Übungsblatt 9

Ü 9.1 Caches

Gegeben sei folgendes Fragment eines C-Programms, welches auf drei Arrays operiert.

```
#define N 2048

int arrA[N];
int arrB[N];
int arrX[N];

int i, s;

for (i=0; i!=N; i++) {
  arrA[i]=i;
  arrB[i]=N-i;
}

s = 0;

for (i=N-1; i>=0; i--) {
  s = s + arrA[i] - arrB[i];
  arrX[i] = s;
}
```

Nehmen Sie an, dass die Arrays beginnend mit arrA lückenlos ab Adresse 0x10010000 im Speicher abgelegt sind. Der Datentyp int belege 32 Bit. Die beiden Variablen i und s werden vom Compiler in Registern abgelegt, d.h. deren Verwendung verursacht keinen Speicherzugriff. Das Fragment wird auf einem System mit einem 4 KiB großen 2-fach satz-assoziativen write-back Datencache mit LRU-Ersetzungsstrategie ausgeführt. Die Größe eines Cache-Blocks beträgt 32 Byte. Gehen Sie davon aus, dass der Cache zu Beginn nur ungültige Einträge enthält.

a) Bestimmen Sie die Hitrate des Caches für die Ausführung des gesamten Fragments, und beurteilen Sie die erzielte Performance.

b) Im Moodle-Kurs finden Sie ein entsprechendes MIPS-Assembler-Programm des obigen Fragmentes. Verwenden Sie den MARS Data Cache Simulator, um Ihr Resultat aus a) zu verifizieren.

Ü 9.2 Caches

Gegeben seien das Code-Fragment und der Datencache aus Aufgabe 9.1.

a) Kann ein Compiler mit Hilfe von Array-Padding eine höhere Hitrate ermöglichen? Wenn ja, wie hoch wäre näherungsweise die verbesserte Hitrate für das gegebene Fragment?

b) Wie würde sich die Hitrate des Caches aus Aufgabe 9.1 (ohne Array-Padding) verändern, wenn der Datencache 4-fach satz-assoziativ organisiert wäre? Bestimmen Sie die Hitrate näherungsweise.

c) Verwenden Sie den MARS Data Cache Simulator, um Ihre Resultate zu verifizieren. Geben Sie die vom Simulator angezeigte Anzahl der Cache-Hits und die resultierende Hitrate an. Begründen Sie gegebenfalls den Unterschied zu Ihren Lösungen von a) und b).

Ü 9.3 Caches

Gegeben sei folgender Pseudocode:

```
int array[100, 100000];
for each element array[i][j] {
      array[i][j] = array[i][j] * 2;
}
```

Schreiben Sie zwei Java-Programme, die diesen Pseudocode implementieren: eines soll das Array zeilenweise durchlaufen, das andere spaltenweise. Die benötigte Ausführungszeit für den Schleifendurchlauf soll ausgegeben werden. Zum Messen der Ausführungszeit kann `java.lang.System.currentTimeMillis()` verwendet werden. Führen Sie beide Programme mehrmals hintereinander aus und vergleichen Sie die minimalen Ausführungszeiten. Was sagt Ihnen die unterschiedliche Cache-Performance über die Anordnung von Java-Arrays im Speicher?

Hinweis: Verringern Sie die Zeilenanzahl (1. Dimension) des Arrays, falls Ihre virtuelle Java-Maschine (JVM) nicht genügend Speicher zur Verfügung hat. Oder vergrößern Sie alternativ die maximale Speichergröße der JVM (z.B. mit java –Xmx128m*).*