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	$Freq * CPI$
ALU	$5nsec$	25%	2	0.5	2	0.5	**1**	0.25
$LOAD$	$10nsec$	25%	4	1.0	**6**	1.5	4	1.0
$STORE$	$7.5nsec$	25%	3	0.75	3	0.75	1	0.75
$Branch$	$7.5nsec$	25%	3	0.75	3	0.75	1	0.75
				3.0		3.5		2.75height

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

2. 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:

$VorzeichenA$	$VorzeichenB$	$Binvert$	$Carryout$	$Vorz.Result.$	$Korr.Vorz.Res.$	$Overflow$
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 B invert 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 ¡ 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

Listing 1: MIPS push und pop

```
1 # Push und pop auf dem Stack. $sp bezeichnet den Stackpointer.
2 push:
3 subi $$sp $sp 4
4 sw  $t0 0($sp)
5
6 pop:
7 lw $t0 0($sp)
8 addi $sp $sp 4
```

1.6 Aufgabe 6

1.7 Aufgabe 7