

# Rechnerarchitektur Serie 3

Dominik Bodenmann 08-103-053

Orlando Signer 12-119-715

8. April 2014

## 1 Theorie-Teil

### 1.1 Aufgabe 1

	$t$	$Freq$	$CPI$	$Freq * CPI$	$CPI$	$Freq * CPI$	$CPI$	$Freq * CPI$
<i>ALU</i>	$5nsec$	25%	2	0.5	2	0.5	<b>1</b>	0.25
<i>LOAD</i>	$10nsec$	25%	4	1.0	<b>6</b>	1.5	4	1.0
<i>STORE</i>	$7.5nsec$	25%	3	0.75	3	0.75	1	0.75
<i>Branch</i>	$7.5nsec$	25%	3	0.75	3	0.75	1	0.75
				3.0		3.5		2.75

Eine Maschine, die für die LOAD Instruktion 6 Taktzyklen braucht, ist also  $3.5/3.0 - 1 = 16.7\%$  langsamer.

Eine CPU, bei der die ALU doppelt so schnell arbeitet, ist also  $2.75/3.0 - 1 = 8.3\%$  schneller.

### 1.2 Aufgabe 2

1. Darauf kann die Rücksprungadresse für die Vortsetzung der Programmbearbeitung gespeichert werden.
2. Darauf können die Aufrufparameter gelegt werden, damit sie von der Subroutine gelesen werden können.

### 1.3 Aufgabe 3

Damit der Overflow behandelt werden kann:

<i>VorzeichenA</i>	<i>VorzeichenB</i>	<i>Binvert</i>	<i>Carryout</i>	<i>Vorz.Result.</i>	<i>Korr.Vorz.Res.</i>	<i>Overflow</i>
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	0	1	1	0
0	1	1	1	0	0	0
1	0	0	0	1	1	0
1	0	1	1	0	0	0
1	1	0	1	0	1	1
1	1	1	1	1	1	0

Die ersten 5 Argumente werden für die Overflowdetection gebraucht. Ist das Binvert Bit nicht gleich dem Carry out Bit, so gibt es einen Overflow. (Bei ALU31, da da die Vorzeichen abgespeichert sind)

### 1.4 Aufgabe 4

- Beim slt-Befehl wird a-b gerechnet. Falls a-b  $\geq 0$  erhält man im Vorzeichenbit (ALU31) eine 1.
- Die ALU unterstützt diesen Befehl, indem sie das Set des ALU31 Bits mit dem Less des ALU0 Bits verbindet. Alle anderen Less sind 0;

### 1.5 Aufgabe 5

### 1.6 Aufgabe 6

### 1.7 Aufgabe 7