olaf.neugebauer [☺] tu-dortmund.de
bjoern.boenninghoff [☺] tu-dortmund.de

Übung zur Vorlesung
Rechnerarchitektur
Sommersemester 2015

Übungsblatt 6

(5 Punkte)

Besprechung ab Montag, 18. Mai 2015

6.1 Sprungvorhersage (2 Punkte)

Die Vergrößerung eines Sprungvorhersage-Puffers reduziert die Wahrscheinlichkeit, dass zwei Sprung-Befehle (mit gleichen niederwertigen Adressbits) dasselbe Prädiktionsbit nutzen. Im Allgemeinen ist ein Prädiktor, der ausschließlich Sprungvorhersagen bezüglich einer einzigen Instruktion macht, genauer als derselbe Prädiktor, der von mehreren Sprungbefehlen geteilt wird.

a) Nehmen Sie an, dass sich zwei Sprungbefehle in einer Schleife befinden und abwechselnd ausgeführt werden. Erstellen Sie eine Sequenz von *branch taken*- und *branch not taken*-Aktionen (für die beiden Sprünge), die verdeutlicht, dass das gemeinsame Nutzen eines 1-bit-Prädiktors die Vorhersage-Fehlerrate erhöht.

b) Listen Sie diesmal eine Abfolge von *branch taken*- und *branch not taken*-Aktionen auf, die das Gegenteil zum Aufgabenteil a) darstellt, nämlich dass die gemeinsame Nutzung des Prädiktors die Fehlerrate verringert.

6.2 Mehrstufige Prädiktoren (3 Punkte)

Betrachten Sie das Verhalten eines mehrstufigen (1,2) Sprungprädiktors anhand des folgenden Codefragmentes, das an Adresse `0xA0000000` im Speicher liegt und zwei voneinander abhängige Sprünge enthält. Nehmen Sie an, dass jede der unten aufgeführten Instruktionen als 32-Bit MIPS-Instruktion codiert ist und dass der Adressraum 2^{32} Bytes umfasst.

```
        li   $s0, 0
loop:   div  $s0, 2
        mfhi $s3
        bne  $s3, $zero, else_1
then_1: addi $s1, $s1, 4
else_1: mul  $s1, $s1, 2
        bne  $s3, $zero, else_2
then_2: addi $s2, $s2, 1
else_2: addi $s0, $s0, 1
        bne  $s0, 5, loop
```

a) Listen Sie alle Zustandsänderungen und Vorhersagen des Sprungprädiktors während der Ausführung des Codefragmentes in chronologisch korrekter Reihenfolge auf. Gehen Sie dazu wie folgt vor:

- Annotieren Sie an jeder Instruktion die Speicheradresse, an der sie steht.
- Stellen Sie dann den Ausgangszustand des Prädiktors mit der Vorlage von der Übungswebseite dar. Nehmen Sie hierfür an, dass die einzelnen 2-Bit Prädiktoren mit dem Wert $(1)_{10}$ (nicht $(11)_2$) initialisiert sind und jeweils 4 Einträge haben und dass `$s1` und `$s2` mit 0 initialisiert sind. Der in der Branch History abgelegte Wert beim Betreten des Codefragments sei 0.
- Gehen Sie das Fragment Schritt für Schritt durch und geben Sie bei jedem ausgeführtem Sprung die getroffene Vorhersage und die Änderungen am Prädiktorzustand an.

b) Welche Vorteile bietet die Kopplung des Prädiktors aus a) mit einem Branch-Target-Buffer bzw. einem Branch-Folding?

c) Um wieviel Prozent steigt der Speicherbedarf des angegebenen Prädiktors, wenn ein Branch-Target-Buffer (*BTB*) bzw. ein Branch-Folding (*BF*) (für eine einzelne Instruktion) integriert wird? Nehmen Sie hierzu an, dass der Prädiktor auch schon in der Konfiguration ohne BTB und BF für jede Zelle einen Tag speichert, um gemeinsame Nutzung einer Prädiktor-Zelle durch mehrere Sprungbefehle zu verhindern.