8. Aufgabenblatt

Abgabe 18.12.15

Allgemeine Hinweise:

- Bitte geben Sie zu jeder Aufgabe entweder die Beantwortung oder Testläufe auf Papier ab.
- Quellcode geben Sie bitte unkomprimiert und kommentiert im KVV ab.
- Beantworten Sie alle Aufgaben mit Ihren eigenen Worten.

Problem 1: Dynamische Sprungvorhersage

a) Wie funktioniert ein 1-Bit-Predictor bei der dynamischen Sprungvorhersage?

b) Was ist beim 2-Bit-Predictor anders? Für welche Fälle ist er besser geeignet?

c) Wie häufig liegen bei den folgenden Fällen der 1-Bit-Predictor und der 2-Bit-Predictor (Hysteresis Scheme) statistisch betrachtet richtig bei ihrer Vorhersage? Gehen Sie von einem großen Wert von n aus, sowie für eine Wahrscheinlichkeit von 50%, dass p wahr ist. Beide Prädiktoren starten dabei im Zustand „Predict Strongly Taken“.

Geben Sie zusätzlich äquivalenten Assembler (Pseudo-)Code an.

1)
```
for i=1:n {
  ...
  }
```

2)
```
for i=1:n {
    for j=1:n {
    ...
    }
  }
```

3)
```
for i=1:n {
    if p {
    ...
    }
  }
```

Problem 2: Pipelining

Gegeben ist folgende Befehlsfolge (die Großbuchstaben dienen nur der Identifikation der Zeilen):

```
A  : a = a + 1;
B  : a = a + 1;
C  : a = a + 1;
IF : if( x == 0 ) {
E  :   a = a / 2;
F  :   a = a + 1;
     }
G  : a = a + 1;
H  : a = a + 1;
```

Die Befehlsfolge werde von einem Rechner abgearbeitet, der über die gleiche 5-Stufige Pipeline wie in der letzten Übung verfügt. Gehen Sie davon aus, dass in der if-Abfrage die Bedingung (x == 0) false liefert, die Sprungvorhersage aber falsch liegt, also (x == 0) = true vorhersagt.

a) Bestimmen Sie die Anzahl der Takte, die der Rechner für die vollständige Abarbeitung der Folge benötigt. Gehen Sie dabei davon aus, dass ein Löschen der Pipeline zusätzliche 10 Takte kostet.

b) Schreiben Sie den Code so um, dass er für die bedingte Anweisung Predicated Instructions (s. Folien ab 3.115) verwendet. Bestimmen Sie wieder die Anzahl der Takte, die der Rechner für die vollständige Abarbeitung benötigt.

Hinweis: Für beide Teilaufgaben ist eine Visualisierung der Pipelinestufen notwendig, die den Pipelinezustand bei jedem Befehl erläutert!

Problem 3: BubbleSort

Implementieren Sie den BubbleSort-Algorithmus in x86 Assembler.

Hinweis: Für diese Aufgabe sollten Sie sich mit den verschiedenen Speicherzugriffsmöglichkeiten/Adressierungsmöglichkeiten des x86 auseinandersetzen.