TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Technische Informatik, Professur für Prozessorentwurf

Rechnerarchitektur: Übungssatz 13

Aufgabe 13.1

Nachfolgender Code soll auf einer fünfstufigen RISC-V Pipeline ausgeführt werden (IF, ID, EX, ME, WB). Nehmen Sie an, dass der Writeback zu Beginn eines Taktzyklus ausgeführt wird.

```
sub  x12, x11, x28          ; I1
add  x15, x14, x12          ; I2
subi x13, x11, 1            ; I3
add  x14, x11, x13          ; I4
addi x14, x0, 2             ; I5
```

(a) Wieviele Zyklen müssen zwischen zwei Befehlen `Ia` und `Ib` liegen, wenn `Ib` vom Ergebnis des Writeback von `Ia` abhängt?

(b) Wie ändert sich dieses Ergebnis, wenn die Pipeline nur 4 Stufen hat, also Speicher- und ALU-Operation gleichzeitig ausgeführt werden.

(c) Benennen Sie die Pipelinekonflikte im Code. Nehmen Sie an, die Pipeline besitze weder Forwarding noch Hazard Detection und fügen Sie, wenn nötig, `NOP` Befehle ein.

(d) Ordnen Sie die Befehle neu an um, wenn möglich, die Anzahl `NOP` Befehle zu verringern! (Hinweis: Bestimmen Sie zuerst unabhängige Berechnungen.)

(e) Nehmen Sie an, dass Ihnen noch ein zusätzliches Register x30 zur Verfügung steht. Ist es möglich, die Anzahl der `NOP` Befehle weiter zu reduzieren, indem Register umbenannt werden? Wenn nicht, eklären Sie warum, wenn doch, zeigen Sie wie!

Aufgabe 13.2

Nehmen Sie für diese Aufgabe an, dass jedes Prozessorwort ein 64-bit Integer ist und der Speicher byteweise adressiert wird. Der nachfolgender Code zum verarbeiten von Matrizen ist in C geschrieben. Dabei stehen Elemente einer *Zeile* nacheinander zusammenhängend im Speicher.

Zum Beispiel für eine Matrix `M` mit Zeilenindex `r` und Spaltenindex `c`:

M[r][c]	M[r][c+1]	...	M[r+1][c]	M[r+1][c+1]	...

Code:

```
for (I=0; I<8, I++0)
  for (J=0; J<8000; J++)
    A[I][J]=B[I][0] + C[J][I]
```

(a) Wieviele 64-bit Integer können in einem 16 Byte Cacheblock gespeichert werden?

(b) Was ist zeitliche Lokalität? Was ist örtliche Lokalität?

(c) Welche Variablenreferenzen zeigen zeitliche Lokalität?
(*"Variablenreferenz"bezieht sich sowohl auf die Verwendung einer Variablen selbst, als auch auf Zugriffe innerhalb eines Arrays.*)

(d) Welche Variablenreferenzen zeigen örtliche Lokalität?

(e) Wieviele 16-byte Cacheblöcke werden benötigt um alle 64-bit Matrixelemente zu referenzieren? Nehmen Sie an, dass es mehr als ein Element pro Zeile in Matrix `B` gibt.

(f) Nehmen Sie nun an, der Code läge in einer Sprache vor, welche die Elemente einer *Spalte* nachfolgend im Speicher ablegt (z.B. Matlab). Wie beeinflusst das die Ergebnisse der Teile c–e?

(g) Beschreiben Sie den Unterschied zwischen den Schreib-Hit Rückschreibstrategien *Write-Through* und *Write-Back*. Für welche der beiden Varianten oben ist diese Wahl wichtig, und warum? Welche Vor- und Nachteile haben diese Strategien allgemein?

Aufgabe 13.3

Caches sind wichtig um schnelle Zugriffe in der Speicherhierarchie eines Prozessors zu ermöglichen. Im Folgenden sind 64-bit Speicherreferenzen in Form von Wortadressen gegeben. Nehmen Sie an, dass auf diese der Reihe nach zugegriffen wird.

```
0x03, 0xb4, 0x2b, 0x02, 0xbf, 0x58, 0xbe, 0xb5
```

Nennen sie die binäre Wortadresse, den Tag, und den Index in einen direct-mapped Cache mit folgenden Konfigurationen für jede Referenz. Nennen Sie außerdem welche Referenzen ein Hit und welche ein Miss sind unter der Bedingung, dass der Cache zu Beginn leer ist.

(a) C1: 16 ein Wort breite Blöcke

(b) C2: zwei Wort breite Blöcke und einer Gesamtgröße von acht Blöcken

Aufgabe 13.4

Ein Cache wird nach der Menge der Daten, die er halten kann, benannt (ein 4 KiB Cache kann 4 KiB Daten speichern); Caches brauchen jedoch auch SRAM Zellen um Metadaten wie Tags und Valid bits zu speichern. In dieser Aufgaben sollen Sie untersuchen wie die Konfiguration eines Caches seinen benötigten Bedarf and SRAM und seine performance beeinflussen. Nehmen Sie für alle Teilaufgaben an, dass der Cache byteweise adressierbar ist und dass Adressen und Worte 64-Bit groß sind.

(a) Berechnen Sie die Gesamtanzahl an Bits die benötigt werden um einen 32 KiB cache mit Zwei-Wort-Blöcken zu realisieren.

(b) Berechnen Sie die Gesamtmenge an Bits die benötigt werden um einen 64 KiB Cache mit 16-Wort-Blöcken zu implementieren. Um wieviel Prozent größer ist der Cache als der 32 KiB Cache aus der vorherigen Teilaufgabe?

(c) Erklären Sie warum der 64 KiB Cache, obwohl größer, langsamer sein könnte als der erste Cache.

(d) Generieren Sie eine Reihe von Lesezugriffen, die auf einem 32 KiB 2-fach satzassoziativen Cache eine geringere Miss rate haben als auf dem Cache der erste Teilaufgabe.

Aufgabe 13.5

(**Zusatzaufgabe**) Der folgende Code soll auf einer fünfstufigen RISC-V Pipeline ausgeführt werden (IF, ID, EX, ME, WB). Nehmen Sie an dass der Writeback zu Beginn eines Taktes ausgeführt wird.

```
add x15, x12, x11
ld  x13, 4(x15)
ld  x12, 0(x2)
or  x13, x15, x13
sd  x13, 0(x15)
```

(a) Fügen sie `NOP` Befehle ein um die korrekte Ausführung zu ermöglichen wenn es kein Forwarding und keine Konflikterkennung in der Pipeline gibt.

(b) Ändern oder ordnen Sie den Code jetzt so um, dass die Anzahl der `NOP` Befehle minimiert wird. Register `x17` darf für Zwischenwerte benutzt werden.