Rechnerarchitektur Serie 4

Dominik Bodenmann 08-103-053
Orlando Signer 12-119-715

29. April 2014

1 Theorie-Teil

1.1 Aufgabe 1

Da bei beq und bne die Positionen , an die gesprungen werden soll, relativ zur aktuellen Positin angegeben wird, braucht es negative Werte.
Ebenfalls bei den Load und Store Befehlen kann der Offset zur Basisadresse negativ sein.

1.2 Aufgabe 2

Eingabe	*Ausgabe*
0xFE & 0xEF	0xEE
0xFE && 0xEF	1
0xFE $\|\|0xEF$	0xFF
0xFE $\|\|\|0xEF$	1
$\tilde{0}$xFE	-0xFF
!0xFE	0

1.3 Aufgabe 3

Mittels Schnittstellen, an denen die Hardware entsprechende Informationen zur Verfügung stellt.
Als Beispiel: Die Switches des Boards, die, je nachdem wie sie gestellt sind, ein Feld entweder mit 0 oder 1 füllen.

1.4 Aufgabe 4

1. Singlecycle-Implementation Das Resultat einfach invertieren, falls die ALU beim Vergleich 0 zurück gibt. (NOT-Gatter am 0-Ausgang der ALU)

2. Multicycle-Implementation Analog Singlecycle-Implementation.

1.5 Aufgabe 5

Um die Daten, die für den nächsten Rechenschritt gebraucht werden, zwischenzuspeichern, da die nächste Berechnungsstufe möglicherweise noch nicht zur Verfügung steht.

1.6 Aufgabe 6

Structural Hazard: Tritt auf, falls während einem Clock-Cycle die gleiche Resource von zwei verschiedenen Instruktionen benötigt wird. Z.B. wenn eine Instruktion im 4. Schritt etwas vom Memory liest, gleichzeitig aber eine andere Instruktion in der Pipeline ihre Instruktion vom Memory liest. Dies kann umgangen werden, indem es separate Memories für Instruktionen und Daten gibt.
Data Hazard: Tritt auf, wenn eine Instruktion noch nicht ausgeführt werden kann, da sie vom Resultat einer anderen Instruktion abhängt. Z.B. eine Instruktion berechnet einen Wert und schreibt ihn in Register $1. Benötigt die nächste Instruktion den Wert von $1, so muss sie in der Pipeline warten, bis die erste Instruktion den Wert ins Register geschrieben hat. Mittels Forwarding kann dieses Problem umgangen werden. Dabei dient der berechnete Wert von der ALU direkt als Input für die ALU. Mittels Mux kann dann vor der ALU entschieden werden, welcher Wert übernommen wird.
Control Hazard: Tritt bei Instruktionen auf, die den Befehlszäher verändern (PC != PC + 4), z.B. bei Branching oder Jumps. Dabei sind bei nach einem Branch/Jump eventuell falsche Instruktionen in der Pipeline, die geflusht werden müssen. Deshalb ist es wichtig, die Zieladresse so früh wie möglich zu berechnen, damit nicht zu viele unnötige Instruktionen in der Pipeline geflusht werden.

1.7 Aufgabe 7

1.8 Aufgabe 8