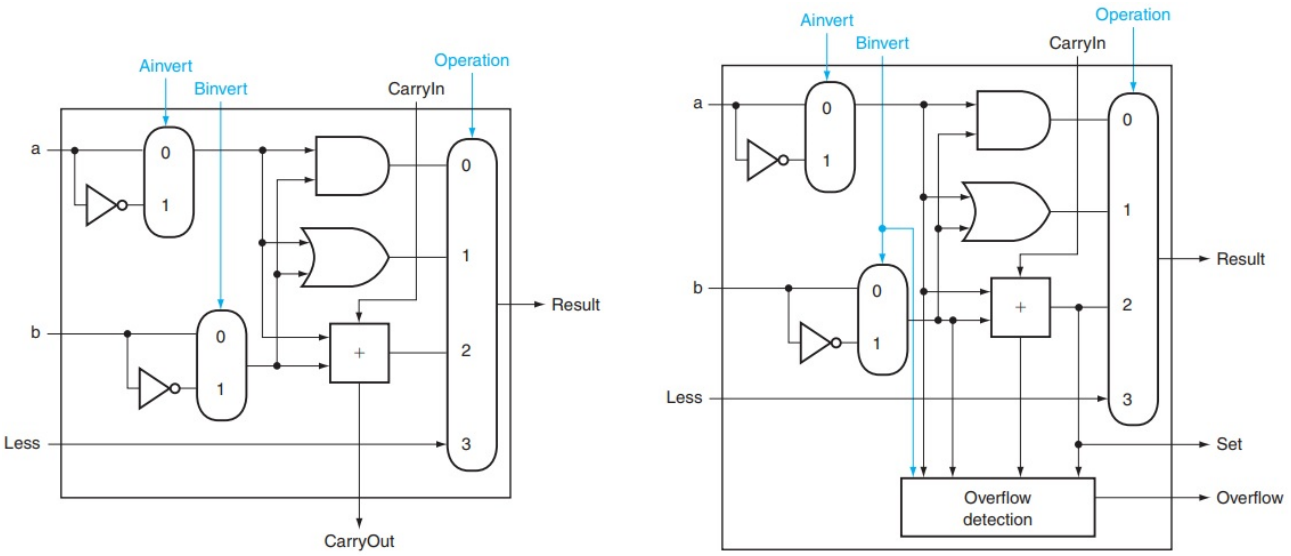

Abbildung 2: 1-Bit ALU